UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

RESISTENCIA DE MATERIALES II

TERCERA PRÁCTICA CALIFICADA

LIMA - PERÚ

NOVIEMBRE 2019

TERCERA PRÁCTICA CALIFICADA

ENTREGADO:

15 NOVIEMBRE 2019

ALUMNO:

Huaroto Villavicencio Josué, 20174070I

PROFESOR:

ING. CUEVA PACHECO RONALD

Índice general

1. Deformada **10**

2. Flecha máxima **11**

3. Ángulo formado por los extremos **12**

4. Mayor esfuerzo **13**

5. Anexos **15**

6. Conclusiones **28**

Cálculo de reacciones

Con $n = 23$.
Hallamos por estática los valores de las reacciones generando el siguiente sistema:

$$R_2 = \frac{5 \cdot (2n^2 + 1135\,n + 110780)}{n + 155}$$
$$R_1 = 7340 - 60\,n - R_2$$

Código 1: Cálculo de reacciones

```
import math
import matplotlib.pyplot as plt
n = 23
R2 = 5*(2*n*n + 1135*n + 110780)/(n+155)
R1 = 7340-60*n-R2
print(R1,R2)
```

De donde obtenemos que:

$$R_1 = 2085.196629213483\,\mathrm{N} \qquad R_2 = 3874.803370786517\,\mathrm{N}$$

Código 2: Cálculo del área

```
area = []
for i in range(0,310+2*n+1):
    if i<10:
        area.append(math.pi*(40/2))
        continue
    if i<80:
        area.append((math.pi*(50/2)))
        continue
    if i<200+2*n:
        area.append((math.pi*(((50+((i-80)*20/(120+2*n)))/2))))
        continue
    if i<260+2*n:
        area.append((math.pi*(70/2)))
        continue
    if i<300+2*n:
        area.append(math.pi*(60/2))
        continue
    area.append((math.pi*(55/2)))
```

Fuerza cortante

Usando las funciones de Macaulay obtenemos la forma reducida de la expresión del cortante:

$$V = R_1 < x >^0 -(12-n) < x-10 >^1 +(12-n) < x-80 >^1 -(5000+10n) < x-(130+2n) >^0 +$$

$$(60000+200n) < x-(130+2n) >^{-1} -\frac{5/6}{2} < x-(200+2n) >^2 +50 < x-(260+2n) >^1 +$$

$$\frac{5/6}{2} < x-(260+2n) >^2 +R_2 < x-(310+2n) >^0$$

Procedemos a hallar los valores de V para cada punto y observar la gráfica formada.

Código 3: Cálculo del cortante

```
v = []
v.append(0)
for x in range(1,310+2*n+1):
    aux = R1
    if x>10:
        aux = aux-(12-n)*(x-10)
    if x>130+2*n:
        aux = aux-(5000+10*n)
    if x>80:
        aux = aux + (12-n)*(x-80)
    if x>260+2*n:
        aux = aux+(50)*(x-(260+2*n)) + (5/12)*(x-(260+2*n))*(x-(260+2*n))
    if x>200+2*n:
        aux = aux-(5/12)*(x - (200+2*n))*(x - (200+2*n))
    if x>=310+2*n:
        aux = aux + R2
    v.append(aux)
plt.figure(figsize=(10,5))
plt.xlabel('Longitud (mm)')
plt.ylabel('Fuerza cortante (N)')
plt.grid()
plt.plot(v)
plt.savefig('Fuerzacortante.pdf')
```

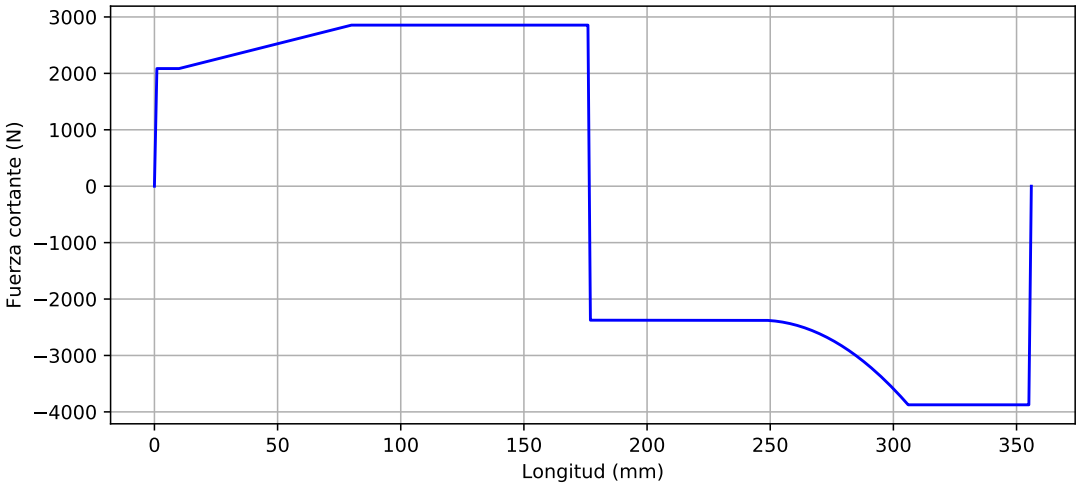


Figura 1: Fuerza cortante

Momento flector

$$M = R_1 < x >^1 - \frac{12-n}{2} < x-10 >^2 + \frac{12-n}{2} < x-80 >^2 - (5000+10n) < x-(130+2n) >^1 +$$

$$(60000+200n) < x-(130+2n) >^0 - \frac{5/6}{6} < x-(200+2n) >^3 + \frac{50}{2} < x-(260+2n) >^2 +$$

$$\frac{5/6}{6} < x-(260+2n) >^3 + R_2 < x-(310+2n) >^1$$

De la misma forma, hallamos los valores para el momento flector

Código 4: Cálculo del momento flector

```
m = []
for x in range(0,310+2*n+1):
    aux = R1*x
    if x>10:
        aux = aux-(12-n)*(x-10)*(x-10)/2
    if x>130+2*n:
        aux = aux-(5000+10*n)*(x-(130+2*n))
    if x>80:
        aux = aux + (12-n)*(x-80)*(x-80)/2
    if x>260+2*n:
        aux = aux+(25)*(x-(260+2*n))**2 + (5/36)*(x-(260+2*n))**3
    if x > 130+2*n:
        aux = aux+(60000+200*n)
    if x>200+2*n:
        aux = aux-(5/36)*(x - (200+2*n))*(x - (200+2*n))*(x - (200+2*n))
    if x>=310+2*n:
        aux = aux + R2*(x-(310+2*n))
    m.append(aux)
plt.figure(figsize=(10,5))
plt.grid()
plt.xlabel('Longitud (mm)')
plt.ylabel('Momento flector (N-mm)')
plt.plot(m)
plt.savefig('momento.pdf')
```

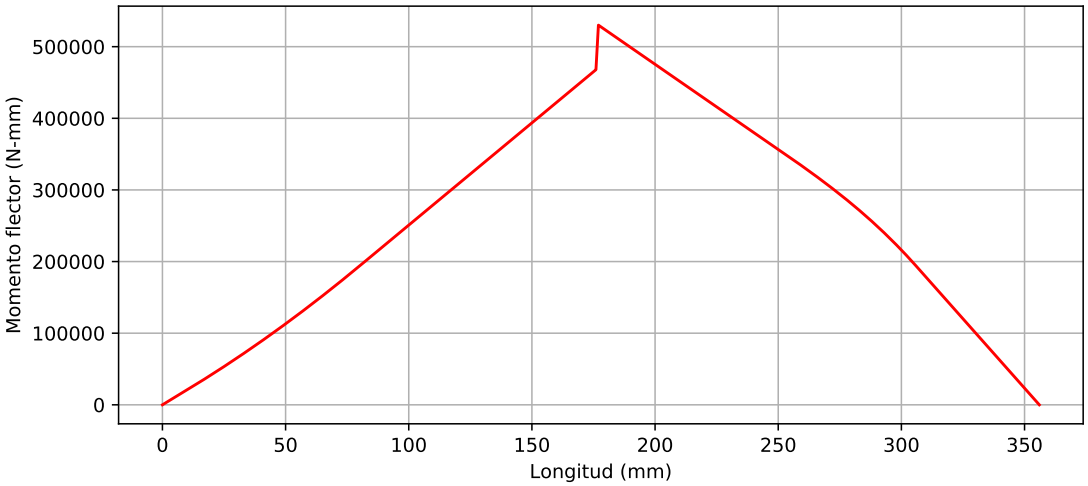


Figura 2: Momento flector

Momento de inercia

Código 5: Cálculo del momento flector

```
I = []
for i in range(0,310+2*n+1):
    if i<10:
        I.append((math.pi*(40)**4)/64)
        continue
    if i<80:
        I.append((math.pi*(50)**4)/64)
        continue
    if i<200+2*n:
        I.append((math.pi*(((50+((i-80)*20/(120+2*n))))**4)/64))
        continue
    if i<260+2*n:
        I.append((math.pi*(70)**4)/64)
        continue
    if i<300+2*n:
        I.append((math.pi*(60)**4)/64)
        continue
    I.append((math.pi*(55)**4)/64)
plt.figure(figsize=(15,5))
plt.grid()
plt.xlabel('Longitud (mm)')
plt.ylabel('Momento de inercia (mm$^4$)')
plt.plot(I,'-m')
plt.savefig('momentoi.pdf')
```

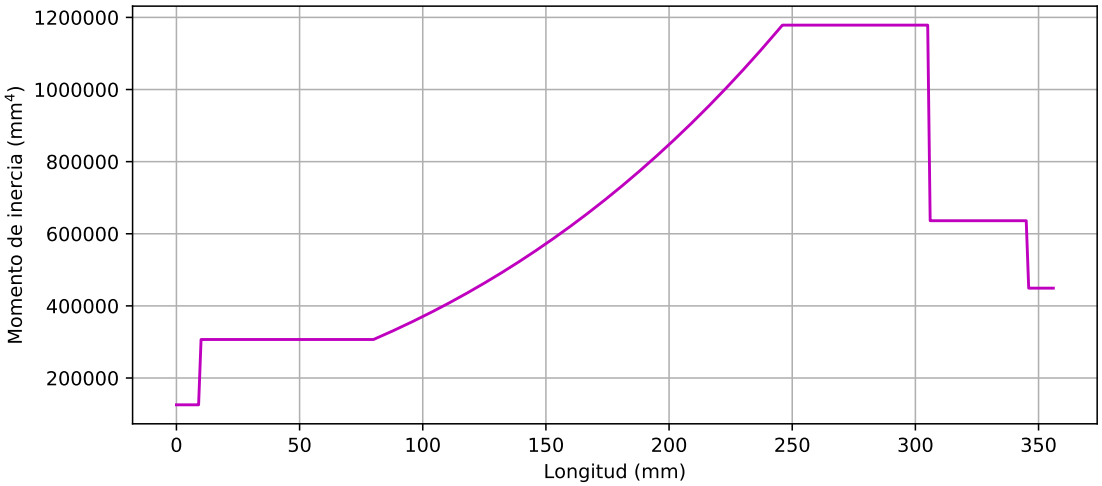


Figura 3: Momento de inercia

Código 6: Cálculo del momento flector

```
E = 2.1*10**5
MEI = []
for i in range(0,310+2*n+1):
    MEI.append(m[i]/(E*I[i]))
plt.figure(figsize=(15,5))
plt.grid()
plt.xlabel('Longitud (mm)')
plt.ylabel('M/EI (mm)')
plt.plot(MEI,'-b')
plt.savefig('mei.pdf')
```

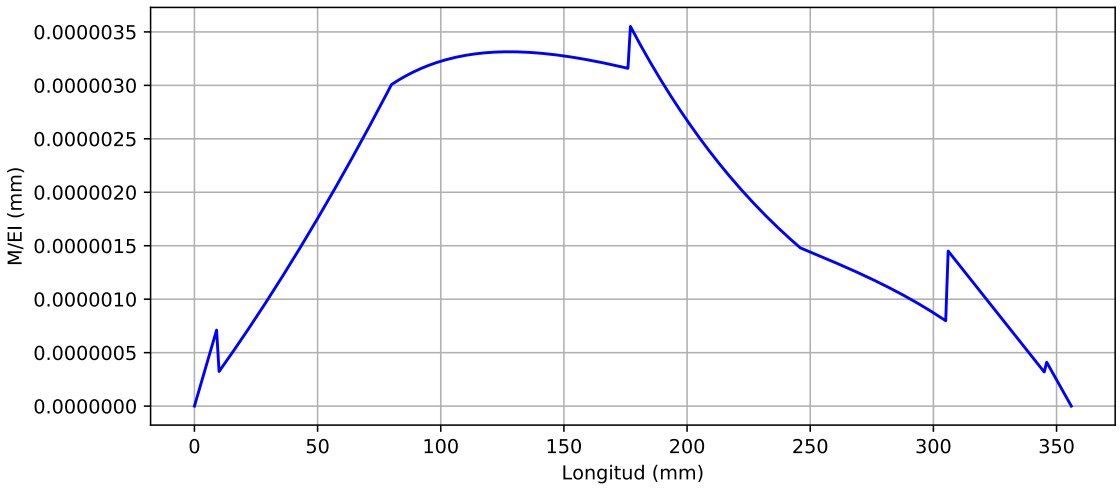


Figura 4: M/EI

Cálculo de la deflexión

Teniendo los valores de M, entonces pasamos I dividiendo e integramos:

$$\int M/I \, \mathrm{d}x = ??$$

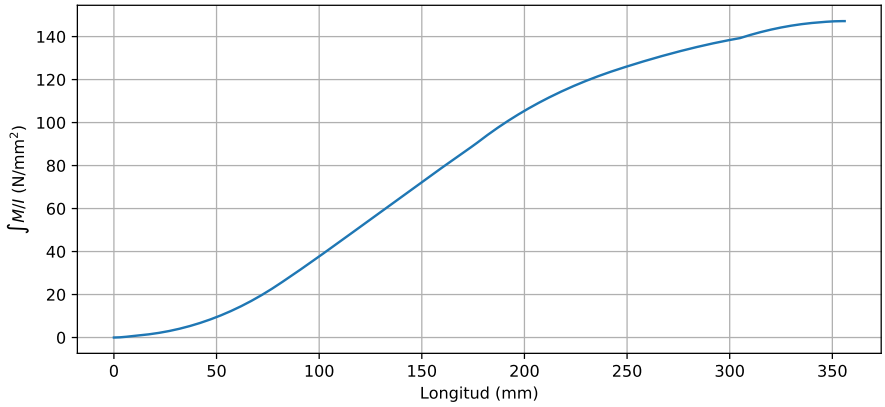
Sabemos sin embargo, que si la función M/I es continua e integrable, entonces podemos hallar su integral definida:

$$\int_0^{310+2n} M/I \, \mathrm{d}x = \left[\int M/I \, \mathrm{d}x\right]_{310+2n} - 0$$

Debido a que es complicado hallar la integral de M/I, optamos por hallar la integral definida para cada punto entre 0 y 310+2n de manera numérica, la cual es fácil de hallar. Recordando que los valores obtenidos corresponden a la de la integral $\int M/I \, \mathrm{d}x$

Código 7: Primera integral de M/I

```
mi = []
mi.append(0)
for i in range(1,310+2*n+1):
    mi.append(m[i]/I[i] + mi[i-1])
plt.figure(figsize=(10,5))
plt.xlabel('Longitud (mm)')
plt.ylabel('$\int M/I$ (N/mm$^{2}$)')
plt.grid()
plt.plot(mi)
plt.savefig('mintegr.pdf')
```



La gráfica obtenida corresponde cuando la constante de integración es 0.
Volvemos a integrar para hallar la deflexión del eje, de la misma forma, debido a que tenemos los

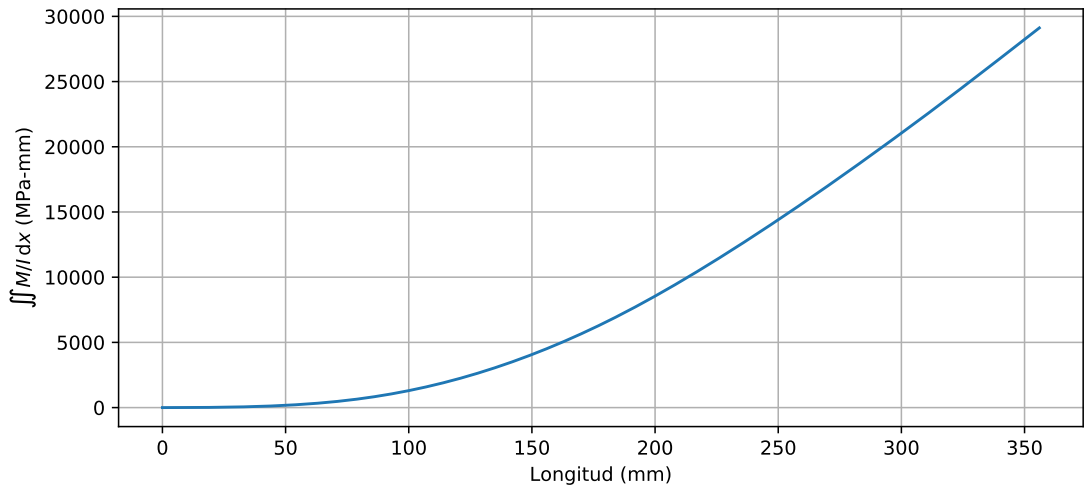
valores de la primera integral de forma numérica. La gráfica resultante debe corresponde a

$$\left[\iint M/I \, \mathrm{d}x\right]_{310+2n}$$

Con constante de integración 0 para las dos constantes.

Código 8: Segunda integral de M/I

```
mii = []
mii.append(0)
for i in range(1,310+2*n+1):
    mii.append(mi[i]+mi[i-1])
plt.figure(figsize=(10,5))
plt.grid()
plt.xlabel('Longitud (mm)')
plt.ylabel('$ \iint M/I\,\mathrm{d}x $ (MPa-mm)')
plt.plot(mii)
plt.savefig('m2i.pdf')
```



Capítulo 1

Deformada

Para corregir el valor de las constantes de integración hallamos los valores que corresponden según las condiciones de frontera y dividimos entre E para hallar la deformada.

Código 1.1: Deformada

```
const = -mii[310+2*n]/(310+2*n)
for i in range(0,310+2*n+1):
    mii[i] = -(mii[i]+const*i)/E
plt.figure(figsize=(10,5))
plt.grid()
plt.xlabel('Longitud (mm)')
plt.ylabel('Deformada (mm)')
plt.plot(mii)
plt.savefig('def.pdf')
```

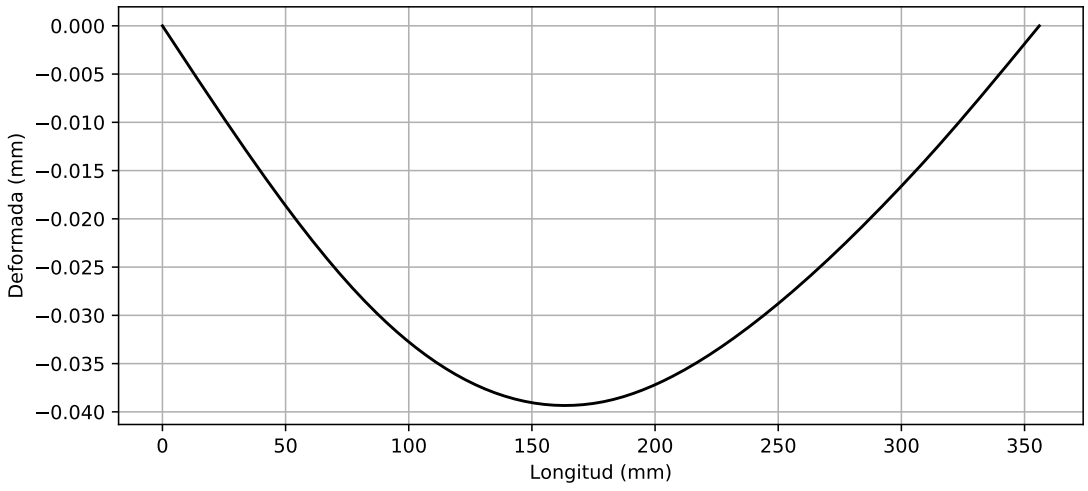


Figura 1.1: Gráfica de la deformada

Capítulo 2

Flecha máxima

Código 2.1: Cálculo de la flecha

```
flechamax = 0
posflecha = 0
for i in range(0,310+2*n+1):
    flechamax = max(flechamax,abs(mii[i]))
for i in range(0,310+2*n+1):
    if(abs(mii[i]) == flechamax):
        posflecha = i
print(flechamax, posflecha)
```

Obtenemos que el máximo valor de la flecha es $\boxed{0.03934077\,\text{mm}}$ y sucede a $\boxed{163\,\text{mm}}$ del extremo izquierdo.

Capítulo 3

Ángulo formado por los extremos

Realizamos una regresión lineal para hallar la ecuación que corresponde a una recta tangente al extremo izquierdo y el extremo derecho:

Ext. izquierdo$(x) = -4.0824 \times 10^{-4}\, x - 2.4 \times 10^{-7}$ Ext. derecho$(x) = 3.12 \times 10^{-4} x - 1.1110683 \times 10^{-1}$

Dado que las pendientes son muy pequeñas, se puede aproximar y considerar que $\tan(x) = x$:

$$\theta = \pi/2 - (4.0824 - 3.12) \times 10^{-4} = 1.57070008 = \boxed{89.9944858^\circ}$$

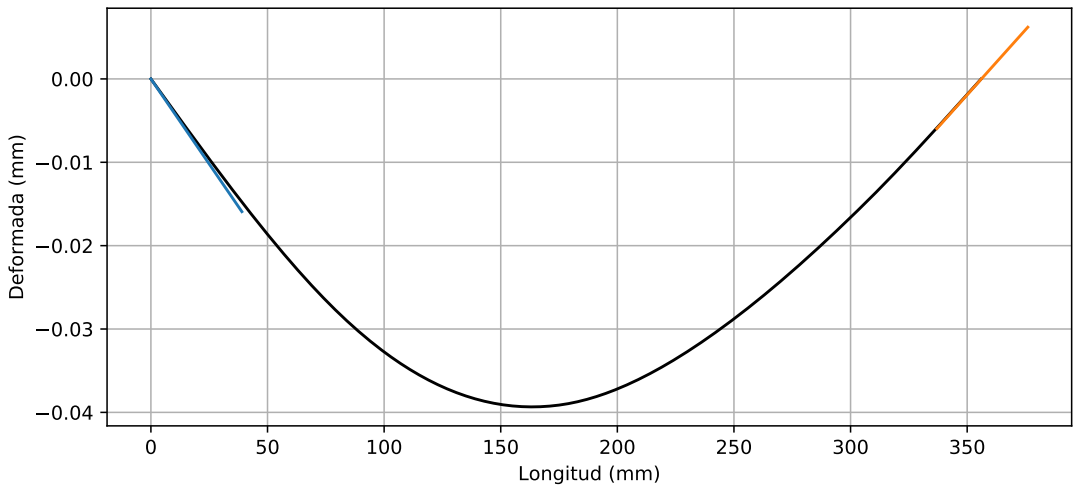


Figura 3.1: Rectas tangentes

Capítulo 4

Mayor esfuerzo

Código 4.1: Cálculo del esfuerzo por flexión

```
esfm = 0
poses = 0
for i in range(0,310+2*n+1):
    if abs(math.sqrt(area[i]/math.pi)*m[i]/I[i]) > esfm:
        esfm = abs(math.sqrt(area[i]/math.pi)*m[i]/I[i])
        poses = i
print(esfm, poses)
```

El esfuerzo máximo por flexión ocurre a 177 mm del extremo izquierdo y es de 4.1418371 MPa.

Código 4.2: Cálculo del esfuerzo máximo

```
esfm = 0
poses = 0
gresf = []
for i in range(0,310+2*n+1):
    esff = abs(math.sqrt(area[i]/math.pi)*m[i]/I[i])
    gresf.append(math.sqrt(esff**2 + (v[i]/area[i])**2))
    if math.sqrt(esff**2 + (v[i]/area[i])**2) > esfm:
        esfm = math.sqrt(esff**2 + (v[i]/area[i])**2)
        poses = i
print(esfm, poses)
plt.figure(figsize=(10,5))
plt.grid()
plt.xlabel('Longitud (mm)')
plt.ylabel('Esfuerzo (MPa)')
plt.plot(gresf)
plt.savefig('esfuerzo.pdf')
```

El mayor esfuerzo, considerando la flexión y el de corte es de $\boxed{44.852762\,\text{MPa}}$ y ocurre a $\boxed{346\,\text{mm}}$.

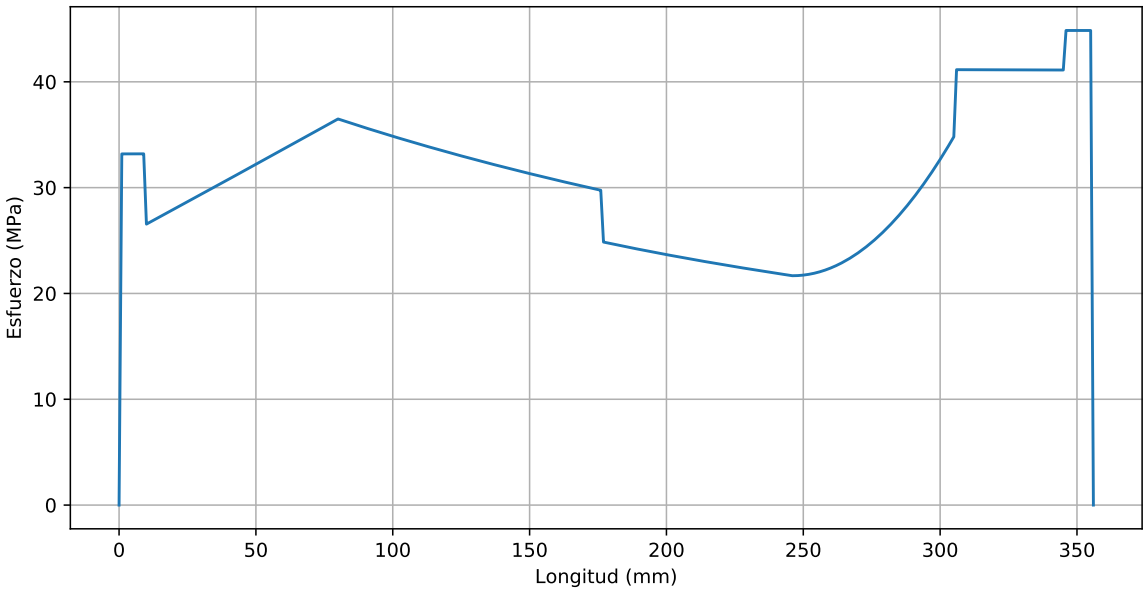


Figura 4.1: Esfuerzo

Capítulo 5

Anexos

Como el problema ha sido programado según el valor de n. Podemos hallar el gráfico de la deformada para varios valores de n de forma muy rápida, se anexa entonces varios gráficos de la deformada con distintos valores de n:

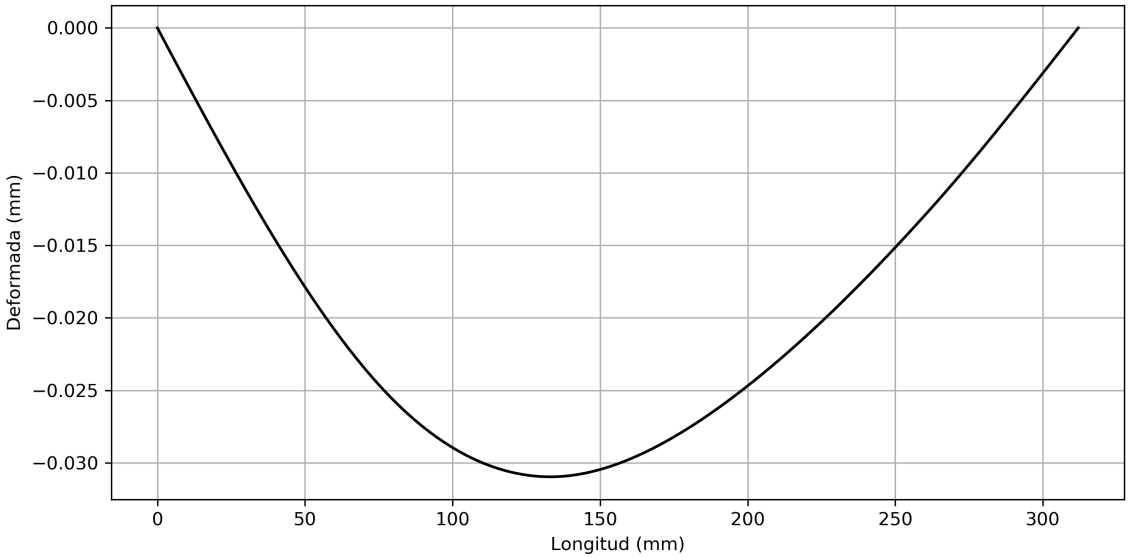


Figura 5.1: $n = 1$

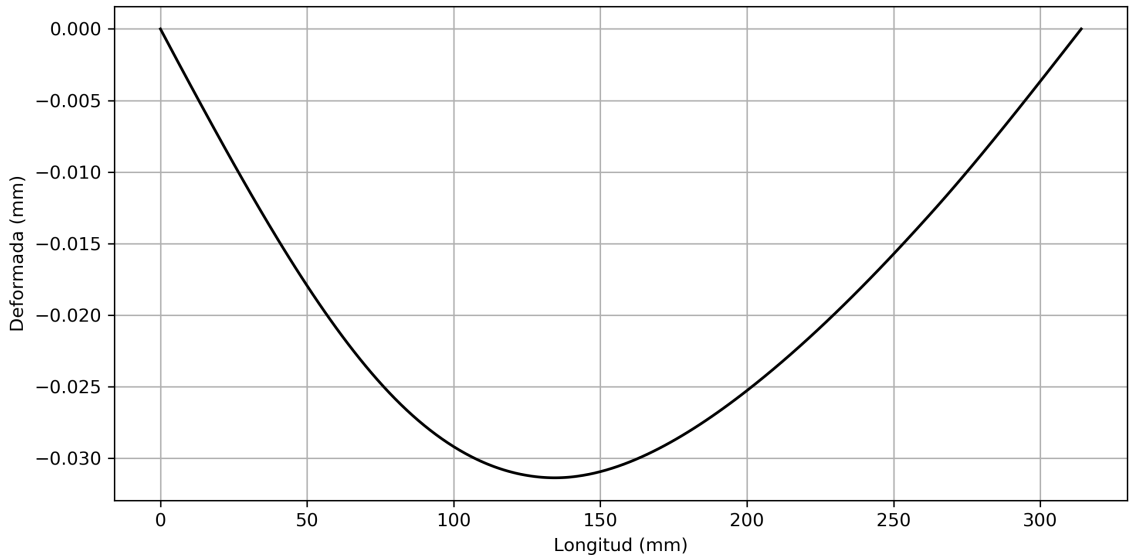


Figura 5.2: $n = 2$

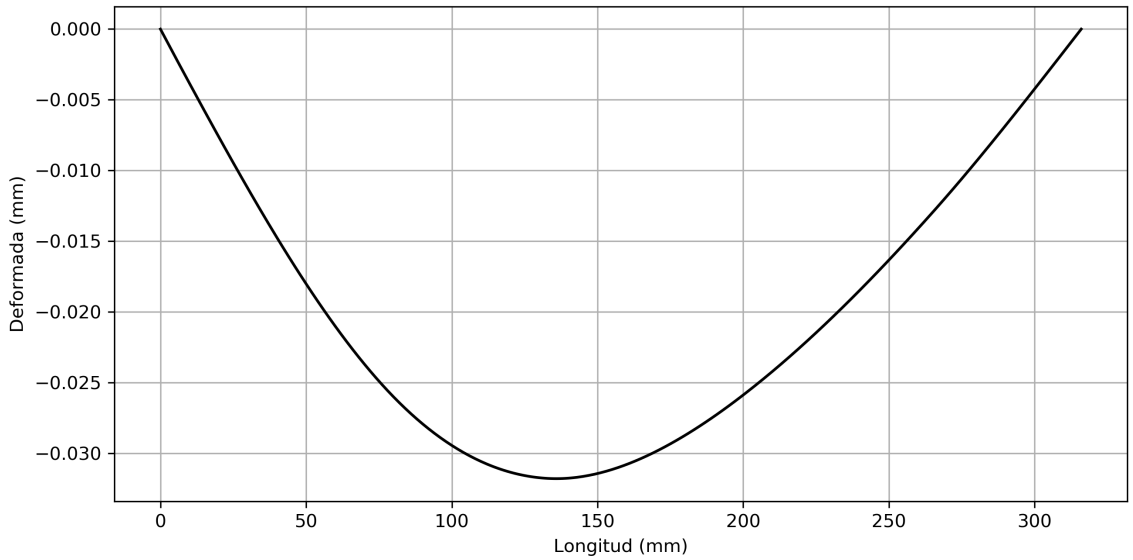


Figura 5.3: $n = 3$

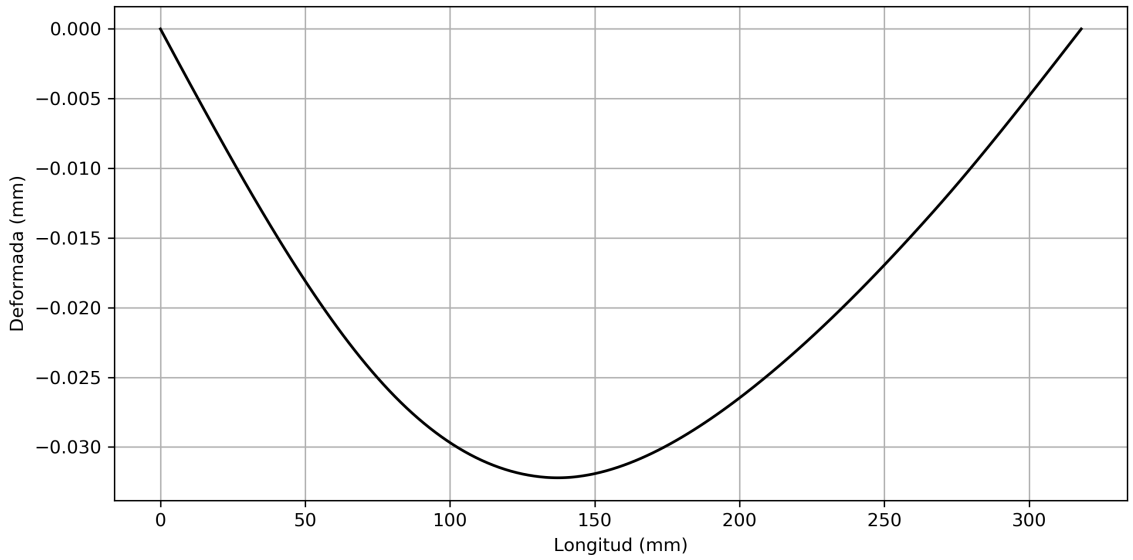


Figura 5.4: $n = 4$

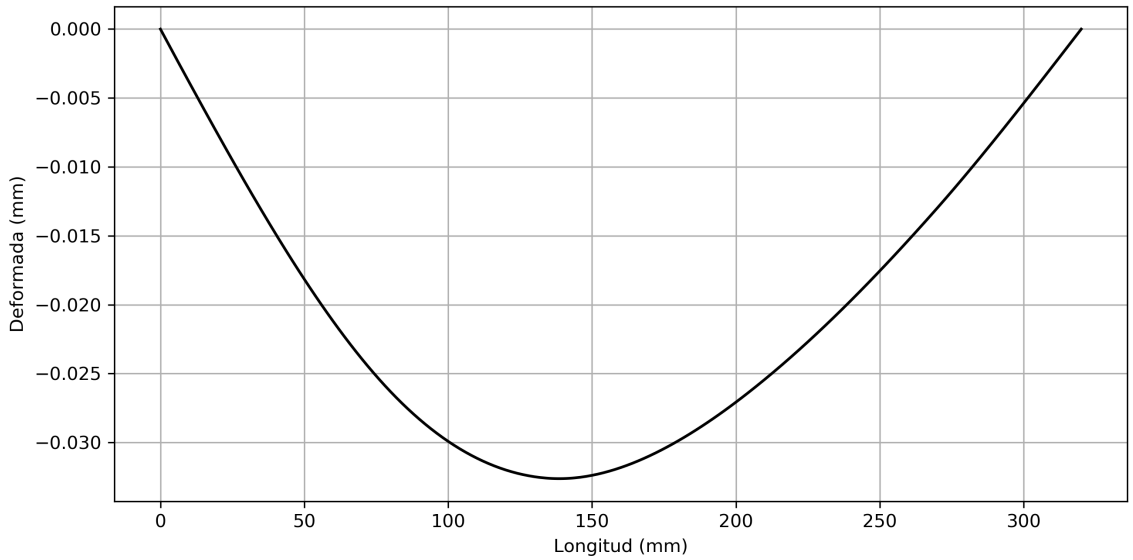


Figura 5.5: $n = 5$

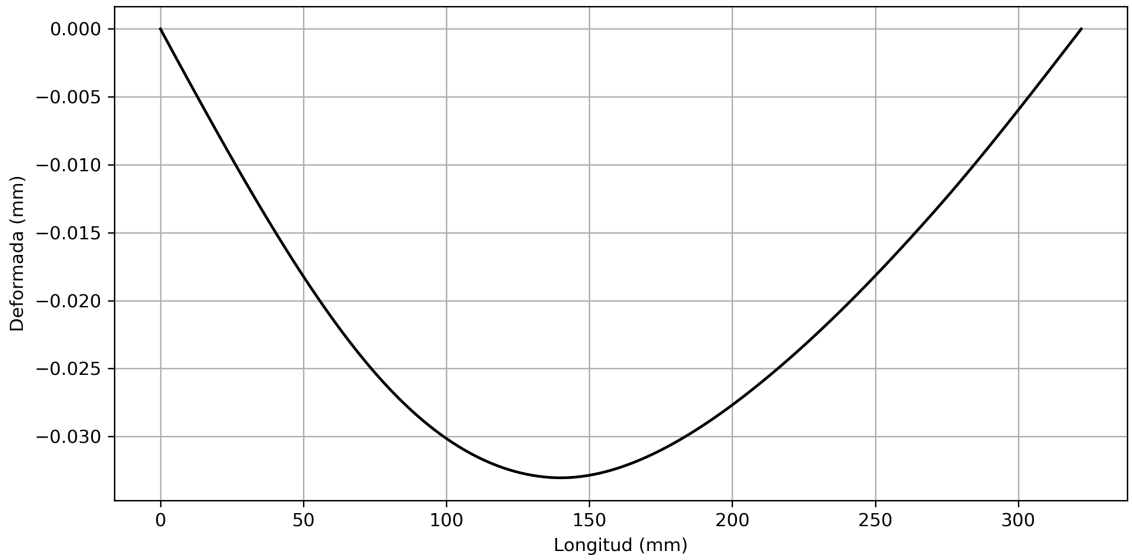


Figura 5.6: $n = 6$

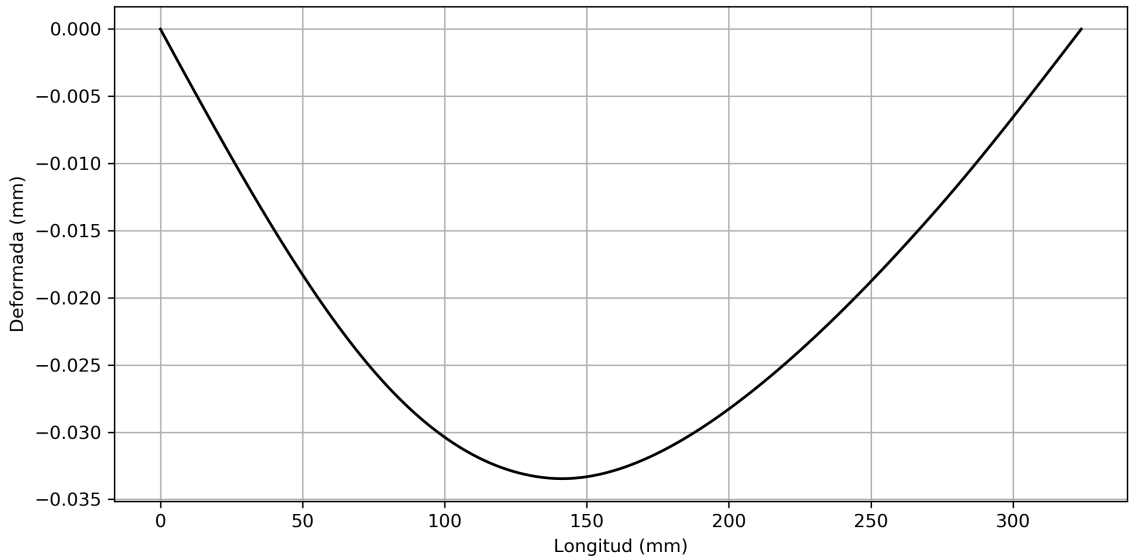


Figura 5.7: $n = 7$

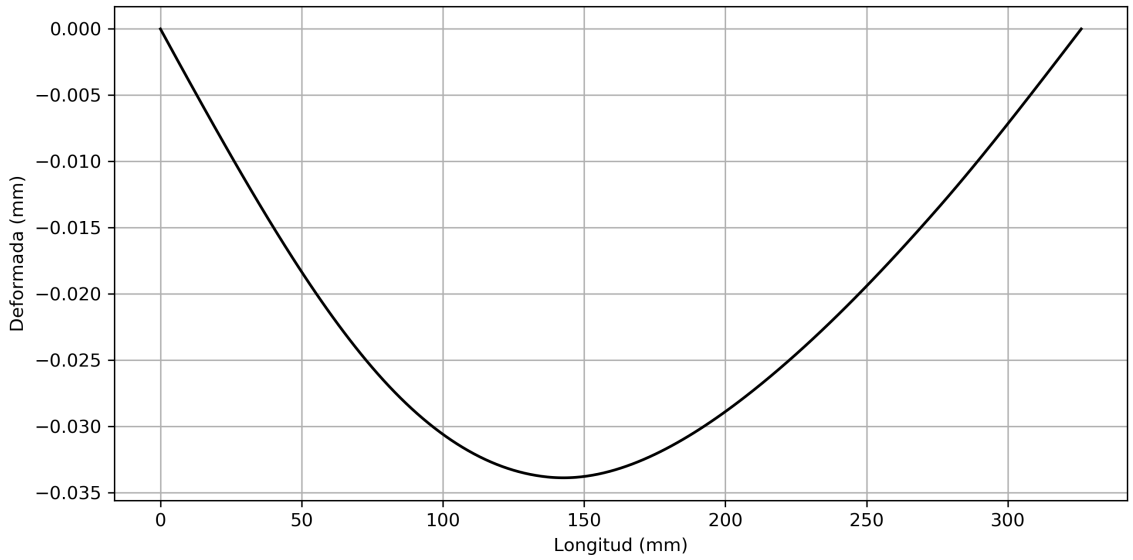


Figura 5.8: $n = 8$

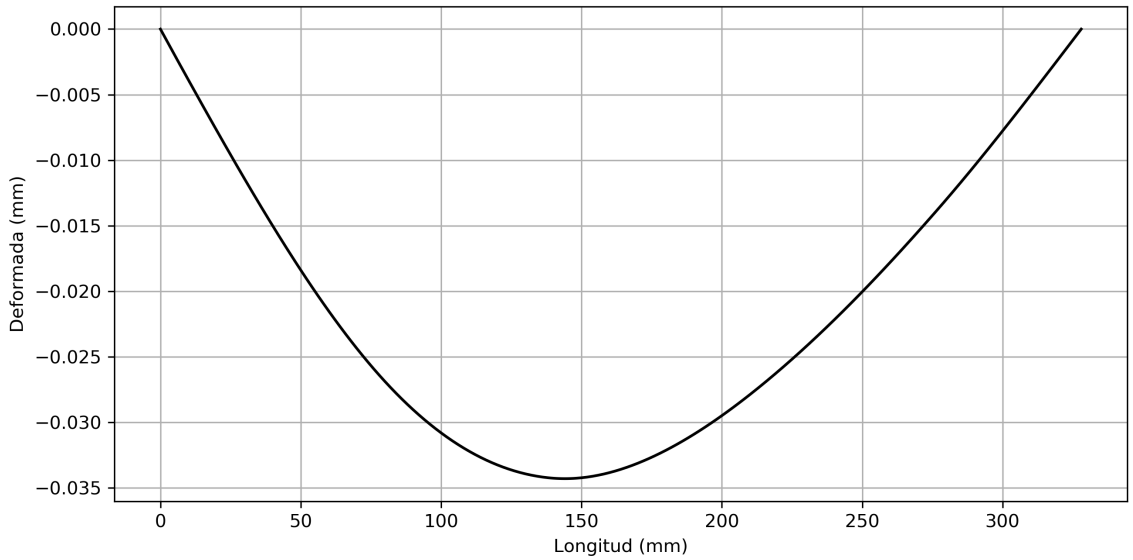


Figura 5.9: $n = 9$

Así mismo, podemos tomar valores muy grandes para n:

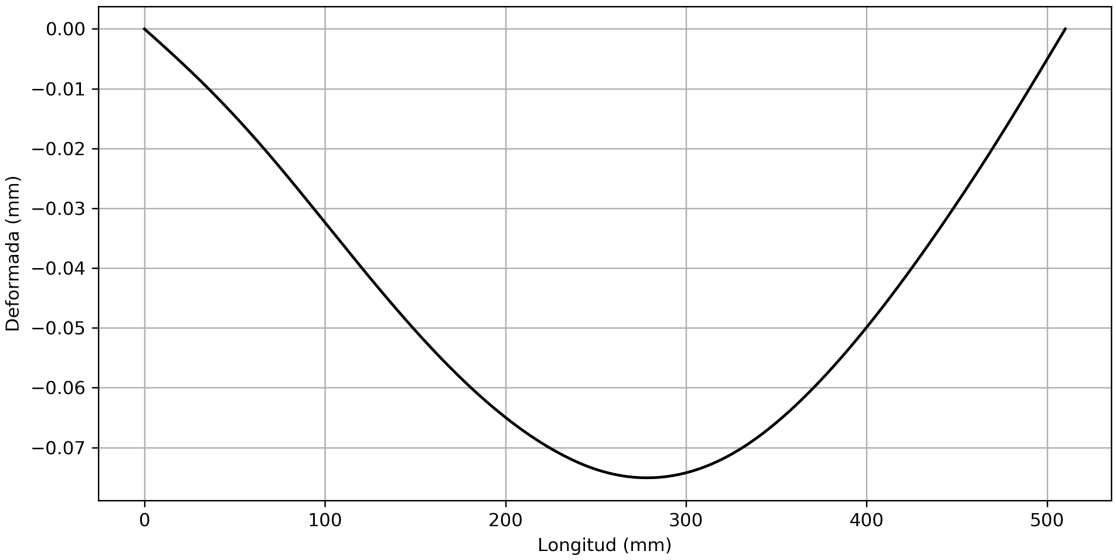


Figura 5.10: $n = 100$

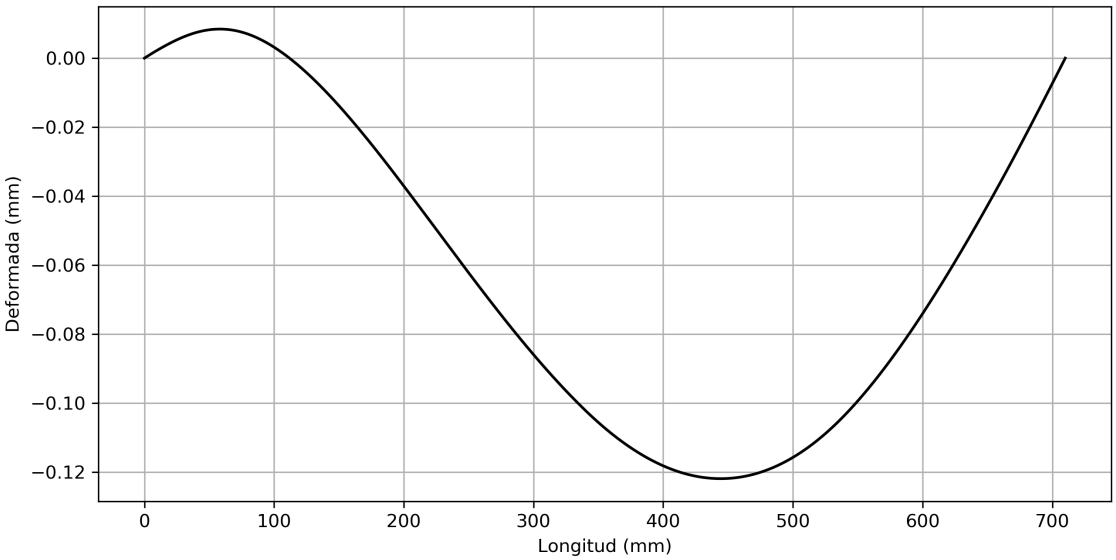


Figura 5.11: $n = 200$

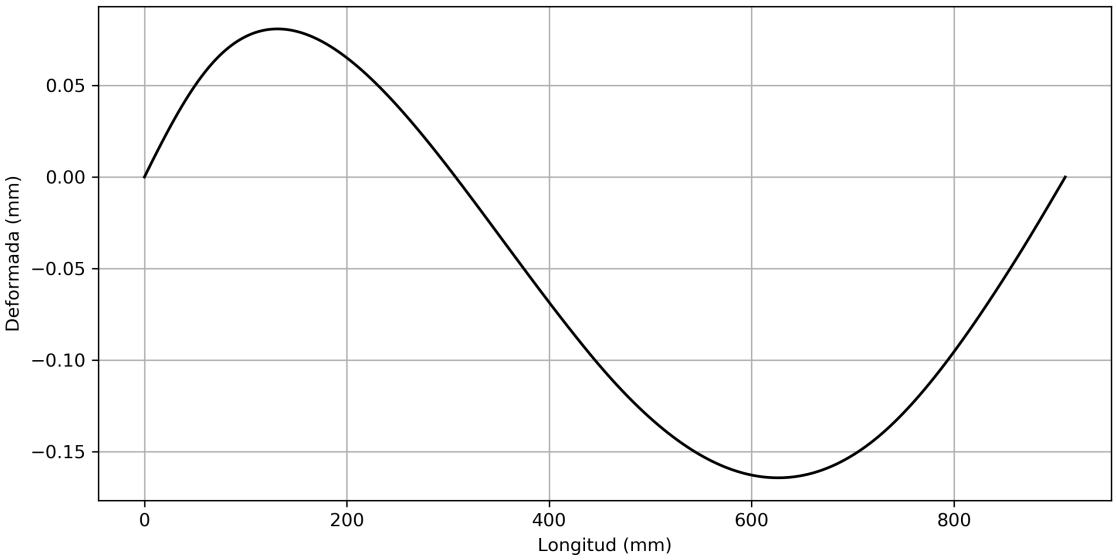


Figura 5.12: $n = 300$

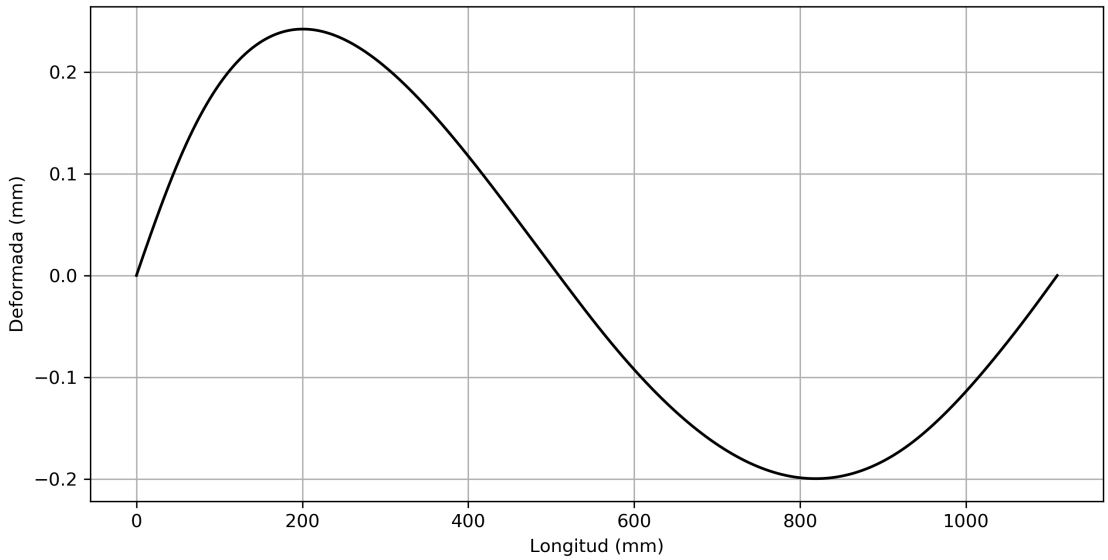


Figura 5.13: $n = 400$

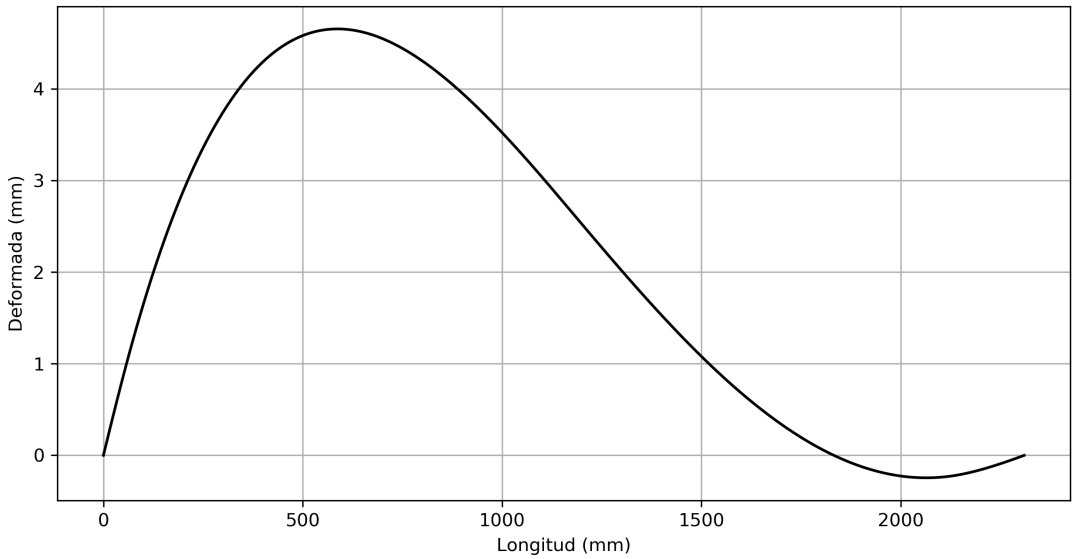


Figura 5.14: $n = 1000$

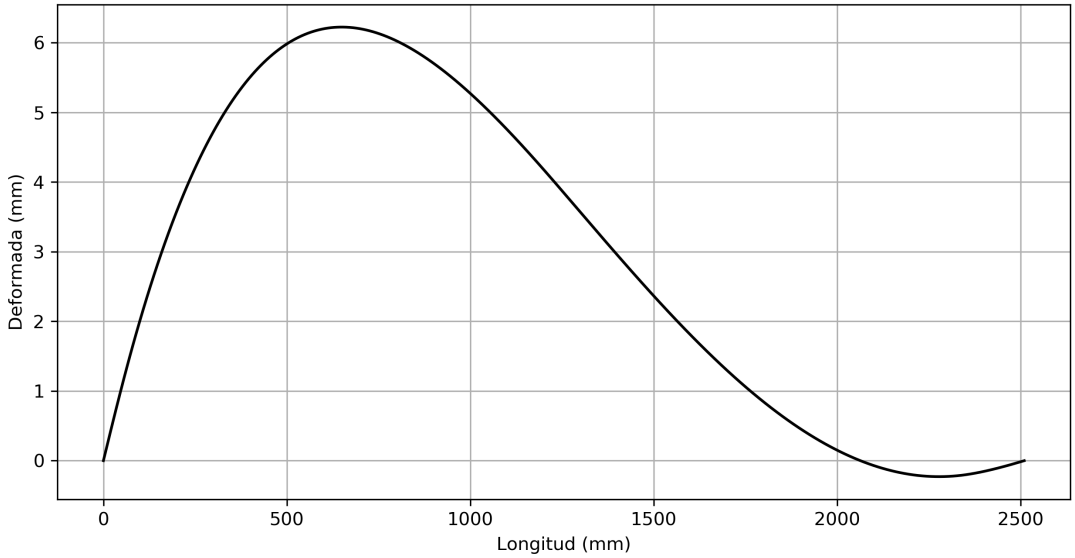


Figura 5.15: $n = 1100$

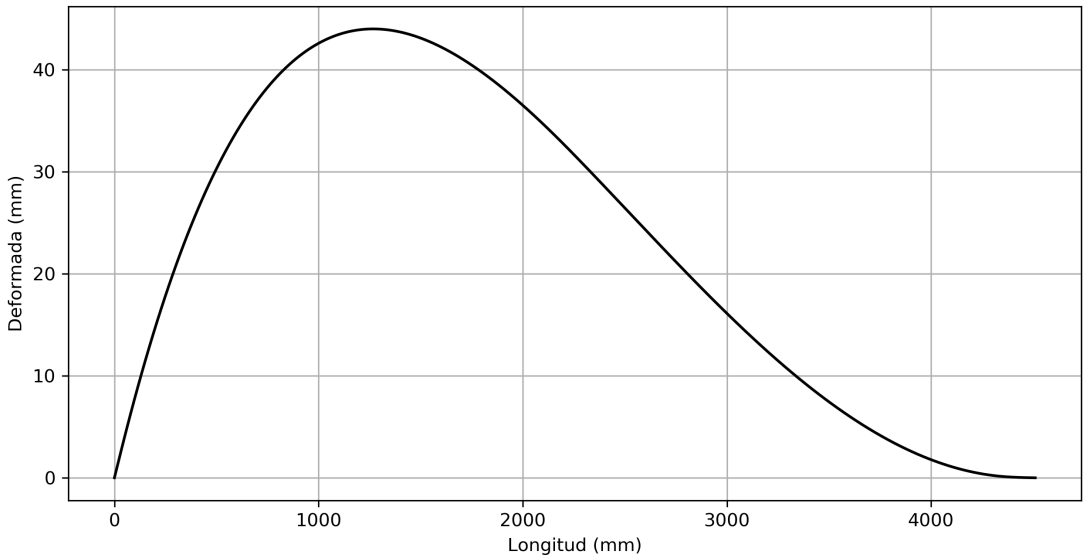


Figura 5.16: $n = 2100$

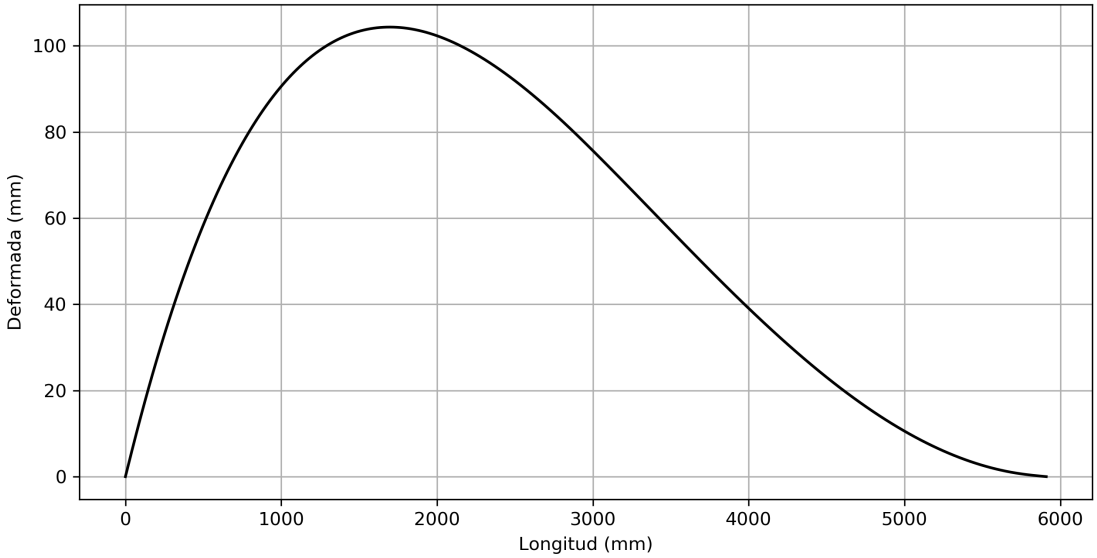


Figura 5.17: $n = 2800$

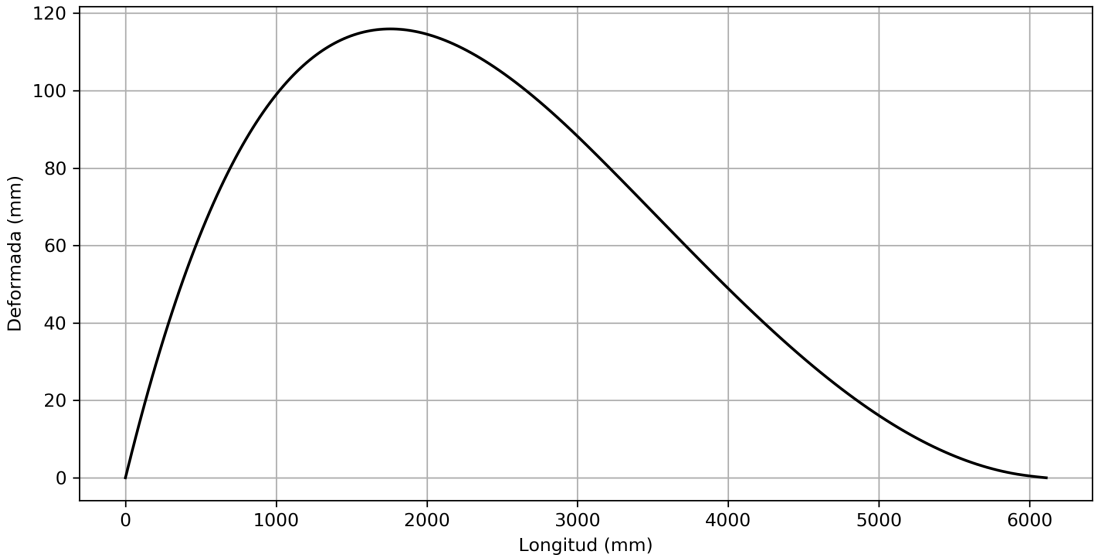


Figura 5.18: $n = 2900$

El código utilizado se muestra a continuación:

Código 5.1: Cálculo de la flecha

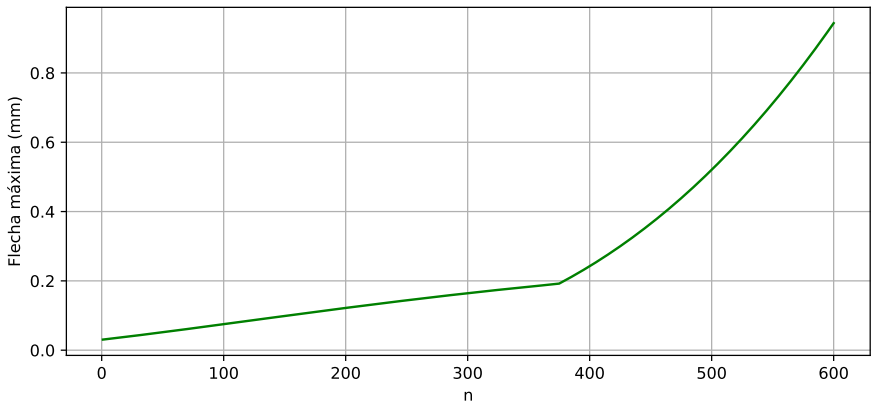
```
for ka in range(1,30):
    n = ka
    R2 = 5*(2*n*n + 1135*n + 110780)/(n+155)
    R1 = 7340-60*n-R2
    v = []
    v.append(0)
    for x in range(1,310+2*n+1):
        aux = R1
        if x>10:
            aux = aux-(12-n)*(x-10)
        if x>130+2*n:
            aux = aux-(5000+10*n)
        if x>80:
            aux = aux + (12-n)*(x-80)
        if x>260+2*n:
            aux = aux+(50)*(x-(260+2*n)) + (5/12)*(x-(260+2*n))*(x-(260+2*n))
        if x>200+2*n:
            aux = aux-(5/12)*(x - (200+2*n))*(x - (200+2*n))
        if x>=310+2*n:
            aux = aux + R2
        v.append(aux)
    m = []
    for x in range(0,310+2*n+1):
        aux = R1*x
        if x>10:
            aux = aux-(12-n)*(x-10)*(x-10)/2
        if x>130+2*n:
```

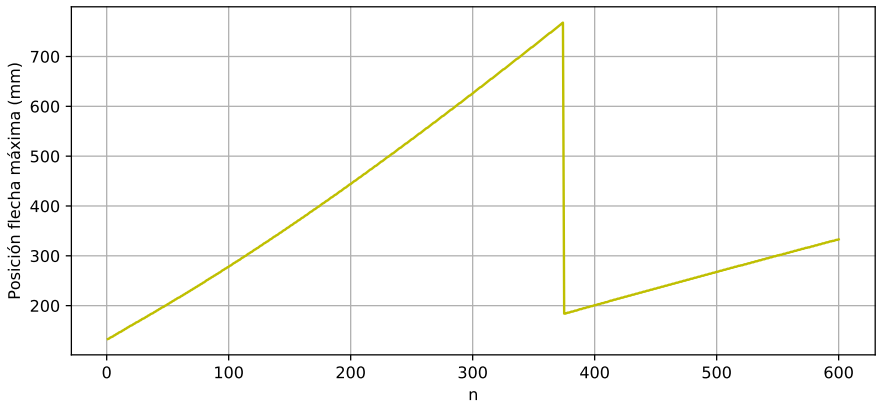
```
            aux = aux-(5000+10*n)*(x-(130+2*n))
        if x>80:
            aux = aux + (12-n)*(x-80)*(x-80)/2
        if x>260+2*n:
            aux = aux+(25)*(x-(260+2*n))*(x-(260+2*n)) + (5/36)*(x-(260+2*n))**3
        if x > 130+2*n:
            aux = aux+(60000+200*n)
        if x>200+2*n:
            aux = aux-(5/36)*(x - (200+2*n))*(x - (200+2*n))*(x - (200+2*n))
        if x>=310+2*n:
            aux = aux + R2*(x-(310+2*n))
        m.append(aux)
    I = []
    for i in range(0,310+2*n+1):
        if i<10:
            I.append((math.pi*(40)**4)/64)
            continue
        if i<80:
            I.append((math.pi*(50)**4)/64)
            continue
        if i<200+2*n:
            I.append((math.pi*(((50+((i-80)*20/(120+2*n))))**4)/64))
            continue
        if i<260+2*n:
            I.append((math.pi*(70)**4)/64)
            continue
        if i<300+2*n:
            I.append((math.pi*(60)**4)/64)
            continue
        I.append((math.pi*(55)**4)/64)
    mi = []
    mi.append(0)
    for i in range(1,310+2*n+1):
        mi.append(m[i]/I[i] + mi[i-1])
    mii = []
    mii.append(0)
    for i in range(1,310+2*n+1):
        mii.append(mi[i]+mii[i-1])
    const = -mii[310+2*n]/(310+2*n)
    for i in range(0,310+2*n+1):
        mii[i] = (mii[i]+const*i)/E
    plt.figure(figsize=(10,5))
    plt.grid()
    plt.xlabel('Longitud (mm)')
    plt.ylabel('Deformada (mm)')
    plt.plot(mii,'-k')
```

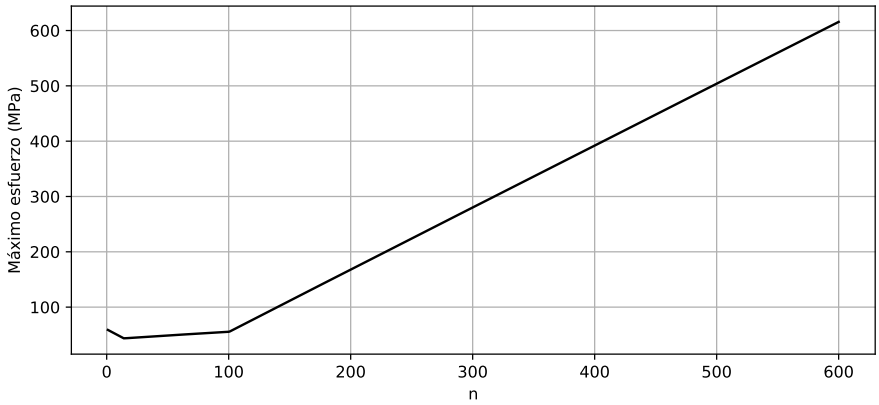
Observe que el valor de la n va desde 1:30. Esto se puede modificar a gusto para observar otros valores de deformada. Así mismo, se calcula el cortante, el momento flector y el momento de inercia para cada posición longitudinal del eje. Para el cálculo se utiliza $310+2n$ puntos.
Observamos que el algoritmo es muy óptimo. Pues la complejidad es de solo $O(n \cdot (310 + 2 \cdot n)) \approx$

$O(2n^2)$. Es decir, en una computadora rutinaria se puede realizar alrededor de 1000 pruebas para distintos valores de n y el resultado se mostraría en menos de un segundo. Todo el código fue implementado en Python 3 y las gráficas hechas en matplotlib, para el cálculo simbólico se utilizó GNU Octave.
Adicionalmente, debido a la facilidad del código escrito, analizamos los valores de flecha máxima, su posición y el esfuerzo máximo para cada valor de n hasta 600:

Valor de n	Flecha máxima (mm)	Posición de flecha máxima (mm)	Máximo esfuerzo (MPa)
1	0.030617	133	58.786495
2	0.031025	134	57.616630
3	0.031434	136	56.447468
4	0.031845	137	55.278996
5	0.032257	138	54.111201
...	...	...	...
596	0.923507	330	611.138406
597	0.928497	331	612.255125
598	0.933504	332	613.371838
599	0.938527	332	614.488545
600	0.943570	333	615.605245







Se observa un comportamiento lineal para los valores de la flecha máxima, cumpliéndose hasta cuando $n = 375$, ahí es cuando la gráfica incrementa su pendiente y comienza a tener un crecimiento de flecha más acelerado.

Capítulo 6

Conclusiones

1. Se concluye que el eje no fallará, pues su deformada es muy pequeña comparada con su longitud y su esfuerzo es pequeño para producir la falla.

2. Se desprecia el peso del eje debido a que dicha fuerza es muy pequeña y solo haría más complicado los cálculos debido a que no es un eje simétrico.

3. Sin tanta pérdida de precisión, observamos que el procedimiento de aproximación de Riemann para el cálculo numérico de una integral nos ayuda para operar rápidamente la deformada del eje.

4. Con alrededor de 350 puntos sobre todo el eje podemos considerar que la aproximación tomada en la integral es lo suficientemente precisa para no cometer errores relativos superiores a 10^{-5}.

5. La velocidad de computación numérica es por mucho menos compleja que la simbólica en paquetes de software. Pues el cálculo exacto de $\iint M/I$ es demasiado complejo debido a las funciones singulares que aparecen; sin embargo por una suma acumulativa puede resolverse en una complejidad $O(n)$, es decir, se puede elevar la precisión hasta 100 veces más y seguir mostrando los resultados rápidamente.

6. Gracias a la facilidad del problema a ser parametrizado según n es posible implementar un código sencillo en algún lenguaje de programación.

7. La ventaja de utilizar un lenguaje de programación para el problema es que podemos modificar un solo valor (n) para observar como varía el esfuerzo cortante, momento flector y deformada para todos esos valores de forma muy rápida; esto tomaría mucho tiempo en un un cálculo por elementos finitos que demanda un proceso de mallado y de diseño.

8. Con gran velocidad y precisión podemos notar observar que el eje no fallará por esfuerzo hasta valores muy grandes de n, el fallo ocurre pues por flexión, donde la flecha crece de manera rápida.

9. Si consideramos que el eje se trata de un acero A-36, comprobamos que para valores de n mayor a 350 se llega al punto de fluencia. Mientras que para n menores a 50 se tiene un factor de seguridad de 3.