Feuille de TP N°02 – Manipulation des Vecteurs et Matrices

Le but de ce TP et d'apprendre à manipuler et utiliser les fonctions sur les vecteurs et les matrices dans Matlab, un langage qui offre une variété d'opérations sur les vecteurs et les matrices, notamment des opérations arithmétiques, des fonctions de recherche et de tri … etc.

Rappel de cours :

- Quelles sont les différentes méthodes permettant de créer des vecteurs ou des matrices dans Matlab ?
  Il y a plusieurs façons pour créer des vecteurs ou/et des matrices dans Matlab, celles que les étudiants ont vu dans le cours sont :
  - **Utiliser les crochets pour insérer les valeurs manuellement. Les éléments appartenant à la même ligne seront séparés par des virgules ou des espaces, et les différentes lignes seront séparées par des point-virgules ou des retours à la ligne.**
    Exemple :
    A = [1 2 3]; % vecteur ligne contenant les valeurs 1, 2 et 3.
    B = [3; 4; 5]; % vecteur colonne contenant les valeurs 3, 4 et 5.
    C = [6 7; 8 9]; % une matrice de deux lignes et deux colonnes.
  - **Utiliser l'opérateur « : » pour initialiser un vecteur qui contient des éléments séparés par la même distance.**
    Exemple :
    X = 1:2:10; % initialise un vecteur X qui contient tous les nombres inférieurs ou égaux à 10 à partir de 1 avec une incrémentation de deux.
  - **Utiliser des fonctions qui créent des vecteurs/matrices, plusieurs fonctions existent, celles vu dans le cours sont.**
    - **linspace(a, b, n) : crée un vecteur contenant 'n' valeurs distribuées uniformément sur l'intervalle [a, b].**
    - **zeros(m, n) : crée une matrice de 'm' lignes et 'n' colonnes contenant des zéros.**
    - **ones(m, n) : comme zéros, mais génèrent une matrice contenant des uns.**
    - **Rand(m, n) : génère une matrice contenant des valeurs aléatoires.**
    - **… etc**

    Pour les fonctions zeros, ones, rand … etc, il est possible de donner un seul paramètre, et dans ce cas la fonction génère une matrice carrée. Il est aussi possible de les utiliser pour générer des vecteurs en passant « 1 » comme paramètre pour le nombre de lignes ou de colonnes.
- Quelle est la condition nécessaire pour pouvoir effectuer une concaténation horizontale ou verticale de deux matrices (ou vecteurs).
  Pour une concaténation horizontale, il faut que le nombre de lignes soit le même, pour une concaténation verticale, il faut que le nombre de colonnes soit le même.
- Que font les fonctions Matlab suivantes (vous pouvez utiliser la commande « help ») :
  - diag.
    Lorsqu'on lui passe une matrice comme paramètre, elle retourne son diagonal principal. Par contre, si on lui passe un vecteur, elle l'utilise pour créer une matrice diagonale.
  - tril.

Retourne la partie triangulaire inférieure d'une matrice. Cela signifie que tous les éléments au-dessus de la diagonale sont remplacés par zéros.

- triu.
  Comme « tril » mais retourne la partie supérieure.

Exercice 01 : créer des vecteurs et matrices simples

- Créer un vecteur ligne ***a*** qui contient les éléments suivants : ***9, 0, -2, -1, -1***.
  `a = [9, 0, -2, -1, -1]`
- Créer un vecteur colonne ***b*** qui contient les éléments suivants : ***8, -9, 1, -6, 8***.
  `b = [8; -9; 1; -6; 8]`
- Créer un vecteur ligne ***c*** qui contient tous les multiples de ***3*** entre ***0*** et ***20*** (intervalle fermé).
  `c = 0:3:20`
- Créer un vecteur colonne ***d*** qui contient ***20*** valeurs distribuées uniformément entre ***10*** et ***15*** (intervalle fermé).
  `d = linspace(10, 15, 20)`
- Créer la matrice suivante :

$$E = \begin{pmatrix} 6 & 43 & 2 & 11 & 87 \\ 12 & 6 & 34 & 0 & 5 \\ 34 & 18 & 7 & 41 & 9 \end{pmatrix}.$$

  `E = [6, 43, 2, 11, 87; 12, 6, 34, 0, 5; 34, 18, 7, 41, 9]`
- Utiliser des fonctions Matlab pour déclarer les matrices suivantes :
  - F : Une matrice de 3 lignes et 2 colonnes contenants des zéros (0).
    `F = zeros(3, 2)`
  - G : Une matrice de 2 lignes et 5 colonnes contenants des uns (1).
    `G = zeros(2, 5)`
  - H : Une matrice carrée identité de taille (4).
    `H = eye(4)`
  - I : Une matrice de 5 lignes et 4 colonnes contenants des valeurs aléatoires.
    `I = rand(5, 4)`

Exercice 02 : manipuler les vecteurs

- Recréer les vecteurs ***a*** et ***b*** de « l'exercice 01 » (ou les utiliser directement s'ils sont toujours déclarés) puis calculer :
  ```
  a = [9, 0, -2, -1, -1]
  b = [8; -9; 1; -6; 8]
  ```
  - Le produit scalaire des vecteurs ***a*** et ***b***.
    `a * b`
    % (b * a) n'est pas juste car la multiplication matricielle n'est pas commutative.
  - Le produit élément à élément des vecteurs ***a*** et ***b***.
    'a' est un vecteur ligne de 5 éléments, alors que 'b' est un vecteur colonne de la même taille, il faut avoir transposé l'un des deux vecteurs pour pouvoir effectuer la multiplication « élément à élément » avec l'opérateur « .* ». Les instructions suivantes sont justes.
    ```
    a' .* b
    a .* b'
    b' .* a
    b .* a'
    ```

- En utilisant les vecteurs ***a*** et ***b***, créer un vecteur ligne ***c*** qui contient les ***4*** premiers éléments du vecteur ***a*** et les ***4*** derniers éléments du vecteur ***b***.
  `c = [a(1:4), b(end-3:end)']`
  Il faut transposer les valeurs extraites à partir de 'b' vu que le dernier est un vecteur colonne.
- A partir du vecteur ***c***, créer les deux vecteurs colonnes suivants :
  - ***d*** qui contient tous les éléments ayant des indices pairs de ***c***.
    `d = c(0:2:end)'`
  - ***e*** qui contient tous les éléments ayant des indices impairs de ***c***.
    `e = c(1:2:end)'`
- Créer le vecteur ***f*** contenant les ***100*** éléments suivants :

$$f = \left(1 \quad \frac{1}{4} \quad \frac{1}{9} \quad \frac{1}{16} \quad \frac{1}{25} \quad \dots\dots \quad \frac{1}{100^2}\right).$$

  `f = 1 ./ ((1:100) .^ 2)`
  1. **Générer un vecteur contenant les valeurs de 1 à 100 =>** `1:100`
  2. **Calculer la puissance 2 de chaque élément =>** `(1:100) .^ 2`
  3. **Puis calculer l'inverse de chaque élément =>** `1 ./ ((1:100) .^ 2)`
- Supprimer le dernier élément de ***c***.
  `c(end) = []`
- Remplacer l'avant dernier élément de ***c*** par la valeur ***5***.
  `c(end-1) = 5`
- Remplacer les ***3*** premières éléments de ***c*** par les valeurs ***0***, ***1*** et ***2*** (en une seule instruction).
  `c(1:3) = [0, 1, 2]`
- Supprimer les ***40*** premiers éléments et les ***40*** derniers éléments de ***f*** en une seule instruction.
  `f([1:40, end-39:end]) = []`
- Calculer le produit des éléments du vecteur ***b***.
  `prod(b)`

Exercice 03 : manipuler les matrices

- Créer les deux matrices suivantes :

$$A = \begin{pmatrix} 2 & -8 \\ -3 & 6 \end{pmatrix}, B = \begin{pmatrix} -7 & 6 \\ -2 & -9 \\ 8 & 8 \end{pmatrix}$$

  `A = [2, -8; -3, 6]`
  `B = [-7, 6; -2, -9; 8, 8]`
- Remplacer l'élément à la ***troisième ligne*** et ***première colonne*** de ***B*** par l'élément à la ***première ligne*** et ***première colonne*** de ***A***.
  `B(3, 1) = A(1, 1)`
- Calculer ***A * B*** et ***B * A*** puis expliquer les résultats (ou les erreurs) obtenus.
  L'opération '*' calcule le produit matriciel de 'A' et 'B'. Pour la commande « `X = A * B` **», le calcul peut être effectué si et seulement si le nombre de colonnes de 'A' est égal au nombre de lignes de 'B'. Les éléments de X sont calculés tel que :**
  - **'X' a le même nombre de lignes que 'A'.**
  - **'X' a le même nombre de colonnes que 'B'.**
  - **L'élément à la ligne 'i' et colonne 'j' de 'X' est calculé par le produit scalaire de la $i^{ème}$ ligne de 'A' et la $j^{ème}$ colonne de 'B'.**

Dans ce cas, « A * B » ne peut pas être calculé car 'A' comporte 2 colonnes alors que 'B' comporte 3 lignes.

B * A est une multiplication matricielle qui donne le résultat suivant.

$$B * A = \begin{pmatrix} -32 & 92 \\ 23 & -38 \\ -8 & -16 \end{pmatrix}$$

- Définir les deux matrices :
  - *I2* une matrice identité de taille *2*.
    `I2 = eye(2)`
  - *I3* une matrice identité de taille *3*.
    `I3 = eye(3)`
- Calculer ce qui suit, puis expliquer les résultats (ou les erreurs) obtenus :
  - ***A * I2*** et ***A * I3***.
    - **A * I2 donne A car 'I2' est une matrice identité.**
    - **A * I3 => erreur car 'A' comporte 2 colonnes et 'I3' comporte 3 lignes.**
  - ***B * I3*** et ***I3 * B***.
    - **B * I3 => erreur car 'B' comporte 2 colonnes et 'I3' comporte 3 lignes.**
    - **I3 * B donne B % car 'I3' est une matrice identité.**
  - ***B * I2*** et ***I2 * B***.
    - **B * I2 donne B % car 'I2' est une matrice identité.**
    - **I2 * B => erreur car 'I2' comporte 2 colonnes et 'B' comporte 3 lignes.**
  - ***A .* I2*** et ***B .* I3***.
    - **A .* I2 donne une matrice diagonale ayant la même diagonale de 'A'. Notez qu'une multiplication élément à élément a été effectuée ici, les éléments de la diagonale ont été multipliés par '1' et les autres par '0'.**
    - **B .* I3 => erreur car le nombre de colonnes n'est pas le même sur 'B' et 'I3'. Notez que pour faire une multiplication élément à élément il faut que les deux matrices aient les mêmes dimensions (même nombre de lignes et de colonnes).**
  - ***A + 5*** et ***B * 2.***
    - **L'opération « A + 5 » va additionner la valeur '5' à toutes les cellules de la matrice A et retourner le résultat. La même chose pour la multiplication de la matrice 'B' par '2'.**
  - ***A ^ 2, B ^ 2, A .^ 2*** et ***B .^ 2.***
    - **L'opérateur de puissance '^' appliqué à une matrice permet de multiplier cette dernière par elle-même. Evidement il est impossible de multiplier une matrice qui n'est pas carrée par elle-même, donc « B ^ 2 » va donner une erreur.**
    - **L'opérateur '.^' par contre calcule la puissance de chaque élément de la matrice, donc il peut s'appliquer à toutes les matrices, qu'elles soient carrées ou pas.**
- Créer la matrice ***C*** suivante (pensez à créer chaque ligne avec « : » ou linspace d'abord) :

$$C = \begin{pmatrix} 1 & 4 & 7 & 10 & 13 \\ 72 & 66 & 60 & 54 & 48 \\ 0 & 0.25 & 0.5 & 0.75 & 1 \end{pmatrix}.$$

```
C = [linspace(1, 13, 5); linspace(72, 48, 5); linspace(0, 1, 5)]
```

- Concaténer les variables ***A, B, C, I2*** et ***I3*** pour créer les matrices suivantes :

$$D = \begin{pmatrix} 1 & 0 & -7 & -2 & 2 \\ 0 & 1 & 6 & -9 & 8 \\ 1 & 4 & 7 & 10 & 13 \\ 72 & 66 & 60 & 54 & 48 \\ 0 & 0.25 & 0.5 & 0.75 & 1 \end{pmatrix}, E = \begin{pmatrix} 1 & 4 & 7 & 10 & 13 \\ 72 & 66 & 60 & 54 & 48 \\ 0 & 0.25 & 0.5 & 0.75 & 1 \\ 1 & 0 & 0 & -7 & 6 \\ 0 & 1 & 0 & -2 & -9 \\ 0 & 0 & 1 & 2 & 8 \end{pmatrix}$$

$$et\ F = \begin{pmatrix} -7 & 6 & -7 & -2 & 2 \\ -2 & -9 & 6 & -9 & 8 \\ 2 & 8 & 1 & 0 & 0 \\ 2 & -8 & 0 & 1 & 0 \\ -3 & 6 & 0 & 0 & 1 \end{pmatrix}.$$

```
D = [I2, B'; C]
E = [C; I3, B]
F = [[B; A], [B'; I3]] % "F = [B; A, B'; I3]" ne marchera pas car la concaténation horizontale est prioritaire à la concaténation verticale.
```

- Supprimer la ***première*** et la ***dernière*** ligne de ***E*** (en une seule instruction).
  ```
  E([1, end], :) = []
  ```
- Supprimer la ***deuxième colonne*** de ***E***.
  ```
  E(:, 2) = []
  ```
- Remplacer la ***troisième colonne*** de ***F*** par la ***quatrième ligne*** de ***D***.
  ```
  F(:, 3) = D(4, :)' % il faut transposer le vecteur ligne extrait pour l'affecter à la colonne choisie.
  ```
- Extraire la ***première colonne*** de ***F*** dans un vecteur ligne ***v***.
  ```
  v = F(:, 1)'
  ```
- Calculer la somme des lignes, puis la somme des colonnes de ***F***.
  ```
  sum(F, 2) % Somme des lignes
  sum(F, 1) % Somme des colonnes
  ```