TP Analyse 2, 02/03/22, Algorithme du pivot de Gauss

On veut obtenir informatiquement l'inverse d'une matrice quelconque, grâce au pivot de Gauss et à la matrice augmentée. Un rappel de cette méthode est disponible sur fr.wikipedia.org/wiki/Élimination_de_Gauss-Jordan
à la section "Calcul de l'inverse d'une matrice carrée".

On rappelle qu'on appelle *matrice identité de taille k*, notée I_k, la matrice carrée composée de 1 sur sa diagonale, et de 0 partout ailleurs.

Exemple: $I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Q.0 *Ne passez pas plus de quinze minutes sur cette question.*

Coder la multiplication de deux matrices.

Si vous avez sauté cette question, utilisez simplement le symbole @ pour la multiplication matricielle. Ainsi, $A@B$ doit vous renvoyer le produit de A et B.

1 Algorithme de Gauss sur des matrices simples

Q.1.1. Quel est A^{-1}, l'inverse de la matrice A ?

$$A = \begin{pmatrix} 6 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

Q.1.2. Coder une fonction prenant une matrice diagonale en entrée, et sortant son inverse. Votre fonction doit donc, lorsqu'on lui donne A, sortir la matrice A^{-1} calculée à la question précédente.

Q.1.3 On étudie maintenant le cas où la matrice a des valeurs non nulles sur sa diagonale et sur sa première ligne. Par exemple :

$$B = \begin{pmatrix} 6 & 1 & 5 & 7 & 6 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{pmatrix}.$$

Faire une fonction pour éliminer les composantes de la première ligne, sauf sa première composante. On se retrouve alors avec une matrice diagonale, cas précédemment traité. Vérifiez que votre code permet bien d'obtenir B^{-1}, l'inverse de B.

Q.1.4 Adapter l'algorithme précédent pour obtenir l'inverse d'une matrice triangulaire quelconque. Que donne-t-il pour la matrice C ?

$$C = \begin{pmatrix} 5 & 3 & 5 & 6 \\ 0 & 2 & 9 & 5 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Q.1.5 Quel est l'inverse de la matrice D ? On pourra ramener D à une matrice triangulaire, grâce à des opérations sur des lignes.

$$D = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 2 & 5 & 8 & 2 & 1 \\ 3 & 0 & 9 & 2 & 8 \\ 5 & 0 & 0 & 4 & 8 \\ 3 & 0 & 0 & 0 & 5 \end{pmatrix}.$$

En déduire un algorithme pour annuler toute la première colonne d'une matrice, sauf sa première case. On supposera que la première ligne est nulle, sauf sa première case.

Q.1.6 Généraliser votre réponse précédente pour obtenir l'inverse de n'importe quelle matrice. Tester votre code sur la matrice E.

$$E = \begin{pmatrix} 1 & 3 & 2 & 5 \\ 2 & 2 & 7 & 2 \\ 9 & 2 & 4 & 3 \\ 1 & 4 & 8 & 2 \end{pmatrix}.$$

Indication : vous savez annuler, en dessous de la diagonale, la première colonne. La question vous demande d'annuler, en dessous de la diagonale, chaque colonne, afin de ramener une matrice quelconque à une matrice triangulaire.

2 Complications

Q.2.1 Tester votre code de la partie 1 sur la matrice F.

$$F = \begin{pmatrix} 0 & 3 & 2 & 5 \\ 0 & 0 & 7 & 2 \\ 0 & 2 & 4 & 3 \\ 1 & 4 & 8 & 2 \end{pmatrix}.$$

Si votre code ne marche pas, pourquoi ? Comment le réparer ?

Q.2.2 Modifier votre algorithme afin d'obtenir aussi le déterminant. Que se passe-t-il sur la matrice G ? Quel est son inverse, et que dit votre code sur cette entrée ?

$$G = \begin{pmatrix} 1 & 3 & 2 & 5 \\ 2 & 9 & 7 & 2 \\ 1 & 5 & 4 & 3 \\ 1 & 9 & 8 & 2 \end{pmatrix}.$$

Q.2.3 (bonus) Pour rentrer dans des considérations plus pratiques, rappelons que l'addition de flottants n'est pas parfaite. Par exemple, $0.2 + 0.1$ est différent de 0.3.

Tester votre algorithme sur la matrice suivante, et le modifier si besoin :

$$H = \begin{pmatrix} 2 & 1 & 1 & 0.3 \\ 1 & 1 & 0 & 0.2 \\ 1 & 0 & 1 & 0.1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

On pourra par exemple transformer automatiquement les valeurs suffisamment petites en 0, ou tester si les nombres sont suffisamment petits plutôt que tester s'ils sont nuls.

Q.2.4 Tester votre algorithme sur des matrices aléatoires. Pour générer des matrices aléatoires, on pourra utiliser les outils en bas de cette feuille.

3 Application à la résolution d'un système linéaire

On se propose de résoudre le système linéaire suivant : $Ax = b$,
où $A \in M_{m,n}, x, b \in M_{m,1}$.
Si on connait A et b, on peut déterminer x de la manière suivante : $x = A^{-1}b$,
où A^{-1} désigne l'inverse de la matrice A.

Utiliser l'algorithme d'inversion de matrice mis en place pour résoudre le système suivant :

$$\begin{pmatrix} 2 & 5 & 3 & 1 \\ 9 & 3 & 4 & 0 \\ 0 & 1 & 5 & 6 \\ 8 & 7 & 2 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 7 \\ 1 \\ 3 \\ 5 \end{pmatrix}.$$

Quelques outils

- Pour gérer les tableaux et matrices, on utilise le package numpy. À ajouter en début de code :
  `import numpy as np`
- Pour créer une matrice diagonale, dont la liste des valeurs est x :
  `np.diag(x)`.
- Pour créer une matrice aléatoire de taille m, n :
  `np.random.random((m,n))`
- Pour récupérer la partie inférieure (resp. supérieure) d'une matrice M :
  `np.tril(M)` (et `np.triu(M)`)
- Pour récupérer la dimension d'un object X de type numpy.array ou numpy.matrix :
  `X.shape`