TP : Manipulation de matrices

Pour ce TP, on n'utilisera pas le type `ndarray` mais on représentera une matrices comme la liste de ses lignes, elles-mêmes représentées comme des listes.
Par exemple, la matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ sera représentée par `A=[[1,2,3],[4,5,6]]`.

1 Pour commencer

a) Si on affiche une liste de listes, on a un affichage qui ressemble à `[[1,2,3],[4,5,6]]`. Proposer une procédure `visible` qui, étant donnée une liste L, affiche tous ses éléments en insérant un saut de ligne après chaque élément.

> Pour mémoire, une **procédure** est analogue à une **fonction** mais ne renvoie rien : il n'y a pas de `return`. La syntaxe pour créer une procédure en Python est la même que pour une fonction.

b) Créer une fonction `verifie` qui, étant donnée une liste de listes, vérifie qu'il s'agit d'une matrice, c'est-à-dire que les lignes ont toutes la même longueur.

c) Créer une fonction `matrice_nulle` qui, étant donnés deux entiers naturels non-nuls n et p, renvoie la matrice nulle de taille $n \times p$.

d) (Facultatif :) adapter la fonction `matrice_nulle` afin qu'on puisse ne saisir qu'un paramètre si l'on souhaite une matrice carrée.

e) Créer une fonction `matrice_id` qui, étant donné un entier naturel non-nul n, renvoie la matrice identité de taille n.

f) Créer une fonction `transpose` qui, étant donnée une matrice M, renvoie la matrice ${}^tM$.

2 Pivot de Gauss

2.1 Copie superficielle Vs copie récursive

Créer une liste quelconque `L`, copier la liste `L` dans une nouvelle variable `L2`, ajouter un élément dans `L`, que se passe-t-il pour `L2` ?

> Pour économiser de la mémoire, Python n'a pas vraiment dupliqué l'objet `L` mais a créé un nouvel objet `L2` qui pointe vers l'objet `L`. C'est ce qu'on appelle une **copie superficielle**.
>
> On peut forcer Python à dupliquer un objet en faisant une **copie récursive** à l'aide de la fonction `deepcopy` du module `copy`. (L'adjectif *récursif* vient de la façon dont on parcoure l'objet à dupliquer).

Dans la suite, on souhaite que les fonctions créées renvoient de nouvelles matrices mais n'altèrent pas celles qui sont passées en paramètres. On commencera donc par créer des copies à l'aide de `deepcopy`.

(Facultatif :) dans le corrigé, vous trouverez deux versions de `matrice_nulle`. Modifiez le code de `matrice_id` pour qu'elle fasse appel à `matrice_nulle2` et constatez qu'il y a un problème. Comprenez-vous pourquoi ?

2.2 Opérations élémentaires sur les lignes

Mettre en oeuvre les opérations élémentaires en programmant trois fonctions `permutation`, `dilatation` et `transvection`.

2.3 Algorithme de Gauss pour résoudre un système de Cramer

Soit $AX = B$ un système linéaire représenté sous forme matricielle avec A une matrice, X et B deux colonnes (de taille convenable). On dit que le système est de Cramer lorsque A est une matrice carrée inversible. Le système admet alors une unique solution qui est $A^{-1}B$.

On se limite dans la suite à la résolution des systèmes de Cramer, ce qui simplifie la programmation (on exclut les systèmes incompatibles, les notions d'inconnues principales et secondaires).

Lorsqu'on a effectué à la main l'algorithme de Gauss en cours, on a cherché à avoir des pivots égaux à 1 car cela simplifiait les calculs. Pour la mise en oeuvre informatique, la problématique est différente : on cherche à limiter les erreurs de calculs. Pour ce faire, dans chaque colonne, on prendra comme pivot le coefficient le plus grand.

a) Proposer une fonction `ajoute_col` qui, étant données une matrice A et une colonne B (ayant autant de lignes que A) renvoie la matrice augmentée $(A|B)$.

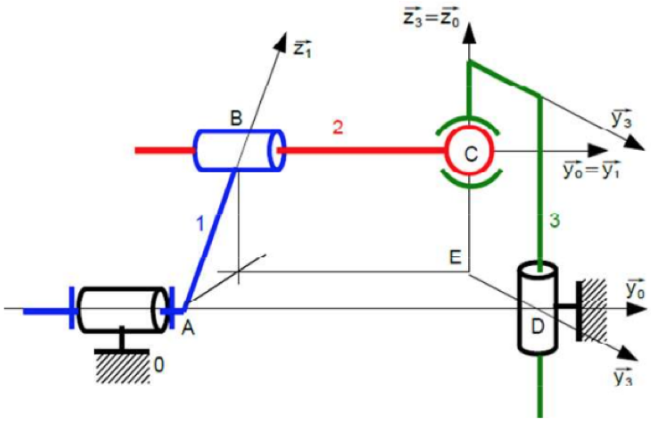
b) Créer une fonction `gauss` qui met en oeuvre l'algorithme de Gauss sur une matrice augmentée $(A|B)$ pour aboutir à $(A'|B')$ avec A' qui est échelonnée et équivalente en lignes à A.

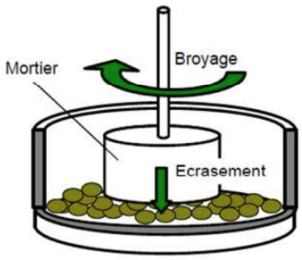
c) Créer une fonction `resol_sys` qui, étant données la matrice A et le second membre B renvoie la solution du système $AX = B$.

d) (Faculatif :) estimer la complexité des fonctions `Gauss` et `resol_sys`.

3 Application à un problème de SII

Lors du premier semestre, en S.I.I, nous nous sommes intéressés aux actions mécaniques présentes dans un broyeur (Exercice 3 du DS4 de S.I.I). La fonction principale de ce système est de réduire en poudre des matériaux dur et cassant (levures, sels, substances homéopathiques, ...). Pour réduire en poudre les matériaux, on utilise un mortier (pièce cylindrique) qui effectue un mouvement combiné de rotation et de translation, afin de broyer et d'écraser les agglomérats de matériaux. On donne ci-dessous le paramétrage mécanique du système.





Données : $\overrightarrow{AB} = R.\overrightarrow{z_1}$; $\overrightarrow{BC} = \lambda(t).\overrightarrow{y_1}$; $\overrightarrow{AD} = d.\overrightarrow{y_1}$; $\overrightarrow{EC} = h(t).\overrightarrow{z_0}$; $\overrightarrow{ED} = L.\overrightarrow{y_3}$

Les dimensions géométriques du système sont les suivantes : R = 3 cm, d = 7 cm, L = 4 cm.

On pose :
$R_0 = (A, \overrightarrow{x_0}, \overrightarrow{y_0}, \overrightarrow{z_0})$ le repère lié au bâti 0 ;
$R_1 = (A, \overrightarrow{x_1}, \overrightarrow{y_1}, \overrightarrow{z_1})$ le repère lié au volant 1 tel que $\overrightarrow{y_0} = \overrightarrow{y_1}$ et $(\overrightarrow{x_0}, \overrightarrow{x_1}) = \theta_{10}$;
$R_2 = (B, \overrightarrow{x_2}, \overrightarrow{y_2}, \overrightarrow{z_2})$ le repère lié à la bielle 2 tel que $\overrightarrow{y_1} = \overrightarrow{y_2}$ et $(\overrightarrow{x_1}, \overrightarrow{x_2}) = \theta_{21}$;
$R_3 = (C, \overrightarrow{x_3}, \overrightarrow{y_3}, \overrightarrow{z_3})$ le repère lié au mortier 3 tel que $\overrightarrow{z_0} = \overrightarrow{z_3}$ et $(\overrightarrow{x_0}, \overrightarrow{x_3}) = \theta_{30}$;
Dans cette étude, nous considérerons que le système se trouve dans la position $\theta_{10} = \theta_{30} = \frac{\pi}{4}$.

Les actions mécaniques exercées par le matériau à écraser sur le mortier 3 sont modélisables par le torseur suivant :

$$\{T_{P\to 3}\} = {}_D\begin{Bmatrix} Z_P.\overrightarrow{z_0} \\ N_P.\overrightarrow{z_0} \end{Bmatrix}$$

avec N_P = 0,1 Nm et $Z_P = 5N$.

La résolution proposée dans le DS de S.I.I avait pour but de déterminer le couple à fournir par l'actionneur ainsi que les efforts de liaisons. Les différentes étapes de cette résolution étaient les suivantes :
- Isolement de la pièce 1 et application du principe fondamentale de la statique (PFS) en moment, au point A, suivant $\overrightarrow{y_1}$;
- Isolement de la pièce 2 et application du PFS en résultante ;
- Isolement de la pièce 3 et application du PFS en moment au point D, suivant $\overrightarrow{z_0}$.

À partir de ces isolements, il est alors possible d'obtenir le système de 6 équations à 6 inconnues suivant (les inconnues sont C_{01}, $X_{1\to2}$, $Z_{1\to2}$, $X_{2\to3}$,$Y_{2\to3}$ et $Z_{2\to3}$) :

$$\begin{cases} C_{01} - R.X_{1\to2}.\cos(\theta_{10}) + R.Z_{1\to2}.\sin(\theta_{10}) &= 0 \\ X_{1\to2} - X_{2\to3} &= 0 \\ Y_{2\to3} &= 0 \\ Z_{1\to2} - Z_{2\to3} &= 0 \\ Z_P + Z_{2\to3} &= 0 \\ N_P + Y_{2\to3}.L.\sin(\theta_{30}) + X_{2\to3}.L.\cos(\theta_{30}) &= 0 \end{cases}$$

Ce système d'équations peut être mis sous la forme matricielle $A.X = B$ où la matrice X contient les inconnues du problème :

$$\begin{pmatrix} 1 & -R.\cos(\theta_{10}) & R.\sin(\theta_{10}) & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & L.\sin(\theta_{30}) & L.\cos(\theta_{30}) \end{pmatrix} \begin{pmatrix} C_{01} \\ X_{1\to2} \\ Z_{1\to2} \\ X_{2\to3} \\ Y_{2\to3} \\ Z_{2\to3} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -Z_P \\ -N_P \end{pmatrix}$$

a) Utiliser la fonction `resol_sys` pour trouver les 6 inconnues.

b) Vérifier la valeur de ces inconnues en résolvant le système d'équation à la main.