

# ELEMENTOS FINITOS APLICADOS A LA MECÁNICA DE SÓLIDOS: PRIMERA ASIGNACIÓN

EDUARDO VIEIRA  
Universidad Central de Venezuela  
Facultad de Ingeniería  
Escuela de Ingeniería Mecánica  
eduardo.vieira@ucv.ve

## I. INTRODUCCIÓN

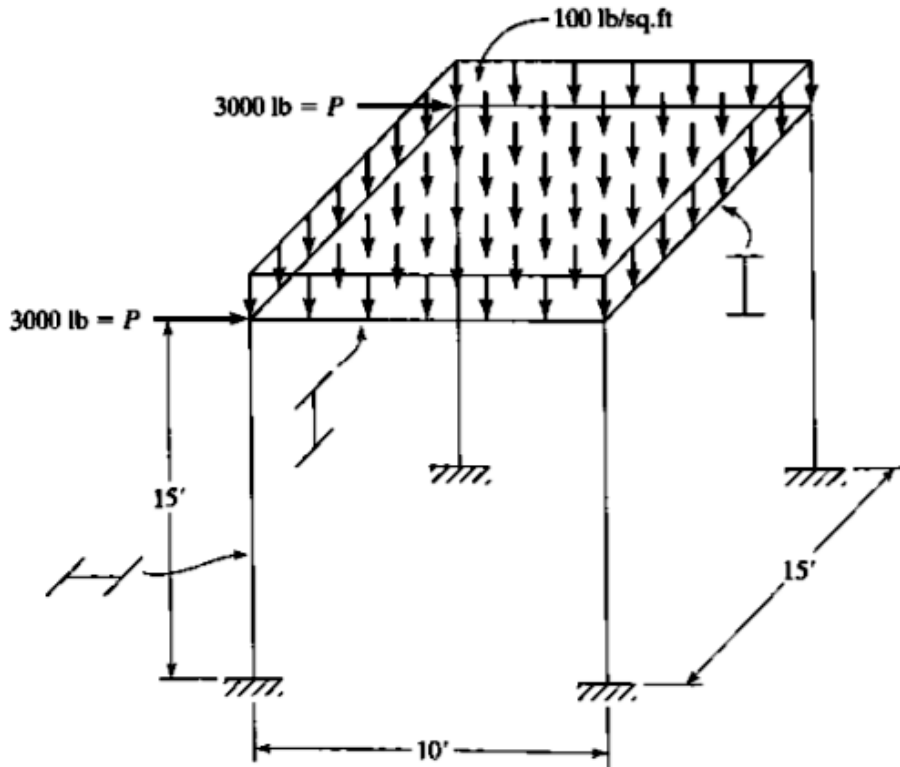
EN esta segunda asignación de la materia se resolverá el ejercicio propuesto 8,14 del libro Introducción al Estudio del Elemento Finito en Ingeniería (Tirupathi R. Chandrupatla y Ashok D. Belegundu, 1999, 2<sup>da</sup> Edición en español)

## II. EJERCICIO 8.14 PAG. 278

### I. 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 flexionantes en la estructura (máximo  $M_{y'}$  y  $M_{z'}$ ).

Se considera además que el perfil está rotado  $30^\circ$  al rededor del eje  $x'$



ejercicio 8.4

Figura 1: Diagrama del

## II. Datos

Área de la sección transversal e inercias del perfil usado

	Área	$I_{y'} [in^4]$	$I_{z'} [in^4]$	$J [in^4]$
Columnas	6.0	3.75	51.0	0.24
Vigas	3.0	1.26	17.0	0.08

**Tabla 1:** Área e inercias de los perfiles

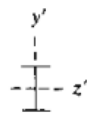


Figura 2: Perfil

## III. Solución

Para la solución del problema se utilizará el lenguaje de programación Python en la distribución científica Anaconda de Continuum Analytics (<https://www.continuum.io/downloads>) disponible para Windows, OS X y Linux.

Configuración utilizada:

Python 2.7.11

Anaconda 2.5.0

Linux 3.19.0-43-generic  
 Arquitectura x86\_64 (64 bit)  
 Elementary OS Freya 0.3 (Basado en Ubuntu 14.04)  
 Paquetes utilizados:  
 ipython 4.1.1  
 ipython-notebook 4.0.4  
 numpy 1.10.2  
 scipy 0.16.1  
 pandas 0.17.1  
 plotly 1.9.5

Al final se compila el código en Fortran 77 del programa FRAME3D.FOR extraído del libro Introducción al Estudio del Elemento Finito en Ingeniería (2<sup>da</sup> edición en español) por Chandrupatla y Belegundu.

Para la compilación se usó el software Intel Parallel Studio XE update 2 bajo Linux 64 bit con el uso de una licencia académica.

### III.1. 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.

```
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
        from IPython.display import display # Salida en HTML
```

Calculamos primero el área:

```
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]

El valor total de la fuerza lo podemos obtener multiplicando la fuerza distribuida por el área.

```
In [3]: fuerza_dist_area = 0.694444 # Valor de la fuerza distribuida sobre la plataforma [lb/in2]
        fuerza_tot = fuerza_dist_area * area
        print("La fuerza total es de: ", fuerza_tot, " [lb]")
```

La fuerza total es de: 104.1666 [lb]

La suma de las longitudes de las vigas será

```
In [4]: long_tot_vigas = 2 * A + 2 * B # Longitud total de las vigas [in]
        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]

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.

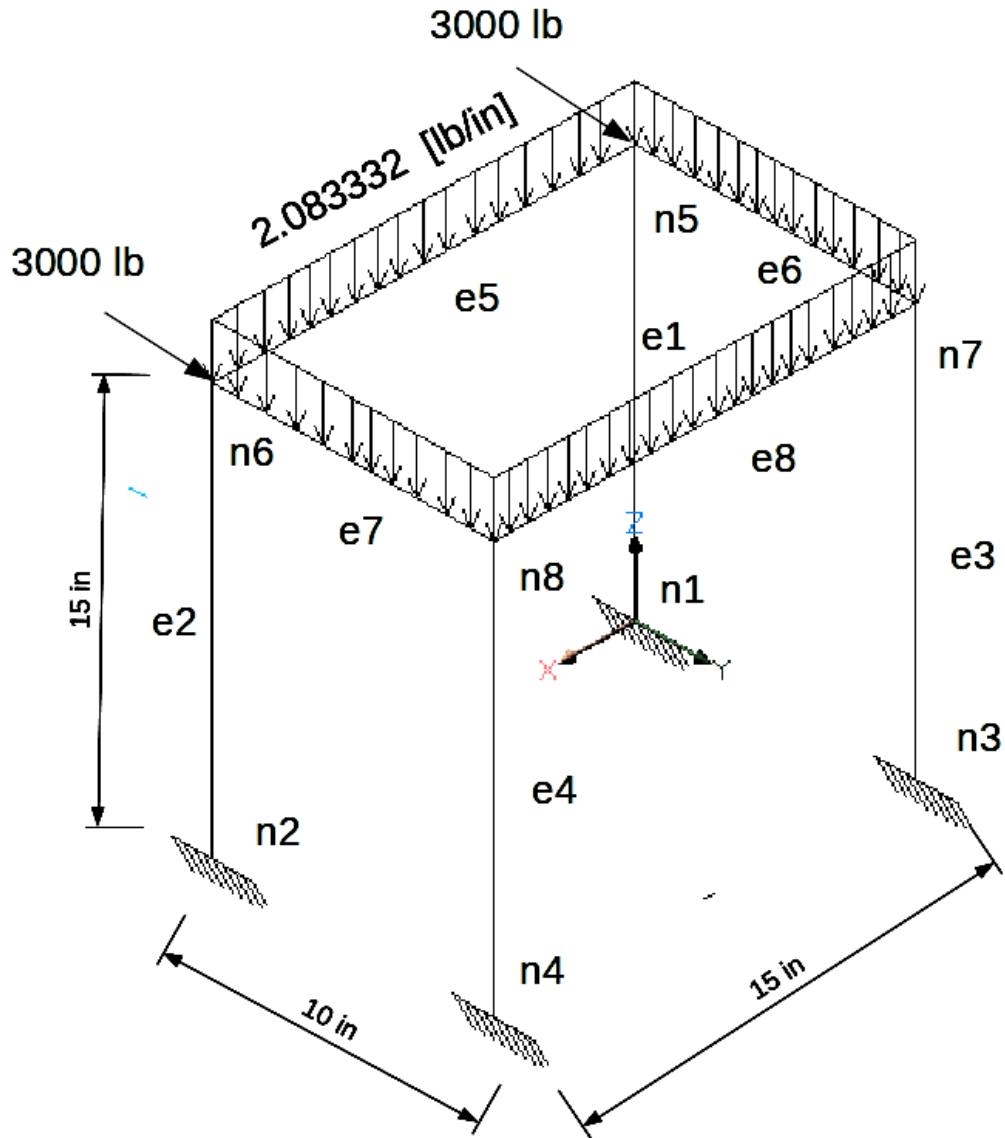


Figura 3: Carga distribuida sobre las vigas, elementos (denotados con la letra e) y nodos (denotados con la letra n) de la estructura

### III.2. Coordenadas nodales

De la figura 3 podemos obtener las coordenadas de los nodos

```
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

**Tabla 2:** *Coordenadas Nodales*

### III.3. 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

**Tabla 3:** *Conectividad*

### III.4. Cosenos directores

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 longitud 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 referencia 3 que está contenido en el plano del eje  $y'$  y la recta que une 12 pero no está contenido 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 unitario a lo largo del eje  $z$  está dado por:

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

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

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

Como en el problema el perfil se encuentra rotado  $30^\circ$  entorno al eje  $x'$  de cada elemento, el punto 3 se calcula construyendo un plano, el vector normal de este plano se obtiene haciendo producto cruz con el vector que forman los puntos inicial y final y otro vector que pertenezca al plano. Luego se genera un punto aleatorio perteneciente al plano

Para los elementos verticales 1, 2, 3 y 4 el punto 3 forma  $30^\circ$  con el plano  $YZ$ . Para los elementos 5 y 8 el punto 3 forma  $30^\circ$  con el plano  $YZ$

In [8]: # Número de elementos

```
ne = 8
```

```
# Creamos los vectores de los cosenos directores y de la longitud 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 referrencia 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° con el plano YZ
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)

# Construimos 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
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 referenciacia 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° con el plano XZ
    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)

# Construimos 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
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 referrencia 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° con el plano YZ
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)

# Construimos 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
display(cosenos_dir)

```

Tabla de cosenos directores

	$l_1$	$l_2$	$l_3$	$l_e$	$m_1$	$m_2$	$m_3$	$n_1$	$n_2$	$n_3$	$x_3$	$y_3$	$z_3$
1	0.000	0.500	-0.866	15.000	0.000	0.866	0.500	1.000	-0.000	0.000	0.480	0.831	0.876
2	0.000	0.500	-0.866	15.000	0.000	0.866	0.500	1.000	-0.000	0.000	15.279	0.483	0.999
3	0.000	-0.500	0.866	15.000	0.000	-0.866	-0.500	1.000	0.000	0.000	-5.399	0.649	0.117
4	0.000	-0.500	0.866	15.000	0.000	-0.866	-0.500	1.000	0.000	0.000	9.404	0.308	0.616
5	1.000	-0.000	0.000	15.000	0.000	0.500	-0.866	0.000	0.866	0.500	0.975	0.254	15.440
6	0.000	0.866	0.500	10.000	1.000	-0.000	0.000	0.000	0.500	-0.866	0.743	0.712	15.429
7	0.000	-0.866	-0.500	10.000	1.000	0.000	0.000	0.000	-0.500	0.866	0.201	0.483	6.456
8	1.000	0.000	0.000	15.000	0.000	-0.500	0.866	0.000	-0.866	-0.500	0.447	0.431	-1.574

Tabla 4: Tabla de cosenos directores

### III.5. Matriz de rigidez de los elementos

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

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

Donde

$$\begin{aligned} a_s &= \frac{EA}{l_e} \\ t_s &= \frac{GJ}{l_e} \\ a_z &= \frac{12EI_z}{l_e^3} \\ b_z &= \frac{6EI_z}{l_e^2} \\ c_z &= \frac{4EI_z}{l_e} \\ d_z &= \frac{2EI_z}{l_e} \\ a_y &= \frac{12EI_y}{l_e^3} \\ b_y &= \frac{6EI_y}{l_e^2} \\ c_y &= \frac{4EI_y}{l_e} \\ d_y &= \frac{2EI_y}{l_e} \end{aligned}$$

Definimos una función que nos calcula la matriz

```
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)
```

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

Tabla 5: Matriz de rigidez del elemento 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	9	10	11	12
1	1.2e+07	0	0	0	0	0	-1.2e+07	0	0	0	0	0
2	0	5.4e+06	0	0	0	4.1e+07	0	-5.4e+06	0	0	0	4.1e+07
3	0	0	4e+05	0	-3e+06	0	0	0	-4e+05	0	-3e+06	0
4	0	0	0	1.9e+05	0	0	0	0	0	-1.9e+05	0	0
5	0	0	-3e+06	0	3e+07	0	0	0	3e+06	0	1.5e+07	0
6	0	4.1e+07	0	0	0	4.1e+08	0	-4.1e+07	0	0	0	2e+08
7	-1.2e+07	0	0	0	0	0	1.2e+07	0	0	0	0	0
8	0	-5.4e+06	0	0	0	-4.1e+07	0	5.4e+06	0	0	0	-4.1e+07
9	0	0	-4e+05	0	3e+06	0	0	0	3e+07	0	3e+06	0
10	0	0	0	-1.9e+05	0	0	0	0	0	1.9e+05	0	0
11	0	0	-3e+06	0	1.5e+07	0	0	0	3e+06	0	3e+07	0
12	0	4.1e+07	0	0	0	2e+08	0	-4.1e+07	0	0	0	4.1e+08

Tabla 6: Matriz de rigidez del elemento 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	9	10	11	12
1	1.2e+07	0	0	0	0	0	-1.2e+07	0	0	0	0	0
2	0	5.4e+06	0	0	0	4.1e+07	0	-5.4e+06	0	0	0	4.1e+07
3	0	0	4e+05	0	-3e+06	0	0	0	-4e+05	0	-3e+06	0
4	0	0	0	1.9e+05	0	0	0	0	0	-1.9e+05	0	0
5	0	0	-3e+06	0	3e+07	0	0	0	3e+06	0	1.5e+07	0
6	0	4.1e+07	0	0	0	4.1e+08	0	-4.1e+07	0	0	0	2e+08
7	-1.2e+07	0	0	0	0	0	1.2e+07	0	0	0	0	0
8	0	-5.4e+06	0	0	0	-4.1e+07	0	5.4e+06	0	0	0	-4.1e+07
9	0	0	-4e+05	0	3e+06	0	0	0	3e+07	0	3e+06	0
10	0	0	0	-1.9e+05	0	0	0	0	0	1.9e+05	0	0
11	0	0	-3e+06	0	1.5e+07	0	0	0	3e+06	0	3e+07	0
12	0	4.1e+07	0	0	0	2e+08	0	-4.1e+07	0	0	0	4.1e+08

Tabla 7: Matriz de rigidez del elemento 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	9	10	11	12
1	1.2e+07	0	0	0	0	0	-1.2e+07	0	0	0	0	0
2	0	5.4e+06	0	0	0	4.1e+07	0	-5.4e+06	0	0	0	4.1e+07
3	0	0	4e+05	0	-3e+06	0	0	0	-4e+05	0	-3e+06	0
4	0	0	0	1.9e+05	0	0	0	0	0	-1.9e+05	0	0
5	0	0	-3e+06	0	3e+07	0	0	0	3e+06	0	1.5e+07	0
6	0	4.1e+07	0	0	0	4.1e+08	0	-4.1e+07	0	0	0	2e+08
7	-1.2e+07	0	0	0	0	0	1.2e+07	0	0	0	0	0
8	0	-5.4e+06	0	0	0	-4.1e+07	0	5.4e+06	0	0	0	-4.1e+07
9	0	0	-4e+05	0	3e+06	0	0	0	3e+07	0	3e+06	0
10	0	0	0	-1.9e+05	0	0	0	0	0	1.9e+05	0	0
11	0	0	-3e+06	0	1.5e+07	0	0	0	3e+06	0	3e+07	0
12	0	4.1e+07	0	0	0	2e+08	0	-4.1e+07	0	0	0	4.1e+08

**Tabla 8:** Matriz de rigidez del elemento 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	10	11	12
1	6e+06	0	0	0	0	0	-6e+06	0	0	0	0	0
2	0	1.8e+06	0	0	0	1.4e+07	0	-1.8e+06	0	0	0	1.4e+07
3	0	0	1.3e+05	0	-1e+06	0	0	0	-1.3e+05	0	-1e+06	0
4	0	0	0	6.4e+04	0	0	0	0	0	-6.4e+04	0	0
5	0	0	-1e+06	0	1e+07	0	0	0	1e+06	0	5e+06	0
6	0	1.4e+07	0	0	0	1.4e+08	0	-1.4e+07	0	0	0	6.8e+07
7	-6e+06	0	0	0	0	0	6e+06	0	0	0	0	0
8	0	-1.8e+06	0	0	0	-1.4e+07	0	1.8e+06	0	0	0	-1.4e+07
9	0	0	-1.3e+05	0	1e+06	0	0	0	1e+07	0	1e+06	0
10	0	0	0	-6.4e+04	0	0	0	0	0	6.4e+04	0	0
11	0	0	-1e+06	0	5e+06	0	0	0	1e+06	0	1e+07	0
12	0	1.4e+07	0	0	0	6.8e+07	0	-1.4e+07	0	0	0	1.4e+08

Tabla 9: Matriz de rigidez del elemento 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	9	10	11	12
1	9e+06	0	0	0	0	0	-9e+06	0	0	0	0	0
2	0	6.1e+06	0	0	0	3.1e+07	0	-6.1e+06	0	0	0	3.1e+07
3	0	0	4.5e+05	0	-2.3e+06	0	0	0	-4.5e+05	0	-2.3e+06	0
4	0	0	0	9.6e+04	0	0	0	0	0	-9.6e+04	0	0
5	0	0	-2.3e+06	0	1.5e+07	0	0	0	2.3e+06	0	7.6e+06	0
6	0	3.1e+07	0	0	0	2e+08	0	-3.1e+07	0	0	0	1e+08
7	-9e+06	0	0	0	0	0	9e+06	0	0	0	0	0
8	0	-6.1e+06	0	0	0	-3.1e+07	0	6.1e+06	0	0	0	-3.1e+07
9	0	0	-4.5e+05	0	2.3e+06	0	0	0	1.5e+07	0	2.3e+06	0
10	0	0	0	-9.6e+04	0	0	0	0	0	9.6e+04	0	0
11	0	0	-2.3e+06	0	7.6e+06	0	0	0	2.3e+06	0	1.5e+07	0
12	0	3.1e+07	0	0	0	1e+08	0	-3.1e+07	0	0	0	2e+08

Tabla 10: Matriz de rigidez del elemento 6

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

Tabla 11: Matriz de rigidez del elemento 7

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

Tabla 12: Matriz de rigidez del elemento 8

### III.6. Matriz de transformación $L$

Las matrices de rigidez de los elementos calculadas anteriormente están referidas al sistema coordenado local de cada elemento. Para transformarlas a el sistema coordenados global es necesario hacer uso de la matriz de transformación  $L$ .

La matriz de transformación  $L$  está definida con base en la matriz  $\lambda$  como

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

Donde la matriz  $\lambda$  es la matriz de cosenos directores:  $\lambda = \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix}$

Definimos una función que nos calcule la matriz de transformación  $L$

```
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)
```

### 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

Tabla 13: Matriz de transformación L del elemento 1

**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

Tabla 14: Matriz de transformación L del elemento 2

**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

Tabla 15: Matriz de transformación L del elemento 3

**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

Tabla 16: Matriz de transformación L del elemento 4

**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

Tabla 17: Matriz de transformación L del elemento 5

**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

Tabla 18: Matriz de transformación L del elemento 6

**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

Tabla 19: Matriz de transformación L del elemento 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

Tabla 20: Matriz de transformación L del elemento 8

**III.7. Grados de libertad de 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 [28]: 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

	Traslación X	Traslación Y	Traslación Z	Rotación X	Rotación Y	Rotación Z
1	1	2	3	4	5	6
2	7	8	9	10	11	12
3	13	14	15	16	17	18
4	19	20	21	22	23	24
5	25	26	27	28	29	30
6	31	32	33	34	35	36
7	37	38	39	40	41	42
8	43	44	45	46	47	48

**Tabla 21:** Grados de libertad de cada nodo

### III.8. 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

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

Tabla 22: Matriz de rigidez del elemento 1 en coordenadas globales

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

Tabla 23: Matriz de rigidez del elemento 2 en coordenadas globales

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

**Tabla 24:** Matriz de rigidez del elemento 3 en coordenadas globales

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

Tabla 25: Matriz de rigidez del elemento 4 en coordenadas globales

## 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	33	34	35	36
25	6e+06	0	0	0	0	0	-6e+06	0	0	0	0	0
26	0	5.5e+05	7.3e+05	0	-5.5e+06	4.2e+06	0	-5.5e+05	-7.3e+05	0	-5.5e+06	4.2e+06
27	0	7.3e+05	1.4e+06	0	-1e+07	5.5e+06	0	-7.3e+05	-1.4e+06	0	-1e+07	5.5e+06
28	0	0	0	6.4e+04	0	0	0	0	0	-6.4e+04	0	0
29	0	-5.5e+06	-1e+07	0	1e+08	-5.5e+07	0	5.5e+06	1e+07	0	5.2e+07	-2.7e+07
30	0	4.2e+06	5.5e+06	0	-5.5e+07	4.2e+07	0	-4.2e+06	-5.5e+06	0	-2.7e+07	2.1e+07
31	-6e+06	0	0	0	0	0	6e+06	0	0	0	0	0
32	0	-5.5e+05	-7.3e+05	0	5.5e+06	-4.2e+06	0	8e+06	-3.6e+06	0	5.5e+06	-4.2e+06
33	0	-7.3e+05	-1.4e+06	0	1e+07	-5.5e+06	0	-3.6e+06	3.9e+06	0	1e+07	-5.5e+06
34	0	0	0	-6.4e+04	0	0	0	0	0	6.4e+04	0	0
35	0	-5.5e+06	-1e+07	0	5.2e+07	-2.7e+07	0	5.5e+06	1e+07	0	1e+08	-5.5e+07
36	0	4.2e+06	5.5e+06	0	-2.7e+07	2.1e+07	0	-4.2e+06	-5.5e+06	0	-5.5e+07	4.2e+07

Tabla 26: Matriz de rigidez del elemento 5 en coordenadas globales

## 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	39	40	41	42
25	4.7e+06	0	2.5e+06	1.2e+07	0	-2.4e+07	-4.7e+06	0	-2.5e+06	1.2e+07	0	-2.4e+07
26	0	9e+06	0	0	0	0	0	-9e+06	0	0	0	0
27	2.5e+06	0	1.9e+06	9.4e+06	0	-1.2e+07	-2.5e+06	0	-1.9e+06	9.4e+06	0	-1.2e+07
28	1.2e+07	0	9.4e+06	6.2e+07	0	-8.2e+07	-1.2e+07	0	-9.4e+06	3.1e+07	0	-4.1e+07
29	0	0	0	0	9.6e+04	0	0	0	0	0	-9.6e+04	0
30	-2.4e+07	0	-1.2e+07	-8.2e+07	0	1.6e+08	2.4e+07	0	1.2e+07	-4.1e+07	0	7.8e+07
37	-4.7e+06	0	-2.5e+06	-1.2e+07	0	2.4e+07	8.4e+06	0	-3.9e+06	-1.2e+07	0	2.4e+07
38	0	-9e+06	0	0	0	0	0	9e+06	0	0	0	0
39	-2.5e+06	0	-1.9e+06	-9.4e+06	0	1.2e+07	-3.9e+06	0	1.3e+07	-9.4e+06	0	1.2e+07
40	1.2e+07	0	9.4e+06	3.1e+07	0	-4.1e+07	-1.2e+07	0	-9.4e+06	6.2e+07	0	-8.2e+07
41	0	0	0	0	-9.6e+04	0	0	0	0	0	9.6e+04	0
42	-2.4e+07	0	-1.2e+07	-4.1e+07	0	7.8e+07	2.4e+07	0	1.2e+07	-8.2e+07	0	1.6e+08

**Tabla 27:** Matriz de rigidez del elemento 6 en coordenadas globales

## 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	45	46	47	48
31	4.7e+06	0	2.5e+06	1.2e+07	0	-2.4e+07	-4.7e+06	0	-2.5e+06	1.2e+07	0	-2.4e+07
32	0	9e+06	0	0	0	0	0	-9e+06	0	0	0	0
33	2.5e+06	0	1.9e+06	9.4e+06	0	-1.2e+07	-2.5e+06	0	-1.9e+06	9.4e+06	0	-1.2e+07
34	1.2e+07	0	9.4e+06	6.2e+07	0	-8.2e+07	-1.2e+07	0	-9.4e+06	3.1e+07	0	-4.1e+07
35	0	0	0	0	9.6e+04	0	0	0	0	0	-9.6e+04	0
36	-2.4e+07	0	-1.2e+07	-8.2e+07	0	1.6e+08	2.4e+07	0	1.2e+07	-4.1e+07	0	7.8e+07
43	-4.7e+06	0	-2.5e+06	-1.2e+07	0	2.4e+07	8.4e+06	0	-3.9e+06	-1.2e+07	0	2.4e+07
44	0	-9e+06	0	0	0	0	0	9e+06	0	0	0	0
45	-2.5e+06	0	-1.9e+06	-9.4e+06	0	1.2e+07	-3.9e+06	0	1.3e+07	-9.4e+06	0	1.2e+07
46	1.2e+07	0	9.4e+06	3.1e+07	0	-4.1e+07	-1.2e+07	0	-9.4e+06	6.2e+07	0	-8.2e+07
47	0	0	0	0	-9.6e+04	0	0	0	0	0	9.6e+04	0
48	-2.4e+07	0	-1.2e+07	-4.1e+07	0	7.8e+07	2.4e+07	0	1.2e+07	-8.2e+07	0	1.6e+08

Tabla 28: Matriz de rigidez del elemento 7 en coordenadas globales

## 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	45	46	47	48
37	6e+06	0	0	0	0	0	-6e+06	0	0	0	0	0
38	0	5.5e+05	7.3e+05	0	-5.5e+06	4.2e+06	0	-5.5e+05	-7.3e+05	0	-5.5e+06	4.2e+06
39	0	7.3e+05	1.4e+06	0	-1e+07	5.5e+06	0	-7.3e+05	-1.4e+06	0	-1e+07	5.5e+06
40	0	0	0	6.4e+04	0	0	0	0	0	-6.4e+04	0	0
41	0	-5.5e+06	-1e+07	0	1e+08	-5.5e+07	0	5.5e+06	1e+07	0	5.2e+07	-2.7e+07
42	0	4.2e+06	5.5e+06	0	-5.5e+07	4.2e+07	0	-4.2e+06	-5.5e+06	0	-2.7e+07	2.1e+07
43	-6e+06	0	0	0	0	0	6e+06	0	0	0	0	0
44	0	-5.5e+05	-7.3e+05	0	5.5e+06	-4.2e+06	0	8e+06	-3.6e+06	0	5.5e+06	-4.2e+06
45	0	-7.3e+05	-1.4e+06	0	1e+07	-5.5e+06	0	-3.6e+06	3.9e+06	0	1e+07	-5.5e+06
46	0	0	0	-6.4e+04	0	0	0	0	0	6.4e+04	0	0
47	0	-5.5e+06	-1e+07	0	5.2e+07	-2.7e+07	0	5.5e+06	1e+07	0	1e+08	-5.5e+07
48	0	4.2e+06	5.5e+06	0	-2.7e+07	2.1e+07	0	-4.2e+06	-5.5e+06	0	-5.5e+07	4.2e+07

Tabla 29: Matriz de rigidez del elemento 8 en coordenadas globales

## III.9. Matriz de rigidez global

Para obtener la matriz de rigidez global de la estructura realizamos la suma componente a componente.

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

Tabla 30: Matriz de rigidez de la estructura parte 1

	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	0	0	0	0	0	0	0	0	-1.7e+06	-2.2e+06	0	-1.6e+07	1.2e+07	0	0	0
2	0	0	0	0	0	0	0	0	-2.2e+06	-4.2e+06	0	-3.1e+07	1.6e+07	0	0	0
3	0	0	0	0	0	0	0	0	0	0	-1.2e+07	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1.6e+07	3.1e+07	0	1.6e+08	-8.2e+07	0	0	0
5	0	0	0	0	0	0	0	0	-1.2e+07	-1.6e+07	0	-8.2e+07	6.2e+07	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	-1.9e+05	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1.7e+06	-2.2e+06
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2.2e+06	-4.2e+06
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.6e+07	3.1e+07
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1.2e+07	-1.6e+07
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1.2e+07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1.6e+07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	-1.6e+08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	1.2e+08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	1.9e+05	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	1.7e+06	2.2e+06	0	-1.6e+07	1.2e+07	0	0	0	0	0	0	0	0	0
20	0	0	2.2e+06	4.2e+06	0	-3.1e+07	1.6e+07	0	0	0	0	0	0	0	0	0
21	0	0	0	0	1.2e+07	0	0	0	0	0	0	0	0	0	0	0
22	0	0	-1.6e+07	-3.1e+07	0	3.1e+08	-1.6e+08	0	0	0	0	0	0	0	0	0
23	0	0	1.2e+07	1.6e+07	0	-1.6e+08	1.2e+08	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	1.9e+05	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	3.5e+07	-1.1e+07	2.5e+06	2.9e+07	-1.2e+07	-2.4e+07	-6e+06	0
26	0	0	0	0	0	0	0	0	-1.1e+07	2.1e+07	7.3e+05	3.1e+07	-2.2e+07	4.2e+06	0	-5.5e+05
27	0	0	0	0	0	0	0	0	2.5e+06	7.3e+05	1.5e+07	9.4e+06	-1e+07	-6.8e+06	0	-7.3e+05
28	0	0	0	0	0	0	0	0	2.9e+07	3.1e+07	9.4e+06	3.8e+08	-1.6e+08	-8.2e+07	0	0
29	0	0	0	0	0	0	0	0	-1.2e+07	-2.2e+07	-1e+07	-1.6e+08	2.3e+08	-5.5e+07	0	5.5e+06
30	0	0	0	0	0	0	0	0	-2.4e+07	4.2e+06	-6.8e+06	-8.2e+07	-5.5e+07	2e+08	0	-4.2e+06
31	0	0	0	0	0	0	0	0	-6e+06	0	0	0	0	0	3.5e+07	-1.1e+07
32	0	0	0	0	0	0	0	0	0	-5.5e+05	-7.3e+05	0	5.5e+06	-4.2e+06	-1.1e+07	2.9e+07
33	0	0	0	0	0	0	0	0	0	-7.3e+05	-1.4e+06	0	1e+07	-5.5e+06	2.5e+06	-3.6e+06
34	0	0	0	0	0	0	0	0	0	0	0	-6.4e+04	0	0	2.9e+07	3.1e+07
35	0	0	0	0	0	0	0	0	0	-5.5e+06	-1e+07	0	5.2e+07	-2.7e+07	-1.2e+07	-1.1e+07
36	0	0	0	0	0	0	0	0	0	4.2e+06	5.5e+06	0	-2.7e+07	2.1e+07	-2.4e+07	-4.2e+06
37	-1.2e+07	0	0	0	0	0	0	0	-4.7e+06	0	-2.5e+06	-1.2e+07	0	2.4e+07	0	0
38	-1.6e+07	0	0	0	0	0	0	0	0	-9e+06	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	-2.5e+06	0	-1.9e+06	-9.4e+06	0	1.2e+07	0	0
40	-8.2e+07	0	0	0	0	0	0	0	1.2e+07	0	9.4e+06	3.1e+07	0	-4.1e+07	0	0
41	6.2e+07	0	0	0	0	0	0	0	0	0	0	0	-9.6e+04	0	0	0
42	0	-1.9e+05	0	0	0	0	0	0	-2.4e+07	0	-1.2e+07	-4.1e+07	0	7.8e+07	0	0
43	0	0	-1.7e+06	-2.2e+06	0	1.6e+07	-1.2e+07	0	0	0	0	0	0	0	-4.7e+06	0
44	0	0	-2.2e+06	-4.2e+06	0	3.1e+07	-1.6e+07	0	0	0	0	0	0	0	0	-9e+06
45	0	0	0	0	-1.2e+07	0	0	0	0	0	0	0	0	0	-2.5e+06	0
46	0	0	-1.6e+07	-3.1e+07	0	1.6e+08	-8.2e+07	0	0	0	0	0	0	0	1.2e+07	0
47	0	0	1.2e+07	1.6e+07	0	-8.2e+07	6.2e+07	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	-1.9e+05	0	0	0	0	0	0	-2.4e+07	0

Tabla 31: Matriz de rigidez de la estructura parte 2

	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	-1.6e+07	1.2e+07	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	-3.1e+07	1.6e+07	0	0	0	0	0	0	0	0	0	0	0	0	0
9	-1.2e+07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1.6e+08	-8.2e+07	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	-8.2e+07	6.2e+07	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	-1.9e+05	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	-1.7e+06	-2.2e+06	0	-1.6e+07	1.2e+07	0	0	0	0	0	0	0
14	0	0	0	0	-2.2e+06	-4.2e+06	0	-3.1e+07	1.6e+07	0	0	0	0	0	0	0
15	0	0	0	0	0	0	-1.2e+07	0	0	0	0	0	0	0	0	0
16	0	0	0	0	1.6e+07	3.1e+07	0	1.6e+08	-8.2e+07	0	0	0	0	0	0	0
17	0	0	0	0	-1.2e+07	-1.6e+07	0	-8.2e+07	6.2e+07	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	-1.9e+05	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	-1.7e+06	-2.2e+06	0	-1.6e+07	1.2e+07	0
20	0	0	0	0	0	0	0	0	0	0	-2.2e+06	-4.2e+06	0	-3.1e+07	1.6e+07	0
21	0	0	0	0	0	0	0	0	0	0	0	0	-1.2e+07	0	0	0
22	0	0	0	0	0	0	0	0	0	0	1.6e+07	3.1e+07	0	1.6e+08	-8.2e+07	0
23	0	0	0	0	0	0	0	0	0	0	-1.2e+07	-1.6e+07	0	-8.2e+07	6.2e+07	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1.9e+05
25	0	0	0	0	-4.7e+06	0	-2.5e+06	1.2e+07	0	-2.4e+07	0	0	0	0	0	0
26	-7.3e+05	0	-5.5e+06	4.2e+06	0	-9e+06	0	0	0	0	0	0	0	0	0	0
27	-1.4e+06	0	-1e+07	5.5e+06	-2.5e+06	0	-1.9e+06	9.4e+06	0	-1.2e+07	0	0	0	0	0	0
28	0	-6.4e+04	0	0	-1.2e+07	0	-9.4e+06	3.1e+07	0	-4.1e+07	0	0	0	0	0	0
29	1e+07	0	5.2e+07	-2.7e+07	0	0	0	0	-9.6e+04	0	0	0	0	0	0	0
30	-5.5e+06	0	-2.7e+07	2.1e+07	2.4e+07	0	1.2e+07	-4.1e+07	0	7.8e+07	0	0	0	0	0	0
31	2.5e+06	2.9e+07	-1.2e+07	-2.4e+07	0	0	0	0	0	0	-4.7e+06	0	-2.5e+06	1.2e+07	0	-2.4e+07
32	-3.6e+06	3.1e+07	-1.1e+07	-4.2e+06	0	0	0	0	0	0	0	-9e+06	0	0	0	0
33	1.8e+07	9.4e+06	1e+07	-1.8e+07	0	0	0	0	0	-2.5e+06	0	-1.9e+06	9.4e+06	0	-1.2e+07	0
34	9.4e+06	3.8e+08	-1.6e+08	-8.2e+07	0	0	0	0	0	-1.2e+07	0	-9.4e+06	3.1e+07	0	-4.1e+07	0
35	1e+07	-1.6e+08	2.3e+08	-5.5e+07	0	0	0	0	0	0	0	0	0	0	-9.6e+04	0
36	-1.8e+07	-8.2e+07	-5.5e+07	2e+08	0	0	0	0	0	2.4e+07	0	1.2e+07	-4.1e+07	0	7.8e+07	0
37	0	0	0	0	3.8e+07	-1.1e+07	-3.9e+06	4.1e+06	-1.2e+07	2.4e+07	-6e+06	0	0	0	0	0
38	0	0	0	0	-1.1e+07	2.1e+07	7.3e+05	3.1e+07	-2.2e+07	4.2e+06	0	-5.5e+05	-7.3e+05	0	-5.5e+06	4.2e+06
39	0	0	0	0	-3.9e+06	7.3e+05	2.6e+07	-9.4e+06	-1e+07	1.8e+07	0	-7.3e+05	-1.4e+06	0	-1e+07	5.5e+06
40	0	0	0	0	4.1e+06	3.1e+07	-9.4e+06	3.8e+08	-1.6e+08	-8.2e+07	0	0	0	-6.4e+04	0	0
41	0	0	0	0	-1.2e+07	-2.2e+07	-1e+07	-1.6e+08	2.3e+08	-5.5e+07	0	5.5e+06	1e+07	0	5.2e+07	-2.7e+07
42	0	0	0	0	2.4e+07	4.2e+06	1.8e+07	-8.2e+07	-5.5e+07	2e+08	0	-4.2e+06	-5.5e+06	0	-2.7e+07	2.1e+07
43	-2.5e+06	-1.2e+07	0	2.4e+07	-6e+06	0	0	0	0	0	3.8e+07	-1.1e+07	-3.9e+06	4.1e+06	-1.2e+07	2.4e+07
44	0	0	0	0	0	-5.5e+05	-7.3e+05	0	5.5e+06	-4.2e+06	-1.1e+07	2.9e+07	-3.6e+06	3.1e+07	-1.1e+07	-4.2e+06
45	-1.9e+06	-9.4e+06	0	1.2e+07	0	-7.3e+05	-1.4e+06	0	1e+07	-5.5e+06	-3.9e+06	-3.6e+06	2.9e+07	-9.4e+06	1e+07	6.8e+06
46	9.4e+06	3.1e+07	0	-4.1e+07	0	0	0	-6.4e+04	0	0	4.1e+06	3.1e+07	-9.4e+06	3.8e+08	-1.6e+08	-8.2e+07
47	0	0	-9.6e+04	0	0	-5.5e+06	-1e+07	0	5.2e+07	-2.7e+07	-1.2e+07	-1.1e+07	1e+07	-1.6e+08	2.3e+08	-5.5e+07
48	-1.2e+07	-4.1e+07	0	7.8e+07	0	4.2e+06	5.5e+06	0	-2.7e+07	2.1e+07	2.4e+07	-4.2e+06	6.8e+06	-8.2e+07	-5.5e+07	2e+08

Tabla 32: Matriz de rigidez de la estructura parte 3

### III.10. Restricciones

Los nodos 1, 2, 3 y 4 están fijos por lo tanto podemos eliminar sus grados de libertad correspondientes 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	33	34	35	36
25	34563400	-10634792	2453623.2	28635996	-12450000	-23517000	-6000000	0	0	0	0	0
26	-10634792	21134133	726999.46	31350000	-21820376	4156000	0	-554133.33	-726999.46	0	-5452495.9	4156000
27	2453623.2	726999.46	15263800	9351000	-10452000	-6815619.9	0	-726999.46	-1393600	0	-10452000	5452495.9
28	28635996	31350000	9351000	3.75904e+08	-1.636788e+08	-81787439	0	0	0	-64000	0	0
29	-12450000	-21820376	-10452000	-1.636788e+08	2.29116e+08	-54524959	0	5452495.9	10452000	0	52260000	-27262480
30	-23517000	4156000	-6815619.9	-81787439	-54524959	1.98532e+08	0	-4156000	-5452495.9	0	-27262480	20780000
31	-6000000	0	0	0	0	0	34563400	-10634792	2453623.2	28635996	-12450000	-23517000
32	0	-554133.33	-726999.46	0	5452495.9	-4156000	-10634792	28593333	-3579571.7	31350000	-10915384	-4156000
33	0	-726999.46	-1393600	0	10452000	-5452495.9	2453623.2	-3579571.7	17750200	9351000	10452000	-17720612
34	0	0	0	-64000	0	0	28635996	31350000	9351000	3.75904e+08	-1.636788e+08	-81787439
35	0	-5452495.9	-10452000	0	52260000	-27262480	-12450000	-10915384	10452000	-1.636788e+08	2.29116e+08	-54524959
36	0	4156000	5452495.9	0	-27262480	20780000	-23517000	-4156000	-17720612	-81787439	-54524959	1.98532e+08
37	-4703400	0	-2453623.2	-12268116	0	23517000	0	0	0	0	0	0
38	0	-9000000	0	0	0	0	0	0	0	0	0	0
39	-2453623.2	0	-1870200	-9351000	0	12268116	0	0	0	0	0	0
40	12268116	0	9351000	31170000	0	-40893720	0	0	0	0	0	0
41	0	0	0	0	-96000	0	0	0	0	0	0	0
42	-23517000	0	-12268116	-40893720	0	78390000	0	0	0	0	0	0
43	0	0	0	0	0	0	-4703400	0	-2453623.2	-12268116	0	23517000
44	0	0	0	0	0	0	0	-9000000	0	0	0	0
45	0	0	0	0	0	0	-2453623.2	0	-1870200	-9351000	0	12268116
46	0	0	0	0	0	0	12268116	0	9351000	31170000	0	-40893720
47	0	0	0	0	0	0	0	0	0	0	-96000	0
48	0	0	0	0	0	0	-23517000	0	-12268116	-40893720	0	78390000

Tabla 33: Matriz de rigidez de la estructura sin las restricciones parte 1

	37	38	39	40	41	42	43	44	45	46	47	48
25	-4703400	0	-2453623.2	12268116	0	-23517000	0	0	0	0	0	0
26	0	-9000000	0	0	0	0	0	0	0	0	0	0
27	-2453623.2	0	-1870200	9351000	0	-12268116	0	0	0	0	0	0
28	-12268116	0	-9351000	31170000	0	-40893720	0	0	0	0	0	0
29	0	0	0	0	-96000	0	0	0	0	0	0	0
30	23517000	0	12268116	-40893720	0	78390000	0	0	0	0	0	0
31	0	0	0	0	0	0	-4703400	0	-2453623.2	12268116	0	-23517000
32	0	0	0	0	0	0	0	-9000000	0	0	0	0
33	0	0	0	0	0	0	-2453623.2	0	-1870200	9351000	0	-12268116
34	0	0	0	0	0	0	-12268116	0	-9351000	31170000	0	-40893720
35	0	0	0	0	0	0	0	0	0	0	-96000	0
36	0	0	0	0	0	0	0	0	0	-40893720	0	78390000
37	38230000	-10634792	-3897114.3	4099764.3	-12450000	23517000	-6000000	0	0	0	0	0
38	-10634792	21134133	726999.46	31350000	-21820376	4156000	0	-554133.33	-726999.46	0	-5452495.9	4156000
39	-3897114.3	726999.46	26263600	-9351000	-10452000	17720612	0	-726999.46	-1393600	0	-10452000	5452495.9
40	4099764.3	31350000	-9351000	3.75904e+08	-1.636788e+08	-81787439	0	0	0	-64000	0	0
41	-12450000	-21820376	-10452000	-1.636788e+08	2.29116e+08	-54524959	0	5452495.9	10452000	0	52260000	-27262480
42	23517000	4156000	17720612	-81787439	-54524959	1.98532e+08	0	-4156000	-5452495.9	0	-27262480	20780000
43	-6000000	0	0	0	0	0	38230000	-10634792	-3897114.3	4099764.3	-12450000	23517000
44	0	-554133.33	-726999.46	0	5452495.9	-4156000	-10634792	28593333	-3579571.7	31350000	-10915384	-4156000
45	0	-726999.46	-1393600	0	10452000	-5452495.9	-3897114.3	-3579571.7	28750000	-9351000	10452000	6815619.9
46	0	0	0	-64000	0	0	4099764.3	31350000	-9351000	3.75904e+08	-1.636788e+08	-81787439
47	0	-5452495.9	-10452000	0	52260000	-27262480	-12450000	-10915384	10452000	-1.636788e+08	2.29116e+08	-54524959
48	0	4156000	5452495.9	0	-27262480	20780000	23517000	-4156000	6815619.9	-81787439	-54524959	1.98532e+08

Tabla 34: Matriz de rigidez de la estructura sin las restricciones parte 2

### III.11. Cargas

En el caso de las cargas poseemos dos fuerzas aplicadas en los nodos 5 y 6 en dirección al grado de libertad 26 y 32 respectivamente. También se tienen una serie de cargas distribuidas sobre los elementos 5, 6, 7 y 8 en dirección  $-z$ . Para tratar esta fuerzas se hallan las cargas nodales equivalentes suponiendo una viga biempotrada y colocando las reacciones como cargas en los grados de libertad correspondientes.

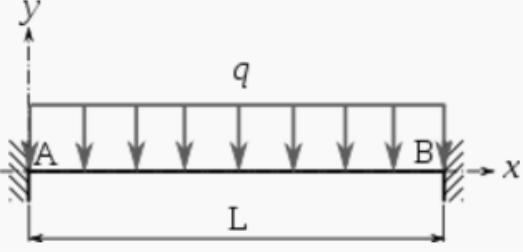
Tipo de carga	Reacciones
<p>Biempotrada con carga uniforme en una porción simétricamente distribuida</p> 	$R_A = +\frac{qL}{2}, R_B = +\frac{qL}{2}$ $M_A = +\frac{qL^2}{12}, M_B = -\frac{qL^2}{12}$ $M_f(x) = -\frac{qL^2}{12} + \frac{qx(L-x)}{2}$

Figura 4: Reacciones en una viga empotrada, extraído de:  
[https://es.wikipedia.org/wiki/Anexo:Pendientes\\_y\\_deformaciones\\_en\\_vigasVigas\\_biempotradas](https://es.wikipedia.org/wiki/Anexo:Pendientes_y_deformaciones_en_vigasVigas_biempotradas)

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	3000	-26.04165	17.3611	39.062475	0
6	0	3000	-26.04165	17.3611	39.062475	0
7	0	0	-26.04165	17.3611	39.062475	0
8	0	0	-26.04165	17.3611	39.062475	0

Tabla 35: Vector de cargas

### III.12. Obtención de los desplazamientos

Para obtener los desplazamientos se resuelve la ecuación

$$F = kU$$

Donde  $F$  es el vector de las cargas nodales,  $k$  la matriz de rigidez de la estructura en coordenadas globales y  $U$  es la matriz que contiene los desplazamientos de cada nodo en cada uno de sus grados de libertad.

```

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.linalg
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	Rotación Z
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0.00027933947	0.00057306256	3.8596174e-05	-5.8338549e-05	1.4533345e-05	-1.5762671e-05
6	0.00016980816	0.0002650649	7.4006314e-05	-2.9486177e-05	6.8046076e-06	-4.165677e-06
7	0.00017558027	0.0004113198	3.6711495e-05	-3.8301462e-05	1.3881656e-05	-9.790664e-06
8	9.5238567e-05	0.00014734332	3.9715686e-05	-1.4461576e-05	7.3759234e-06	7.8937998e-08

Tabla 36: Desplazamientos de los nodos

### III.13. Obtención de los momentos flectores

Se utiliza la ecuación

$$M = \frac{EI}{l_e^2} [6\zeta q_1 + (3\zeta - 1)l_e q_2 - 6\zeta q_3 + (3\zeta + 1)l_e q_4]$$

```

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])

```

In [41]: # Realizamos los gráficos

```

ipplot([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'))]
```

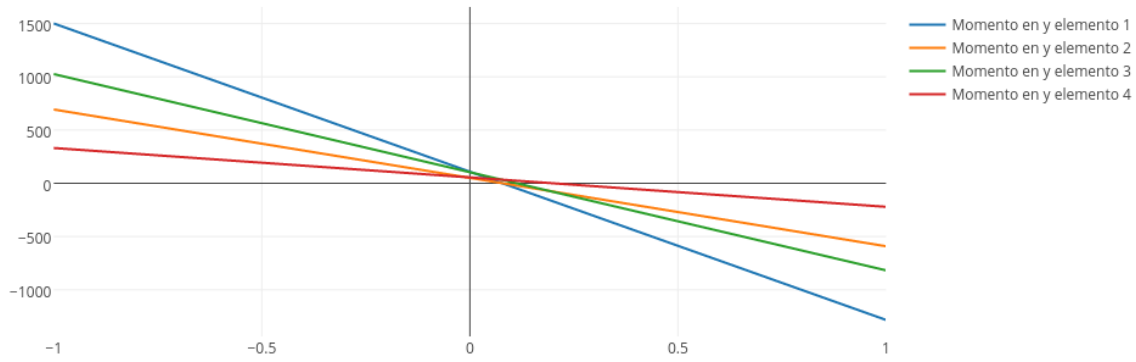


Figura 5: Momento en Y en los elementos 1, 2, 3 y 4 (Máximo: 1501,188 para elemento 1)

```
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'))])
```

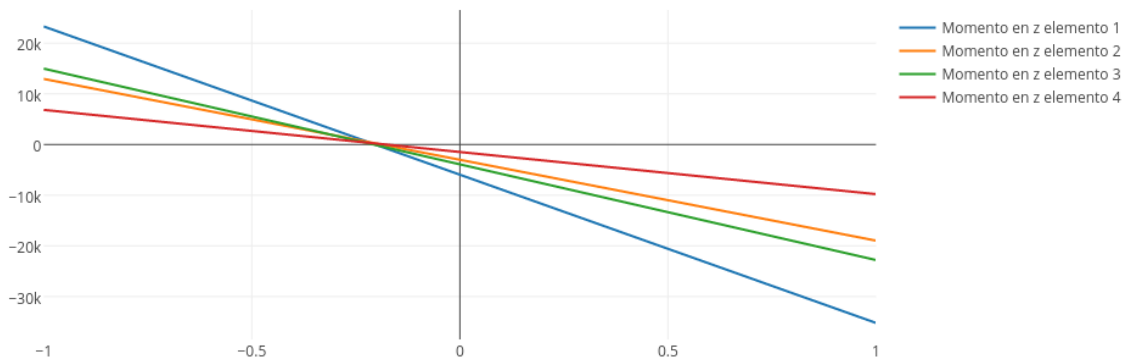


Figura 6: Momento en Z en los elementos 1, 2, 3 y 4 (Máximo: -35,199e3 para elemento 1)

```
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

```

ipplot([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')])

```

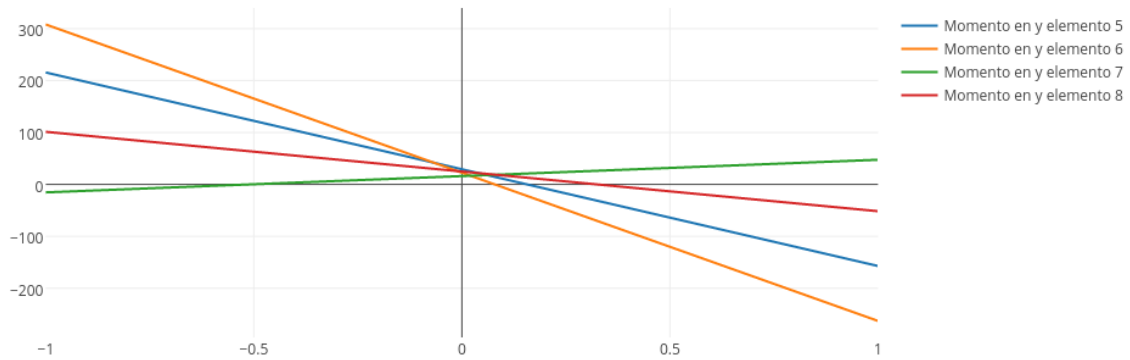


Figura 7: Momento en Y en los elementos 5, 6, 7 y 8 (Máximo: 308,0745 para elemento 6)

```
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'))])
```

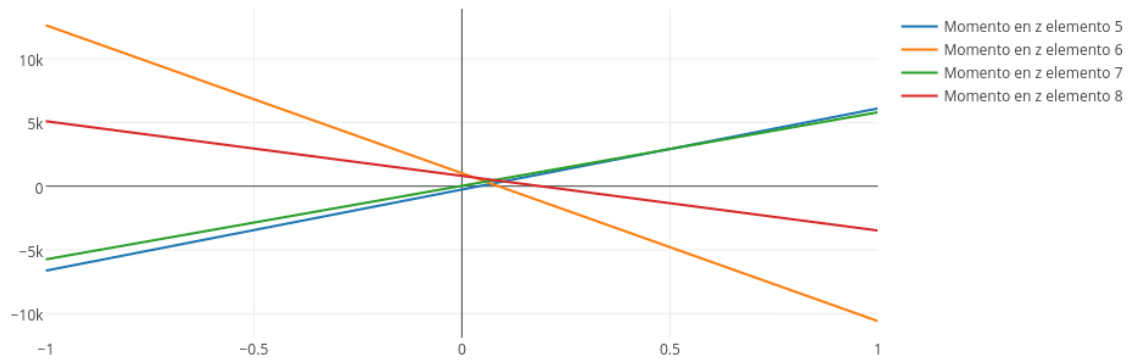


Figura 8: Momento en Z en los elementos 5, 6, 7 y 8 (Máximo: 12,6327e3 para elemento 6)