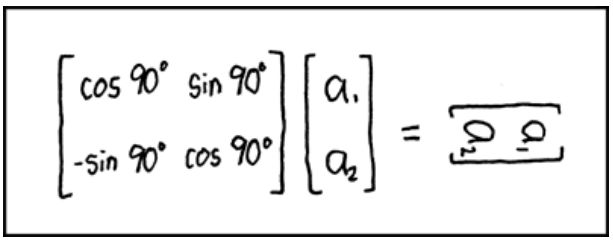
TP12 Pivot de Gauss

As usual, l'invitation GitHub pour votre dossier de travail vous a été envoyée par mail.

Partie I — Matrices



xkcd.com

```
In fact, draw all your rotational matrices sideways.
Your professors will love it!
And then they'll go home and shrink.
```

Nous allons voir les matrices comme une liste de listes représentant les lignes. Par exemple, la matrice $M = \begin{bmatrix} 5 & 3 \\ 0 & 7 \\ 4 & 1 \end{bmatrix}$ sera crée avec :

```
>>> M=[[5,3],[0,7],[4,1]]
>>> M
[[5, 3], [0, 7], [4, 1]]
>>> M[0]
[5, 3]
>>> M[2][1]
1
```

On accède alors à la première ligne de M par `M[0]` (Hé oui...Python commence sa numérotation à 0...). Cette première ligne est elle-même une liste. Pour accéder au deuxième coefficient de la troisième ligne, on utilise `M[2][1]`. Nous allons à présent construire tous les outils naturels nous permettant de manipuler les matrices.

1. Que fait la fonction `deepcopy` du module `copy` ? Pourquoi en aura-t-on besoin ?
2. Créer une fonction `lignes(M)` qui renvoie le nombre de lignes de la matrice M. (on pourra utiliser la fonction `len()`)
3. Créer une fonction `colonnes(M)` qui renvoie le nombre de colonnes de la matrice M. (on pourra utiliser la fonction `len()`)

Stop GitHub

Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

4. Pour plus de lisibilité, on crée la fonction `affiche(M)` suivante :

```
def affiche(M):
    '''Procédure simple d'affichage.'''
    for i in range(lignes(M)):
        print(M[i])
```

   Que fait-elle ? En quoi peut-elle vous être utile pour vos tests personnels ?

5. On fournit aussi la fonction `matrice_nulle(n,p)` suivante qui fabrique une matrice à n lignes et p colonnes ne contenant que des 0.

```
def matrice_nulle(n,p):
    '''Renvoie une matrice nulle à n lignes et p colonnes'''
    return [[0 for j in range(p)] for i in range(n)]
```

   On pourra s'inspirer de la syntaxe employée ici pour faciliter l'implémentation des fonctions suivantes.

6. Écrire une fonction `identite(n)` qui renvoie la matrice identité à n lignes et n colonnes.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

7. Écrire une fonction `matrice_constante(n,p,v)` qui renvoie une matrice à `n` lignes et `p` colonnes dont tous les coefficients sont égaux à `v`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

8. Écrire une fonction `transpose(M)` qui renvoie la transposée de la matrice M.
   NB : la fonction, comme toutes les suivantes, ne doit **pas** modifier la matrice `M` donnée en argument

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

9. Écrire une fonction `trace(M)` qui renvoie la trace d'une matrice carrée M.
   Il faudra vérifier que la matrice M est bien carrée à l'aide de l'instruction `assert`.
   Rappel : la trace d'une matrice carrée est égale à la somme de ses coefficients diagonaux.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

10. Écrire une fonction `somme_matrices(M,N)` qui prend comme arguments deux matrices M et N et renvoie leur somme.
    Il faudra vérifier que les matrices à additionner ont des tailles compatibles via des instructions `assert`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

11. Écrire une fonction `produit_matrice_avec_scalaire(M,k)` qui prend comme arguments une matrice M et un scalaire k renvoie la matrice kM.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

12. Écrire une fonction `multiplication_matrices(M,N)` qui prend comme arguments deux matrices M et N et renvoie leur produit.
    Il faudra vérifier que les matrices à multiplier ont des tailles compatibles via des instructions `assert`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

13. Écrire une fonction `puissance_matrice(M,k)` qui prend comme arguments une matrice carrée M et un entier naturel k et renvoie la matrice M^k.
    Il faudra vérifier que la matrice M est bien carrée à l'aide de l'instruction `assert`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Partie II

Pivot de Gauß

On considère un système linéaire carré de n équations à n inconnues $(x_1, \dots, x_n) \in \mathbb{R}^n$

$$(\mathrm{S}) \begin{cases} a_{11}\,x_1 + a_{12}\,x_2 + \cdots + a_{1j}\,x_j + \cdots + a_{1n}\,x_n = b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + \cdots + a_{2j}\,x_j + \cdots + a_{2n}\,x_n = b_2 \\ \quad\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ a_{i1}\,x_1 + a_{i2}\,x_2 + \cdots + a_{ij}\,x_j + \cdots + a_{in}\,x_n = b_i \\ \quad\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ a_{n1}\,x_1 + a_{n2}\,x_2 + \cdots + a_{nj}\,x_j + \cdots + a_{nn}\,x_n = b_n \end{cases}$$

où $\forall i \in [\![1, n]\!]$, $\forall j \in [\![1, n]\!]$, $a_{ij}, b_i \in \mathbb{R}$.

On a déjà vu que l'on peut voir ce système sous forme matricielle : (S) $AX = B$ avec $A = (a_{ij})_{\substack{1 \leqslant i \leqslant n \\ 1 \leqslant j \leqslant n}}$, élément de $\mathscr{M}_n(\mathbb{R})$ la matrice du système, $B = (b_i)_{1 \leqslant i \leqslant n}$, élément de $\mathscr{M}_{n1}(\mathbb{R})$, la matrice colonne des seconds membres du système, que l'on identifie à un vecteur de $\mathbb{R}^n$ et $X = (x_i)_{1 \leqslant i \leqslant n}$, élément de $\mathscr{M}_{n1}(\mathbb{R})$, la matrice colonne des inconnues du système que l'on identifie aussi à un vecteur de $\mathbb{R}^n$.

La fonction `pivot_gauss` prend en paramètre la matrice A donnée sous forme de liste de listes et le vecteur B donné simplement sous forme d'une liste de n réels : $B = [b_1, \dots, b_n]$ et renvoie le vecteur X sous forme d'une liste de n réels : $[x_1, \dots, x_n]$. On suppose que le système (S) est de Cramer c'est à dire qu'il n'a qu'une seule solution ou encore que la matrice A est inversible.

Comme on l'a vu en cours on va d'abord implémenter les opérations élémentaires nécessaires à la résolution d'un système par la méhode du pivot de Gauß. Néanmoins, contrairement au cas du cours, on va fusionner la matrice carrée A et la matrice colonne B pour que les opérations d'échange et de transvection se fassent « tout seul ».

1. Écrire une fonction `fusion(A,B)` qui fusionne la matrice A et la matrice colonne B, c'est-à-dire qu'elle doit renvoyer au final une matrice à n lignes et $n+1$ colonnes (que l'on nommera `A_B` par la suite) dont les n premières colonnes sont celles de la matrice A et la dernière correspond à la matrice colonne B. NB : on pourra tout de suite penser à généraliser le procédé de sorte que l'on puisse donner pour B une matrice à n lignes et p colonnes pour, par exemple, résoudre plusieurs systèmes « en parallèle » ou encore l'utiliser pour inverser la matrice A dans la prochaine section.

Stop GitHub

Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

2. Écrire une fonction `separation(A_B)` qui prend en entrée la matrice fusionnée précédente et renvoie un couple (A, B) de matrices, l'une de taille $n \times n$ et l'autre de taille $n \times p$ où p est à déterminer en fonction de la taille de la matrice `A_B` donnée en entrée.

Stop GitHub

Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

3. Écrire une fonction `cherche_pivot(A_B,i)` qui recherche le meilleur (au fait, qu'est ce que cela veut dire ?) pivot pour la i-ème variable dans les lignes suivant la ligne i courante. Cette fonction doit renvoyer le numéro de la ligne du pivot.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

4. Écrire une fonction `echange_lignes(A_B,i,j)` qui échange la i-ème et la j-ième ligne de la matrice `A_B`. Cette fonction ne renvoie rien mais modifie la matrice `A_B`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

5. Écrire une fonction `transvection(A_B,k,i,alpha)` qui effectue la transvection $L_k \leftarrow L_k - \alpha.L_i$. Cette fonction ne renvoie rien mais modifie la matrice `A_B`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

6. Écrire une fonction `dilatation(A_B,i,alpha)` qui effectue la dilatation $L_i \leftarrow \alpha.L_i$. Cette fonction ne renvoie rien mais modifie la matrice `A_B`.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

7. Écrire une fonction `pivot_gauss(A,B)` qui rend la solution X du système $AX = B$ sous forme d'une liste (ou d'une matrice $n \times p$ avec p le nombre de colonnes de la matrice B donnée en entrée). Il faudra vérifier que la matrice A est bien carrée et que le vecteur B a bien le même nombre de lignes que A.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Partie III

D'autres fonctions sur les matrices

1. Le déterminant d'une matrice carrée, que l'on définira proprement en mathématiques un peu plus tard vérifie les propriétés suivantes :
   - le déterminant d'une matrice triangulaire est le produit de ses coefficients diagonaux ;
   - le déterminant ne change pas lorsque l'on applique une transvection à la matrice ;
   - lorsque l'on échange deux lignes (respectivement colonnes) d'une matrice carrée, son déterminant est multiplié par -1.

   Calculer (à la maison et à la main) le déterminant de la matrice $M = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 4 & 1 \\ 1 & 0 & 1 \end{bmatrix}$.

   Écrire une fonction `determinant(M)` qui renvoie le déterminant d'une matrice carrée M (attention à penser à traiter le cas non inversible où le déterminant est nul). Il faudra vérifier que la matrice M est bien carrée.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

2. Écrire une fonction `inversion(M)` qui renvoie l'inverse d'une matrice carrée M. Il faudra vérifier que la matrice M est bien carrée.

Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

3. Écrire une fonction `puissance_relative_matrice(M,k)` qui prend comme arguments une matrice carrée M (que l'on suppose inversible) et un entier relatif k et renvoie la matrice M^k. Il faudra vérifier que la matrice M est bien carrée.

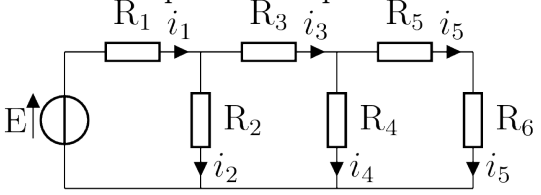
Stop GitHub
Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

Partie IV

Applications du pivot de Gauß

En physique, on a souvent des système linéaires à résoudre. En voici deux petits exemples.

IV.1 Réseau en électrocinétique



On considère le circuit suivant où les trois lois des mailles et deux lois des nœudss'écrivent

$$E = R_1\,i_1 + R_2\,i_2 \qquad R_2\,i_2 = R_3\,i_3 + R_4\,i_4 \qquad R_4\,i_4 = R_5\,i_5 + R_6\,i_5 \qquad i_1 = i_2 + i_3 \qquad \text{et} \qquad i_3 = i_4 + i_5$$

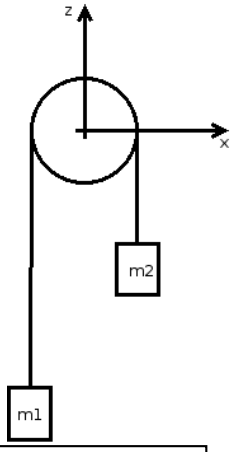
Les valeurs des paramètres E et R sont respectivement $E = 5{,}0$ V, $R_1 = 100\ \Omega$, $R_2 = R_3 = 220\ \Omega$, et $R_4 = R_5 = R_6 = 100\ \Omega$.

Écrire une fonction `reseau_electrocinetique()` qui renvoie les valeurs des 5 courants i_1 à i_5.

Stop GitHub

Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.

IV.2 Poulie accrochée avec deux masses



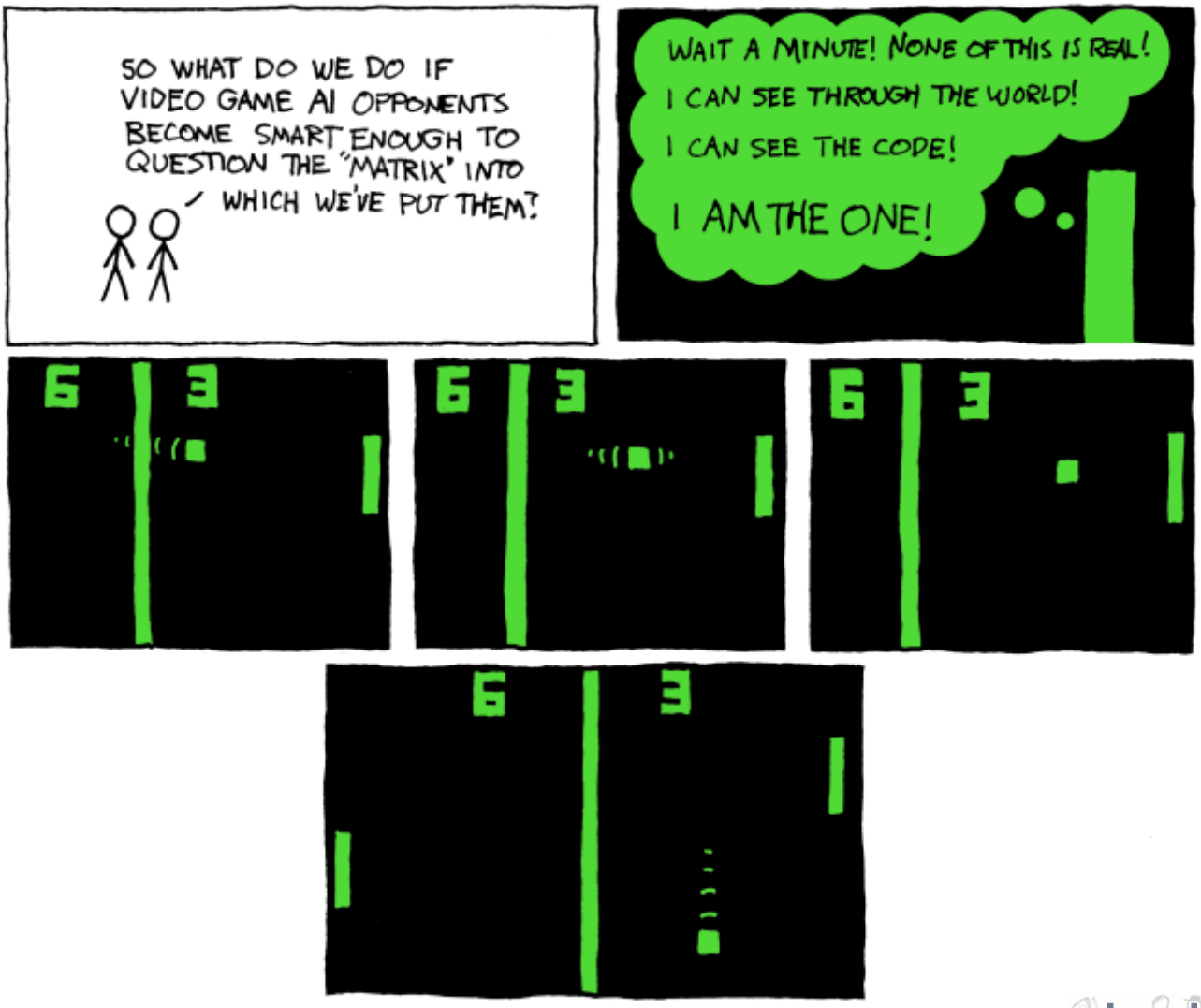
On considère le système mécanique ci-contre où $I_\Delta = 5.10^{-3}$ kg.m^2, $m_1 = 100$ g, $m_2 = 200$ g et $R = 10$ cm (rayon de la poulie). Le but est de déterminer les tensions T_1 et T_2 ainsi que l'accélération angulaire $\ddot{\theta}$ de la poulie. L'écriture des deux relations fondamentales de la dynamique pour les deux masses ainsi que du théorème du moment cinétique scalaire pour la poulie donne les équations

$$-m_1\,R\ddot{\theta} = T_1 - m_1 g \qquad m_2\,R\ddot{\theta} = T_2 - m_2 g \qquad \text{et} \qquad I_\Delta\,\ddot{\theta} = R\,(T_1 - T_2)$$

Écrire une fonction `poulie_a_deux_masses()` qui renvoie les valeurs de $\ddot{\theta}$, T_1 et T_2, respectivement en rad.s^{-2}, newton et newton. On rappelle que $g = 9{,}81$ m.s^{-2}.

Stop GitHub

Allez sur Github Desktop pour faire un commit. Choisissez vous-même (avec pertinence) le résumé. Pensez aussi à appuyer sur le bouton «Push origin» en haut à droite pour mettre à jour sur le web.



Following this, the pong paddle went on a mission to destroy Atari headquarters and, due to a mixup, found himself inside the game The Matrix Reloaded.
Boy, was THAT ever hard to explain to him.