1.2 Méthodes directes : Élimination de Gauss et ses variantes

La formule de Cramer nous apporte une façon systématique de résoudre un système linéaire :

$$\text{Trouver un vecteur } \mathbf{x} \in \mathbb{R}^n, \text{ tel que } A\mathbf{x} = \mathbf{b} \iff \forall i \in [1, \dots, n] \quad x_i = \frac{\det(A_i(b))}{\det(A)}.$$

Toutefois il n'est pas toujours possible appliquer cette approche en pratique. Effectivement, les calculs des déterminants se font avec une relation récursive, et l'effort de calcul est de l'ordre $(n+1)!$ flops. Or la dimension typique d'un problème dans les applications est de l'ordre de $n = 10^2$ à $n = 10^7$.

	No. de flops de l'ordinateur				
n	10^9 (Giga)	10^{10}	10^{11}	10^{12} (Tera)	10^{15} (Peta)
10	10^{-1} sec	10^{-2} sec	10^{-3} sec	10^{-4} sec	négligeable
15	17 heures	1.74 heures	10.46 min	1 min	0.6 10^{-1} sec
20	4860 ans	486 ans	48.6 ans	4.86 ans	1.7 jour
25	h.l.	h.l.	h.l.	h.l.	38365 ans

FIGURE 5 – cf. Livre A. Quarteroni, R. Sacco, F. Saleri "Numerical Mathematics"

Objectif : Construire des algorithmes de resolution avec un coût de calculs raisonnable.

1.2.1 Algorithmes de descente et remontée

Définition 1.33 (Matrices triangulaires)**.**

(i) On dit qu'une matrice U est triangulaire supérieure si $\forall\, i > j \quad U_{i,j} = 0$,

(ii) On dit qu'une matrice L est triangulaire inférieur si $\forall\, i < j \quad L_{i,j} = 0$,

C'est à dire :

$$U = \begin{pmatrix} u_{11} & u_{21} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix}, \qquad L = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix}.$$

Proposition 1.34. *Soit L_1, L_2 (resp. U_1, U_2) deux matrices triangulaires inférieures (resp. supérieures), le produit L_1L_2 (resp. U_1U_2) est une matrice triangulaire inférieure (resp. supérieure).*

Démonstration.

□

Proposition 1.35. *Soit L (resp. U) une matrice triangulaire inférieure (resp. supérieure) inversible, son inverse L^{-1} (resp. U^{-1}) est une matrice triangulaire inférieure (resp. supérieure).*

Démonstration.

□

Dans le cas ou le système est de la forme :

$$L\mathbf{y} = \mathbf{b} \quad \text{ou} \quad U\mathbf{x} = \mathbf{y}$$

on a des algorithme de résolution très efficaces.

Exemple. Démontrons l'algorithme sur un exemple simple : Soit $L \in \mathbb{M}_{3\times 3}(\mathbb{R})$ une matrice inversible triangulaire inférieure.

▼

A présent, soient $L \in \mathbb{M}_{n\times n}(\mathbb{R})$ une matrice inversible triangulaire inférieure et

$\mathbf{b} \in \mathbb{R}^n$ un vecteur colonne. La résolution de $L\mathbf{y} = \mathbf{b}$ sous la forme

$$\begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \Bigg\downarrow$$

s'écrit sous la forme de l'algorithme de descente :

Remarque. Pour que cet algorithme puisse être appliqué, on doit avoir $l_{ii} \neq 0$ ce qui revient à dire que la matrice L est inversible.

Algorithme 1 : [de descente] Résoudre $L\mathbf{y} = \mathbf{b}$.

```
pour i = 1 à n faire
    y_i ← b_i;
    pour j = 1 à i − 1 faire
        y_i ← y_i − l_ij y_j;
    fin
    y_i ← y_i / l_ii
fin
```

Pour calculer la *complexité* de l'algorithme on doit estimer le nombre d'opérations algébriques à chaque étape : à chaque itération i, on effectue $i-1$ sommes, $i-1$ produits et une division, soit au total

$$\sum_{i=1}^{n}(2(i-1)+1) = 2\left(\sum_{i=1}^{n} i\right) - n = n^2 \text{ opérations.}$$

De même, on construit l'*algorithme de remontée* pour résolution d'un système triangulaire supérieure donné par :

$$\begin{pmatrix} u_{11} & u_{21} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \Bigg\uparrow$$

Soit $U \in \mathbb{M}_{n\times n}(\mathbb{R})$ une matrice inversible triangulaire supérieure, alors $\mathbf{x} \in \mathbb{R}^n$, tel que $U\mathbf{x} = \mathbf{y}$, vérifie :

Algorithme 2 : [de remontée] Résoudre $U\mathbf{x} = \mathbf{y}$

pour $i = n$ *à* 1 **faire**
  $x_i \leftarrow y_i$;
  pour $j = i + 1$ *à* n **faire**
    $x_i \leftarrow x_i - u_{ij}x_j$;
  fin
  $x_i \leftarrow x_i/u_{ii}$
fin

1.2.2 Méthode de Gauss : Réduction

L'idée de la méthode d'élimination de Gauss est de réduire le système linéaire $A\mathbf{x} = \mathbf{b}$ à un système équivalent (c'est à dire à un système qui admet la même solution que le système initial) de la forme suivante :

$$U\mathbf{x} = \widetilde{\mathbf{b}},$$

où U est une matrice triangulaire supérieure et $\widetilde{\mathbf{b}}$ est le vecteur de second membre modifié. Ensuite ce nouveau système se résout par l'algorithme de remontée.

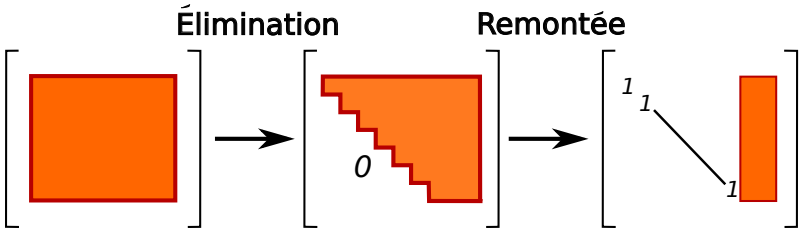


FIGURE 6 – Idée de l'algorithme de Gauss

Notons le système initial par $A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$. On suppose maintenant $a_{11} \neq 0$ et on introduit les multiplicateurs

$$m_{i1} = \frac{a_{i1}}{a_{11}}, \quad i = 2, \ldots, n.$$

La première étape de l'algorithme est alors :

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \ldots & a_{2n}^{(1)} \\ \vdots & \vdots & \vdots & \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \ldots & a_{nn}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix} \iff \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \ldots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}$$

où,

On note ce dernier système par $A^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$. En procédant ainsi de suite, on

construit un système $A^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$ où

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

Le coefficient $a_{kk}^{(k)}$ est appelé le *pivot*. Si on suppose $\forall\, k = 1, \dots, n-1$ $a_{kk}^{(k)} \neq 0$, pour $k = n$ on obtient $A^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$ avec

$$A^{(n)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & \dots & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & & & a_{2n}^{(2)} \\ \vdots & & \ddots & & & \vdots \\ 0 & & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & & & \ddots & \vdots \\ 0 & & & & & a_{nn}^{(n)} \end{pmatrix}$$ est une matrice triangulaire supérieure.

Résumé de passage d'une étape k à une étape $k+1$: Soit $a_{kk}^{(k)} \neq 0$, alors en définissant les multiplicateurs :

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1, \dots, n \tag{2}$$

on définit

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad i, j = k+1, \dots, n \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)}, \quad i = k+1, \dots, n. \end{aligned} \tag{3}$$

FIGURE 7 – Voir pour un exemple détaillé d'application de la méthode de Gauss.

Exercice. Calculer la solution $H_3\mathbf{x} = \mathbf{b}$ où H_3 est une matrice de Hilbert d'ordre 3 définie dans exemple 1.1.7 et $\mathbf{b} = [11/6, 13/12, 47/60]^t$, en utilisant (2) et (3).

En résumé :

Algorithme 3 : Méthode de Gauss

Données : A une matrice inversible de taille n, $\mathbf{b}$ un vecteur colonne de taille n

Résultat : $\mathbf{x}$ un vecteur de taille n tel que $A\mathbf{x} = \mathbf{b}$

// Triangularisation de A et transformation de $\mathbf{b}$

pour $k = 1$ *à* $n - 1$ **faire**
  pour $i = k + 1$ *à* n **faire**
    $m_{ik} = a_{ik}/a_{kk}$
    pour $j = k$ *à* n **faire**
      $a_{ij} = a_{ij} - m_{ik}a_{kj}$
    fin
    $b_i = b_i - m_{ik}b_k$
  fin
fin

// Résolution du système transformé par l'algorithme de remontée

pour $i = n$ *à* 1 **faire**
  $x_i = b_i$
  pour $j = i + 1$ *à* n **faire**
    $x_i = x_i - a_{ij}x_j$
  fin
  $x_i = x_i/a_{ii}$
fin

Remarque (Choix du pivot (voir TD))**.** Considérons $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{pmatrix}$ une matrice inversible. L'application de procédure d'élimination de Gauss au première étape donne : $A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{pmatrix}$ (et l'exécution s'arrête si on pense à rajouter la vérification dans le code). Si $a_{kk}^{(k)} = 0$ on cherche un autre pivot :

1. Soit en permutant les lignes k et p où $|a_{pk}^{(k)}| = \max_{j=k+1,\dots,n} |a_{jk}^{(k)}|$.
2. Soit en permutant les colonnes k et p où $|a_{kp}^{(k)}| = \max_{j=k+1,\dots,n} |a_{kj}^{(k)}|$.
3. Soit en permutant les deux $|a_{pq}^{(k)}| = \max_{i=k+1,\dots,n} \max_{j=k+1,\dots,n} |a_{ij}^{(k)}|$.

Considérons maintenant le coût de calculs de la méthode de Gauss. Pour l'étape d'élimination on a besoin de

$$2(n-1)n(n+1)/3 + n(n-1)$$

opérations. Ensuite n^2 pour l'algorithme de remontée. Ainsi au total $2n^3/3 + 2n^2$ opérations. Si on néglige les termes d'ordre inférieure on obtient le coût $\mathcal{O}(2n^3)$.

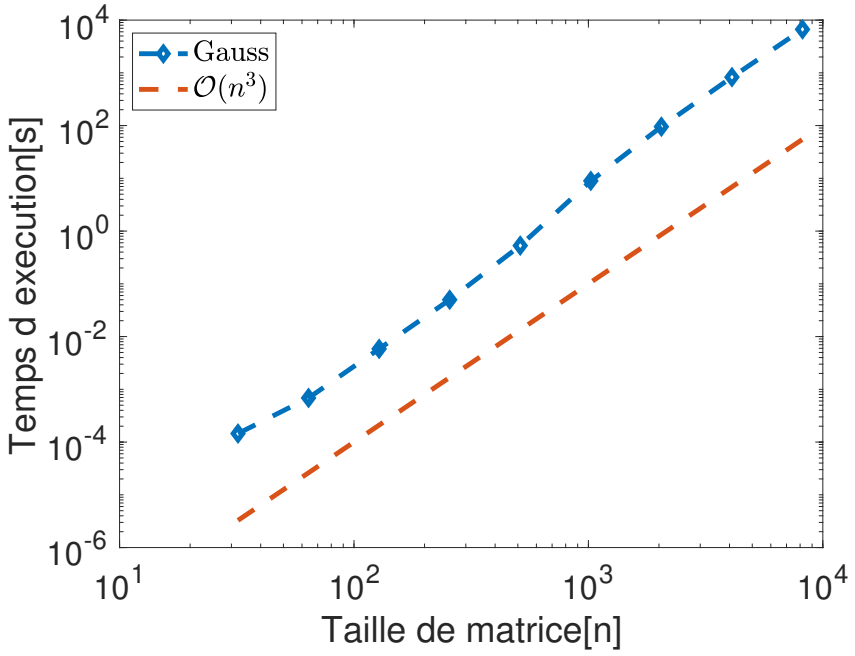


FIGURE 8 – Vérification numérique

	No. de flops de l'ordinateur		
n	10^9 (Giga)	10^{12} (Tera)	10^{15} (Peta)
10^2	$7 \cdot 10^{-4}$ sec	négligeable	négligeable
10^4	11 min	0.7 sec	$7 \cdot 10^{-4}$sec
10^6	21 ans	7.7 mois	11 min
10^8	h.l.	h.l.	21 ans

FIGURE 9 – Coût Numérique

1.2.3 Méthode de Gauss : factorisation LU

On peut voir l'algorithme décrit dans la section précédente comme une factorisation. Effectivement, en posant

$$M_k = \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & & 1 & 0 & & 0 \\ 0 & & -m_{k+1,k} & 1 & & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -m_{n,k} & 0 & \dots & 1 \end{pmatrix} = \mathrm{I}_n - \mathbf{m}_k \mathbf{e}_k^t$$

où $\mathbf{m}_k = (0, ..., 0, m_{k+1,k}, ..., m_{n,k})^t \in \mathbb{R}^n$, on trouve

$$A^{(k+1)} = M_k A^{(k)}, \text{ et donc } M_{n-1} M_{n-2} \dots M_1 A = U.$$

La matrice M_k à diagonale unité admet une matrice inverse définie par

$$M_k^{-1} = 2\mathrm{I}_n - M_k = \mathrm{I}_n + \mathbf{m}_k \mathbf{e}_k^T.$$

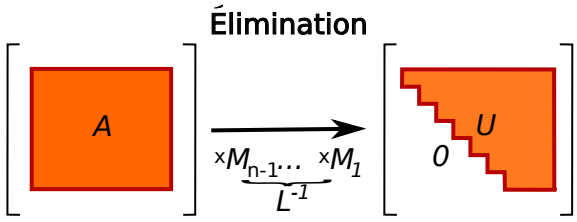
On remarque que $(\mathbf{m}_j \mathbf{e}_j^T)(\mathbf{m}_k \mathbf{e}_k^T)$ est une matrice nulle si $j \neq k$. Par conséquent,

$$M_1^{-1} M_2^{-1} \dots M_{n-1}^{-1} = \mathrm{I}_n + \sum_{k=1}^{n-1} \mathbf{m}_k \mathbf{e}_k = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ m_{21} & 1 & & & \vdots \\ \vdots & m_{32} & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ m_{n1} & m_{n2} & \dots & m_{n,n-1} & 1 \end{pmatrix}.$$

On définit alors

$$L = (M_{n-1} M_{n-2} \dots M_1)^{-1} = M_1^{-1} M_2^{-1} \dots M_{n-1}^{-1}$$

Et ainsi, $A = LU$.



Exercice. Calculer la décomposition LU pour la matrice $A = H_3$.

Algorithme 4 : Factorisation LU

Données : A une matrice inversible de taille n, $\mathbf{b}$ un vecteur colonne de taille n
Résultat : $\mathbf{x}$ un vecteur de taille n tel que $A\mathbf{x} = \mathbf{b}$
// Calcul de la décomposition LU. On initialise $L = I_n$. U est stockée dans A.
pour $k = 1$ *à* $n - 1$ **faire**
  pour $i = k + 1$ *à* n **faire**
    $\ell_{ik} = a_{ik}/a_{kk}$
    pour $j = k$ *à* n **faire**
      $a_{ij} = a_{ij} - \ell_{ik} a_{kj}$
    fin
  fin
fin
// Résolution de $L\mathbf{y} = \mathbf{b}$ par descente
pour $i = 1$ *à* n **faire**
  $y_i = b_i - \sum_{k=1}^{i-1} \ell_{ik} y_k$
fin
// Résolution de $U\mathbf{x} = \mathbf{y}$ par remontée
pour $i = n$ *à* 1 **faire**
  $x_i = 1/a_{ii} \left(y_i - \sum_{j=i+1}^{n} a_{ij} x_j \right)$
fin

Un autre avantage de la factorisation LU (par rapport à la réduction de Gauss) est qu'elle ne dépend pas du vecteur $\mathbf{b}$. On peut donc la calculer une fois pour A et ensuite seulement appliquer les algorithmes de descente et remontée pour chaque $\mathbf{b}$.

1.2.4 Méthode de Gauss : conditions sur A

Pour s'assurer que la méthode de Gauss atteint l'étape $(n-1)$, il ne suffit pas que A soit inversible, car cela ne garanti pas la condition nécessaire $\forall\, k = 1, \ldots, n-1$ $a_{kk}^{(k)} \neq 0$.

Theorem 1.36. *Soit A une matrice (pas forcément inversible). Elle admet une décomposition LU unique où L est à diagonale unité, i.e. $\forall\, i = 1, \ldots, n$ $L_{ii} = 1$ et U est une matrice triangulaire supérieure, si et seulement si toutes les sous matrices A_k, pour $k = 1, \ldots, n-1$, sont inversibles, où*

$$A_k \in \mathbb{M}_{k\times k}(\mathbb{R}) \quad \text{et} \quad (A_k)_{ij} = A_{ij}, \forall (i,j) \in [1, \ldots, k]^2.$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1k} & \ldots & a_{1n} \\ a_{21} & a_{22} & & \ldots & & a_{2n} \\ \vdots & & \ddots & & & \vdots \\ a_{k1} & & \ldots & a_{kk}^{(k)} & \ldots & a_{kn}^{(k)} \\ \vdots & & & & \ddots & \vdots \\ a_{n1} & & \ldots & & & a_{nn}^{(n)} \end{bmatrix}$$

Démonstration.

□

Exercice. Dire si les matrices suivantes admettent une unique décomposition LU :

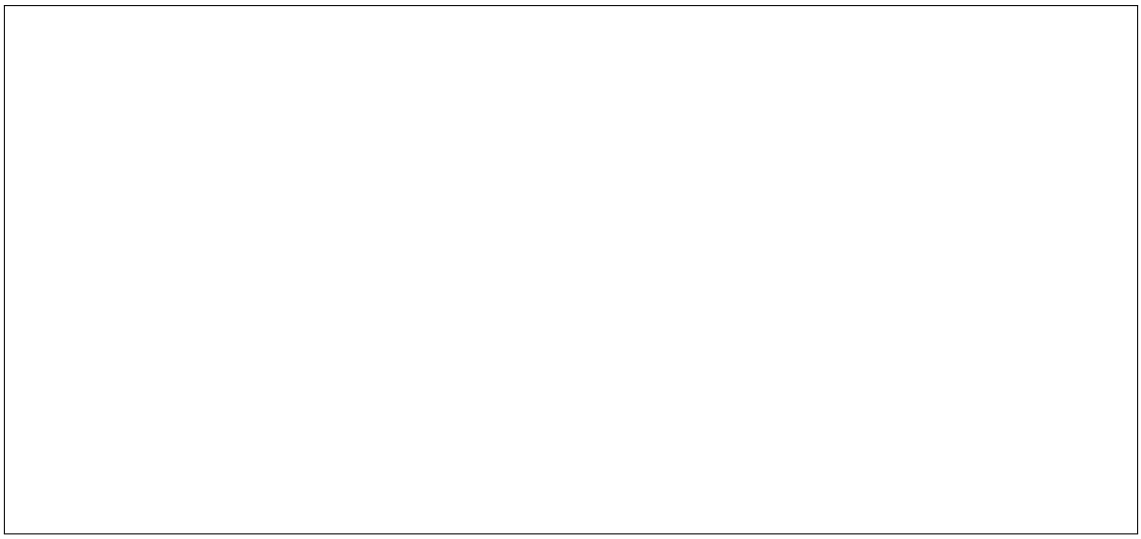
$$B = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 0 & 1 \\ 0 & 2 \end{pmatrix}.$$

En fait, C n'admet aucune décomposition LU alors que D admet une infinité de de décomposition de la forme :

$$D = L_\alpha U_\alpha, \quad \text{avec} \quad L_\alpha = \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix}, \quad U_\alpha = \begin{pmatrix} 0 & 1 \\ 0 & 2-\alpha \end{pmatrix}.$$

Proposition 1.37. *Si A est une matrice symétrique définie positive, alors ses sous matrices sont inversibles, et donc elle admet une décomposition LU.*

Démonstration.



□

Remarque (Stabilité de la méthode LU)**.** Pour une analyse profond d'erreurs d'arrondi (erreurs machine) vous pouvez consulter *A. Quarteroni, R. Sacco, F. Saleri.* Ici on mentionne juste que la stabilité de l'algorithme d'élimination de Gauss dépend de magnitudes des éléments de $A^{(k)}$, plus précisément de $\rho = \frac{\max\limits_{i,j,k} |A_{ij}^{(k)}|}{\max\limits_{i,j} |A_{ij}|}$. Mauvaise nouvelle : croissance exponentielle $\rho \sim 2^n$ est possible.

Exemple (**Wilkinson**)**.** Considérons une matrice définie comme suit

$$a_{ij} = \begin{cases} 1 & , \text{ si } i = j \vee j = n \\ -1 & , \text{ si } i > j \\ 0 & , \text{ sinon.} \end{cases}, \quad A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & 1 & 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \end{pmatrix}.$$

Alors on a :

$$A = LU, \quad l_{ij} = \begin{cases} 1 & , \text{ si } i = j, \\ -1 & , \text{ si } i > j, \\ 0 & , \text{ sinon}\,; \end{cases} \quad u_{ij} = \begin{cases} 1 & , \text{ si } i = j \\ 2^{\text{i-1}} & , \text{ si } j = n \\ 0 & , \text{ sinon.} \end{cases}$$

▼

Cependant, bonne nouvelle : en pratique, $\rho \sim \sqrt{n}$.

1.2.5 Méthode Choleski

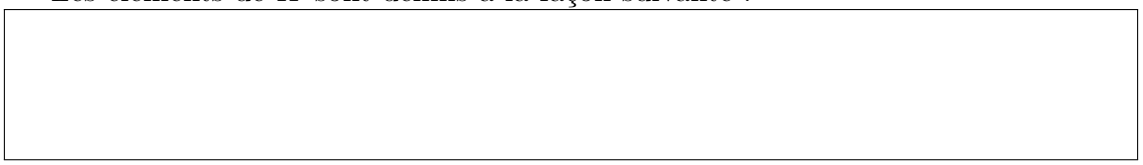
On va maintenant étudier la méthode de Choleski, qui est une méthode directe adaptée au cas où A est symétrique définie positive. On rappelle (Théorème 1.28) que si A est symétrique définie positive elle est en particulier inversible.

Proposition 1.38. *Si A est une matrice symétrique définie positive, alors elle admet une décomposition unique de la forme $A = HH^t$ où H est une matrice triangulaire inférieure à diagonale positive.*

Démonstration.

□

Les éléments de H sont définis à la façon suivante :



Exercice. Démontrez les formules ci-dessus.

Algorithme 5 : Méthode de Choleski

pour $k = 1$ *à* n **faire**
  $h_{kk} = a_{kk}$
  pour $i = 1$ *à* $k-1$ **faire**
    $h_{kk} = h_{kk} - h_{ik}^2$
  fin
  $h_{kk} = \sqrt{h_{kk}}$
  pour $i = k+1$ *à* n **faire**
    $h_{ik} = a_{ik}$
    pour $j = 1$ *à* $k-1$ **faire**
      $h_{ik} = h_{ik} - h_{ij}h_{kj}$
    fin
    $h_{ik} = h_{ik}/h_{kk}$
  fin
fin

Remarque (Pivot partiel et Choleski.)**.** Considérons une matrice A symétrique définie positive. On n'a pas besoin de permutation pour obtenir sa décomposition HH^t de Choleski. Par contre, numériquement, on utilise malgré tout la technique de pivot partiel pour minimiser les erreurs d'arrondi.

Comparaison Gauss/Choleski : Il est important de souligner que l'algorithme de Choleski permet à garder la nature symétrique de la matrice A ce qui permet d'économiser de la mémoire. Le coût de calcul de la résolution d'un système linéaire par la méthode de Choleski est $n^3/3 + \mathcal{O}(n^2)$, tandis que la méthode de Gauss demande $2n^3/3 + \mathcal{O}(n^2)$ opérations (cf TD). Dans le cas d'une matrice symétrique définie positive, la méthode de Choleski est donc environ deux fois moins chère. Mais attention dans le contexte actuel des ordinateurs modernes ça ne veut pas dire qu'elle est deux fois plus rapide (la gestion de la mémoire joue un rôle très important).

1.2.6 Factorisation QR : orthogonalisation de Gram-Schmidt

Idée : Factoriser la matrice A sous la forme $A = QR$ où R est une matrice triangulaire supérieure et Q est une matrice orthogonale ($Q^{-1} = Q^t$). La résolution du problème linéaire se fait alors en appliquant l'algorithme de remontée au vecteur $Q^t\mathbf{b}$.

Remarque. Dans un cas d'une matrice à coefficients complexes, Q est une matrice unitaire ($Q^{-1} = Q^*$).

Cette factorisation se construit en utilisant soit l'algorithme d'orthogonalisation de Gram-Schmidt soit les matrices de transformation : Householder ou Givens (cf TD).

Theorem 1.39 (Gram-Schmidt)**.** *Soit un ensemble de k vecteurs $(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ linéairement indépendant. On peut construire un ensemble de vecteurs $(\mathbf{q}_1, \ldots, \mathbf{q}_k)$ vérifiant :*

(i) $Vect(\mathbf{v}_1, \ldots, \mathbf{v}_k) = Vect(\mathbf{q}_1, \ldots, \mathbf{q}_k)$

(ii) $(\mathbf{q}_i, \mathbf{q}_j) = \delta_{ij}$

Démonstration.

□

Ainsi les nouveaux vecteurs $\mathbf{q}_i$ sont construit à la façon suivante :

$$\widetilde{\mathbf{q}}_i = \mathbf{v}_i - \sum_{j=1}^{i-1} (\mathbf{v}_i, \mathbf{q}_j)\, \mathbf{q}_j, \qquad \mathbf{q}_i = \widetilde{\mathbf{q}}_i / \|\widetilde{\mathbf{q}}_i\|$$

Algorithme 6 : Algorithme de Gram-Schmidt

```
pour k = 1 à n faire
    q_i = v_i
    pour k = 1 à i − 1 faire
        q_i = q_i − (v_i, q_j)q_j
    fin
    q_i = q_i/‖q_i‖
fin
```

Remarque. Attention, le calcul de la complexité pour cet algorithme ne peut pas être déterminée directement. En effet, les produits scalaires ne sont pas des opérations élémentaires

Theorem 1.40. *Soit A une matrice inversible. Elle admet une décomposition QR où Q est une matrice orthogonale et R est une matrice triangulaire supérieure.*

Démonstration.

□

Remarque. Numériquement, on évite l'utilisation de cette approche car l'algorithme d'orthogonalisation de G.S. est instable et conduit à une matrice Q non parfaitement orthogonale.

Exemple. Les ordinateurs ne font pas les opérations sur $\mathbb{R}$ "correctement". Calculons décomposition QR de matrice de Hilbert $H_{10} \in \mathbb{M}_{10\times 10}$ en utilisant l'algorithme 6, et faisons une simple vérification :

$$Q^tQ = \begin{pmatrix} 1.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0000 \\ 0.0000 & 1.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0000 \\ -0.0000 & -0.0000 & 1.0000 & 0.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0000 \\ -0.0000 & -0.0000 & -0.0000 & -0.0000 & 1.0000 & 0.0000 & -0.0008 & -0.0007 & -0.0007 & -0.0006 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & -0.0540 & -0.0430 & -0.0360 & -0.0289 \\ -0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0008 & -0.0540 & 1.0000 & 0.9999 & 0.9998 & 0.9996 \\ -0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0007 & -0.0430 & 0.9999 & 1.0000 & 1.0000 & 0.9999 \\ -0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0007 & -0.0360 & 0.9998 & 1.0000 & 1.0000 & 1.0000 \\ -0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0006 & -0.0289 & 0.9996 & 0.9999 & 1.0000 & 1.0000 \end{pmatrix}$$

▼

Il faut donc proposer un moyen stable de calculer la décomposition QR.

1.2.7 Méthode de Householder

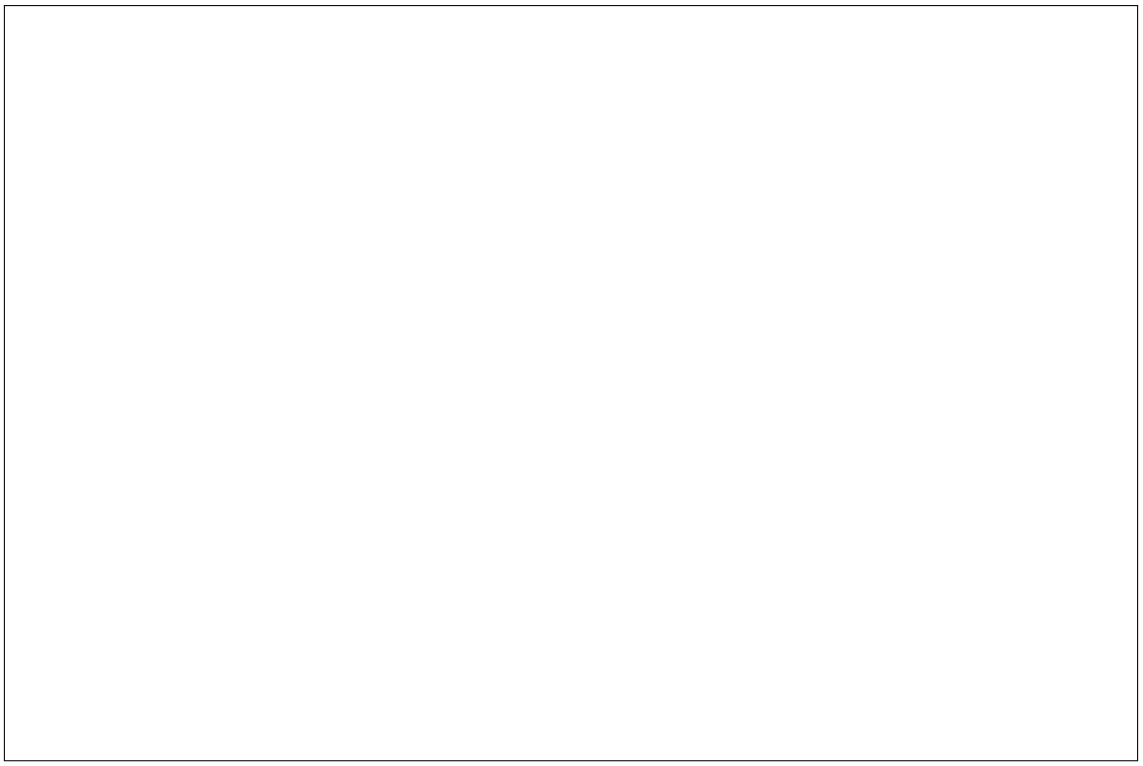
Soit $\mathbf{v} \in \mathbb{R}^n$. On définit la matrice élémentaire de Householder $H(\mathbf{v})$ comme suit

$$H(\mathbf{v}) = I - 2\frac{\mathbf{v}\,\mathbf{v}^t}{\|\mathbf{v}\|^2}$$

Interprétation géométrique :

Proposition 1.41. *La matrice $H(\mathbf{v})$ est symétrique et orthogonale et $H(\mathbf{v}\pm\|\mathbf{v}\|\mathbf{e}_k)\mathbf{v} = \mp\|\mathbf{v}\|\mathbf{e}_k$*

Démonstration.



□

Exercice. De la même manière que dans la preuve précédente, montrez que si $\mathbf{v} = \mathbf{b} - \mathbf{a} \neq 0$ avec $\|b\| = \|a\|$, alors $H(\mathbf{v})\mathbf{a} = \mathbf{b}$.

Remarque. On déduit de la proposition précédente qu'on peut transformer un vecteur $\mathbf{x}$ quelconque en un vecteur n'ayant qu'une coordonnée non nulle en multipliant par la matrice de Householder qui convient, via le choix du bon vecteur unitaire de la base canonique

Exemple. Considérons $\mathbf{x} = [1, 1, 1, 1]^t$.

▼

On remarque également que les matrices $H(\mathbf{u})$ (pour des vecteurs $\mathbf{u}$ bien choisis) nous permettent de réduire la matrice A à une matrice triangulaire inférieure. Cette idée forme une base pour la méthode de Householder :

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{2n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \times H^{(1)} \left(\mathbf{a}_1 + \|\mathbf{a}_1\| \, \mathbf{e}_1 \right) \quad \text{où} \quad \mathbf{a}_1 = (a_{11}, \dots, a_{n1})^t$$

$$\to A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix} \times H^{(2)} \left(\tilde{\mathbf{a}}_2 + \|\tilde{\mathbf{a}}_2\| \, \mathbf{e}_2 \right) \quad \text{où} \quad \tilde{\mathbf{a}}_2 = \left(0, a_{22}^{(2)}, \dots, a_{n2}^{(2)} \right)^t$$

$$\to A^{(3)} = \begin{pmatrix} a_{11}^{(3)} & a_{12}^{(3)} & a_{13}^{(3)} & \dots & a_{1n}^{(3)} \\ 0 & a_{22}^{(3)} & a_{23}^{(3)} & \dots & a_{2n}^{(3)} \\ 0 & 0 & a_{33}^{(3)} & \dots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & a_{nn}^{(3)} \end{pmatrix} \quad \dots$$

A l'étape k on construit une matrice $A^{(k)}$ comme

$$A^{(k)} = H^{(k-1)} H^{(k-2)} \dots H^{(1)} A$$

avec$\forall\, k$ $H^{(k)} = H\left(\tilde{\mathbf{a}}_k^{(k)} + \|\tilde{\mathbf{a}}_k^{(k)}\| \mathbf{e}_k \right), \tilde{\mathbf{a}}_k^{(k)} = (0 \dots, a_{kk}^{(k)}, \dots a_{nk}^{(k)})^t$ et

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & a_{1k}^{(k)} & \dots & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & \dots & a_{2k}^{(k)} & \dots & a_{2n}^{(k)} \\ 0 & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & 0 & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix}.$$

On déduit alors :

$$A^{(n)} = H^{(n-1)} H^{(n-2)} \dots H^{(1)} A = R(\text{triangulaire supérieure !})$$

et

$$Q = (H^{(n-1)} H^{(n-2)} \dots H^{(1)})^{-1} = \left(H^{(1)}\right)^t \dots \left(H^{(n-2)}\right)^t \left(H^{(n-1)}\right)^t = H^{(1)} \dots H^{(n-2)} H^{(n-1)}$$

Remarque. Attention ! ! La décomposition QR conduit a une matrice triangulaire supérieure différente de la matrice U de la décomposition LU, $R \neq U$.

La *complexité* de la méthode de Householder s'estime à $\mathcal{O}\left(\frac{4n^3}{3}\right)$ (donc plus élevée que pour l'algorithme de la décomposition LU). Mais une des avantages importants de la méthode de Householder est sa stabilité numérique.

Cette méthode s'applique également pour le cas des systèmes surdéterminés où $A \in \mathbb{M}_{n\times p}$, $n \geq p$ est une matrice rectangulaire, ainsi que pour des problèmes de minimisation écrits :

$$\text{Trouver } \min_{x\in\mathbb{R}^p} \|A\mathbf{x} - \mathbf{b}\|$$

où $A \in \mathbb{M}_{n\times p}$, $(n \neq p)$ (cf TD).