Opérations sur les matrices

1 Objectif

Objectif :

L'objectif de ce tp est d'effectuer les opérations de bases de l'algèbre linéaire :

- comment effectuer le produit de deux matrices ?
- comment inverser une matrice ?
- comment résoudre un système de Cramer ?
- comment calculer un déterminant ?

Nous verrons que ces opérations de bases, *simples* à programmer, ont des temps d'exécution longs par rapport aux commandes implantées dans les bibliothèques.

Il s'agit ici:

- d'écrire sous **Python** , le programme permettant d'effectuer le produit de deux matrices
- d'écrire sous **Python** , le programme permettant d'inverser une matrice
- de découvrir **Scilab** en écrivant les deux programmes précédant sous **Scilab** (si vous êtes motivés...)

Pour ce tp, le type *tableau* (`np.array([...])` sera préféré au type *liste*. La syntaxe pour accéder aux éléments est différentes. Retrouvez les principales commandes **Python** dans la partie **??**.

2 Opérations sur les matrices

2.1 Produit matriciel

Soient un couple de deux matrices $(A, B) \in \mathcal{M}_{m,n}(\mathbb{R}) \times \mathcal{M}_{n,o}(\mathbb{R})$. On note $P \in \mathcal{M}_{m,o}(\mathbb{R})$ le résultat du produit à droite de A par B (P = A.B).

Rappel Le coefficient $p_{i,j}$ de la matrice P est donné par la formule suivante : $[P]_{i,j} = p_{i,j} = \sum_{k=1}^{n} a_{i,k}.b_{k,j}$ (1)

Q - 1 : *A partir de la formule (1), construire le programme* ***Prodmat*** *ayant pour argument deux matrices* A *et* B *et comme résultat, le produit* A.B .

Remarque : On récupérera la taille des matrices grâce à la commande `.shape` et on veillera à la compatibilité des matrices lors des produits matriciels.

Q - 2 : *Tester votre programme avec la matrice triangulaire supérieure suivante* A *et son inverse* B *:*

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}, \text{ de déterminant 24, donc inversible}$$

Remarque : Pour obtenir B rapidement, on utilisera la commande **np.linalg.inv** .

2.2 Opérations élémentaires

Pour déterminer l'inverse d'une matrice inversible A, il suffit d'appliquer à la matrice identité I, les mêmes opérations élémentaires uniquement sur les lignes (ou uniquement sur les colonnes) qui permettent de passer de A à I.

Les opérations élémentaires sur les lignes (resp. sur les colonnes) sont telles que :

- on multiplie tous les termes d'une ligne (resp. d'une colonne) par une constante non nulle :

$$L_i \leftarrow \lambda.L_i \quad (\text{resp.} \quad C_i \leftarrow \lambda.C_i) \quad \text{avec} \quad \lambda \neq 0$$

- on ajoute à une ligne (resp. une colonne), une autre ligne (resp. colonne) de la matrice :

$$L_i \leftarrow L_i + \lambda.L_j \quad (\text{resp.} \quad C_i \leftarrow C_i + \lambda.C_j)$$

On introduit quelques fonctions permettant d'effectuer des opérations élémentaires sur les matrices pour pouvoir ensuite appliquer la méthode du pivot de Gauss ou inverser une matrice.

Q - 3 : *Construire une fonction* `permuation(A,i,j)` *permettant de permuter les lignes i et j de la matrice* A.

Q - 4 : *Construire une fonction* `ajoutligne(A,i,j,lamb)` *qui ajoute à la ligne i de* A *la ligne j de* A *multipliée par* `lamb`.

Q - 5 : *Construire une fonction* `multiplieligne(A,i,lamb)` *qui multiplie la ligne i de* A *par* `lamb`.

Q - 6 : *Construire une fonction* `rechpivot(L)` *qui détermine l'indice de l'élément de* L *ayant la plus grande valeur absolue.*

2.3 Calcul de l'inverse d'une matrice (inversible...), résolution d'un système de Cramer

Q - 7 : *Mettre en place, pour une matrice inversible* A *de dimension n et* Y *un vecteur de dimension n, une fonction* `pivot(A,Y)` *permettant de déterminer* X *tel que* A.X = Y. *La fonction* `pivot` *devra trianguliariser la matrice* A *puis résoudre le système* A'.X = Y' *ou* A' *est une matrice triangulaire supérieure construite à partir de* A.

Q - 8 : *Tester la fonction* `pivot(A,Y)` *dans les deux cas suivants :*

$$A = \begin{bmatrix} 2. & 1. & 3. \\ 1. & 3. & 1. \\ -2. & -1. & -1. \end{bmatrix}, \; Y = \begin{bmatrix} 3. \\ 6. \\ -1. \end{bmatrix} \quad \textit{et} \quad A = \begin{bmatrix} 1. & 1. & -1. & 1. \\ -2. & 1. & 2. & 1. \\ 2. & -1. & 1. & 2. \\ 1. & 2. & 2. & -3 \end{bmatrix}, \; Y = \begin{bmatrix} 3. \\ 0. \\ 12. \\ -7. \end{bmatrix}$$

Remarque : Ne pas oublier les points (liés aux virgules) en entrant les valeurs...

Q - 9 : *Pour une matrice inversible* A, *mettre en place une fonction* `invGauss(A)` *permettant d'obtenir la matrice inverse de* A. *La fonction devra rendre la matrice triangulaire supérieure par opérations sur les lignes, puis diagonale et enfin égale à la matrice identité, toujours par opérations sur les lignes.*

3 Syntaxes Python et Scilab

Syntaxes Python et Scilab ici.

3.1 Vecteurs

Description	Python	Scilab
Vecteur ligne	v=array([1, 2, 3])	v=[1, 2, 3] ou [1 2 3]
Vecteur colonne	v=array([[1],[2],[3]]) v=array([1, 2, 3])[:,newaxis]	v=[1 ; 2 ; 3]
Tableau à 2 dimensions	M=array([[1,2,3],[4,5,6]])	M=[1, 2, 3 ; 4, 5, 6]
Accéder à un élément	v[0], M[0,1]	v(1), M(1,2)
Accéder au dernier élément, et l'avant dernier	v[-1], v[-2]	v($), v($-1)
Dimensions d'un tableau	M.shape	size(M)
Extraire la 2ème ligne ou 2ème colonne	M[1,:] ou M[:,1]	M(2,:) ou M(:,2)
Extraire une portion de tableau (2 premières colonnes)	M[:,0:2]	M(:,1:2)
Extraire des éléments d'un tableau par leurs indices	M[0,(2,1)]	M([1],[3,2])
Séquence équirépartie d'entiers	range(1,11)	1:10
Séquence équirépartie quelconque	arange(0,10.1,0.1)	0:0.1:10
Tableau de zéros	zeros((2,3),float)	zeros(2,3)
Tableau de uns	ones((2,3),float)	ones(2,3)
Matrice identité	eye(3,3)	eye(3,3)
Copier un tableau dans une autre variable	w=v.copy()	w=v
Multiplication élément par élément	v*w	v.*w
Maximum et minimum d'un tableau	v.max(0), v.min(0)	max(v), min(v)
Indice i du maximum	v.argmax(0)	[m,i] = max(v)

3.2 Matrices

Description	Python	Scilab
Produit matriciel	dot(v,w)	v*w
Transposée	M.transpose()	M'
Résolution de système matriciel M.X=Y	linalg.solve(M,Y)	X=MY
Produit scalaire de deux vecteurs	vdot(v,w)	v'*w
Produit vectoriel	cross(v,w)	cross(v,w)
Déterminant	linalg.det(M)	det(M)
Inverse	linalg.inv(M)	inv(M)
Valeurs propres	linalg.eig(M)[0]	spec(M)
Vecteurs propres	linalg.eig(M)[1]	[v,l] = spec

4 Algorithmes

Algorithm 1 Inversion de matrice

entrée: A
M ← A # copier la matrice
déterminer la taille n, m de M
vérifier que la matrice est carrée
construire la matrice identité I_n
pour i de 1 à n **faire**
    rechercher le pivot k sous le coefficient $M_{i,i}$
    si $M_{k,i} = 0$ **alors**
        stop, la matrice n'est pas inversible
    fin si
    si $k \neq i$ **alors**
        inverser ligne i et ligne k
    fin si
    obtenir des 0 sous le coefficient diagonal $M_{i,i}$
    pour k de $i+1$ à n **faire**
        $L_k \rightarrow L_k - \frac{M_{k,i}}{M_{i,i}}.L_i$
    fin pour
fin pour
obtenir $M_{i,i} = 1$
pour i de 1 à n **faire**
    $L_i = L_i/M_{i,i}$
fin pour
éliminer tous les coefficients au dessus de $M_{i,i}$
pour i de n à 2 **faire**
    pour k de 1 à $i-1$ **faire**
        $L_k \rightarrow L_k - M_{k,i}.L_i$
    fin pour
fin pour
renvoi: I_n