De nombreuses méthodes numériques conduisent à la résolution d'un système matriciel de la forme

$$Ax = b.$$

L'inconnue est le vecteur $x$, tandis que la matrice $A$ et le second membre $b$ sont connus.

Le but de ce chapitre est de voir des méthodes de résolution «automatique» des systèmes, qui soient le plus efficace possible.

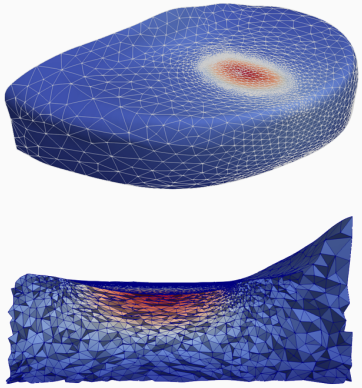
Aujourd'hui, la taille du vecteur $x$, représentant les inconnues, dépasse souvent les milliers, et peut même dépasser le milliard d'inconnues[1] !

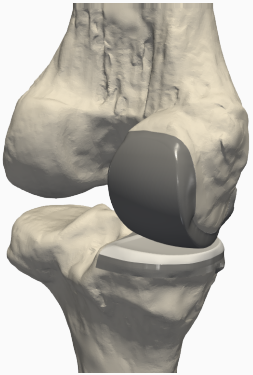
$\rightarrow$ Que peut représenter ce vecteur $x$ en pratique ?

[1] **Calcul à 4.6 milliards d'inconnues**

Exemples de simulations numériques de la «vraie vie» :

1) Simulation du contact prothèse-fémur [2]

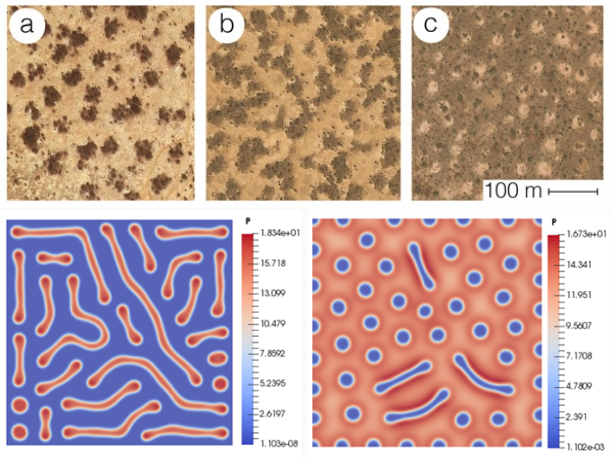




Le vecteur $x$ peut représenter une variable physique (déplacement, pression,...) calculée à chaque sommet d'un maillage.

[2] Prothèse de genou, Bodycad

2) Croissance de plante dans un milieu semi-aride. Le vecteur $x$ peut par exemple représenter la densité de plante présente, calculée sur une grille fine [3]




[3] Aerial images of flat terrain vegetation patterns in Sudan. 1a. Spot (11.6280, 27.9177). 1b. Labyrinth (11.1024, 27.8228). 1c. Gap patterns (10.7549, 28.5955). Images © Google, DigitalGlobe. Image credit: Karna Gowda.

Moral:

- De nombreuses méthodes numériques conduisent à la construction d'un système matriciel, souvent de grande taille.
- Nécessité de se munir de méthodes de résolutions efficaces.

**Plan de bataille du chapitre:**

- Étude des systèmes dit linéaires (élimination de Gauss, factorisation LU, Cholesky)
- Normes et conditionnement de matrices,
- Généralisation de la méthode de Newton pour résoudre les systèmes d'équations non linéaires.

Pour les systèmes linéaires, nous allons uniquement étudier les **méthodes dites directes**, i.e donnant la solution en un nombre fini et prédéterminé d'opérations.

Des **méthodes itératives**, nécessitant en théorie un nombre infinie d'opérations, existent également pour résoudre ces systèmes.

Pour les systèmes d'équations non-linéaires, l'utilisation de méthodes itératives devient obligatoire.

De manière générale, un système de $m$ **équations linéaires** à $n$ inconnues, est un système pouvant être écrit sous la forme suivante :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n = b_3 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n = b_m \end{cases} \tag{1}$$

où $\boldsymbol{x_1, x_2, \cdots, x_n}$ sont **les inconnues** et les $\boldsymbol{a_{ij}}$ **les coefficients** du système.

Le système (1), où $m = n$, s'écrit sous la forme matricielle $Ax = b$

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{b}$$

En pratique, la matrice $A$ et le second membre $b$ seront connus, il s'agira de déterminer le vecteur $x = (x_1, x_2, \cdots, x_n)$.

Pour le cours, **on aura toujours $n = m$**, donc autant d'inconnues que d'équations.

**On étudiera donc les systèmes où la matrice $A$ est carrée.**

Rappelons quelles conditions garantissent l'existence d'une solution au système.

**Rappel**

Les propositions équivalentes suivantes garantissent l'existence d'une solution au système matriciel $Ax = b$ (avec $A$ matrice carrée)

- La matrice $A$ est inversible, c-à-d $A^{-1}$ existe ($AA^{-1} = A^{-1}A = I$),
- Le déterminant de la matrice A, noté $\det(A)$ est non nul,
- La matrice $A$ est de rang maximal (nombre d'inconnues=nombre d'équations),
- Le système $Ax = 0$ admet seulement la solution nulle.

Dans ce chapitre, on traitera uniquement le cas où l'une des propositions précédente est vérifiée. On considérera donc toujours le cas où $A$ est inversible (non singulière), on écrira $x = A^{-1}b$.

✓ En pratique, on ne calculera jamais explicitement $A^{-1}$. Calculer l'inverse d'une matrice n'est pas numériquement efficace ! On va plutôt travailler sur le système linéaire de départ afin de déterminer $x$.

Comment résoudre de **manière systématique** un système linéaire ?

Regardons d'abord un cas particulier de matrices, les matrices diagonales.

**Définition (Matrice diagonale)**

La matrice $A = (a_{ij})_{1 \leq i,j \leq n}$ est dite **diagonale**, si ses entrées sont nulles en dehors de sa diagonale (i.e. si $i \neq j$ alors $a_{ij} = 0$).
Une telle matrice est donc de la forme

$$A = \begin{pmatrix} a_{11} & 0 & \dots & \dots & 0 \\ 0 & a_{22} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & a_{nn} \end{pmatrix}$$

On a alors, $\det(A) = a_{11} \times a_{22} \times ... \times a_{nn} = \Pi_{i=1}^{n} a_{ii}$

Dans ce cas, la résolution du système $Ax = b$ est immédiate,

$$x_i = \frac{b_i}{a_{ii}}, \ \ i = 1, \cdots, n$$

**Définition (Matrice triangulaire)**

La matrice $A = (a_{ij})_{1 \leq i,j \leq n}$ est dite **triangulaire inférieure** si tous les $a_{ij}$ sont nuls pour $i < j$. Une matrice triangulaire inférieure est donc de la forme

$$A = \begin{pmatrix} a_{11} & 0 & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & 0 & 0 & \dots & 0 \\ a_{31} & a_{32} & a_{33} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1\,1} & a_{n_1\,2} & a_{n-1\,3} & \dots & a_{n-1\,n-1} & 0 \\ a_{n\,1} & a_{n\,2} & a_{n\,3} & \dots & a_{n\,n-1} & a_{n\,n} \end{pmatrix}$$

Une matrice est dite **triangulaire supérieure** si tous les $a_{ij}$ sont nuls pour $j < i$ (transposée d'une matrice triangulaire inférieure). Dans les deux cas, $\det(A) = a_{11} \times a_{22} \times ... \times a_{nn} = \Pi_{i=1}^{n} a_{ii}$

La résolution du système $Ax = b$ avec $A$ triangulaire inférieure ou supérieure est également aisée. On effectue une **descente triangulaire** ou **remontée triangulaire**, selon le cas.

- $A$ triangulaire inférieure $\rightarrow$ descente triangulaire

$$x_1 = \frac{b_1}{a_{11}}, \quad x_i = \frac{\left(b_i - \sum_{k=1}^{i-1} a_{ik}x_k\right)}{a_{ii}}, \ i = 2, 3, \cdots, n. \tag{2}$$

- $A$ triangulaire supérieure $\rightarrow$ remontée triangulaire

$$x_n = \frac{b_n}{a_{nn}}, \quad x_i = \frac{\left(b_i - \sum_{k=i+1}^{n} a_{ik}x_k\right)}{a_{ii}}, \ i = n-1, n-2, \cdots, 2, 1. \tag{3}$$

**☣ Attention !**

Les formules (2) et (3) sont valides si les $a_{ii}$ sont tous non nuls. Dans le cas contraire, cela signifie que $\det(A) = 0$ est donc que la matrice n'est pas inversible. Le système $Ax = b$ n'admet alors pas de solution unique.

Combien coûte (en nombre d'opérations) une descente (resp. remontée) triangulaire ?

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ multiplications/divisions $\rightarrow\ O(n^2)$
- $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ additions/soustractions $\rightarrow\ O(n^2)$

Les méthodes de descentes et remontées triangulaires sont très génériques, mais nécessitent que $A$ soit triangulaire.

L'idée va donc être de transformer une matrice carrée quelconque en matrice triangulaire.

$\rightarrow$ **Élimination de Gauss**

L'élimination de Gauss consiste à effectuer des opérations élémentaires sur les lignes de la matrice $A$.

**But:** Introduire des zéros sous la diagonale de la matrice $A$ afin de la rendre triangulaire supérieure.

On note $l_i$ la ligne $i$ de la matrice $A$. Les trois opérations élémentaires sont les suivantes :

- $(l_i \leftarrow \lambda l_i)$ : remplacer la ligne $i$ par un multiple d'elle-même,
- $(l_i \leftrightarrow l_j)$ : intervertir la ligne $i$ et $j$,
- $(l_i \leftarrow l_i + \lambda l_j)$ : remplacer la ligne $i$ par la ligne $i$ plus un multiple de la ligne $j$.

Chacune de ces opérations revient à multiplier la matrice $A$ par une matrice inversible $W$.

**Remarque(s)**

En multipliant par une telle matrice inversible $W$, la solution du système de départ n'est pas modifiée

$$WAx = Wb \iff W^{-1}WAx = W^{-1}Wb \iff Ax = b$$

- $(l_i \leftarrow \lambda l_i)$ : $W$ est une matrice ressemblant à la matrice identité, sauf pour le coefficient $W_{ii}$ qui vaut $\lambda$,
- $(l_i \leftrightarrow l_j)$ : $W$ est une matrice où la ligne $i$ et la ligne $j$ de la matrice identité sont permutées,
- $(l_i \leftarrow l_i + \lambda l_j)$ : $W$ est une matrice ressemblant à la matrice identité, sauf pour le coefficient $W_{ij}$ qui vaut $\lambda$.

En pratique, dans l'élimination de Gauss, on ne multiplie pas par les matrices $W$, on fait plutôt les opérations élémentaires sur la matrice dite augmentée, *i.e* la matrice que l'on obtient en ajoutant le membre de droite $b$ à la matrice $A$, c'est-à-dire:

$$\left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right)$$

Les opérations élémentaires sont faites **à la fois sur la matrice $A$ et le second membre $b$**.

Une fois la matrice $A$ rendu triangulaire supérieure, il ne reste plus qu'à effectuer une remontée triangulaire afin d'obtenir la solution $x$.

Avantage : permet de traiter des matrices quelconques,

Inconvénient : **si l'on change le second membre $b$, on doit tout recommencer !**

Dans l'élimination de Gauss, on effectue les opérations directement sur $A$ et $b$, les matrices des opérations élémentaires ne sont pas conservées.

$\rightarrow$ On «jette» de l'information en effectuant l'élimination de Gauss !

La solution va être de conserver ces matrices.

Soit $T = W_N W_{N-1} \cdots W_1$ la matrice représentant la suite d'opérations élémentaires rendant la matrice $A$ triangulaire supérieure. Cette matrice ne dépend pas de $b$ et l'on a

$$TAx = Tb$$

Posons $U = TA$, $U$ est la matrice triangulaire supérieure, et

$$T^{-1}Ux = T^{-1}Tb = b$$

Autrement dit, $T^{-1}U = A$, et posons

$$L = T^{-1} = (W_N W_{N-1} \cdots W_1)^{-1} = W_1^{-1} W_2^{-1} \cdots W_N^{-1}$$

$L$ est la matrice triangulaire inférieure qui a permis d'éliminer les termes non nuls sous la diagonale de la matrice $A$ dans l'élimination de Gauss.

On a donc la décomposition $A = LU$, cependant **cette décomposition n'est pas unique**. Elle le sera en ajoutant des contraintes supplémentaires sur $L$ ou $U$.

**Théorème (Décomposition LU)**

Si $A \in M_n(\mathbb{R})$, telle que les sous matrices d'ordre $1 \leq k \leq n$ (sous matrices principales):

$$A^k = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kk} \end{pmatrix}$$

soient inversibles ($\det(A^k) \neq 0,\ 1 \leq k \leq n$). Alors il existe une matrice triangulaire supérieure U («Upper»), et une matrice triangulaire inférieure $L$ («Lower»), telle que

$$A = LU$$

**Si l'on fixe la diagonale de $L$ (ou de $U$) avec des $1$, la décomposition est unique.**

On a donc le choix dans la matrice où l'on place les 1 sur la diagonale. La factorisation $LU$ sera différente suivant que l'on place les 1 sur la diagonale de $L$ ou la diagonale de $U$.

En quoi cela nous aide-t-il à résoudre $Ax = b$ ?

$$Ax = b \iff LUx = b$$

Posons $Ux = y$. **La résolution de $Ax = b$ se fait alors en deux étapes :**

1. Résoudre, par une descente, $Ly = b \ \ (\rightarrow \ y)$
2. Résoudre, par une remontée, $Ux = y \ \ (\rightarrow \ x)$

C'est la **méthode de résolution par décomposition $LU$**.

- Si l'on change le second membre $b$, on peut réutiliser la factorisation $LU$ de la matrice $A$. Il suffira à nouveau d'effectuer une descente puis une remontée pour trouver $x$,
- L'hypothèse $\det(A^k) \neq 0$ permet de s'assurer qu'il n'y aura pas de pivots nuls pendant la décomposition,
- Lorsque les $1$ sont sur la diagonale de $L$ ($L_{ii} = 1$), il s'agit de la **décomposition de Doolittle** (ce que fait Matlab par défaut). Lorsque les $1$ sont sur la diagonale de $U$ ($U_{ii} = 1$), il s'agit de la **décomposition de Crout**.

Que faire en cas de pivots nuls ? → On réalise une **permutation des lignes** (exemple 3.33 p.115). Cela permet alors d'avoir un théorème un peu plus général.

**Théorème (Décomposition $PA = LU$)**

Soit $A$ une matrice inversible. Il existe $P$ une matrice produit de matrice de permutations, $L$ triangulaire inférieure et $U$ triangulaire supérieure telles que:

$$PA = LU, \text{ avec } |\det(P)| = 1$$

En pratique:

- Une permutation de ligne est toujours faite avant de faire les opérations pour annuler les termes de la colonne.
- Même lorsque les permutations de lignes ne sont pas «nécessaire», il est courant d'en réaliser afin que le pivot ($l_{ii}$ ou $u_{jj}$) soit de valeur absolue la plus grande possible.

Voyons comment déterminer de façon automatique les matrices $L$ et $U$, pour en tirer un algorithme implémentable.

**Algorithme de décomposition LU (Crout):** Supposons connues les $j-1$ premières colonnes de $L$ et les $j-1$ premières lignes de $U$, on veut calculer la ligne $j$ de $U$ et la colonne $j$ de $L$

- Calcul du pivot:

$$l_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk} u_{kj}$$

- Calcul de la colonne $j$ de $L$ pour $j+1 \leq i \leq n$:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}$$

- Calcul de la ligne $j$ de $U$ pour $j+1 \leq i \leq n$:

$$u_{ji} = \frac{1}{l_{jj}} \left( a_{ji} - \sum_{k=1}^{j-1} l_{jk} u_{ki} \right)$$

Je vous **déconseille fortement** de retenir ces formules !

Il suffit d'écrire ce que l'on «veut». Exemple sur une matrice $4 \times 4$ et la décomposition de Crout.

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

1. Première colonne de $L$ : (lignes de $L$) $\times$ (première colonne de $U$)

$$l_{11} = a_{11}, \quad l_{21} = a_{21}, \quad l_{31} = a_{31}, \quad l_{41} = a_{41}.$$

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix}$$

2. Première ligne de $U$ : (première ligne de $L$) $\times$ (colonnes de $U$)

$$l_{11}u_{12} = a_{12} \Rightarrow u_{12} = a_{12}/l_{11}, \quad u_{13} = a_{13}/l_{11}, \quad u_{14} = a_{14}/l_{11},$$

$$\begin{pmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Deuxième colonne de $L$ : (lignes de $L$) $\times$ deuxième colonne de $U$

$$l_{22} = a_{22} - l_{21}u_{12}, \quad l_{32} = a_{32} - l_{31}u_{12}, \quad l_{42} = a_{42} - l_{41}u_{12}.$$

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix}$$

4. Deuxième ligne de $U$ : (deuxième ligne de $L$) $\times$ (colonnes de $U$)

$$u_{23} = (a_{23} - l_{21}u_{13})/l_{22}, \quad u_{24} = (a_{24} - l_{21}u_{14})/l_{22}.$$

$$\begin{pmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5. Troisième colonne de $L$ : (lignes de $L$) $\times$ troisième colonne de $U$

$$l_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23}, \quad l_{43} = a_{43} - l_{41}u_{13} - l_{42}u_{23}.$$

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix}$$

6. Troisième ligne de $U$ : (troisième ligne de $L$) $\times$ (quatrième colonne de $U$)

$$u_{34} = (a_{34} - l_{31}u_{14} - l_{32}u_{24})/l_{33}.$$

$$\begin{pmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

7. Quatrième colonne de $L$ : (lignes de $L$) $\times$ (quatrième colonne de $U$)

$$l_{44} = a_{44} - l_{41}u_{14} - l_{42}u_{24} - l_{43}u_{34}.$$

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix}$$

Pour limiter l'espace mémoire utilisé, la décomposition $LU$ peut être stockée dans la matrice $A$ (la matrice $A$ est détruite au cours de l'opération)[4].

**Définition (LU compacte)**

La matrice $LU$ compacte est la matrice:

$$\begin{pmatrix} l_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ l_{21} & l_{22} & u_{23} & \dots & u_{2n} \\ l_{31} & l_{32} & l_{33} & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{pmatrix}$$

[4] voir `LU_Crout.m` et `LU_Doolittle.m` pour implémentations de la factorisation $LU$. Voir `script_LU.m` pour un exemple numérique.

**Nombre d'opérations pour la décomposition $LU$**

- nombre d'additions/soustractions : $\frac{2n^3-3n^2+n}{6} \rightarrow O(\frac{2n^3}{6})$
- nombre de divisions/multiplications : $\frac{n^3-n}{3} \rightarrow O(\frac{n^3}{3})$

**Nombre d'opérations pour les remontées**

- nombre d'additions/soustractions : $n^2 - n \rightarrow O(n^2)$
- nombre de divisions/multiplications : $n^2 \rightarrow O(n^2)$

Est-ce toujours raisonnable ?

En faisant $10^{12}$ opérations par seconde, il faudra :

- 185 heures $\approx$ 7.7 jours pour $10^6$ inconnues

La décomposition $LU$ permet de calculer facilement le déterminant de $A$, pour la décomposition de Crout, on aura:

$$\det(A) = \det(L)\det(U) = \det(L) = \prod_{i=1}^{n} l_{ii}$$

Et pour Doolittle,

$$\det(A) = \det(L)\det(U) = \det(U) = \prod_{i=1}^{n} u_{ii}$$

✓ Même si la matrice $A$ est une matrice creuse, *i.e* une matrice comportant de nombreux zéros, il n'y a aucune raison que la matrice compacte résultant de la factorisation $LU$ soit également creuse.

**Définition (Matrice à diagonale strictement dominante)**

Une matrice $A$ est à **diagonale strictement dominante** par lignes si:

$$\forall i,\ |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|$$

ou par colonnes si:

$$\forall j,\ |a_{jj}| > \sum_{i=1, i\neq j}^{n} |a_{ij}|$$

**Théorème**

Si $A$ est à diagonale strictement dominante par colonne ou par ligne:

- A est inversible.
- L'élimination de Gauss et la décomposition $LU$ se font sans permuter de lignes

**Conclusion factorisation LU:**

- Si $\det(A) = 0$, le système n'a pas de solution unique,
- Peu efficace pour $n$ grand,
- Si tous les déterminants des sous-matrices principales $A^k$ de $A$ **sont non nuls, alors la décomposition n'exige pas de permutations**. De même si $A$ est à diagonale strictement dominante,
- Si on a seulement $A$ inversible ($\det(A) \neq 0$) alors on a l'existence d'une décomposition de la forme $PA = LU$, avec $P$ une matrice de permutation.
- En pratique, on stocke la factorisation directement dans la matrice $A$. Attention, il est possible que la factorisation $LU$ prenne plus de place en mémoire que la matrice $A$ elle-même !

Voyons à présent la décomposition pour un type particulier de matrice.

**Définition (Matrice Symétrique Définie Positive (SDP))**

Soit $A$ une matrice symétrique ($A = A^t$). $A$ est dite définie positive si

$$Ax \cdot x = x^t Ax > 0, \quad \forall x \neq 0$$

**Théorème (Factorisation de Cholesky)**

Soit $A$ une **matrice symétrique**, les propositions suivantes sont équivalentes:

- $A$ est définie positive,
- Les déterminants de toutes les sous matrices principales $A^k$ de $A$ sont **strictement positifs**,
- (les valeurs propres de $A$ sont réelles et strictement positives),
- Il existe une factorisation dite de **Cholesky** de $A$ : $\boldsymbol{A = LL^t}$, $L$ étant une matrice triangulaire inférieure dont les termes sur la diagonale ($l_{ii}$) sont strictement positifs.

En pratique, on cherche donc une matrice $L$, triangulaire inférieure telle que

$$\begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ 0 & l_{22} & l_{32} & l_{42} \\ 0 & 0 & l_{33} & l_{43} \\ 0 & 0 & 0 & l_{44} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

Comme pour $LU$, on se sert directement de la matrice $A$ pour déterminer les coefficients de $L$.

Quelques considérations sur la factorisation de Cholesky:

- Si $A = LL^t$, alors $\det(A) = \det(L)\det(L^t) = \det(L)^2 = (\Pi_{i=1}^n l_{ii})^2$
- La condition $l_{ii} > 0$ dans le théorème précédent, sert à assurer l'unicité de la factorisation,
- On ne construit et ne stocke qu'une matrice triangulaire inférieure $L$,
- La résolution de $Ax = b$ se fait toujours en deux étapes, comme pour $LU$. Seule l'étape de décomposition coûte moins cher.

**☢ Attention !**

Il faut bien faire attention à vérifier les hypothèses du théorème avant d'effectuer la décomposition de Cholesky, c-à-d vérifier que $A$ est symétrique et définie positive.

Avant de se lancer dans les calculs pour voir si une matrice est «Choleskysable», peut-on éliminer d'office certaines matrices ?

Supposons que $A$ est «Choleskysable», alors $A$ est symétrique et $A = LL^t$

$$\Rightarrow a_{ii} = \sum_{k=1}^{n} l_{ik}^2 > 0$$

Autrement dit, si la matrice $A$ possède au moins un coefficient négatif sur sa diagonale, alors la décomposition de Cholesky ne sera pas applicable.

**Proposition**

Soit $A$ une matrice **symétrique** et a **diagonale strictement dominante**, dont les **termes diagonaux sont strictement positifs**, alors $A$ admet une factorisation de Cholesky.

**Attention !**

Si la matrice $A$ ne vérifie pas les conditions de la proposition, on ne peut rien conclure. Ce sont des conditions suffisantes mais pas nécessaires.

Surprise: les calculs pour la décomposition et la résolution ne seront pas exacts ! Deux sources d'erreurs:

- **Erreurs liées à la représentation** de $A$ et $b$, Exemple:

$$\begin{pmatrix} 1 & 2 \\ 1.1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10 \\ 10.4 \end{pmatrix}$$

La solution exacte est $x = (4, 3)$. Si dans la matrice on prend $1.05$ au lieu de $1.1$, la solution exacte est $x = (8, 1)$.

→ **Mauvaise stabilité** par rapport aux variations des coefficients[5].

✓ En pratique la représentation des nombres sur ordinateur fera en sorte que $A$ et $b$ ne seront pas représentés exactement, ce qui peut donc avoir de grandes répercussions sur la solution du système.

- **Erreurs liées à la méthode de résolution** et provenant des calculs en arithmétique flottante (ex: divisions et soustractions dangereuses pendant les remontées ou descentes triangulaires).

[5] Voir le fichier `script_stabilite.m` pour un exemple numérique.

**Solutions partielles pour limiter les erreurs:**

- Dans $LU$ faire des permutations de lignes pour choisir le plus grand pivot.
- Mettre les lignes à l'échelle, on divise chaque ligne par le plus grand coefficient de la ligne en valeur absolue,

$$a_{ij} \leftarrow \frac{a_{ij}}{\max_j a_{ij}}, \quad i = 1, \ldots, n.$$

Cependant, pas de solutions miracles. On va se définir des outils pour chercher à mesurer ces erreurs.

**Objectifs :**

- Mesurer l'écart entre la solution numérique et la solution exacte,
- Mesurer la sensibilité du système à des perturbations.

Comment mesurer la distance entre deux vecteurs ?

**Définition (Norme vectorielle)**

Une norme est une application de $\mathbb{R}^n$ dans $\mathbb{R}^+$ qui à $x \in \mathbb{R}^n$ associe $\|x\|$ telle que:

- $\|x\| \geq 0$ et $\|x\| = 0 \Rightarrow x = 0$.
- Si $\alpha \in \mathbb{R}$, $\|\alpha x\| = |\alpha| \, \|x\|$
- $\|x + y\| \leq \|x\| + \|y\|$

Des exemples de normes:

- Norme 1 : $\|x\|_1 = \sum_{k=1}^{n} |x_k| = |x_1| + |x_2| + \cdots + |x_n|$
- Norme 2 : $\|x\|_2 = \sqrt{\sum_{k=1}^{n} |x_k|^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$
- Norme infinie : $\|x\|_\infty = \max_k |x_k| = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$

Comment mesurer la distance entre deux matrices ?

**Définition (Norme matricielle)**

Une norme matricielle est une application de $M_n(\mathbb{R})$ dans $\mathbb{R}^+$ qui à $A \in M_n(\mathbb{R})$ associe $\|A\|$ telle que:

- $\|A\| \geq 0$ et $\|A\| = 0 \Rightarrow A = 0$.
- Si $\alpha \in \mathbb{R}$, $\|\alpha A\| = |\alpha| \, \|A\|$
- $\|A + B\| \leq \|A\| + \|B\|$
- $\|AB\| \leq \|A\|\|B\|$

**Définition (Norme matricielle induite ou subordonnée)**

Pour une norme vectorielle donnée $\| \cdot \|_i$, la norme matricielle induite est définie par:

$$\|A\|_i = \sup_{\|x\|_i \neq 0} \frac{\|Ax\|_i}{\|x\|_i} = \sup_{\|x\|_i = 1} \|Ax\|_i$$

**Exemples de normes matricielles usuelles**:

- La norme matricielle induite par la norme 1 (max sur la somme, en valeurs absolues, par colonne):

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}| = \max_{1\leq j\leq n} \{|a_{1j}| + |a_{2j}| + \cdots + |a_{nj}|\}$$

- La norme matricielle induite par la norme infinie (max sur la somme, en valeurs absolues, par ligne):

$$\|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}| = \max_{1\leq i\leq n} \{|a_{i1}| + |a_{i2}| + \cdots + |a_{in}|\}$$

✓ On a $\|A\|_1 = \|A^t\|_\infty$, et donc si $A$ symétrique $\|A\|_1 = \|A\|_\infty$

- Norme de Frobenius (norme induite par aucune norme):

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$$

**Définition (Normes compatibles)**

Une norme vectorielle $\|\cdot\|_v$ et une norme matricielle $\|\cdot\|_M$ sont compatibles si pour tout $x \in \mathbb{R}^n$ et $A \in M_n(\mathbb{R})$:

$$\|Ax\|_v \leq \|A\|_M \|x\|_v$$

**Théorème (Normes compatibles)**

- Toutes les normes matricielles induites sont compatibles avec leurs normes vectorielles,

$$\|Ax\|_1 \leq \|A\|_1 \|x\|_1, \quad \|Ax\|_\infty \leq \|A\|_\infty \|x\|_\infty,$$

- La norme de Frobenius est compatible avec la norme euclidienne,

$$\|Ax\|_2 \leq \|A\|_F \|x\|_2,$$

En pratique, on utilise souvent la norme $1$ ou la norme $\infty$, car plus simple à calculer.

**Erreur absolue et relative pour la résolution** de $Ax = b$ : Soit $x^*$ la solution numérique et $x$ la solution exacte. Le **vecteur de l'erreur** est défini par

$$e = x - x^*.$$

On souhaite que la norme de ce vecteur, $\|e\|_v$ soit la plus faible possible. Pour chaque $x^*$, on peut également associé le vecteur

$$r = b - Ax^*.$$

$r$ est appelé **le vecteur résidu**. On a immédiatement

$$r = b - Ax^* = Ax - Ax^* = A(x - x^*) = Ae. \tag{4}$$

En utilisant (4), et en utilisant le fait que les normes sont compatibles, on montre le résultat suivant

**Proposition**

On contrôle l'erreur relative due à l'algorithme de résolution de $Ax = b$ via l'inégalité suivante

$$\frac{1}{\|A\|_M \|A^{-1}\|_M} \frac{\|r\|_v}{\|b\|_v} \leq \frac{\|e\|_v}{\|x\|_v} \leq \|A\|_M \|A^{-1}\|_M \frac{\|r\|_v}{\|b\|_v}.$$

La quantité $\|A\|_M \|A_M^{-1}\|$ permet de mesurer l'effet de la matrice sur la résolution du système et porte un nom particulier

**Définition (Conditionnement)**

Le conditionnement d'une matrice $A$ inversible pour la norme matricielle $\|\cdot\|_M$ est défini par:

$$\mathrm{cond}_M(A) = \|A\|_M \|A^{-1}\|_M$$

- Le conditionnement n'est définie que pour les matrices inversibles,
- Le conditionnement **dépend de la norme choisie**,
- $\mathrm{cond}_M(A) \geq 1$,
- **Problème** : le conditionnement fait intervenir $A^{-1}$ qui en règle général est très difficile à calculer.

L'inégalité précédente s'écrit alors

$$\frac{1}{\mathrm{cond}_M(A)} \frac{\|r\|_v}{\|b\|_v} \leq \frac{\|e\|_v}{\|x\|_v} \leq \mathrm{cond}_M(A) \frac{\|r\|_v}{\|b\|_v}. \tag{5}$$

Concrètement, l'inégalité permet de dire que **dans le pire des cas**, l'erreur relative sera

$$\frac{\|e\|_v}{\|x\|_v} = \mathrm{cond}_M(A)\frac{\|r\|_v}{\|b\|_v}.$$

Il est donc préférable que le **conditionnement soit petit**, idéalement $\mathrm{cond}_M(A) \approx 1$. Un «grand» conditionnement indique qu'il est possible que l'erreur soit importante.

$\rightarrow$ Plus le conditionnement de $A$ est grand, plus on a intérêt à avoir un petit résidu, on devra alors être attentif à la qualité de la méthode de résolution.

L'inégalité suppose que la matrice $A$ et le second membre $b$ sont représentés exactement en machine, ce qui n'est pas le cas !

Cela nous amène à **l'erreur de représentation**.

**Erreur de représentation:** L'analyse est faite seulement sur $A$, mais le raisonnement est le même sur $b$.

Soit $\Delta A$ la matrice des erreurs de représentation sur $A$:

$$A^* = A + \Delta A$$

et $x^*$ la solution du système perturbé, $A^*x^* = b$. Alors on a le résultat suivant,

**Proposition**

Soit $x^*$ la solution de $(A + \Delta A)x^* = b$ et $x$ la solution exacte de $Ax = b$

$$\frac{\|x - x^*\|_v}{\|x^*\|_v} = \frac{\|e\|_v}{\|x^*\|_v} \leq \text{cond}_M(A) \frac{\|\Delta A\|_M}{\|A\|_M}$$

Que nous dit cette inégalité ?

- Si $\text{cond}_M(A)$ est grand, une petite variation de $A$ pourrait résulter en une grande variation sur la solution du système.
- Inversement, si $\text{cond}_M(A)$ est petit, une petite variation de $A$ entraîne une petite variation sur la solution du système.

Il reste un point à voir, comment calculer $\mathrm{cond}_M(A)$ **sans inverser** $A^{-1}$ ? En utilisant l'inégalité (5), on obtient **une borne minimale**

$$\mathrm{cond}_M(A) \geq \max\left(\frac{\|e\|_v\|b\|_v}{\|r\|_v\|x\|_v}, \frac{\|r\|_v\|x\|_v}{\|e\|_v\|b\|_v}\right)$$

✓ La borne dépend de $b$, $x$ et la méthode de résolution. Le conditionnement ne dépend que de $A$.

En pratique, $x$ est inconnu ! $\rightarrow$ on se construit un problème[6].

1. On choisit $y$ quelconque et on calcul $b = Ay$,
2. On calcul $y^*$ avec la méthode de résolution désirée,
3. On calcul $r = b - Ay^*$ et $e = y - y^*$,
4. On calcul le $\max$

La borne obtenue dépend du point $y$ choisi arbitrairement, de la méthode de résolution et de la norme choisie. Mais cela donne toujours «une idée» du conditionnement. C'est le mieux que l'on puisse faire sans calculer $A^{-1}$.

[6] Voir le fichier `script_borne_conditionnement.m` pour exemple numérique.

**Conclusion sur le conditionnement**:

- On a toujours $\mathrm{cond}_M(A) \geq 1$.
- Le conditionnement permet de contrôler l'erreur (relative), il ne dit rien, a priori, sur la valeur de l'erreur relative,
- Un conditionnement petit assure un bon contrôle de l'erreur de notre solution,
- Avoir un petit conditionnement, n'assure pas une erreur relative petite,
- Un conditionnement grand indique qu'il faut être attentif à l'algorithme choisi et aux possibles propagations d'erreurs,
- Un grand conditionnement n'assure pas que l'erreur relative sera grande,
- On évite, en pratique, de calculer $A^{-1}$ pour calculer le conditionnement. On préférera se servir de la borne inférieure trouvée.

BONUS ! Pas à l'examen

La notion de conditionnement est lié à la notion de stabilité d'un problème,

**Définition (Problème bien posé)**

Un problème est dit **bien posé (ou stable)** si la solution **existe**, est **unique**, et si **la solution dépend continûment des données**.

Le dernier point de la définition signifie que pour un problème stable, une petite perturbation des données (le second membre $b$ ou la matrice $A$), doit entraîner une petite perturbation de la solution (le vecteur $x$).

Lorsque ce n'est pas le cas (une petite perturbation des données entraîne une grande variation de la solution), on dit que le problème est mal posé, ou instable.

Si le conditionnement de $A$ est «petit» (et $A$ inversible), la résolution de $Ax = b$ est un problème stable par rapport aux perturbations des données.

BONUS ! Pas à l'examen

Si l'on sait que notre matrice $A$ est mal conditionnée, que peut-on faire ?
$\rightarrow$ On transforme le système de départ ! Soit $S$ une matrice inversible, alors

$$Ax = b \iff SAx = Sb$$

Le but est donc de trouver une matrice $S$ telle que $\mathrm{cond}_M(SA)$ soit meilleur que $\mathrm{cond}_M(A)$ (c'est-à-dire plus petit).
$\rightarrow$ On dit que $S$ est une **matrice de préconditionnement** (à gauche).

Trouver une bonne matrice de préconditionnement reste en général un problème délicat...

On fera cependant toujours en sorte que $S$ préserve les «bonnes propriétés» de la matrice $A$, telle que la symétrie.

Voyons à présent la résolution d'un **système d'équations non linéaires**. **Notation**:

- Soit $(f_i)_{1\leq i\leq n}$ $n$ fonctions de $n$ variables $(x_1, x_2, \ldots, x_n)$, différentiables,
- On définit pour $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, $F(x) = (f_1(x), f_2(x), \cdots, f_n(x))$.
- On veut résoudre $F(x) = 0$. On note $r = (r_1, r_2, \ldots, r_n)$ la solution (racine).

En terme de système, cela revient à trouver $x = (x_1, \ldots, x_n)$ tel que:

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0 \\ f_2(x_1, x_2, \cdots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \cdots, x_n) = 0 \end{cases}$$

Mis à part la méthode de la bissection, les méthodes du chapitre 2 peuvent être adaptées pour résoudre ce système. On présente ici la méthode la plus utilisée: la **méthode de Newton**.

**Principe**: On part de $x^0 = (x_1^0, x_2^0, \dots, x_n^0)$ - idéalement proche de la racine - et on cherche une correction $\delta x^0 = (\delta x_1^0, \delta x_2^0, \dots, \delta x_n^0)$ telle que pour tout $i$, $f_i(x^0 + \delta x^0) = 0$.
Cela nous donne, en réalisant un développement de Taylor à l'ordre 1,

$$\begin{cases} 0 = f_1(x^0 + \delta x^0) \approx f_1(x^0) + \delta x_1^0 \dfrac{\partial f_1}{\partial x_1}(x^0) + \delta x_2^0 \dfrac{\partial f_1}{\partial x_2}(x^0) + \cdots + \delta x_n^0 \dfrac{\partial f_1}{\partial x_n}(x^0) \\ 0 = f_2(x^0 + \delta x^0) \approx f_2(x^0) + \delta x_1^0 \dfrac{\partial f_2}{\partial x_1}(x^0) + \delta x_2^0 \dfrac{\partial f_2}{\partial x_2}(x^0) + \cdots + \delta x_n^0 \dfrac{\partial f_2}{\partial x_n}(x^0) \\ \vdots \\ 0 = f_n(x^0 + \delta x^0) \approx f_n(x^0) + \delta x_1^0 \dfrac{\partial f_n}{\partial x_1}(x^0) + \delta x_2^0 \dfrac{\partial f_n}{\partial x_2}(x^0) + \cdots + \delta x_n^0 \dfrac{\partial f_n}{\partial x_n}(x^0) \end{cases}$$

Soit, sous forme matricielle,

$$\vec{0} = F(x^0) + \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x^0) & \frac{\partial f_1}{\partial x_2}(x^0) & \dots & \frac{\partial f_1}{\partial x_n}(x^0) \\ \frac{\partial f_2}{\partial x_1}(x^0) & \frac{\partial f_2}{\partial x_2}(x^0) & \dots & \frac{\partial f_2}{\partial x_n}(x^0) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x^0) & \frac{\partial f_n}{\partial x_2}(x^0) & \dots & \frac{\partial f_n}{\partial x_n}(x^0) \end{pmatrix} \begin{pmatrix} \delta x_1^0 \\ \delta x_2^0 \\ \vdots \\ \delta x_n^0 \end{pmatrix}$$

**Définition (Matrice jacobienne)**

On note $J(x^i)$, la matrice dite jacobienne définie par

$$J(x^i) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x^i) & \frac{\partial f_1}{\partial x_2}(x^i) & \dots & \frac{\partial f_1}{\partial x_n}(x^i) \\ \frac{\partial f_2}{\partial x_1}(x^i) & \frac{\partial f_2}{\partial x_2}(x^i) & \dots & \frac{\partial f_2}{\partial x_n}(x^i) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x^i) & \frac{\partial f_n}{\partial x_2}(x^i) & \dots & \frac{\partial f_n}{\partial x_n}(x^i) \end{pmatrix} = \begin{pmatrix} \left(\nabla f_1(x^i)\right)^t \\ \left(\nabla f_2(x^i)\right)^t \\ \vdots \\ \left(\nabla f_n(x^i)\right)^t \end{pmatrix}$$

On cherche alors une correction $\delta x^0$ telle que

$$\vec{0} = F(x^0) + J(x^0)\delta x^0 \iff J(x^0)\delta x^0 = -F(x^0)$$

$-F(x^0)$ est appelé le **résidu** en $x^0$. La résolution de ce système matriciel nous donne la correction $\delta x^0 = -J(x^0)^{-1}F(x^0)$ à apportée à $x^0$.

**On ne calcul pas l'inverse, on résout le système !**

On recommence ensuite la démarche à partir de $x^1 = x^0 + \delta x^0$.

**Algorithme de Newton pour les systèmes:**

**Algorithme de Newton**, pseudo-code `script_Newton_systeme.m`

**Données :**

- Des fonctions $f_i$, $1 \leq i \leq n$ deux fois différentiables et un point de départ $x^0 = (x_0, x_1, \ldots, x_n)$. $F(x^0) = (f_1(x^0), \ldots, f_n(x^0))$
- Une tolérance d'arrêt $\varepsilon_a$ et un nombre max d'itérations $n_{max}$
- la précision machine $\varepsilon_m$.

**Résultat :** le vecteur $x^{i+1}$ qui contiendra une approximation de la racine $r = (r_1, \ldots, r_n)$

1 **tant que** $\frac{\|x^{i+1} - x^i\|}{\|x^{i+1}\| + \varepsilon_m} > \varepsilon_a$ *et* $n \leq n_{max}$ **faire**
2 Résoudre $J(x^i)\delta x^i = -F(x^i)$ ;
3 $x^{i+1} = x^i + \delta x^i$;
4 $i = i + 1$;
5 **fin**

- L'inverse de la matrice jacobienne $J_F(x^0)^{-1}$ joue le rôle que jouait $\frac{1}{f'(x_0)}$ en dimension 1,
- À chaque étape, on résout un système linéaire de dimension $n$. À chaque itération il faut que $J(x^i)$ soit inversible,
- La convergence est quadratique si $J_F(r)$ est inversible. Sinon elle est linéaire (on retrouve le comportement 1D dans le cas des racines multiples),
- Pour étudier la convergence on fera les mêmes tableaux qu'en 1D, en remplaçant les valeurs absolues par des normes vectorielles,
- Il est primordial (plus qu'en 1D) que $x^0$ soit proche de $r$ sous peine de divergence possible.
- Il faut aussi calculer la matrice jacobienne d'une fonction à plusieurs variables ce qui peut être très fastidieux.

**Des variantes**:

- **Quasi-Newton** : On ne calcule pas la matrice jacobienne à chaque itération. On garde la même pendant plusieurs itérations (on économise le calcul de dérivées et si factorisation $LU$ utilisée, on réduit le nombre de factorisation).
- **Newton modifiée**: On remplace les dérivées partielles par des approximations précises, pour $h > 0$ petit:

$$\frac{\partial f_i}{\partial x_j}(x) = \frac{f_i(x_1, \cdots, x_j + h, \cdots, x_n) - f_i(x_1, \cdots, x_j - h, \cdots, x_n)}{2h}$$

**Je dois être capable de :**

- Faire une méthode d'élimination de Gauss d'un système linéaire et définir les matrices équivalentes aux opérations élémentaires,
- Faire une décomposition et une résolution par $LU$ et Cholesky,
- Distinguer Doolittle et Crout,
- Reconnaître des matrices permettant d'appliquer ou d'exclure Cholesky,
- Comprendre la notion de conditionnement d'une matrice,
- Connaître le théorème sur le conditionnement,
- Appliquer le théorème de conditionnement pour caractériser une solution et son erreur relative,
- Construire une borne inférieure du conditionnement et je comprends ses limitations,
- Calculer une matrice Jacobienne,
- Utiliser la méthode de Newton pour les systèmes non linéaires, et comprend ses limitations et ses variantes.