MTH1504

METHODES DIRECTES DE RESOLUTION DES SYSTEMES LINEAIRES

Table des matières

Le but est de résoudre le système linéaire dans $\mathbb{C}^n$: trouver $x \in \mathbb{C}^n$ tel que

$$Ax = b, \quad A \in \mathcal{M}_n(\mathbb{C}),$$

où $b \in \mathbb{C}^n$. On suppose que $A \in \mathcal{M}_n(\mathbb{C})$ est inversible.

1 Algorithme d'élimination de Gauss sans recherche de pivot, et interprétation matricielle

Le but est de se ramener à un système triangulaire

Première étape : élimination de l'inconnue x_1 des lignes 2 à n. On va chercher un système équivalent (ayant la même solution x), où x_1 n'apparaît que dans la ligne 1.
Première Hypothèse :

$$a_{11} \neq 0 \rightarrow \text{ on appelle premier pivot } \pi^{(1)} = a_{11}.$$

On peut alors former une combinaison linéaire de la ligne $i > 1$ avec la ligne 1 pour éliminer l'inconnue x_1 dans la ligne i : la ligne i devient

$$0x_1 + (a_{i2} - \frac{a_{i1}}{\pi^{(1)}} a_{12})x_2 + \cdots + (a_{in} - \frac{a_{i1}}{\pi^{(1)}} a_{1n})x_n = b_i - \frac{a_{i1}}{\pi^{(1)}} b_1,$$

ce qu'on peut réécrire

$$a_{i2}^{(2)} x_2 + \cdots + a_{in}^{(2)} x_n = b_i^{(2)}, \quad \forall i > 1$$

en posant

$$\begin{array}{rcl} a_{i2}^{(2)} & = & a_{i2} - \frac{a_{i1}}{\pi^{(1)}} a_{12}, \\ & \vdots & \\ a_{in}^{(2)} & = & a_{in} - \frac{a_{i1}}{\pi^{(1)}} a_{1n}, \\ b_i^{(2)} & = & b_i - \frac{a_{i1}}{\pi^{(1)}} b_1. \end{array}$$

Si on pose aussi, pour $1 \leq j \leq n$,

$$a_{1j}^{(2)} = a_{1j} \quad \forall 1 \leq j \leq n \quad \text{ et } \quad b_1^{(2)} = b_1,$$

on a obtenu le nouveau système équivalent

$$A^{(2)} x = b^{(2)},$$

avec

$$A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}$$

On voit que

$$A^{(2)} = M^{(1)} A \text{ et } b^{(2)} = M^{(1)} b, \quad \text{où} \quad M^{(1)} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -\frac{a_{n1}}{a_{11}} & 0 & \dots & 0 & 1 \end{pmatrix}$$

$k$ième étape : élimination de l'inconnue x_k des lignes $k+1$ à n On suppose qu'on a pu itérer le procédé ci-dessus $k-1$ fois, c'est à dire que les pivots apparus jusqu'à l'étape k sont non nuls :

$$\pi^{(i)} = a_{ii}^{(i)} \neq 0, \quad \text{pour } 1 \leq i \leq k-1.$$

On obtient alors le système équivalent

$$A^{(k)}x = b^{(k)}$$

où $A^{(k)}$ est de la forme

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & \dots & \dots & \dots & & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & \ddots & & & & & \vdots \\ \vdots & \ddots & a_{33}^{(k)} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \dots & \dots & 0 & a_{k-1k-1}^{(k)} & \dots & \dots & a_{k-1n}^{(k)} \\ \vdots & & & \vdots & 0 & & & \\ \vdots & & & \vdots & \vdots & & \widetilde{A^{(k)}} & \\ 0 & \dots & \dots & 0 & 0 & & & \end{pmatrix}$$

et $\widetilde{A^{(k)}}$ est une matrice carrée d'ordre $n-k+1$:

$$\widetilde{A^{(k)}} = \begin{pmatrix} a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & \vdots \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix}.$$

On a

$$A^{(k)} = M^{(k-1)} \cdots M^{(1)} A \text{ et } b^{(k)} = M^{(k-1)} \cdots M^{(1)} b$$

On va chercher un système équivalent (ayant la même solution x), où x_k n'apparaît pas dans les lignes $k+1$ à n.

k-ième Hypothèse :

$$a_{kk}^{(k)} \neq 0 \rightarrow \text{ on appelle } k\text{ième pivot } \pi^{(k)} = a_{kk}^{(k)}.$$

On introduit

$$M^{(k)} = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & & \vdots \\ \vdots & & \vdots & -\frac{a_{k+1\,k}^{(k)}}{\pi^{(k)}} & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -\frac{a_{n\,k}^{(k)}}{\pi^{(k)}} & 0 & \dots & 0 & 1 \end{pmatrix}$$

Remarquons que l'inverse de $M^{(k)}$ est $L^{(k)}$:

$$L^{(k)} = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & & \vdots \\ \vdots & & \vdots & \frac{a^{(k)}_{k+1\,k}}{\pi^{(k)}} & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{a^{(k)}_{n\,k}}{\pi^{(k)}} & 0 & \dots & 0 & 1 \end{pmatrix}$$

Soit

$$A^{(k+1)} = M^{(k)} A^{(k)} \quad \text{et} \quad b^{(k+1)} = M^{(k)} b^{(k)},$$

alors

$$A^{(k+1)} x = b^{(k+1)},$$

et

$$A^{(k+1)} = \begin{pmatrix} a^{(k+1)}_{11} & a^{(k+1)}_{12} & \dots & \dots & \dots & \dots & & a^{(k+1)}_{1n} \\ 0 & a^{(k+1)}_{22} & \ddots & & & & & \vdots \\ \vdots & \ddots & a^{(k+1)}_{33} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \dots & \dots & 0 & a^{(k+1)}_{kk} & \dots & \dots & a^{(k+1)}_{kn} \\ \vdots & & & \vdots & 0 & & & \\ \vdots & & & \vdots & \vdots & & \widetilde{A^{(k+1)}} & \\ 0 & \dots & \dots & 0 & 0 & & & \end{pmatrix},$$

où $\widetilde{A^{(k+1)}}$ est une matrice carrée d'ordre $n-k$:

$$\widetilde{A^{(k+1)}} = \begin{pmatrix} a^{(k+1)}_{k+1k+1} & \dots & a^{(k+1)}_{k+1n} \\ \vdots & & \vdots \\ a^{(k+1)}_{nk+1} & \dots & a^{(k+1)}_{nn} \end{pmatrix}.$$

On a

$$A^{(k+1)} = M^{(k)} \cdots M^{(1)} A \text{ et } b^{(k+1)} = M^{(k)} \cdots M^{(1)} b$$

Après $n-1$ étapes Si on itère $n-1$ fois, et si les pivots apparus sont tous non nuls :
Hypothèses des $n-1$ étapes :

$$\pi^{(i)} = a^{(i)}_{ii} \neq 0, \quad \text{pour } 1 \leq i \leq n-1.$$

on arrive au système **triangulaire équivalent**

$$A^{(n)} x = b^{(n)},$$

avec

$$A^{(n)} = \begin{pmatrix} \pi^{(1)} & * & & \dots & * \\ 0 & \pi^{(2)} & * & \dots & * \\ 0 & 0 & \ddots & & \\ 0 & \dots & 0 & \ddots & * \\ 0 & & \dots & 0 & \pi^{(n)} \end{pmatrix}$$

qui est inversible si de plus
Hypothèse :

$$\pi^{(n)} \neq 0.$$

Notons U la matrice triangulaire supérieure

$$U = A^{(n)} = M^{(n-1)} \cdots M^{(1)} A,$$

et L la matrice triangulaire inférieure avec des 1 sur la diagonale

$$L = L^{(1)} \dots L^{(n-1)}$$

Alors on a

$$A = LU.$$

On dit qu'on a effectué une factorisation de Gauss ou factorisation LU de A.

Remarque 1.1 *Il est aussi possible avec le même algorithme d'obtenir la factorisation LU d'une matrice de $\mathcal{M}_m(\mathbb{C})$, avec $m \geq n$, si les pivots qui apparaissent sont non nuls.*

1.1 Coût de la méthode d'élimination de Gauss.

On estime le coût de l'élimination de x_k. Rappelons qu'à l'étape $k-1$, on obtient la matrice $A^{(k)}$

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & \dots & \dots & \dots & & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & \ddots & & & & & \vdots \\ \vdots & \ddots & a_{33}^{(k)} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \dots & \dots & 0 & a_{k-1k-1}^{(k)} & \dots & \dots & a_{k-1n}^{(k)} \\ \vdots & & & \vdots & 0 & & & \\ \vdots & & & \vdots & \vdots & & \widetilde{A^{(k)}} & \\ 0 & \dots & \dots & 0 & 0 & & & \end{pmatrix}$$

C'est sur le bloc $\widetilde{A^{(k)}}$ de $A^{(k)}$ qu'il faut travailler.
Construction de $M^{(k)}$: $n-k$ divisions par le pivot.

$$M^{(k)} = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & & \vdots \\ \vdots & & \vdots & -\frac{a^{(k)}_{k+1\,k}}{\pi^{(k)}} & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -\frac{a^{(k)}_{n\,k}}{\pi^{(k)}} & 0 & \dots & 0 & 1 \end{pmatrix}$$

Multiplication de $A^{(k)}$ par $M^{(k)}$: $(n-k)^2$ adds+mults.
Total

$$\sum_{k=1}^{n}(n-k)^2 = \frac{1}{3}n(n-1)(n-\frac{1}{2}) \sim \frac{n^3}{3} \text{ additions+multiplications.}$$
$$\sum_{k=1}^{n}(n-k) = \frac{1}{2}n(n-1) \text{ divisions.}$$

1.2 Utilisations de la factorisation LU

Calcul de déterminant Une utilisation de la factorisation LU est le calcul du déterminant de A : en effet, si A admet une factorisation LU, on a

$$\det(A) = \det(U) = \text{ produit des pivots}$$

car $\det(L) = 1$. La factorisation LU permet donc de calculer le déterminant de A avec une complexité de l'ordre de $\frac{n^3}{3}$ adds + mults, plutôt que $n!$ avec la formule du déterminant !

Remarque 1.2 *Calculer le déterminant d'une matrice d'ordre 100 avec la formule du déterminant prendrait un temps supérieur à l'age de l'univers sur le calculateur le plus puissant disponible aujourd'hui.*

Résolution de systèmes linéaires Si on a plusieurs systèmes (par exemple p) à résoudre avec la même matrice A, il est intéressant de calculer une fois pour toutes et stocker la factorisation LU de $A = LU$, puis de résoudre les systèmes par descentes-remontées plutôt que d'utiliser l'algorithme d'élimination pour chaque système. On résout le système $Ax = b$ en résolvant d'abord

$$Ly = b,$$

soit une descente, puis

$$Ux = y,$$

soit une remontée. Si on a p systèmes à résoudre, la complexité est de l'ordre de $\frac{n^3}{3} + pn^2$ adds+mults plutôt que $p\frac{n^3}{3}$.

Remarque 1.3 *Comme à la remarque précédente, calculer la solution du système linéaire en appliquant les formules de Cramer nécessiterait un temps inimaginable dès que n vaut quelques dizaines.*

1.3 Algorithme

Voici un algorithme réalisant la factorisation LU d'une matrice A et stockant cette factorisation dans la place mémoire occupée par A : (la matrice A est perdue, on dit qu'on écrase A).

```
for j=1:n-1
   for i=j+1:n
      A(i,j)=A(i,j)/A(j,j);   //construction de la j eme colonne de L

   for i=j+1:n
      for k=j+1:n
          A(i,k)=A(i,k)-A(i,j)*A(j,k);
                    //actualisation des n-j-1 derni{\`e}res lignes de A
```

Remarque 1.4 *La diagonale de L (qui ne contient que des 1) n'est pas stockée.*

Le programme pour la descente-remontée pour calculer la solution de $Ax = b$ est alors, en écrasant b :

```
for i=1:n  //descente pour calculer y tq Ly=b
      sum=0;
      for k=1:i-1
        sum= A(i,k)*b(k);
      b(i) = b(i) - sum ;

for i=n:-1:1  //remontee pour calculer x tq Ux=y
      sum=0;
      for k=i+1:n
        sum= A(i,k)*b(k);
      b(i) = (b(i)-sum)/A(i,i) ;
```

1.4 Condition nécessaire et suffisante d'existence d'une factorisation LU, unicité

Théorème 1.1 *Soit A une matrice de $\mathcal{M}_n(\mathbb{C})$. Pour $1 \leq p \leq n$, on note A_p le bloc*

$$A_p = \begin{pmatrix} a_{1\,1} & \dots & \dots & a_{1\,p} \\ a_{2\,1} & \dots & \dots & a_{2\,p} \\ \vdots & & & \vdots \\ a_{p\,1} & \dots & \dots & a_{p\,p} \end{pmatrix}$$

La matrice A admet une factorisation

$$A = LU$$

où L est triangulaire inférieure avec des 1 sur la diagonale et U est triangulaire supérieure et inversible si et seulement si tous les blocs A_p, $1 \leq p \leq n$, sont inversibles. De plus, cette factorisation est unique. De plus si A est réelle, alors L et U le sont aussi.

Démonstration. Si A admet une factorisation LU avec L et U inversibles, alors les blocs A_p sont bien inversibles.La démonstration de la réciproque se fait par récurrence.

Nous nous contentons ici de démontrer l'unicité : soient deux factorisations LU de A : $A = LU = L'U'$ où L sont triangulaires inférieures avec des 1 sur la diagonale, et U sont triangulaires supérieures et inversibles. On a donc l'identité

$$L^{-1}L' = U'U^{-1}$$

Mais $L^{-1}L'$ est triangulaire inférieure avec des 1 sur la diagonale et $U'U^{-1}$ est triangulaire supérieure. On a donc

$$L^{-1}L' = U'U^{-1} = Id \Rightarrow \left| \begin{array}{l} L = L' \\ U = U' \end{array} \right.$$

■

1.5 Cas de matrices bandes

Définition 1.1 *Soit A une matrice de $C^{n\times n}$. On appelle largeur de bande de A le plus petit entier $n_b \leq n$ tel que pour tout i, $1 \leq i \leq n$*

$$\begin{array}{ll} a_{i,j} = 0 & \text{si } n \geq j > i + n_b, \\ a_{j,i} = 0 & \text{si } 0 < j < i - n_b, \end{array}$$

Les coefficients non nuls de la matrice A sont contenus dans une bande de largeur $2n_b + 1$, centrée sur la diagonale.

Dans le cas où la largeur de bande n_b de A est très inférieure à n, on parle de matrice bande. Si A est une matrice bande, sa factorisation LU demande moins d'opérations et de place mémoire :

Proposition 1.1 *Si A, (de largeur de bande n_b) admet une factorisation LU, alors les largeurs de bandes de L et de U sont inférieures à n_b et la complexité nécessaire à effectuer la factorisation LU est inférieure à*

$$\sum_{k=1}^{n} (n_b)^2 = n(n_b)^2 \ \textit{additions+multiplications.}$$
$$\sum_{k=1}^{n} n_b = nn_b \ \textit{divisions.}$$

Exercice. Écrire l'algorithme de factorisation LU pour une matrice bande de largeur de bande n_b.

2 Méthode de Cholesky

Dans le cas où A est hermitienne et définie positive, on peut toujours effectuer la factorisation décrite ci-dessus. De plus, on peut trouver une factorisation du type $A = LL^*$ moins gourmande en place mémoire.

2.1 Existence de la factorisation de Cholesky

Théorème 2.1 *Si A est hermitienne et définie positive, alors il existe une unique matrice L triangulaire inférieure et inversible, avec des coefficients réels positifs sur la diagonale, telle que*

$$A = LL^*$$

Cette factorisation porte le nom de factorisation de Cholesky. Si A est réelle symétrique définie positive, L est réelle.

Démonstration.

On part de la factorisation LU de A. Il existe une unique factorisation $A = L'U'$ où la matrice L' est triangulaire inférieure avec des 1 sur la diagonale et U' est triangulaire supérieure inversible. Appelons D la diagonale de U'. Comme pour tout p, $1 \leq p \leq n$, $\det(D_p) = \det(A_p) > 0$, tous les coefficients de D sont strictement positifs. Notons $U = D^{-\frac{1}{2}}U'$ et $L = L'D^{\frac{1}{2}}$. On a $A = LU$. Montrons que $U = L^*$. Comme A est hermitienne, $U^*L^* = A^* = A = LU$. On a donc $L^{-1}U^* = U(L^*)^{-1}$. Mais $L^{-1}U^*$ est triangulaire inférieure, avec des 1 sur la diagonale tandis que $U(L^*)^{-1}$ est triangulaire supérieure, avec des 1 sur la diagonale. Donc $U = L^*$, et on a $A = LL^*$. Pour l'unicité, on procède comme dans la démonstation de l'unicité pour la factorisation LU. ■

Remarque 2.1 *Il est important de comprendre que les matrices L dans les factorisation LU et de Cholesky sont différentes.*

2.2 Algorithme

Considérons la k ième étape de la méthode de Cholesky : A ce point, on suppose que l'on a déjà construit la matrice $L^{(k-1)}$ et $A^{(k)}$ telles que

$$L^{(k-1)} = \begin{pmatrix} \sqrt{\pi_1} & 0 & & & \dots & & 0 \\ * & \sqrt{\pi_2} & 0 & & \dots & & 0 \\ & & \ddots & & & & \\ * & \dots & * & \sqrt{\pi_{k-1}} & 0 & \dots & 0 \\ * & \dots & * & * & 1 & \dots & 0 \\ * & \dots & * & \vdots & 0 & \ddots & 0 \\ * & \dots & * & * & 0 & \dots & 1 \end{pmatrix}$$

et

$$A^{(k)} = \begin{pmatrix} * & * & & & \dots & & * \\ 0 & * & & & \dots & & * \\ 0 & 0 & * & & \dots & & * \\ & & & \ddots & & & \\ 0 & 0 & \dots & 0 & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & & & \vdots & \ddots & \\ 0 & 0 & \dots & 0 & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix}$$

et $\pi_k = a_{k,k}^{(k)}$.

Remarque 2.2 *Le bloc carré $k \to n$ de $A^{(k)}$ est hermitien et défini positif.*

Pour effectuer la $k$ième étape, on construit d'abord $L^{(k)}$ en remplaçant la $k$ième colonne de $L^{(k-1)}$ par le vecteur $l_k = (0, \ldots, 0, \sqrt{\pi_k}, \frac{a^{(k)}_{k+1\,k}}{\sqrt{\pi_k}}, \ldots \frac{a^{(k)}_{n\,k}}{\sqrt{\pi_k}})^T$.

Observation 2.1 *D'après la Remarque 2.2,* $(0, \ldots, 0, a^{(k)}_{kk}, \ldots, a^{(k)}_{kn}) = \sqrt{\pi_k} l^*_k$.

On construit maintenant $A^{(k+1)}$ en effectuant les éliminations de Gauss et d'après l'Observation 2.1, on a

$$A^{(k+1)} = A^{(k)} - l_k l^*_k. \tag{1}$$

On voit en fait que l'on peut ne construire que la partie triangulaire inférieure de la matrice $A^{(k+1)}$, ce qui économise la moitié des opérations. On aboutit à l'algorithme suivant dans le cas d'une matrice réelle symétrique définie positive :

```
for k = 1:n
  A(k,k)=sqrt(A(k,k));
  for i = k+1:n
    A(i,k)= A(i,k) / A(k,k);
  for i = k + 1:n
    for j = k+1:i
      A(i,j)=A(i,j) - A(i,k)*A(j,k);
```

2.3 Complexité de la factorisation de Cholesky

$\sim \frac{n^3}{6}$ additions+multiplications.
$\frac{1}{2}n(n-1)$ divisions.
n évaluations de racines carrées.

Exercice Montrer l'évaluation précédente.

Remarque 2.3 *L'intérêt de la factorisation de Cholesky est qu'elle demande une place mémoire deux fois inférieure à celles de la factorisation LU ci-dessus, car on ne doit stocker que L et que sa complexité arithmétique est deux fois moindre.*

3 Programmes Scilab pour les factorisation LU et de Cholesky

3.1 Factorisation LU

Construction de la factorisation LU

```
//LU factorization
function [A]= LUfact(A)
[m,n]=size(A);
for i=1:n,
  if (A(i,i)==0) then
    print(%io(2)," zero pivot")
  else
    A(i+1:m,i)=A(i+1:m,i)/A(i,i);
    for j=i+1:n,
      A(i+1:m,j)=A(i+1:m,j)-A(i+1:m,i)*A(i,j);
```

```
    end ;
  end ;
end;
```

Descente-Remontée pour la factorisation LU

```
// upward sweep : solves an upper triangular system
function  [y]=up_sweep_LU(A,x)
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
else
  y=x;
  y(n)=y(n)/A(n,n);
  for i=n-1:-1:1,
    y(i)=(y(i)-A(i,i+1:n)*y(i+1:n))/A(i,i);
  end;
end;

// downward sweep : solves a lower triangular system with ones on the diagonal
function  [y]=down_sweep_LU(A,x)
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
else
  y=x;
  for i=2:n
    y(i)=y(i)-A(i,1:i-1)*y(1:i-1);
  end;
end;

//assumes A contains the LU-factorization
function [y]=SolveLU(A,x)
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
else
  y=down_sweep_LU(A,x);
  y=up_sweep_LU(A,y);
end;
```

3.2 Factorisation de Cholesky

Construction de la factorisation de Cholesky

```
//Cholesky factorization
//the matrix is stored in the lower part
function [A]= Cholesky_fact(A)
[m,n]=size(A);
```

```
for i=1:n,
  if (A(i,i)<=0) then
    print(%io(2)," non positive pivot")
  else
    A(i,i)=sqrt(A(i,i));
    A(i+1:m,i)=A(i+1:m,i)/A(i,i);
    for j=i+1:n,
      A(j:m,j)=A(j:m,j)-A(j:m,i)*A(j,i);
    end ;
  end ;
end ;
```

Descente-Remontée pour une factorisation de Cholesky

```
// up sweep : solves the upper triangular system
function  [y]=up_sweep_Cholesky(A,x)
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
else
  y=x;
  y(n)=y(n)/A(n,n);
  for i=n-1:-1:1,
    y(i)=(y(i)-(A(i+1:n,i))'*y(i+1:n))/A(i,i);
  end;
end;

// down sweep : solves the lower triangular system
function  [y]=down_sweep_Cholesky(A,x)
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
else
  y=x;
   y(1)=y(1)/A(1,1);
  for i=2:n
    y(i)=y(i)-A(i,1:i-1)*y(1:i-1);
    y(i)=y(i)/A(i,i)
  end;
end;

function [y]=SolveCholesky(A,x)
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
else
  y=down_sweep_Cholesky(A,x);
  y=up_sweep_Cholesky(A,y);
end;
```

4 Méthode de Gauss avec pivot partiel

La méthode de Gauss sans pivotage peut être bloquée si on tombe sur un pivot nul. Pour pallier cet inconvénient, on introduit une méthode qui permet d'éviter ce blocage en échangeant les lignes du système considéré. On obtient alors une méthode qui fonctionne pour toute matrice inversible : la méthode de Gauss avec pivot partiel.

Théorème 4.1 *Soit $A \in \mathcal{M}_n(\mathbb{C})$, inversible. Alors il existe*
- *une matrice de permutation P,*
- *une matrice triangulaire inférieure L, avec des 1 sur la diagonale,*
- *une matrice triangulaire supérieure U, et inversible,*

telles que

$$PA = LU$$

Démonstration. Par récurrence sur les éliminations de Gauss : à la $k$ième étape, supposons que l'on ait prouvé l'égalité :

$$Q^{(k-1)}A = L^{(1)} \dots L^{(k-1)}A^{(k)},$$

où $Q^{(k-1)}$ est une matrice de permutation et $\forall i < k$,

$$L^{(i)} = \overbrace{\begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & & \vdots \\ \vdots & & \vdots & * & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & * & 0 & \dots & 0 & 1 \end{pmatrix}}^{1 \quad \dots \quad i \qquad\qquad n}$$

et

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & \dots & \dots & \dots & \dots & & a_{1n}^{(k)} \\ 0 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & & & \vdots \\ \vdots & & & 0 & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & \dots & \dots & 0 & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix}$$

On effectue la $k$ième élimination de Gauss avec pivotage partiel : avant d'éliminer l'inconnue x_k, on échange les lignes k et $r_k > k$. On peut choisir $r_k \geq k$ tel que

$$|a_{r_k\,k}^{(k)}| = \max_{k \leq j \leq n} |a_{j\,k}^{(k)}|.$$

L'inversibilité de A assure que $a^{(k)}_{r_k\,k} \neq 0$. On peut donc s'en servir de pivot pour la prochaine élimination de Gauss.
Le pivotage puis l'élimination de Gauss s'écrivent matriciellement

$$P^{(kr_k)}A^{(k)} = L^{(k)}A^{(k+1)},$$

avec $P^{(kr_k)}$ matrice de permutation élémentaire entre les lignes k et r, $L^{(k)}$ triangulaire inférieure L, avec des 1 sur la diagonale, et

$$A^{(k+1)} = \begin{pmatrix} a^{(k+1)}_{11} & \dots & \dots & \dots & \dots & & a^{(k+1)}_{1n} \\ 0 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & & & \vdots \\ \vdots & & & 0 & a^{(k+1)}_{k+1k+1} & \dots & a^{(k+1)}_{k+1n} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & \dots & \dots & 0 & a^{(k+1)}_{nk+1} & \dots & a^{(k+1)}_{nn} \end{pmatrix}.$$

Comme

$$(P^{(kr_k)})^{-1} = P^{(kr_k)},$$
$$A^{(k)} = P^{(kr_k)}L^{(k)}A^{(k+1)},$$

ce qui implique

$$Q^{(k-1)}A = L^{(1)} \dots L^{(k-1)}P^{(kr_k)}L^{(k)}A^{(k+1)}.$$

A ce point, on utilise le lemme :

Lemme 4.1 *Soit $P^{(kr_k)}$ la matrice de permutation élémentaire entre les lignes k et $r_k > k$, alors $\forall i < k$, si $L^{(i)}$ s'écrit*

$$L^{(i)} = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & & \vdots \\ \vdots & & \vdots & & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & C_i & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & & 0 & \dots & 0 & 1 \end{pmatrix} \quad et \quad C_i = \begin{pmatrix} \vdots \\ \alpha \\ \vdots \\ \beta \\ \vdots \end{pmatrix}$$

alors

$$L^{(i)}P^{(kr_k)} = P^{(kr_k)}L'^{(i)},$$

où

$$L'^{(i)} = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & & 0 & 1 & \ddots & & & \vdots \\ \vdots & & \vdots & & \ddots & \ddots & & \vdots \\ \vdots & & \vdots & C'_i & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & & 0 & \dots & 0 & 1 \end{pmatrix} \quad et \quad C'_i = \begin{pmatrix} \vdots \\ \beta \\ \vdots \\ \alpha \\ \vdots \end{pmatrix}.$$

Donc,

$$\begin{aligned} Q^{(k-1)}A &= L^{(1)} \dots L^{(k-2)} P^{(kr_k)} L'^{(k-1)} L^{(k)} A^{(k+1)} \\ &= P^{(kr_k)} L'^{(1)} \dots L'^{(k-1)} L^{(k)} A^{(k+1)}. \end{aligned}$$

Par suite,

$$P^{(kr_k)} Q^{(k-1)} A = L'^{(1)} \dots L'^{(k-1)} L^{(k)} A^{(k+1)},$$

où

- $Q^{(k)} = P^{(kr_k)} Q^{(k-1)}$ est une matrice de permutation,
- $L'^{(1)} \dots L'^{(k-1)} L^{(k)}$ est triangulaire inférieure avec des 1 sur la diagonale.

On a montré que la propriété est vraie pour $k+1$. ■

Remarque 4.1 *Dans la méthode de Gauss avec pivotage partiel, on ne doit pas stocker la matrice P mais seulement les r_k. Le coût mémoire et la complexité sont du même ordre que ceux de la factorisation LU. La résolution du système linéaire $Ax = b$ connaissant P L et U se fait de la manière suivante :*

$$\begin{aligned} y &= Pb \\ Lz &= y \\ Ux &= z \end{aligned}$$

Remarque 4.2 *La méthode de Gauss avec pivotage partiel ne conserve pas l'aspect bande : si A est une matrice bande dont la largeur de bande est n_b, les matrices L et U n'ont pas en général cete propriété. La complexité et le coût mémoire de la méthode de Gauss avec pivotage partiel deviennent alors largement supérieurs à ceux de la factorisation LU.*

Remarque 4.3 *La méthode du pivot total consiste à échanger non seulement les lignes mais aussi les colonnes de manière à choisir comme pivot le plus grand coefficient en module du bloc carré restant à factoriser. Cette méthode conduit à l'existence de deux matrices de permutations P et Q telles que $PAQ = LU$.*

Influence du pivotage sur la précision Le pivotage partiel ne sert pas seulement à garantir l'obtention d'une factorisation, il garantit aussi une bien meilleure stabilité que la factorisation LU, pour des matrices A mal conditionnées : prenons le système $Ax = b$ où

$$A = \begin{pmatrix} 10^{-9} & 1 \\ 1 & 1 \end{pmatrix} \quad \text{et} \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

dont la solution est $x = (\frac{1}{1-10^{-9}}, \frac{1-2\,10^{-9}}{1-10^{-9}})^T \sim (1,1)^T$. Supposons que la machine utilisée pour résoudre ce système ne garde que 8 chiffres significatifs : la factorisation LU calculée par la machine est

$$U = \begin{pmatrix} 10^{-9} & 1 \\ 0 & -10^9 \end{pmatrix} \quad \text{et} \quad L = \begin{pmatrix} 1 & 0 \\ 10^9 & 1 \end{pmatrix}$$

En utilisant cette factorisation, la machine retourne $x = (0,1)^T$!
Si on utilise le pivotage partiel, on obtient :

$$U = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{et} \quad L = \begin{pmatrix} 1 & 0 \\ 10^{-9} & 1 \end{pmatrix}$$

La machine retourne $x = (1,1)^T$.

5 Factorisations QR

5.1 Les réflexions de Householder

Définition 5.1 *Soit v un vecteur non nul de $\mathbb{C}^n$. On appelle réflexion de Householder ou matrice de Householder relative au vecteur v la matrice*

$$H_v = I - 2\frac{vv^*}{v^*v}. \tag{2}$$

Les réflexions de Householder ont les propriétés suivantes :

1. H_v est une matrice hermitienne.
2. H_v est une matrice unitaire.
3. $H_v - I$ est une matrice de rang un.
4. $H_{\lambda v} = H_v$, pour tout $\lambda \neq 0$.

Proposition 5.1 *Soit e un vecteur unitaire de $\mathbb{C}^n$ et x un vecteur non nul de $\mathbb{C}^n$. Il existe un vecteur non nul $v \in \mathbb{C}^n$ tel que $H_v x$ soit colinéaire à e.*

Démonstration. On cherche v sous la forme $v = x + \lambda e$. Dans le cas réel, ($x \in \mathbb{R}^n$ et $e \in \mathbb{R}^n$), les vecteurs $v = x \pm \|x\|_2 e$ (au moins l'un des deux est non nul) sont les seuls vecteurs de cette forme ayant la propriété demandée et on a $H_v x = \mp\|x\|_2 e$. Dans le cas général, on trouve aussi deux vecteurs v sous cette forme, et au moins l'un d'entre eux est non nul. ■

Remarque 5.1 *Si x est presque colinéaire à e, on a intérêt en pratique à choisir le vecteur v pour que v^*v, qui est au dénominateur de (2), ne soit pas petit. En effet, en précision finie, il faut éviter les divisions par des nombres petits. En particulier si $v \in \mathbb{R}^n$, on préférera le vecteur $v^+ = x + \text{signe}(x^*e)\|x\|_2 e$ au vecteur $v^- = x - \text{signe}(x^*e)\|x\|_2 e$, car $\|v^-\|_2$ est petit.*

5.2 Factorisations QR à l'aide des réflexions de Householder

Théorème 5.1 *Soit A une matrice de $\mathcal{M}_{m,n}(\mathbb{C})$ avec $m \geq n$. Il existe une matrice unitaire $Q \in \mathcal{M}_{m,m}(\mathbb{C})$et une matrice triangulaire supérieure $R \in \mathcal{M}_{m,n}(\mathbb{C})$, telles que*

$$A = QR.$$

Démonstration. On démontre ce résultat par récurrence : supposons qu'à l'aide de deux réflexions de Householder H_1 et H_2, on ait obtenu

$$H_2H_1A = \begin{pmatrix} * & * & * & \dots & * \\ 0 & * & * & \dots & * \\ 0 & 0 & * & \dots & * \\ \vdots & \vdots & * & \dots & * \\ 0 & 0 & * & \dots & * \end{pmatrix}$$

On appelle x le vecteur de $\mathbb{C}^{m-2}$ obtenu en prenant les $m-2$ derniers coefficients de la troisième colonne de H_2H_1A. Si x est non nul, on sait trouver $v_3 \in \mathbb{C}^{m-2}$, tel que

$$H_{v_3}\, x \text{ soit colinéaire à } \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

On prend alors H_3

$$H_3 = \begin{pmatrix} I_2 & 0 \\ 0 & H_{v_3} \end{pmatrix}.$$

Si $x = 0$ on prend $H_3 = I$. Dans les deux cas H_3 est hermitienne et unitaire. On a

$$H_3H_2H_1A = \begin{pmatrix} * & * & * & & \dots & * \\ 0 & * & * & & \dots & * \\ 0 & 0 & * & & \dots & * \\ 0 & 0 & 0 & * & \dots & * \\ \vdots & \vdots & \vdots & * & \dots & * \\ 0 & 0 & 0 & * & \dots & * \end{pmatrix}$$

Comme $m \geq n$, on peut itérer ce procédé et construire n matrices H_i hermitiennes et unitaires, telles que $H_n \dots H_1A$ soit une matrice triangulaire supérieure R. On note Q la matrice unitaire (mais pas forcément hermitienne) $Q = H_1 \dots H_n$, on a

$$A = QR.$$

■

Les éléments fondamentaux de l'algorithme de la factorisation QR à l'aide de réflexions de Householder sont

1. Le choix des vecteurs v_i, de manière à ce que $\|v_i\|$ ne soit pas petit, cf. Remarque 5.1.
2. Le produit à gauche d'une matrice M par la matrice H_m : H_mM, doit être programmé intelligemment : on ne doit surtout pas stocker la matrice H_m. Il suffit de garder en mémoire le vecteur v_m, et d'opérer les réflexions de Householder à chaque colonne de M.
3. On ne calcule donc pas Q en général, mais on garde en mémoire les vecteurs $v_1, \dots, v_n$.
4. On peut écraser la matrice A que l'on factorise en stockant les vecteurs v_i dans la partie triangulaire inférieure stricte, et la matrice R dans la partie triangulaire supérieure à condition de normaliser les vecteurs v_i de manière à ce que leur premier coefficient soit 1 : ainsi on n'a pas besoin de stocker le premier coefficient des vecteurs v_i.

5. Si la matrice A est réelle, alors la matrice Q l'est aussi.

La factorisation QR a de nombreuses propriétés intéressantes :

1. Résoudre le système $Qy = b$ est facile car

$$Q^{-1} = Q^* = H_n \dots H_1.$$

   La complexité de la résolution du système $Qy = b$ est donc $3\sum_i^n (m-i)$ adds+mults. Si $m = n$, la complexité est de l'ordre de $\frac{3}{2}n^2$. Si la matrice A est carrée d'ordre n inversible, pour résoudre $Ax = b$, on résout d'abord $Qy = b$ comme ci-dessus, puis $Rx = y$ par une remontée. La complexité totale est de l'ordre de $2n^2$.

2. Si $A \in \mathcal{M}_n(\mathbb{C})$ est inversible, alors

$$\mathrm{cond}_2(A) = \mathrm{cond}_2(R),$$

   car Q est unitaire.

3. la méthode de Householder permet de calculer $|\det(A)|$. En effet,

$$|\det(R)| = |\text{produit des coefficients diagonaux de } R| = |\det(A)|.$$

4. Si $A \in \mathcal{M}_{m,n}(\mathbb{C})$, $m \geq n$, est factorisée sous la forme $A = QR$, alors résoudre une système de la forme $A^*Ax = b$ est facile si $\mathrm{rang}(A) = n$, car $A^*A = R^*R$ et $\mathrm{rang}(R) = n$, et on peut résoudre $A^*Ax = b$ par une descente(de R^*)-remontée (de R). La factorisation QR d'une matrice se prête donc bien aux problèmes de moindres carrés (voir §6).

Théorème 5.2 *Soit $A \in \mathcal{M}_m(\mathbb{C})$. On peut trouver une factorisation QR telle que tous les coefficients diagonaux de R sont réels positifs. Si A est inversible, cette factorisation est unique.*

Démonstration. On part de la factorisation QR de A obtenue par la méthode de Householder ci dessus : $A = Q'R'$ où la matrice Q' est unitaire et R' est triangulaire supérieure. Appelons D la matrice diagonale $D = diag(d_1, \dots d_m)$ où

$$d_i = \begin{cases} \frac{\overline{r_{i\,i}}}{|r_{i\,i}|} & \text{si } r_{i\,i} \neq 0, \\ 1 & \text{si } r_{i\,i} = 0, \end{cases}$$

La matrice D est clairement unitaire. Posons $Q = Q'D^{-1}$ et $R = DR'$. Ces matrices ont les propriétés désirées.
Supposons que A soit inversible et soient deux factorisation QR de A,

$$A = Q_1R_1 = Q_2R_2$$

telles que tous les coefficients diagonaux de R_1 et R_2 sont réels positifs. Ils sont strictement positifs car A est inversible. On a alors $Q_2^*Q_1 = R_2R_1^{-1}$. Mais $Q_2^*Q_1$ est unitaire tandis que $R_2R_1^{-1}$ est triangulaire supérieure avec des coefficients diagonaux réels positifs strictement. On peut vérifier que l'Identité est la seule matrice unitaire et triangulaire supérieure avec des coefficients diagonaux réels positifs. Donc $Q_2 = Q_1$ et $R_2 = R_1$. ■

5.3 Algorithme de factorisation QR avec des symétries de Householder

Recherche du Vecteur pour la Réflexion de Householder

```
// householder reflexion : find Householder vector
//normalized so that v(1)=1
function [v]= householder_vector(x)
delta=1;
if (x(1)<0) then
  delta=-1;
end;
v=x;
beta=v(1)+delta* sqrt(x'*x);
v=v/beta;
v(1)=1;
```

Réflexion de Householder

```
// performs Householder reflexion of vector v on a vector x
function [y]= householder_reflexion(v,x)
beta=2*(v'*x)/(v'*v);
y=x-beta*v;
```

Factorisation QR

```
//performs QR factorisation of a matrix
//the R part is stored in the upper part of A
//the Q part (normalized so that the upper line is one)
//is stored in the strict lower part of A
function[A]=QRfactorization(A)
[m,n]=size(A);
for i=1:n,
  v=householder_vector(A(i:m,i))
  delta=2/(v'*v);
  A(i,i)=A(i,i)-delta* (v'* A(i:m,i));
  for j=i+1:n,
    c=v'*A(i:m,j);
    A(i:m,j)=A(i:m,j)-c*delta*v;
  end ;
  if (i<m) then
    A(i+1:m,i)=v(2:m-i+1);
  end ;
end;
```

Résolution du système

```
function  [y]=up_sweep(A,x) //solves Ry=x
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
```

```
else
  y=x;
  y(n)=y(n)/A(n,n);
  for i=n-1:-1:1,
    y(i)=(y(i)-A(i,i+1:n)*y(i+1:n))/A(i,i);
  end;
end;

//assumes A contains a QR-factorization of A
function [y]=SolveQR(A,x)
[m,n]=size(A);
if (m~=n) then
  print(%io(2), "error, not a square matrix");
else
  y=x;
  for i=1:n-1,
    y(i:n)= householder_reflexion( [1;A(i+1:n,i)] , y(i:n));
  end ;
  y(n)=-y(n);
  y=up_sweep(A,y);
//  print(%io(2), y);
end;
```

6 Problèmes aux moindres carrés

On va travailler avec des systèmes réels mais toute la suite serait valable avec des systèmes dans $\mathbb{C}^n$, à condition de changer A^T en A^*.
On considère le problème de trouver $x \in \mathbb{R}^n$ tel que $Ax = b$ où $A \in \mathcal{M}_{m,n}(\mathbb{R})$ et où $b \in \mathbb{R}^m$, avec $m \geq n$. Il y a plus d'équations que d'inconnues, on dit que le système est sur-déterminé. Un système sur-déterminé n'a généralement pas de solution, sauf si $b \in \mathrm{Im}(A)$.
L'idée est alors de chercher à minimiser $\|Ax - b\|$ plutôt que de chercher à résoudre exactement le système. La solution du problème si elle existe dépend de la norme choisie. La méthode des moindres carrés consiste à minimiser $\|Ax - b\|_2$ où $\|f\|_2^2 = \sum_{i=1}^m f_i^2$. On choisit cette norme car la fonction $J(x) = \|Ax - b\|_2^2$ est convexe et différentiable dans $\mathbb{R}^n$, et son gradient est $gradJ(x) = 2(A^TAx - A^Tb)$. On verra que minimiser J revient à résoudre le système linéaire $A^TAx = A^Tb$, qui a toujours une solution, et dont la solution est unique si $\mathrm{rang}(A) = n$.

Lemme 6.1 *Soit $A \in \mathcal{M}_{m,n}(\mathbb{R})$ avec $m \geq n$: le problème de trouver $x \in \mathbb{R}^n$ minimisant la fonction $J : \mathbb{R}^n \to \mathbb{R}$, $J(y) = \|Ay - b\|_2^2$ a au moins une solution. Le problème de trouver $x \in \mathbb{R}^n$ minimisant J est équivalent à trouver x tel que*

$$A^TAx - A^Tb = 0. \tag{3}$$

L'équation (3) est appelée équation normale.

Démonstration. Le problème de minimiser J revient à trouver $z \in \mathrm{Im}(A)$ minimisant la distance $\|z - b\|_2$. On sait que $\mathrm{Im}(A)^\perp = \ker(A^T)$, et donc que $\mathrm{Im}(A) \oplus \ker(A^T) = \mathbb{R}^n$. Prendre pour z la projection de b sur $\mathrm{Im}(A)$ parallèlement à $\ker(A^T)$ répond donc à la question, et après on prend x tel que $Ax = z$ (qui existe car $z \in \mathrm{Im}(A)$ mais qui n'est pas unique si $\ker(A) \neq \{0\}$).

On a donc trouvé x solution du problème au sens des moindres carrés.
La fonction J est différentiable sur $\mathbb{R}^n$ et son gradient vaut

$$grad(J)(x) = 2(A^T Ax - A^T b).$$

En effet, on vérifie facilement que

$$J(x+y) - J(x) = 2y^T(A^T Ax - A^T b) + \|Ay\|_2^2. \tag{4}$$

Si x minimise J, on a $grad(J)(x) = 0$ et x vérifie (3). Réciproquement, si x vérifie (3), on voit d'après (4) que $J(x+y) - J(x) = \|Ay\|_2^2 \geq 0$, $\forall y \in \mathbb{R}^n$, ce qui montre qye x minimise J. ■

Proposition 6.1 *Si* $\text{rang}(A) = n$*, le problème aux moindres carrés a une solution unique* $x = (A^T A)^{-1} A^T b$*. Sinon, les solutions du problème aux moindres carrés forment un espace affine parallèle à* $\ker(A)$*.*

Démonstration. On a

$$y \in \ker(A^T A) \Rightarrow A^T Ay = 0 \Rightarrow y^T A^T Ay = 0 \Leftrightarrow \|Ay\|_2 = 0 \Leftrightarrow y \in \ker(A)$$

et $\ker(A) \subset \ker(A^T A)$. Donc $\ker(A) = \ker(A^T A)$.
Si $\text{rang}(A) = n$, $\ker(A) = \{0\}$ et l'équation normale (3) a une solution unique $x = (A^T A)^{-1} A^T b$ et il en va de même pour le problème aux moindres carrés par le Lemme 6.1. Sinon, les solutions de (3) forment un espace affine. ■

On peut trouver une matrice orthogonale $Q \in \mathcal{M}_m(\mathbb{R})$ et une matrice triangulaire supérieure $R \in \mathcal{M}_{m,n}(\mathbb{R})$ telles que $A = QR$. Le système $A^T Ax = A^T b$ est équivalent à $R^T Rx = R^T Q^T b$, qui est facile à résoudre (une descente et une remontée), si $\text{rang}(R) = n$. De plus $\text{cond}_2(R^T R) = \text{cond}_2(A^T A)$.

Remarque 6.1 *Il n'est pas recommandé d'utiliser une méthode des moindres carrés pour résoudre un système carré car* $\text{cond}_2(A^T A) = (\text{cond}_2(A))^2$ *(exercice, le vérifier), et on a vu qu'il fallait éviter les grands nombres de conditionnement pour des raisons de précision.*

Exercice. Trouver la droite du plan la plus proche au sens des moindres carrés des points $(0,1)$ $(1,0)$ $(2,3)$.