In [1]:
```
# Se importan las librerías a utilizar
import pandas as pd
import numpy as np
from __future__ import division, print_function

# Datos
E = 200e9              # Módulo elśtico del material [Pa]
A = 250 / (1000 ** 2)  # Área transversal en [m2]
F = 18e3               # Fuerza aplicda en 1 [N]

# Coordenadas respecto a 1 [m]
# 1
x_1 = 0
y_1 = 0

# 2
x_2 = -.45
y_2 = .6

# 3
x_3 = .45 + .35
y_3 = .6

# 4
x_4 = .45
y_4 = .6
```

In [2]:
```
# Solución

# Coordenadas nodales
coordenadas_nodales = pd.DataFrame({'X': [x_1, x_2, x_3, x_4],
                                    'Y': [y_1, y_2, y_3, y_4]},
                                    index = [1,2,3,4])
print('La tabla de coordenadas nodales es la siguiente: ')
coordenadas_nodales
```

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

Out[2]:

	X	Y
1	0.00	0.0
2	-0.45	0.6
3	0.80	0.6
4	0.45	0.6

In [3]:
```
# Conectividad

conectividad = pd.DataFrame({1: [1, 1, 1],
                             2: [2, 3, 4]},
```

```
                                index = [1, 2, 3])
print('Tabla de conectividad')
conectividad
```

```
Tabla de conectividad
```

Out[3]:

	1	2
1	1	2
2	1	3
3	1	4

In [4]:

```
# Cosenos directores

# 1
le_1 = np.sqrt((x_2 - x_1) ** 2 + (y_2 - y_1) ** 2)
l_1 = (x_2 - x_1) / le_1
m_1 = (y_2 - y_1) / le_1

# 2
le_2 = np.sqrt((x_3 - x_1) ** 2 + (y_3 - y_1) ** 2)
l_2 = (x_3 - x_1) / le_2
m_2 = (y_3 - y_1) / le_2

# 3
le_3 = np.sqrt((x_4 - x_1) ** 2 + (y_4 - y_1) ** 2)
l_3 = (x_4 - x_1) / le_3
m_3 = (y_4 - y_1) / le_3

# Tabla de cosenos directores
cosenos_dir = pd.DataFrame({'le': [le_1, le_2, le_3],
                            'l' : [l_1, l_2, l_3],
                            'm' : [m_1, m_2, m_3]},
                           index = [1, 2, 3])
print('Tabla de cosenos directores')
cosenos_dir
```

```
Tabla de cosenos directores
```

Out[4]:

	l	le	m
1	-0.6	0.75	0.8
2	0.8	1.00	0.6
3	0.6	0.75	0.8

In [5]:

```
# Matriz de rigidez

# Se define una función que nos calcula la matriz de rigidez
def matrizDeRigidez(E, A, le, l, m, i, j):
    k = (E * A) / le
    matriz = k * np.array([[l**2, l*m, -l**2, -l*m],
```

```
                                [l*m, m**2, -l*m, -m**2],
                                [-l**2, -l*m, l**2, l*m],
                                [-l*m, -m**2, l*m, m**2]])
    tabla = pd.DataFrame(matriz, columns = [2*i-1, 2*i, 2*j-1, 2*j],index
    return tabla

# 1
matriz_de_rig_1 = matrizDeRigidez(E, A, le_1, l_1, m_1, 1, 2)
print('La matriz de rigidez 1 es: ')
matriz_de_rig_1
```

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

Out[5]:

	1	2	3	4
1	24000000	-32000000.000000	-24000000	32000000.000000
2	-32000000	42666666.666667	32000000	-42666666.666667
3	-24000000	32000000.000000	24000000	-32000000.000000
4	32000000	-42666666.666667	-32000000	42666666.666667

In [6]:
```
# 2
matriz_de_rig_2 = matrizDeRigidez(E, A, le_2, l_2, m_2, 1, 3)
print('La matriz de rigidez 2 es: ')
matriz_de_rig_2
```

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

Out[6]:

	1	2	5	6
1	32000000	24000000	-32000000	-24000000
2	24000000	18000000	-24000000	-18000000
5	-32000000	-24000000	32000000	24000000
6	-24000000	-18000000	24000000	18000000

In [7]:
```
# 3
matriz_de_rig_3 = matrizDeRigidez(E, A, le_3, l_3, m_3, 1, 4)
print('La matriz de rigidez 3 es: ')
matriz_de_rig_3
```

```
La matriz de rigidez 3 es:
```

Out[7]:

	1	2	7	8
1	24000000	32000000.000000	-24000000	-32000000.000000
2	32000000	42666666.666667	-32000000	-42666666.666667
7	-24000000	-32000000.000000	24000000	32000000.000000
8	-32000000	-42666666.666667	32000000	42666666.666667

In [8]:
```
# Se ubican las matrices de rigidez de los elementos en la matriz de rigid

# Creamos una matriz 8x8 llena de zeros
matriz_de_rigidez_estr = pd.DataFrame(np.zeros((8,8)), index = [1,2,3,4,5,

# Ubicamos la matriz del elemento 1
for i in matriz_de_rig_1.index:
    for j in matriz_de_rig_1.columns:
        matriz_de_rigidez_estr.loc[i,j] = matriz_de_rigidez_estr.loc[i,j]

# 2
for i in matriz_de_rig_2.index:
    for j in matriz_de_rig_2.columns:
        matriz_de_rigidez_estr.loc[i,j] = matriz_de_rigidez_estr.loc[i,j]

# 3
for i in matriz_de_rig_3.index:
    for j in matriz_de_rig_3.columns:
        matriz_de_rigidez_estr.loc[i,j] = matriz_de_rigidez_estr.loc[i,j]
```

In [9]:
```
# Encendemos la notación científica con 2 decimales
pd.options.display.float_format = '{:,.2e}'.format

# Mostramos la matriz
matriz_de_rigidez_estr
```

Out[9]:

	1	2	3	4	5	6	7	8
1	8.00e+07	2.40e+07	-2.40e+07	3.20e+07	-3.20e+07	-2.40e+07	-2.40e+07	-3.20e+
2	2.40e+07	1.03e+08	3.20e+07	-4.27e+07	-2.40e+07	-1.80e+07	-3.20e+07	-4.27e+
3	-2.40e+07	3.20e+07	2.40e+07	-3.20e+07	0.00e+00	0.00e+00	0.00e+00	0.00e+0
4	3.20e+07	-4.27e+07	-3.20e+07	4.27e+07	0.00e+00	0.00e+00	0.00e+00	0.00e+0
5	-3.20e+07	-2.40e+07	0.00e+00	0.00e+00	3.20e+07	2.40e+07	0.00e+00	0.00e+0
6	-2.40e+07	-1.80e+07	0.00e+00	0.00e+00	2.40e+07	1.80e+07	0.00e+00	0.00e+0
7	-2.40e+07	-3.20e+07	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.40e+07	3.20e+0
8	-3.20e+07	-4.27e+07	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.20e+07	4.27e+0

In [10]:
```
# los grados de libertad corresponden a 1 y 2asi que eliminamos los demás
matriz_k_tabla = matriz_de_rigidez_estr.loc[0:2,[1,2]]
matriz_k_tabla
```

Out[10]:

	1	2
1	8.00e+07	2.40e+07
2	2.40e+07	1.03e+08

In [11]:
```
# se ha trabajo con tablas, ahora se lleva realmente a un array o matriz

# creamos una matriz 2x2 de ceros
matriz_k = np.zeros([2,2])

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

# mostramos la matriz
print(matriz_k)
```

```
[[  8.00000000e+07   2.40000000e+07]
 [  2.40000000e+07   1.03333333e+08]]
```

In [12]:
```
# para resolver importamos el módulo solve de la librería scipy.linag
# estos módulos son programados a bajo nivel con lenguajes como C o Fortra
from scipy.linalg import solve

# creamos la matriz de fuerza
f = np.array([0, -F])

# resolvemos el problema F = kU obteniendo el desplazamiento
desp = solve(matriz_k,f)
print('El desplazamiento en el punto 1 es [m]:')
print('En el eje x', desp[0])
print('En el eje y', desp[1])
```

```
El desplazamiento en el punto 1 es [m]:
En el eje x 5.61719833564e-05
En el eje y -0.000187239944521
```

In [13]:
```
# esfuerzo en 3
a = np.array([-l_3, -m_3, l_3, m_3])
q = np.array([desp[0], desp[1], 0, 0])
sigma_3 = (E / le_3) * np.dot(a,q)
print('El esfuerzo en 3 es [Pa]:',sigma_3)
```

```
El esfuerzo en 3 es [Pa]: 30957004.1609
```