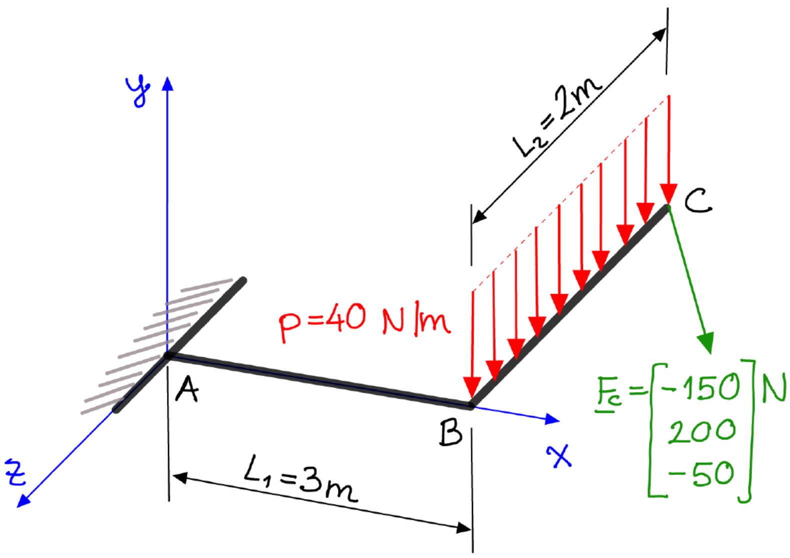
Megjegyzés:

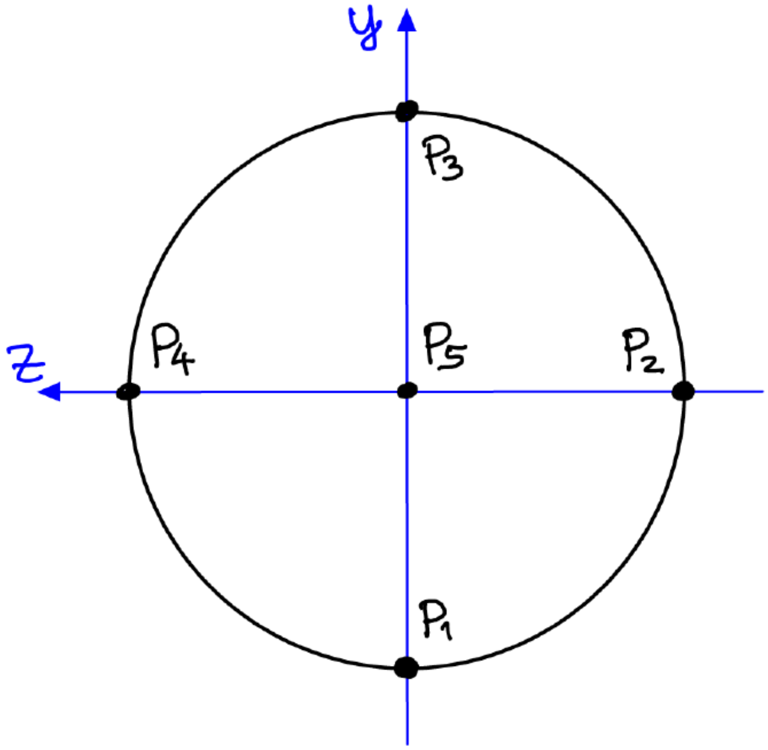
Nyomtatásban, elektronikus kidolgozásban a mérnöki konszenzus szerint a **mátrixokat álló, vastag nagybetűvel**, míg a **vektorokat álló, vastag betűvel** jelöljük (a megegyezés szerint inkább kisbetűvel, ám ez számos helyen eltér) a kézi kidolgozásban ismert aláhúzás helyett, továbbá minden ismeretlent/paramétert dőlt betűvel. A geometriai pontok, valamint a mértékegységek jelölése álló betűvel történik. Ha igazán igényesek szeretnénk lenni, akkor figyelünk arra, hogy ha az elnevezés tartalmaz alsó indexben szöveget, akkor az álló betűvel legyen írva (pl: $U_{input} \rightarrow U_{\mathrm{input}}$). A kidolgozások során ezek szerint fogunk eljárni.

1 Példa 1.25

Egy törtvonalú tartó terheléseit és méreteit mutatja az alábbi ábra. A tartó keresztmetszete állandó, $d = 30\,\mathrm{mm}$ átmérőjű kör. A C-ben az $\mathbf{F}_\mathrm{C}$ koncentrált erő terheli a tartót, míg a BC szakaszon az y-irányú p nagyságú, állandó intenzitású megoszló erőrendszer.

Feladatok: határozza meg a befogás keresztmetszetében az igénybevételekből adódó feszültségeloszlásokat, valamint ábrázolja a $\mathrm{P}_1 \ldots \mathrm{P}_5$ pontokban a feszültségeket!





2 Megoldás

A megoldás során szükségünk lesz a `sympy` modulra. Definiáljuk a feladatban szereplő szimbólumokat. A számolások során a vektorokat 3×1-es mátrixként adhatjuk meg.

In [1]:

```
import sympy as sp
sp.init_printing()

L1,L2,p,d = sp.symbols("L1,L2,p,d")

F_Cx,F_Cy,F_Cz = sp.symbols("F_Cx,F_Cy,F_Cz") #az FC vektor komponensei

FC = sp.Matrix([F_Cx,F_Cy,F_Cz]) #definiáljuk az FC vektort, azaz 3x1-es mátrix
#ha sorvektorra, azaz 1x3-as mátrixra lenne szükségünk,
#akkor azt sp.Matrix([[F_Cx,F_Cy,F_Cz]])-ként csinálhatnánk meg (2 db [ és ] ka
FC
```

executed in 1.32s, finished 10:32:20 2020-03-10

Out[1]:

$$\begin{bmatrix} F_{Cx} \\ F_{Cy} \\ F_{Cz} \end{bmatrix}$$

Megadjuk a feladatban szereplő adatokat, majd készítünk belőlük egy lisitát, aminek segítségével az összes adatot egyszerre tudjuk behelyettesíteni.

In [2]:

```
L1_adat = 3 #m
L2_adat = 2 #m
p_adat = 40 #N/m
d_adat = 30 #mm
FCx_adat = -150 #N
FCy_adat = 200 #N
FCz_adat = -50 #N

osszesadat = [(L1,L1_adat),(L2,L2_adat),(p,p_adat),(d,d_adat),(F_Cx,FCx_adat),(
```

executed in 6ms, finished 10:32:20 2020-03-10

2.1 Az igénybevételek meghatározása

A befogási keresztmetszet súlypontjára redukált erő:

$$\mathbf{F}_{\mathrm{A}} = \mathbf{F}_{\mathrm{C}} + pL_2(-\mathbf{j}),$$

ahol $\mathbf{j}$ az y irányba mutató egységvektor:

$$\mathbf{j} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

In [3]:

```
jV = sp.Matrix([0,1,0])
FA = FC+p*L2*(-jV)
FA.subs(osszesadat) #N
```

executed in 530ms, finished 10:32:20 2020-03-10

Out[3]:

$$\begin{bmatrix} -150 \\ 120 \\ -50 \end{bmatrix}$$

A befogási keresztmetszet súlypontjára redukált nyomaték:

$$\mathbf{M}_{\mathrm{A}} = \mathbf{r}_{\mathrm{AC}} \times \mathbf{F}_{\mathrm{C}} + \frac{1}{2}(\mathbf{r}_{\mathrm{AC}} + \mathbf{r}_{\mathrm{AB}}) \times (pL_2(-\mathbf{j})),$$

ahol $\mathbf{r}_{\mathrm{AB}}$ és $\mathbf{r}_{\mathrm{AC}}$:

$$\mathbf{r}_{\mathrm{AB}} = \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{r}_{\mathrm{AC}} = \begin{bmatrix} L_1 \\ 0 \\ -L_2 \end{bmatrix}.$$

In [4]:

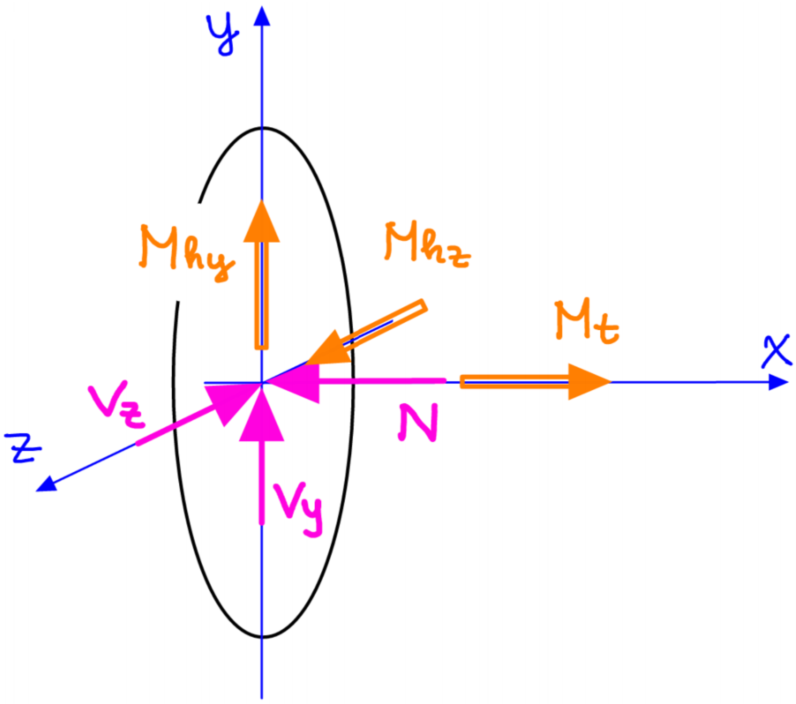
```python
rAB = sp.Matrix([L1,0,0])
rAC = sp.Matrix([L1,0,-L2])

#vektoriális szorzat angolul: cross product
#Sympy-ban: 'vektor1.cross(vektor2)'
#Például: a 'sp.Matrix([1,0,0]).cross(sp.Matrix([0,1,0]))' utasítás 'sp.Matrix(
MA = rAC.cross(FC)+((rAB+rAC)/2).cross(p*L2*(-jV))
MA.subs(osszesadat) #Nm
```

executed in 512ms, finished 10:32:21 2020-03-10

Out[4]:

$$\begin{bmatrix} 320 \\ 450 \\ 360 \end{bmatrix}$$

A vizsgált keresztmetszet igénybevételei a lenti ábrán láthatóak. A nyilak jelölik az erők és nyomatékok értelmét, a hozzájuk tartozó értékek pedig az igénybevétel nagyságát.



$\mathbf{F}_{\mathrm{A}}$ és $\mathbf{M}_{\mathrm{A}}$ alapján az igénybevételek nagysága az alábbi módon számítható:

$$N = -F_{\mathrm{Ax}}, \quad V_{\mathrm{y}} = F_{\mathrm{Ay}}, \quad V_{\mathrm{z}} = -F_{\mathrm{Az}},$$
$$M_{\mathrm{t}} = M_{\mathrm{Ax}}, \quad M_{\mathrm{hy}} = M_{\mathrm{Ay}}, \quad M_{\mathrm{hz}} = M_{\mathrm{Az}}.$$

A programban a vektorok x, y, z elemeit a `0,1,2` indexekkel tudjuk elérni.

In [5]:

```python
N = -FA[0] #[0]: első elem: x
Vy = FA[1] #[1]: második elem: y
Vz = -FA[2] #[2]: harmadik elem: z

Mt = MA[0]
Mhy = MA[1]
Mhz = MA[2]
```

executed in 5ms, finished 10:32:21 2020-03-10

2.2 Különböző terhelésekből származó feszültségek

Definiálnunk kell a keresztmetszeti jellemzőket. Innentől figyelnünk kell a mértékegységekre: eddig m-ben számoltunk, de a keresztmetszet adatai mm-ben adottak. Hogy majd MPa-ban kapjuk a feszültségeket, áttérünk mm-re.

$$A = \frac{d^2\pi}{4}, \qquad I_y = I_z = \frac{d^4\pi}{64}, \qquad I_\mathrm{p} = \frac{d^4\pi}{32}, \qquad K_y = K_z = \frac{d^3\pi}{32}, \qquad K_\mathrm{p} = \frac{d^3\pi}{16}.$$

In [6]:

```
A  = d**2*sp.pi/4
Iy = d**4*sp.pi/64
Iz = Iy
Ip = d**4*sp.pi/32
Ky = d**3*sp.pi/32
Kz = Ky
Kp = d**3*sp.pi/16
```

executed in 26ms, finished 10:32:21 2020-03-10

A különféle igénybevételekből származó feszültségek megadása során használni fogjuk az $r = d/2$ jelölést, valamint a ϱ koordinátát, mely az adott pont távolságát jelöli a keresztmetszet középpontjától. Ezeket, valamint az y és z koordinátákat definiálnunk kell a programban.

In [7]:

```
r = d/2
y,z,ϱ = sp.symbols("y,z,ϱ") #ϱ: \varrho + tab (a sima \rho + tab a ρ-t adja, ami
```

executed in 11ms, finished 10:32:21 2020-03-10

Az N normálerőből származó feszültség (negatív előjel: x és N ellentétes irányú):

$$\sigma_{x,N} = -\frac{N}{A}.$$

In [8]:

```
σxN = -N/A
σxN = σxN.subs(osszesadat).evalf(5) #behelyettesítjük az adatokat, numerikus ki
σxN #MPa
```

executed in 510ms, finished 10:32:21 2020-03-10

Out[8]:

-0.21221

A V_y nyíróerőből származó feszültség:

$$\tau_{xy,V_y} = \frac{4V_y}{3A}\left(1 - \left(\frac{y}{r}\right)^2\right).$$

In [9]:

```
τxyVy = 4*Vy/3/A*(1-(y/r)**2)
τxyVy = τxyVy.subs(osszesadat).evalf(5) #behelyettesítjük az adatokat, numeriku
τxyVy #MPa
```

executed in 543ms, finished 10:32:22 2020-03-10

Out[9]:

$$0.22635 - 0.001006y^2$$

A V_z nyíróerőből származó feszültség (negatív előjel: z és V_z ellentétes irányú):

$$\tau_{xz,V_z} = -\frac{4V_z}{3A}\left(1-\left(\frac{z}{r}\right)^2\right).$$

In [10]:

```
τxzVz = -4*Vz/3/A*(1-(z/r)**2)
τxzVz = τxzVz.subs(osszesadat).evalf(5) #behelyettesítjük az adatokat, numeriku
τxzVz #MPa
```

executed in 586ms, finished 10:32:22 2020-03-10

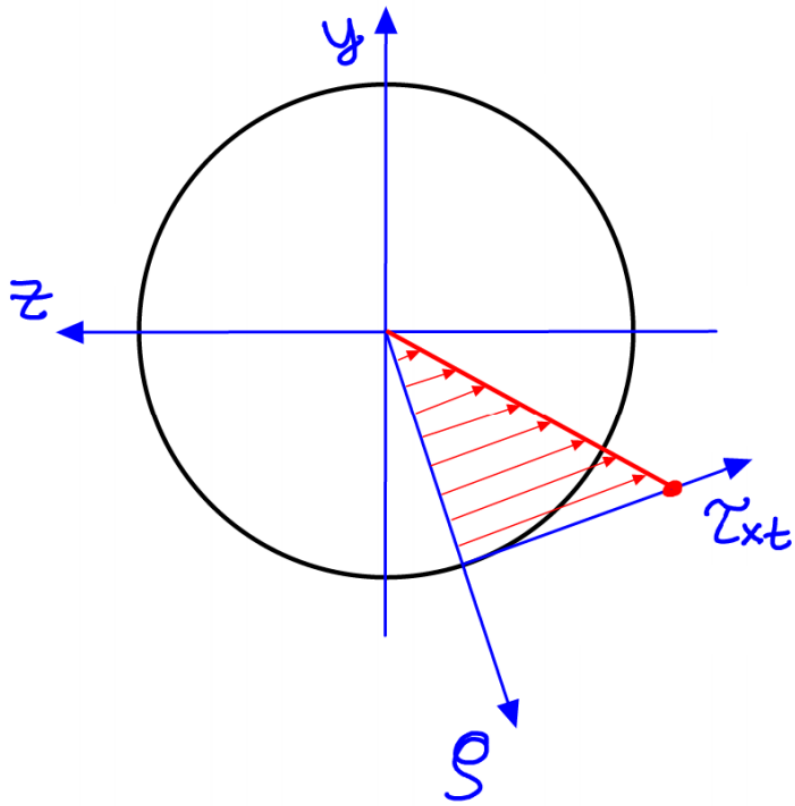
Out[10]:

$$0.00041917z^2 - 0.094314$$

Az M_t nyomatékból származó feszültség (ϱ: a pont távolsága a keresztmetszet középpontjától):

$$\tau_{xt} = \frac{M_\mathrm{t}}{I_\mathrm{p}}\varrho.$$

Ennek a feszültségnek az iránya mindig merőleges a vizsgált pontba húzott sugárra.



τ_{xt}-t kifejezhetük az y és z koordináták segítségével. $\varrho = \sqrt{y^2 + z^2}$. Ezek alapján:

$$\tau_{xt,y} = -\frac{M_t}{I_p} z,$$
$$\tau_{xt,z} = \frac{M_t}{I_p} y.$$

Ez a felírás teljesíti mind a feszültség nagyságára, mind az irányára vonatkozó követelményeket. Ez később, a számítások automatizálása során lesz számunkra hasznos felírás.

M_t-t Nmm-ben kell megadni!

In [11]:

```
τxt = 1000*Mt/Ip*ϱ
τxty = -1000*Mt/Ip*z
τxtz = 1000*Mt/Ip*y
τxt = τxt.subs(osszesadat).evalf(5)
τxty = τxty.subs(osszesadat).evalf(5) #behelyettesítjük az adatokat, numerikus 
τxtz = τxtz.subs(osszesadat).evalf(5)
τxt #MPa
```

executed in 1.38s, finished 10:32:24 2020-03-10

Out[11]:

$$4.0241\varrho$$

Az M_{hy} nyomatékből származó feszültség:

$$\sigma_{x,M_{hy}} = \frac{M_{hy}}{I_y} z.$$

M_{hy}-t Nmm-ben kell megadni!

In [12]:

```
σxMhy = 1000*Mhy/Iy*z
σxMhy = σxMhy.subs(osszesadat).evalf(5) #behelyettesítjük az adatokat, numeriku
σxMhy #MPa
```

executed in 509ms, finished 10:32:24 2020-03-10

Out[12]:

$$11.318z$$

Az M_{hz} nyomatékből származó feszültség (negatív előjel: a keresztmetszet igénybevételeinek értelme alapján):

$$\sigma_{x,M_{hz}} = -\frac{M_{hz}}{I_z} y.$$

M_{hz}-t Nmm-ben kell megadni!

In [13]:

```
σxMhz = -1000*Mhz/Iz*y
σxMhz = σxMhz.subs(osszesadat).evalf(5) #behelyettesítjük az adatokat, numeriku
σxMhz #MPa
```

executed in 524ms, finished 10:32:25 2020-03-10

Out[13]:

$$-9.0542y$$

2.3 Ábrázolás

Mivel nagyságrendi különbség van az értékek között, mindegyiket külön ábrázoljuk.

Az ábrázoláshoz szükségünk lesz a `matplotlib` modulra és a `numpy` modulból a `linspace` függvényre. Elkészítjük az $y = -r \ldots r$, $z = -r \ldots r$ és $\varrho = 0 \ldots r$ adatsorokat. Ehhez r-et "gyári" Python `float` -tá kell alakítanunk, mert a `linspace` nem fogadja el a `sympy` által haszált `float` -ot.

Megjegyzés: mivel y és z értékei ugyanúgy $-r$-től r-ig kellenek nekünk, ezért defininálhatnánk egy közös adatsort nekik.

In [14]:

```
import matplotlib.pyplot as plt
from numpy import linspace

r_ertek = float(r.subs(osszesadat).evalf(5))

y_ertekek = linspace(-r_ertek,r_ertek,201) #felveszünk 201 pontot '-r' és 'r' k
z_ertekek = linspace(-r_ertek,r_ertek,201) #felveszünk 201 pontot '-r' és 'r' k
ϱ_ertekek = linspace(0,r_ertek,201) #felveszünk 201 pontot 0 és 'r' között
```

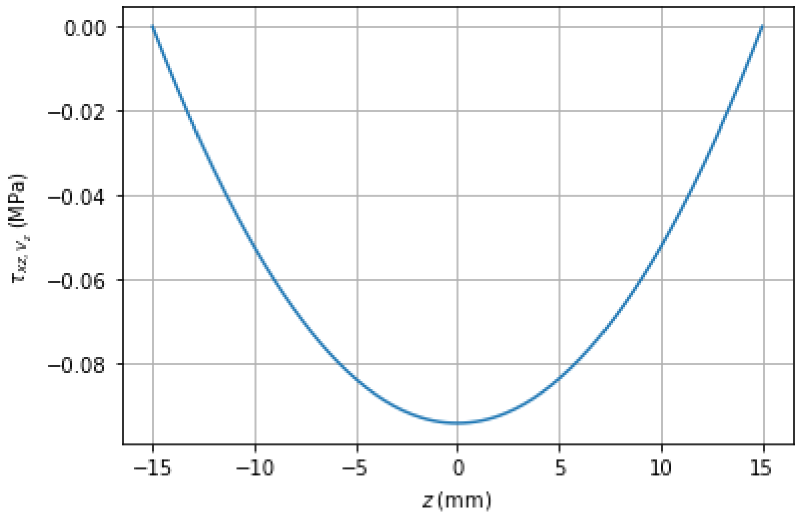
executed in 172ms, finished 10:32:25 2020-03-10

2.3.1 $\tau_{xz,V_z}(z)$ ábrázolása, a feszültség minimuma és maximuma:

In [15]:

```
#behelyettesítjük 'z' értékeit, kiértékeljük a függvényt
τxz_ertekek = [τxzVz.subs(z,z_ert).evalf(5) for z_ert in z_ertekek]
plt.plot(z_ertekek,τxz_ertekek)
plt.grid() #rács
plt.xlabel(r"$z \; \rm{(mm)}$") #vízszintes tengely felirata, 'r' karakter az el
plt.ylabel(r"$\tau_{xz,V_z} \;\rm{(MPa)}$") #függőleges tengely felirata, 'r' ka

print("Minimum feszültség, maximum feszültség:") #kiíratunk egy magyarázó szöveg
#a display parancs a számokat 'szépen' írja ki, ha korábban meghívtuk az sp.init
display(min(τxz_ertekek),max(τxz_ertekek))
```

executed in 1.49s, finished 10:32:26 2020-03-10




```
Minimum feszültség, maximum feszültség:
```

$$-0.094314$$

$$-8.5682 \cdot 10^{-8}$$

A számábrázolás pontatlanságából származó hibák miatt nem pontosan 0-t kapunk maximumnak.

A többi feszültség ábrázolását $\tau_{xz,V_z}(z)$ ábrázolásához nagyon hasonló módon kell megcsinálnunk. Amit módosítanunk kell a többi esetben, az a koordináta, a kiértékelendő függvény és a tengelyfeliratok. Ezt megoldhatnánk a fenti kód másolásával és kézi módosításával. Viszont ennél sokkal hatékonyabb, ha írunk rá egy saját függvényt.

Általában megéri saját függvényt írnunk, ha többször kell egy nagyon hasolnló utasítássorozatot kiadnunk. Jelen esetben az ábrázolás menete a következő lépésekből áll:

- függvény kiértékelése az adott helyeken,
- `plot` parancs,
- rács, tengelyfeliratok létrehozása,
- kirajzolás,
- minimum és maximum feszültség kiírása.

A Pythonban saját függvényt a `def fuggvenyneve(bemenet1,bemenet2,...):` szintaktikával definiálhatunk. A függvényhez tartozó utasításokat egy tabulátornyival (vagy 4 szóköznyivel) beljebb írjuk, azaz indentáljuk. Ha a függvényünknek valamilyen értéket, eredményt, stb.-t kell visszaadnia, azt a `return` `eredmeny` parancssal érhetjük el. Ezt tudjuk hozzárendelni egy változóhoz, például `valtozo=szamolas(adatok)` parancsra a `valtozo` -ba a `szamolas` függvény visszatérési értéke kerül. Most nem lesz szükségünk a `return` parancsra, mert csak ábrázolni szeretnénk az eredményeket.

In [16]:

```python
def abrazolas(koordinatak, kifejezes, xfelirat, yfelirat):
    #'koordinatak': azok a koordináta értékek, ahol ki akarjuk majd értékelni a
    #'kifejezes': amibe a 'koordinatak'-at behelyettesítjük
    #'xfelirat': vízszintes tengely felirata, LaTeX kód esetén kell az elejére 
    #'yfelirat': függőleges tengely felirata, LaTeX kód esetén kell az elejére 

    #függvény kiértékelése:
    #nem tudjuk előre, hogy 'y', 'z' és 'ϱ' közül melyik szerepel a kifejezésbe
    #ezért mind a 3 szimbólumba behelyettesítünk
    #most nem írhatunk listát, hogy egyszerre helyettesíthessünk be, mert 'list
    #ezért egyesével kell behelyettesítenünk, a '.subs()' parancs többszöri kia
    fesz_ertekek=[kifejezes.subs([(y,ertek),(z,ertek),(ϱ,ertek)]) for ertek in 
    plt.plot(koordinatak,fesz_ertekek)
    plt.grid() #rács
    plt.xlabel(xfelirat) #vízszintes tengely feliratának létrehozása
    plt.ylabel(yfelirat) #függőleges tengely feliratának létrehozása
    plt.show() #kirajzolás
    print("Minimum feszültség, maximum feszültség MPa-ban:") #kiíratunk egy mag
    display(min(fesz_ertekek),max(fesz_ertekek)) #visszatérési érték: min. és m
```

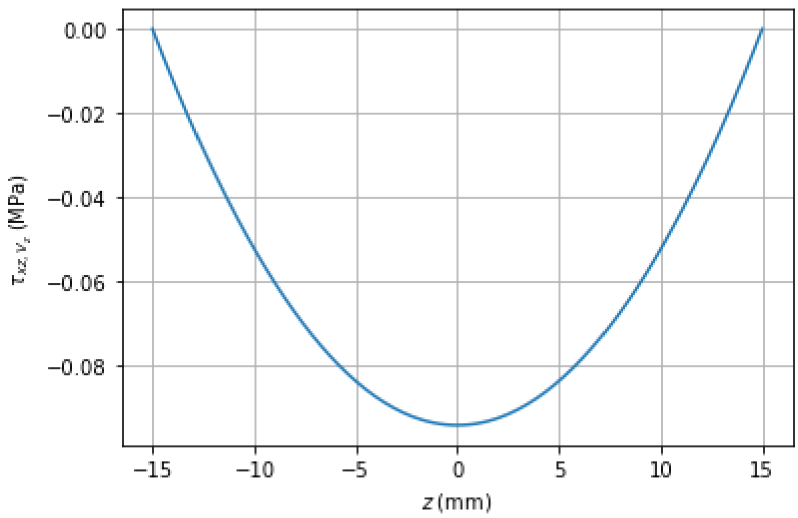
executed in 7ms, finished 10:32:26 2020-03-10

$\tau_{xz,V_z}(z)$-t ábrázoljuk a saját függvénnyel:

In [17]:

```python
abrazolas(z_ertekek,τxzVz,r"$z \; \rm{(mm)}$",r"$\tau_{xz,V_z} \;\rm{(MPa)}$")
```

executed in 1.49s, finished 10:32:28 2020-03-10




```
Minimum feszültség, maximum feszültség MPa-ban:
```

$$-0.094314$$

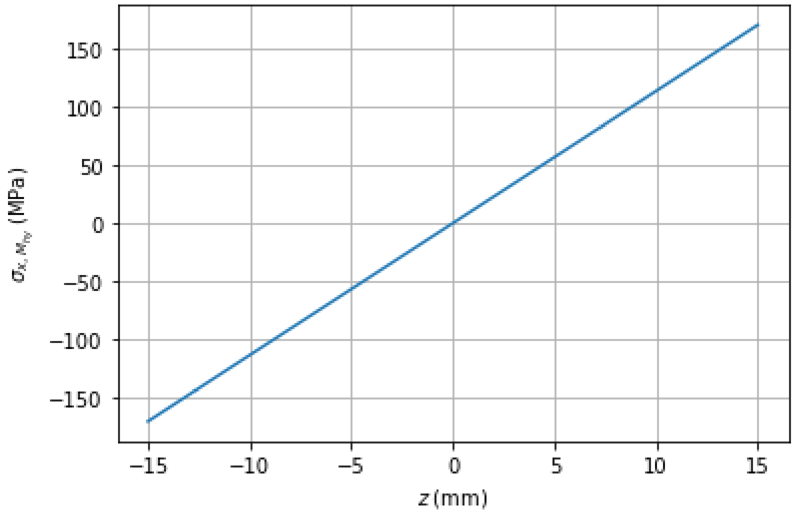
$$-8.5681676864624 \cdot 10^{-8}$$

2.3.2 $\sigma_{x,M_{hy}}(z)$ ábrázolása:

In [18]:

```
abrazolas(z_ertekek,σxMhy,r"$z \; \rm{(mm)}$",r"$\sigma_{x,M_{hy}} \;\rm{(MPa)}
```

executed in 1.53s, finished 10:32:29 2020-03-10




```
Minimum feszültség, maximum feszültség MPa-ban:
```

-169.765319824219

169.765319824219

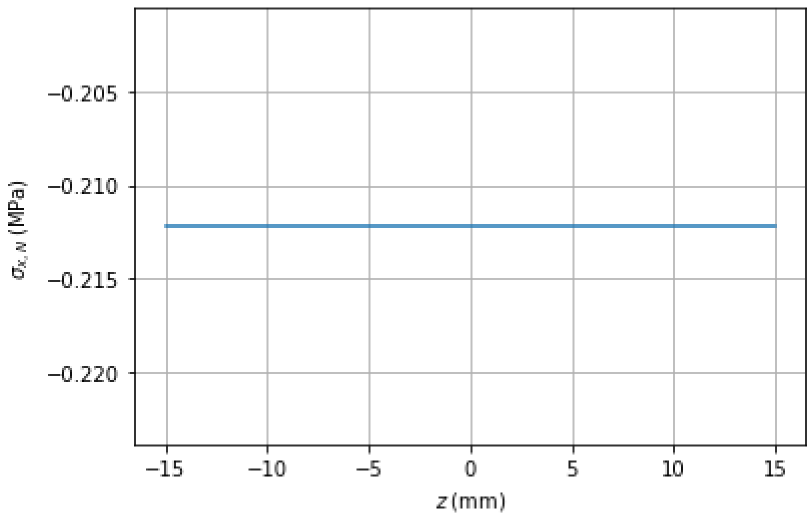
2.3.3 $\sigma_{x,N}$ ábrázolása:

$\sigma_{x,N}$ az egész keresztmetszeten konstans értékű, így bármelyik koordináta függvényében ábrázolhatnánk.

In [19]:

```
abrazolas(z_ertekek,σxN,r"$z \; \rm{(mm)}$",r"$\sigma_{x,N} \;\rm{(MPa)}$")
```

executed in 1.33s, finished 10:32:31 2020-03-10




```
Minimum feszültség, maximum feszültség MPa-ban:
```

-0.21221

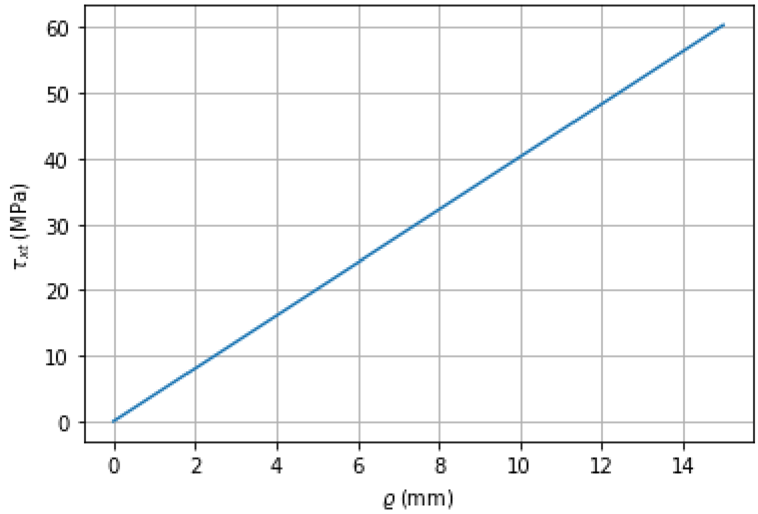
-0.21221

2.3.4 $\tau_{xt}(\varrho)$ ábrázolása:

In [20]:

```
abrazolas(ϱ_ertekek,τxt,r"$\varrho \; \rm{(mm)}$",r"$\tau_{xt}\;\rm{(MPa)}$")
```

executed in 1.44s, finished 10:32:32 2020-03-10




```
Minimum feszültség, maximum feszültség MPa-ban:
```

0

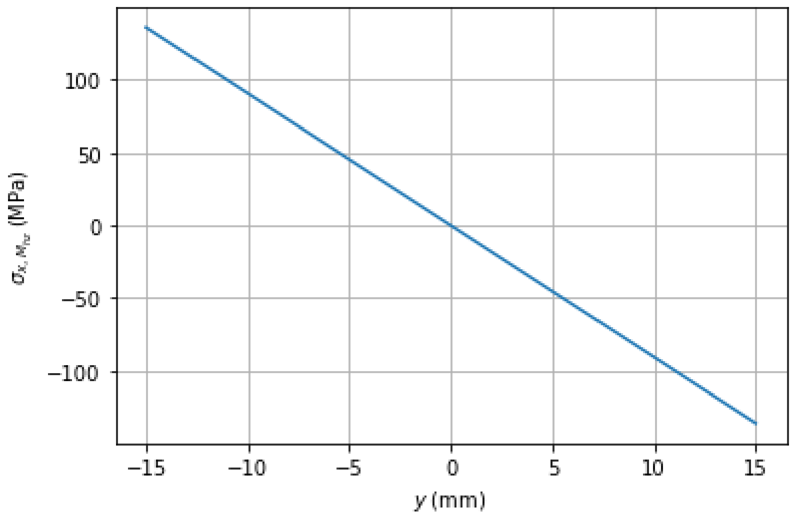
60.3610610961914

2.3.5 $\sigma_{x,M_{hz}}(y)$ ábrázolása:

In [21]:

```
abrazolas(y_ertekek,σxMhz,r"$y \; \rm{(mm)}$",r"$\sigma_{x,M_{hz}} \;\rm{(MPa)}
```

executed in 1.38s, finished 10:32:34 2020-03-10




```
Minimum feszültség, maximum feszültség MPa-ban:
```

−135.812530517578

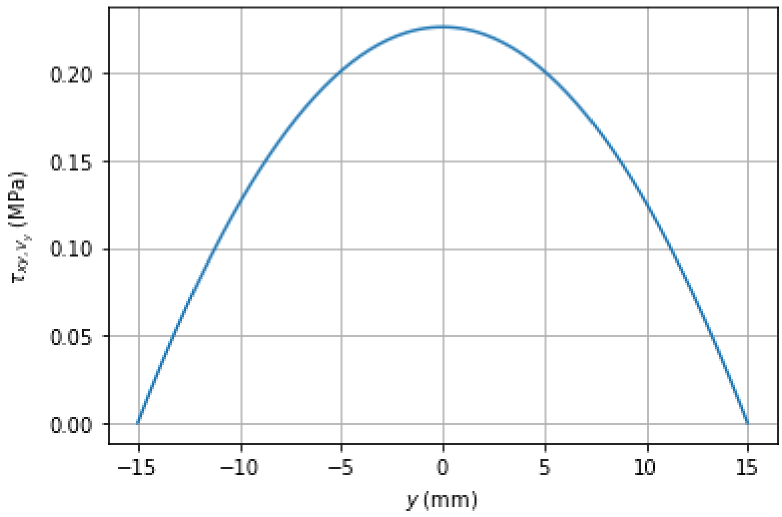
135.812530517578

2.3.6 $\tau_{xy,V_y}(y)$ ábrázolása:

In [22]:

```
abrazolas(y_ertekek,τxyVy,r"$y \; \rm{(mm)}$",r"$\tau_{xy,V_y} \;\rm{(MPa)}$")
```

executed in 1.50s, finished 10:32:35 2020-03-10

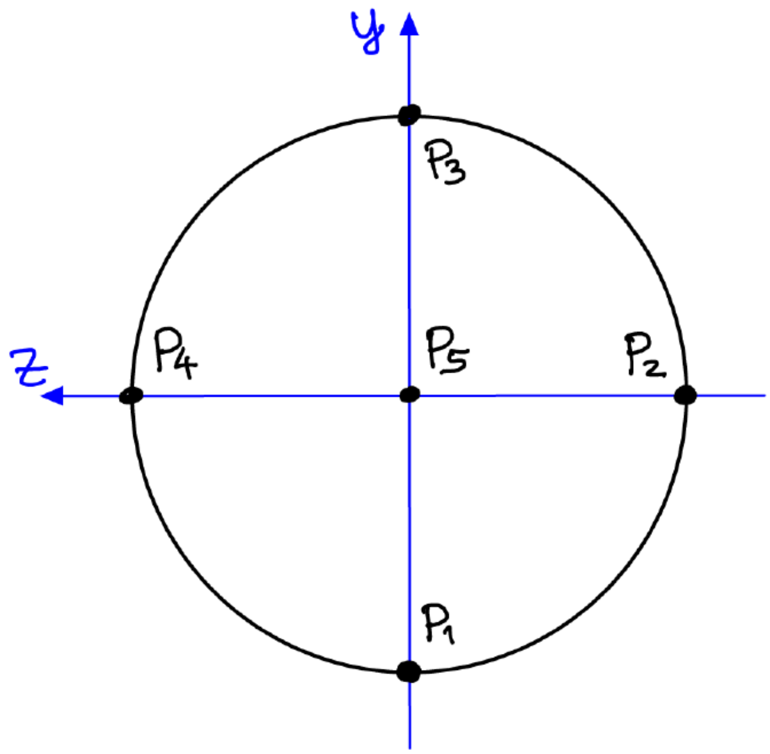



```
Minimum feszültség, maximum feszültség MPa-ban:
```

$1.69500708580017 \cdot 10^{-7}$

0.22635

2.4 Kijelölt pontok vizsgálata



A feszültségtenzor elemei:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix}.$$

A mi esetünkben $\sigma_y = \sigma_z = \tau_{yz} = \tau_{zy} = 0$, valamint $\sigma_x = \sigma_{x,N} + \sigma_{x,M_{hy}}(z) + \sigma_{x,M_{hz}}(y)$.

Definiáljuk az ennek megfelelő mátrixot szimbolikusan, amihez szükségünk lesz az egyes elemek szimbólumaira. Ennek során kihasználjuk, hogy a tenzor szimmetrikus és néhány eleméről tudjuk, hogy 0 lesz. Hogy elkerüljük az ütközést a korábbi változónevekkel, a `_TE` (mint Tenzor Elem) utótagot fogjuk használni. A σ_x-re fentebb leírt kifejezést is definiáljuk, mert σ_x minden pontban így számolható.

In [23]:

```
σx,τxy,τxz = sp.symbols(r"\sigma_{x},\tau_{xy},\tau_{xz}") #'r' karakter kell a
σx_eredo = σxN+σxMhy+σxMhz
σMx = sp.Matrix([[σx,τxy,τxz],[τxy,0,0],[τxz,0,0]])
σMx
```

executed in 520ms, finished 10:32:36 2020-03-10

Out[23]:

$$\begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & 0 & 0 \\ \tau_{xz} & 0 & 0 \end{bmatrix}$$

2.4.1 P1 pont

A pont (y, z, ϱ) koordinátái:

$$y_{\mathrm{P1}} = -r, \quad z_{\mathrm{P1}} = 0, \quad \varrho_{\mathrm{P1}} = r.$$

Ebben a pontban a τ_{xt} feszültség a $-z$ irányba mutat. Ennek megfelelően τ_{xz}-hez adjuk hozzá, negatív előjellel.

Korábban láthattuk, hogy a számábrázolási pontatlanság miatt azok az értékekre, melyeknek 0-nak kellett volna lennie (a keresztmetszet szabad szélén nem lehet 0-tól eltérő a nyíróerőből adódó feszültség), kicsi, de nem 0 számokat kaptunk. Ezeket a `round(n)` utasítással tudjuk kerekíteni, `n` tizedesjegyre. Ebben az esetben például a `round(5)` megfelelő eredményt ad.

In [24]:

```
yP1 = -r_ertek
zP1 = 0
ϱ_ertekP1 = r_ertek
σxP1 = σx_eredo.subs([(y,yP1),(z,zP1)]).round(5) #behelyettesítjük a koordinátá
τxyP1 = τxyVy.subs([(y,yP1),(z,zP1)]).round(5) #behelyettesítjük a koordinátaka
τxzP1 = (τxzVz+(-τxt)).subs([(y,yP1),(z,zP1),(ϱ,ϱ_ertekP1)]).round(5) #itt 'ϱ'-
σMxP1 = σMx.subs([(σx,σxP1),(τxy,τxyP1),(τxz,τxzP1)]) #beírjuk a mátrixba a 'P1
σMxP1.evalf(5) #5 értékesjegyre írjuk ki
```

executed in 523ms, finished 10:32:36 2020-03-10

Out[24]:

$$\begin{bmatrix} 135.6 & 0 & -60.455 \\ 0 & 0 & 0 \\ -60.455 & 0 & 0 \end{bmatrix}$$

Ezt a műveletsort végig kell csinálni az összes pontra. Emiatt ismét célszerű egy sajár függvényt definiálnunk, ami az y és z koordináták függvénynében visszatér a feszültségtenzorral. τ_{xy} és τ_{xz} esetén $\tau_{xt}(\varrho)$ helyett $\tau_{xt,y}(y,z)$-t és $\tau_{xt,z}(y,z)$-t használjuk. Így $\tau_{xy} = \tau_{xy}(y) + \tau_{xt,y}(y,z)$, valamint $\tau_{xz} = \tau_{xz}(z) + \tau_{xt,z}(y,z)$. Így ϱ nem szerepel a képletekben, és a feszültség iránya automatikusan kiadódik.

In [25]:

```
τxy_eredo=τxyVy+τxty
τxz_eredo=τxzVz+τxtz

def σMatrix(P):
    #'yP': a pont 'y' koordinátája
    #'zP': a pont 'z' koordinátája
    yP, zP = P
    koord_ertekek = [(y,yP),(z,zP)] #készítünk egy listát, hogy egyszerre tudju
    σxP = σx_eredo.subs(koord_ertekek).round(5) #behelyettesítjük a koorbinátá
    τxyP = τxy_eredo.subs(koord_ertekek).round(5)
    τxzP = τxz_eredo.subs(koord_ertekek).round(5)
    σMxP = σMx.subs([(σx,σxP),(τxy,τxyP),(τxz,τxzP)])
    return σMxP.evalf(5)
```

executed in 13ms, finished 10:32:36 2020-03-10

A P1-hez tartozó feszültségtenzort kiszámolhatjuk a saját függvénnyel:

In [26]:

```
P1 = [-r_ertek,0]
σMatrix(P1)
```

executed in 553ms, finished 10:32:37 2020-03-10

Out[26]:

$$\begin{bmatrix} 135.6 & 0 & -60.455 \\ 0 & 0 & 0 \\ -60.455 & 0 & 0 \end{bmatrix}$$

2.4.2 P2 pont

A pont (y, z) koordinátái:

$$y_{P1} = 0, \quad z_{P1} = -r.$$

In [27]:

```
P2 = [0,-r_ertek]
σMatrix(P2)
```

executed in 514ms, finished 10:32:37 2020-03-10

Out[27]:

$$\begin{bmatrix} -169.98 & 60.587 & 0 \\ 60.587 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

2.4.3 P3 pont

A pont (y, z) koordinátái:

$$y_{P1} = r, \quad z_{P1} = 0.$$

In [28]:

```
P3 = [r_ertek,0]
σMatrix(P3)
```

executed in 583ms, finished 10:32:38 2020-03-10

Out[28]:

$$\begin{bmatrix} -136.02 & 0 & 60.267 \\ 0 & 0 & 0 \\ 60.267 & 0 & 0 \end{bmatrix}$$

2.4.4 P4 pont

A pont (y, z) koordinátái:

$$y_{P1} = 0, \quad z_{P1} = r.$$

In [29]:

```
P4 = [0,r_ertek]
σMatrix(P4)
```

executed in 575ms, finished 10:32:38 2020-03-10

Out[29]:

$$\begin{bmatrix} 169.55 & -60.135 & 0 \\ -60.135 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

2.4.5 P5 pont

A pont (y, z) koordinátái:

$$y_{P5} = 0, \quad z_{P5} = 0.$$

In [30]:

```
P5 = [0,0]
σMatrix(P5)
```

executed in 582ms, finished 10:32:39 2020-03-10

Out[30]:

$$\begin{bmatrix} -0.21221 & 0.22635 & -0.09431 \\ 0.22635 & 0 & 0 \\ -0.09431 & 0 & 0 \end{bmatrix}$$

Mind az feszültségek ábrázolásának, mind különböző pontok példáján jól látható, hogy célszerű saját függvény írni a kód másolgatása és kézi szerkesztése helyett. Fontos előny az is, hogy ha egységesen módosítani szeretnénk például az összes ábra formázását, akkor elég egy helyen átírni a kódot. Ez a programhibák javítását is nagyban megkönnyíti.