Példa 1.10

Határozzuk meg az alábbi tartó AC és CB részein a keresztmetszet méreteit úgy, hogy a $b_2/a_2 = b_1/a_1 = 2$ feltétel mellett hajlításra megfeleljen a tartó! Adatok:

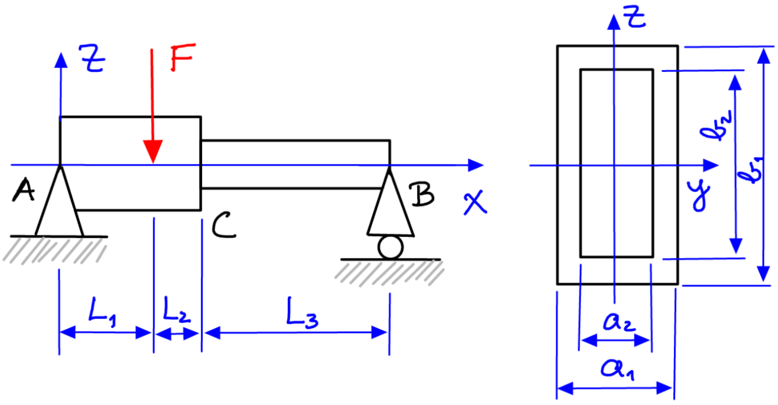
$L_1 = 2\mathrm{m},$

$L_2 = 1\mathrm{m},$

$L_3 = 4\mathrm{m},$

$F = 14\mathrm{kN},$

$\sigma_{\mathrm{meg}} = 100\mathrm{MPa}.$



Megoldás

A megoldás során szimbolikus számításokat fogunk végezni, ehhez importáljuk a `sympy` modult.

In [1]:

```
import sympy as sp #betöltjük a sympy modult
```

Definiáljuk a szükséges szimbólumokat. A feladatkiírás alapján b_1-t és b_2-t az a_1-ből és a_2-ből állítjuk elő, ezért ahhoz nem definiálunk külön szimbólumot.

In [2]:

```
a1,a2,L1,L2,L3,F,σ_meg = sp.symbols("a1,a2,L1,L2,L3,F,σ_meg")
b1 = 2*a1
b2 = 2*a2
```

A következő lépésben megadjuk a rendelkezésre álló adatokat.

In [3]:

```
L1_adat = 2 #m
L2_adat = 1 #m
L3_adat = 4 #m
F_adat = 14 #kN
σ_meg_adat = 100 #MPa
```

Ahhoz, hogy a behelyettesítést ne kelljen egyesével megtenni, csinálunk egy behelyettesítési listát, így amikor behelyettesítésre kerül sor, futtatáskor a program tudni fogja, hogy az adott szimbólum helyére milyen numerikus értéket kívánunk behelyettesíteni. Ennek formája:

```
behelyettesítési lista = [(szimbólum, adat), (szimbólum, adat),...,(szimb
ólum, adat)]
```

In [4]:

```
adatok = [(L1,L1_adat),(L2,L2_adat),(L3,L3_adat),(F,F_adat),(σ_meg,σ_meg_adat)]
```

Igénybevételi függvények

Reakcióerők

Az igénybevételi függvények meghatározásához szükségünk van a reakcióerőkre.
Az egyensúlyi egyenletek: erőegyensúly a z irányban és nyomatéki egyensúly az A pontra:

$$\sum F_z = 0: \qquad F_A + F_B - F = 0$$

$$\sum M_y^{\mathrm{A}} = 0: \qquad F_B(L_1 + L_2 + L_3) - FL_1 = 0$$

In [5]:

```
FB = L1/(L1+L2+L3)*F #nyomatéki egyensúlyból; FB-re szimbolikus kifejezés
FB #FB pozitív z irányba mutat, azaz felfelé
```

Out[5]:

$$\frac{FL_1}{L_1 + L_2 + L_3}$$

In [6]:

```
FA = (F-FB).simplify() #erőegyensúlyból; FA-ra szimbolikus kifejezés (egyszerűsí
tve a .simplify() függvénnyel)
FA #FA pozitív z irányba mutat, azaz felfelé
```

Out[6]:

$$\frac{F\,(L_2 + L_3)}{L_1 + L_2 + L_3}$$

F_A, F_B numerikusan:

In [7]:

```
#numerikus adatok behelyettesítése, eredmény kN-ban
FAn = FA.subs(adatok)
FBn = FB.subs(adatok)
```

In [8]:

```
FAn #kN
```

Out[8]:

$$10$$

In [9]:

```
FBn #kN
```

Out[9]:

$$4$$

Igénybevételi függvények

Az igénybevételi függvények felírhatók ez alapján. Ehhez szükségünk lesz az x koordinátára, mint szimbólumra.

In [10]:

```
x = sp.symbols("x")
```

Az igénybevételi függvények szakaszonként adhatóak meg. Ehhez a `Piecewise()` függvényt használhatjuk. Fontos, hogy a megfelelő sorrendben adjuk meg a szakaszokat, növekvő x szerint. A nyíró igénybevétel ($V(x)$):

In [11]:

```
V = sp.Piecewise((FA,x<=L1),(FA-F,x<=L1+L2+L3))
# Azaz: V(x) értéke konstans FA, ha x ≤ L1, illetve V(x) értéke konstans FA-F, h
a  L1 < x ≤ L1+L2+L3 (teljes hossz).
# Az L1 < x ≤ teljes hossz esetén az L1 < x részt nem kell megadni, azt a progra
m kezeli automatikusan.
V
```

Out[11]:

$$\begin{cases} \frac{F(L_2+L_3)}{L_1+L_2+L_3} & \text{for}\ L_1 \geq x \\ \frac{F(L_2+L_3)}{L_1+L_2+L_3} - F & \text{for}\ x \leq L_1 + L_2 + L_3 \end{cases}$$

Nyíró igénybevétel numerikusan:

In [12]:

```
Vn = V.subs(adatok) #numerikus értékek behelyettesítve
Vn #kN
```

Out[12]:

$$\begin{cases} 10 & \text{for } x \leq 2 \\ -4 & \text{for } x \leq 7 \end{cases}$$

Hasonlóan írhatjuk fel az $M_\mathrm{h}(x)$ hajlító igénybevételi függvényt is:

In [13]:

```
Mh = sp.Piecewise((-FA*x,x<=L1),(-FA*x+F*(x-L1),x<=L1+L2+L3))
# Azaz: Mh(x) értéke -FA*x, ha x ≤ L1, illetve Mh(x) értéke -FA*x+F*(x-L1), ha
 L1 < x ≤ teljes hossz.
Mh
```

Out[13]:

$$\begin{cases} -\frac{Fx(L_2+L_3)}{L_1+L_2+L_3} & \text{for } L_1 \geq x \\ -\frac{Fx(L_2+L_3)}{L_1+L_2+L_3} + F\left(-L_1 + x\right) & \text{for } x \leq L_1 + L_2 + L_3 \end{cases}$$

In [14]:

```
Mhn = Mh.subs(adatok) #numerikus értékek behelyettesítve
Mhn #kNm
```

Out[14]:

$$\begin{cases} -10x & \text{for } x \leq 2 \\ 4x - 28 & \text{for } x \leq 7 \end{cases}$$

Ha ellenőrizni szeretnénk, hogy a hajlító nyomatéki függvényt jól írtuk-e fel, akkor felhasználhatjuk a $\frac{d}{dx} M_h(x) = -V(x)$ összefüggést:

In [15]:

```
V_ellenorzes = -Mh.diff(x) #.diff(x): egyszer deriválunk x szerint
V_ellenorzes
```

Out[15]:

$$-\begin{cases} -\frac{F(L_2+L_3)}{L_1+L_2+L_3} & \text{for } L_1 \geq x \\ -\frac{F(L_2+L_3)}{L_1+L_2+L_3} + F & \text{for } x \leq L_1 + L_2 + L_3 \end{cases}$$

Az eredmény helyes, de a `simplify()` utasítással szebbé tehetjük:

In [16]:

```
V_ellenorzes = V_ellenorzes.simplify() #felülírjuk az eredeti függvényt az egyszerűsített verzióval
display(V_ellenorzes) #kiíratjuk az egyszerűsített szimbolikus eredményt
V_ellenorzes.subs(adatok).simplify() #numerikusan
```

$$\begin{cases} \frac{F(L_2+L_3)}{L_1+L_2+L_3} & \text{for}\ L_1 \geq x \\ -\frac{FL_1}{L_1+L_2+L_3} & \text{for}\ L_1 \geq -L_2 - L_3 + x \end{cases}$$

Out[16]:

$$\begin{cases} 10 & \text{for}\ x \leq 2 \\ -4 & \text{for}\ x \leq 7 \end{cases}$$

Ez ránézésre valóban azonos a korábban megadott V-vel. Ezt úgy is ellenőrizhetjük, hogy kivonjuk egymásból a kettőt, és 0 eredményt kapunk:

In [17]:

```
(V - V_ellenorzes).simplify()
```

Out[17]:

$$0$$

Ábrázolás

Ehhez szükségünk lesz matplotlib modulra és egy olyan függvényre, amivel könnyen hozhatunk létre egyenlő távolságra lévő értékeket.

In [18]:

```
import matplotlib.pyplot as plt # Betöltjük a matplotlib modult, amivel plotolhatunk.
from numpy import linspace # Betöltjük csak linspace függvényt a numpy modulból (tehát nem a modul
                            # összes függvényét), amivel majd a plotoláshoz készítjük az x értékek sorozatát
```

Létrehozzuk az x, V és Mh adatsort a plotoláshoz:

- `xs` : `N_osztas` darab x érték `0` és `L` között:

  ```
  xs = linspace(0,L,N_osztas)
  ```

- `Vxs` és `Mhxs` : `N_osztas` darab nyíróerő és hajlítónyomatéki érték az `xs` -ben megadott helyeken [list comprehension (https://www.datacamp.com/community/tutorials/python-list-comprehension?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c486527602543&utm_loc_interest_ms=&utm_loc_physical_ms=9063082&gclid=EAIaIQobChMIrZvm-9Hi5wIVR7TtCh0n4w3nEAAYASAAEgJNk_D_BwE)](https://www.datacamp.com/community/tutorials/python-list-comprehension?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c486527602543&utm_loc_interest_ms=&utm_loc_physical_ms=9063082&gclid=EAIaIQobChMIrZvm-9Hi5wIVR7TtCh0n4w3nEAAYASAAEgJNk_D_BwE) segítségével (tehát gyakorlatilag a létrehozott osztáshelyeken kiértékeljük az adott függvényeket):

  ```
  Vxs = [V(x) for x in xs]
  Mhxs = [Mh(x) for x in xs]
  ```

  A *list comprehension* gyakorlatilag egy tömör `for` ciklus listák gyors létrehozására.

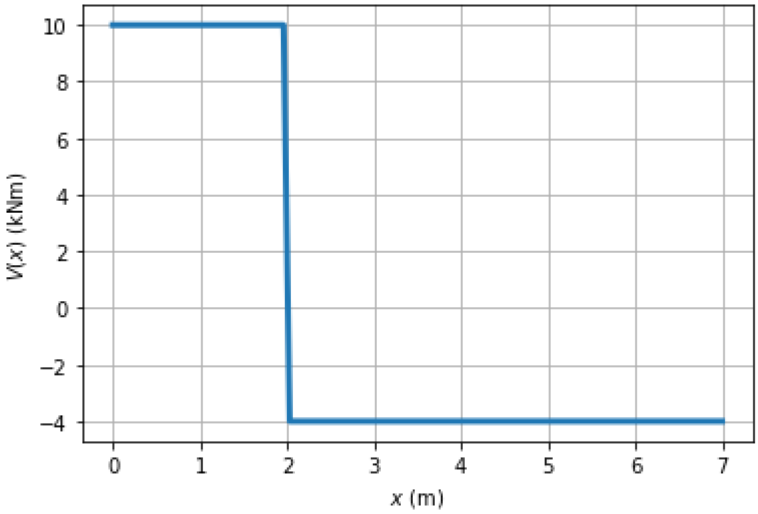
In [19]:

```
L = (L1 + L2 + L3).subs(adatok) # A rúd teljes hossza.
xs = linspace(0.,float(L),101)  # Felveszünk 101 darab x értéket a rúd hossza mentén
                                # (az L hosszat át kell alakítani float-tá).
Vxs = [Vn.subs(x,xi) for xi in xs] # Kiszámoljuk a nyíróerő függvény értékeit a megadott x helyeken.
# Vn.subs(x,xi): a Vn függvényben lévő x változó helyére behelyettesítjük xi-t.
Mhxs = [Mhn.subs(x,xi) for xi in xs] # Kiszámoljuk a hajlítónyomatéki függvény értékeit a megadott x helyeken.
```

Plotoljuk a függvényeket. Mivel V(x)-ben szakadás van x=2-nél, láthatjuk, hogy a vonal nem teljesen függőleges. Ez csak a kirajzolás "hibája".

In [20]:

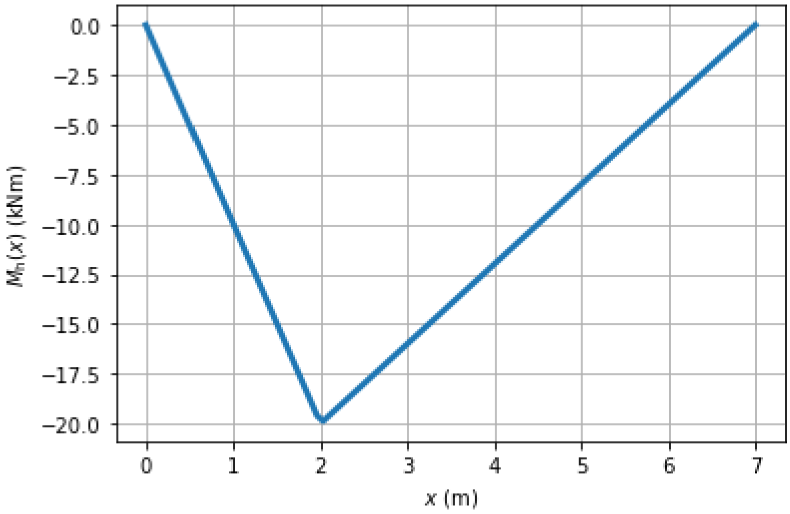
```
plt.plot(xs, Vxs, lw = 3) # A V(x) függvény képének létrehozása 3-as vonalvastag
sággal (linewidth = lw).
plt.xlabel("$x$ (m)") #x tengelyhez tartozó tengelyfelirat
plt.ylabel("$V(x)$ (kNm)") #y tengelyhez tartozó tengelyfelirat
plt.grid() #rács
plt.show() #kirajzolás
```



In [34]:

```
plt.plot(xs, Mhxs, lw = 3) #az Mh(x) függvény képének létrehozása 3-as vonalvast
agsággal (linewidth = lw)
plt.ylabel(r"$M_{\rm h}(x)$ (kNm)") # x tengelyhez tartozó tengelyfelirat. Az r
 kell az "" elé, ha '\' (backslash)
                                     # karaktert szeretnénk a stringbe tenni a La
TeX parancsok miatt.

plt.xlabel("$x$ (m)") #y tengelyhez tartozó tengelyfelirat
plt.grid() #rács létrehozása
plt.show() #kirajzolás
```



A nyomaték igénybevételi ábrájáról leolvasható, hogy az AC szakaszon $x = 2$-nél (vastag KM), míg a CB szakaszon (vékony KM) a keresztmetszetváltásnál, azaz $x = 3$-nál a veszi fel hajlító igénybevétel az abszolút értékben maximális értékét.

Méretezés

AC szakasz

Maximális hajlító igénybevétel:

$$M_{\text{h,ACmax}} = M_{\text{h}}(x = 2)$$

Keresztmetszeti tényező:

$$K_{\text{AC}} = \frac{I_{\text{AC}}}{\frac{b_1}{2}} = \frac{\frac{a_1 b_1^3}{12}}{\frac{b_1}{2}}$$

Megengedett feszülség:

$$\sigma_{\text{meg}} = \frac{|M_{\text{h,ACmax}}|}{K_{\text{AC}}}$$

Mértékegységek egyeztetése: $M_{\text{h,ACmax}}$ átváltása Nmm-re.

In [22]:

```
Mh_ACmax=Mhn.subs(x,2)*1e6 #mértékegység átváltás 1 kNm = 10^6 Nmm -> 1e6 = 10^6
Mh_ACmax #Nmm
```

Out[22]:

$$-20000000.0$$

In [23]:

```
I_AC=a1*b1**3/12 #ism: ** a hatványozás jele
I_AC #b1 = 2*a1 automatikus behelyettesítésével (a szimbólumok definiálása miat
t, lsd. a dokumentum eleje)
```

Out[23]:

$$\frac{2a_1^4}{3}$$

In [24]:

```
K_AC=I_AC/(b1/2) #keresztmetszeti tényező értéke a1 függvényében
K_AC
```

Out[24]:

$$\frac{2a_1^3}{3}$$

In [25]:

```
# Kiszámoljuk a minimálisan szükséges keresztmetszeti tényezőt:
K_AC_min=sp.Abs(Mh_ACmax)/σ_meg #sp.Abs(): abszolút érték
K_AC_min=K_AC_min.subs(adatok)
K_AC_min # mm^3
```

Out[25]:

$$200000.0$$

A minimálisan szükséges a_1 méret:

In [26]:

```
# Kiszámoljuk a1-t a keresztmetszeti tényező értékének átrendezésével:
a1_min=sp.root(3/2*K_AC_min,3)  #sp.root(valami,n): valaminek az n. gyöke
a1_min.evalf(5) #mm  - 5 értékes jegyre
```

Out[26]:

$$66.943$$

A minimálisan szükséges b_1 méret:

In [27]:

```
b1.subs(a1,a1_min).evalf(5) #mm
```

Out[27]:

$$133.89$$

CB szakasz

Maximális hajlító igénybevétel:

$$M_{\mathrm{h,CBmax}} = M_{\mathrm{h}}(3)$$

Keresztmetszeti tényező:

$$K_{\mathrm{CB}} = \frac{I_{\mathrm{CB}}}{\frac{b_2}{2}} = \frac{\frac{a_2 b_2^3}{12}}{\frac{b_2}{2}}$$

Megengedett feszülség:

$$\sigma_{\mathrm{meg}} = \frac{|M_{\mathrm{h,CBmax}}|}{K_{\mathrm{CB}}}$$

Mértékegységek egyeztetése: $M_{\mathrm{h,CBmax}}$ átváltása Nmm-re.

In [28]:

```
Mh_CBmax = Mhn.subs(x,3)*1e6 #mértékegység átváltás 1 kNm = 10^6 Nmm -> 1e6 = 10
^6
Mh_CBmax #Nmm
```

Out[28]:

$$-16000000.0$$

In [29]:

```
I_CB = a2*b2**3/12
I_CB #b2 = 2*a2 automatikus behelyettesítésével (a szimbólumok definiálása miat
t)
```

Out[29]:

$$\frac{2a_2^4}{3}$$

In [30]:

```
K_CB=I_CB/(b2/2) #kifejezzük a keresztmetszeti tényezőt a2-vel
K_CB
```

Out[30]:

$$\frac{2a_2^3}{3}$$

In [31]:

```
#kiszámoljuk a minimális szükséges keresztmetszeti tényezőt
K_CB_min=sp.Abs(Mh_CBmax)/σ_meg #sp.Abs(): abszolút érték
K_CB_min=K_CB_min.subs(adatok)
K_CB_min # mm^3
```

Out[31]:

$$160000.0$$

A minimálisan szükséges a_2 méret:

In [32]:

```
#kiszámoljuk a1-t a keresztmetszeti tényező értékének átrendezésével
a2_min=sp.root(3/2*K_CB_min,3)  #sp.root(valami,n): valaminek az n. gyöke
a2_min.evalf(5) #mm  - 5 értékes jegyre
```

Out[32]:

$$62.145$$

A minimálisan szükséges b_2 méret:

In [33]:

```
b2.subs(a2,a2_min).evalf(5) #mm
```

Out[33]:

$$124.29$$