Bem Vindo à Calculadora de Vigas!

Feita por Laryssa Ferro e Lucas Silva

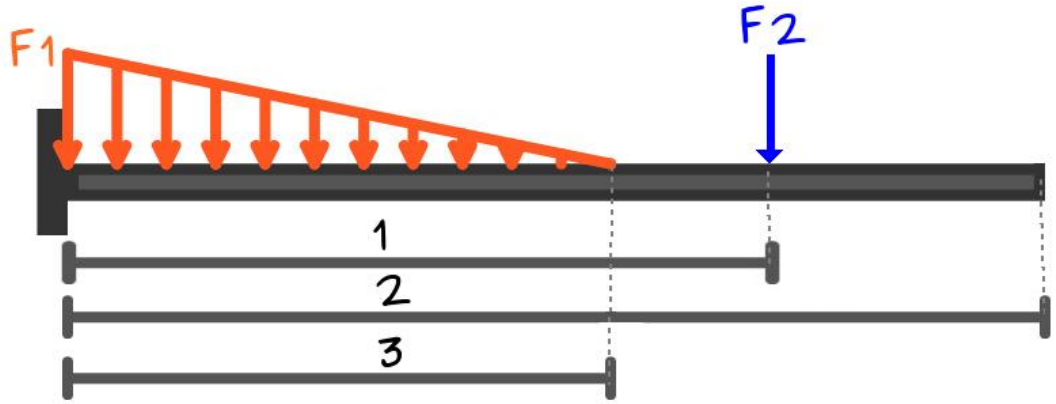
- Aqui é possivel ver o codigo escrito e os dados que ele gera!
- É só clicar no Run na barra no topo da pagina para entrar com novos dados ou graficos.
- Sempre que uma celular gerar uma saida, ela aparecera logo abaixo da celula!
- Prefira seguir de forma a descer a página, ou seja, ao alterar um dado reinicie desde o inicio da pagina!

As duas celulas a seguir sao apenas para que o codigo funcione e seja capaz de computar e gerar gráficos

In [1]:
```
!pip install matplotlib
import matplotlib

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

In [2]:
```
# permite que se mostre os gráficos nesse ambiente
%matplotlib inline
```

A celula a seguir recebe os dados essenciais da Viga, clique nela e em Run para inserir os valores, que serao solicitados a seguir embaixo da celula.



In [57]:

```
#recebendo dados

viga = float(input("Digite o comprimento (2) da Viga (m):"))

while viga <= 0:
    print("    O  valor de comprimento nao pode ser 0 ou negativo! Tente n
    viga = float(input("Digite o comprimento (2) da Viga (m):"))

else:
    print("    Comprimento da viga registrado!")


Fp = float(input("Digite o valor F2 da carga concentrada (N):"))

while Fp <= 0:
    print("    O  valor de carga nao pode ser 0 ou negativo! Tente novamen
    Fp = float(input("Digite o valor F2 da carga concentrada (N):"))

else:
    print("    Valor da carga concentrada registrado!")


Fpx = float(input("Digite a posicao (1) da carga concentrada (m):"))

while Fpx > viga:
    print("    A posicao da carga concentrada nao pode fora da extensao da
    Fpx = float(input("Digite a posicao (1) da carga concentrada (m):"))

else:
    print("    Posicao da carga concentrada registrada!")

Ft = float(input("Digite o valor maximo F1 da carga distribuida (N):"))
while Ft <= 0:
    print("    O  valor de carga nao pode ser 0 ou negativo! Tente novamen
    Ft = float(input("Digite o valor F1 da carga concentrada (N):"))

else:
    print("    Valor da carga distribuida registrado!")

Ftx = float(input("Digite a posicao final (3) da carga distribuida (m):"))

while Ftx > viga:
    print("    A posicao final da carga distribuida nao pode fora da exten
    Ftx = float(input("Digite a posicao final (3) da carga distribuida (m)

else:
    print("    Posicao da carga distribuida registrada!")
```

```
Digite o comprimento (2) da Viga (m):5.4
    Comprimento da viga registrado!
Digite o valor F2 da carga concentrada (N):2000
    Valor da carga concentrada registrado!
Digite a posicao (1) da carga concentrada (m):5.4
    Posicao da carga concentrada registrada!
Digite o valor maximo F1 da carga distribuida (N):2000
    Valor da carga distribuida registrado!
Digite a posicao final (3) da carga distribuida (m):3
    Posicao da carga distribuida registrada!
```

Com os dados inseridos, é possivel calcular as

reacoes de apoio no engaste.

- Clique na celula abaixo e em Run para obter dados!

In [58]:
```
#calcula reacao do apoio cortante
def ReacaoV(b,h,c):
    return b*h/2 + c

print("A reacao de apoio cortante tem valor(N):", ReacaoV(Ftx, Ft, Fp))

# calcula a reacao de apoio momento
def ReacaoM(a,b,c,d):
    return a*b + c*d

#forca distribuida -> ponto

fmed= (Ft*Ftx)/2
print("O valor equivalente da carga distribuida para um ponto (N):", fmed)

xmed = Ftx/3
print("A posição de aplicação equivalente da carga distribuida (m):", xmed

print("A reacao de apoio de momento tem valor(N*m):", ReacaoM(fmed,xmed,Fp
```

```
A reacao de apoio cortante tem valor(N): 5000.0
O valor equivalente da carga distribuida para um ponto (N): 3000.0
A posição de aplicação equivalente da carga distribuida (m): 1.0
A reacao de apoio de momento tem valor(N*m): 13800.0
```

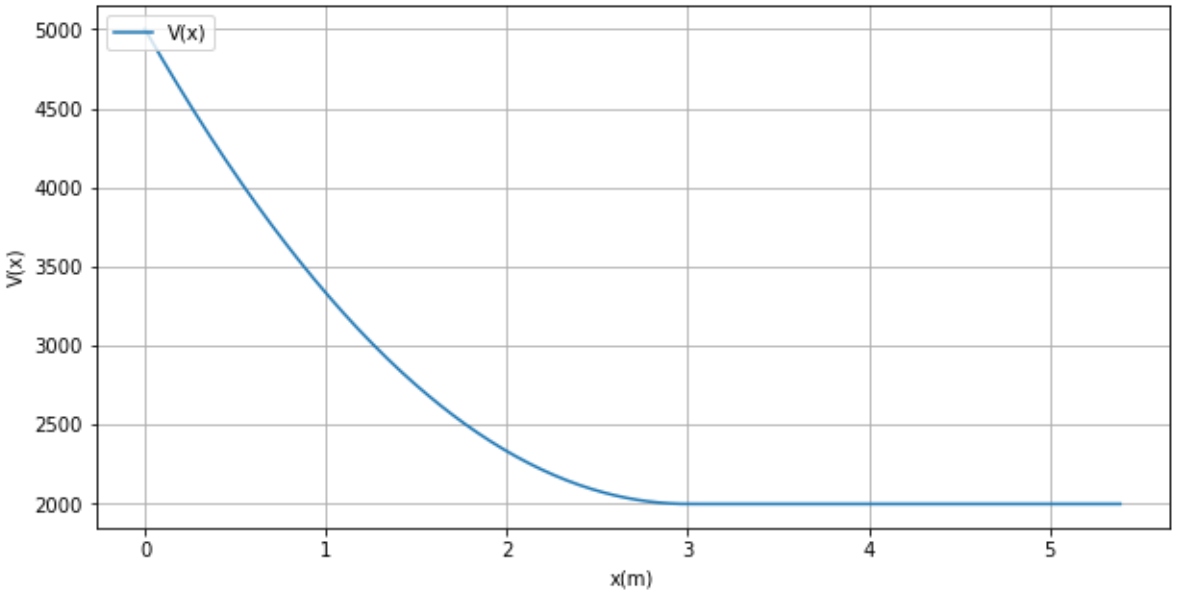
A celula a seguir define, na linguagem de programacao escolhida, como calcular o diagrama de esforco cortante.

In [59]:
```
#diagrama cortante

def W(a):
    return(Ft*a - (Ft*a**2)/(2*Ftx))

def Cortante1(b):
    return ReacaoV(Ftx, Ft, Fp) - W(b)
```

A seguir, com essa funcao, é possivel gerar o grafico!

In [60]:

```
# Fazendo o gráfico da viga
import numpy as np                    # obtém acesso aos rápidos arrays da nu
import matplotlib.pyplot as plt       # Para fazer os gráficos das funções



x0 = 0              # Valor inicial de x
xf = viga       # Valor final de x
dx = 0.01             # Passo da discretização dx = (xf - x0)/(np-1)

x = np.arange(x0, xf, dx)                  # cria um vetor de dados x-data
y = Cortante1(x)                          # calcula o vetor de dados y-data

for i in range (0,len(x)):
   if x[i] > Ftx:
      y[i] = Fp


largura = 10  # Largura da figura
altura  = 5   # Altura da figura
plt.figure(figsize =(largura, altura))     # Define o tamanho do gráfico
plt.grid()                           # Habilita a grade
plt.xlabel('x(m)')
plt.ylabel('V(x)')
plt.title('')
plt.plot(x, y, label="V(x)")   # Faz o gráfico
plt.legend(loc="upper left")        # Habilita as legendas
plt.show()
```



Para ter acesso a um valor especifico, use a celula seguinte.

- Clique em Run quantas vezes quiser.

In [61]:

```
#valor de V(x) em um ponto

valorV=float(input("Escolha um valor de x para saber o V(X): "))

if (valorV > viga) or (valorV < 0):
    while valorV > viga or (valorV < 0):
        print("    A posicao nao pode fora da extensao da viga! Tente novar
        valorV = float(input("Escolha um valor de x para saber o V(X):" ))
    else: print(Cortante1(valorV))
elif (valorV>=Ftx):
    while valorV > viga or (valorV < 0):
        print("    A posicao nao pode fora da extensao da viga! Tente novar
        valorV = float(input("Escolha um valor de x para saber o V(X):" ))
    else: print(Fp)

elif (valorV==0):
    print(Cortante1(0))
```

```
Escolha um valor de x para saber o V(X): 6
    A posicao nao pode fora da extensao da viga! Tente novamente
Escolha um valor de x para saber o V(X):2
2333.333333333333
```

A celula a seguir define, na linguagem de programacao escolhida, como calcular o diagrama de momento.

- Nao é necessario adicionar nenhum dado nem clicar em Run aqui.

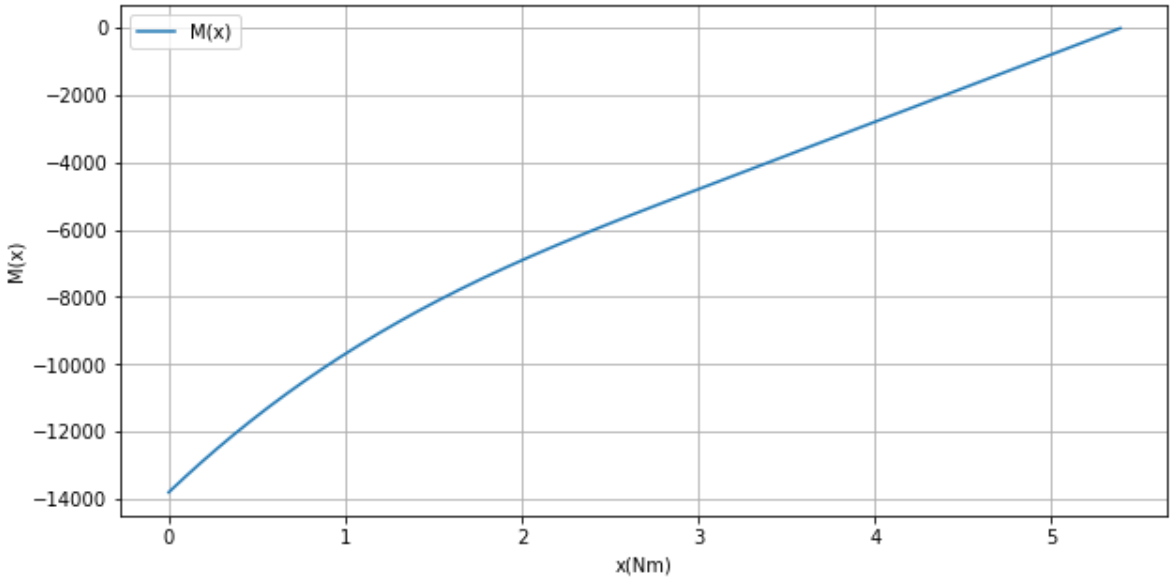
In [62]:

```
#diagrama momento

def M(a):
    v = ReacaoV(Ftx, Ft, Fp)
    m = ReacaoM(fmed,xmed,Fp,Fpx)
    terc= Ft/(6*Ftx)
    seg= -Ft/2
    return ( seg*(a**2) + terc*(a**3) + v*a -m)

def M2(b):
    return -((M(Ftx))/(viga-Ftx))*b + (viga*M(Ftx)/(viga-Ftx))
```

A seguir, com essa funcao, é possivel gerar o grafico!

In [63]:

```
# Fazendo o gráfico da viga
import numpy as np
import matplotlib.pyplot as plt



x0 = 0              # Valor inicial de x
xf = viga            # Valor final de x
dx = 0.01            # Passo da discretização dx = (xf - x0)/(np-1)

x = np.arange(x0, xf, dx)                # cria um vetor de dados x-data
y = M(x)                          # calcula o vetor de dados y-data

for i in range (0,len(x)):
    if x[i] > Ftx:
      y[i] = -((M(Ftx))/(viga-Ftx))*x[i] + (viga*M(Ftx)/(viga-Ftx))


largura = 10  # Largura da figura
altura  = 5   # Altura da figura
plt.figure(figsize =(largura, altura))     # Define o tamanho do gráfico
plt.grid()                          # Habilita a grade
plt.xlabel('x(Nm)')
plt.ylabel('M(x)')
plt.title('')
plt.plot(x, y, label="M(x)")   # Faz o gráfico
plt.legend(loc="upper left")        # Habilita as legendas
plt.show()
```



Para ter acesso a um valor especifico, use a celula seguinte.

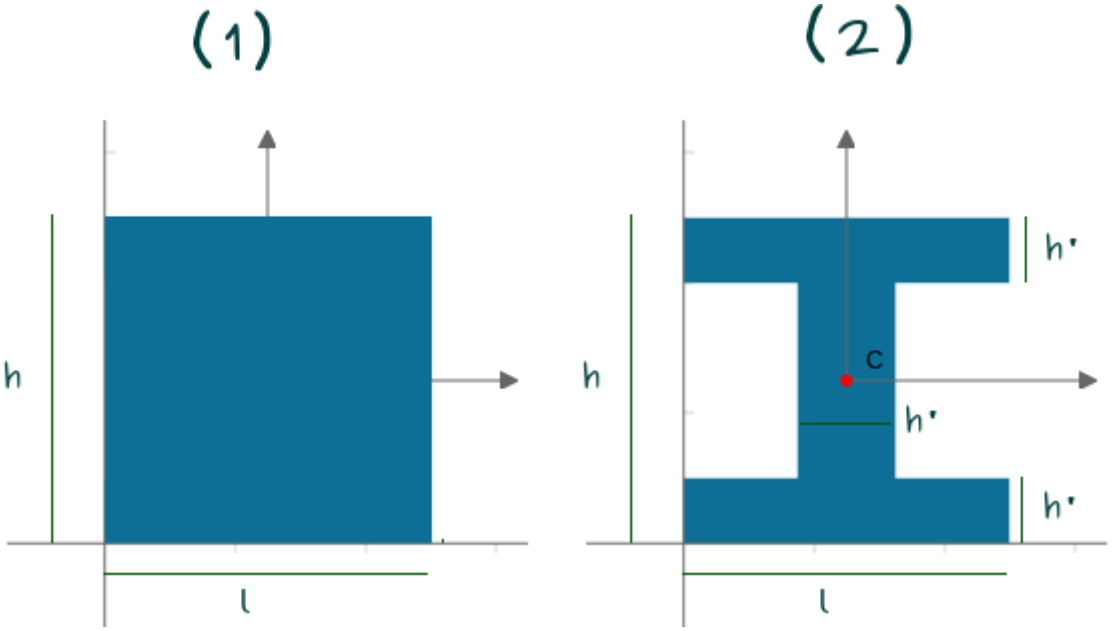
- Clique em Run quantas vezes quiser.

In [64]:

```
#valor de M(x) em um ponto


valorM=float(input("Escolha um valor de x para saber o M(x): "))

if (valorM==viga):
    print("Na borda:", M2(viga))

if (valorM==Ftx):
    print("Na quina:", M(valorM))

if (valorM < Ftx) and (valorM>0):
    print("Dentro da distribuida tenta1:", M(valorM))

if (valorM > Ftx) and (valorM<viga):
    print("Fora da distribuida tenta1:", M2(valorM))

if (valorM > viga) or (valorM <0):
    while valorM > viga or valorM < 0 :
        print("    A posicao nao pode fora da extensao da viga! Tente novar
        valorM = float(input("Escolha um valor de x para saber o M(x):" ))
        if(valorM<Ftx) and (valorM>0):
            print("Dentro da distribluida", M(valorM))
        elif (valorM>Ftx) and (valorM<viga):
            print("Fora da distribuida:", M2(valorM))

elif (valorM==0):
    print("No zero:", M(0))
```

```
Escolha um valor de x para saber o M(x): 0
No zero: -13800.0
```

A celula seguinte serve para escolher a geometria da secao transversal da viga.



- Digite 1 ou 2 a seguir:

In [65]:

```
#entradas para tensao

geo=int(input("Escolha uma configuracao de geometria, digitando 1 ou 2: ")

while geo > 2 :
    geo=int(input("    Escolha invalida, digite 1 ou 2: "))

while geo < 0 :
    geo=int(input("    Escolha invalida, digite 1 ou 2: "))

if (geo == 1):
    print("Geometria retangular da secao transversal foi escolhida!")
elif (geo == 2):
        print("Geometria I da secao transversal foi escolhida!")

#dimensoes

h=float(input("Digite a altura h da secao transversal (cm): "))

while h <= 0 :
    h=float(input("    Escolha invalida, digite um numero positivo nao nul

l=float(input("Digite a largura l da secao transversal (cm): "))

while l <= 0 :
    l=float(input("    Escolha invalida, digite um numero positivo nao nul

if geo==2:
    hlinha=float(input("Digite a altura h' da secao transversal (cm): "))
    while hlinha <= 0 :
        hlinha=float(input("    Escolha invalida, digite um numero positiv
```

```
Escolha uma configuracao de geometria, digitando 1 ou 2: 2
Geometria I da secao transversal foi escolhida!
Digite a altura h da secao transversal (cm): 2
Digite a largura l da secao transversal (cm): 2
Digite a altura h' da secao transversal (cm): 0.4
```

A celula a seguir define as funcoes para as variacoes possiveis de calculo de momento de inercia e tensao normal possiveis, a serem usadas de acordo com a escolha de secao transversal

In [66]:

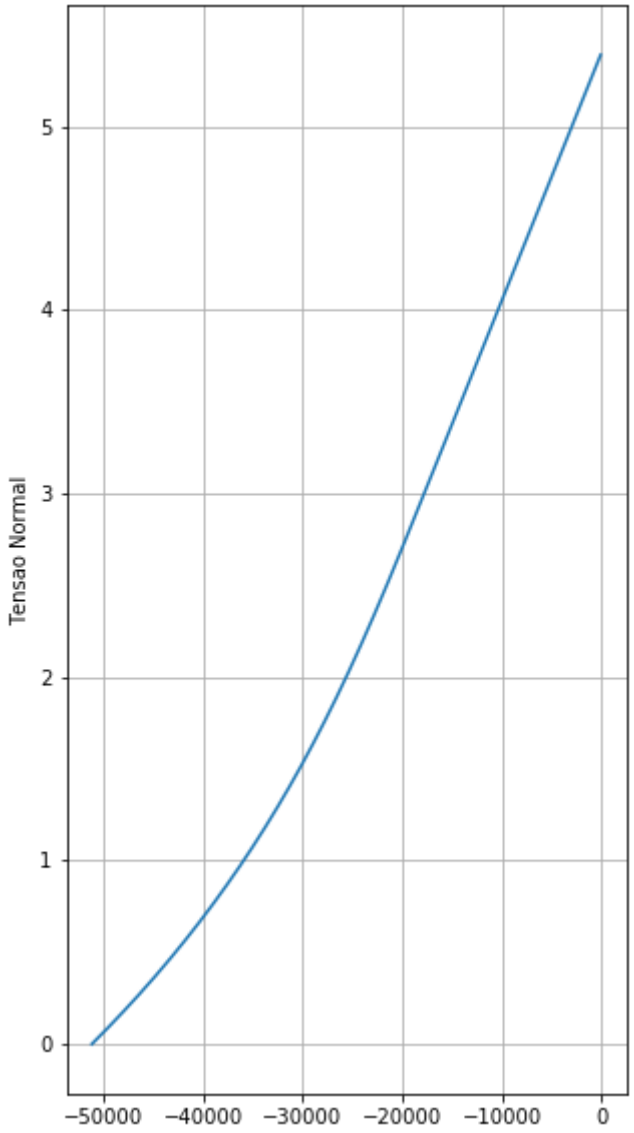
```
#calcular momento de inercia da viga I, nao tenho certeza se preciso em x (

def inerciaCalcIx(var):
    H=var-(2*hlinha)
    b=hlinha
    a=hlinha
    B=l
    Ix= (b*H**3)/12 + ((B*a**3)/6 + (a*B*(H+a)**2)/2)
    Ixx = ((H**3)*hlinha)/12 + 2*(((hlinha**3)*l)/12 + hlinha*l*(H+hlinha)
    return Ixx

def inerciaCalcIy(var):
    H=var-(2*hlinha)
    b=hlinha
    a=hlinha
    B=l
    Iy= (H*b**3)/12 + (a*B**3)/6
    return Iy

if geo==1:
    inercia=(h*l*l*l)/12
    Q=h*l**2*0.5
if geo==2:
    inercia=inerciaCalcIy(h)
    Q=2*inercia/l


def normalFt(a):
    mom=M(a)
    normal=(mom*h)/inercia
    return normal

def normalFp(a):
    mom=M2(a)
    return (mom*h)/inercia

def cortanteFt(a):
    return (Cortante1(a)*Q)/(inercia*l)

def cortanteFp(a):
    return (Fp*Q)/(inercia*l)
```

A seguir, de acordo com a escolha de secao transversal, o codigo recorre as diferentes funcoes descritas.

Finalmente, com as funcoes e escolhas, é possivel gerar o grafico da tensao normal.

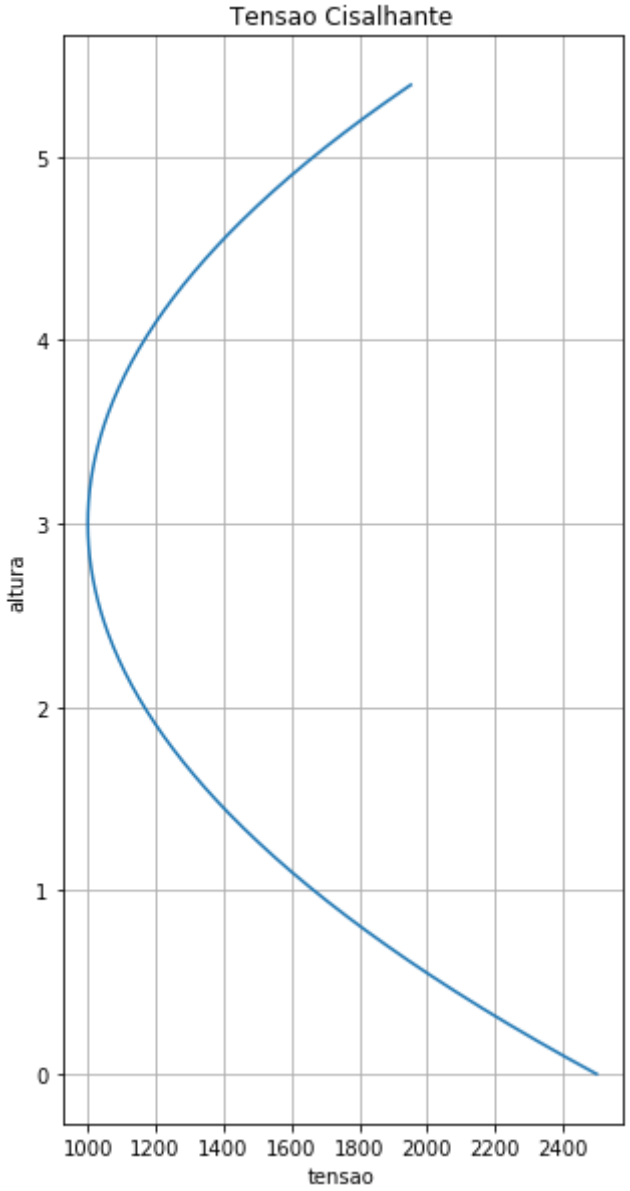
In [67]:

```
# Fazendo o gráfico da viga
import numpy as np
import matplotlib.pyplot as plt


x0 = 0              # Valor inicial de x
xf = viga            # Valor final de x
dx = 0.01            # Passo da discretização dx = (xf - x0)/(np-1)


x = np.arange(x0, xf, dx)                 # cria um vetor de dados x-data
y = normalFt(x)


for i in range (0,len(x)):
    if x[i] > Ftx:
        y[i] = normalFp(x[i])


largura = 5  # Largura da figura
altura  = 10   # Altura da figura
plt.figure(figsize =(largura, altura))     # Define o tamanho do gráfico
plt.grid()                          # Habilita a grade
plt.xlabel('')
plt.ylabel('Tensao Normal')
plt.title('')
plt.plot(y,x)   # Faz o gráfico
plt.show()
```


Tensao Normal
0
1
2
3
4
5
−50000
−40000
−30000
−20000
−10000
0

In [68]:

```
# Fazendo o gráfico da viga
import numpy as np
import matplotlib.pyplot as plt


x0 = 0              # Valor inicial de x
xf = viga            # Valor final de x
dx = 0.01            # Passo da discretização dx = (xf - x0)/(np-1)

x = np.arange(x0, xf, dx)                  # cria um vetor de dados x-data
y = cortanteFt(x)                          # calcula o vetor de dados y-data

for i in range (0,len(x)):
   if x[i] > Ftx:
      y[i] = cortanteFt(x[i])


largura = 5  # Largura da figura
altura  = 10  # Altura da figura
plt.figure(figsize =(largura, altura))     # Define o tamanho do gráfico
plt.grid()                          # Habilita a grade
plt.xlabel('tensao')
plt.ylabel('altura')
plt.title('Tensao Cisalhante')
plt.plot(y, x)   # Faz o gráfico
      # Habilita as legendas

plt.show()
```



Por fim, vamos checar se algum limite é ultrapassado.

Insira os valores a seguir:

In [74]:

```
lim=float(input("Digite o valor limite de cisalhamento: "))

import numpy as np
x0 = 0                # Valor inicial de x
xf = viga              # Valor final de x
dx = 0.1              # Passo da discretização dx = (xf - x0)/(np-1)

x = np.arange(x0, xf, dx)                    # cria um vetor de dados x-data
y = cortanteFt(x)                            # calcula o vetor de dados y-data

for i in range (0,len(x)):
    if y[i] > lim:
        print("O valor limite foi ultrapassado na coordenada de comprimento
```

```
Digite o valor limite de cisalhamento:2000
O valor limite foi ultrapassado na coordenada de comprimento (m): 0.0
```

```
O valor limite foi ultrapassado na coordenada de comprimento (m): 0.1
O valor limite foi ultrapassado na coordenada de comprimento (m): 0.2
O valor limite foi ultrapassado na coordenada de comprimento (m): 0.3000000
0000000004
O valor limite foi ultrapassado na coordenada de comprimento (m): 0.4
O valor limite foi ultrapassado na coordenada de comprimento (m): 0.5
```

In [ ]: