TP2 : MÉTHODES DIRECTES DE RÉSOLUTION DE SYSTÈMES LINÉAIRES

Vous êtes invités à créer un répertoire `TP2` et un fichier `.m` pour chacun des exercices ci-dessous (comportant les lignes de code correspondant à la résolution de l'exercice).

En cas de blocage, commencez toujours par regarder l'aide !!

PRÉSENTATION ET INDICATIONS

La première partie du TP est consacrée à la factorisation LU d'une matrice A. On rappelle que cette factorisation n'existe que si toutes les sous-matrices principales de A sont inversibles. Dans une deuxième partie, on s'intéresse à une factorisation avec permutations, de type $PA = LU$. Elle peut s'appliquer à n'importe quelle matrice et être utilisée ensuite pour la résolution des systèmes linéaires.

Pour la programmation en MATLAB, on utilisera au mieux
la vectorisation pour éviter toutes les boucles inutiles.

1. FACTORISATION LU

On rappelle tout d'abord l'algorithme de factorisation LU. Il consiste en la construction successive des factorisations LU des n sous-matrices principales.

– Initialisation : $L_1 = 1$, $U_1 = a_{11}$.

– Pour tout $1 \leq k \leq n-1$,

$$L_{k+1} = \begin{array}{|c|c|}\hline L_k & \begin{matrix}0\\ \vdots \\ 0\end{matrix} \\ \hline Y & 1 \\ \hline \end{array} \quad \text{et} \quad U_{k+1} = \begin{array}{|c|c|}\hline U_k & X \\ \hline 0 \cdots 0 & z \\ \hline \end{array}.$$

avec $X \in \mathcal{M}_{k,1}(\mathbb{R})$, vecteur colonne, $Y \in \mathcal{M}_{1,k}(\mathbb{R})$, vecteur ligne, et $z \in \mathbb{R}$ solutions de

$$\left\{ \begin{array}{rcll} L_k X & = & (a_{i,k+1})_{1\leq i\leq k} & \text{(vecteur colonne)} \\ Y U_k & = & (a_{k+1,i})_{1\leq i\leq k} & \text{(vecteur ligne)} \\ z & = & a_{k+1,k+1} - YX. & \end{array} \right.$$

A chaque itération, la matrice U_k est inversible si $z \neq 0$.

1. Etant donnés une matrice `A`$= [a_{i,j}]$ et un entier `k` ($1 \leq k \leq n-1$), comment extrait-on avec Matlab le vecteur colonne $(a_{i,k+1})_{1\leq i\leq k}$? le vecteur ligne $(a_{k+1,i})_{1\leq i\leq k}$?

2. Programmer la fonction `[X] = descente(L,b)` qui résout un système triangulaire inférieur $LX = b$, puis la fonction `[X] = remontee(U,b)` qui résout un système triangulaire supérieur $UX = b$.

3. Écrire une fonction `compare(A,B)` qui vérifie si deux matrices sont identiques à 10^{-10} près et affiche en message en fonction. Tester les fonctions `descente` et `remontee` sur des matrices et des seconds membres de votre choix. Vous pourrez comparer vos résultats avec ceux obtenus avec la commande `\` de Matlab.

4. Programmer la fonction `[L,U] = factolu(A)` qui calcule la factorisation LU d'une matrice `A` quand celle-ci existe. On utilisera l'algorithme rappelé ci-dessus. Il peut être programmé de manière itérative (une boucle `for`) ou de manière récursive (pas de boucle mais un appel à `factolu` à l'intérieur de la fonction). Si la factorisation n'existe pas, on utilisera la fonction `error` pour faire afficher un message d'erreur.

5. Tester `factolu` sur des matrices de votre choix (celles vues en cours ou en TD par exemple) et comparer vos résultats avec ceux obtenus avec la commande `lu` de Matlab. Que constatez-vous ?

6. Programmer la fonction `[X] = resoudre_systeme_lu(A,b)` qui résout le système $AX = b$ en utilisant la factorisation LU de A, quand elle existe.

2. Factorisation LU avec permutations

2.1. Un exercice pour commencer.

Rappel : Étant donnée une permutation σ de $\{1, \dots, n\}$ la matrice de permutation $P_\sigma \in \mathcal{M}_n(\mathbb{R})$ est définie par $(P_\sigma)_{i,j} = \delta_{i,\sigma(j)}$ où $\delta_{i,j}$ est le symbole de Kronecker. Si τ est la permutation qui échange i et j on note $P_{i,j}$ la matrice P_τ.

On se propose pour commencer de démontrer le résultat suivant :

Théorème. Soit $A \in \mathcal{M}_n(\mathbb{R})$, avec $n \geq 2$. Il existe une matrice de permutation P, une matrice triangulaire inférieure à diagonale unité L et une matrice triangulaire supérieure U telles que

$$PA = LU.$$

On notera que la matrice A n'est pas supposée inversible, et par conséquent la matrice U non plus. Le théorème va se démontrer par récurrence sur n.

1. Montrer le résultat pour $n = 2$. On pourra distinguer les cas : $a_{11} = a_{21} = 0$; $a_{11} = 0$ et $a_{21} \neq 0$; et finalement $a_{11} \neq 0$.

2. On suppose maintenant que le résultat est vrai pour toute matrice $A \in \mathcal{M}_n(\mathbb{R})$. Soit $A \in \mathcal{M}_{n+1}(\mathbb{R})$.

a) Soit α le plus grand coefficient en valeur absolue dans la première colonne de A. Montrer qu'il existe r tel que

$$P_{1,r}\,A = \begin{pmatrix} \alpha & v \\ w & B \end{pmatrix}$$

avec $B \in \mathcal{M}_{n,n}(\mathbb{R})$, $v \in \mathcal{M}_{1,n}(\mathbb{R})$ et $w \in \mathcal{M}_{n,1}(\mathbb{R})$ avec $|w_i| \leq \alpha$.

b) En déduire que $P_{1,r}\,A$ peut se factoriser sous la forme

$$P_{1,r}\,A = \begin{pmatrix} 1 & \mathbf{0} \\ m & I_n \end{pmatrix} \begin{pmatrix} \alpha & v \\ \mathbf{0}^t & C \end{pmatrix},$$

où $\mathbf{0}$ est le vecteur nul de $\mathcal{M}_{1,n}(\mathbb{R})$. Exprimer m et C en fonction de α, w, v et B (on pourra distinguer le cas $\alpha = 0$ du cas $\alpha \neq 0$).

c) Soit $P_nC = L_nU_n$ avec $P_n \in \mathcal{M}_n$ une matrice de permutation, $L_n \in \mathcal{M}_n$, triangulaire inférieure à diagonale unité, et $U_n \in \mathcal{M}_n$, triangulaire supérieure. Montrer que

$$P_{1,r}\,A = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0}^t & (P_n)^t \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0} \\ P_nm & L_n \end{pmatrix} \begin{pmatrix} \alpha & v \\ \mathbf{0}^t & U_n \end{pmatrix}.$$

(On rappelle que $P^{-1} = P^t$ pour toute matrice de permutation P.)

d) En déduire l'expression de P, L et U telles que $PA = LU$, en fonction de r, α, v, m, P_n, L_n et U_n.

2.2. **Programmation de la factorisation avec permutations.**

1. Étant donnés une matrice `A` et deux entiers `i` et `j`, quel est le résultat de l'instruction Matlab : `A([i j],:)=A([j i],:)` ? Qu'obtient-on si on applique cette instruction à la matrice `A=eye(n)` ?

2. Étudier l'application `max` de Matlab. Comment obtient-on les valeurs r et α telles que $\alpha = |a_{r1}|$ est le plus grand coefficient en valeur absolue dans la première colonne de A ? Tester sur les matrices suivantes :

$$A = \begin{pmatrix} 2 & -1 & 0 & 1 \\ 1 & 1 & 2 & 1 \\ -4 & -1 & -1 & -2 \\ 1 & 1 & 3 & 1 \end{pmatrix} \text{ et } A = \begin{pmatrix} 2 & 1 & 0 & 4 \\ 4 & 1 & -2 & 8 \\ -4 & -2 & 3 & -7 \\ 0 & 3 & -12 & -1 \end{pmatrix}$$

3. Programmer la fonction `[L,U,P] = factolu_avec_perm(A)` en utilisant ce qui a été démontré en 2.1.

4. Tester votre fonction sur les matrices données ci-dessus ou sur d'autres matrices de votre choix. Comparer les résultats avec ceux obtenus d'une part avec `factolu`, d'autre part avec la fonction `lu` de Matlab.

5. Sur la base de la fonction `resoudre_systeme_lu` de la première partie créer une fonction `resoudre_systeme_plu` en utilisant la factorisation avec permutation de façon à la rendre utilisable pour n'importe quelle matrice inversible. Si la matrice n'est pas inversible (condition à déterminer sans utiliser la commande `det` !), la fonction devra afficher un message d'erreur. Faire des tests.