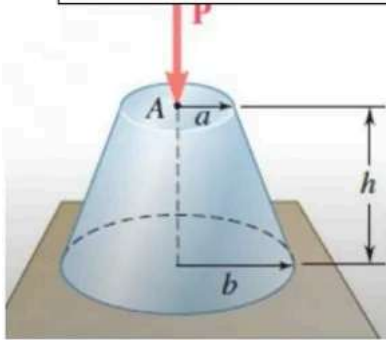


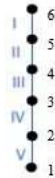
✓ Tarea Cono truncado

Miguel Angel Reales Gaitan Cc. 1007694581

Una carga P vertical se aplica en el cono truncado que se muestra en la figura. Calcule la deformación del punto A usando la discretización sugerida y compárela con el desplazamiento exacto dado por el resultado de la expresión (1).



Discretización propuesta



Valores

P	50000 N
h	0.2 m
E	2.10×10^{11} Pa
a	0.005 m
b	0.01 m

$$\delta_A = \int_x^{x+h} \frac{P dy}{E A(y)} \quad (1)$$

Recuerde, para hallar $A(y)$ tenga en cuenta:

$$A(y) = \pi \cdot \left(\frac{y(b-a)}{h} \right)^2$$

Datos del problema:

- $P = 50000$ N (carga vertical)
- $E = 2.10 \times 10^{11}$ Pa (módulo de Young)
- $a = 0.005$ m (radio superior)
- $b = 0.01$ m (radio inferior)

```
1 import numpy as np
2 from scipy.integrate import quad
3
4 # Datos del problema
5 P = 50000 # N
6 E = 2.10e11 # Pa
7 a = 0.005 # m
8 b = 0.01 # m
9 h = 0.05 # m
10
11 # Discretización con 6 segmentos
12 n = 6
13 x = np.linspace(0, h, n+1)
14 dx = h/n
15
16 def A(y):
17     """Calcula el área en función de la altura y"""
18     return np.pi * ((y*(b-a)/h + a)**2)
19
20 # Método del trapecio
21 deformacion_trap = 0
22 for i in range(n):
23     A_promedio = (A(x[i]) + A(x[i+1]))/2
24     deformacion_trap += (P * dx)/(E * A_promedio)
25
26 # Método de integración numérica
27 deformacion_quad, _ = quad(lambda y: P/(E*A(y)), 0, h)
28
29 print(f"Deformación (trapecio): {deformacion_trap*1000:.6f} mm")
30 print(f"Deformación (quad): {deformacion_quad*1000:.6f} mm")
```

Deformación (trapecio): 0.075183 mm
Deformación (quad): 0.075788 mm

✓ Cálculo de errores

```
1 # Cálculo de errores
2 error_absoluto = abs(deformacion_quad - deformacion_trap)
3 error_relativo = (error_absoluto/deformacion_quad) * 100
```

```

4 print(f"Error absoluto: {error_absoluto*1000:.6f} mm")
5 print(f"Error relativo: {error_relativo:.4f}%")

```

```

↗ Error absoluto: 0.000605 mm
Error relativo: 0.7980%

```

✓ Deformación punto a

```

1 import numpy as np
2 from scipy.integrate import quad
3
4 # Datos del problema
5 P = 50000 # N
6 E = 2.10e11 # Pa
7 a = 0.005 # m (radio en punto A)
8 b = 0.01 # m (radio en base)
9 h = 0.05 # m (altura total)
10
11 def A(y):
12     """Calcula el área en función de la altura y"""
13     return np.pi * ((y*(b-a)/h + a)**2)
14
15 # Método del trapecio para punto A
16 n = 6 # número de segmentos
17 x = np.linspace(0, h, n+1) # desde punto A (y=0) hasta base (y=h)
18 dx = h/n
19
20 deformacion_A_trap = 0
21 for i in range(n):
22     x_i = x[i]
23     x_i1 = x[i+1]
24     A_promedio = (A(x_i) + A(x_i1))/2
25     deformacion_A_trap += (P * dx)/(E * A_promedio)
26
27 # Método quad para punto A
28 deformacion_A_quad, _ = quad(lambda y: P/(E*A(y)), 0, h)
29
30 # Cálculo de errores
31 error_absoluto = abs(deformacion_A_quad - deformacion_A_trap)
32 error_relativo = (error_absoluto/deformacion_A_quad) * 100
33
34 print(f"Deformación en punto A (trapecio): {deformacion_A_trap*1000:.6f} mm")
35 print(f"Deformación en punto A (quad): {deformacion_A_quad*1000:.6f} mm")
36 print(f"Error absoluto: {error_absoluto*1000:.6f} mm")
37 print(f"Error relativo: {error_relativo:.4f}%")
38
39 # Validación con la fórmula exacta dada en el problema
40 deformacion_A_exacta = (P*h)/(E*np.pi*a*b)
41 print(f"\nDeformación exacta según fórmula (1): {deformacion_A_exacta*1000:.6f} mm")
42 print(f"Error vs fórmula exacta: {abs(deformacion_A_exacta - deformacion_A_quad)*1000:.6f} mm")

```

```

↗ Deformación en punto A (trapecio): 0.075183 mm
Deformación en punto A (quad): 0.075788 mm
Error absoluto: 0.000605 mm
Error relativo: 0.7980%

```

```

Deformación exacta según fórmula (1): 0.075788 mm
Error vs fórmula exacta: 0.000000 mm

```

Comparación deformacion total y deformación punto a

```

1 import numpy as np
2 from scipy.integrate import quad
3
4 # Datos del problema
5 P = 50000 # N
6 E = 2.10e11 # Pa
7 a = 0.005 # m
8 b = 0.01 # m
9 h = 0.05 # m
10
11 def A(y):
12     """Calcula el área en función de la altura y"""
13     return np.pi * ((y*(b-a)/h + a)**2)
14
15 # 1. Deformación usando método del trapecio
16 n = 6 # número de segmentos
17 x = np.linspace(0, h, n+1)

```

```
18 dx = h/n
19
20 deformacion_trap = 0
21 for i in range(n):
22     A_promedio = (A(x[i]) + A(x[i+1]))/2
23     deformacion_trap += (P * dx)/(E * A_promedio)
24
25 # 2. Deformación en punto A usando la fórmula exacta
26 deformacion_A_exacta = (P*h)/(E*np.pi*a*b)
27
28 print("\nCOMPARACIÓN DE RESULTADOS:")
29 print(f"Deformación por trapecio: {deformacion_trap*1000:.6f} mm")
30 print(f"Deformación en punto A (exacta): {deformacion_A_exacta*1000:.6f} mm")
31
32 # Cálculo del error
33 error_absoluto = abs(deformacion_A_exacta - deformacion_trap)
34 error_relativo = (error_absoluto/deformacion_A_exacta) * 100
35
36 print(f"\nDiferencia absoluta: {error_absoluto*1000:.6f} mm")
37 print(f"Error relativo: {error_relativo:.2f}%")
```



```
COMPARACIÓN DE RESULTADOS:
Deformación por trapecio: 0.075183 mm
Deformación en punto A (exacta): 0.075788 mm

Diferencia absoluta: 0.000605 mm
Error relativo: 0.80%
```