Asignaci?n 2 elementos finitos

January 25, 2017

1 Asignación # 2 de Elementos Finitos Aplicados a la Mecánica de Sólidos

Eduardo Vieira
Universidad Central de Venezuela
Facultad de Ingeniería
Escuela de Ingeniería Mecánica
eduardo.vieira@ucv.ve

1.1 1. Introducción

En esta segunda asignación de la materia se resolverá el ejercicio propuesto 8.14 del libro Introducción al Estudio del Elemento Finito en Ingeniería (Tirupathi R. Chandrupatla y Ashok D. Belegundu, 1999, 2^{da} Edición en español)

1.2 2. El Problema

Considere el marco de acero mostrado en la figura 1, que está sometido a una carga de viento y a una carga de techo. Determine los momentos flexionanes en la estructura (máximo $M_{y'}$ y $M_{z'}$).

Se considera además que el perfil está rotado 30° respecto al eje x'

Figura 1: Esctructura a estudiar

1.3 3. Datos

Área de la sección transversal e inercias del perfil usado.

	Área in^2	$I'_y in^4$	$I'_z in^4$	$J in^4$
Columnas	6.0	3.75	51.0	0.24
Vigas	3.0	1.26	17.0	0.08

Perfil usado

Figura 2: Perfil

1.4 4. Solución

1.5 Fuerza distribuida

Podemos observar que la fuerza distribuida sobre el área superior se distribuye a por toda la longitud de las vigas. Así podemos calcular la fuerza total ejercida verticalmente multiplicando el valor de la fuerza distribuida por el área y como esta fuerza se distribuye por toda la longitud de la viga la fuerza por unidad de longitud será el valor de la fuerza entre la longitud total de las vigas.

1.6 Se importan las librerías a utilizar

```
In [1]: import pandas as pd                                  # Librería para el uso de tablas
        import numpy as np                                   # Librería de Python numérico
        from __future__ import division, print_function      # Se importan las nuevas funciones print y el
        from IPython.display import display                  # Salida en HTML
```

```
/home/eduardo/anaconda2/lib/python2.7/site-packages/pandas/computation/__init__.py:19: UserWarning: The i

  UserWarning)
```

1.7 Calculamos el área

Podemos observar que la fuerza distribuida sobre el área superior se distribuye a por toda la longitud de las vigas.

```
In [2]: A = 10 # Longitud a en [in]
        B = 15 # Longitud b en [in]
        area = A * B
        print("El area es: ", area, " [in2]")
```

```
El area es:  150  [in2]
```

1.8 La fuerza equivalente producida por la fuerza distribuida

El valor total de la fuerza lo podemos obtener multiplicando la fuerza distribuida $100[lb/ft^2] = 0.694444[lb/in^2]$ por el área.

```
In [3]: fuerza_dist_area = 0.694444 # Valor de la fuerza distribuida sobre el área de la plataforma [lb
        fuerza_tot = fuerza_dist_area * area
        print("La fuerza total es de: ", fuerza_tot, " [lb]")
```

```
La fuerza total es de:  104.1666  [lb]
```

1.9 La suma de las longitudes de las vigas

La suma de las longitudes de las vigas será

```
In [4]: long_tot_vigas = 2 * A + 2 * B # Longitud total de las vigas
        print("La suma de longitudes de las vigas es de: ", long_tot_vigas, " [in]")
```

```
La suma de longitudes de las vigas es de:  50  [in]
```

1.10 Valor de la fuerza distribuida sobre las vigas

Entonces la fuerza distribuida sobre las vigas será el valor de la fuerza equivalente entre la suma de las longitudes de las vigas.

```
In [5]: fuerza_dist_viga = fuerza_tot / long_tot_vigas
        print("La fuerza distribuida en las vigas es de: ", fuerza_dist_viga, " [lb/in]")
```

```
La fuerza distribuida en las vigas es de:  2.083332  [lb/in]
```

La estructura final nos quedaría como se muestra en la figura 3.
Figura 3: Carga distribuida sobre las vigas, elementos y nodos de la estructura

1.11 Coordenadas nodales

De la figura 3 podemos obtener las coordenadas de los nodos representadas en la tabla siguiente.

```
In [6]: x = np.array([0, 15,  0, 15,  0, 15,  0, 15])
        y = np.array([0,  0, 10, 10,  0,  0, 10, 10])
        z = np.array([0,  0,  0,  0, 15, 15, 15, 15])
        coordenadas_nodales = pd.DataFrame({'X': x,
                                            'Y': y,
                                            'Z': z},
                                            index = [1,2,3,4,5,6,7,8])
        print('La tabla de coordenadas nodales es la siguiente: ')
        display(coordenadas_nodales)
```

```
La tabla de coordenadas nodales es la siguiente:
```

	X	Y	Z
1	0	0	0
2	15	0	0
3	0	10	0
4	15	10	0
5	0	0	15
6	15	0	15
7	0	10	15
8	15	10	15

1.12 Tabla de conectividad

Una vez determinadas la coordenadas nodales se establece la conectividad de los elementos.

```
In [7]: conectividad = pd.DataFrame({1:      [1, 2, 3, 4, 5, 5, 6, 7],
                                     2:      [5, 6, 7, 8, 6, 7, 8, 8]},
                                     index = [1, 2, 3, 4, 5, 6, 7, 8])
        print('Tabla de conectividad')
        display(conectividad)
```

```
Tabla de conectividad
```

	1	2
1	1	5
2	2	6
3	3	7
4	4	8
5	5	6
6	5	7
7	6	8
8	7	8

1.13 Cosenos directores y longitud de elemento

Debemos determinar la matriz

$$\lambda = \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix}$$

Para ello determinaremos l_1, m_1 y n_1 como sigue

$l_1 = \frac{(x_2 - x_1)}{l_e}$

$m_1 = \frac{(y_2 - y_1)}{l_e}$

$n_1 = \frac{(z_2 - z_1)}{l_e}$

Calculamos la longiud del elemento

$l_e = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$

Sea $V'_x = [l_1, m_1, n_1]$ el vector unitario a lo largo del eje x'. Sea también un punto de refencia 3 que está contenido en el plano del eje y' y la recta que une 12 pero no esta conenido en la misma, entonces

$V_{13} = \left[\frac{(x_3 - x_1)}{l_{13}}, \frac{(y_3 - y_1)}{l_{13}}, \frac{(z_3 - z_1)}{l_{13}}\right]$

El vector unitrio a lo largo del eje z está dado por:

$V'_z = \left[l_3, m_3, n_3\right] = \frac{V'_x \times V_{13}}{|V'_x \times V_{13}|}$

Y por último podemos calcular los cosenos directores para el eje y'

$V'_y = \left[l_2, m_2, n_2\right] = V'_z \times V'_x$

In [8]:
```
# Número de elementos
ne = 8

# Creamos los vectores de los cosenos directores y de la longiud del elemento
# y los llenamos con ceros
le = np.zeros(ne)
l1 = np.zeros(ne)
m1 = np.zeros(ne)
n1 = np.zeros(ne)
l2 = np.zeros(ne)
m2 = np.zeros(ne)
n2 = np.zeros(ne)
l3 = np.zeros(ne)
m3 = np.zeros(ne)
n3 = np.zeros(ne)

# Punto de referencia 3
x3 = np.zeros(ne)
y3 = np.zeros(ne)
z3 = np.zeros(ne)

# Columnas (elementos 1,2,3 y 4)
for i in [0,1,2,3]:
    # De la tabla de conectividad extraemos el nodo inicial
    # y el nodo final para el elemento
    nodo_i = conectividad.loc[i+1,1]
    nodo_f = conectividad.loc[i+1,2]

    # De la tabla de coordenadas nodales podemos extraer
    # las coordenadas de los nodos inicial y final
    x_i = coordenadas_nodales.loc[nodo_i,'X']
    x_f = coordenadas_nodales.loc[nodo_f,'X']
    y_i = coordenadas_nodales.loc[nodo_i,'Y']
    y_f = coordenadas_nodales.loc[nodo_f,'Y']
    z_i = coordenadas_nodales.loc[nodo_i,'Z']
    z_f = coordenadas_nodales.loc[nodo_f,'Z']


    # Calculamos la longitud y los cosenos directores
```

```
        le[i] = np.sqrt((x_f - x_i) ** 2 + (y_f - y_i) ** 2 + (z_f - z_i) ** 2)
        l1[i] = (x_f - x_i) / le[i]
        m1[i] = (y_f - y_i) / le[i]
        n1[i] = (z_f - z_i) / le[i]

        V_x = np.array([l1[i], m1[i], n1[i]])

        # Punto de refenrecia 3, en este caso se trat de un plano que forma 30° con el plano yz
        # El primer vector del plano es el que va del punto inicial al punto final
        vec_1 = np.array([x_f-x_i, y_f-y_i, z_f-z_i])
        # El segundo vector del plano se tomará como un vector unitario en el plano XY que forma 30°
        vec_2 = np.array([np.sin(30 * np.pi / 180), np.cos(30 * np.pi / 180), 0])
        # El vector normal del plano será el producto cruz entre ellos 2
        vec_norm = np.cross(vec_1,vec_2)

        # Se escogen unas componentes Y y Z cualesquiera
        y_3 = np.random.random()
        z_3 = np.random.random()

        # Se calcula X con la ecuación del plano usando el vector normal y el punto inicial
        x_3 = x_i - (vec_norm[1] * (y_3 - y_i) + vec_norm[2] * (z_3 - z_i)) / vec_norm[0]

        le_13 = np.sqrt((x_3 - x_i) ** 2 + (y_3 - y_i) ** 2 + (z_3 - z_i) ** 2)

        # Costruimos el vector de cosenos directores de referencia V_13
        V_13 = np.array([(x_3 - x_i) / le_13, (y_3 - y_i) / le_13, (z_3 - z_i) / le_13])

        # Calculamos V_z
        cross_vx_v13 = np.cross(V_x,V_13)
        V_z = (cross_vx_v13) / np.sqrt(np.dot(cross_vx_v13,cross_vx_v13))

        # Calculamos V_y
        V_y = np.cross(V_z, V_x)

        # Llenamos los vectores l2, m2, n2, l3, m3, n3, x3, y3 y z3
        l2[i] = V_y[0]
        m2[i] = V_y[1]
        n2[i] = V_y[2]

        l3[i] = V_z[0]
        m3[i] = V_z[1]
        n3[i] = V_z[2]

        x3[i] = x_3
        y3[i] = y_3
        z3[i] = z_3

    # Elementos 5 y 8 (punto 3 en plano paralelo a XZ)
    for i in [4,7]:
        # De la tabla de conectividad extraemos el nodo inicial
        # y el nodo final para el elemento
        nodo_i = conectividad.loc[i+1,1]
        nodo_f = conectividad.loc[i+1,2]
```

```
# De la tabla de coordenadas nodales podemos extraer
# las coordenadas de los nodos inicial y final
x_i = coordenadas_nodales.loc[nodo_i,'X']
x_f = coordenadas_nodales.loc[nodo_f,'X']
y_i = coordenadas_nodales.loc[nodo_i,'Y']
y_f = coordenadas_nodales.loc[nodo_f,'Y']
z_i = coordenadas_nodales.loc[nodo_i,'Z']
z_f = coordenadas_nodales.loc[nodo_f,'Z']


# Calculamos la longitud y los cosenos directores
le[i] = np.sqrt((x_f - x_i) ** 2 + (y_f - y_i) ** 2 + (z_f - z_i) ** 2)
l1[i] = (x_f - x_i) / le[i]
m1[i] = (y_f - y_i) / le[i]
n1[i] = (z_f - z_i) / le[i]

V_x = np.array([l1[i], m1[i], n1[i]])

# Punto de refenrecia 3, en este caso se trata de un plano que forma 30° con el plano xz
# El primer vector del plano es el que va del punto inicial al punto final
vec_1 = np.array([x_f-x_i, y_f-y_i, z_f-z_i])

# El segundo vector del plano se tomará como un vector unitario en el plano YZ que forma 30°
vec_2 = np.array([0,np.sin(30 * np.pi / 180), np.cos(30 * np.pi / 180)])

# El vector normal del plano será el producto cruz entre ellos 2
vec_norm = np.cross(vec_1,vec_2)

# Se escogen unas componentes X y Y cualesquiera
x_3 = np.random.random()
y_3 = np.random.random()

# Se calcula Z con la ecuación del plano usando el vector normal y el punto inicial
z_3 = z_i - (vec_norm[1] * (y_3 - y_i) + vec_norm[0] * (x_3 - x_i)) / vec_norm[2]

le_13 = np.sqrt((x_3 - x_i) ** 2 + (y_3 - y_i) ** 2 + (z_3 - z_i) ** 2)

# Costruimos el vector de cosenos directores de referencia V_13
V_13 = np.array([(x_3 - x_i) / le_13, (y_3 - y_i) / le_13, (z_3 - z_i) / le_13])

# Calculamos V_z
cross_vx_v13 = np.cross(V_x,V_13)
V_z = (cross_vx_v13) / np.sqrt(np.dot(cross_vx_v13,cross_vx_v13))

# Calculamos V_y
V_y = np.cross(V_z, V_x)

# Llenamos los vectores l2, m2, n2, l3, m3, n3, x3, y3 y z3
l2[i] = V_y[0]
m2[i] = V_y[1]
n2[i] = V_y[2]

l3[i] = V_z[0]
m3[i] = V_z[1]
```

```
        n3[i] = V_z[2]

        x3[i] = x_3
        y3[i] = y_3
        z3[i] = z_3

    # Elementos 6 y 7 (punto 3 en plano paralelo a YZ)
    for i in [5,6]:
        # De la tabla de conectividad extraemos el nodo inicial
        # y el nodo final para el elemento
        nodo_i = conectividad.loc[i+1,1]
        nodo_f = conectividad.loc[i+1,2]

        # De la tabla de coordenadas nodales podemos extraer
        # las coordenadas de los nodos inicial y final
        x_i = coordenadas_nodales.loc[nodo_i,'X']
        x_f = coordenadas_nodales.loc[nodo_f,'X']
        y_i = coordenadas_nodales.loc[nodo_i,'Y']
        y_f = coordenadas_nodales.loc[nodo_f,'Y']
        z_i = coordenadas_nodales.loc[nodo_i,'Z']
        z_f = coordenadas_nodales.loc[nodo_f,'Z']


        # Calculamos la longitud y los cosenos directores
        le[i] = np.sqrt((x_f - x_i) ** 2 + (y_f - y_i) ** 2 + (z_f - z_i) ** 2)
        l1[i] = (x_f - x_i) / le[i]
        m1[i] = (y_f - y_i) / le[i]
        n1[i] = (z_f - z_i) / le[i]

        V_x = np.array([l1[i], m1[i], n1[i]])

        # Punto de refenrecia 3, en este caso se trata de un plano que forma 30° con el plano YZ
        # El primer vector del plano es el que va del punto inicial al punto final
        vec_1 = np.array([x_f-x_i, y_f-y_i, z_f-z_i])

        # El segundo vector del plano se tomará como un vector unitario en el plano XZ que forma 30°
        vec_2 = np.array([np.cos(30 * np.pi / 180), 0, np.sin(30 * np.pi / 180)])

        # El vector normal del plano será el producto cruz entre ellos 2
        vec_norm = np.cross(vec_1,vec_2)

        # Se escogen unas componentes X y Y cualesquiera
        x_3 = np.random.random()
        y_3 = np.random.random()

        # Se calcula Z con la ecuación del plano usando el vector normal y el punto inicial
        z_3 = z_i - (vec_norm[1] * (y_3 - y_i) + vec_norm[0] * (x_3 - x_i)) / vec_norm[2]

        le_13 = np.sqrt((x_3 - x_i) ** 2 + (y_3 - y_i) ** 2 + (z_3 - z_i) ** 2)

        # Costruimos el vector de cosenos directores de referencia V_13
        V_13 = np.array([(x_3 - x_i) / le_13, (y_3 - y_i) / le_13, (z_3 - z_i) / le_13])

        # Calculamos V_z
```

```
            cross_vx_v13 = np.cross(V_x,V_13)
            V_z = (cross_vx_v13) / np.sqrt(np.dot(cross_vx_v13,cross_vx_v13))

            # Calculamos V_y
            V_y = np.cross(V_z, V_x)

            # Llenamos los vectores l2, m2, n2, l3, m3, n3, x3, y3 y z3
            l2[i] = V_y[0]
            m2[i] = V_y[1]
            n2[i] = V_y[2]

            l3[i] = V_z[0]
            m3[i] = V_z[1]
            n3[i] = V_z[2]

            x3[i] = x_3
            y3[i] = y_3
            z3[i] = z_3

        # Construimos la tabla de cosenos directores
        cosenos_dir = pd.DataFrame({'le':le,
                                    'l1' :l1,
                                    'm1' :m1,
                                    'n1' :n1,
                                    'l2' :l2,
                                    'm2' :m2,
                                    'n2' :n2,
                                    'l3' :l3,
                                    'm3' :m3,
                                    'n3' :n3,
                                    'x3' :x3,
                                    'y3' :y3,
                                    'z3' :z3},
                                    index = [1, 2, 3, 4, 5, 6, 7, 8])
        print('Tabla de cosenos directores')
        pd.options.display.float_format = '{:,.3f}'.format  # Salida con tres decimales para tablas con
        display(cosenos_dir)
```

```
Tabla de cosenos directores
```

	l1	l2	l3	le	m1	m2	m3	n1	n2	n3	x3	\
1	0.000	0.500	-0.866	15.000	0.000	0.866	0.500	1.000	-0.000	0.000	0.093	
2	0.000	0.500	-0.866	15.000	0.000	0.866	0.500	1.000	-0.000	0.000	15.192	
3	0.000	-0.500	0.866	15.000	0.000	-0.866	-0.500	1.000	0.000	0.000	-5.490	
4	0.000	-0.500	0.866	15.000	0.000	-0.866	-0.500	1.000	0.000	0.000	9.692	
5	1.000	-0.000	0.000	15.000	0.000	0.500	-0.866	0.000	0.866	0.500	0.644	
6	0.000	0.866	0.500	10.000	1.000	-0.000	0.000	0.000	0.500	-0.866	0.287	
7	0.000	-0.866	-0.500	10.000	1.000	0.000	0.000	0.000	-0.500	0.866	0.098	
8	1.000	0.000	0.000	15.000	0.000	-0.500	0.866	0.000	-0.866	-0.500	0.967	

	y3	z3
1	0.160	0.448
2	0.332	0.489
3	0.490	0.441
4	0.806	0.810

```
5 0.801 16.387
6 0.526 15.166
7 0.480  6.396
8 0.416 -1.599
```

1.14 Matriz de rigidez del elemento

La matriz de rigidez k' de un elemento en el sistema coordenado local la podemos calcular como

$$k' = \begin{bmatrix}
a_s & 0 & 0 & 0 & 0 & 0 & -a_s & 0 & 0 & 0 \\
0 & 0 & & & & & & & & \\
0 & a_z & 0 & 0 & 0 & b_z & 0 & -a_z & 0 & 0 \\
0 & b_z & & & & & & & & \\
0 & 0 & a_y & 0 & -b_y & 0 & 0 & 0 & -a_y & 0 \\
-b_y & 0 & & & & & & & & \\
0 & 0 & 0 & t_s & 0 & 0 & 0 & 0 & 0 & -t_s \\
0 & 0 & & & & & & & & \\
0 & 0 & -b_y & 0 & c_y & 0 & 0 & 0 & b_y & 0 \\
d_y & 0 & & & & & & & & \\
0 & b_z & 0 & 0 & 0 & c_z & 0 & -b_z & 0 & 0 \\
0 & d_z & & & & & & & & \\
-a_s & 0 & 0 & 0 & 0 & 0 & a_s & 0 & 0 & 0 \\
0 & 0 & & & & & & & & \\
0 & -a_z & 0 & 0 & 0 & -b_z & 0 & a_z & 0 & 0 \\
0 & -b_z & & & & & & & & \\
0 & 0 & -a_y & 0 & b_y & 0 & 0 & 0 & c_y & 0 \\
b_y & 0 & & & & & & & & \\
0 & 0 & 0 & -t_s & 0 & 0 & 0 & 0 & 0 & t_s \\
0 & 0 & & & & & & & & \\
0 & 0 & -b_y & 0 & d_y & 0 & 0 & 0 & b_y & 0 \\
c_y & 0 & & & & & & & & \\
0 & b_z & 0 & 0 & 0 & d_z & 0 & -b_z & 0 & 0 \\
0 & c_z & & & & & & & &
\end{bmatrix}$$

Donde

$a_s = \frac{EA}{l_e}$

$t_s = \frac{GJ}{l_e}$

$a_z = \frac{12EI_z}{l_e^3}$

$b_z = \frac{6EI_z}{l_e^2}$

$c_z = \frac{4EI_z}{l_e}$

$d_z = \frac{2EI_z}{l_e}$

$a_y = \frac{12EI_y}{l_e^3}$

$b_y = \frac{6EI_y}{l_e^2}$

$c_y = \frac{4EI_y}{l_e}$

$d_y = \frac{2EI_y}{l_e}$

```
In [9]: def matrizDeRigidez(E, A, G, J, I_y, I_z, l_e):
            a_s = E * A / l_e
            t_s = G * J / l_e
            a_z = 12 * E * I_z / (l_e ** 3)
            b_z = 6 * E * I_z / (l_e ** 2)
            c_z = 4 * E * I_z / l_e
            d_z = 2 * E * I_z / l_e
```

```
    a_y = 12 * E * I_y / (l_e ** 3)
    b_y = 6 * E * I_y / (l_e ** 2)
    c_y = 4 * E * I_y / l_e
    d_y = 2 * E * I_y / l_e
    matriz = np.array([[a_s, 0, 0, 0, 0, 0, -a_s, 0, 0, 0, 0, 0],
                       [0, a_z, 0, 0, 0, b_z, 0, -a_z, 0, 0, 0, b_z],
                       [0, 0, a_y, 0, -b_y, 0 , 0, 0, -a_y, 0, -b_y, 0],
                       [0, 0, 0, t_s, 0, 0, 0, 0, 0, -t_s, 0, 0],
                       [0, 0, -b_y, 0, c_y, 0, 0, 0, b_y, 0, d_y, 0],
                       [0, b_z, 0, 0, 0, c_z, 0, -b_z, 0, 0, 0, d_z],
                       [-a_s, 0, 0, 0, 0, 0, a_s, 0, 0, 0, 0, 0],
                       [0, -a_z, 0, 0, 0, -b_z, 0, a_z, 0, 0, 0, -b_z],
                       [0, 0, -a_y, 0, b_y, 0, 0, 0, c_y, 0, b_y, 0],
                       [0, 0, 0, -t_s, 0, 0, 0, 0, 0, t_s, 0, 0],
                       [0, 0, -b_y, 0, d_y, 0, 0, 0, b_y, 0, c_y, 0],
                       [0, b_z, 0, 0, 0, d_z, 0, -b_z, 0, 0, 0, c_z]])

    pd.set_option('display.float_format', '{:.2g}'.format) # Notación científica
    tabla = pd.DataFrame(matriz,
                         columns = [1,2,3,4,5,6,7,8,9,10,11,12],
                         index = [1,2,3,4,5,6,7,8,9,10,11,12])
    return (matriz, tabla)
```

1.15 Primer elemento

```
In [10]: E = 30e6   # Módulo elástico en psi
         G = 12e6   # Módulo cortante en psi
         A = 6.0    # Área en in2
         I_y = 3.75 # in4
         I_z = 51.0 # in4
         J = 0.24   # in4

         (matriz_ele_1, tabla_ele_1) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[1,'le'])
         print('La matriz de rigidez de elemento 1 es: ')
         display(tabla_ele_1)
```

```
La matriz de rigidez de elemento 1 es:
```

	1	2	3	4	5	6	7	8
1	1.2e+07	0	0	0	0	0	-1.2e+07	0
2	0	5.4e+06	0	0	0	4.1e+07	0	-5.4e+06
3	0	0	4e+05	0	-3e+06	0	0	0
4	0	0	0	1.9e+05	0	0	0	0
5	0	0	-3e+06	0	3e+07	0	0	0
6	0	4.1e+07	0	0	0	4.1e+08	0	-4.1e+07
7	-1.2e+07	0	0	0	0	0	1.2e+07	0
8	0	-5.4e+06	0	0	0	-4.1e+07	0	5.4e+06
9	0	0	-4e+05	0	3e+06	0	0	0
10	0	0	0	-1.9e+05	0	0	0	0
11	0	0	-3e+06	0	1.5e+07	0	0	0
12	0	4.1e+07	0	0	0	2e+08	0	-4.1e+07

	9	10	11	12
1	0	0	0	0
2	0	0	0	4.1e+07

```
3  -4e+05        0  -3e+06        0
4       0 -1.9e+05       0        0
5   3e+06        0 1.5e+07        0
6       0        0       0    2e+08
7       0        0       0        0
8       0        0       0 -4.1e+07
9   3e+07        0   3e+06        0
10      0  1.9e+05       0        0
11  3e+06        0   3e+07        0
12      0        0       0  4.1e+08
```

1.15.1 Segundo elemento

```
In [11]: A = 6.0    # Área en in2
         I_y = 3.75 # in4
         I_z = 51.0 # in4
         J = 0.24   # in4

         (matriz_ele_2, tabla_ele_2) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[2,'le'])
         print('La matriz de rigidez de elemento 2 es: ')
         display(tabla_ele_2)
```

```
La matriz de rigidez de elemento 2 es:
```

```
         1        2      3        4       5        6        7        8   \
1   1.2e+07        0      0        0       0        0 -1.2e+07        0
2         0  5.4e+06      0        0       0  4.1e+07        0 -5.4e+06
3         0        0  4e+05        0  -3e+06        0        0        0
4         0        0      0  1.9e+05       0        0        0        0
5         0        0 -3e+06        0   3e+07        0        0        0
6         0  4.1e+07      0        0       0  4.1e+08        0 -4.1e+07
7  -1.2e+07        0      0        0       0        0  1.2e+07        0
8         0 -5.4e+06      0        0       0 -4.1e+07        0  5.4e+06
9         0        0 -4e+05        0   3e+06        0        0        0
10        0        0      0 -1.9e+05       0        0        0        0
11        0        0 -3e+06        0 1.5e+07        0        0        0
12        0  4.1e+07      0        0       0    2e+08        0 -4.1e+07

       9        10      11       12
1       0        0       0        0
2       0        0       0  4.1e+07
3  -4e+05        0  -3e+06        0
4       0 -1.9e+05       0        0
5   3e+06        0 1.5e+07        0
6       0        0       0    2e+08
7       0        0       0        0
8       0        0       0 -4.1e+07
9   3e+07        0   3e+06        0
10      0  1.9e+05       0        0
11  3e+06        0   3e+07        0
12      0        0       0  4.1e+08
```

1.15.2 Tercer elemento

```
In [12]: A = 6.0    # Área en in2
         I_y = 3.75 # in4
         I_z = 51.0 # in4
         J = 0.24   # in4

         (matriz_ele_3, tabla_ele_3) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[3,'le'])
         print('La matriz de rigidez de elemento 3 es: ')
         display(tabla_ele_3)
```

La matriz de rigidez de elemento 3 es:

	1	2	3	4	5	6	7	8
1	1.2e+07	0	0	0	0	0	-1.2e+07	0
2	0	5.4e+06	0	0	0	4.1e+07	0	-5.4e+06
3	0	0	4e+05	0	-3e+06	0	0	0
4	0	0	0	1.9e+05	0	0	0	0
5	0	0	-3e+06	0	3e+07	0	0	0
6	0	4.1e+07	0	0	0	4.1e+08	0	-4.1e+07
7	-1.2e+07	0	0	0	0	0	1.2e+07	0
8	0	-5.4e+06	0	0	0	-4.1e+07	0	5.4e+06
9	0	0	-4e+05	0	3e+06	0	0	0
10	0	0	0	-1.9e+05	0	0	0	0
11	0	0	-3e+06	0	1.5e+07	0	0	0
12	0	4.1e+07	0	0	0	2e+08	0	-4.1e+07

	9	10	11	12
1	0	0	0	0
2	0	0	0	4.1e+07
3	-4e+05	0	-3e+06	0
4	0	-1.9e+05	0	0
5	3e+06	0	1.5e+07	0
6	0	0	0	2e+08
7	0	0	0	0
8	0	0	0	-4.1e+07
9	3e+07	0	3e+06	0
10	0	1.9e+05	0	0
11	3e+06	0	3e+07	0
12	0	0	0	4.1e+08

1.15.3 Cuarto elemento

```
In [13]: A = 6.0    # Área en in2
         I_y = 3.75 # in4
         I_z = 51.0 # in4
         J = 0.24   # in4

         (matriz_ele_4, tabla_ele_4) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[4,'le'])
         print('La matriz de rigidez de elemento 4 es: ')
         display(tabla_ele_4)
```

La matriz de rigidez de elemento 4 es:

	1	2	3	4	5	6	7	8

```
1   1.2e+07        0      0        0       0        0 -1.2e+07        0
2         0  5.4e+06      0        0       0  4.1e+07        0 -5.4e+06
3         0        0  4e+05        0  -3e+06        0        0        0
4         0        0      0  1.9e+05       0        0        0        0
5         0        0 -3e+06        0   3e+07        0        0        0
6         0  4.1e+07      0        0       0  4.1e+08        0 -4.1e+07
7  -1.2e+07        0      0        0       0        0  1.2e+07        0
8         0 -5.4e+06      0        0       0 -4.1e+07        0  5.4e+06
9         0        0 -4e+05        0   3e+06        0        0        0
10        0        0      0 -1.9e+05       0        0        0        0
11        0        0 -3e+06        0 1.5e+07        0        0        0
12        0  4.1e+07      0        0       0    2e+08        0 -4.1e+07

       9        10      11       12
1       0        0       0        0
2       0        0       0  4.1e+07
3  -4e+05        0  -3e+06        0
4       0 -1.9e+05       0        0
5   3e+06        0 1.5e+07        0
6       0        0       0    2e+08
7       0        0       0        0
8       0        0       0 -4.1e+07
9   3e+07        0   3e+06        0
10      0  1.9e+05       0        0
11  3e+06        0   3e+07        0
12      0        0       0  4.1e+08
```

1.15.4 Quinto elemento

```
In [14]: A = 3.0    # Área en in2
         I_y = 1.26 # in4
         I_z = 17.0 # in4
         J = 0.08   # in4

         (matriz_ele_5, tabla_ele_5) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[5,'le'])
         print('La matriz de rigidez de elemento 5 es: ')
         display(tabla_ele_5)
```

```
La matriz de rigidez de elemento 5 es:

       1        2        3        4      5        6      7        8        9   \
1   6e+06        0        0        0      0        0 -6e+06        0        0
2       0  1.8e+06        0        0      0  1.4e+07      0 -1.8e+06        0
3       0        0  1.3e+05        0 -1e+06        0      0        0 -1.3e+05
4       0        0        0  6.4e+04      0        0      0        0        0
5       0        0   -1e+06        0  1e+07        0      0        0    1e+06
6       0  1.4e+07        0        0      0  1.4e+08      0 -1.4e+07        0
7  -6e+06        0        0        0      0        0  6e+06        0        0
8       0 -1.8e+06        0        0      0 -1.4e+07      0  1.8e+06        0
9       0        0 -1.3e+05        0  1e+06        0      0        0    1e+07
10      0        0        0 -6.4e+04      0        0      0        0        0
11      0        0   -1e+06        0  5e+06        0      0        0    1e+06
12      0  1.4e+07        0        0      0  6.8e+07      0 -1.4e+07        0
```

```
          10     11       12
1          0      0        0
2          0      0  1.4e+07
3          0 -1e+06        0
4  -6.4e+04      0        0
5          0  5e+06        0
6          0      0  6.8e+07
7          0      0        0
8          0      0 -1.4e+07
9          0  1e+06        0
10  6.4e+04      0        0
11         0  1e+07        0
12         0      0  1.4e+08
```

1.15.5 Sexto elemento

In [15]:
```
A = 3.0    # Área en in2
I_y = 1.26 # in4
I_z = 17.0 # in4
J = 0.08   # in4

(matriz_ele_6, tabla_ele_6) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[6,'le'])
print('La matriz de rigidez de elemento 6 es: ')
display(tabla_ele_6)
```

```
La matriz de rigidez de elemento 6 es:
```

```
       1        2        3        4        5        6      7        8   \
1   9e+06        0        0        0        0        0 -9e+06        0
2       0  6.1e+06        0        0        0  3.1e+07      0 -6.1e+06
3       0        0  4.5e+05        0 -2.3e+06        0      0        0
4       0        0        0  9.6e+04        0        0      0        0
5       0        0 -2.3e+06        0  1.5e+07        0      0        0
6       0  3.1e+07        0        0        0    2e+08      0 -3.1e+07
7  -9e+06        0        0        0        0        0  9e+06        0
8       0 -6.1e+06        0        0        0 -3.1e+07      0  6.1e+06
9       0        0 -4.5e+05        0  2.3e+06        0      0        0
10      0        0        0 -9.6e+04        0        0      0        0
11      0        0 -2.3e+06        0  7.6e+06        0      0        0
12      0  3.1e+07        0        0        0    1e+08      0 -3.1e+07

          9        10       11       12
1         0        0        0        0
2         0        0        0  3.1e+07
3  -4.5e+05        0 -2.3e+06        0
4         0 -9.6e+04        0        0
5   2.3e+06        0  7.6e+06        0
6         0        0        0    1e+08
7         0        0        0        0
8         0        0        0 -3.1e+07
9   1.5e+07        0  2.3e+06        0
10        0  9.6e+04        0        0
11  2.3e+06        0  1.5e+07        0
12        0        0        0    2e+08
```

1.16 Séptimo elemento

```
In [16]: A = 3.0    # Área en in2
         I_y = 1.26 # in4
         I_z = 17.0 # in4
         J = 0.08   # in4

         (matriz_ele_7, tabla_ele_7) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[7,'le'])
         print('La matriz de rigidez de elemento 7 es: ')
         display(tabla_ele_7)
```

La matriz de rigidez de elemento 7 es:

	1	2	3	4	5	6	7	8	\
1	9e+06	0	0	0	0	0	-9e+06	0	
2	0	6.1e+06	0	0	0	3.1e+07	0	-6.1e+06	
3	0	0	4.5e+05	0	-2.3e+06	0	0	0	
4	0	0	0	9.6e+04	0	0	0	0	
5	0	0	-2.3e+06	0	1.5e+07	0	0	0	
6	0	3.1e+07	0	0	0	2e+08	0	-3.1e+07	
7	-9e+06	0	0	0	0	0	9e+06	0	
8	0	-6.1e+06	0	0	0	-3.1e+07	0	6.1e+06	
9	0	0	-4.5e+05	0	2.3e+06	0	0	0	
10	0	0	0	-9.6e+04	0	0	0	0	
11	0	0	-2.3e+06	0	7.6e+06	0	0	0	
12	0	3.1e+07	0	0	0	1e+08	0	-3.1e+07	

	9	10	11	12
1	0	0	0	0
2	0	0	0	3.1e+07
3	-4.5e+05	0	-2.3e+06	0
4	0	-9.6e+04	0	0
5	2.3e+06	0	7.6e+06	0
6	0	0	0	1e+08
7	0	0	0	0
8	0	0	0	-3.1e+07
9	1.5e+07	0	2.3e+06	0
10	0	9.6e+04	0	0
11	2.3e+06	0	1.5e+07	0
12	0	0	0	2e+08

1.16.1 Octavo elemento

```
In [17]: A = 3.0    # Área en in2
         I_y = 1.26 # in4
         I_z = 17.0 # in4
         J = 0.08   # in4

         (matriz_ele_8, tabla_ele_8) = matrizDeRigidez(E, A, G, J, I_y, I_z, cosenos_dir.loc[8,'le'])
         print('La matriz de rigidez de elemento 8 es: ')
         display(tabla_ele_8)
```

La matriz de rigidez de elemento 8 es:

```
       1         2         3         4      5         6      7         8         9   \
1   6e+06         0         0         0      0         0 -6e+06         0         0
2       0  1.8e+06         0         0      0  1.4e+07      0 -1.8e+06         0
3       0         0  1.3e+05         0 -1e+06         0      0         0 -1.3e+05
4       0         0         0 6.4e+04      0         0      0         0         0
5       0         0  -1e+06         0  1e+07         0      0         0    1e+06
6       0  1.4e+07         0         0      0  1.4e+08      0 -1.4e+07         0
7  -6e+06         0         0         0      0         0  6e+06         0         0
8       0 -1.8e+06         0         0      0 -1.4e+07      0  1.8e+06         0
9       0         0 -1.3e+05         0  1e+06         0      0         0    1e+07
10      0         0         0 -6.4e+04      0         0      0         0         0
11      0         0  -1e+06         0  5e+06         0      0         0    1e+06
12      0  1.4e+07         0         0      0  6.8e+07      0 -1.4e+07         0

          10     11        12
1          0      0         0
2          0      0  1.4e+07
3          0 -1e+06         0
4  -6.4e+04      0         0
5          0  5e+06         0
6          0      0  6.8e+07
7          0      0         0
8          0      0 -1.4e+07
9          0  1e+06         0
10  6.4e+04      0         0
11         0  1e+07         0
12         0      0  1.4e+08
```

1.17 Matriz de transformación L

La matriz de transformación L está definida con base en la matriz λ como

$$L = \begin{bmatrix} \lambda & & & 0 \\ & \lambda & & \\ & & \lambda & \\ 0 & & & \lambda \end{bmatrix}$$

```
In [18]: def matrizDeTransformacionL(elemento):

             l1 = cosenos_dir.loc[elemento,'l1']
             m1 = cosenos_dir.loc[elemento,'m1']
             n1 = cosenos_dir.loc[elemento,'n1']
             l2 = cosenos_dir.loc[elemento,'l2']
             m2 = cosenos_dir.loc[elemento,'m2']
             n2 = cosenos_dir.loc[elemento,'n2']
             l3 = cosenos_dir.loc[elemento,'l3']
             m3 = cosenos_dir.loc[elemento,'m3']
             n3 = cosenos_dir.loc[elemento,'n3']

             matriz = np.array([[l1,m1,n1,0,0,0,0,0,0,0,0,0],
                                [l2,m2,n2,0,0,0,0,0,0,0,0,0],
                                [l3,m3,n3,0,0,0,0,0,0,0,0,0],
```

```
                                [0,0,0,l1,m1,n1,0,0,0,0,0,0],
                                [0,0,0,l2,m2,n2,0,0,0,0,0,0],
                                [0,0,0,l3,m3,n3,0,0,0,0,0,0],
                                [0,0,0,0,0,0,l1,m1,n1,0,0,0],
                                [0,0,0,0,0,0,l2,m2,n2,0,0,0],
                                [0,0,0,0,0,0,l3,m3,n3,0,0,0],
                                [0,0,0,0,0,0,0,0,0,l1,m1,n1],
                                [0,0,0,0,0,0,0,0,0,l2,m2,n2],
                                [0,0,0,0,0,0,0,0,0,l3,m3,n3]])

            pd.set_option('display.float_format', '{:.2g}'.format) # Notación científica
            tabla = pd.DataFrame(matriz,
                                 columns = [1,2,3,4,5,6,7,8,9,10,11,12],
                                 index = [1,2,3,4,5,6,7,8,9,10,11,12])
            return (matriz, tabla)
```

1.17.1 Primer elemento

```
In [19]: matriz_transformacion_L_1, tabla_transformacion_L_1 = matrizDeTransformacionL(1)
         print('La matriz de transformación L del elemento 1')
         display(tabla_transformacion_L_1)
```

```
La matriz de transformación L del elemento 1
```

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	0	0	0	0	0	0	0	0	0
2	0.5	0.87	-0	0	0	0	0	0	0	0	0	0
3	-0.87	0.5	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0.5	0.87	-0	0	0	0	0	0	0
6	0	0	0	-0.87	0.5	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0.5	0.87	-0	0	0	0
9	0	0	0	0	0	0	-0.87	0.5	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0.5	0.87	-0
12	0	0	0	0	0	0	0	0	0	-0.87	0.5	0

1.17.2 Segundo Elemento

```
In [20]: matriz_transformacion_L_2, tabla_transformacion_L_2 = matrizDeTransformacionL(2)
         print('La matriz de transformación L del elemento 2')
         display(tabla_transformacion_L_2)
```

```
La matriz de transformación L del elemento 2
```

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	0	0	0	0	0	0	0	0	0
2	0.5	0.87	-0	0	0	0	0	0	0	0	0	0
3	-0.87	0.5	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0.5	0.87	-0	0	0	0	0	0	0
6	0	0	0	-0.87	0.5	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0.5	0.87	-0	0	0	0

9	0	0	0	0	0	0	-0.87	0.5	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	0.5	0.87	-0
12	0	0	0	0	0	0	0	0	0	-0.87	0.5	0

1.17.3 Tercer elemento

```
In [21]: matriz_transformacion_L_3, tabla_transformacion_L_3 = matrizDeTransformacionL(3)
         print('La matriz de transformación L del elemento 3')
         display(tabla_transformacion_L_3)
```

La matriz de transformación L del elemento 3

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	0	0	0	0	0	0	0	0	0
2	-0.5	-0.87	0	0	0	0	0	0	0	0	0	0
3	0.87	-0.5	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	-0.5	-0.87	0	0	0	0	0	0	0
6	0	0	0	0.87	-0.5	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	-0.5	-0.87	0	0	0	0
9	0	0	0	0	0	0	0.87	-0.5	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	-0.5	-0.87	0
12	0	0	0	0	0	0	0	0	0	0.87	-0.5	0

1.17.4 Cuarto elemento

```
In [22]: matriz_transformacion_L_4, tabla_transformacion_L_4 = matrizDeTransformacionL(4)
         print('La matriz de transformación L del elemento 4')
         display(tabla_transformacion_L_4)
```

La matriz de transformación L del elemento 4

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	0	0	0	0	0	0	0	0	0
2	-0.5	-0.87	0	0	0	0	0	0	0	0	0	0
3	0.87	-0.5	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	-0.5	-0.87	0	0	0	0	0	0	0
6	0	0	0	0.87	-0.5	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	-0.5	-0.87	0	0	0	0
9	0	0	0	0	0	0	0.87	-0.5	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	0	0	0	-0.5	-0.87	0
12	0	0	0	0	0	0	0	0	0	0.87	-0.5	0

1.17.5 Quinto elemento

```
In [23]: matriz_transformacion_L_5, tabla_transformacion_L_5 = matrizDeTransformacionL(5)
         print('La matriz de transformación L del elemento 5')
         display(tabla_transformacion_L_5)
```

La matriz de transformación L del elemento 5

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	0	0	0	0	0	0	0	0	0	0
2	-0	0.5	0.87	0	0	0	0	0	0	0	0	0
3	0	-0.87	0.5	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	-0	0.5	0.87	0	0	0	0	0	0
6	0	0	0	0	-0.87	0.5	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	-0	0.5	0.87	0	0	0
9	0	0	0	0	0	0	0	-0.87	0.5	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	-0	0.5	0.87
12	0	0	0	0	0	0	0	0	0	0	-0.87	0.5

1.17.6 Sexto elemento

In [24]:
```
matriz_transformacion_L_6, tabla_transformacion_L_6 = matrizDeTransformacionL(6)
print('La matriz de transformación L del elemento 6')
display(tabla_transformacion_L_6)
```

La matriz de transformación L del elemento 6

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	0	0	0	0	0
2	0.87	-0	0.5	0	0	0	0	0	0	0	0	0
3	0.5	0	-0.87	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0.87	-0	0.5	0	0	0	0	0	0
6	0	0	0	0.5	0	-0.87	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0.87	-0	0.5	0	0	0
9	0	0	0	0	0	0	0.5	0	-0.87	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	0.87	-0	0.5
12	0	0	0	0	0	0	0	0	0	0.5	0	-0.87

1.17.7 Séptimo elemento

In [25]:
```
matriz_transformacion_L_7, tabla_transformacion_L_7 = matrizDeTransformacionL(7)
print('La matriz de transformación L del elemento 7')
display(tabla_transformacion_L_7)
```

La matriz de transformación L del elemento 7

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	0	0	0	0	0
2	-0.87	0	-0.5	0	0	0	0	0	0	0	0	0
3	-0.5	0	0.87	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	-0.87	0	-0.5	0	0	0	0	0	0
6	0	0	0	-0.5	0	0.87	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0

```
8      0   0    0     0   0    0 -0.87   0 -0.5     0   0    0
9      0   0    0     0   0    0  -0.5   0 0.87     0   0    0
10     0   0    0     0   0    0     0   0    0     0   1    0
11     0   0    0     0   0    0     0   0    0 -0.87   0 -0.5
12     0   0    0     0   0    0     0   0    0  -0.5   0 0.87
```

1.17.8 Octavo elemento

```
In [26]: matriz_transformacion_L_8, tabla_transformacion_L_8 = matrizDeTransformacionL(8)
         print('La matriz de transformación L del elemento 8')
         display(tabla_transformacion_L_8)
```

```
La matriz de transformación L del elemento 8
```

```
    1    2     3   4    5     6   7    8     9   10   11    12
1    1    0     0   0    0     0   0    0     0   0    0     0
2    0 -0.5 -0.87   0    0     0   0    0     0   0    0     0
3    0 0.87  -0.5   0    0     0   0    0     0   0    0     0
4    0    0     0   1    0     0   0    0     0   0    0     0
5    0    0     0   0 -0.5 -0.87   0    0     0   0    0     0
6    0    0     0   0 0.87  -0.5   0    0     0   0    0     0
7    0    0     0   0    0     0   1    0     0   0    0     0
8    0    0     0   0    0     0   0 -0.5 -0.87   0    0     0
9    0    0     0   0    0     0   0 0.87  -0.5   0    0     0
10   0    0     0   0    0     0   0    0     0   1    0     0
11   0    0     0   0    0     0   0    0     0   0 -0.5 -0.87
12   0    0     0   0    0     0   0    0     0   0 0.87  -0.5
```

1.18 Grados de libertad por cada nodo

Sea i el nodo inicial y j el nodo final. Sabemos que cada nodo tiene asociado 6 GDL, 3 de translación y 3 de rotación en cada eje. Como se tienen 8 nodos tendremos $8 * 6 = 48$ grados de libertad. Los grados de libertad de cada nodo se pueden calcular como sigue: Para i: $6i-5, 6i-4, 6i-3, 6i-2, 6i-1, 6i$ Para j: $6j-5, 6j-4, 6j-3, 6j-2, 6j-1, 6j$

```
In [27]: gdl = pd.DataFrame({' Traslacion_x': [6*1-5,6*2-5,6*3-5,6*4-5,6*5-5,6*6-5,6*7-5,6*8-5],
                             ' Traslacion_y': [6*1-4,6*2-4,6*3-4,6*4-4,6*5-4,6*6-4,6*7-4,6*8-4],
                             ' Traslacion_z': [6*1-3,6*2-3,6*3-3,6*4-3,6*5-3,6*6-3,6*7-3,6*8-3],
                             'Rotacion_x '  : [6*1-2,6*2-2,6*3-2,6*4-2,6*5-2,6*6-2,6*7-2,6*8-2],
                             'Rotacion_y '  : [6*1-1,6*2-1,6*3-1,6*4-1,6*5-1,6*6-1,6*7-1,6*8-1],
                             'Rotacion_z '  : [6*1,  6*2,  6*3,  6*4,  6*5,  6*6,  6*7,  6*8  ]},
                             index = [1,2,3,4,5,6,7,8])
         print("Grados de libertad correspondientes a cada elemento")
         display(gdl)
```

```
Grados de libertad correspondientes a cada elemento
```

```
    Traslacion_x   Traslacion_y   Traslacion_z  Rotacion_x   Rotacion_y   \
1              1              2             3           4            5
2              7              8             9          10           11
3             13             14            15          16           17
4             19             20            21          22           23
5             25             26            27          28           29
6             31             32            33          34           35
```

```
7             37             38             39           40           41
8             43             44             45           46           47

   Rotacion_z
1           6
2          12
3          18
4          24
5          30
6          36
7          42
8          48
```

1.19 Matriz de rigidez del elemento en coordenadas globales

Para obtener la matriz de rigidez del elemento en coordenadas globales utilizamos la matriz de transformación L

$k = L^T k' L$

Y los índices de la matriz vienen dados por los grados de libertad de los nodos inicial y final

1.19.1 Primer elemento

```
In [28]: matriz_rig_global_ele_1 = np.dot(np.dot(np.transpose(matriz_transformacion_L_1),
                                                 matriz_ele_1),matriz_transformacion_L_1)

         i = conectividad.loc[1,1]
         j = conectividad.loc[1,2]

         tabla_rig_global_ele_1 = pd.DataFrame(matriz_rig_global_ele_1,
                                  columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4
                                             ,6*j-3,6*j-2,6*j-1,6*j],
                                  index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                           6*j-3,6*j-2,6*j-1,6*j])

         display(tabla_rig_global_ele_1)
```

```
         1        2        3        4        5        6        25       26  \
1   1.7e+06  2.2e+06        0 -1.6e+07  1.2e+07        0 -1.7e+06 -2.2e+06
2   2.2e+06  4.2e+06        0 -3.1e+07  1.6e+07        0 -2.2e+06 -4.2e+06
3         0        0  1.2e+07        0        0        0        0        0
4  -1.6e+07 -3.1e+07        0  3.1e+08 -1.6e+08        0  1.6e+07  3.1e+07
5   1.2e+07  1.6e+07        0 -1.6e+08  1.2e+08        0 -1.2e+07 -1.6e+07
6         0        0        0        0        0  1.9e+05        0        0
25 -1.7e+06 -2.2e+06        0  1.6e+07 -1.2e+07        0  2.4e+07 -1.1e+07
26 -2.2e+06 -4.2e+06        0  3.1e+07 -1.6e+07        0 -1.1e+07  1.2e+07
27        0        0 -1.2e+07        0        0        0        0        0
28 -1.6e+07 -3.1e+07        0  1.6e+08 -8.2e+07        0  1.6e+07  3.1e+07
29  1.2e+07  1.6e+07        0 -8.2e+07  6.2e+07        0 -1.2e+07 -1.6e+07
30        0        0        0        0        0 -1.9e+05        0        0

         27       28       29       30
1         0 -1.6e+07  1.2e+07        0
2         0 -3.1e+07  1.6e+07        0
3  -1.2e+07        0        0        0
```

```
4         0  1.6e+08 -8.2e+07        0
5         0 -8.2e+07  6.2e+07        0
6         0        0        0 -1.9e+05
25        0  1.6e+07 -1.2e+07        0
26        0  3.1e+07 -1.6e+07        0
27  1.2e+07        0        0        0
28        0  3.1e+08 -1.6e+08        0
29        0 -1.6e+08  1.2e+08        0
30        0        0        0  1.9e+05
```

1.19.2 Segundo elemento

```
In [29]: matriz_rig_global_ele_2 = np.dot(np.dot(np.transpose(matriz_transformacion_L_2),
                                                 matriz_ele_2),matriz_transformacion_L_2)

         i = conectividad.loc[2,1]
         j = conectividad.loc[2,2]

         tabla_rig_global_ele_2 = pd.DataFrame(matriz_rig_global_ele_2,
                                  columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,
                                             6*j-4,6*j-3,6*j-2,6*j-1,6*j],
                                  index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,
                                           6*j-4,6*j-3,6*j-2,6*j-1,6*j])

         display(tabla_rig_global_ele_2)
```

```
         7        8        9        10       11       12       31       32  \
7   1.7e+06  2.2e+06        0 -1.6e+07  1.2e+07        0 -1.7e+06 -2.2e+06
8   2.2e+06  4.2e+06        0 -3.1e+07  1.6e+07        0 -2.2e+06 -4.2e+06
9         0        0  1.2e+07        0        0        0        0        0
10 -1.6e+07 -3.1e+07        0  3.1e+08 -1.6e+08        0  1.6e+07  3.1e+07
11  1.2e+07  1.6e+07        0 -1.6e+08  1.2e+08        0 -1.2e+07 -1.6e+07
12        0        0        0        0        0  1.9e+05        0        0
31 -1.7e+06 -2.2e+06        0  1.6e+07 -1.2e+07        0  2.4e+07 -1.1e+07
32 -2.2e+06 -4.2e+06        0  3.1e+07 -1.6e+07        0 -1.1e+07  1.2e+07
33        0        0 -1.2e+07        0        0        0        0        0
34 -1.6e+07 -3.1e+07        0  1.6e+08 -8.2e+07        0  1.6e+07  3.1e+07
35  1.2e+07  1.6e+07        0 -8.2e+07  6.2e+07        0 -1.2e+07 -1.6e+07
36        0        0        0        0        0 -1.9e+05        0        0

         33       34       35       36
7         0 -1.6e+07  1.2e+07        0
8         0 -3.1e+07  1.6e+07        0
9  -1.2e+07        0        0        0
10        0  1.6e+08 -8.2e+07        0
11        0 -8.2e+07  6.2e+07        0
12        0        0        0 -1.9e+05
31        0  1.6e+07 -1.2e+07        0
32        0  3.1e+07 -1.6e+07        0
33  1.2e+07        0        0        0
34        0  3.1e+08 -1.6e+08        0
35        0 -1.6e+08  1.2e+08        0
36        0        0        0  1.9e+05
```

1.19.3 Tercer elemento

```
In [30]: matriz_rig_global_ele_3 = np.dot(np.dot(np.transpose(matriz_transformacion_L_3),
                                                 matriz_ele_3),matriz_transformacion_L_3)

         i = conectividad.loc[3,1]
         j = conectividad.loc[3,2]

         tabla_rig_global_ele_3 = pd.DataFrame(matriz_rig_global_ele_3,
                                  columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                             6*j-3,6*j-2,6*j-1,6*j],
                                  index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                           6*j-3,6*j-2,6*j-1,6*j])

         display(tabla_rig_global_ele_3)
```

	13	14	15	16	17	18	37	38
13	1.7e+06	2.2e+06	0	-1.6e+07	1.2e+07	0	-1.7e+06	-2.2e+06
14	2.2e+06	4.2e+06	0	-3.1e+07	1.6e+07	0	-2.2e+06	-4.2e+06
15	0	0	1.2e+07	0	0	0	0	0
16	-1.6e+07	-3.1e+07	0	3.1e+08	-1.6e+08	0	1.6e+07	3.1e+07
17	1.2e+07	1.6e+07	0	-1.6e+08	1.2e+08	0	-1.2e+07	-1.6e+07
18	0	0	0	0	0	1.9e+05	0	0
37	-1.7e+06	-2.2e+06	0	1.6e+07	-1.2e+07	0	2.4e+07	-1.1e+07
38	-2.2e+06	-4.2e+06	0	3.1e+07	-1.6e+07	0	-1.1e+07	1.2e+07
39	0	0	-1.2e+07	0	0	0	0	0
40	-1.6e+07	-3.1e+07	0	1.6e+08	-8.2e+07	0	1.6e+07	3.1e+07
41	1.2e+07	1.6e+07	0	-8.2e+07	6.2e+07	0	-1.2e+07	-1.6e+07
42	0	0	0	0	0	-1.9e+05	0	0

	39	40	41	42
13	0	-1.6e+07	1.2e+07	0
14	0	-3.1e+07	1.6e+07	0
15	-1.2e+07	0	0	0
16	0	1.6e+08	-8.2e+07	0
17	0	-8.2e+07	6.2e+07	0
18	0	0	0	-1.9e+05
37	0	1.6e+07	-1.2e+07	0
38	0	3.1e+07	-1.6e+07	0
39	1.2e+07	0	0	0
40	0	3.1e+08	-1.6e+08	0
41	0	-1.6e+08	1.2e+08	0
42	0	0	0	1.9e+05

1.19.4 Cuarto elemento

```
In [31]: matriz_rig_global_ele_4 = np.dot(np.dot(np.transpose(matriz_transformacion_L_4),
                                                 matriz_ele_4),matriz_transformacion_L_4)

         i = conectividad.loc[4,1]
         j = conectividad.loc[4,2]

         tabla_rig_global_ele_4 = pd.DataFrame(matriz_rig_global_ele_4,
                                  columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
```

```
                                        6*j-3,6*j-2,6*j-1,6*j],
                                index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                         6*j-3,6*j-2,6*j-1,6*j])

        display(tabla_rig_global_ele_4)
```

```
         19       20       21       22       23       24       43       44  \
19  1.7e+06  2.2e+06        0 -1.6e+07  1.2e+07        0 -1.7e+06 -2.2e+06
20  2.2e+06  4.2e+06        0 -3.1e+07  1.6e+07        0 -2.2e+06 -4.2e+06
21        0        0  1.2e+07        0        0        0        0        0
22 -1.6e+07 -3.1e+07        0  3.1e+08 -1.6e+08        0  1.6e+07  3.1e+07
23  1.2e+07  1.6e+07        0 -1.6e+08  1.2e+08        0 -1.2e+07 -1.6e+07
24        0        0        0        0        0  1.9e+05        0        0
43 -1.7e+06 -2.2e+06        0  1.6e+07 -1.2e+07        0  2.4e+07 -1.1e+07
44 -2.2e+06 -4.2e+06        0  3.1e+07 -1.6e+07        0 -1.1e+07  1.2e+07
45        0        0 -1.2e+07        0        0        0        0        0
46 -1.6e+07 -3.1e+07        0  1.6e+08 -8.2e+07        0  1.6e+07  3.1e+07
47  1.2e+07  1.6e+07        0 -8.2e+07  6.2e+07        0 -1.2e+07 -1.6e+07
48        0        0        0        0        0 -1.9e+05        0        0

         45       46       47       48
19        0 -1.6e+07  1.2e+07        0
20        0 -3.1e+07  1.6e+07        0
21 -1.2e+07        0        0        0
22        0  1.6e+08 -8.2e+07        0
23        0 -8.2e+07  6.2e+07        0
24        0        0        0 -1.9e+05
43        0  1.6e+07 -1.2e+07        0
44        0  3.1e+07 -1.6e+07        0
45  1.2e+07        0        0        0
46        0  3.1e+08 -1.6e+08        0
47        0 -1.6e+08  1.2e+08        0
48        0        0        0  1.9e+05
```

1.19.5 Quinto elemento

```
In [32]: matriz_rig_global_ele_5 = np.dot(np.dot(np.transpose(matriz_transformacion_L_5),
                                                 matriz_ele_5),matriz_transformacion_L_5)

         i = conectividad.loc[5,1]
         j = conectividad.loc[5,2]

         tabla_rig_global_ele_5 = pd.DataFrame(matriz_rig_global_ele_5,
                                 columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                            6*j-3,6*j-2,6*j-1,6*j],
                                 index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                          6*j-3,6*j-2,6*j-1,6*j])

         display(tabla_rig_global_ele_5)
```

```
       25       26       27       28       29       30     31       32  \
25  6e+06        0        0        0        0        0 -6e+06        0
26      0  5.5e+05  7.3e+05        0 -5.5e+06  4.2e+06      0 -5.5e+05
27      0  7.3e+05  1.4e+06        0   -1e+07  5.5e+06      0 -7.3e+05
```

```
28      0        0        0  6.4e+04        0        0      0        0
29      0 -5.5e+06   -1e+07        0    1e+08 -5.5e+07      0  5.5e+06
30      0  4.2e+06  5.5e+06        0 -5.5e+07  4.2e+07      0 -4.2e+06
31 -6e+06        0        0        0        0        0  6e+06        0
32      0 -5.5e+05 -7.3e+05        0  5.5e+06 -4.2e+06      0    8e+06
33      0 -7.3e+05 -1.4e+06        0    1e+07 -5.5e+06      0 -3.6e+06
34      0        0        0 -6.4e+04        0        0      0        0
35      0 -5.5e+06   -1e+07        0  5.2e+07 -2.7e+07      0  5.5e+06
36      0  4.2e+06  5.5e+06        0 -2.7e+07  2.1e+07      0 -4.2e+06

         33       34       35       36
25        0        0        0        0
26 -7.3e+05        0 -5.5e+06  4.2e+06
27 -1.4e+06        0   -1e+07  5.5e+06
28        0 -6.4e+04        0        0
29    1e+07        0  5.2e+07 -2.7e+07
30 -5.5e+06        0 -2.7e+07  2.1e+07
31        0        0        0        0
32 -3.6e+06        0  5.5e+06 -4.2e+06
33  3.9e+06        0    1e+07 -5.5e+06
34        0  6.4e+04        0        0
35    1e+07        0    1e+08 -5.5e+07
36 -5.5e+06        0 -5.5e+07  4.2e+07
```

1.19.6 Sexto elemento

```
In [33]: matriz_rig_global_ele_6 = np.dot(np.dot(np.transpose(matriz_transformacion_L_6),
                                                 matriz_ele_6),matriz_transformacion_L_6)

         i = conectividad.loc[6,1]
         j = conectividad.loc[6,2]

         tabla_rig_global_ele_6 = pd.DataFrame(matriz_rig_global_ele_6,
                                  columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                             6*j-3,6*j-2,6*j-1,6*j],
                                  index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                           6*j-3,6*j-2,6*j-1,6*j])

         display(tabla_rig_global_ele_6)
```

```
         25     26       27       28       29       30       37     38  \
25  4.7e+06      0  2.5e+06  1.2e+07        0 -2.4e+07 -4.7e+06      0
26        0  9e+06        0        0        0        0        0 -9e+06
27  2.5e+06      0  1.9e+06  9.4e+06        0 -1.2e+07 -2.5e+06      0
28  1.2e+07      0  9.4e+06  6.2e+07        0 -8.2e+07 -1.2e+07      0
29        0      0        0        0  9.6e+04        0        0      0
30 -2.4e+07      0 -1.2e+07 -8.2e+07        0  1.6e+08  2.4e+07      0
37 -4.7e+06      0 -2.5e+06 -1.2e+07        0  2.4e+07  8.4e+06      0
38        0 -9e+06        0        0        0        0        0  9e+06
39 -2.5e+06      0 -1.9e+06 -9.4e+06        0  1.2e+07 -3.9e+06      0
40  1.2e+07      0  9.4e+06  3.1e+07        0 -4.1e+07 -1.2e+07      0
41        0      0        0        0 -9.6e+04        0        0      0
42 -2.4e+07      0 -1.2e+07 -4.1e+07        0  7.8e+07  2.4e+07      0
```

```
         39       40       41       42
25 -2.5e+06  1.2e+07        0 -2.4e+07
26        0        0        0        0
27 -1.9e+06  9.4e+06        0 -1.2e+07
28 -9.4e+06  3.1e+07        0 -4.1e+07
29        0        0 -9.6e+04        0
30  1.2e+07 -4.1e+07        0  7.8e+07
37 -3.9e+06 -1.2e+07        0  2.4e+07
38        0        0        0        0
39  1.3e+07 -9.4e+06        0  1.2e+07
40 -9.4e+06  6.2e+07        0 -8.2e+07
41        0        0  9.6e+04        0
42  1.2e+07 -8.2e+07        0  1.6e+08
```

1.19.7 Séptimo elemento

```
In [34]: matriz_rig_global_ele_7 = np.dot(np.dot(np.transpose(matriz_transformacion_L_7),
                                                 matriz_ele_7),matriz_transformacion_L_7)

         i = conectividad.loc[7,1]
         j = conectividad.loc[7,2]

         tabla_rig_global_ele_7 = pd.DataFrame(matriz_rig_global_ele_7,
                                  columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                             6*j-3,6*j-2,6*j-1,6*j],
                                  index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                           6*j-3,6*j-2,6*j-1,6*j])

         display(tabla_rig_global_ele_7)
```

```
         31     32       33       34       35       36       43     44  \
31  4.7e+06      0  2.5e+06  1.2e+07        0 -2.4e+07 -4.7e+06      0
32        0  9e+06        0        0        0        0        0 -9e+06
33  2.5e+06      0  1.9e+06  9.4e+06        0 -1.2e+07 -2.5e+06      0
34  1.2e+07      0  9.4e+06  6.2e+07        0 -8.2e+07 -1.2e+07      0
35        0      0        0        0  9.6e+04        0        0      0
36 -2.4e+07      0 -1.2e+07 -8.2e+07        0  1.6e+08  2.4e+07      0
43 -4.7e+06      0 -2.5e+06 -1.2e+07        0  2.4e+07  8.4e+06      0
44        0 -9e+06        0        0        0        0        0  9e+06
45 -2.5e+06      0 -1.9e+06 -9.4e+06        0  1.2e+07 -3.9e+06      0
46  1.2e+07      0  9.4e+06  3.1e+07        0 -4.1e+07 -1.2e+07      0
47        0      0        0        0 -9.6e+04        0        0      0
48 -2.4e+07      0 -1.2e+07 -4.1e+07        0  7.8e+07  2.4e+07      0

         45       46       47       48
31 -2.5e+06  1.2e+07        0 -2.4e+07
32        0        0        0        0
33 -1.9e+06  9.4e+06        0 -1.2e+07
34 -9.4e+06  3.1e+07        0 -4.1e+07
35        0        0 -9.6e+04        0
36  1.2e+07 -4.1e+07        0  7.8e+07
43 -3.9e+06 -1.2e+07        0  2.4e+07
44        0        0        0        0
45  1.3e+07 -9.4e+06        0  1.2e+07
```

```
46 -9.4e+06  6.2e+07        0 -8.2e+07
47        0        0  9.6e+04        0
48  1.2e+07 -8.2e+07        0  1.6e+08
```

1.19.8 Octavo elemento

```
In [35]: matriz_rig_global_ele_8 = np.dot(np.dot(np.transpose(matriz_transformacion_L_8),
                                                  matriz_ele_8),matriz_transformacion_L_8)

         i = conectividad.loc[8,1]
         j = conectividad.loc[8,2]

         tabla_rig_global_ele_8 = pd.DataFrame(matriz_rig_global_ele_8,
                                  columns = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                             6*j-3,6*j-2,6*j-1,6*j],
                                  index = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,
                                           6*j-3,6*j-2,6*j-1,6*j])

         display(tabla_rig_global_ele_8)
```

```
       37       38       39       40       41       42     43       44  \
37  6e+06        0        0        0        0        0 -6e+06        0
38      0  5.5e+05  7.3e+05        0 -5.5e+06  4.2e+06      0 -5.5e+05
39      0  7.3e+05  1.4e+06        0   -1e+07  5.5e+06      0 -7.3e+05
40      0        0        0  6.4e+04        0        0      0        0
41      0 -5.5e+06   -1e+07        0    1e+08 -5.5e+07      0  5.5e+06
42      0  4.2e+06  5.5e+06        0 -5.5e+07  4.2e+07      0 -4.2e+06
43 -6e+06        0        0        0        0        0  6e+06        0
44      0 -5.5e+05 -7.3e+05        0  5.5e+06 -4.2e+06      0    8e+06
45      0 -7.3e+05 -1.4e+06        0    1e+07 -5.5e+06      0 -3.6e+06
46      0        0        0 -6.4e+04        0        0      0        0
47      0 -5.5e+06   -1e+07        0  5.2e+07 -2.7e+07      0  5.5e+06
48      0  4.2e+06  5.5e+06        0 -2.7e+07  2.1e+07      0 -4.2e+06

         45       46       47       48
37        0        0        0        0
38 -7.3e+05        0 -5.5e+06  4.2e+06
39 -1.4e+06        0   -1e+07  5.5e+06
40        0 -6.4e+04        0        0
41    1e+07        0  5.2e+07 -2.7e+07
42 -5.5e+06        0 -2.7e+07  2.1e+07
43        0        0        0        0
44 -3.6e+06        0  5.5e+06 -4.2e+06
45  3.9e+06        0    1e+07 -5.5e+06
46        0  6.4e+04        0        0
47    1e+07        0    1e+08 -5.5e+07
48 -5.5e+06        0 -5.5e+07  4.2e+07
```

1.20 Matriz de rigidez global

```
In [36]: # Se crea la matriz de rigidez global de todo el
         tabla_rig_estructura = pd.DataFrame(np.zeros((48,48)),
                                             index = np.arange(1,49),
```

```
                                columns = np.arange(1,49))

# Ubicamos la matriz del elemento 1
for i in tabla_rig_global_ele_1.index:
    for j in tabla_rig_global_ele_1.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_1.loc[i,j])

# 2
for i in tabla_rig_global_ele_2.index:
    for j in tabla_rig_global_ele_2.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_2.loc[i,j])

# 3
for i in tabla_rig_global_ele_3.index:
    for j in tabla_rig_global_ele_3.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_3.loc[i,j])

# 4
for i in tabla_rig_global_ele_4.index:
    for j in tabla_rig_global_ele_4.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_4.loc[i,j])

# 5
for i in tabla_rig_global_ele_5.index:
    for j in tabla_rig_global_ele_5.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_5.loc[i,j])

# 6
for i in tabla_rig_global_ele_6.index:
    for j in tabla_rig_global_ele_6.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_6.loc[i,j])

# 7
for i in tabla_rig_global_ele_7.index:
    for j in tabla_rig_global_ele_7.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_7.loc[i,j])

# 8
for i in tabla_rig_global_ele_8.index:
    for j in tabla_rig_global_ele_8.columns:
        tabla_rig_estructura.loc[i,j] = tabla_rig_estructura.loc[i,j] + (
            tabla_rig_global_ele_8.loc[i,j])



display(tabla_rig_estructura)
```

	1	2	3	4	5	6	7	8	\
1	1.7e+06	2.2e+06	0	-1.6e+07	1.2e+07	0	0	0	
2	2.2e+06	4.2e+06	0	-3.1e+07	1.6e+07	0	0	0	
3	0	0	1.2e+07	0	0	0	0	0	
4	-1.6e+07	-3.1e+07	0	3.1e+08	-1.6e+08	0	0	0	
5	1.2e+07	1.6e+07	0	-1.6e+08	1.2e+08	0	0	0	
6	0	0	0	0	0	1.9e+05	0	0	
7	0	0	0	0	0	0	1.7e+06	2.2e+06	
8	0	0	0	0	0	0	2.2e+06	4.2e+06	
9	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	-1.6e+07	-3.1e+07	
11	0	0	0	0	0	0	1.2e+07	1.6e+07	
12	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	
20	0	0	0	0	0	0	0	0	
21	0	0	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	0	
23	0	0	0	0	0	0	0	0	
24	0	0	0	0	0	0	0	0	
25	-1.7e+06	-2.2e+06	0	1.6e+07	-1.2e+07	0	0	0	
26	-2.2e+06	-4.2e+06	0	3.1e+07	-1.6e+07	0	0	0	
27	0	0	-1.2e+07	0	0	0	0	0	
28	-1.6e+07	-3.1e+07	0	1.6e+08	-8.2e+07	0	0	0	
29	1.2e+07	1.6e+07	0	-8.2e+07	6.2e+07	0	0	0	
30	0	0	0	0	0	-1.9e+05	0	0	
31	0	0	0	0	0	0	-1.7e+06	-2.2e+06	
32	0	0	0	0	0	0	-2.2e+06	-4.2e+06	
33	0	0	0	0	0	0	0	0	
34	0	0	0	0	0	0	-1.6e+07	-3.1e+07	
35	0	0	0	0	0	0	1.2e+07	1.6e+07	
36	0	0	0	0	0	0	0	0	
37	0	0	0	0	0	0	0	0	
38	0	0	0	0	0	0	0	0	
39	0	0	0	0	0	0	0	0	
40	0	0	0	0	0	0	0	0	
41	0	0	0	0	0	0	0	0	
42	0	0	0	0	0	0	0	0	
43	0	0	0	0	0	0	0	0	
44	0	0	0	0	0	0	0	0	
45	0	0	0	0	0	0	0	0	
46	0	0	0	0	0	0	0	0	
47	0	0	0	0	0	0	0	0	
48	0	0	0	0	0	0	0	0	

	9	10	...	39	40	41	42	43	\
1	0	0	...	0	0	0	0	0	
2	0	0	...	0	0	0	0	0	
3	0	0	...	0	0	0	0	0	

4	0	0	...	0	0	0	0	0
5	0	0	...	0	0	0	0	0
6	0	0	...	0	0	0	0	0
7	0	-1.6e+07	...	0	0	0	0	0
8	0	-3.1e+07	...	0	0	0	0	0
9	1.2e+07	0	...	0	0	0	0	0
10	0	3.1e+08	...	0	0	0	0	0
11	0	-1.6e+08	...	0	0	0	0	0
12	0	0	...	0	0	0	0	0
13	0	0	...	0	-1.6e+07	1.2e+07	0	0
14	0	0	...	0	-3.1e+07	1.6e+07	0	0
15	0	0	...	-1.2e+07	0	0	0	0
16	0	0	...	0	1.6e+08	-8.2e+07	0	0
17	0	0	...	0	-8.2e+07	6.2e+07	0	0
18	0	0	...	0	0	0	-1.9e+05	0
19	0	0	...	0	0	0	0	-1.7e+06
20	0	0	...	0	0	0	0	-2.2e+06
21	0	0	...	0	0	0	0	0
22	0	0	...	0	0	0	0	1.6e+07
23	0	0	...	0	0	0	0	-1.2e+07
24	0	0	...	0	0	0	0	0
25	0	0	...	-2.5e+06	1.2e+07	0	-2.4e+07	0
26	0	0	...	0	0	0	0	0
27	0	0	...	-1.9e+06	9.4e+06	0	-1.2e+07	0
28	0	0	...	-9.4e+06	3.1e+07	0	-4.1e+07	0
29	0	0	...	0	0	-9.6e+04	0	0
30	0	0	...	1.2e+07	-4.1e+07	0	7.8e+07	0
31	0	1.6e+07	...	0	0	0	0	-4.7e+06
32	0	3.1e+07	...	0	0	0	0	0
33	-1.2e+07	0	...	0	0	0	0	-2.5e+06
34	0	1.6e+08	...	0	0	0	0	-1.2e+07
35	0	-8.2e+07	...	0	0	0	0	0
36	0	0	...	0	0	0	0	2.4e+07
37	0	0	...	-3.9e+06	4.1e+06	-1.2e+07	2.4e+07	-6e+06
38	0	0	...	7.3e+05	3.1e+07	-2.2e+07	4.2e+06	0
39	0	0	...	2.6e+07	-9.4e+06	-1e+07	1.8e+07	0
40	0	0	...	-9.4e+06	3.8e+08	-1.6e+08	-8.2e+07	0
41	0	0	...	-1e+07	-1.6e+08	2.3e+08	-5.5e+07	0
42	0	0	...	1.8e+07	-8.2e+07	-5.5e+07	2e+08	0
43	0	0	...	0	0	0	0	3.8e+07
44	0	0	...	-7.3e+05	0	5.5e+06	-4.2e+06	-1.1e+07
45	0	0	...	-1.4e+06	0	1e+07	-5.5e+06	-3.9e+06
46	0	0	...	0	-6.4e+04	0	0	4.1e+06
47	0	0	...	-1e+07	0	5.2e+07	-2.7e+07	-1.2e+07
48	0	0	...	5.5e+06	0	-2.7e+07	2.1e+07	2.4e+07

	44	45	46	47	48
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0

```
8         0        0        0        0        0
9         0        0        0        0        0
10        0        0        0        0        0
11        0        0        0        0        0
12        0        0        0        0        0
13        0        0        0        0        0
14        0        0        0        0        0
15        0        0        0        0        0
16        0        0        0        0        0
17        0        0        0        0        0
18        0        0        0        0        0
19 -2.2e+06        0 -1.6e+07  1.2e+07        0
20 -4.2e+06        0 -3.1e+07  1.6e+07        0
21        0 -1.2e+07        0        0        0
22  3.1e+07        0  1.6e+08 -8.2e+07        0
23 -1.6e+07        0 -8.2e+07  6.2e+07        0
24        0        0        0        0 -1.9e+05
25        0        0        0        0        0
26        0        0        0        0        0
27        0        0        0        0        0
28        0        0        0        0        0
29        0        0        0        0        0
30        0        0        0        0        0
31        0 -2.5e+06  1.2e+07        0 -2.4e+07
32   -9e+06        0        0        0        0
33        0 -1.9e+06  9.4e+06        0 -1.2e+07
34        0 -9.4e+06  3.1e+07        0 -4.1e+07
35        0        0        0 -9.6e+04        0
36        0  1.2e+07 -4.1e+07        0  7.8e+07
37        0        0        0        0        0
38 -5.5e+05 -7.3e+05        0 -5.5e+06  4.2e+06
39 -7.3e+05 -1.4e+06        0   -1e+07  5.5e+06
40        0        0 -6.4e+04        0        0
41  5.5e+06    1e+07        0  5.2e+07 -2.7e+07
42 -4.2e+06 -5.5e+06        0 -2.7e+07  2.1e+07
43 -1.1e+07 -3.9e+06  4.1e+06 -1.2e+07  2.4e+07
44  2.9e+07 -3.6e+06  3.1e+07 -1.1e+07 -4.2e+06
45 -3.6e+06  2.9e+07 -9.4e+06    1e+07  6.8e+06
46  3.1e+07 -9.4e+06  3.8e+08 -1.6e+08 -8.2e+07
47 -1.1e+07    1e+07 -1.6e+08  2.3e+08 -5.5e+07
48 -4.2e+06  6.8e+06 -8.2e+07 -5.5e+07    2e+08

[48 rows x 48 columns]
```

1.21 Restricciones

Los nodos 1, 2, 3 y 4 están fijos por lo tanto podemos eliminar sus grados de libertad que son del 1 al 24.

```
In [37]: tabla_rig_estructura_sin_r = tabla_rig_estructura.loc[25:,25:]
         display(tabla_rig_estructura_sin_r)
```

```
         25       26       27       28       29       30       31       32  \
25  3.5e+07 -1.1e+07  2.5e+06  2.9e+07 -1.2e+07 -2.4e+07   -6e+06        0
26 -1.1e+07  2.1e+07  7.3e+05  3.1e+07 -2.2e+07  4.2e+06        0 -5.5e+05
```

```
27  2.5e+06  7.3e+05  1.5e+07  9.4e+06   -1e+07 -6.8e+06        0 -7.3e+05
28  2.9e+07  3.1e+07  9.4e+06  3.8e+08 -1.6e+08 -8.2e+07        0         0
29 -1.2e+07 -2.2e+07   -1e+07 -1.6e+08  2.3e+08 -5.5e+07        0  5.5e+06
30 -2.4e+07  4.2e+06 -6.8e+06 -8.2e+07 -5.5e+07    2e+08        0 -4.2e+06
31   -6e+06        0        0        0        0        0  3.5e+07 -1.1e+07
32        0 -5.5e+05 -7.3e+05        0  5.5e+06 -4.2e+06 -1.1e+07  2.9e+07
33        0 -7.3e+05 -1.4e+06        0    1e+07 -5.5e+06  2.5e+06 -3.6e+06
34        0        0        0 -6.4e+04        0        0  2.9e+07  3.1e+07
35        0 -5.5e+06   -1e+07        0  5.2e+07 -2.7e+07 -1.2e+07 -1.1e+07
36        0  4.2e+06  5.5e+06        0 -2.7e+07  2.1e+07 -2.4e+07 -4.2e+06
37 -4.7e+06        0 -2.5e+06 -1.2e+07        0  2.4e+07        0         0
38        0   -9e+06        0        0        0        0        0         0
39 -2.5e+06        0 -1.9e+06 -9.4e+06        0  1.2e+07        0         0
40  1.2e+07        0  9.4e+06  3.1e+07        0 -4.1e+07        0         0
41        0        0        0        0 -9.6e+04        0        0         0
42 -2.4e+07        0 -1.2e+07 -4.1e+07        0  7.8e+07        0         0
43        0        0        0        0        0        0 -4.7e+06         0
44        0        0        0        0        0        0        0   -9e+06
45        0        0        0        0        0        0 -2.5e+06         0
46        0        0        0        0        0        0  1.2e+07         0
47        0        0        0        0        0        0        0         0
48        0        0        0        0        0        0 -2.4e+07         0

         33       34   ...          39       40       41       42       43  \
25        0        0   ...    -2.5e+06  1.2e+07        0 -2.4e+07        0
26 -7.3e+05        0   ...           0        0        0        0        0
27 -1.4e+06        0   ...    -1.9e+06  9.4e+06        0 -1.2e+07        0
28        0 -6.4e+04   ...    -9.4e+06  3.1e+07        0 -4.1e+07        0
29    1e+07        0   ...           0        0 -9.6e+04        0        0
30 -5.5e+06        0   ...     1.2e+07 -4.1e+07        0  7.8e+07        0
31  2.5e+06  2.9e+07   ...           0        0        0        0 -4.7e+06
32 -3.6e+06  3.1e+07   ...           0        0        0        0        0
33  1.8e+07  9.4e+06   ...           0        0        0        0 -2.5e+06
34  9.4e+06  3.8e+08   ...           0        0        0        0 -1.2e+07
35    1e+07 -1.6e+08   ...           0        0        0        0        0
36 -1.8e+07 -8.2e+07   ...           0        0        0        0  2.4e+07
37        0        0   ...    -3.9e+06  4.1e+06 -1.2e+07  2.4e+07   -6e+06
38        0        0   ...     7.3e+05  3.1e+07 -2.2e+07  4.2e+06        0
39        0        0   ...     2.6e+07 -9.4e+06   -1e+07  1.8e+07        0
40        0        0   ...    -9.4e+06  3.8e+08 -1.6e+08 -8.2e+07        0
41        0        0   ...      -1e+07 -1.6e+08  2.3e+08 -5.5e+07        0
42        0        0   ...     1.8e+07 -8.2e+07 -5.5e+07    2e+08        0
43 -2.5e+06 -1.2e+07   ...           0        0        0        0  3.8e+07
44        0        0   ...    -7.3e+05        0  5.5e+06 -4.2e+06 -1.1e+07
45 -1.9e+06 -9.4e+06   ...    -1.4e+06        0    1e+07 -5.5e+06 -3.9e+06
46  9.4e+06  3.1e+07   ...           0 -6.4e+04        0        0  4.1e+06
47        0        0   ...      -1e+07        0  5.2e+07 -2.7e+07 -1.2e+07
48 -1.2e+07 -4.1e+07   ...     5.5e+06        0 -2.7e+07  2.1e+07  2.4e+07

         44       45       46       47       48
25        0        0        0        0        0
26        0        0        0        0        0
27        0        0        0        0        0
28        0        0        0        0        0
```

```
29        0        0        0        0        0
30        0        0        0        0        0
31        0 -2.5e+06  1.2e+07        0 -2.4e+07
32   -9e+06        0        0        0        0
33        0 -1.9e+06  9.4e+06        0 -1.2e+07
34        0 -9.4e+06  3.1e+07        0 -4.1e+07
35        0        0        0 -9.6e+04        0
36        0  1.2e+07 -4.1e+07        0  7.8e+07
37        0        0        0        0        0
38 -5.5e+05 -7.3e+05        0 -5.5e+06  4.2e+06
39 -7.3e+05 -1.4e+06        0   -1e+07  5.5e+06
40        0        0 -6.4e+04        0        0
41  5.5e+06    1e+07        0  5.2e+07 -2.7e+07
42 -4.2e+06 -5.5e+06        0 -2.7e+07  2.1e+07
43 -1.1e+07 -3.9e+06  4.1e+06 -1.2e+07  2.4e+07
44  2.9e+07 -3.6e+06  3.1e+07 -1.1e+07 -4.2e+06
45 -3.6e+06  2.9e+07 -9.4e+06    1e+07  6.8e+06
46  3.1e+07 -9.4e+06  3.8e+08 -1.6e+08 -8.2e+07
47 -1.1e+07    1e+07 -1.6e+08  2.3e+08 -5.5e+07
48 -4.2e+06  6.8e+06 -8.2e+07 -5.5e+07    2e+08

[24 rows x 24 columns]
```

1.22 Cargas

In [38]:
```
# Se crea el vector de cargas
F = np.zeros(48)

# Fuerza aplicada en el nodo 5 con direccion y corresponde al gdl 26
F[26-1] = 3000

# Fuerza aplicada en el nodo 6 con direccion y corresponde al gdl 32
F[32-1] = 3000

# Fuerza distribuida elemento 5, nodo i = 5, nodo f = 6, dirección -z
# gdl 25 a 30 para nodo 5 gdl 31 a 36 para nodo 6
p = -2.083332
F[27-1] = p * cosenos_dir.loc[5,'le'] / 2
F[29-1] = - p * (cosenos_dir.loc[5,'le']) **2 / 12
F[33-1] = p * cosenos_dir.loc[5,'le'] / 2
F[35-1] = - p * (cosenos_dir.loc[5,'le']) **2 / 12

# Fuerza distribuida elemento 6, nodo i = 5, nodo f = 7, dirección -z
# gdl 25 a 30 para nodo 5 gdl 37 a 42 para nodo 7
F[27-1] = F[27-1] + p * cosenos_dir.loc[6,'le'] / 2
F[28-1] = F[28-1] - p * (cosenos_dir.loc[6,'le']) **2 / 12
F[39-1] = F[39-1] + p * cosenos_dir.loc[6,'le'] / 2
F[40-1] = F[41-1] - p * (cosenos_dir.loc[6,'le']) **2 / 12

# Fuerza distribuida elemento 7, nodo i = 6, nodo f = 8, dirección -z
# gdl 31 a 36 para nodo 6 gdl 43 a 48 para nodo 8
F[33-1] = F[33-1] + p * cosenos_dir.loc[7,'le'] / 2
F[34-1] = F[34-1] - p * (cosenos_dir.loc[7,'le']) **2 / 12
```

```
        F[45-1] = F[45-1] + p * cosenos_dir.loc[7,'le'] / 2
        F[46-1] = F[47-1] - p * (cosenos_dir.loc[7,'le']) **2 / 12

        # Fuerza distribuida elemento 8, nodo i = 7, nodo f = 8, dirección -z
        # gdl 37 a 42 para nodo 7 gdl 43 a 48 para nodo 8
        F[39-1] = F[39-1] + p * cosenos_dir.loc[8,'le'] / 2
        F[41-1] = F[41-1] - p * (cosenos_dir.loc[8,'le']) **2 / 12
        F[45-1] = F[45-1] + p * cosenos_dir.loc[8,'le'] / 2
        F[47-1] = F[47-1] - p * (cosenos_dir.loc[8,'le']) **2 / 12

        # Eliminamos los gdl restringidos
        F = F[24:]
        print("el Vector de carga es")

        F_tabla = pd.DataFrame(F.reshape([4,6]), index = [5,6,7,8],
                               columns = [' Fuerza_x', ' Fuerza_y', ' Fuerza_z',
                                          'Momento_x ', 'Momento_y ', 'Momento_z '])

        pd.options.display.float_format = '{:,.6f}'.format  # Salida con seis decimales
        display(F_tabla)
```

```
el Vector de carga es
```

	Fuerza_x	Fuerza_y	Fuerza_z	Momento_x	Momento_y	Momento_z
5	0.000000	3,000.000000	-26.041650	17.361100	39.062475	0.000000
6	0.000000	3,000.000000	-26.041650	17.361100	39.062475	0.000000
7	0.000000	0.000000	-26.041650	17.361100	39.062475	0.000000
8	0.000000	0.000000	-26.041650	17.361100	39.062475	0.000000

1.23 Obtención de los desplazamientos

In [39]:
```
matriz_k = np.zeros([24,24])

        # se llenan con las filas y columnas
        for i in tabla_rig_estructura_sin_r.index:
            for j in tabla_rig_estructura_sin_r.columns:
                matriz_k[i-1-24,j-1-24] = tabla_rig_estructura_sin_r.loc[i,j]

        # para resolver importamos el módulo solve de la librería scipy.linag
        from scipy.linalg import solve

        # creamos la matriz de fuerza

        # resolvemos el problema F = kU obteniendo el desplazamiento
        desp = solve(matriz_k,F)

        desp = desp.reshape((4,6))
        desp_tabla = pd.DataFrame(desp, columns = ['Traslación X', 'Traslación Y', 'Traslación Z',
                                                   'Rotación X', 'Rotación Y', 'Rotación Z'],
                                        index = [5, 6, 7, 8])
        desp_rest = pd.DataFrame(np.zeros((4,6)), columns = ['Traslación X', 'Traslación Y',
                                                             'Traslación Z', 'Rotación X',
                                                             'Rotación Y', 'Rotación Z'],
                                        index = [1, 2, 3, 4])
```

```
        desp_tot = pd.concat([desp_rest, desp_tabla])
        pd.set_option('display.float_format', '{:.8g}'.format)
        display(desp_tot)
```

```
   Traslación X  Traslación Y  Traslación Z     Rotación X    Rotación Y  \
1             0             0             0              0             0
2             0             0             0              0             0
3             0             0             0              0             0
4             0             0             0              0             0
5 0.00027933947 0.00057306256 3.8596174e-05 -5.8338549e-05 1.4533345e-05
6 0.00016980816  0.0002650649 7.4006314e-05 -2.9486177e-05 6.8046076e-06
7 0.00017558027  0.0004113198 3.6711495e-05 -3.8301462e-05 1.3881656e-05
8 9.5238567e-05 0.00014734332 3.9715686e-05 -1.4461576e-05 7.3759234e-06

      Rotación Z
1              0
2              0
3              0
4              0
5 -1.5762671e-05
6  -4.165677e-06
7  -9.790664e-06
8  7.8937998e-08
```

1.24 Momentos flectores

In [40]:
```
from plotly.offline import download_plotlyjs, init_notebook_mode, iplot
import plotly.graph_objs as go
init_notebook_mode()


q_x = np.zeros((8,4))
q_y = np.zeros((8,4))
q_z = np.zeros((8,4))

e = np.linspace(-1,1)

for i in [1,2,3,4,5,6,7,8]:
    nodo_i = conectividad.loc[i,1]
    nodo_f = conectividad.loc[i,2]
    q_x[i-1, 0] = desp_tot.loc[nodo_i, 'Traslación X']
    q_x[i-1, 1] = desp_tot.loc[nodo_i, 'Rotación X']
    q_x[i-1, 2] = desp_tot.loc[nodo_f, 'Traslación X']
    q_x[i-1, 3] = desp_tot.loc[nodo_f, 'Rotación X']
    q_y[i-1, 0] = desp_tot.loc[nodo_i, 'Traslación Y']
    q_y[i-1, 1] = desp_tot.loc[nodo_i, 'Rotación Y']
    q_y[i-1, 2] = desp_tot.loc[nodo_f, 'Traslación Y']
    q_y[i-1, 3] = desp_tot.loc[nodo_f, 'Rotación Y']
    q_z[i-1, 0] = desp_tot.loc[nodo_i, 'Traslación Z']
    q_z[i-1, 1] = desp_tot.loc[nodo_i, 'Rotación Z']
    q_z[i-1, 2] = desp_tot.loc[nodo_f, 'Traslación Z']
    q_z[i-1, 3] = desp_tot.loc[nodo_f, 'Rotación Z']
```

```
        # Columnas
        # Elemento 1
        le_1 = cosenos_dir.loc[1, 'le']
        I_y = 3.75
        I_z = 51.0
        M_y_1 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_y[0,0] + (3 * e - 1) * le_1 * q_y[0,1]
                                             - 6 * e * q_y[0,2] + (3 * e + 1) * le_1 * q_y[0,3])
        M_z_1 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_x[0,0] + (3 * e - 1) * le_1 * q_x[0,1]
                                             - 6 * e * q_x[0,2] + (3 * e + 1) * le_1 * q_x[0,3])



        # Elemento 2
        le_1 = cosenos_dir.loc[2, 'le']
        I_y = 3.75
        I_z = 51.0
        M_y_2 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_y[1,0] + (3 * e - 1) * le_1 * q_y[1,1]
                                             - 6 * e * q_y[1,2] + (3 * e + 1) * le_1 * q_y[1,3])
        M_z_2 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_x[1,0] + (3 * e - 1) * le_1 * q_x[1,1]
                                             - 6 * e * q_x[1,2] + (3 * e + 1) * le_1 * q_x[1,3])

        # Elemento 3
        le_1 = cosenos_dir.loc[3, 'le']
        I_y = 3.75
        I_z = 51.0
        M_y_3 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_y[2,0] + (3 * e - 1) * le_1 * q_y[2,1]
                                             - 6 * e * q_y[2,2] + (3 * e + 1) * le_1 * q_y[2,3])
        M_z_3 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_x[2,0] + (3 * e - 1) * le_1 * q_x[2,1]
                                             - 6 * e * q_x[2,2] + (3 * e + 1) * le_1 * q_x[2,3])

        # Elemento 4
        le_1 = cosenos_dir.loc[4, 'le']
        I_y = 3.75
        I_z = 51.0
        M_y_4 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_y[3,0] + (3 * e - 1) * le_1 * q_y[3,1]
                                             - 6 * e * q_y[3,2] + (3 * e + 1) * le_1 * q_y[3,3])
        M_z_4 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_x[3,0] + (3 * e - 1) * le_1 * q_x[3,1]
                                             - 6 * e * q_x[3,2] + (3 * e + 1) * le_1 * q_x[3,3])
```

```
<IPython.core.display.HTML object>
```

```
In [41]: # Realizamos los gráficos

        iplot([go.Scatter(x = e, y = M_y_1, name = 'Momento en y elemento 1'),
               go.Scatter(x = e, y = M_y_2, name = 'Momento en y elemento 2'),
               go.Scatter(x = e, y = M_y_3, name = 'Momento en y elemento 3'),
               go.Scatter(x = e, y = M_y_4, name = 'Momento en y elemento 4')])
```

```
<IPython.core.display.HTML object>
```

```
In [42]: iplot([go.Scatter(x = e, y = M_z_1, name = 'Momento en z elemento 1'),
               go.Scatter(x = e, y = M_z_2, name = 'Momento en z elemento 2'),
               go.Scatter(x = e, y = M_z_3, name = 'Momento en z elemento 3'),
               go.Scatter(x = e, y = M_z_4, name = 'Momento en z elemento 4')])
```

```
<IPython.core.display.HTML object>
```

In [43]:
```
# Vigas

        # Elemento 5
        le_1 = cosenos_dir.loc[5, 'le']
        I_y = 1.26
        I_z = 17.0
        M_y_5 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_z[4,0] + (3 * e - 1) * le_1 * q_z[4,1]
                                             - 6 * e * q_z[4,2] + (3 * e + 1) * le_1 * q_z[4,3])
        M_z_5 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_y[4,0] + (3 * e - 1) * le_1 * q_y[4,1]
                                             - 6 * e * q_y[4,2] + (3 * e + 1) * le_1 * q_y[4,3])

        # Elemento 6
        le_1 = cosenos_dir.loc[6, 'le']
        I_y = 1.26
        I_z = 17.0
        M_y_6 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_z[5,0] + (3 * e - 1) * le_1 * q_z[5,1]
                                             - 6 * e * q_z[5,2] + (3 * e + 1) * le_1 * q_z[5,3])
        M_z_6 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_x[5,0] + (3 * e - 1) * le_1 * q_x[5,1]
                                             - 6 * e * q_x[5,2] + (3 * e + 1) * le_1 * q_x[5,3])

        # Elemento 7
        le_1 = cosenos_dir.loc[7, 'le']
        I_y = 1.26
        I_z = 17.0
        M_y_7 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_z[6,0] + (3 * e - 1) * le_1 * q_z[6,1]
                                             - 6 * e * q_z[6,2] + (3 * e + 1) * le_1 * q_z[6,3])
        M_z_7 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_y[6,0] + (3 * e - 1) * le_1 * q_y[6,1]
                                             - 6 * e * q_y[6,2] + (3 * e + 1) * le_1 * q_y[6,3])

        # Elemento 8
        le_1 = cosenos_dir.loc[8, 'le']
        I_y = 1.26
        I_z = 17.0
        M_y_8 = ((E * I_y) / (le_1) ** 2) * (6 * e * q_z[7,0] + (3 * e - 1) * le_1 * q_z[7,1]
                                             - 6 * e * q_z[7,2] + (3 * e + 1) * le_1 * q_z[7,3])
        M_z_8 = ((E * I_z) / (le_1) ** 2) * (6 * e * q_x[7,0] + (3 * e - 1) * le_1 * q_x[7,1]
                                             - 6 * e * q_x[7,2] + (3 * e + 1) * le_1 * q_x[7,3])
```

In [44]:
```
# Realizamos los gráficos

        iplot([go.Scatter(x = e, y = M_y_5, name = 'Momento en y elemento 5'),
               go.Scatter(x = e, y = M_y_6, name = 'Momento en y elemento 6'),
               go.Scatter(x = e, y = M_y_7, name = 'Momento en y elemento 7'),
               go.Scatter(x = e, y = M_y_8, name = 'Momento en y elemento 8')])
```

```
<IPython.core.display.HTML object>
```

In [45]:
```
iplot([go.Scatter(x = e, y = M_z_5, name = 'Momento en z elemento 5'),
       go.Scatter(x = e, y = M_z_6, name = 'Momento en z elemento 6'),
       go.Scatter(x = e, y = M_z_7, name = 'Momento en z elemento 7'),
       go.Scatter(x = e, y = M_z_8, name = 'Momento en z elemento 8')])
```

```
<IPython.core.display.HTML object>
```

1.25 Momento flector en las vigas

Para el cáculo del momento flector en las figas suponemos que las mismas se encuentran doblemente empotradas y la función que describe el momento flexionante será:

$$M_f(x) = -\frac{pL^2}{12} + \frac{px(L-x)}{2}$$

https://es.wikipedia.org/wiki/Anexo:Pendientes_y_deformaciones_en_vigas#Vigas_biempotradas

1.25.1 Vigas de 10 [in]

```
In [46]: L = 10
         p = fuerza_dist_viga

         x = np.linspace(0,L)
         y = - (p * L ** 2) / 12 +  (p * x * (L - x)) / 2

         iplot([go.Scatter(x = x, y = y, name = 'Momento en las vigas de 10 [in]')])
         print('El valor máximo de momento es de: ', np.max(y), '[in lbf]')
```

```
<IPython.core.display.HTML object>


El valor máximo de momento es de:  8.66970383174 [in lbf]
```

1.25.2 Vigas de 15 [in]

```
In [47]: L = 15
         p = fuerza_dist_viga

         x = np.linspace(0,L)
         y = - (p * L ** 2) / 12 +  (p * x * (L - x)) / 2

         iplot([go.Scatter(x = x, y = y, name = 'Momento en las vigas de 10 [in]')])
         print('El valor máximo de momento es de: ', np.max(y), '[in lbf]')
```

```
<IPython.core.display.HTML object>


El valor máximo de momento es de:  19.5068336214 [in lbf]
```

1.26 Uso del programa en Fortran FRAME3D.FOR (Chandrupatla y Belegundu)

1.26.1 Archivo de entrada input.dat

MARCO CON FUERZA DISTRIBUIDA << 3-D Frame Analysis >> PROBLEMA 8.14 NN NE NM NDIM NEN NDN NNREF 8 8 1 3 2 6 8 ND NL NCH NPR NMPC 24 2 2 0 Node# X Y Z 1 0 0 0 2 15 0 0 3 0 10 0 4 15 10 0 5 0 0 15 6 15 0 15 7 0 10 15 8 15 10 15 9 0.06558577 0.11359788 0.31757592 10 15.20990463 0.36356548 0.43839612 11 -5.57798575 0.33864527 0.54110147 12 9.48639779 0.45016083 0.24749095 13 0.30643551 0.94496036 16.63671935 14 0.49614481 0.47012508 15.28644934 15 0.17259530 0.73380952 6.43939391 16 0.41518595 0.48306162 -1.48382081 Elem# N1 N2 Ref_Pt Mat# Area Iy Iz J

UDLy' UDLz' (NCH = 4: Area, Iy, Iz, J) 1 1 5 9 1 6.0 3.75 51.0 0.24 0. 0. 2 2 6 10 1 6.0 3.75 51.0 0.24 0. 0. 3 3 7 11 1 6.0 3.75 51.0 0.24 0. 0. 4 4 8 12 1 6.0 3.75 51.0 0.24 0. 0. 5 5 6 13 1 3.0 1.26 17.0 0.08 0. -2.083332 6 5 7 14 1 3.0 1.26 17.0 0.08 0. -2.083332 7 6 8 15 1 3.0 1.26 17.0 0.08 0. -2.083332 8 7 8 16 1 3.0 1.26 17.0 0.08 0. -2.083332 DOF# Displacement 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0 10 0 11 0 12 0 13 0 14 0 15 0 16 0 17 0 18 0 19 0 20 0 21 0 22 0 23 0 24 0 DOF# Load 26 3000 32 3000 MAT# PROP1(E) PROP2(G) 1 30E6 12E6 B1 i B2 j B3 (multipoint constraint B1 * Q1 + B2 * Qj = B3)

1.26.2 Archivo de salida output.dat

Output for Input Data from File input.dat PROBLEMA 8.14 NODE# X-Displ. Y-Displ. Z-Rot. X-Rot. Y-Rot. Z-Rot. 1 -0.5616E-11 0.4070E-09 0.6459E-10 -0.5140E-08 0.8598E-09 -0.4326E-11 2 -0.1915E-11 0.4116E-09 0.3120E-09 -0.5220E-08 0.9078E-09 -0.4427E-11 3 0.2471E-11 0.3850E-09 -0.2957E-09 -0.4842E-08 0.8654E-09 -0.4079E-11 4 0.5060E-11 0.3926E-09 -0.8093E-10 -0.4932E-08 0.9050E-09 -0.4016E-11 5 -0.2119E-02 0.2223E-02 0.2023E-04 -0.1380E-03 -0.7250E-04 -0.8469E-04 6 -0.2119E-02 0.2244E-02 0.9774E-04 -0.1408E-03 -0.7374E-04 -0.8668E-04 7 -0.1936E-02 0.2060E-02 -0.9262E-04 -0.1288E-03 -0.6700E-04 -0.7985E-04 8 -0.1936E-02 0.2079E-02 -0.2535E-04 -0.1296E-03 -0.6568E-04 -0.7863E-04 DOF# Reaction 1 0.2111E+02 2 -0.1530E+04 3 -0.2428E+03 4 0.1932E+05 5 -0.3232E+04 6 0.1626E+02 7 0.7197E+01 8 -0.1547E+04 9 -0.1173E+04 10 0.1962E+05 11 -0.3412E+04 12 0.1664E+02 13 -0.9290E+01 14 -0.1447E+04 15 0.1111E+04 16 0.1820E+05 17 -0.3253E+04 18 0.1533E+02 19 -0.1902E+02 20 -0.1476E+04 21 0.3042E+03 22 0.1854E+05 23 -0.3402E+04 24 0.1510E+02 Member End-Forces Member # 1 -0.2428E+03 -0.1314E+04 -0.7832E+03 0.1626E+02 0.6862E+04 -0.1835E+05 0.2428E+03 0.1314E+04 0.7832E+03 -0.1626E+02 0.4885E+04 -0.1364E+04 Member # 2 -0.1173E+04 -0.1336E+04 -0.7799E+03 0.1664E+02 0.6856E+04 -0.1870E+05 0.1173E+04 0.1336E+04 0.7799E+03 -0.1664E+02 0.4842E+04 -0.1346E+04 Member # 3 0.1111E+04 0.1258E+04 0.7155E+03 0.1533E+02 -0.6284E+04 0.1739E+05 -0.1111E+04 -0.1258E+04 -0.7155E+03 -0.1533E+02 -0.4448E+04 0.1478E+04 Member # 4 0.3042E+03 0.1288E+04 0.7214E+03 0.1510E+02 -0.6323E+04 0.1776E+05 -0.3042E+03 -0.1288E+04 -0.7214E+03 -0.1510E+02 -0.4498E+04 0.1557E+04 Member # 5 0.8281E+00 0.4168E+03 0.2361E+03 0.1788E+00 -0.1687E+04 0.3123E+04 -0.8281E+00 -0.4168E+03 -0.2048E+03 -0.1788E+00 -0.1620E+04 0.3129E+04 Member # 6 0.1466E+04 -0.3433E+03 0.6353E+03 -0.5281E+00 -0.3181E+04 -0.1739E+04 -0.1466E+04 0.3433E+03 -0.6144E+03 0.5281E+00 -0.3067E+04 -0.1695E+04 Member # 7 0.1484E+04 0.3478E+03 -0.6184E+03 -0.7738E+00 0.3179E+04 0.1669E+04 -0.1484E+04 -0.3478E+03 0.6392E+03 0.7738E+00 0.3110E+04 0.1809E+04 Member # 8 0.5771E+00 -0.3626E+03 -0.1872E+03 0.5441E-01 0.1487E+04 -0.2738E+04 -0.5771E+00 0.3626E+03 0.2185E+03 -0.5441E-01 0.1556E+04 -0.2701E+04