TD/TP 6 - Calculer l'inverse d'une matrice

I Inversion d'une matrice

Notons $(e_1, \ldots, e_n)$ la base canonique de $\mathbb{R}^n$ pour un entier $n \geq 1$. On notera en colonne les vecteurs de $\mathbb{R}^n$.

Considérons une matrice $A \in GL_n(\mathbb{R})$. Pour $1 \leq i \leq n$, la i-ème colonne de A^{-1} est égale au vecteur $u_i = A^{-1}e_i$. On a ainsi le système suivant :

$$A \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ . \\ 1 \\ . \\ 0 \end{bmatrix} \text{ où } u_i = \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} \text{ et } e_i = \begin{bmatrix} 0 \\ . \\ 1 \\ . \\ 0 \end{bmatrix}, \text{ le } 1 \text{ se trouvant sur la } i\text{-ème ligne.}$$

Il suffit donc de résoudre n systèmes linéaires pour calculer l'inverse de A.

Dans la pratique, on résout simultanément ces n systèmes linéaires. Pour cela, on applique la méthode du pivot de Gauss en travaillant en même temps sur l'ensemble des seconds membres.

1. Afin de représenter simultanément les n systèmes, on adjoint tout d'abord à A les n colonnes correspondant aux seconds membres (c'est-à-dire la matrice identité).
Par exemple si $n = 3$, on considère

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 1 \end{bmatrix}$$

2. En utilisant des pivots de Gauss récursivement sur les n colonnes de A, on triangularise les n systèmes.
Pour $n = 3$, on obtient n systèmes représentés par

$$\begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & b'_{11} & b'_{12} & b'_{13} \\ 0 & a'_{22} & a'_{23} & b'_{21} & b'_{22} & b'_{23} \\ 0 & 0 & a'_{33} & b'_{31} & b'_{32} & b'_{33} \end{bmatrix}.$$

3. Pour $1 \leq i \leq n$, on divise chaque ligne i par a'_{ii} de façon à obtenir des 1 sur la diagonale. (Les a'_{ii} sont tous non nuls, sinon la matrice de départ ne serait pas diagonalisable.)
Dans le cas $n = 3$, on obtient

$$\begin{bmatrix} 1 & a''_{12} & a''_{13} & b''_{11} & b''_{12} & b''_{13} \\ 0 & 1 & a''_{23} & b''_{21} & b''_{22} & b''_{23} \\ 0 & 0 & 1 & b''_{31} & b''_{32} & b''_{33} \end{bmatrix}.$$

4. Pour terminer, on transforme les lignes afin d'amener des 0 au-dessus de la diagonale. Pour cela on retranche le multiple par a''_{12} de la ligne 2 à la ligne 1, puis un multiple adéquat de la ligne 3 à la ligne 1, puis un multiple de la ligne 3 à la ligne 1, et ainsi de suite (en pratique, afin de limiter le nombre de calculs, on commence par la dernière colonne).
Pour $n = 3$, on a alors

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & b_{11} & b_{12} & b_{13} \\ 0 & 1 & 0 & b_{21} & b_{22} & b_{23} \\ 0 & 0 & 1 & b_{31} & b_{32} & b_{33} \end{array}\right] \text{ et ainsi } A^{-1} = \left[\begin{array}{ccc} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{array}\right].$$

II EN TD

Q.1) - Dérouler l'algorithme d'inversion d'une matrice décrit dans la première partie sur l'exemple suivant :

$$A = \left[\begin{array}{ccc} 0 & 1 & 3 \\ 1 & 0 & -2 \\ 2 & 1 & 1 \end{array}\right].$$

Q.2) - Quelle est la complexité de cet algorithme ?

Q.3) - En utilisant une partie de l'algorithme d'inversion, donner un algorithme permettant de calculer le déterminant d'une matrice.

Q.4) - Préparer le TP décrit dans la partie suivante.

Q.5) - Question facultative : si l'on considère uniquement des matrices à coefficients rationnels, comment peut-on implémenter cet algorithme avec des calculs exacts (et non des valeurs approchées) ?

III EN TP

Le but de ce TP est d'implémenter une classe permettant d'inverser une matrice à l'aide du pivot de Gauss (algorithme décrit dans la première partie). Les matrices seront représentées par des tableaux à deux dimensions.

Q.1) - On considère une nouvelle classe que l'on nommera par exemple **`PivotGauss`**. On commence par un constructeur prenant en entrée la matrice à inverser et initialisant un tableau représentant les n systèmes linéaires à résoudre (c.f. **I.1**).

Q.2) - Compléter la classe par une méthode permettant d'afficher le tableau représentant les n systèmes linéaires.

Q.3) - Implémenter au fur et à mesure les étapes de l'algorithme décrit dans la première partie. Afin de tenir compte des erreurs de calculs, on fixera une constante valant 10^{-8} au dessous de laquelle les valeurs seront considérées comme nulles.

Q.4) - Ajouter à la classe une méthode statique `Inverse(double[][] A)` qui prend en entrée une matrice A et l'inverse.

Q.5) - Compléter la classe par une méthode statique retournant le déterminant d'une matrice passée en argument.

Q.6) - Compléter la classe de façon à lever des exceptions si la matrice en entrée n'est pas carrée ou si elle n'est pas inversible.