TP 3 – CORRIGE

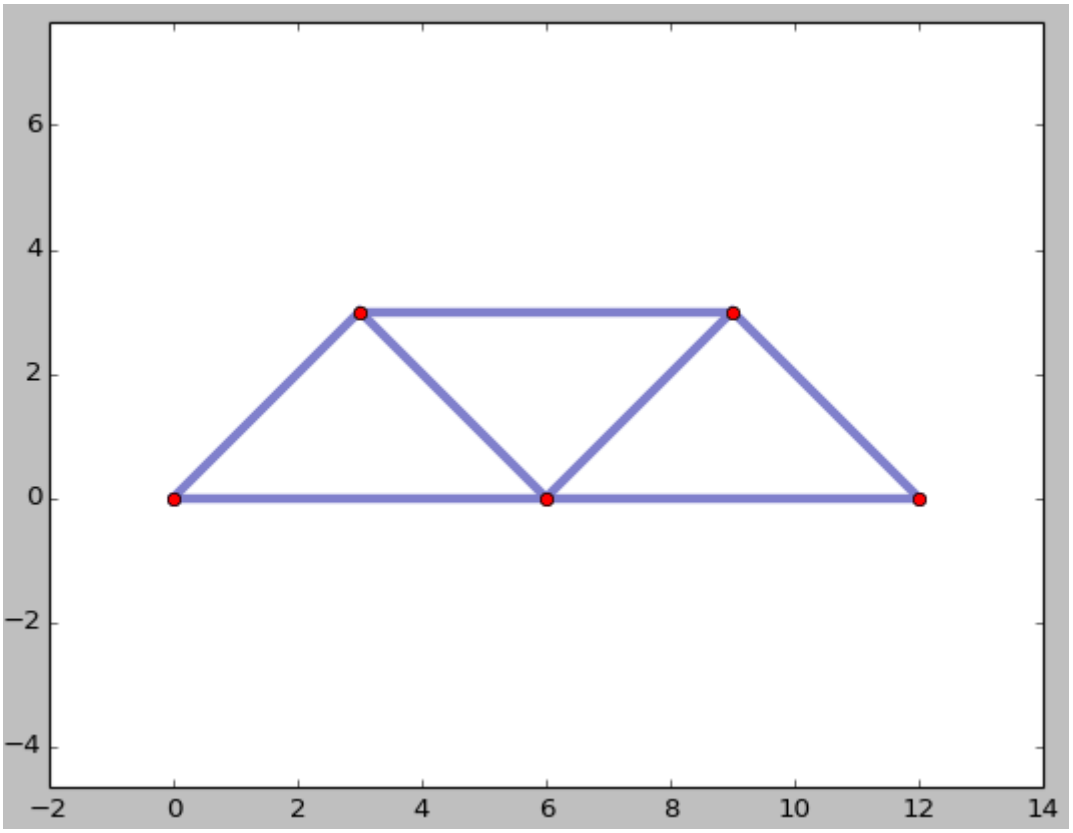
1.1 Dichotomie (voir python)

1.2 Newton(voir python)

1.3 Chimie(voir python)

CONSTRUCTION DE L'ENSEMBLE DES POINTS DANS UN GRAPHIQUE ET DES BARRES

On construit d'abord la liste contenant la position (dans un plan) des pivots. On peut choisir (par exemple) d'utiliser une liste

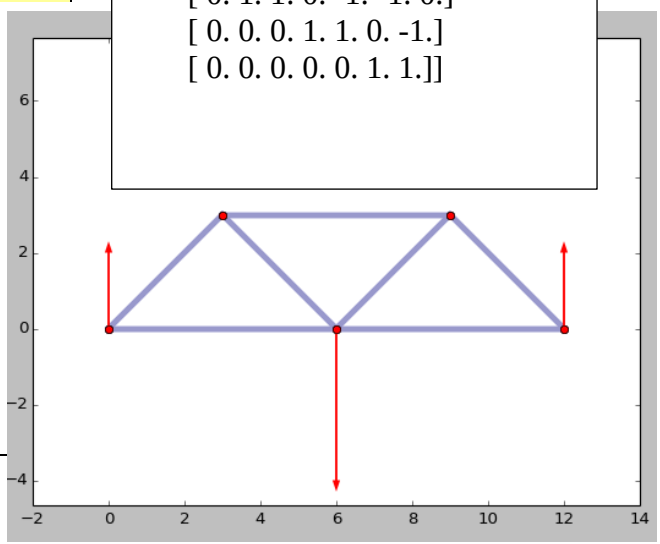


1.4 Construction des matrices A et b

```
mat_inc= np.zeros((N_piv,N_bar))

for j, bar_j in enumerate(barres):
    P1, P2 = bar_j
    # Remarquons la convention de signe:
    i1 = pivots.index(P1)
    mat_inc[i1,j] = -1 # la barre j "quitte" P1
    i2 = pivots.index(P2)
    mat_inc[i2,j] = +1 # la barre j "arrive à" P2
    print(mat_inc)
```

```
M=[[-1. -1. 0. 0. 0. 0. 0.]
   [ 1. 0. -1. -1. 0. 0. 0.]
   [ 0. 1. 1. 0. -1. -1. 0.]
   [ 0. 0. 0. 1. 1. 0. -1.]
   [ 0. 0. 0. 0. 0. 1. 1.]]
```



1.5 Chargement

```
F_ext = np.zeros((2,len(pivots)))
F_ext.T # .T signifie "transposée". (Utilisé ici pour l'esthétique)

#Soit une charge de 1,5 tonne au centre du pont (15KN sur y)
```

```
F_ext[1,2]=-15
#En isolant l'ensemble
F_ext[1,0]=7.5
F_ext[1,4]=7.5
```

1.6 Vecteurs directeurs des barres

```
# On obtient un tableau 3D
Matrice barres_arr:
[[[ 0 0]
[ 3 3]]

[[ 0 0]
[ 6 0]]

[[ 3 3]
[ 6 0]]

[[ 3 3]
[ 9 3]]

[[ 6 0]
[ 9 3]]

[[ 6 0]
[12 0]]

[[ 9 3]
[12 0]]]
```

vecteur P_1P_2

```
barres_OP1 = barres_arr[:,0,:]
barres_OP2 = barres_arr[:,1,:]
barres_P1P2 = barres_OP2 - barres_OP1
print('Matrice barres_P1P2:')
print(barres_P1P2)
```

```
Matrice barres_P1P2:
[[ 3 3]
[ 6 0]
[ 3 -3]
[ 6 0]
[ 3 3]
[ 6 0]
[ 3 -3]]
```

On détermine alors le vecteur normalisé intitulé *barre_dir*

$$\vec{u} = \frac{\overrightarrow{P_1P_2}}{\left\|\overrightarrow{P_1P_2}\right\|}$$

```
# Étape 1 : calcul de la longueur des barres :
barre_l = np. sqrt (( barres_P1P2 **2). sum( axis = 1))
# Étape 2 : Transformation en vecteur colonne :
barre_l = barre_l . reshape ( -1 ,1)
```

Q1.Ecrire alors la matrice barre_dir

```
Matrice barre_dir:
[[ 0.70710678 0.70710678]
[ 1. 0. ]
[ 0.70710678 -0.70710678]
[ 1. 0. ]
[ 0.70710678 0.70710678]
[ 1. 0. ]
[ 0.70710678 -0.70710678]]
```

4.5 Construction de la matrices A et du vecteur b

La matrice A est fabriquée par superposition de 2 blocs :

- A_x *contient les équations d'équilibre projetées sur l'axe horizontal*
- A_y *contient les équations d'équilibre projetées sur l'axe vertical*

Nous associons ensuite les deux matrices avec ***A = np. vstack ((Ax , Ay))***

```
Inc_mat = np.zeros((N_piv, len(barres)), dtype=int)

for j, bar_j in enumerate(barres):
P1, P2 = bar_j
# Remarquons la convention de signe:
i1=pivots.index((P1))
Inc_mat[i1,j] = -1 # la barre j "quitte" P1
i2=pivots.index((P2))
Inc_mat[i2,j] = +1 # la barre j "arrive à" P2

print('Matrice d\'incidence:')
print(str(Inc_mat).replace('0','.')) # Méthode d'affichage cosmétique

#Construction de la Matrice A
Ax = Inc_mat*barre_dir[:,0]
Ay = Inc_mat*barre_dir[:,1]

#La superposition (concaténation) des blocs se fait avec np.vstack (il existe np.hstack et np.concatenate)
A = np.vstack((Ax, Ay))
```

```
A= [[-0.71 -1. 0. 0. 0. 0. 0. ]
[ 0.71 0. -0.71 -1. 0. 0. 0. ]
[ 0. 1. 0.71 0. -0.71 -1. 0. ]
[ 0. 0. 0. 1. 0.71 0. -0.71]
[ 0. 0. 0. 0. 0. 1. 0.71]
[-0.71 -0. -0. 0. 0. 0. -0. ]
[ 0.71 0. 0.71 -0. 0. 0. -0. ]
[ 0. 0. -0.71 0. -0.71 -0. -0. ]
[ 0. 0. -0. 0. 0.71 0. 0.71]
[ 0. 0. -0. 0. 0. 0. -0.71]]
```

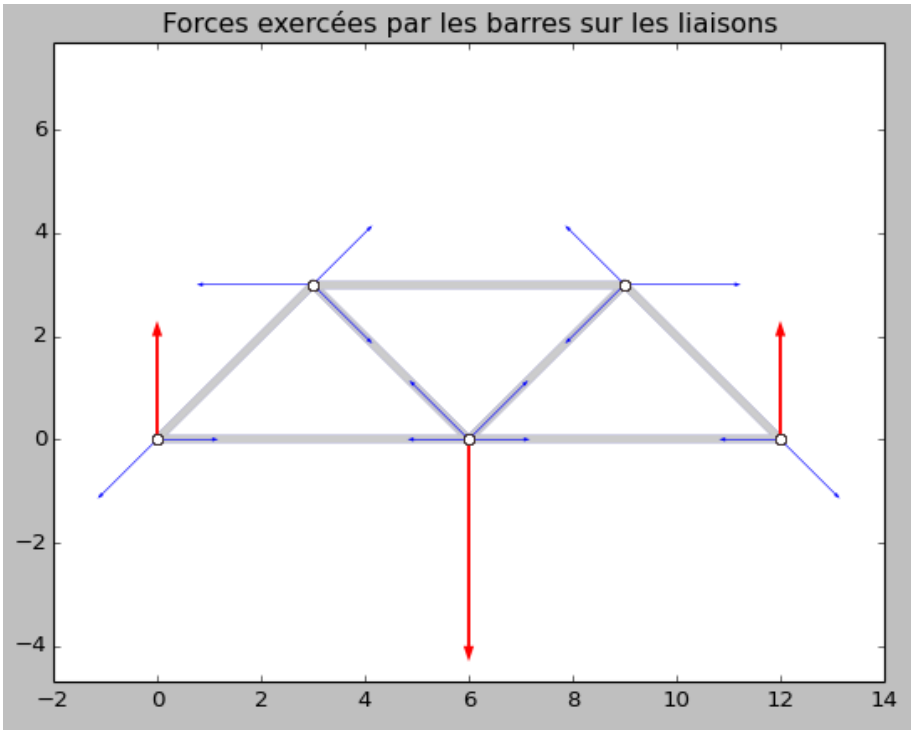
Le vecteur b est construit selon le même principe de concaténation que la matrice d'incidence.

```
b= np.concatenate((F_ext[0,:],  F_ext[1,:]))
```

le second membre b.

[0. 0. 0. 0. 0. 7.5 0. -15. 0. 7.5]

4.6 Résolution



Requêtes :

SELECT x_ini,y_ini,z_ini FROM nœuds

SELECT count (DISTINCT id_noeuds) FROM etat WHERE datem < tmin ( )

pas besoin de distinct si la table est juste !

SELECT id_noeuds , MAX(datem) FROM etat WHERE datem < tmin ( ) GROUP BY id_noeuds

```
SELECT b.x , b.y , b.z , b.Fx , b. Fy , b.Fz, a.x_ini , a.y_ini   FROM date_mesure  AS c
JOIN etat AS b ON c . id_noeuds=b . id_noeuds AND c . date_der=b . datem
JOIN noeuds AS a ON c . id_noeuds=a . id_noeuds
WHERE  SQRT(POWER(b.Fx ,2)+POWER(b.Fy ,2)+POWER(b.Fz , 2 ) ) AS norme > force_min( )
ORDER BY norme ASC
```