PROPIETA’ DELLA SEZIONE

```
## Dati sezione

B=2.45          # [m]
L=5.8           # [m]
d=3.10          # [m]
t=0.30          # [m]
Hn=91.5         # [m]

b=B-t/2         # [m]
l=L-t           # [m]

## Prop Materiale

E1 = 25000 # MPa
G1 = 10800 # Mpa
En=E1*1000 # KN/m2
Gn=G1*1000 # KN/m2

## Area

An=6*(B-t)+2*L

## Momenti d’inerzia

# [m**4] Momento d’inerzia asse 1
I1=L*(2*B+d)**3/12-2*(l/2-t)*(2*(B-t)+d)**3/12-3*t*d**3/12

#I1=(L*t**3/12+L*t*(B-t/2+d/2)**2)*2+6*((B-t)**3*t/12+(B-t)*t*((B-t)/2+d/2)**2)

# [m**4] Momento d’inerzia asse 2
I2=2*t*L**3/12+2*t**3*(B-t)/12+4*t**3*(B-t)/12+4*t*(B-t)*((L-t)/2)**2

# [m**4] Momento d’inerzia polare
I0=I1+I2

# [m**4] Momento d’inerzia settoriale
Iw=t*(b+d/2)**2*l**3/6+t*l**2*((b+d/2)**2*b+b**3/3+b**2*(b+d/2))

# [m**4] momento d’inerzia torsionale
It=6*(b*t**3)/3+2*(l*t**3)/3

print(’A=’, round(An, 3), ’m^2’)
print(’I1=’, round(I1, 3), ’m^4’)
print(’I2=’, round(I2, 3), ’m^4’)
print(’I0=’, round(I0, 3), ’m^4’)
print(’Iw=’, round(Iw, 3), ’m^4’)
print(’It=’, round(It, 3), ’m^4’)
```

SOLUZIONE LINEA ELASTICA PER LA FLESSIONE

```
import numpy as np
import sympy as sy
import matplotlib.pyplot as plt

## simbolico

c1, c2, c3, c4 = sy.symbols ('c1 c2 c3 c4') # costanti d'integrazione
p1=sy.symbols('p1') # carico distribuito in direzione 1
E,J2,H =sy.symbols ('E J2 H') # Modulo di Young & Momento d'inerzia asse 2
z=sy.symbols ('z') # ascissa curvilinea & altezza

## Soluzione dell'equazione differenziale della trave con carico distribuito

v1=(p1*z**4/24+c1*z**3/6+c2*z**2/2+c3*z+c4)*1/(E*J2) #spostamento tratto 1
r2=sy.diff(v1,z) #rotazione
M2=E*J2*sy.diff(v1,z,z) #Momento
T2=E*J2*sy.diff(v1,z,z,z) #Taglio

## condizioni al contorno

cc1=sy.Eq(v1.subs(z,0),0) #spostamento nullo in s=0
cc2=sy.Eq(r2.subs(z,0),0) #rotazione nulla in s=0
cc3=sy.Eq(M2.subs(z,H),0) #Momento in s=h
cc4=sy.Eq(T2.subs(z,H),0) #Taglio in s=h

## RISOLVO IL SISTEMA

sol=sy.solve((cc1,cc2,cc3,cc4),(c1,c2,c3,c4))
v1=v1.subs(c1,sol[c1]).subs(c2,sol[c2]).subs(c3,sol[c3]).subs(c4,sol[c4])
r2=r2.subs(c1,sol[c1]).subs(c2,sol[c2]).subs(c3,sol[c3]).subs(c4,sol[c4])
M2=M2.subs(c1,sol[c1]).subs(c2,sol[c2]).subs(c3,sol[c3]).subs(c4,sol[c4])
T1=T2.subs(c1,sol[c1]).subs(c2,sol[c2]).subs(c3,sol[c3]).subs(c4,sol[c4])

#print(sol)
print('v1=', sy.simplify(v1) )
print('M2=', sy.simplify(M2) )
print('T2=', sy.simplify(T2) )
```

SOLUZIONE LINEA ELASTICA PER LA TORSIONE

```
import numpy as np
import sympy as sy
import matplotlib.pyplot as plt

## simbolico

A, B, C, D = sy.symbols ('A B C D') # costanti d'integrazione

w,m=sy.symbols ('w m') # rotazione e momento distribuito

# Modulo di Young, Momento d'inerzia settoriale ,constante di torsione
E,Jw=sy.symbols ('E Jw')
z,H=sy.symbols ('z H ') # ascissa curvilinea & altezza
a = sy.Symbol('a', real=True) # rigidezza torsionale

## Soluzione dell'equazione differenziale della trave con momento torcente distribuito

w=A*sy.cosh(a*z)+B*sy.sinh(a*z)+C*z+D-m*z**2/(2*E*Jw*a**2) #rotazione
dw=sy.diff(w,z) #ingobbamento
Mt=a**2*E*Jw*dw-E*Jw*sy.diff(w,z,z,z) #Momento torcente
Bw=E*Jw*sy.diff(w,z,z,z) #Bimomento

## condizioni al contorno

cc1=sy.Eq(w.subs(z,0),0) #rotazione nulla in z=0
cc2=sy.Eq(dw.subs(z,0),0) #ingobbamento nullo in z=0
cc3=sy.Eq(Mt.subs(z,H),0) #Momento torcente nullo in z=H
cc4=sy.Eq(Bw.subs(z,H),0) #Bimomento nullo in z=H

## risolvo il sistema
sol=sy.solve((cc1,cc2,cc3,cc4),(A,B,C,D))
w=w.subs(A,sol[A]).subs(B,sol[B]).subs(C,sol[C]).subs(D,sol[D])
dw=dw.subs(A,sol[A]).subs(B,sol[B]).subs(C,sol[C]).subs(D,sol[D])
Mt=Mt.subs(A,sol[A]).subs(B,sol[B]).subs(C,sol[C]).subs(D,sol[D])
Bw=Bw.subs(A,sol[A]).subs(B,sol[B]).subs(C,sol[C]).subs(D,sol[D])
Mts=-sy.diff(Bw,z)


#print(sol)
print('w=', sy.simplify(w))
print('dw=', sy.simplify(dw))
print('Mt=', sy.simplify(Mt))
print('Mts=', sy.simplify(Mts))
print('Bw=', sy.simplify(Bw))
```

CALCOLO DELLE TENSIONI

```
exec (open ("Numerico.py") .read ())
from mpl_toolkits import mplot3d

## MESHGRID
ee1 = (np.linspace (0,b, num=len(zn), endpoint=True))
ee2 = (np.linspace (0,l/2, num=len(zn), endpoint=True))


Z , T1nn = np.meshgrid(zn,T1n)
Z , Mtsnn = np.meshgrid(zn,Mtsn)
Z , Bwnn = np.meshgrid(zn,Bwn)
Z , M2nn = np.meshgrid(zn,M2n)
e1, Z = np.meshgrid(ee1,zn)
e2, Z = np.meshgrid(ee2,zn)

s2=b-e1
s1=l/2-e2
#x1 = (np.linspace (-l/2,0, num=92, endpoint=True))

## funzione ingobamento

psi_1 =-s1*(b+d/2)
psi_2 =-(b+d/2)*l/2-s2*l/2

## Calcolo momenti statici
# Momento statico S1
S1_1 = t*e1*(-(e1+d)*0.5)
S1_2 = t*b*(-(b+d)/2)+e2*t*(-(b+d))

# Momento statico S2
S2_1 = t*e1*l/2
S2_2 = t*b*l/2+t*e2*(l/2-e2/2)

# Momento statico settoriale
Spsi_1 = -t*l/2*((b+d/2)*b+b**2/2)-t*(b+d/2)*(l**2/8-s1**2/2)
Spsi_2 = -t*l/2*((b+d/2)*(b-s2)+(b**2/2-s2**2/2))

## Calcolo tensioni tangenziali
# tensioni tangenziali dovuti a (T1)
tau_T1_1 = - T1nn*S2_1/(I2*t)
tau_T1_2 = - T1nn*S2_2/(I2*t)

# tensioni tangenziali dovuti a (Mts)
tau_Mts_1 =  Mtsnn*Spsi_1/(Iw*t)
tau_Mts_2 =  Mtsnn*Spsi_2/(Iw*t)

# tensioni tangenziali totali
tau_1=tau_T1_1+tau_Mts_2
tau_2=tau_T1_2+tau_Mts_1

## Calcolo tensioni normali
# tensioni normali dovuti a (M2)
sigma_M2_1 = -M2nn/I2*l/2
sigma_M2_2 = -M2nn/I2*s1
```

```
# tensioni normali dovuti a (B)
sigma_B_1 = Bwnn*psi_1/Iw
sigma_B_2 = Bwnn*psi_2/Iw

# tensioni normali totali
sigma_1 = sigma_M2_1 + sigma_B_2
sigma_2 = sigma_M2_2 + sigma_B_1

### PLOT
exec (open ("plot_tensioni_tang.py") .read ())
exec (open ("plot_tensioni_normal.py") .read ())
```

VALORI NUMERICI

```
exec (open ("LE_Bending.py") .read ())
exec (open ("LE_Torsion.py") .read ())
exec (open ("Prop_sez.py") .read ())

# Problema Numerico

p1n=1.2*16.5 # KN/m
q=I0/It
an=np.sqrt(Gn*I0/(En*Iw*q))
mn=p1n*4.25

zn=np.linspace( 0.00, Hn , 20) # ascissa

v1n=np.zeros(len(zn)) # spostamento in direzione 1
r2n=np.zeros(len(zn)) # rotazione in direzione 2
T1n=np.zeros(len(zn)) # Taglio in direzione 1
M2n=np.zeros(len(zn)) # Momento in direzione 2

wn=np.zeros(len(zn)) # rotazione torzionale
dwn=np.zeros(len(zn)) # ingobbamento
Bwn=np.zeros(len(zn)) # Bimomento
Mtsn=np.zeros(len(zn)) # Momento torcente secondario
Mtpn=np.zeros(len(zn)) # Momento torcente primario
Mtn=np.zeros(len(zn)) # Momento torcente totale

for i in range(0,(len(zn))):
    v1n[i]=np.float(v1.subs(z,zn[i]).subs(E,En).subs(J2,I2).subs(p1,p1n).subs(H,Hn))
    r2n[i]=np.float(r2.subs(z,zn[i]).subs(E,En).subs(J2,I2).subs(p1,p1n).subs(H,Hn))
    M2n[i]=np.float(M2.subs(z,zn[i]).subs(E,En).subs(J2,I2).subs(p1,p1n).subs(H,Hn))
    T1n[i]=np.float(T1.subs(z,zn[i]).subs(E,En).subs(J2,I2).subs(p1,p1n).subs(H,Hn))
    wn[i]=np.float(w.subs(z,zn[i]).subs(E,En).subs(Jw,Iw).subs(a,an).subs(H,Hn).subs(m,mn))
    dwn[i]=np.float(dw.subs(z,zn[i]).subs(E,En).subs(Jw,Iw).subs(a,an).subs(H,Hn).subs(m,mn))
    Mtsn[i]=np.float(Mts.subs(z,zn[i]).subs(E,En).subs(Jw,Iw).subs(a,an).subs(H,Hn).subs(m,mn))
    Mtn[i]=np.float(Mt.subs(z,zn[i]).subs(E,En).subs(Jw,Iw).subs(a,an).subs(H,Hn).subs(m,mn))
    Bwn[i]=np.float(Bw.subs(z,zn[i]).subs(E,En).subs(Jw,Iw).subs(a,an).subs(H,Hn).subs(m,mn))
    # Mtpn[i]=Gn*I0*dwn[i]/q
    Mtpn[i]=Mtn[i]-Mtsn[i]


print('v1{max}=', round(max(v1n)*1000,3), 'mm' )
print('spostamento massimo consentito=', 'H/400=',Hn*1000/400, 'mm' )
print('M2{max}=', round(max(M2n),3),'KN*m' )
print('T1{max}=', round(min(T1n),3),'KN')
print('w{max}=', round(max(wn),3),'1/m')
print('ingobbamento{max}=', round(min(dwn),3),'1/m')
print('Mtp{max}=', round(max(Mtpn),3),'KN*m' )
print('Mts{max}=', round(max(Mtsn),3),'KN*m')
print('Mtn{max}=', round(max(Mtn),3),'KN*m')

## plot
exec (open ("Bending_plot.py") .read ())
exec (open ("Torsion_plot.py") .read ())
```

```
v1= p1*z**2*(6*H**2 - 4*H*z + z**2)/(24*E*J2)
M2= p1*(H**2 - 2*H*z + z**2)/2
T2= c1 + p1*z
w= -m*(-2*H*a*z + 2*H*sinh(a*z) - 2*H*cosh(a*z)/tanh(H*a) + 2*H/tanh(H*a) + a*z**2)/(2*E*Jw*a**3)
dw= m*(H*sinh(a*z)/tanh(H*a) - H*cosh(a*z) + H - z)/(E*Jw*a**2)
Mt= m*(H - z)
Mts= H*a*m*(sinh(a*z) - cosh(a*z)/tanh(H*a))
Bw= H*m*(sinh(a*z)/tanh(H*a) - cosh(a*z))
A= 24.5 m^2
Il= 79.766 m^4
I2= 29.296 m^4
I0= 109.062 m^4
Iw= 654.318 m^4
It= 0.223 m^4
v1{max}= 236.872 mm
spostamento massimo consentito= H/400= 228.75 mm
M2{max}= 82885.275 KN*m
T1{max}= -1811.7 KN
w{max}= 0.013 1/m
ingobbamento{max}= 0.0 1/m
Mtp{max}= 7815.933 KN*m
Mts{max}= -69.05 KN*m
Mtn{max}= 7699.725 KN*m
```

Figura 1: VALORI