



Pontificia Universidad Católica de Chile
Departamento de Ingeniería Mecánica
ICM3323 – Elementos Finitos en Diseño Mecánico

Tarea 2: Vigas

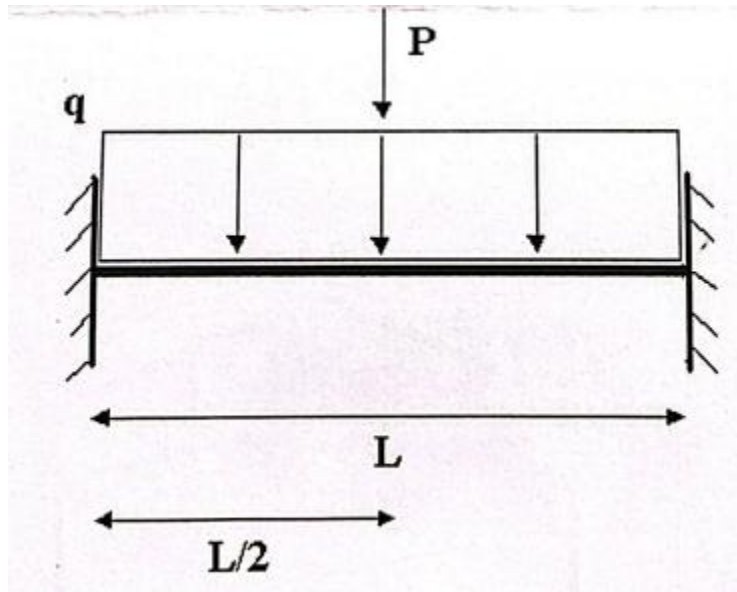
Fecha de entrega: 29/09/2025
Integrantes: Sebastián Walberg

Problema 1

En el problema 1 se pide:

- Obtener la expresión analítica de los campos de tensión y desplazamiento axial.
- Obtener la solución por MEF de dichos campos utilizando al menos tres discretizaciones diferentes con un número creciente de elementos finitos de viga de dos nodos. Justificar la elección de cada una de las discretizaciones propuestas.
- Comparar y discutir las soluciones obtenidas en a) y b).

Esto se debe realizar para el siguiente problema:



Con los siguientes datos: $E = 210 \text{ GPa}$, $S_y = 300 \text{ MPa}$, $I = 2 \cdot 10^6 \text{ mm}^4$, $L = 1 \text{ m}$, $q = 1 \text{ kN/m}$, $P = 2 \text{ kN}$.

Solución

Forma analítica:

En primer lugar, se calcula la expresión analítica de los campos de tensión y desplazamiento axial. Para ello se calculan en primer lugar las reacciones y los momentos en los empotramientos:

$$R_A = R_B = \frac{P + qL}{2}$$
$$M_A = M_B = - \left(\frac{qL^2}{12} + \frac{PL}{8} \right)$$

Luego, para obtener las funciones de corte, momento y theta se siguen las siguientes relaciones (Euler-Bernoulli):

$$\theta(x) = \frac{dw}{dx}$$

$$M(x) = EI \frac{d^2w}{dx^2}$$

$$\frac{dM}{dx} = V(x)$$

$$\frac{dV}{dx} = -q(x)$$

Con esto, se calculan las funciones para el caso de la carga distribuida “ q ” de extremo a extremo:

$$V_q(x) = \frac{q}{2} (L - 2x)$$

$$M_q(x) = -\frac{q}{12} (6x^2 - 6Lx + L^2)$$

$$\theta_q(x) = w'_q(x) = -\frac{q}{12EI} x (L - x) (L - 2x)$$

$$w_q(x) = -\frac{q}{24EI} x^2 (L - x)^2$$

Luego, se repite el procedimiento para la carga puntual P :

$$V_P(x) = \begin{cases} \frac{P}{2}, & 0 \leq x < \frac{L}{2}, \\ -\frac{P}{2}, & \frac{L}{2} < x \leq L. \end{cases}$$

$$M_P(x) = \begin{cases} -\frac{PL}{8} + \frac{P}{2}x, & 0 \leq x < \frac{L}{2}, \\ \frac{3PL}{8} - \frac{P}{2}x, & \frac{L}{2} \leq x \leq L. \end{cases}$$

$$\theta_P(x) = w'_P(x) = \begin{cases} -\frac{P}{8EI} x (L - 2x), & 0 \leq x < \frac{L}{2}, \\ -\frac{P}{8EI} (2x^2 - 3Lx + L^2), & \frac{L}{2} \leq x \leq L. \end{cases}$$

$$w_P(x) = \begin{cases} -\frac{P}{48EI} x^2 (3L - 4x), & 0 \leq x < \frac{L}{2}, \\ -\frac{P}{48EI} (4x^3 - 9Lx^2 + 6L^2x - L^3), & \frac{L}{2} \leq x \leq L. \end{cases}$$

Finalmente, aplicando regla de superposición, las funciones para el caso de la carga distribuida más la carga puntual son:

$$V(x) = \begin{cases} \frac{q}{2} (L - 2x) + \frac{P}{2}, & 0 \leq x < \frac{L}{2}, \\ \frac{q}{2} (L - 2x) - \frac{P}{2}, & \frac{L}{2} < x \leq L, \end{cases}$$

$$M(x) = \begin{cases} -\frac{q}{12} (6x^2 - 6Lx + L^2) - \frac{PL}{8} + \frac{P}{2}x, & 0 \leq x < \frac{L}{2}, \\ -\frac{q}{12} (6x^2 - 6Lx + L^2) + \frac{3PL}{8} - \frac{P}{2}x, & \frac{L}{2} \leq x \leq L, \end{cases}$$

$$\theta(x) = \begin{cases} -\frac{q}{12EI} x (L - x) (L - 2x) - \frac{P}{8EI} x (L - 2x), & 0 \leq x < \frac{L}{2}, \\ -\frac{q}{12EI} x (L - x) (L - 2x) - \frac{P}{8EI} (2x^2 - 3Lx + L^2), & \frac{L}{2} \leq x \leq L, \end{cases}$$

$$w(x) = \begin{cases} -\frac{q}{24EI} x^2 (L - x)^2 - \frac{P}{48EI} x^2 (3L - 4x), & 0 \leq x < \frac{L}{2}, \\ -\frac{q}{24EI} x^2 (L - x)^2 - \frac{P}{48EI} (4x^3 - 9Lx^2 + 6L^2x - L^3), & \frac{L}{2} \leq x \leq L. \end{cases}$$

Una vez obtenidas las funciones analíticas, estas se grafican en Python:

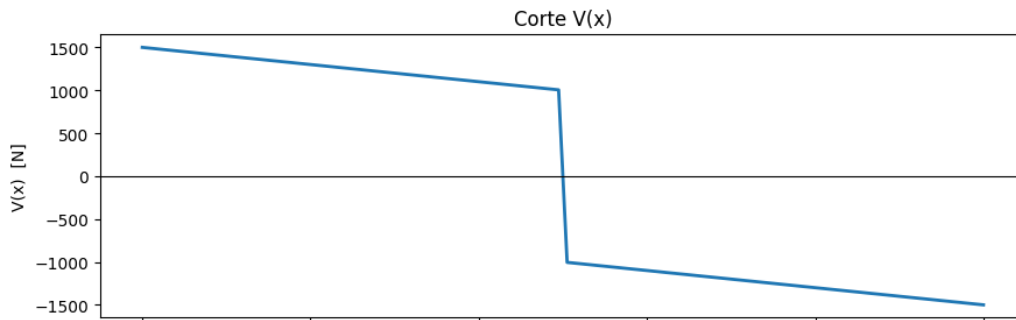


Figura 1: Diagrama de corte.

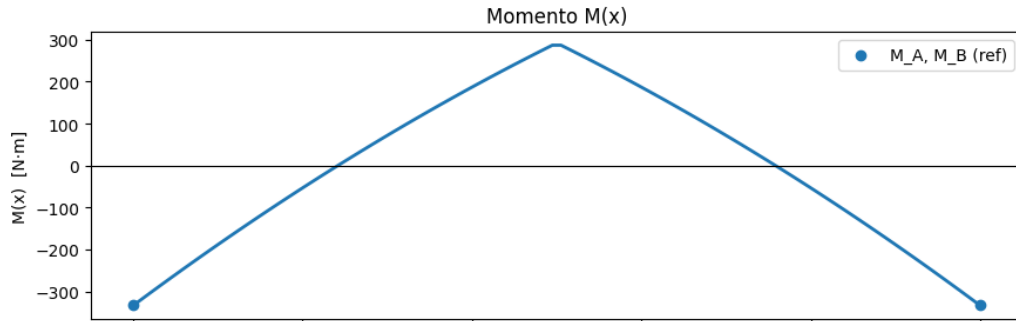


Figura 2: Diagrama de momento.

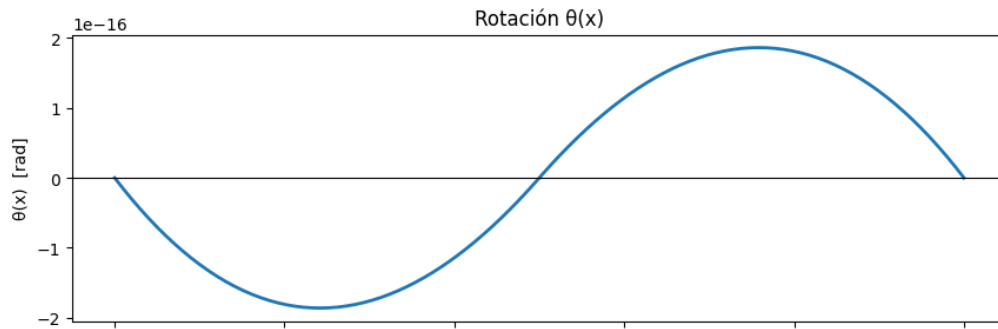


Figura 3: Diagrama de rotación.

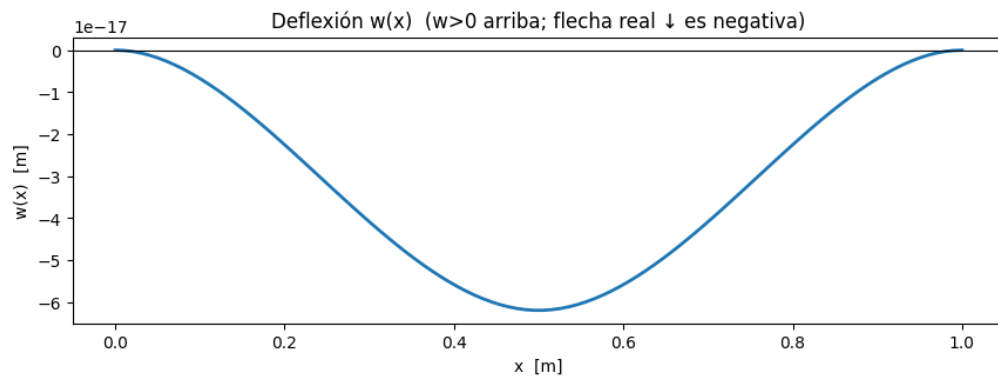


Figura 4: Diagrama de deflexión.

Forma MEF:

Aplicando elementos finitos para analizar la viga, se arma un código en Python. Para ello se tiene un archivo “funciones_P1”, donde se tienen las funciones necesarias para el calculo por MEF. Algunas de las principales son:

```

1 def beam_eb_local_stiffness(E: float, I: float, L: float) -> np.ndarray:
2     EI = E * I
3     L2, L3 = L*L, L*L*L
4     k = np.array([
5         [ 12*EI/L3,    6*EI/L2,   -12*EI/L3,    6*EI/L2],
6         [  6*EI/L2,    4*EI/L,    -6*EI/L2,    2*EI/L ],
7         [-12*EI/L3,   -6*EI/L2,    12*EI/L3,   -6*EI/L2],
8         [  6*EI/L2,    2*EI/L,    -6*EI/L2,    4*EI/L ],
9     ], dtype=float)
10    return k

```

Figura 5: Función para la matriz k local de cada elemento.

```

1 def hermite_shapes(xi: float, L: float) -> Tuple[float, float, float, float]:
2     N1 = 1 - 3*xi**2 + 2*xi**3
3     N2 = L * (xi - 2*xi**2 + xi**3)
4     N3 = 3*xi**2 - 2*xi**3
5     N4 = L * (-xi**2 + xi**3)
6     return N1, N2, N3, N4

```

Figura 6: Definición de los “N”.

```

1 def hermite_shapes_all(xi: float, L: float):
2     # N
3     N1 = 1 - 3*xi**2 + 2*xi**3
4     N2 = L * (xi - 2*xi**2 + xi**3)
5     N3 = 3*xi**2 - 2*xi**3
6     N4 = L * (-xi**2 + xi**3)
7     # dN/dx
8     dN1dx = (-6*xi + 6*xi**2) / L
9     dN2dx = (1 - 4*xi + 3*xi**2)
10    dN3dx = ( 6*xi - 6*xi**2) / L
11    dN4dx = (-2*xi + 3*xi**2)
12    # d2N/dx2
13    d2N1dx2 = (-6 + 12*xi) / L**2
14    d2N2dx2 = (-4 + 6*xi) / L
15    d2N3dx2 = ( 6 - 12*xi) / L**2
16    d2N4dx2 = (-2 + 6*xi) / L
17    return (N1, N2, N3, N4, dN1dx, dN2dx, dN3dx, dN4dx, d2N1dx2, d2N2dx2, d2N3dx2, d2N4dx2)

```

Figura 7: Calculo de derivadas para obtener las funciones correspondientes.

Luego, se obtienen los gráficos para distintas cantidades de elementos. Específicamente se utilizaron 2, 4 y 10.

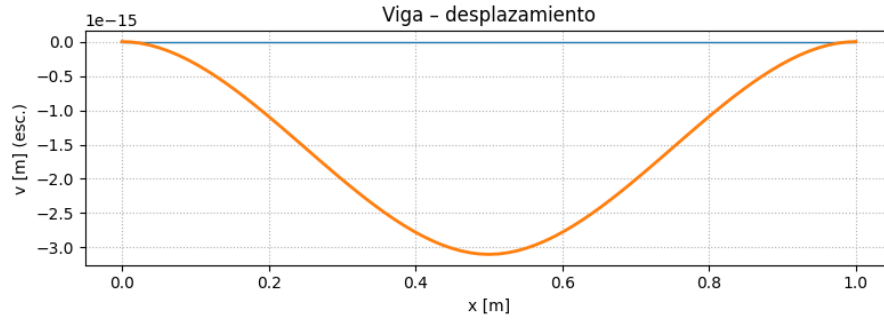


Figura 8: Diagrama de desplazamiento para 2 elementos.

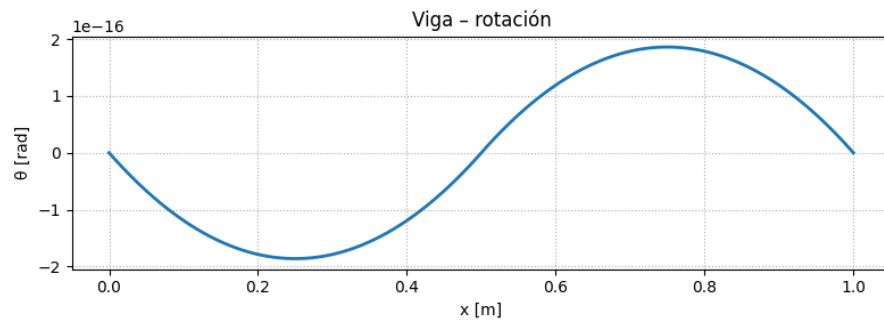


Figura 9: Diagrama de rotación para 2 elementos.

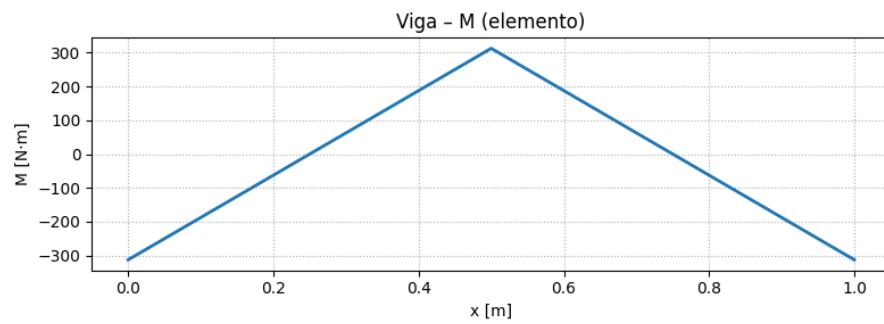


Figura 10: Diagrama de momento para 2 elementos.

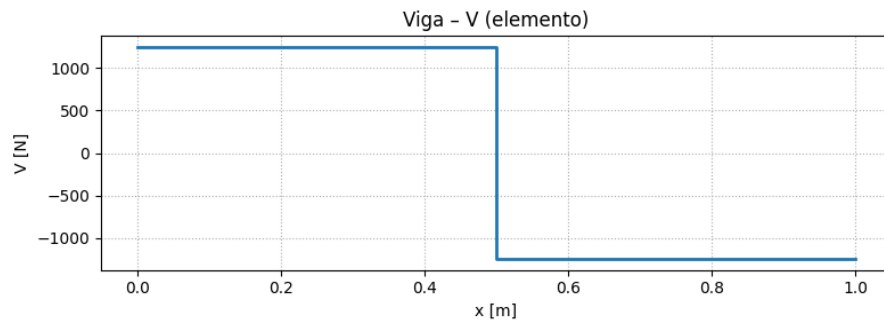


Figura 11: Diagrama de corte para 2 elementos.

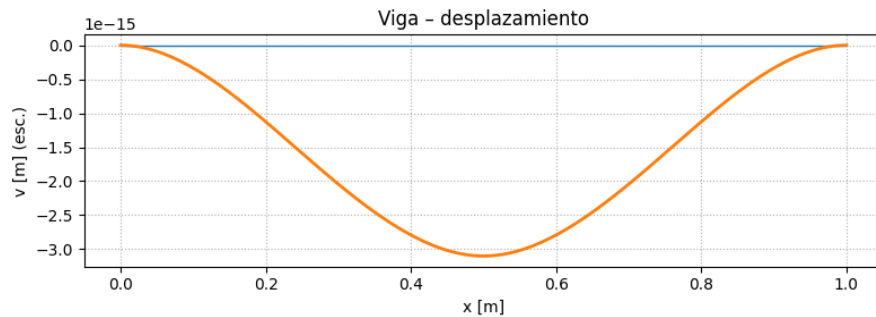


Figura 12: Diagrama de desplazamiento para 4 elementos.

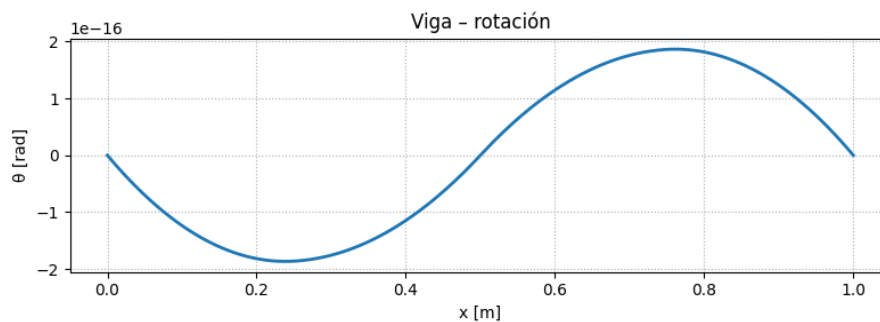


Figura 13: Diagrama de rotación para 4 elementos.

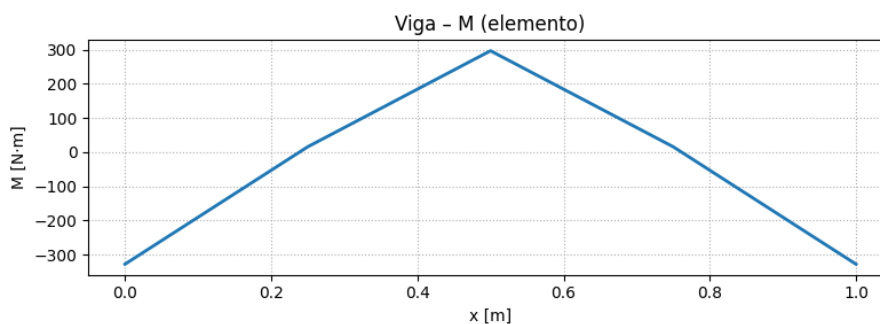


Figura 14: Diagrama de momento para 4 elementos.

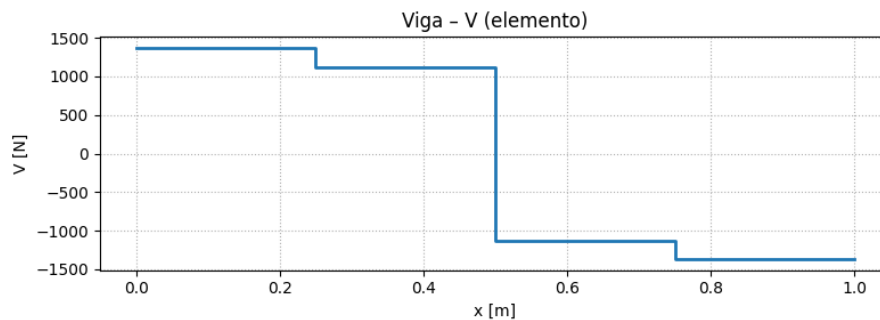


Figura 15: Diagrama de corte para 4 elementos.

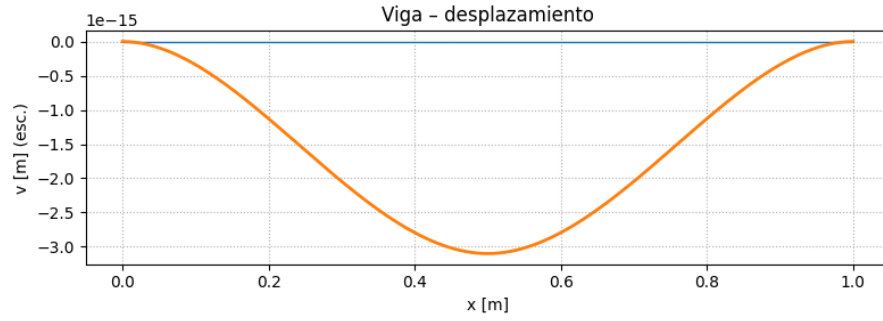


Figura 16: Diagrama de desplazamiento para 10 elementos.

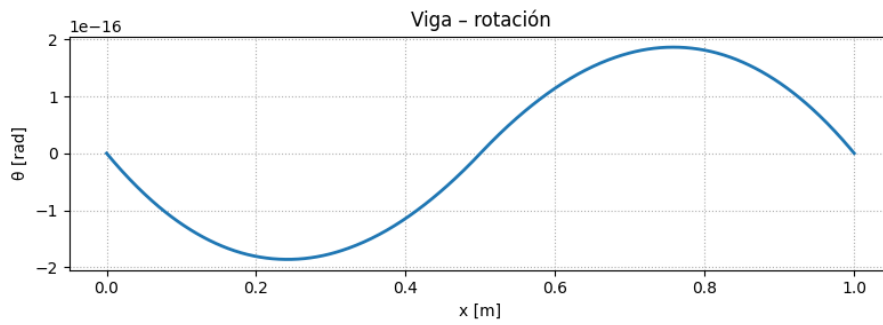


Figura 17: Diagrama de rotación para 10 elementos.

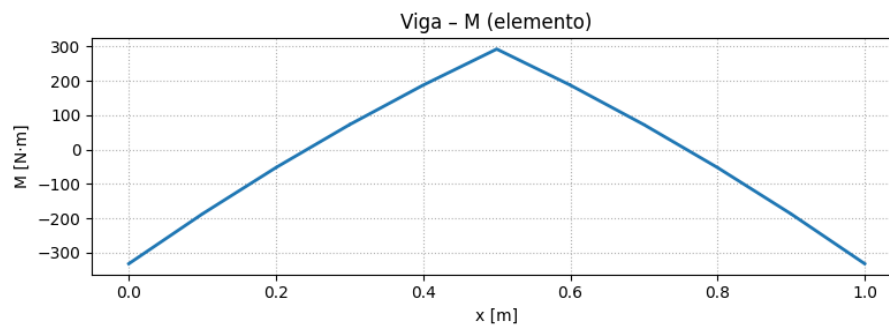


Figura 18: Diagrama de momento para 10 elementos.

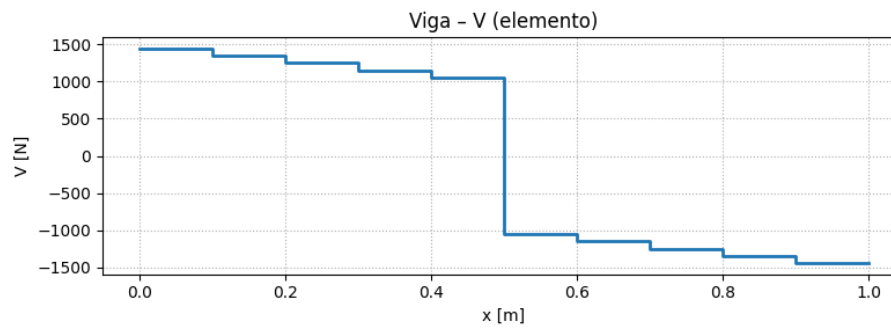


Figura 19: Diagrama de corte para 10 elementos.

Comparación entre ambos métodos:

Como se observan los gráficos, la principal diferencia de la forma analítica y MEF se presenta en el diagrama de momento y corte. Si bien en ambos casos se tienen las mismas formas de gráficos, estos difieren en cómo se construyen según la cantidad de elementos.

Para el momento, la forma analítica se compone de 2 curvas que se encuentran en el medio, donde el momento es máximo. Sin embargo, en el cálculo MEF se tienen rectas según la cantidad de elementos. Con 2 discretizaciones se tienen 2 rectas, similar a la recta de valor absoluto pero invertida, luego con 4, se tienen 4 rectas que empiezan a formar la curva analítica. Finalmente, con 10 elementos ya se logra apreciar de mejor manera la curva teórica del problema.

En el caso del corte, es más notoria la diferencia entre ambos métodos. Con 2 elementos las 2 curvas que componen el gráfico analítico de corte, se vuelven rectas sin pendiente. Luego, al incrementando la cantidad de elementos, se aprecian las pendientes, pero ahora más cercano a formar la recta analítica. Con esto, se puede decir que, a mayor cantidad de elementos, más se parecerá la curva MEF a la analítica, no obstante, con una discretización de 2 elementos ya se podría considerar acertados los gráficos, ya que en general (sin contar el corte), se tiene un diagrama cercano al analítico.

Extra: 100 elementos

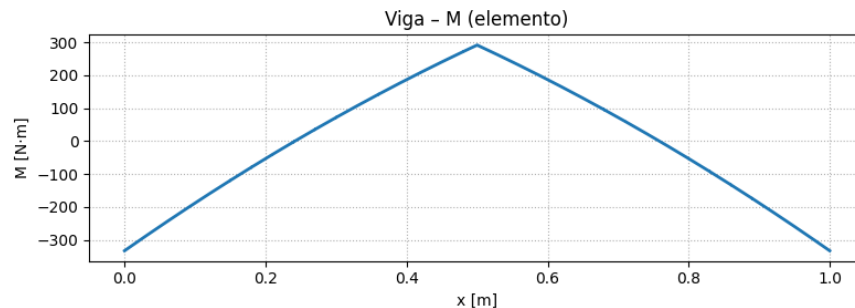


Figura 18: Diagrama de momento para 100 elementos.

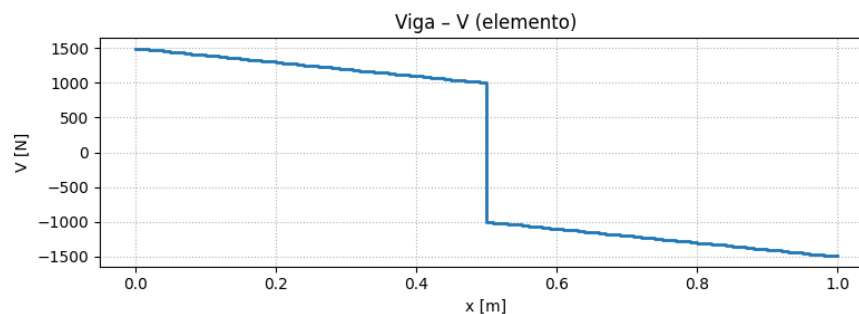
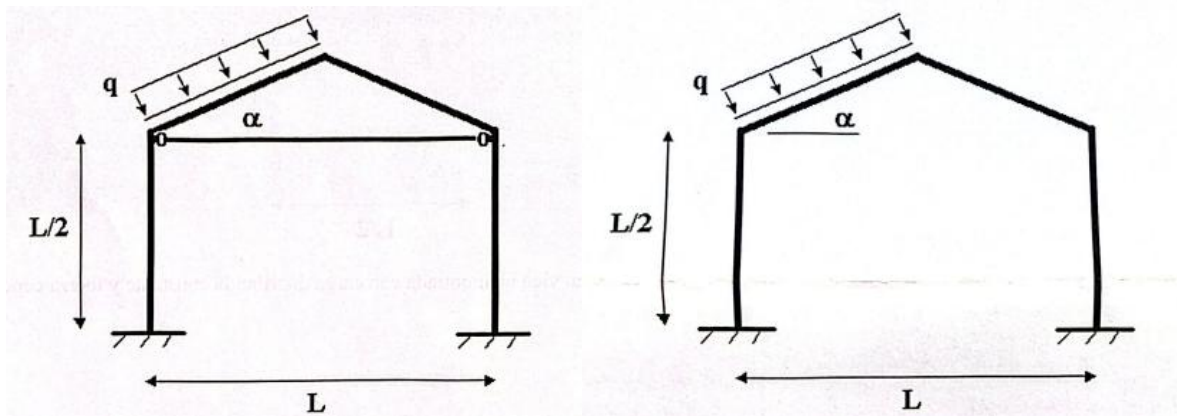


Figura 21: Diagrama de corte para 100 elementos.

Problema 2

En el problema 2 se pide comparar 2 diseños de pórticos, haciendo análisis MEF:



Con los siguientes datos: $E = 206 \text{ GPa}$, $S_y = 300 \text{ MPa}$, $I = 2 \cdot 10^6 \text{ mm}^4$, $A = 200 \text{ mm}^2$, $L = 6 \text{ m}$, $\alpha = 30^\circ$, $q = 1 \text{ kN/m}$.

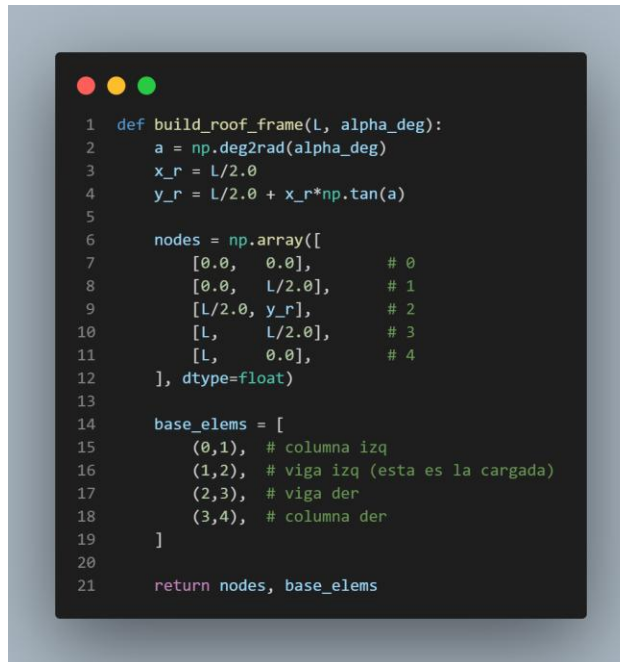
Solución

Para el cálculo MEF del pórtico sin tensor, es necesario plantear las funciones correspondientes, las cuales se encuentran en el archivo “funciones_P2”. En primer lugar, se establece la matriz de rigidez local, la cual es la misma que para el problema 1:

```
1 def k_local_beam(E,A,I,L):
2     EA_L = E*A/L
3     EI_L3 = E*I/L**3
4     EI_L2 = E*I/L**2
5     EI_L = E*I/L
6     k = np.array([
7         [EA_L, 0, 0, -EA_L, 0, 0],
8         [0, 12*EI_L3, 6*EI_L2, 0, -12*EI_L3, 6*EI_L2],
9         [0, 6*EI_L2, 4*EI_L, 0, -6*EI_L2, 2*EI_L],
10        [-EA_L, 0, 0, EA_L, 0, 0],
11        [0, -12*EI_L3, -6*EI_L2, 0, 12*EI_L3, -6*EI_L2],
12        [0, 6*EI_L2, 2*EI_L, 0, -6*EI_L2, 4*EI_L]
13    ], dtype=float)
14
15     return k
```

Figura 22: Función de cálculo de matriz de rigidez local.

Luego, se arma la función para construir el pórtico, la cual presenta la posición de los nodos, además de definir los elementos principales en base a los nodos, tal si fuera la matriz de conectividad.



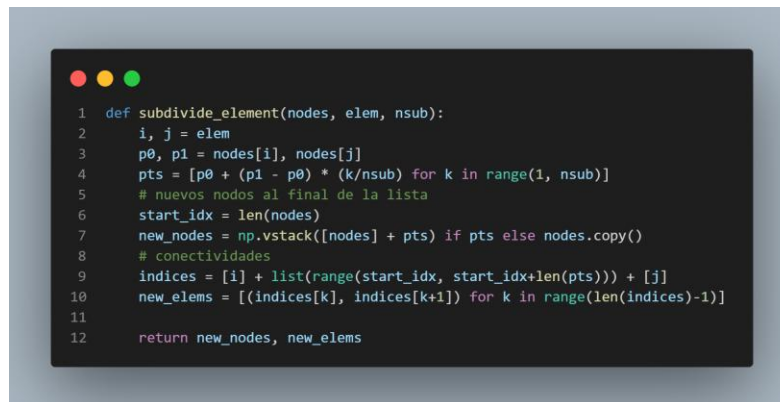
```

1 def build_roof_frame(L, alpha_deg):
2     a = np.deg2rad(alpha_deg)
3     x_r = L/2.0
4     y_r = L/2.0 + x_r*np.tan(a)
5
6     nodes = np.array([
7         [0.0, 0.0], # 0
8         [0.0, L/2.0], # 1
9         [L/2.0, y_r], # 2
10        [L, L/2.0], # 3
11        [L, 0.0], # 4
12    ], dtype=float)
13
14    base_elems = [
15        (0,1), # columna izq
16        (1,2), # viga izq (esta es la cargada)
17        (2,3), # viga der
18        (3,4), # columna der
19    ]
20
21    return nodes, base_elems

```

Figura 23: Función para armado de pórtico.

Cabe mencionar que se arma la función “*subdivide_element*” para discretizar en una mayor cantidad elementos la viga que sostiene la carga q , con tal de obtener una mejor precisión en como se desplaza esta parte del pórtico. Para ello se agregan nodos, y se crean nuevos elementos tomando en cuenta la función de la Figura 23. En este caso, se utilizó una discretización de 10 elementos para la viga cargada.



```

1 def subdivide_element(nodes, elem, nsub):
2     i, j = elem
3     p0, p1 = nodes[i], nodes[j]
4     pts = [p0 + (p1 - p0) * (k/nsub) for k in range(1, nsub)]
5     # nuevos nodos al final de la lista
6     start_idx = len(nodes)
7     new_nodes = np.vstack([nodes] + pts) if pts else nodes.copy()
8     # conectividades
9     indices = [i] + list(range(start_idx, start_idx+len(pts))) + [j]
10    new_elems = [(indices[k], indices[k+1]) for k in range(len(indices)-1)]
11
12    return new_nodes, new_elems

```

Figura 24: Función para subdividir la viga cargada.

Luego, se resuelve el sistema con la función “*assemble_solve*”, la cual permite transformar cada elemento local a una forma global, para posteriormente realizar el calculo de los desplazamientos, las fuerzas y posteriormente los esfuerzos. Estos resultados se presentan en los siguientes diagramas:

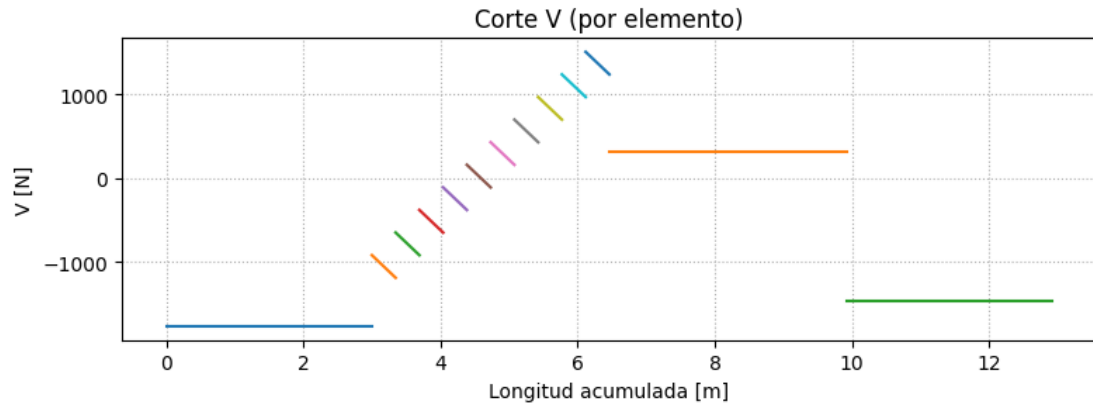


Figura 25: Diagrama de corte para pórtico, según elemento.

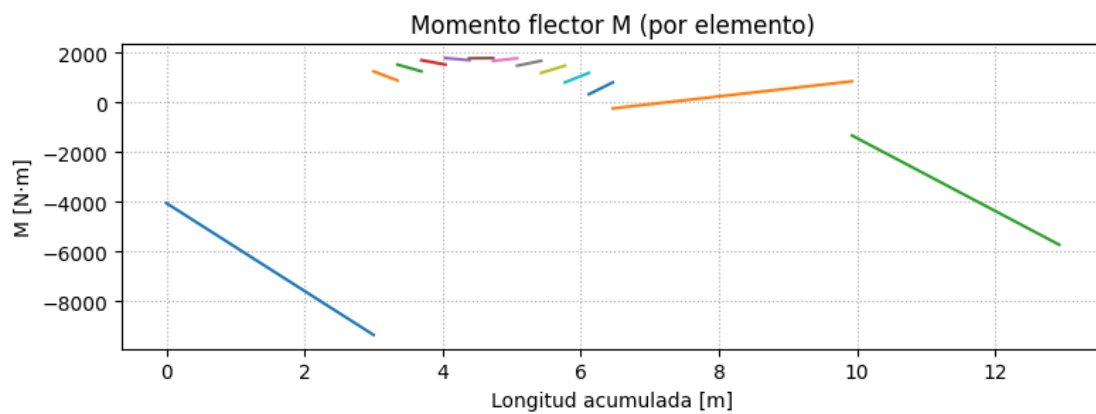


Figura 26: Diagrama de momento para pórtico, según elemento.

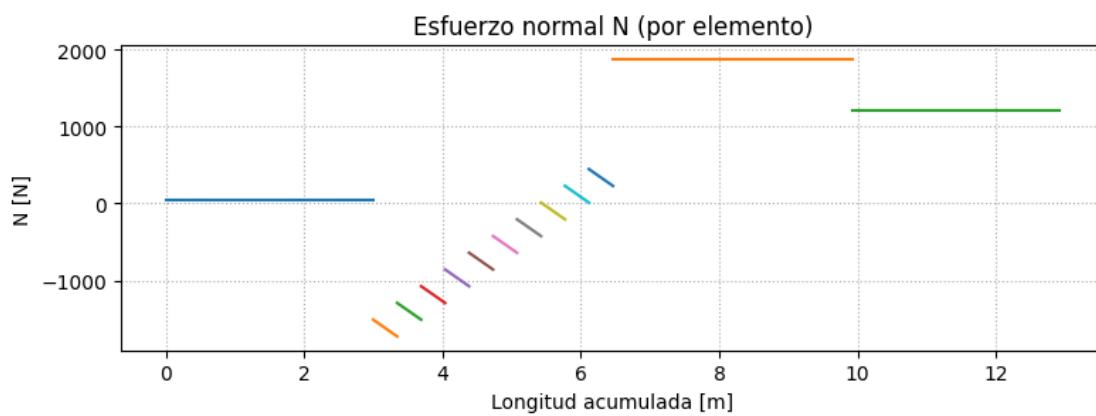


Figura 27: Diagrama de esfuerzo para pórtico, según elemento.

Como se observa en los gráficos, para cada uno de los elementos que componen la diagonal cargada, se tiene su recta para el corte, momento y esfuerzo. Además, se grafica el pórtico comparando antes y después los desplazamientos con la función *plot_portico*, utilizando una escala de 10 para una mejor visualización de lo que ocurre.

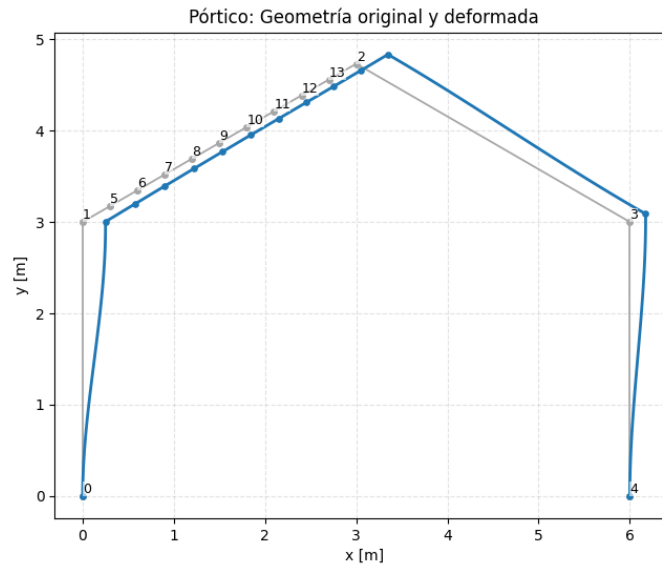


Figura 28: Diagrama de pórtico donde se tiene antes y después de la carga.

Por otro lado, para el cálculo del pórtico con tensor se realizó más funciones, con tal de poder calcular como se deforma la estructura contando con esta viga, la cual tiene un área de $A/2$ y una inercia de 0.

```

1 def k_local_tie_axial(E, A, L):
2
3     if L <= 0.0:
4         return np.zeros((6,6), dtype=float)
5     k = np.zeros((6,6), dtype=float)
6     k_ax = E*A/L
7     k[0,0] = k_ax; k[0,3] = -k_ax
8     k[3,0] = -k_ax; k[3,3] = k_ax
9
10    return k
11
12 def add_horizontal_tie(nodes, elems, n1=1, n2=3):
13     elems_new = list(elems)
14     tie_elem_ids = [len(elems_new)]
15     elems_new.append((n1, n2))
16
17     return elems_new, tie_elem_ids

```

Figura 29: Funciones para el cálculo de pórtico con tensor.

Además, se presenta el diagrama de la estructura con el tensor, con una escala de 10, al igual que anteriormente:

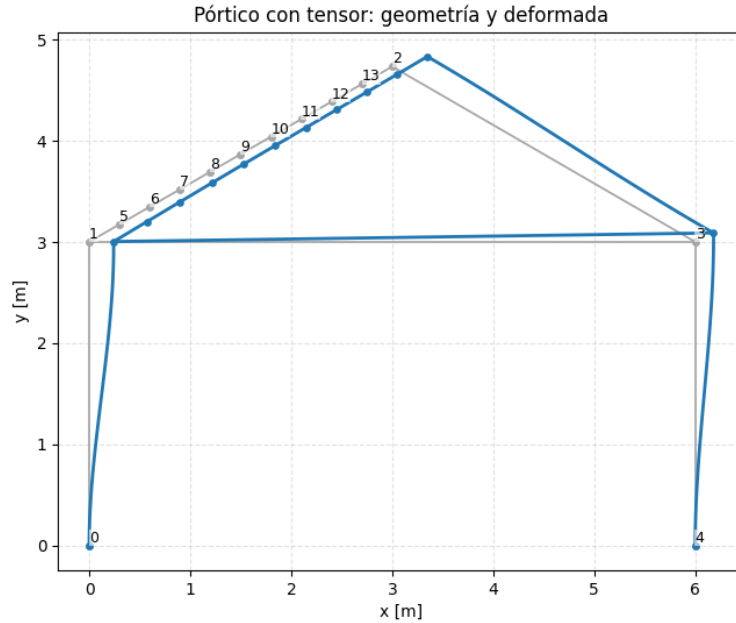


Figura 30: Diagrama de pórtico con tensor donde se tiene antes y después de la carga.

Luego, para la comparación de ambas estructuras se realizaron funciones con tal de visualizar la diferencia en el desplazamiento. Las funciones “*report_nodes_disp*”, “*get_node_disp*”, “*barplot_nodes_disp*” calculan y entregan los resultados:

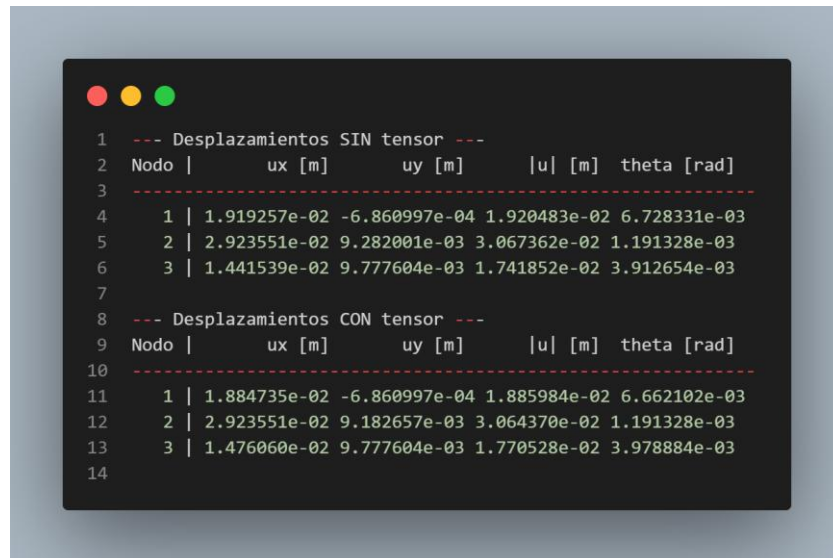


Figura 31: Resultados numéricos de desplazamiento para ambos casos.

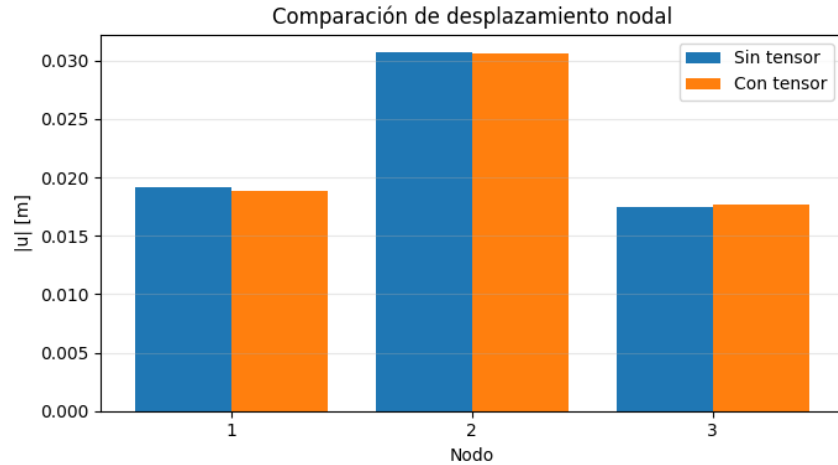


Figura 32: Grafico de barras para comparar desplazamiento para ambos casos.

Como se observa en la Figura 31 y en la Figura 32, los desplazamientos no se ven afectados en gran medida, siendo una diferencia no mayor al 2% para los 3 nodos principales (1.8% para nodo 1, 0.1% para nodo 2 y 1.76% para nodo 3). Esto se puede deber a la manera en que esta direccionada la carga, ya que el tensor debería aportar mayor estabilidad al aplicarse una carga completamente vertical en el techo del pórtico. También se suma que el tensor tiene un área de $A/2$, lo cual puede que no sea suficiente el tamaño para hacer una diferencia notoria. Sin embargo, este tensor puede ayudar a este pórtico según el uso que tendrá, y las demás cargas que se pueden presentar en esta estructura, por lo que, a pesar de la diferencia marginal, convendría agregarlo para una mayor seguridad.

Anexo

Cabe mencionar que se adjuntan los archivos de código, los cuales siguen la siguiente estructura:

- “Resultados_T2_FEM”: Jupyter notebook que compila los resultados de la pregunta 1 y 2, haciendo llamado a las funciones para las respectivas partes.
- “funciones_P1” y “funciones_P2”: Codigos .py los cuales contienen las funciones que se aplican en “Resultados_T2_FEM”, para la pregunta 1 y 2, respectivamente.

Se adjunta enlace a repositorio de Github para visualizar más fácilmente los códigos:

<https://github.com/ElCba12/Tarea-2---FEM.git>