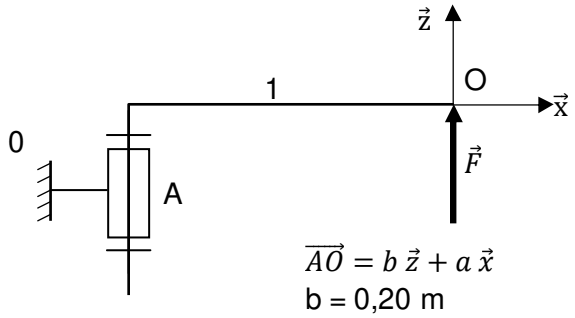
td	td 11 (2)	TSI2
	Systèmes d'équations linéaires	Période 3
		1h

Problématique

Un support 1 est en liaison pivot d'axe(A,$\vec{z}$) avec le bâti et subit une force $\vec{\mathbf{F}}$ en O.

Il a été choisi que la résolution numérique de l'équilibre des mécanismes conduirait à écrire de façon systématique l'équilibre au point O.

Le problème à résoudre est de déterminer les inconnues dans la liaison pivot qui permettent d'équilibrer la force $\vec{F} = F.\vec{z}$ (avec F=2000N).



$\overrightarrow{AO} = b\,\vec{z} + a\,\vec{x}$

b = 0,20 m

a = 1,4 m

Hypothèses :

- le problème admet le plan (O, $\vec{x}$, $\vec{z}$) comme plan de symétrie de la géométrie et des actions mécaniques extérieures. Le torseur des actions mécaniques exercé par 0 sur 1 s'écrit ainsi

$$\{T_{0\to1}\} = \begin{Bmatrix} X & 0 \\ 0 & M \\ Z & 0 \end{Bmatrix}_{A,R}$$

- le poids de 1 est négligé par rapport aux autres actions mécaniques.

Equilibre de 1 en 0 :

On montre que l'équilibre de 1 en O, conduit au système d'équations suivant (les longueurs sont par défaut en mm):

$$S : \begin{cases} X = 0 \\ Z = 2000 \\ N + 1.4 * Z = 0 \end{cases}$$

Les fonctions utiles à la résolution des systèmes linéaires sont les suivantes :

Fonction	Entrées	Sortie	Utilité
mult(L, a)	L: liste de nombres a : nombre	L	L ← a*L
echanger(M, p, q)	M : tableau à n lignes et n+1 colonnes p : entier tel que 0 ≤ p ≤ n q : entier tel que 0 ≤ q ≤ n	M : tableau à n lignes et n+1 colonnes	échanger les lignes p et q du tableau M
combiner(M, a, p, q)	M : tableau à n lignes et n+1 colonnes a : nombre p : entier tel que 0 ≤ p ≤ n q : entier tel que 0 ≤ q ≤ n	M : tableau à n lignes et n+1 colonnes	changer la ligne q : Lq ←Lq + a*Lp

Matrice augmentée

1) Ecrire la matrice augmentée Ma du système d'équations (on placera les inconnues dans l'ordre suivant : X,Z,N).

2) Ecrire l'instruction qui permet d'affecter à la variable Ma1 la matrice augmentée du système d'équations (on utilisera la bibliothèque numpy importée sous le nom np).

Etape 1 : Echelonnage (méthode du pivot partiel)

3) Ecrire une fonction maximum qui admet en entrées une matrice M à n lignes et un numéro de colonne j (0≤j<n) et qui renvoie en sortie la position de la ligne k où situe la maximum (en valeur absolue) des éléments de la colonne j pour les lignes situées en dessous de la ligne j ($j \leq k < n$).

```
def zeros_bas(M,i):
    n = len(M)
    if M[i,i]==0:
        for k in range(i+1,n):
            if M[k,i]!=0:
                echange(M,k,i)
                break
    for k in range (i+1,n):
        combiner(M,-M[k,i]/M[i,i],i,k)
    return M
```

4) Compléter les tableaux suivants suite aux instructions qui les précède.

```
def zeros_bas_partiel(M,i):
    n = len(M)
    maxi = maximum(M,i)
    echange(M, i, maxi)
    for k in range (i+1,n):
        combiner(M,-M[k,i]/M[i,i], i ,k)
    return M
```

zeros_bas(Ma1, 1)

iteration	init (à l'entrée du 2e for)	1
k		2
M	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2000 \\ 0 & 1.4 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix}$

zeros_bas_partiel(Ma1, 1)

iteration	init (à l'entrée du for)	1
k		2
M	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1.4 & 1 & 0 \\ 0 & 1 & 0 & 2000 \end{bmatrix}$	$\begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix}$

Etape 2 : Matrice réduite

5) Ecrire la fonction réduire(M) qui permet d'obtenir la matrice réduite du système :

```
[[ 1.00000000e+00,  0.00000000e+00,  0.00000000e+00, 0.00000000e+00],
 [ 0.00000000e+00,  1.00000000e+00,  7.14285714e-01, 0.00000000e+00],
 [-0.00000000e+00, -0.00000000e+00,  1.00000000e+00, -2.80000000e+03]]
```

Etape 3 : Remontée

6) En vous inspirant de la fonction zeros_bas, créer une fonction zeros_haut (tous les pivots étant unitaire à ce stade, il n'est pas utile de vérifier s'ils sont nuls ou maximums; on rappelle que la remontée commence par les dernières colonnes).

7) Ecrire une fonction remonter qui admet une matrice M échelonnée réduite et renvoie une matrice diagonale pour la partie non augmentée.

8) Ecrire l'instruction faisant appel à la fonction remonter qui permet d'obtenir la matrice Ma1 suivante

```
[[  1.00000000e+00,   0.00000000e+00,   0.00000000e+00, 0.00000000e+00],
 [  0.00000000e+00,   1.00000000e+00,   0.00000000e+00, 2.00000000e+03],
 [ -0.00000000e+00,  -0.00000000e+00,  1.00000000e+00, -2.80000000e+03]]
```

Etape 4 : Récupération de la solution

9) Ecrire une instruction qui permet de récupérer la solution du problème à partir de la matrice Ma échelonnée réduite et remontée :

10) ¨Donner les inconnues de liaison solution du problème en précisant les unités correspondantes.

ANNEXE Tableaux avec la bibliothèque **numpy**

import numpy as np

Vecteur ligne	v=np.array([1, 2, 3])
Vecteur colonne	v= np.array([[1],[2],[3]])
Tableau à 2 dimensions	M= np.array([[1,2,3],[4,5,6]])
Accéder à un élément	v[0], M[0,1]
Accéder au dernier élément, et l'avant dernier	v[-1], v[-2]
Extraire la 2ème ligne ou 2ème colonne	M[1,:] ou M[:,1]
Extraire une portion de tableau (2 premières colonnes)	M[:,0:2]
Extraire des éléments d'un tableau par leurs indices	M[0,(2,1)]
Copier un tableau dans une autre variable	w=v.copy()
Multiplication élément par élément	v*w
Maximum et minimum d'un tableau	v.max(0), v.min(0)
Indice *i* du maximum	v.argmax(0)
Résolution de système matriciel M.X=Y	np.linalg.solve(M,Y)