Université Euromed de Fès

École d'Ingénierie Digitale et d'Intelligence Artificielle

Cours d'Analyse Numérique

Auteur

Prof. Safae ELHAJ-BEN-ALI

safae.elhajbenali@usmba.ac.ma

E.I.D.I.A.

Année scolaire 2020-2021

Table de matière

Table de matière

Qu'est ce que l'analyse numérique ?

L'*analyse numérique* est une branche des mathématiques appliquées née après la fondation en 1947 de "The Institute of Numerical Analysis" à l'Université de California de Los Angeles, dont l'objectif est de donner une réponse numérique à des problèmes qui n'ont pas de solution analytique (solution calculée à la main) où de concevoir et d'étudier des méthodes de résolution de certains problèmes mathématiques, en général issus de la modélisation de problèmes "réels", et dont on cherche à calculer la solution à l'aide d'un ordinateur.

Exemple

Calculer l'intégrale $I = \int_0^1 e^{x^2} dx$.

Il est impossible de donner une valeur exacte à I, dans ce cas, on peut appliquer des méthodes numériques pour évaluer la valeur de l'intégrale donnée. Pour cela on note $f(x) = e^{x^2}$ et on subdivise l'intervalle $[0,1]$ en des intervalles $[x_i, x_{i+1}]_{i=0,\dots,n-1}$, c'est à dire nous choisissons des points $x_i,\ i = 0, \dots, n$ tel que

$$x_0 = 0,\ x_1 = x_0 + h,\ \dots,\ x_n = x_{n-1} + h = x_0 + nh, \quad h = \max_{0 \le i \le n-1} |x_{i+1} - x_i|. \tag{1}$$

Par exemple pour $n = 10$ et après quelques opérations algébriques on obtient :

$$I = (x_1 - x_0) f(x_0) + (x_2 - x_1) f(x_1) + \cdots + (x_{10} - x_9) f(x_9).$$

Ainsi connaissant la valeur de $f(x_0)$ jusqu'à $f(x_9)$ on peut obtenir une valeur approchée de I.

Exemple

Calculer l'intégrale $I = \int_0^1 e^{x^2} dx$.
Il est impossible de donner une valeur exacte à I, dans ce cas, on peut appliquer des méthodes numériques pour évaluer la valeur de l'intégrale donnée. Pour cela on note $f(x) = e^{x^2}$ et on subdivise l'intervalle $[0, 1]$ en des intervalles $[x_i, x_{i+1}]_{i=0,\dots,n-1}$, c'est à dire nous choisissons des points x_i, $i = 0, \dots, n$ tel que

$$x_0 = 0,\ x_1 = x_0 + h,\ \dots,\ x_n = x_{n-1} + h = x_0 + nh, \quad h = \max_{0 \le i \le n-1} |x_{i+1} - x_i|. \tag{1}$$

Par exemple pour $n = 10$ et après quelques opérations algébriques on obtient :

$$I = (x_1 - x_0)f(x_0) + (x_2 - x_1)f(x_1) + \cdots + (x_{10} - x_9)f(x_9).$$

Ainsi connaissant la valeur de $f(x_0)$ jusqu'à $f(x_9)$ on peut obtenir une valeur approchée de I.

Exemple

Calculer l'intégrale $I = \int_0^1 e^{x^2} dx$.
Il est impossible de donner une valeur exacte à I, dans ce cas, on peut appliquer des méthodes numériques pour évaluer la valeur de l'intégrale donnée. Pour cela on note $f(x) = e^{x^2}$ et on subdivise l'intervalle $[0, 1]$ en des intervalles $[x_i, x_{i+1}]_{i=0,\dots,n-1}$, c'est à dire nous choisissons des points x_i, $i = 0, \dots, n$ tel que

$$x_0 = 0,\ x_1 = x_0 + h,\ \dots,\ x_n = x_{n-1} + h = x_0 + nh, \quad h = \max_{0 \le i \le n-1} |x_{i+1} - x_i|. \tag{1}$$

Par exemple pour $n = 10$ et après quelques opérations algébriques on obtient :

$$I = (x_1 - x_0)f(x_0) + (x_2 - x_1)f(x_1) + \cdots + (x_{10} - x_9)f(x_9).$$

Ainsi connaissant la valeur de $f(x_0)$ jusqu'à $f(x_9)$ on peut obtenir une valeur approchée de I.

L'analyse numérique $\Longrightarrow$ Résultat approché

Conséquence : L'analyse numérique permet d'évaluer et de développer les processus (Méthodes) de résolution de problèmes mathématiques par la voie du calcul numérique à partir de données numériques accessibles (par l'experience).

L'analyse numérique $\Longrightarrow$ Traitement de l'information

L'analyse numérique $\Longrightarrow$ Résultat approché

Conséquence : L'analyse numérique permet d'évaluer et de développer les processus (Méthodes) de résolution de problèmes mathématiques par la voie du calcul numérique à partir de données numériques accessibles (par l'experience).

L'analyse numérique $\Longrightarrow$ Traitement de l'information

L'analyse numérique $\Longrightarrow$ Résultat approché

Conséquence : L'analyse numérique permet d'évaluer et de développer les processus (Méthodes) de résolution de problèmes mathématiques par la voie du calcul numérique à partir de données numériques accessibles (par l'experience).

L'analyse numérique $\Longrightarrow$ Traitement de l'information

Analyse numérique et ordinateur

Depuis la deuxième guerre mondiale, les applications des mathématiques s'étendent à tous les secteurs d'activité et ceci grâce aux ordinateurs et leur rapidité de calcul. L'ordinateur est un outil incontournable pour simuler et modéliser les systèmes. Il existe souvent plusieurs façon d'approcher un problème pour le résoudre $\Longrightarrow$ **L'existence d'*algorithme***.

Definition

*On appelle **Algorithme** un processus ou une méthode numérique pour la résolution d'un problème toujours à partir de données accessible, en général, le problème d'analyse numérique se résume dans l'organigramme suivant :*

Données d'entrée ——— *Algorithme* ——— *Données de sortie*

Depuis la deuxième guerre mondiale, les applications des mathématiques s'étendent à tous les secteurs d'activité et ceci grâce aux ordinateurs et leur rapidité de calcul. L'ordinateur est un outil incontournable pour simuler et modéliser les systèmes. Il existe souvent plusieurs façon d'approcher un problème pour le résoudre $\Longrightarrow$ **L'existence d'*algorithme***.

Definition

On appelle ***Algorithme*** *un processus ou une méthode numérique pour la résolution d'un problème toujours à partir de données accessible, en général, le problème d'analyse numérique se résume dans l'organigramme suivant :*

Données d'entrée ——— *Algorithme* ——— *Données de sortie*

Definition

Un organigramme est une représentation schématique de toute les possibilités pour résoudre un problème considéré c-à-d une succession de calcul et de décision traduisant le processus de résolution.

Exemple

Considérant le système linéaire :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \iff \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\iff Ax = b$$

Pour résoudre $Ax = b$ on procède par l'organigramme suivant

Definition

Un organigramme est une représentation schématique de toute les possibilités pour résoudre un problème considéré c-à-d une succession de calcul et de décision traduisant le processus de résolution.

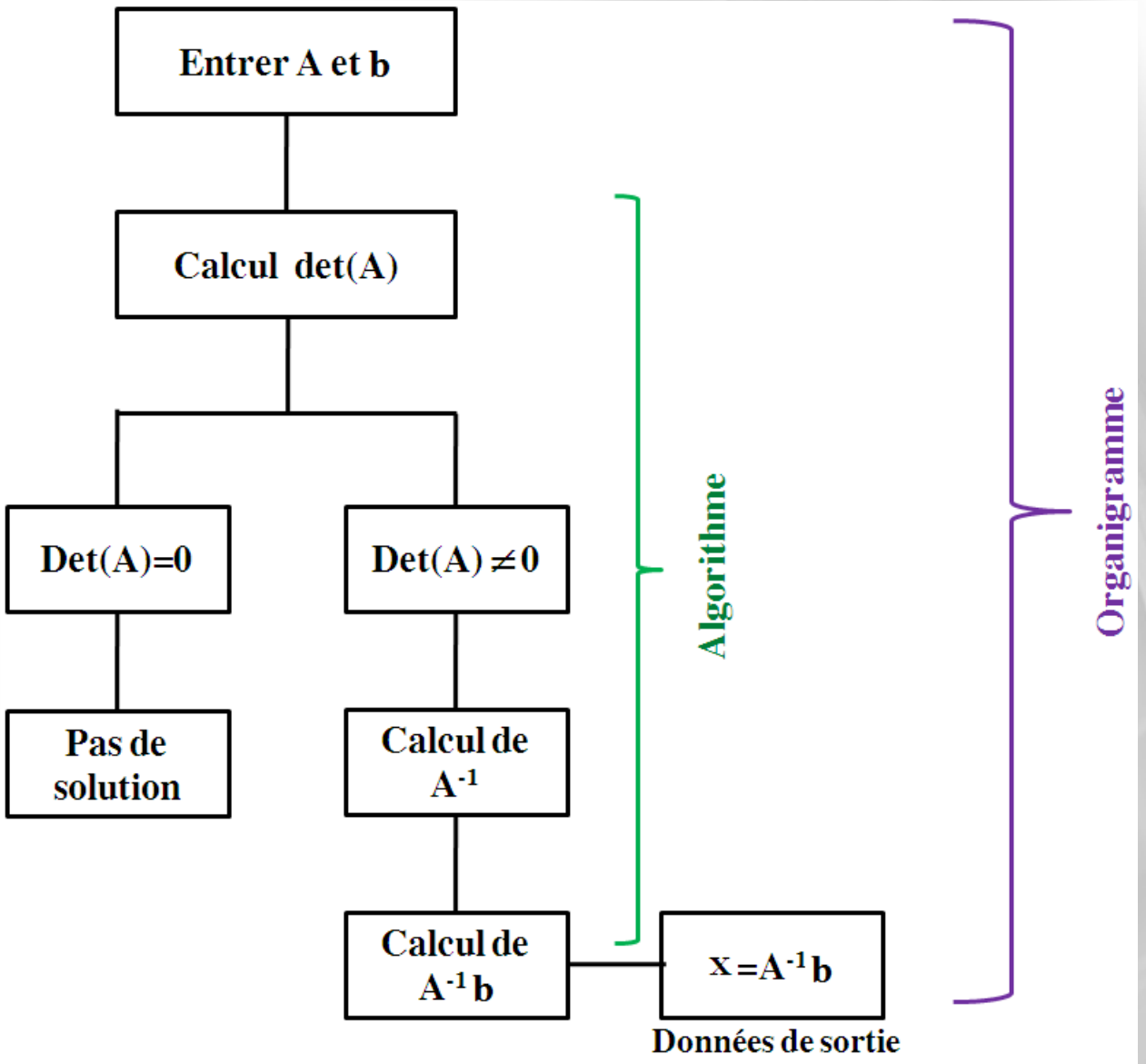
Exemple

Considérant le système linéaire :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \iff \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\iff Ax = b$$

Pour résoudre $Ax = b$ on procède par l'organigramme suivant


Entrer A et b
Calcul det(A)
Det(A)=0
Det(A) ≠0
Pas de solution
Calcul de A^{-1}
Calcul de $A^{-1}b$
$x=A^{-1}b$
Données de sortie
Algorithme
Organigramme

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*
- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - **Rapidité :** Réduire au maximum le nombre des opérations menant au résultat.
  - **Précision :** Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).
  - **Souple :** L'algorithme doit être facilement transposable à des problèmes différents.

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

Remarque

- *Un organigramme considère toute les possibilités de calcul puis donne une représentation du résultat (possibilité de visualiser les résultats sous forme de tableau, graphe, figure).*

- *Un algorithme peut être utile s'il satisfait un certain nombre de conditions :*

  - ***Rapidité :*** *Réduire au maximum le nombre des opérations menant au résultat.*

  - ***Précision :*** *Négliger toutes les erreurs commise au calcul c-à-d savoir contenir les effets des erreurs (erreurs de modélisation, de données, de représentation sur ordinateur ou de troncature).*

  - ***Souple :*** *L'algorithme doit être facilement transposable à des problèmes différents.*

D'où, l'elaboration d'un algorithme nécessite beaucoup d'effort et présente de nombreuses difficultés.

On considérera un réel comme connu numériquement si on connaît un certain nombre de chiffres de son développement décimal et la précision avec laquelle ce développement approchée est donnée.

Exemple

- $\sqrt{2} = 1.414213 \pm 10^{-6}$.
- $\pi = 3.14159 \pm 10^{-5}$.

On considérera un réel comme connu numériquement si on connaît un certain nombre de chiffres de son développement décimal et la précision avec laquelle ce développement approchée est donnée.

Exemple

- $\sqrt{2} = 1.414213 \pm 10^{-6}$.
- $\pi = 3.14159 \pm 10^{-5}$.

Les opérations algébrique sur ordinateur sont des approximations des opérations algébriques des mathématiques.

Exemple

Associativité : $u = \frac{(y+x)-x}{y}$; $v = \frac{y+(x-x)}{y}$

- **Calcul algébrique :** $\forall y \neq 0$ on a $u = v = 1$.
- **Informatique :** si $x = 1$ et $y = 10^{-17}$ on trouve $u = 0$ et $v = 1$ c-à-d dans la parenthèse $y + x$ l'ordinateur a négligé la valeur de y d'où l'ordinateur ne prend pas en considération l'infiniment petit.

Les opérations algébrique sur ordinateur sont des approximations des opérations algébriques des mathématiques.

Exemple

Associativité : $u = \frac{(y+x)-x}{y}$; $v = \frac{y+(x-x)}{y}$

- **Calcul algébrique :** $\forall y \neq 0$ on a $u = v = 1$.
- **Informatique :** si $x = 1$ et $y = 10^{-17}$ on trouve $u = 0$ et $v = 1$ c-à-d dans la parenthèse $y + x$ l'ordinateur a négligé la valeur de y d'où l'ordinateur ne prend pas en considération l'infiniment petit.

Les opérations algébrique sur ordinateur sont des approximations des opérations algébriques des mathématiques.

Exemple

Associativité : $u = \frac{(y+x)-x}{y}$; $v = \frac{y+(x-x)}{y}$

- **Calcul algébrique :** $\forall y \neq 0$ on a $u = v = 1$.
- **Informatique :** si $x = 1$ et $y = 10^{-17}$ on trouve $u = 0$ et $v = 1$ c-à-d dans la parenthèse $y + x$ l'ordinateur a négligé la valeur de y d'où l'ordinateur ne prend pas en considération l'infiniment petit.

Les opérations algébrique sur ordinateur sont des approximations des opérations algébriques des mathématiques.

Exemple

Associativité : $u = \frac{(y+x)-x}{y}$; $v = \frac{y+(x-x)}{y}$

- **Calcul algébrique :** $\forall y \neq 0$ on a $u = v = 1$.
- **Informatique :** si $x = 1$ et $y = 10^{-17}$ on aura $u = 0$ et $v = 1$ c-à-d dans la parenthèse $y + x$ l'ordinateur a négligé la valeur de y d'où l'ordinateur ne prend pas en considération l'infiniment petit.

Les opérations algébrique sur ordinateur sont des approximations des opérations algébriques des mathématiques.

Exemple

Associativité : $u = \frac{(y+x)-x}{y}$; $v = \frac{y+(x-x)}{y}$

- **Calcul algébrique :** $\forall y \neq 0$ on a $u = v = 1$.
- **Informatique :** si $x = 1$ et $y = 10^{-17}$ on aura $u \neq 0$ et $v = 1$ c-à-d dans la parenthèse $y + x$ l'ordinateur a négligé la valeur de y d'où l'ordinateur ne prend pas en considération l'infiniment petit.

Les opérations algébrique sur ordinateur sont des approximations des opérations algébriques des mathématiques.

Exemple

Associativité : $u = \frac{(y+x)-x}{y}$; $v = \frac{y+(x-x)}{y}$

- **Calcul algébrique :** $\forall y \neq 0$ on a $u = v = 1$.
- **Informatique :** si $x = 1$ et $y = 10^{-17}$ on aura $u = 0$ et $v = 1$ c-à-d dans la parenthèse $y + x$ l'ordinateur a négligé la valeur de y d'où l'ordinateur ne prend pas en considération l'infiniment petit.

- **Capacité de mémoire :**

La mémoire de l'ordinateur est limité, en effet, il effectue ses calculs en général par l'énumération binaire en base de 2 avec un nombre de digit qui est fini. Par exemple, les nombres qui possèdent un nombre infini de chiffres non nuls après la virgule, or ce n'est pas le cas pour $\frac{1}{3} = 0.3333...$ ou $\sqrt{2} = 1.4142....$

- **Capacité de mémoire :**

La mémoire de l'ordinateur est limité, en effet, il effectue ses calculs en général par l'énumération binaire en base de 2 avec un nombre de digit qui est fini. Par exemple, les nombres qui possèdent un nombre infini de chiffres non nuls après la virgule, or ce n'est pas le cas pour $\frac{1}{3} = 0.3333...$ ou $\sqrt{2} = 1.4142....$

Les erreurs commises dans les problème mathématiques peuvent être en principe classées en quatre catégories.

- Erreurs de modèle : Ces erreurs sont dues au fait que les modèles mathématiques sont plus ou moins idéalisés, ce qui donne lieu à plusieurs erreurs. Un exemple une planète n'est jamais parfaitement sphérique.

- Erreurs d'entrée : L'information d'entrée est rarement exacte, par exemple les mesures physiques ne sont jamais exactes : π, la gravité g, charge élémentaire e, ...

- Erreurs d'algorithme : En plus des erreurs d'entrée, l'algorithme lui même engendre des erreurs qui sont particulièrement dû aux opérations arithmétiques et à la précision de l'ordinateur.

- Erreurs de sortie : Elle sont engendrées par les erreurs précédentes et la précision de l'ordinateur.

Les erreurs commises dans les problème mathématiques peuvent être en principe classées en quatre catégories.

- Erreurs de modèle : Ces erreurs sont dues au fait que les modèles mathématiques sont plus ou moins idéalisés, ce qui donne lieu à plusieurs erreurs. Un exemple une planète n'est jamais parfaitement sphérique.
- Erreurs d'entrée : L'information d'entrée est rarement exacte, par exemple les mesures physiques ne sont jamais exactes : π, la gravité g, charge élémentaire e, ...
- Erreurs d'algorithme : En plus des erreurs d'entrée, l'algorithme lui même engendre des erreurs qui sont particulièrement dû aux opérations arithmétiques et à la précision de l'ordinateur.
- Erreurs de sortie : Elle sont engendrées par les erreurs précédentes et la précision de l'ordinateur.

Les erreurs commises dans les problème mathématiques peuvent être en principe classées en quatre catégories.

- Erreurs de modèle : Ces erreurs sont dues au fait que les modèles mathématiques sont plus ou moins idéalisés, ce qui donne lieu à plusieurs erreurs. Un exemple une planète n'est jamais parfaitement sphérique.

- Erreurs d'entrée : L'information d'entrée est rarement exacte, par exemple les mesures physiques ne sont jamais exactes : π, la gravité g, charge élémentaire e, ...

- Erreurs d'algorithme : En plus des erreurs d'entrée, l'algorithme lui même engendre des erreurs qui sont particulièrement dû aux opérations arithmétiques et à la précision de l'ordinateur.

- Erreurs de sortie : Elle sont engendrées par les erreurs précédentes et la précision de l'ordinateur.

Les erreurs commises dans les problème mathématiques peuvent être en principe classées en quatre catégories.

- Erreurs de modèle : Ces erreurs sont dues au fait que les modèles mathématiques sont plus ou moins idéalisés, ce qui donne lieu à plusieurs erreurs. Un exemple une planète n'est jamais parfaitement sphérique.
- Erreurs d'entrée : L'information d'entrée est rarement exacte, par exemple les mesures physiques ne sont jamais exactes : π, la gravité g, charge élémentaire e, ...
- Erreurs d'algorithme : En plus des erreurs d'entrée, l'algorithme lui même engendre des erreurs qui sont particulièrement dû aux opérations arithmétiques et à la précision de l'ordinateur.
- Erreurs de sortie : Elle sont engendrées par les erreurs précédentes et la précision de l'ordinateur.

Les erreurs commises dans les problème mathématiques peuvent être en principe classées en quatre catégories.

- Erreurs de modèle : Ces erreurs sont dues au fait que les modèles mathématiques sont plus ou moins idéalisés, ce qui donne lieu à plusieurs erreurs. Un exemple une planète n'est jamais parfaitement sphérique.

- Erreurs d'entrée : L'information d'entrée est rarement exacte, par exemple les mesures physiques ne sont jamais exactes : π, la gravité g, charge élémentaire e, ...

- Erreurs d'algorithme : En plus des erreurs d'entrée, l'algorithme lui même engendre des erreurs qui sont particulièrement dû aux opérations arithmétiques et à la précision de l'ordinateur.

- Erreurs de sortie : Elle sont engendrées par les erreurs précédentes et la précision de l'ordinateur.

Erreur d'entrée + Erreur d'algorithme $\longrightarrow$ Erreur de sortie

♦ Mesure de l'erreur :

Soit x une valeur exacte d'une quantité et $\overline{x}$ sa valeur approchée.
On peut envisager plusieurs façons pour mesurer l'erreur ϵ entre une valeur approchée $\overline{x}$ et une valeur exacte x.

→ L'erreur absolue :

On appelle erreur absolue de x la quantité $\epsilon_{abs} = \Delta x = x - \overline{x}$. Autrement, c'est la mesure de l'écart entre la valeur exacte et la valeur approchée.
Si $\Delta x \leq 0$ alors $\overline{x}$ est une valeur approchée par excès.

→ L'erreur relative

On appelle erreur relative de x la quantité $\epsilon_{rel} = \frac{\Delta x}{x} = \frac{x - \overline{x}}{x}$.

Erreur d'entrée + Erreur d'algorithme $\longrightarrow$ Erreur de sortie

♦ Mesure de l'erreur :

Soit x une valeur exacte d'une quantité et $\overline{x}$ sa valeur approchée.

On peut envisager plusieurs façons pour mesurer l'erreur ϵ entre une valeur approchée $\overline{x}$ et une valeur exacte x.

→ L'erreur absolue :

On appelle erreur absolue de x la quantité $\epsilon_{abs} = \Delta x = x - \overline{x}$. Autrement, c'est la mesure de l'écart entre la valeur exacte et la valeur approchée.
Si $\Delta x \leq 0$ alors $\overline{x}$ est une valeur approchée par excès.

→ L'erreur relative

On appelle erreur relative de x la quantité $\epsilon_{rel} = \frac{\Delta x}{x} = \frac{x-\overline{x}}{x}$.

Erreur d'entrée + Erreur d'algorithme $\longrightarrow$ Erreur de sortie

♦ Mesure de l'erreur :

Soit x une valeur exacte d'une quantité et $\overline{x}$ sa valeur approchée.
On peut envisager plusieurs façons pour mesurer l'erreur ϵ entre une valeur approchée $\overline{x}$ et une valeur exacte x.

→ L'erreur absolue :

On appelle erreur absolue de x la quantité $\epsilon_{abs} = \Delta x = x - \overline{x}$. Autrement, c'est la mesure de l'écart entre la valeur exacte et la valeur approchée.
Si $\Delta x \leq 0$ alors $\overline{x}$ est une valeur approchée par excès

→ L'erreur relative

On appelle erreur relative de x la quantité $\epsilon_{rel} = \frac{\Delta x}{x} = \frac{x-\overline{x}}{x}$.

Erreur d'entrée + Erreur d'algorithme $\longrightarrow$ Erreur de sortie

♦ Mesure de l'erreur :

Soit x une valeur exacte d'une quantité et $\overline{x}$ sa valeur approchée.
On peut envisager plusieurs façons pour mesurer l'erreur ϵ entre une valeur approchée $\overline{x}$ et une valeur exacte x.

→ L'erreur absolue :

On appelle erreur absolue de x la quantité $\epsilon_{abs} = \Delta x = x - \overline{x}$. Autrement, c'est la mesure de l'écart entre la valeur exacte et la valeur approchée.
Si $\Delta x \leq 0$ alors $\overline{x}$ est une valeur approchée par excès

→ L'erreur relative

On appelle erreur relative de x la quantité $\epsilon_{rel} = \frac{\Delta x}{x} = \frac{x-\overline{x}}{x}$.

Erreur d'entrée + Erreur d'algorithme $\longrightarrow$ Erreur de sortie

♦ Mesure de l'erreur :

Soit x une valeur exacte d'une quantité et $\overline{x}$ sa valeur approchée.
On peut envisager plusieurs façons pour mesurer l'erreur ϵ entre une valeur approchée $\overline{x}$ et une valeur exacte x.

→ L'erreur absolue :

On appelle erreur absolue de x la quantité $\epsilon_{abs} = \Delta x = x - \overline{x}$. Autrement, c'est la mesure de l'écart entre la valeur exacte et la valeur approchée.
Si $\Delta x \leq 0$ alors $\overline{x}$ est une valeur approchée par excès.

→ L'erreur relative

On appelle erreur relative de x la quantité ϵ_{rel} $\frac{\Delta x}{x} = \frac{x-\overline{x}}{x}$.

Erreur d'entrée + Erreur d'algorithme $\longrightarrow$ Erreur de sortie

♦ Mesure de l'erreur :

Soit x une valeur exacte d'une quantité et $\overline{x}$ sa valeur approchée.
On peut envisager plusieurs façons pour mesurer l'erreur ϵ entre une valeur approchée $\overline{x}$ et une valeur exacte x.

→ L'erreur absolue :

On appelle erreur absolue de x la quantité $\epsilon_{abs} = \Delta x = x - \overline{x}$. Autrement, c'est la mesure de l'écart entre la valeur exacte et la valeur approchée.
Si $\Delta x \leq 0$ alors $\overline{x}$ est une valeur approchée par excès.

→ L'erreur relative

On appelle erreur relative de x la quantité ϵ_{rel} $\frac{\Delta x}{x} = \frac{x-\overline{x}}{x}$.

Erreur d'entrée + Erreur d'algorithme $\longrightarrow$ Erreur de sortie

♦ Mesure de l'erreur :

Soit x une valeur exacte d'une quantité et $\overline{x}$ sa valeur approchée.
On peut envisager plusieurs façons pour mesurer l'erreur ϵ entre une valeur approchée $\overline{x}$ et une valeur exacte x.

→ L'erreur absolue :

On appelle erreur absolue de x la quantité $\epsilon_{abs} = \Delta x = x - \overline{x}$. Autrement, c'est la mesure de l'écart entre la valeur exacte et la valeur approchée.
Si $\Delta x \leq 0$ alors $\overline{x}$ est une valeur approchée par excès.

→ L'erreur relative

On appelle erreur relative de x la quantité $\epsilon_{rel} = \frac{\Delta x}{x} = \frac{x-\overline{x}}{x}$.

Remarque

Puisque la valeur de x est inconnu donc on peut jamais calculer de façon exacte Δx. En pratique, on cherche à majorer l'erreur absolue Δx en $\widehat{\Delta x}$ et estimer l'erreur relative en $\frac{\widehat{\Delta x}}{\widehat{x}}$, en général, la majoration de l'erreur absolue est donnée par le constructeur (tensiomètre, Thermomètre, balance, ...).

Généralités sur les matrices : Norme et conditionnement

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \bar{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \bar{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel.)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel.)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel.)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel.)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel.)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel.)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

• Quelques matrices particulières

1. **Matrice symétrique :** $A = (a_{ij})$, $A^t = A$ ou $A^t = (b_{ij})$, $b_{ij} = a_{ji}$.
2. **Matrice hermitienne :** $A^* = A$ où $A^* = (a^*_{ij})$, $a^*_{ij} = \overline{a}_{ji}$.
3. **Matrice symétrique positive et définie positive :** une matrice est dites semi-définie positive ssi $\langle Ax, x\rangle \geq 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$ (resp. $\langle Ax, x\rangle > 0$, $\forall x \in \mathbb{C}^n \setminus \{0\}$)
4. **Matrice normale :** ssi $AA^* = A^*A$.
5. **Matrice unitaire :** ssi $AA^* = A^*A$=I.
   (A est orthogonale = unitaire dans le cas réel.)
6. **Matrice diagonale :** ssi $a_{ij} = 0$, $\forall i \neq j$.
7. **Matrice triangulaire supérieur :** ssi $a_{ij} = 0$, $\forall i \geq j$.
8. **Matrice triangulaire inférieur :** ssi $a_{ij} = 0$, $\forall i \leq j$.

- **Normes matricielles :**

Definition

Étant donné E un espace vectoriel sur $\mathbb{R}$ ou $\mathbb{C}$, on appelle norme sur E notée $\|\cdot\|$, toute application de E à valeur dans $\mathbb{R}^+$ et qui vérifie :

1. $\|x\| = 0 \iff x = 0.$
2. $\|\lambda x\| = |\lambda|\|x\|, \quad \forall x \in E, \forall \lambda \in \mathbb{C}.$
3. $\|x + y\| \leq \|x\| + \|y\|, \quad \forall x, y \in E.$

Exemple

$$x \in \mathbb{C}^n, \quad \|x\|_p = \left(\sum_{k=1}^{n} |x_k|^p\right)^{\frac{1}{p}}, \qquad p \geq 1 \text{ (norme de Hölder).}$$

- $p = 2$ Norme euclidienne, elle correspond à la norme habituellement utilisée pour la distance entre deux points dans le plan ou l'espace.
- $p = 1$ est donnée par la somme des modules (ou valeurs absolues) des coefficients de x.

• Normes matricielles :

Definition

Étant donné E un espace vectoriel sur $\mathbb{R}$ ou $\mathbb{C}$, on appelle norme sur E notée $\|\cdot\|$, toute application de E à valeur dans $\mathbb{R}^+$ et qui vérifie :

1. $\|x\| = 0 \iff x = 0.$
2. $\|\lambda x\| = |\lambda|\|x\|, \quad \forall x \in E, \ \forall \lambda \in \mathbb{C}.$
3. $\|x + y\| \leq \|x\| + \|y\|, \quad \forall x, y \in E.$

Exemple

$$x \in \mathbb{C}^n, \quad \|x\|_p = \left(\sum_{k=1}^{n} |x_k|^p\right)^{\frac{1}{p}}, \qquad p \geq 1 \text{ (norme de Hölder).}$$

- $p = 2$ Norme euclidienne, elle correspond à la norme habituellement utilisée pour la distance entre deux points dans le plan ou l'espace.
- $p = 1$ est donnée par la somme des modules (ou valeurs absolues) des coefficients de x.

• Normes matricielles :

Definition

Étant donné E un espace vectoriel sur $\mathbb{R}$ ou $\mathbb{C}$, on appelle norme sur E notée $\|\cdot\|$, toute application de E à valeur dans $\mathbb{R}^+$ et qui vérifie :

1. $\|x\| = 0 \iff x = 0.$
2. $\|\lambda x\| = |\lambda|\|x\|, \quad \forall x \in E, \forall \lambda \in \mathbb{C}.$
3. $\|x + y\| \leq \|x\| + \|y\|, \quad \forall x, y \in E.$

Exemple

$$x \in \mathbb{C}^n, \quad \|x\|_p = \left(\sum_{k=1}^{n} |x_k|^p\right)^{\frac{1}{p}}, \qquad p \geq 1 \text{ (norme de Hölder).}$$

- $p = 2$ Norme euclidienne, elle correspond à la norme habituellement utilisée pour la distance entre deux points dans le plan ou l'espace.
- $p = 1$ est donnée par la somme des modules (ou valeurs absolues) des coefficients de x.

→ Normes matricielles induites

Soit E un K-espace vectoriel de dimension n muni d'une norme (vectorielle) notée $\|\cdot\|$.

$\|\cdot\|$ est une norme matricielle induite ou subordonnée si et seulement si $\|\cdot\|$ est une norme définie à partir d'une norme vectorielle par

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

(On a confondu les notations norme matricielle et norme vectorielle) qui vérifie

$$\begin{aligned} \|AB\| &\leq \|A\|\|B\|, \\ \forall x \in \mathrm{E}, \quad \|Ax\| &\leq \|A\|\|x\|. \end{aligned}$$

→ Normes matricielles induites

Soit E un K-espace vectoriel de dimension n muni d'une norme (vectorielle) notée $\|\cdot\|$.
$\|\cdot\|$ est une norme matricielle induite ou subordonnée si et seulement si $\|\cdot\|$ est une norme définie à partir d'une norme vectorielle par $\|A\| = \sup_{\|x\|=1} \|Ax\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$ (On a confondu les notations norme matricielle et norme vectorielle) qui vérifie

$$\begin{aligned} \|AB\| &\leq \|A\|\|B\|, \\ \forall x \in \mathrm{E}, \quad \|Ax\| &\leq \|A\|\|x\|. \end{aligned}$$

Proposition

Soit $A = (a_{ij})$ une matrice carrée. Alors

- $\|A\|_1 \equiv \sup_{x \neq 0} \dfrac{\|Ax\|_1}{\|x\|_1} = \max_j \sum_i |a_{ij}|.$
- $\|A\|_2 \equiv \sup_{x \neq 0} \dfrac{\|Ax\|_2}{\|x\|_2} = \sqrt{\rho(A^*A)} = \sqrt{\rho(AA^*)} = \|A^*\|_2$ *où* $\rho(A)$ *désigne le rayon spectral de A c-à-d* $\rho(A) = \max_i |\lambda_i(A)|.$
- $\|A\|_\infty \equiv \sup_{x \neq 0} \dfrac{\|Ax\|_\infty}{\|x\|_\infty} = \max_i \sum_j |a_{ij}|.$

Démonstration.

Exercice. □

Exemple

1. Calculer $\|A\|_\infty$, $\|A^t\|_\infty$, $\|A\|_1$ et $\|A^t\|_1$ où $A = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & -3 \\ 1 & -1 & -1 \end{pmatrix}$.
2. Calculer $\|A\|_\infty$ pour la matrice d'Hilbert suivante $\begin{pmatrix} 1 & \frac{1}{2} & \dots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \dots & \frac{1}{n+1} \\ \vdots & \vdots & \dots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \dots & \frac{1}{2n-1} \end{pmatrix}$.
3. Soit la matrice symétrique $A = \begin{pmatrix} 0 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 0 \end{pmatrix}$, calculer la $\|A\|_2$.
4. Soit la matrice non symétrique $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, comparer $\varrho(A)$ et $\|A\|_2$.

Proposition

1. *Si A est une matrice carrée d'ordre n de norme $\|\cdot\|$ (quelconque) alors $\rho(A) \leq \|A\|$ on a l'égalité dans le cas où A est symétrique hermitienne.*
2. *Étant données une matrice A et $\epsilon > 0$, il existe au moins une norme matricielle induite $\|\cdot\|_{A,\epsilon}$ telle que*

$$\|A\|_p \leq \rho(A) + \epsilon.$$

→ Convergence d'une suite de matrices

Definition

On note $A^k = \underbrace{AA\dots A}_{k \text{ fois}} = \left(a_{ij}^k\right)$ pour $A \in \mathcal{M}_n(\mathbb{C})$ et $k \in \mathbb{N}^*$, on écrit

$$\lim_{k\to+\infty} A^k = 0 \iff \lim_{k\to+\infty} a_{ij}^k = 0 \iff \lim_{k\to+\infty} \|A^k\| = 0$$

Theorème

Pour A une matrice carrée d'ordre n et $k \in \mathbb{N}^*$, les propriétés suivantes sont équivalentes :

1. $\lim_{k\to+\infty} A^k = 0$.
2. $\lim_{k\to+\infty} A^k x = 0,\ \forall x \in \mathbb{C}^n$.
3. $\rho(A) < 1$.
4. $\|A\|_p < 1$ pour au moins un $p \in \{1, 2, \dots, \infty\}$.

→ Convergence d'une suite de matrices

Definition

On note $A^k = \underbrace{AA\dots A}_{k \text{ fois}} = \left(a_{ij}^k\right)$ *pour* $A \in \mathcal{M}_n(\mathbb{C})$ *et* $k \in \mathbb{N}^*$, *on écrit*

$$\lim_{k\to+\infty} A^k = 0 \iff \lim_{k\to+\infty} a_{ij}^k = 0 \iff \lim_{k\to+\infty} \|A^k\| = 0$$

Theorème

Pour A une matrice carrée d'ordre n et $k \in \mathbb{N}^*$, *les propriétés suivantes sont équivalentes :*

1. $\lim_{k\to+\infty} A^k = 0$.
2. $\lim_{k\to+\infty} A^k x = 0,\ \forall x \in \mathbb{C}^n$.
3. $\rho(A) < 1$.
4. $\|A\|_p < 1$ *pour au moins un* $p \in \{1, 2, \dots, \infty\}$.

→ Convergence d'une suite de matrices

Definition

On note $A^k = \underbrace{AA\dots A}_{k \text{ fois}} = \left(a_{ij}^k\right)$ *pour* $A \in \mathcal{M}_n(\mathbb{C})$ *et* $k \in \mathbb{N}^*$, *on écrit*

$$\lim_{k\to+\infty} A^k = 0 \iff \lim_{k\to+\infty} a_{ij}^k = 0 \iff \lim_{k\to+\infty} \|A^k\| = 0$$

Theorème

Pour A une matrice carrée d'ordre n et $k \in \mathbb{N}^*$, *les propriétés suivantes sont équivalentes :*

1. $\lim_{k\to+\infty} A^k = 0.$
2. $\lim_{k\to+\infty} A^k x = 0,\ \forall x \in \mathbb{C}^n.$
3. $\rho(A) < 1.$
4. $\|A\|_p < 1$ *pour au moins un* $p \in \{1,\ 2,\ \dots,\ \infty\}.$

Démonstration.

(1) $\Longrightarrow$ (2) On a $\|A^k x\| \leq \|A^k\|\|x\|$, $\forall$ la norme et le résultat se déduit.

(2) $\Longrightarrow$ (3) Supposons que $\rho(A) \geq 1$, $\quad |\lambda| = \rho(A)$

$$\exists x \in \mathbb{C}^n,\ x \neq 0 \text{ tel que } Ax = \lambda x \quad \Longrightarrow \quad A^k x = \lambda^k x$$

$$A^k x \underset{k\to+\infty}{\longrightarrow} 0 \quad \Longrightarrow \quad \lambda^k x \underset{k\to+\infty}{\longrightarrow} 0$$

absurde car $\lambda^k \geq 1$.

(3) $\Longrightarrow$ (4) Soit $\epsilon = \frac{1-\rho(A)}{2}$ On sait qu'il existe au moins une norme matricielle telle que

$$\|A\|_p \leq \rho(A) + \epsilon < 1.$$

(4) $\Longrightarrow$ (1) Évidente car $\|A^k\| \leq \|A\|^k$.

□

On peut maintenant étudier quelques propriétés des séries de matrices :

Theorème

Si $A \in \mathcal{M}_n(\mathbb{C})$ tel que $\rho(A) < 1$ alors $I_n - A$ est inversible et

$$(I_n - A)^{-1} = \lim_{k \to +\infty} \sum_{i=0}^{k} A^i.$$

Démonstration.

On a

$\rho(A) < 1$, $\forall \lambda \in sp(A)$, alors $(1-\lambda) \in sp(I-A) \implies |\lambda| \leq \rho(A) < 1 \implies 1-\lambda \neq 0 \implies 0 \notin sp(I-A) \implies (I-A)$ est inversible.

De plus $(I-A)(I+A+\ldots A^k) = I - A^{k+1}$,

$$\implies I + A + \ldots A^k = (I-A)^{-1} - (I-A)^{-1} A^{k+1},$$

$$\implies \lim_{k \to +\infty} \sum_{i=0}^{k} A^i = (I-A)^{-1}, \quad \text{car } A^k x \underset{k \to +\infty}{\longrightarrow} 0.$$

□

On peut maintenant étudier quelques propriétés des séries de matrices :

Theorème

Si $A \in \mathcal{M}_n(\mathbb{C})$ tel que $\rho(A) < 1$ alors $I_n - A$ est inversible et

$$(I_n - A)^{-1} = \lim_{k \to +\infty} \sum_{i=0}^{k} A^i.$$

Démonstration.

On a

$\rho(A) < 1$, $\forall \lambda \in sp(A)$, alors $(1-\lambda) \in sp(I-A) \implies |\lambda| \leq \rho(A) < 1 \implies 1-\lambda \neq 0 \implies 0 \notin sp(I-A) \implies (I-A)$ est inversible.

De plus $(I-A)(I+A+\ldots A^k) = I - A^{k+1}$,

$\implies I + A + \ldots A^k = (I-A)^{-1} - (I-A)^{-1}A^{k+1}$,

$\implies \lim_{k \to +\infty} \sum_{i=0}^{k} A^i = (I-A)^{-1}$, car $A^k x \underset{k \to +\infty}{\longrightarrow} 0$.

□

On peut maintenant étudier quelques propriétés des séries de matrices :

Theorème

Si $A \in \mathcal{M}_n(\mathbb{C})$ tel que $\rho(A) < 1$ alors $I_n - A$ est inversible et

$$(I_n - A)^{-1} = \lim_{k \to +\infty} \sum_{i=0}^{k} A^i.$$

Démonstration.

On a

$\rho(A) < 1$, $\forall \lambda \in sp(A)$, alors $(1-\lambda) \in sp(I-A) \implies |\lambda| \leq \rho(A) < 1 \implies 1-\lambda \neq 0 \implies 0 \notin sp(I-A) \implies (I-A)$ est inversible.

De plus $(I-A)(I+A+\ldots A^k) = I - A^{k+1}$,

$\implies I + A + \ldots A^k = (I-A)^{-1} - (I-A)^{-1}A^{k+1}$,

$\implies \lim_{k \to +\infty} \sum_{i=0}^{k} A^i = (I-A)^{-1}$, car $A^k x \underset{k \to +\infty}{\longrightarrow} 0$.

□

Notions de conditionnement de matrice

Considérons le système linéaire suivant

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}, \text{ de solution } \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

On considère le système perturbé, où les seconds membres ont été "légèrement" modifiés, la matrice restant inchangée :

$$\begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \begin{pmatrix} x_1 + \delta x_1 \\ x_2 + \delta x_2 \\ x_3 + \delta x_3 \\ x_4 + \delta x_4 \end{pmatrix} = \begin{pmatrix} 32.1 \\ 22.9 \\ 33.1 \\ 30.9 \end{pmatrix}, \text{ de solution } \begin{pmatrix} 9.2 \\ -12.6 \\ 4.5 \\ -1.1 \end{pmatrix}$$

Autrement dit, une erreur relative de l'ordre de $\frac{1}{300}$ sur les données (ici les composantes du second membre) entraîne une erreur relative de 8.1736 sur le résultat, soit une multiplication de l'erreur relative par 2452.

Considérons également le système où, cette fois, ce sont les éléments de la matrice qui ont été "légèrement" modifiées :

$$\begin{pmatrix} 10 & 7 & 8.1 & 7.2 \\ 7.08 & 5.04 & 6 & 5 \\ 8 & 5.98 & 9.89 & 9 \\ 6.99 & 4.99 & 9 & 9.98 \end{pmatrix} \begin{pmatrix} x_1 + \delta x_1 \\ x_2 + \delta x_2 \\ x_3 + \delta x_3 \\ x_4 + \delta x_4 \end{pmatrix} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix},$$

$$\text{de solution} \begin{pmatrix} -81 \\ 137 \\ -34 \\ 22 \end{pmatrix}$$

Donc une petite perturbation sur les coefficients de la matrice ou sur le second membre entraîne une grosse perturbation sur la solution. C'est un problème de stabilité. A cause des erreurs de mesure ou d'arrondi qui sont inévitables on veut éviter le genre de situation présentée ci-dessus.
Un critère pour évaluer le degré d'instabilité du système est le nombre de conditionnement de la matrice : si ce nombre est petit la matrice est dite bien conditionnée et le système sera stable (c'est-à-dire peu sensible aux perturbations).

Definition

Soit $\|\cdot\|$ une norme matricielle et A une matrice inversible. Le nombre de conditionnement de A relativement à la norme matricielle choisie est

$$K(A) = cond(A) = \|A\|\|A^{-1}\|.$$

Lorsque $\|\cdot\|$ est la norme matricielle induite par la norme l_p, avec $1 \leq p \leq \pm\infty$, on note $cond_p(A)$ ou $K_p(A)$.

Donc une petite perturbation sur les coefficients de la matrice ou sur le second membre entraîne une grosse perturbation sur la solution. C'est un problème de stabilité. A cause des erreurs de mesure ou d'arrondi qui sont inévitables on veut éviter le genre de situation présentée ci-dessus.
Un critère pour évaluer le degré d'instabilité du système est le nombre de conditionnement de la matrice : si ce nombre est petit la matrice est dite bien conditionnée et le système sera stable (c'est-à-dire peu sensible aux perturbations).

Definition

Soit $\|\cdot\|$ une norme matricielle et A une matrice inversible. Le nombre de conditionnement de A relativement à la norme matricielle choisie est

$$K(A) = cond(A) = \|A\|\|A^{-1}\|.$$

Lorsque $\|\cdot\|$ est la norme matricielle induite par la norme l_p, avec $1 \leq p \leq \pm\infty$, on note $cond_p(A)$ ou $K_p(A)$.

→ Majoration des perturbations

On s'interesse maintenant à la résolution d'un système linéaire de la forme $Ax = b$ et à sa stabilité.

Theorème (Perturbation du second membre)

Soit x la solution de $Ax = b$ et $x + \delta x$ la solution de $A(x + \delta x) = b + \delta b$. Alors, si $\|\cdot\|$ est une norme matricielle induite

$$\frac{\|\delta x\|}{\|x\|} \leq K(A)\frac{\|\delta b\|}{\|b\|}.$$

Démonstration.

Laissé en exercice. □

→ Majoration des perturbations

On s'interesse maintenant à la résolution d'un système linéaire de la forme $Ax = b$ et à sa stabilité.

Theorème (Perturbation du second membre)

Soit x la solution de $Ax = b$ et $x + \delta x$ la solution de $A(x + \delta x) = b + \delta b$. Alors, si $\|\cdot\|$ est une norme matricielle induite

$$\frac{\|\delta x\|}{\|x\|} \leq K(A)\frac{\|\delta b\|}{\|b\|}.$$

Démonstration.

Laissé en exercice. □

→ Majoration des perturbations

On s'interesse maintenant à la résolution d'un système linéaire de la forme $Ax = b$ et à sa stabilité.

Theorème (Perturbation du second membre)

Soit x la solution de $Ax = b$ et $x + \delta x$ la solution de $A(x + \delta x) = b + \delta b$. Alors, si $\|\cdot\|$ est une norme matricielle induite

$$\frac{\|\delta x\|}{\|x\|} \leq K(A) \frac{\|\delta b\|}{\|b\|}.$$

Démonstration.

Laissé en exercice. □

Theorème (Perturbation du premier membre)

Soit x la solution de $Ax = b$ et $x + \delta x$ la solution de $(A + \delta A)(x + \delta x) = b$. Alors, si $\|\cdot\|$ est une norme matricielle induite

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq K(A) \frac{\|\delta A\|}{\|A\|}.$$

Démonstration.

Laissé en exercice. □

Donnons quelques propriétés immédiates de $cond(A)$.

Theorème

Soit A une matrice inversible à coefficients dans $\mathbb{K} = \mathbb{C}$, $\mathbb{R}$.

1. $K(\alpha A) = K(A), \quad \forall \alpha \neq 0.$
2. $K(A) \geq 1$ *si la norme matricielle est induite.*
3. $K_2(A) = \frac{\mu_{max}}{\mu_{min}}$ *où* μ_{max} *et* μ_{min} *sont les plus grande et petite valeurs singulières* *.
4. $K_2(A) = 1 \iff A = \alpha Q$ *où* $\alpha \in \mathbb{K}$ *et* Q *est unitaire.*

(* : On appelle valeurs singulières de M les racines carrées des valeurs propres de la matrice $M * M$)

Donnons quelques propriétés immédiates de $cond(A)$.

Theorème

Soit A une matrice inversible à coefficients dans $\mathbb{K} = \mathbb{C}, \mathbb{R}$.

1. $K(\alpha A) = K(A), \quad \forall \alpha \neq 0.$
2. $K(A) \geq 1$ *si la norme matricielle est induite.*
3. $K_2(A) = \frac{\mu_{max}}{\mu_{min}}$ *où* μ_{max} *et* μ_{min} *sont les plus grande et petite valeurs singulières* *.
4. $K_2(A) = 1 \iff A = \alpha Q$ *où* $\alpha \in \mathbb{K}$ *et* Q *est unitaire.*

(* : On appelle valeurs singulières de M les racines carrées des valeurs propres de la matrice $M * M$)

Démonstration.

1. Trivial.
2. On a $AA^{-1} = I \implies \|I\| \leq \|A\|\|A^{-1}\|$, pour la norme induite $\|I\| = \sup_{\|x\|=1} \|Ix\|$ on a $\|I\| = 1 \implies K(A) \geq 1$.
3. $K_2^2(A) = \|A\|_2^2\|A^{-1}\|_2^2 \stackrel{?}{=} \rho(A^*A)\rho((A^{-1})^*A^{-1})$?

$$\rho((A^{-1})^*A^{-1}) = \rho((A^*A)^{-1}) = \max\{|\lambda_i|,\ \lambda_i \text{ valeur propore de } (A^*A)$$
$$= \frac{1}{\min\{|\lambda_i|,\ \lambda_i \text{ v. p. de } A^*A\}}$$

Donc $K_2^2(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$, λ_i v. p. de A^*A,
$\implies K_2(A) = \frac{\mu_{max}}{\mu_{min}}$ où μ_i est une valeur singulière de A.

4. Soit A une matrice inversible. On peut trouver U et V unitaires et $D = diag(\mu_i)$ telles ques $A = UDV^*$.

Remarque

1. Une matrice est bien conditionnée si $K(A)$ est très proche de 1.
2. Une matrice unitaire est bien conditionnée.
3. Une matrice qui a un spectre large est mal conditionnée ($K(A) >> 1$).

Remarque

Si on change de norme matricielle, il existe des constantes K_1 et K_2 positives telles que

$$\forall A \in \mathcal{M}_n(\mathbb{R}), \quad K_1 cond_{\|\cdot\|}(A) \leq cond_{\|\cdot\|'}(A) \leq K_2 cond_{\|\cdot\|}(A).$$

La dernière propriété montre que si le conditionnement dans une norme est mauvais, on n'a que très peu de chance de l'améliorer en changeant de norme. Il faut recourir à d'autres méthodes pour améliorer le conditionnement d'une matrice. par exemple pré-multiplier A par une bonne matrice P dite de pré-conditionnement afin que $cond(PA)$ soit sensiblement plus petit que $cond(A)$. La recherche de telles matrices de pré-conditionnement ne peut se faire qu'au cas par cas et est de toute manière très difficile. Deux choix classiques sont de prendre pour P la matrice diagonale formée des termes diagonaux de A. ou. si $A = Id - B$ avec $\|B\| < 1$. de prendre pour P la matrice $Id + B$.

Exercice

On considère le système $A = \begin{pmatrix} 10 & 7 \\ 7 & -2 \end{pmatrix}$, $b = \begin{pmatrix} 17 \\ 7 \end{pmatrix}$.

1. Étudier la réponse du système à une perturbation du second membre $\delta b = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$.
2. Étudier la réponse du système à une perturbation du second membre $\delta A = \begin{pmatrix} 0 & 0.5 \\ -0.5 & 0 \end{pmatrix}$.
3. Calculer $Cond_2(A)$ et conclure.

Table de matière

Soit $A \in \mathcal{M}_n(\mathbb{R})$, le système $Ax = b$ admet une solution unique si une des conditions équivalentes suivantes est remplie :

1. A est inversible (i.e. $\det(A) \neq 0$).
2. le système $Ax = 0$ admet seulement la solution nulle.

La solution du système peut être calculée par la formule de CRAMER $x_k = \frac{\det(A_k)}{\det(A)}$. Évaluons le nombre d'opérations nécessaire au calcul pour une matrice pleine.

$\rightarrow$ Le calcul du $\det(A) = \sum_{j=1}^{n} a_{i,j}(-1)^{i+j} det(A_{i,j})$ nécessite à peut prêt n! additions et $n * n!$ multiplications. On a donc $n(n+1)!$ opérations à effectuer.

$\rightarrow$ Si on dispose d'un ordinateur qui effectue 2×10^{11} opérations par seconde (l'Intel Core i7-3770). Pour résoudre un système 50×50. il faudrait donc

$\frac{50 \times 51!}{2 \times 10^{11}}$ secondes, soit $\frac{50 \times 51!}{2 \times 10^{11} \times 3600 \times 24 \times 365} \simeq 1.225 \times 10^{49}$ années.

Soit $A \in \mathcal{M}_n(\mathbb{R})$, le système $Ax = b$ admet une solution unique si une des conditions équivalentes suivantes est remplie :

1. A est inversible (i.e. $\det(A) \neq 0$).
2. le système $Ax = 0$ admet seulement la solution nulle.

La solution du système peut être calculée par la formule de CRAMER $x_k = \frac{\det(A_k)}{\det(A)}$. Évaluons le nombre d'opérations nécessaire au calcul pour une matrice pleine.

$\rightarrow$ Le calcul du $\det(A) = \sum_{j=1}^{n} a_{i,j}(-1)^{i+j} det(A_{i,j})$ nécessite à peut prêt n!

additions et $n * n!$ multiplications. On a donc $n(n+1)!$ opérations à effectuer.

$\rightarrow$ Si on dispose d'un ordinateur qui effectue 2×10^{11} opérations par seconde (l'Intel Core i7-3770). Pour résoudre un système 50×50. il faudrait donc

$\frac{50 \times 51!}{2 \times 10^{11}}$ secondes, soit $\frac{50 \times 51!}{2 \times 10^{11} \times 3600 \times 24 \times 365} \simeq 1.225 \times 10^{49}$ années.

Soit $A \in \mathcal{M}_n(\mathbb{R})$, le système $Ax = b$ admet une solution unique si une des conditions équivalentes suivantes est remplie :

1. A est inversible (i.e. $\det(A) \neq 0$).
2. le système $Ax = 0$ admet seulement la solution nulle.

La solution du système peut être calculée par la formule de CRAMER $x_k = \frac{\det(A_k)}{\det(A)}$. Évaluons le nombre d'opérations nécessaire au calcul pour une matrice pleine.

$\rightarrow$ Le calcul du $\det(A) = \sum_{j=1}^{n} a_{i,j}(-1)^{i+j} det(A_{i,j})$ nécessite à peut prêt n!

additions et $n * n!$ multiplications. On a donc $n(n+1)!$ opérations à effectuer.

$\rightarrow$ Si on dispose d'un ordinateur qui effectue 2×10^{11} opérations par seconde (l'Intel Core i7-3770). Pour résoudre un système 50×50. il faudrait donc

$\frac{50 \times 51!}{2 \times 10^{11}}$ secondes, soit $\frac{50 \times 51!}{2 \times 10^{11} \times 3600 \times 24 \times 365} \simeq 1.225 \times 10^{49}$ années.

Soit $A \in \mathcal{M}_n(\mathbb{R})$, le système $Ax = b$ admet une solution unique si une des conditions équivalentes suivantes est remplie :

1. A est inversible (i.e. $\det(A) \neq 0$).
2. le système $Ax = 0$ admet seulement la solution nulle.

La solution du système peut être calculée par la formule de CRAMER $x_k = \frac{\det(A_k)}{\det(A)}$. Évaluons le nombre d'opérations nécessaire au calcul pour une matrice pleine.

$\rightarrow$ Le calcul du $\det(A) = \sum_{j=1}^{n} a_{i,j}(-1)^{i+j} det(A_{i,j})$ nécessite à peut prêt n! additions et $n * n!$ multiplications. On a donc $n(n+1)!$ opérations à effectuer.

$\rightarrow$ Si on dispose d'un ordinateur qui effectue 2×10^{11} opérations par seconde (l'Intel Core i7-3770). Pour résoudre un système 50×50. il faudrait donc

$\frac{50 \times 51!}{2 \times 10^{11}}$ secondes, soit $\frac{50 * 51!}{2 \times 10^{11} \times 3600 \times 24 \times 365} \simeq 1.225 \times 10^{49}$ années.

Soit $A \in \mathcal{M}_n(\mathbb{R})$, le système $Ax = b$ admet une solution unique si une des conditions équivalentes suivantes est remplie :

1. A est inversible (i.e. $\det(A) \neq 0$).
2. le système $Ax = 0$ admet seulement la solution nulle.

La solution du système peut être calculée par la formule de CRAMER $x_k = \frac{\det(A_k)}{\det(A)}$. Évaluons le nombre d'opérations nécessaire au calcul pour une matrice pleine.

$\rightarrow$ Le calcul du $\det(A) = \sum_{j=1}^{n} a_{i,j}(-1)^{i+j} det(A_{i,j})$ nécessite à peut prêt n! additions et $n * n!$ multiplications. On a donc $n(n+1)!$ opérations à effectuer.

$\rightarrow$ Si on dispose d'un ordinateur qui effectue 2×10^{11} opérations par seconde (l'Intel Core i7-3770). Pour résoudre un système 50×50. il faudrait donc

$$\frac{50 \times 51!}{2 \times 10^{11}} \text{ secondes, soit } \frac{50 * 51!}{2 \times 10^{11} \times 3600 \times 24 \times 365} \simeq 1.225 \times 10^{49} \text{ années.}$$

Soit $A \in \mathcal{M}_n(\mathbb{R})$, le système $Ax = b$ admet une solution unique si une des conditions équivalentes suivantes est remplie :

1. A est inversible (i.e. $\det(A) \neq 0$).
2. le système $Ax = 0$ admet seulement la solution nulle.

La solution du système peut être calculée par la formule de CRAMER $x_k = \frac{\det(A_k)}{\det(A)}$. Évaluons le nombre d'opérations nécessaire au calcul pour une matrice pleine.

$\rightarrow$ Le calcul du $\det(A) = \sum_{j=1}^{n} a_{i,j}(-1)^{i+j} det(A_{i,j})$ nécessite à peut prêt n! additions et $n * n!$ multiplications. On a donc $n(n+1)!$ opérations à effectuer.

$\rightarrow$ Si on dispose d'un ordinateur qui effectue 2×10^{11} opérations par seconde (l'Intel Core i7-3770). Pour résoudre un système 50×50. il faudrait donc

$$\frac{50 \times 51!}{2 \times 10^{11}} \text{ secondes, soit } \frac{50 * 51!}{2 \times 10^{11} \times 3600 \times 24 \times 365} \simeq 1.225 \times 10^{49} \text{ années.}$$

- Pour cette raison, des méthodes numériques alternatives aux formules de CRAMER ont été développées.

- Elles sont dites *directes* si elles fournissent la solution du système en un nombre fini d'étapes, *itératives* si elles nécessitent (théoriquement) un nombre infini d'étapes.

- Notons dès à présent que le choix entre une méthode directe et une méthode itérative pour la résolution d'un système dépend non seulement de l'efficacité théorique des algorithmes, mais aussi du type de matrice, des capacités de stockage en mémoire et enfin de l'architecture de l'ordinateur.

- Pour cette raison, des méthodes numériques alternatives aux formules de CRAMER ont été développées.
- Elles sont dites *directes* si elles fournissent la solution du système en un nombre fini d'étapes, *itératives* si elles nécessitent (théoriquement) un nombre infini d'étapes.
- Notons dès à présent que le choix entre une méthode directe et une méthode itérative pour la résolution d'un système dépend non seulement de l'efficacité théorique des algorithmes, mais aussi du type de matrice, des capacités de stockage en mémoire et enfin de l'architecture de l'ordinateur.

- Pour cette raison, des méthodes numériques alternatives aux formules de CRAMER ont été développées.
- Elles sont dites *directes* si elles fournissent la solution du système en un nombre fini d'étapes, *itératives* si elles nécessitent (théoriquement) un nombre infini d'étapes.
- Notons dès à présent que le choix entre une méthode directe et une méthode itérative pour la résolution d'un système dépend non seulement de l'efficacité théorique des algorithmes, mais aussi du type de matrice, des capacités de stockage en mémoire et enfin de l'architecture de l'ordinateur.

Méthodes directes de résolution de $Ax = b$

La résolution de $Ax = b$ est très simple dans plusieurs cas :

- si A est diagonale.
- si A est unitaire : si $A^t A = AA^t = I$, alors $A^{-1} = A^t$.
- si A est triangulaire supérieure ou inférieure.

La résolution de $Ax = b$ est très simple dans plusieurs cas :

- si A est diagonale.
- si A est unitaire : si $A^t A = AA^t = I$, alors $A^{-1} = A^t$.
- si A est triangulaire supérieure ou inférieure.

La résolution de $Ax = b$ est très simple dans plusieurs cas :

- si A est diagonale.
- si A est unitaire : si $A^tA = AA^t = I$, alors $A^{-1} = A^t$.
- si A est triangulaire supérieure ou inférieure.

La résolution de $Ax = b$ est très simple dans plusieurs cas :

- si A est diagonale.
- si A est unitaire : si $A^t A = AA^t = I$, alors $A^{-1} = A^t$.
- si A est triangulaire supérieure ou inférieure.

La résolution de $Ax = b$ est très simple dans plusieurs cas :

- si A est diagonale.
- si A est unitaire : si $A^t A = AA^t = I$, alors $A^{-1} = A^t$.
- si A est triangulaire supérieure ou inférieure.

- **Résolution d'un système triangulaire (supérieur)**

Definition (Matrices et systèmes triangulaires)

On dit qu'une matrice carrée $A = (a_{ij})$ est triangulaire supérieure (respectivement triangulaire inférieure) si $i > j \Longrightarrow a_{ij} = 0$ (resp. si $i < j \Longrightarrow a_{ij} = 0$).
Si la matrice est triangulaire supérieure (resp. triangulaire inférieure), on dira que le système linéaire est un système triangulaire supérieur (resp. triangulaire inférieur).

- **Résolution d'un système triangulaire (supérieur)**

Definition (Matrices et systèmes triangulaires)

On dit qu'une matrice carrée $A = (a_{ij})$ est triangulaire supérieure (respectivement triangulaire inférieure) si $i > j \Longrightarrow a_{ij} = 0$ (resp. si $i < j \Longrightarrow a_{ij} = 0$).
Si la matrice est triangulaire supérieure (resp. triangulaire inférieure), on dira que le système linéaire est un système triangulaire supérieur (resp. triangulaire inférieur).

Soit, donc le système linéaire suivant :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ \quad\quad a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ \quad\quad\quad \ddots \quad\quad \vdots \quad\quad\quad \vdots \quad\quad \vdots \\ \quad\quad\quad\quad\quad a_{kk}x_k + \cdots + a_{kn}x_n = b_k \\ \quad\quad\quad\quad\quad\quad \ddots \quad\quad \vdots \quad\quad \vdots \\ \quad\quad\quad\quad\quad\quad\quad\quad a_{nn}x_n = b_n \end{cases}$$

On remarque que le déterminant de la matrice A définie par

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk} & \cdots & a_{kn} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & a_{nn} \end{pmatrix},$$

est non nul et égal à $\prod_{i=1}^{n} a_{ii}$, donc $\forall i,\ a_{ii} \neq 0$.

Soit, donc le système linéaire suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + & \cdots + & a_{1n}x_n & = b_1 \\ & a_{22}x_2 + & a_{23}x_3 + & \cdots + & a_{2n}x_n & = b_2 \\ & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & a_{kk}x_k + & \cdots + & a_{kn}x_n & = b_k \\ & & \ddots & \cdots & \vdots & \vdots \\ & & & & a_{nn}x_n & = b_n \end{cases}$$

On remarque que le déterminant de la matrice A définie par

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk} & \cdots & a_{kn} \\ \vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

est non nul et égal à $\prod_{i=1}^{n} a_{ii}$, donc $\forall i,\ a_{ii} \neq 0$.

Soit, donc le système linéaire suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + & \cdots + & a_{1n}x_n & = b_1 \\ & a_{22}x_2 + & a_{23}x_3 + & \cdots + & a_{2n}x_n & = b_2 \\ & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & a_{kk}x_k + & \cdots + & a_{kn}x_n & = b_k \\ & & \ddots & \cdots & \vdots & \vdots \\ & & & & a_{nn}x_n & = b_n \end{cases}$$

On remarque que le déterminant de la matrice A définie par

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk} & \cdots & a_{kn} \\ \vdots & \ddots & \vdots & \ddots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

est non nul et égal à $\prod_{i=1}^{n} a_{ii}$, donc $\forall i,\ a_{ii} \neq 0$.

Soit, donc le système linéaire suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + & \cdots + & a_{1n}x_n & = b_1 \\ & a_{22}x_2 + & a_{23}x_3 + & \cdots + & a_{2n}x_n & = b_2 \\ & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & a_{kk}x_k + & \cdots + & a_{kn}x_n & = b_k \\ & & \ddots & \cdots & \vdots & \vdots \\ & & & & a_{nn}x_n & = b_n \end{cases}$$

On remarque que le déterminant de la matrice A définie par

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & \cdots & a_{2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk} & \cdots & a_{kn} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & a_{nn} \end{pmatrix},$$

est non nul et égal à $\prod_{i=1}^{n} a_{ii}$, donc $\forall i,\ a_{ii} \neq 0$.

L'algorithme de résolution de ce système est le suivant

Algorithme

- $x_n = \frac{b_n}{a_{nn}}$.
- *Pour* $k = n-1, \ldots, 1$, $\quad x_k = \frac{b_k}{a_{kk}} - \frac{\sum_{i=k+1}^{n} a_{ki} x_i}{a_{kk}}$.

Comptons le nombre d'opérations nécessaires pour résoudre ce système :

1. A chaque étape k : 1 division et $n-k$ multiplications, soit $n-k+1$ multiplications au sens large.
2. A chaque étape k : $n-k$ additions si $k = 1, \ldots, n-1$.

Au total,

$$\sum_{k=1}^{n}(n-k+1) + \sum_{k=1}^{n-1}(n-k).$$

opérations, c'est-à-dire $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} = n^2$.

L'algorithme de résolution de ce système est le suivant

Algorithme

- $x_n = \frac{b_n}{a_{nn}}$.
- Pour $k = n-1, \ldots, 1$, $\quad x_k = \frac{b_k}{a_{kk}} - \frac{\sum_{i=k+1}^{n} a_{ki} x_i}{a_{kk}}$.

Comptons le nombre d'opérations nécessaires pour résoudre ce système :

1. A chaque étape k : 1 division et $n-k$ multiplications, soit $n-k+1$ multiplications au sens large.
2. A chaque étape k : $n-k$ additions si $k \leq n-1$.

Au total,

$$\sum_{k=1}^{n}(n-k+1) + \sum_{k=1}^{n-1}(n-k).$$

opérations, c'est-à-dire $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} = n^2$.

L'algorithme de résolution de ce système est le suivant

Algorithme

- $x_n = \frac{b_n}{a_{nn}}$.
- Pour $k = n-1, \ldots, 1$, $\quad x_k = \frac{b_k}{a_{kk}} - \frac{\sum_{i=k+1}^{n} a_{ki} x_i}{a_{kk}}$.

Comptons le nombre d'opérations nécessaires pour résoudre ce système :

1. A chaque étape k : 1 division et $n-k$ multiplications, soit $n-k+1$ multiplications au sens large.
2. A chaque étape k : $n-k$ additions si $k \leq n-1$.

Au total,

$$\sum_{k=1}^{n}(n-k+1) + \sum_{k=1}^{n-1}(n-k).$$

opérations, c'est-à-dire $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} = n^2$.

L'algorithme de résolution de ce système est le suivant

Algorithme

- $x_n = \frac{b_n}{a_{nn}}$.
- *Pour* $k = n-1, \ldots, 1$, $\quad x_k = \frac{b_k}{a_{kk}} - \frac{\sum_{i=k+1}^{n} a_{ki} x_i}{a_{kk}}$.

Comptons le nombre d'opérations nécessaires pour résoudre ce système :

1. A chaque étape k : 1 division et $n-k$ multiplications, soit $n-k+1$ multiplications au sens large.
2. A chaque étape k : $n-k$ additions si $k \leq n-1$.

Au total,

$$\sum_{k=1}^{n}(n-k+1) + \sum_{k=1}^{n-1}(n-k).$$

opérations, c'est-à-dire $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} = n^2$.

L'algorithme de résolution de ce système est le suivant

Algorithme

- $x_n = \frac{b_n}{a_{nn}}$.
- *Pour* $k = n-1, \ldots, 1$, $\quad x_k = \frac{b_k}{a_{kk}} - \frac{\sum_{i=k+1}^{n} a_{ki} x_i}{a_{kk}}$.

Comptons le nombre d'opérations nécessaires pour résoudre ce système :

1. A chaque étape k : 1 division et $n-k$ multiplications, soit $n-k+1$ multiplications au sens large.
2. A chaque étape k : $n-k$ additions si $k \leq n-1$.

Au total,

$$\sum_{k=1}^{n}(n-k+1) + \sum_{k=1}^{n-1}(n-k).$$

opérations, c'est-à-dire $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} = n^2$.

L'algorithme de résolution de ce système est le suivant

Algorithme

- $x_n = \frac{b_n}{a_{nn}}$.
- *Pour* $k = n-1, \ldots, 1$, $\quad x_k = \frac{b_k}{a_{kk}} - \frac{\sum_{i=k+1}^{n} a_{ki} x_i}{a_{kk}}$.

Comptons le nombre d'opérations nécessaires pour résoudre ce système :

1. A chaque étape k : 1 division et $n-k$ multiplications, soit $n-k+1$ multiplications au sens large.
2. A chaque étape k : $n-k$ additions si $k \leq n-1$.

Au total,

$$\sum_{k=1}^{n}(n-k+1) + \sum_{k=1}^{n-1}(n-k).$$

opérations, c'est-à-dire $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} = n^2$.

La méthode de Gauss

On considère maintenant le système « plein » suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2n}x_n & = b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{k1}x_1 + & a_{k2}x_2 + & \cdots + & a_{kn}x_n & = b_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}x_1 + & a_{n2}x_2 + & \cdots + & a_{nn}x_n & = b_n \end{cases}$$

La méthode du pivot de GAUSS transforme le système $Ax = b$ en un système équivalent (ç-à-d ayant la même solution) de la forme $Ux = y$, où U est une matrice triangulaire supérieure et y est un second membre convenablement modifié. Enfin on résout le système triangulaire $Ux = y$.

La méthode de Gauss

On considère maintenant le système « plein » suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2n}x_n & = b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{k1}x_1 + & a_{k2}x_2 + & \cdots + & a_{kn}x_n & = b_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}x_1 + & a_{n2}x_2 + & \cdots + & a_{nn}x_n & = b_n \end{cases}$$

La méthode du pivot de GAUSS transforme le système $Ax = b$ en un système équivalent (ç-à-d ayant la même solution) de la forme $Ux = y$, où U est une matrice triangulaire supérieure et y est un second membre convenablement modifié. Enfin on résout le système triangulaire $Ux = y$.

La méthode de Gauss

On considère maintenant le système « plein » suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2n}x_n & = b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{k1}x_1 + & a_{k2}x_2 + & \cdots + & a_{kn}x_n & = b_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}x_1 + & a_{n2}x_2 + & \cdots + & a_{nn}x_n & = b_n \end{cases}$$

La méthode du pivot de GAUSS transforme le système $Ax = b$ en un système équivalent (ç-à-d ayant la même solution) de la forme $Ux = y$, où U est une matrice triangulaire supérieure et y est un second membre convenablement modifié. Enfin on résout le système triangulaire $Ux = y$.

La méthode de Gauss

On considère maintenant le système « plein » suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2n}x_n & = b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{k1}x_1 + & a_{k2}x_2 + & \cdots + & a_{kn}x_n & = b_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1}x_1 + & a_{n2}x_2 + & \cdots + & a_{nn}x_n & = b_n \end{cases}$$

La méthode du pivot de GAUSS transforme le système $Ax = b$ en un système équivalent (ç-à-d ayant la même solution) de la forme $Ux = y$, où U est une matrice triangulaire supérieure et y est un second membre convenablement modifié. Enfin on résout le système triangulaire $Ux = y$.

Le principe de la méthode est le suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 & \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2n}x_n & = b_2 & L_2^{(1)} \leftarrow L_2 - L_1 * \frac{a_{21}}{a_{11}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ a_{k1}x_1 + & a_{k2}x_2 + & \cdots + & a_{kn}x_n & = b_k & L_k^{(1)} \leftarrow L_k - L_1 * \frac{a_{k1}}{a_{11}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ a_{n1}x_1 + & a_{n2}x_2 + & \cdots + & a_{nn}x_n & = b_n & L_n^{(1)} \leftarrow L_n - L_1 * \frac{a_{n1}}{a_{11}} \end{cases}$$

donne

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 \\ 0 & +a_{22}^{(1)}x_2 + & \cdots + & a_{2n}^{(1)}x_n & = b_2^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & +a_{k2}^{(1)}x_2 + & \cdots + & a_{kn}^{(1)}x_n & = b_k^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & +a_{n2}^{(1)}x_2 + & \cdots + & a_{nn}^{(1)}x_n & = b_n^{(1)} \end{cases}$$

avec $\forall j, k = 2, ..., n \quad a_{kj}^{(1)} = a_{kj} - a_{1j} * \frac{a_{k1}}{a_{11}}$ et $b_k^{(1)} = b_k - b_1 * \frac{a_{k1}}{a_{11}}$.

Le principe de la méthode est le suivant :

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 & \\ a_{21}x_1 + & a_{22}x_2 + & \cdots + & a_{2n}x_n & = b_2 & \mathrm{L}_2^{(1)} \leftarrow \mathrm{L}_2 - \mathrm{L}_1 * \frac{a_{21}}{a_{11}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ a_{k1}x_1 + & a_{k2}x_2 + & \cdots + & a_{kn}x_n & = b_k & \mathrm{L}_k^{(1)} \leftarrow \mathrm{L}_k - \mathrm{L}_1 * \frac{a_{k1}}{a_{11}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ a_{n1}x_1 + & a_{n2}x_2 + & \cdots + & a_{nn}x_n & = b_n & \mathrm{L}_n^{(1)} \leftarrow \mathrm{L}_n - \mathrm{L}_1 * \frac{a_{n1}}{a_{11}} \end{cases}$$

donne

$$\begin{cases} a_{11}x_1 & + a_{12}x_2 + & \cdots + & a_{1n}x_n & = b_1 \\ 0 & + a_{22}^{(1)}x_2 + & \cdots + & a_{2n}^{(1)}x_n & = b_2^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & + a_{k2}^{(1)}x_2 + & \cdots + & a_{kn}^{(1)}x_n & = b_k^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & + a_{n2}^{(1)}x_2 + & \cdots + & a_{nn}^{(1)}x_n & = b_n^{(1)} \end{cases}$$

avec $\forall j, k = 2, ..., n \quad a_{kj}^{(1)} = a_{kj} - a_{1j} * \frac{a_{k1}}{a_{11}}$ et $b_k^{(1)} = b_k - b_1 * \frac{a_{k1}}{a_{11}}$.

On recommence ainsi de proche en proche : à l'étape k on suppose que $a_{kk}^{(k)}$ est non nul (quitte à permuter avec les lignes en dessous). Si on ne peut pas trouver de coefficient diagonal non nul à ce stade c'est que la matrice A n'est pas inversible. Détaillons l'étape k

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix} \quad \text{Ligne pivot numéro } k : \mathrm{L}_k^{(k)}.$$

On recommence ainsi de proche en proche : à l'étape k on suppose que $a_{kk}^{(k)}$ est non nul (quitte à permuter avec les lignes en dessous). Si on ne peut pas trouver de coefficient diagonal non nul à ce stade c'est que la matrice A n'est pas inversible. Détaillons l'étape k

$$\left(\begin{array}{cccccc} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{array}\right.$$ Ligne pivot numéro k : $\mathrm{L}_k^{(k)}$ $\left.\right)$.

On effectue les opérations suivantes sur les lignes se trouvant en dessous de L_k :

$$L_{k+1} \leftarrow L_{k+1} - L_k * \frac{a^{(k)}_{(k+1)1}}{a^{(k)}_{kk}}$$

$$\vdots$$

$$L_n \leftarrow L_n - L_k * \frac{a^{(k)}_{n1}}{a^{(k)}_{kk}}$$

de sorte que la matrice A_k devienne

$$\begin{pmatrix} a^{(1)}_{11} & a^{(1)}_{12} & a^{(1)}_{13} & \cdots & \cdots & a^{(1)}_{1n} \\ 0 & a^{(2)}_{22} & a^{(2)}_{23} & \cdots & \cdots & a^{(2)}_{2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a^{(k)}_{kk} & \cdots & a^{(k)}_{kn} \\ \vdots & \ddots & \vdots & 0 & a^{(k+1)}_{(k+1)(k+1)} & \vdots \\ \vdots & \ddots & \vdots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a^{(n)}_{nn} \end{pmatrix}.$$

Au bout de toutes ces manipulations on obtient une matrice triangulaire supérieure U.

On effectue les opérations suivantes sur les lignes se trouvant en dessous de L_k :

$$L_{k+1} \leftarrow L_{k+1} - L_k * \frac{a^{(k)}_{(k+1)1}}{a^{(k)}_{kk}}$$

$$\vdots$$

$$L_n \leftarrow L_n - L_k * \frac{a^{(k)}_{n1}}{a^{(k)}_{kk}}$$

de sorte que la matrice A_k devienne

$$\begin{pmatrix} a^{(1)}_{11} & a^{(1)}_{12} & a^{(1)}_{13} & \cdots & \cdots & a^{(1)}_{1n} \\ 0 & a^{(2)}_{22} & a^{(2)}_{23} & \cdots & \cdots & a^{(2)}_{2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a^{(k)}_{kk} & \cdots & a^{(k)}_{kn} \\ \vdots & \ddots & \vdots & 0 & a^{(k+1)}_{(k+1)(k+1)} & \vdots \\ \vdots & \ddots & \vdots & 0 & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a^{(n)}_{nn} \end{pmatrix}.$$

Au bout de toutes ces manipulations on obtient une matrice triangulaire supérieure U.

On effectue les opérations suivantes sur les lignes se trouvant en dessous de L_k :

$$L_{k+1} \leftarrow L_{k+1} - L_k * \frac{a^{(k)}_{(k+1)\mathbf{1}}}{a^{(k)}_{kk}}$$

$$\vdots$$

$$L_n \leftarrow L_n - L_k * \frac{a^{(k)}_{n\mathbf{1}}}{a^{(k)}_{kk}}$$

de sorte que la matrice A_k devienne

$$\begin{pmatrix} a^{(1)}_{11} & a^{(1)}_{12} & a^{(1)}_{13} & \cdots & \cdots & a^{(1)}_{1n} \\ 0 & a^{(2)}_{22} & a^{(2)}_{23} & \cdots & \cdots & a^{(2)}_{2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a^{(k)}_{kk} & \cdots & a^{(k)}_{kn} \\ \vdots & \ddots & \vdots & 0 & a^{(k+1)}_{(k+1)(k+1)} & \vdots \\ \vdots & \ddots & \vdots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a^{(n)}_{nn} \end{pmatrix}.$$

Au bout de toutes ces manipulations on obtient une matrice triangulaire supérieure U.

On effectue les opérations suivantes sur les lignes se trouvant en dessous de L_k :

$$L_{k+1} \leftarrow L_{k+1} - L_k * \frac{a^{(k)}_{(k+1)1}}{a^{(k)}_{kk}}$$

$$\vdots$$

$$L_n \leftarrow L_n - L_k * \frac{a^{(k)}_{n1}}{a^{(k)}_{kk}}$$

de sorte que la matrice A_k devienne

$$\begin{pmatrix} a^{(1)}_{11} & a^{(1)}_{12} & a^{(1)}_{13} & \cdots & \cdots & a^{(1)}_{1n} \\ 0 & a^{(2)}_{22} & a^{(2)}_{23} & \cdots & \cdots & a^{(2)}_{2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a^{(k)}_{kk} & \cdots & a^{(k)}_{kn} \\ \vdots & \ddots & \vdots & 0 & a^{(k+1)}_{(k+1)(k+1)} & \vdots \\ \vdots & \ddots & \vdots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a^{(n)}_{nn} \end{pmatrix}.$$

Au bout de toutes ces manipulations on obtient une matrice triangulaire supérieure U.

Exemple

Soit le système linéaire

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases},$$

Résolution par la méthode du pivot de GAUSS

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases} \begin{matrix} \\ L_2 \leftarrow L_2 - 2L_1 \\ L_3 \leftarrow L_3 - 3L_1 \\ L_4 \leftarrow L_4 - 4L_1 \end{matrix} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 - 2x_2 - 8x_3 - 10x_4 &= 0 \\ 0 - 7x_2 - 10x_3 - 13x_4 &= 0 \end{cases} \begin{matrix} \\ \\ L_3 \leftarrow L_3 - 2L_2 \\ L_4 \leftarrow L_4 - 7L_2 \end{matrix}$$

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 4x_3 + 36x_4 &= 0 \end{cases} \begin{matrix} \\ \\ \\ L_4 \leftarrow L_4 + L_3 \end{matrix} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 0 + 40x_4 &= 0 \end{cases}$$

Donc $x_4 = 0$, $x_3 = 0$, $x_2 = 0$, $x_1 = 1$,

Exemple

Soit le système linéaire

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases},$$

Résolution par la méthode du pivot de GAUSS

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases} \begin{matrix} \\ L_2 \leftarrow L_2 - 2L_1 \\ L_3 \leftarrow L_3 - 3L_1 \\ L_4 \leftarrow L_4 - 4L_1 \end{matrix} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 - 2x_2 - 8x_3 - 10x_4 &= 0 \\ 0 - 7x_2 - 10x_3 - 13x_4 &= 0 \end{cases} \begin{matrix} \\ \\ L_3 \leftarrow L_3 - 2L_2 \\ L_4 \leftarrow L_4 - 7L_2 \end{matrix}$$

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 4x_3 + 36x_4 &= 0 \end{cases} \begin{matrix} \\ \\ \\ L_4 \leftarrow L_4 + L_3 \end{matrix} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 0 + 40x_4 &= 0 \end{cases}$$

Donc $x_4 = 0$, $x_3 = 0$, $x_2 = 0$, $x_1 = 1$,

Exemple

Soit le système linéaire

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases},$$

Résolution par la méthode du pivot de GAUSS

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases} \begin{array}{l} \\ L_2 \leftarrow L_2 - 2L_1 \\ L_3 \leftarrow L_3 - 3L_1 \\ L_4 \leftarrow L_4 - 4L_1 \end{array} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 - 2x_2 - 8x_3 - 10x_4 &= 0 \\ 0 - 7x_2 - 10x_3 - 13x_4 &= 0 \end{cases} \begin{array}{l} \\ \\ \\ L_3 \leftarrow L_3 - 2L_2 \\ L_4 \leftarrow L_4 - 7L_2 \end{array}$$

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 4x_3 + 36x_4 &= 0 \end{cases} \begin{array}{l} \\ \\ \\ L_4 \leftarrow L_4 + L_3 \end{array} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 0 + 40x_4 &= 0 \end{cases}$$

Donc $x_4 = 0$, $x_3 = 0$, $x_2 = 0$, $x_1 = 1$,

Exemple

Soit le système linéaire

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 = 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 4 \end{cases},$$

Résolution par la méthode du pivot de GAUSS

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 = 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 4 \end{cases} \begin{array}{l} \\ L_2 \leftarrow L_2 - 2L_1 \\ L_3 \leftarrow L_3 - 3L_1 \\ L_4 \leftarrow L_4 - 4L_1 \end{array} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 0 - x_2 - 2x_3 - 7x_4 = 0 \\ 0 - 2x_2 - 8x_3 - 10x_4 = 0 \\ 0 - 7x_2 - 10x_3 - 13x_4 = 0 \end{cases} \begin{array}{l} \\ \\ L_3 \leftarrow L_3 - 2L_2 \\ L_4 \leftarrow L_4 - 7L_2 \end{array}$$

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 0 - x_2 - 2x_3 - 7x_4 = 0 \\ 0 + 0 - 4x_3 + 4x_4 = 0 \\ 0 + 0 + 4x_3 + 36x_4 = 0 \end{cases} \quad L_4 \leftarrow L_4 + L_3 \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 0 - x_2 - 2x_3 - 7x_4 = 0 \\ 0 + 0 - 4x_3 + 4x_4 = 0 \\ 0 + 0 + 0 + 40x_4 = 0 \end{cases}$$

Donc $x_4 = 0,\ x_3 = 0,\ x_2 = 0,\ x_1 = 1,$

Exemple

Soit le système linéaire

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases},$$

Résolution par la méthode du pivot de GAUSS

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 &= 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 &= 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 &= 4 \end{cases} \begin{matrix} \\ L_2 \leftarrow L_2 - 2L_1 \\ L_3 \leftarrow L_3 - 3L_1 \\ L_4 \leftarrow L_4 - 4L_1 \end{matrix} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 - 2x_2 - 8x_3 - 10x_4 &= 0 \\ 0 - 7x_2 - 10x_3 - 13x_4 &= 0 \end{cases} \begin{matrix} \\ \\ L_3 \leftarrow L_3 - 2L_2 \\ L_4 \leftarrow L_4 - 7L_2 \end{matrix}$$

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 4x_3 + 36x_4 &= 0 \end{cases} \begin{matrix} \\ \\ \\ L_4 \leftarrow L_4 + L_3 \end{matrix} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 &= 1 \\ 0 - x_2 - 2x_3 - 7x_4 &= 0 \\ 0 + 0 - 4x_3 + 4x_4 &= 0 \\ 0 + 0 + 0 + 40x_4 &= 0 \end{cases}$$

Donc $x_4 = 0,\ x_3 = 0,\ x_2 = 0,\ x_1 = 1,$

Exemple

Soit le système linéaire

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 = 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 4 \end{cases},$$

Résolution par la méthode du pivot de GAUSS

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 2 \\ 3x_1 + 4x_2 + x_3 + 2x_4 = 3 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 4 \end{cases} \begin{array}{l} \\ L_2 \leftarrow L_2 - 2L_1 \\ L_3 \leftarrow L_3 - 3L_1 \\ L_4 \leftarrow L_4 - 4L_1 \end{array} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 0 - x_2 - 2x_3 - 7x_4 = 0 \\ 0 - 2x_2 - 8x_3 - 10x_4 = 0 \\ 0 - 7x_2 - 10x_3 - 13x_4 = 0 \end{cases} \begin{array}{l} \\ \\ \\ L_3 \leftarrow L_3 - 2L_2 \\ L_4 \leftarrow L_4 - 7L_2 \end{array}$$

$$\begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 0 - x_2 - 2x_3 - 7x_4 = 0 \\ 0 + 0 - 4x_3 + 4x_4 = 0 \\ 0 + 0 + 4x_3 + 36x_4 = 0 \end{cases} \begin{array}{l} \\ \\ \\ L_4 \leftarrow L_4 + L_3 \end{array} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3_3 + 4x_4 = 1 \\ 0 - x_2 - 2x_3 - 7x_4 = 0 \\ 0 + 0 - 4x_3 + 4x_4 = 0 \\ 0 + 0 + 0 + 40x_4 = 0 \end{cases}$$

Donc $x_4 = 0,\ x_3 = 0,\ x_2 = 0,\ x_1 = 1,$

▷ Cas d'un pivot nul

Il se peut que $a_{kk}^{(k)} = 0$. Dans ce cas, il faut permuter les équations c-à-d. chercher $i > k$ tel que $a_{ik}^{(k)} \neq 0$ et permuter les lignes i et k dans $A^{(k)}$ et $b^{(k)}$. Remarquons que puisque A est inversible, $A^{(k)}$ est inversible. Son déterminant est :

$$\det(A^{(k)}) = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{k-1\,k-1}^{(k-1)} \det(A'),$$

où A' est le bloc restant de taille $(n-k+1) \times (n-k+1)$. Cela montre que $\det(A') \neq 0$, donc nécessairement l'un des termes $a_{ik}^{(k)}$ avec $i > k$ est non nul. Notons i_0 l'un des entiers supérieurs à k tel que $a_{i_0 k}^{(k)} \neq 0$, il faut permuter les lignes k et i_0. Cela se fait en multipliant à gauche par une matrice de permutation bien choisie. On en rappelle ici la définition.

▷ Cas d'un pivot nul

Il se peut que $a_{kk}^{(k)} = 0$. Dans ce cas, il faut permuter les équations c-à-d. chercher $i > k$ tel que $a_{ik}^{(k)} \neq 0$ et permuter les lignes i et k dans $A^{(k)}$ et $b^{(k)}$. Remarquons que puisque A est inversible, $A^{(k)}$ est inversible. Son déterminant est :

$$\det(A^{(k)}) = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{k-1\,k-1}^{(k-1)} \det(A'),$$

où A' est le bloc restant de taille $(n-k+1) \times (n-k+1)$. Cela montre que $\det(A') \neq 0$, donc nécessairement l'un des termes $a_{ik}^{(k)}$ avec $i > k$ est non nul. Notons i_0 l'un des entiers supérieurs à k tel que $a_{i_0\,k}^{(k)} \neq 0$, il faut permuter les lignes k et i_0. Cela se fait en multipliant à gauche par une matrice de permutation bien choisie. On en rappelle ici la définition.

Definition (Matrice de permutation)

Soient i et j deux entiers compris entre 1 et n et soit σ une permutation de l'ensemble $\{1, 2, ...n\}$. On appelle matrice de permutation P_σ de terme général

$$[P_\sigma]_{ij} = \delta_{i,\sigma(j)} = \begin{cases} 1, & si\ i = \sigma(j) \\ 0, & sinon \end{cases}$$

Exemple

pour $n = 3$, $i = 1$ et $j = 3$ on a $\sigma = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ et $P_\sigma = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$.

Definition (Matrice de permutation)

Soient i et j deux entiers compris entre 1 et n et soit σ une permutation de l'ensemble $\{1, 2, ...n\}$. On appelle matrice de permutation P_σ de terme général

$$[P_\sigma]_{ij} = \delta_{i,\sigma(j)} = \begin{cases} 1, & si\ i = \sigma(j) \\ 0, & sinon \end{cases}$$

Exemple

pour $n = 3$, $i = 1$ et $j = 3$ on a $\sigma = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ *et* $P_\sigma = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$.

▷ Choix du pivot

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous. Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{cases} \epsilon x + y &= 1 \\ x + y &= 2 \end{cases}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{cases} \epsilon x + y &= 1 \\ \left(1 - \frac{1}{\epsilon}\right) y &= 2 - \frac{1}{\epsilon} \end{cases}$$

On a donc immédiatement la valeur de y

$$y = \frac{1 - \frac{2}{\epsilon}}{1 - \frac{1}{\epsilon}} \simeq 1.$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq 0$$

▷ Choix du pivot

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous. Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{cases} \epsilon x + y = 1 \\ x + y = 2 \end{cases}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{cases} \epsilon x + y = 1 \\ \left(1 - \frac{1}{\epsilon}\right) y = 2 - \frac{1}{\epsilon} \end{cases}$$

On a donc immédiatement la valeur de

$$y = \frac{1 - \frac{2}{\epsilon}}{1 - \frac{1}{\epsilon}} \simeq 1.$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq$$

▷ Choix du pivot

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous. Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{cases} \epsilon x + y &= 1 \\ x + y &= 2 \end{cases}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{cases} \epsilon x + y &= 1 \\ \left(1 - \frac{1}{\epsilon}\right) y &= 2 - \frac{1}{\epsilon} \end{cases}$$

On a donc immédiatement la valeur de y

$$y = \frac{1 - \frac{2}{\epsilon}}{1 - \frac{1}{\epsilon}} \simeq 1.$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq 0$$

▷ Choix du pivot

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous. Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{cases} \epsilon x + y &= 1 \\ x + y &= 2 \end{cases}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{cases} \epsilon x + y &= 1 \\ \left(1 - \frac{1}{\epsilon}\right) y &= 2 - \frac{1}{\epsilon} \end{cases}$$

On a donc immédiatement la valeur de y :

$$y = \frac{1 - 2\epsilon}{1 - \epsilon} \simeq 1$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq 0$$

▷ Choix du pivot

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous. Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{cases} \epsilon x + y &= 1 \\ x + y &= 2 \end{cases}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{cases} \epsilon x + y &= 1 \\ \left(1 - \frac{1}{\epsilon}\right) y &= 2 - \frac{1}{\epsilon} \end{cases}$$

On a donc immédiatement la valeur de y :

$$y = \frac{1 - 2\epsilon}{1 - \epsilon} \simeq 1.$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq 0.$$

▷ Choix du pivot

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous. Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{cases} \epsilon x + y &= 1 \\ x + y &= 2 \end{cases}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{cases} \epsilon x + y &= 1 \\ \left(1 - \frac{1}{\epsilon}\right) y &= 2 - \frac{1}{\epsilon} \end{cases}$$

On a donc immédiatement la valeur de y :

$$y = \frac{1 - 2\epsilon}{1 - \epsilon} \simeq 1.$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq 0.$$

▷ Choix du pivot

Si un pivot est très petit, sans pour autant être nul, le procédé d'élimination ne s'arrête pas, mais le résultat peut être entaché d'erreurs considérables, comme on va le voir dans l'exemple ci-dessous. Soit ϵ un petit nombre réel et considérons le système de deux équations à deux inconnues

$$\begin{cases} \epsilon x + y &= 1 \\ x + y &= 2 \end{cases}$$

Par élimination sans recherche de pivot, nous obtenons le système équivalent suivant

$$\begin{cases} \epsilon x + y &= 1 \\ \left(1 - \frac{1}{\epsilon}\right) y &= 2 - \frac{1}{\epsilon} \end{cases}$$

On a donc immédiatement la valeur de y :

$$y = \frac{1 - 2\epsilon}{1 - \epsilon} \simeq 1.$$

Par substitution, on obtient x :

$$x = \frac{1 - y}{\epsilon} \simeq 0.$$

Essayons maintenant l'élimination après avoir échangé les lignes dans le système :

$$\begin{cases} x + y = 2 \\ \epsilon x + y = 1 \end{cases}$$

Le même procédé d'élimination que précédemment conduit au système équivalent suivant

$$\begin{cases} x + y = 2 \\ (1 - \epsilon)y = 1 - 2\epsilon \end{cases}$$

qui devient,

$$\begin{cases} x + y = 2 \\ y = 1 \end{cases}$$

La solution, tout à fait acceptable cette fois-ci, est $x = 1$ et $y = 1$.

Essayons maintenant l'élimination après avoir échangé les lignes dans le système :

$$\begin{cases} x + y & = 2 \\ \epsilon x + y & = 1 \end{cases}$$

Le même procédé d'élimination que précédemment conduit au système équivalent suivant

$$\begin{cases} x + y & = 2 \\ (1 - \epsilon)y & = 1 - 2\epsilon \end{cases}$$

qui devient,

$$\begin{cases} x + y & = 2 \\ y & = 1 \end{cases}$$

La solution, tout à fait acceptable cette fois-ci, est $x = 1$ et $y = 1$.

Essayons maintenant l'élimination après avoir échangé les lignes dans le système :

$$\begin{cases} x + y & = 2 \\ \epsilon x + y & = 1 \end{cases}$$

Le même procédé d'élimination que précédemment conduit au système équivalent suivant

$$\begin{cases} x + y & = 2 \\ (1 - \epsilon)y & = 1 - 2\epsilon \end{cases}$$

qui devient,

$$\begin{cases} x + y & = 2 \\ y & = 1 \end{cases}$$

La solution, tout à fait acceptable cette fois-ci, est $x = 1$ et $y = 1$.

Essayons maintenant l'élimination après avoir échangé les lignes dans le système :

$$\begin{cases} x + y & = 2 \\ \epsilon x + y & = 1 \end{cases}$$

Le même procédé d'élimination que précédemment conduit au système équivalent suivant

$$\begin{cases} x + y & = 2 \\ (1 - \epsilon)y & = 1 - 2\epsilon \end{cases}$$

qui devient,

$$\begin{cases} x + y & = 2 \\ y & = 1 \end{cases}$$

La solution, tout à fait acceptable cette fois, est $x = 1$ et $y = 1$.

Essayons maintenant l'élimination après avoir échangé les lignes dans le système :

$$\begin{cases} x + y & = 2 \\ \epsilon x + y & = 1 \end{cases}$$

Le même procédé d'élimination que précédemment conduit au système équivalent suivant

$$\begin{cases} x + y & = 2 \\ (1 - \epsilon)y & = 1 - 2\epsilon \end{cases}$$

qui devient,

$$\begin{cases} x + y & = 2 \\ y & = 1 \end{cases}$$

La solution, tout à fait acceptable cette fois, est $x = 1$ et $y = 1$.

Essayons maintenant l'élimination après avoir échangé les lignes dans le système :

$$\begin{cases} x + y & = 2 \\ \epsilon x + y & = 1 \end{cases}$$

Le même procédé d'élimination que précédemment conduit au système équivalent suivant

$$\begin{cases} x + y & = 2 \\ (1 - \epsilon)y & = 1 - 2\epsilon \end{cases}$$

qui devient,

$$\begin{cases} x + y & = 2 \\ y & = 1 \end{cases}$$

La solution, tout à fait acceptable cette fois-ci, est $x = 1$ et $y = 1$.

Remarque

Comptant le nombre d'opération nécessaire dans la méthode de Gauss, au total l'algorithme nécessite $\frac{2n^3+6n^2-2n}{6}$. *Ainsi, pour n très grand l'algorithme devient instable et donne des résultats erronés.*

Exemple

pour $n = 50$ *on a 44150 opérations à faire.*
Si on utilise le même l'Intel Core i7-3770, il faudrait donc

$$\frac{44150}{2\times 10^{11}} \text{ secondes, soit } 2.21 \times 10^{-8} \text{ secondes.}$$

Remarque

Comptant le nombre d'opération nécessaire dans la méthode de Gauss, au total l'algorithme nécessite $\frac{2n^3+6n^2-2n}{6}$. *Ainsi, pour n très grand l'algorithme devient instable et donne des résultats erronés.*

Exemple

pour $n = 50$ *on a 44150 opérations à faire.*
Si on utilise le même l'Intel Core i7-3770, il faudrait donc

$$\frac{44150}{2\times 10^{11}} \text{ secondes, soit } 2.21 \times 10^{-8} \text{ secondes.}$$

La méthode de factorisation LU

La factorisation LU d'une matrice A est une astuce très importante dans le domaine de l'analyse numérique. Sa base est très simple, mais ses applications sont très nombreuses et très utiles. Avant de montrer ces applications, regardons ce qu'est la factorisation LU et comment l'obtenir.

La factorisation LU consiste à écrire une matrice non-singulière A comme le produit de deux autres matrices L et U où L est une matrice triangulaire inférieure ayant des 1 sur la diagonale et U une matrice triangulaire supérieure.

La méthode de factorisation LU

La factorisation LU d'une matrice A est une astuce très importante dans le domaine de l'analyse numérique. Sa base est très simple, mais ses applications sont très nombreuses et très utiles. Avant de montrer ces applications, regardons ce qu'est la factorisation LU et comment l'obtenir.

La factorisation LU consiste à écrire une matrice non-singulière A comme le produit de deux autres matrices L et U où L est une matrice triangulaire inférieure ayant des 1 sur la diagonale et U une matrice triangulaire supérieure.

La méthode de factorisation LU

La factorisation LU d'une matrice A est une astuce très importante dans le domaine de l'analyse numérique. Sa base est très simple, mais ses applications sont très nombreuses et très utiles. Avant de montrer ces applications, regardons ce qu'est la factorisation LU et comment l'obtenir.

La factorisation LU consiste à écrire une matrice non-singulière A comme le produit de deux autres matrices L et U où L est une matrice triangulaire inférieure ayant des 1 sur la diagonale et U une matrice triangulaire supérieure.

Definition (Les mineurs principaux dominants)

Une sous-matrice $k \times k$ formé à partir de A en éliminant les $n - k$ dernières colonnes et les mêmes $n - k$ dernières lignes, est appelée une sous-matrice de A, d'ordre principal dominant k. Le déterminant d'une sous-matrice principale dominante $k \times k$ est appelé le mineur principal dominant d'ordre k de la matrice A.

Les mineurs principauts dominants de la matrice $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ sont

- $\Delta_1 = a$.
- $\Delta_2 = \det\left(\begin{pmatrix} a & b \\ d & e \end{pmatrix}\right)$.
- $\Delta_3 = \det(A)$.

Definition (Les mineurs principaux dominants)

Une sous-matrice $k \times k$ formé à partir de A en éliminant les $n - k$ dernières colonnes et les mêmes $n - k$ dernières lignes, est appelée une sous-matrice de A, d'ordre principal dominant k. Le déterminant d'une sous-matrice principale dominante $k \times k$ est appelé le mineur principal dominant d'ordre k de la matrice A.

Les mineurs principauts dominants de la matrice $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ sont

- $\Delta_1 = a$.
- $\Delta_2 = \det\left(\begin{pmatrix} a & b \\ d & e \end{pmatrix}\right)$.
- $\Delta_3 = \det(A)$.

Definition (Les mineurs principaux dominants)

Une sous-matrice $k \times k$ formé à partir de A en éliminant les $n - k$ dernières colonnes et les mêmes $n - k$ dernières lignes, est appelée une sous-matrice de A, d'ordre principal dominant k. Le déterminant d'une sous-matrice principale dominante $k \times k$ est appelé le mineur principal dominant d'ordre k de la matrice A.

Les mineurs principauts dominants de la matrice $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ sont

- $\Delta_1 = a$.
- $\Delta_2 = \det\left(\begin{pmatrix} a & b \\ d & e \end{pmatrix}\right)$.
- $\Delta_3 = \det(A)$.

Theorème (Décomposition LU de A)

Soit $A = (a_{ij})$ une matrice carrée d'ordre n telle que les mineurs principaux dominants soient non nuls. Alors il existe une matrice triangulaire inférieure L dont les coefficients diagonaux sont égaux à 1 et une matrice triangulaire supérieure U telles que $A = LU$. De plus, une telle décomposition est unique.

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

Theorème (Décomposition LU de A)

Soit $A = (a_{ij})$ une matrice carrée d'ordre n telle que les mineurs principaux dominants soient non nuls. Alors il existe une matrice triangulaire inférieure L dont les coefficients diagonaux sont égaux à 1 et une matrice triangulaire supérieure U telles que $A = LU$. De plus, une telle décomposition est unique.

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

Algorithme (Algorithme de factorisation LU)

Soit le système linéaire $Ax = b$.

Factorisation : On commence par factoriser la matrice A sous la forme d'un produit de deux matrices $A = LU$. Les termes non nuls de U et les termes non nuls en-dessous de la diagonale principale de L sont mémorisés encore dans la matrice A et sont ainsi calculées :

for $k = 1$ **to** $n - 1$ **do**
  for $i = k + 1$ **to** n **do**
    $a_{ik} \leftarrow \frac{a_{ik}}{a_{kk}}$
    for $j = k + 1$ **to** n **do**
    $a_{ij} \leftarrow a_{ij} - a_{ik} a_{kj}$
    end for
  end for
end for

Algorithme

Résolution : *Résoudre le système linéaire revient maintenant à résoudre successivement*

1. *le système triangulaire inférieur $Ly = b$: les éléments non nuls de la matrice triangulaire inférieure L sont donné par $l_{ij} = a_{ij}$ pour $i = 1, ..., n$ et $j = 1, ..., i$ et $l_{ii} = 1$ pour tout $i = 1, ..., n$, donc l'algorithme s'écrit*

$y_1 \leftarrow b_1$
for $i = 2$ **to** n **do**
  $s_i \leftarrow 0$
  for $j = 1$ **to** $i - 1$ **do**
    $s_i \leftarrow s_i + a_{ij} y_j$
  end for
  $y_i \leftarrow b_i - s_i$
end for

2. *le système triangulaire supérieur $Ux = y$: les éléments non nuls de la matrice triangulaire supérieure U sont donné par $u_{ij} = a_{ij}$ pour $i = 1, ..., n$ et $j = i, ..., n$ donc l'algorithme s'écrit*

$x_n \leftarrow \frac{y_n}{a_{nn}}$
for $i = n - 1$ **to** 1 **do**
  $s_i \leftarrow 0$
  for $j = 1$ **to** $i - 1$ **do**
    $s_i \leftarrow s_i + a_{ij} y_j$
  end for
  $x_i \leftarrow \frac{y_i - s_i}{a_{ii}}$
end for

Remarque

Comptant le nombre d'opération nécessaire dans la factorisation LU, au total l'algorithme nécessite $\frac{2n^3+3n^2+n}{6}$.

Remarque

Une fois que la décomposition est faite, la résolution de $Ax = b$ *se fait en résolvant d'abord* $Ly = b$, *puis* $Ux = y$ *par descente remontée. Cela permet de traiter facilement les problèmes où plusieurs seconds membres successifs avec la même matrice A apparaissent. Il faut alors calculer et stocker L et U. Remarquons que le stockage de U nécessite* $\frac{n(n+1)}{2}$ *entrées, tandis que celui de L en demande* $\frac{n(n-1)}{2}$ *car il est inutile de stocker les éléments diagonaux.*

Remarque

Comptant le nombre d'opération nécessaire dans la factorisation LU, au total l'algorithme nécessite $\frac{2n^3+3n^2+n}{6}$.

Remarque

Une fois que la décomposition est faite, la résolution de $Ax = b$ *se fait en résolvant d'abord* $Ly = b$, *puis* $Ux = y$ *par descente remontée. Cela permet de traiter facilement les problèmes où plusieurs seconds membres successifs avec la même matrice A apparaissent. Il faut alors calculer et stocker L et U. Remarquons que le stockage de U nécessite* $\frac{n(n+1)}{2}$ *entrées, tandis que celui de L en demande* $\frac{n(n-1)}{2}$ *car il est inutile de stocker les éléments diagonaux.*

▷ Décomposition $PA = LU$

- On a vu que la décomposition $A = LU$ est soumise à des conditions sur la matrice inversible A.
- Si le pivot est nul, la décomposition est impossible, à moins qu on puisse échanger les lignes.
- Si ce pivot est "presque nul", on procédera aussi par un échange de lignes.
- Ceci conduira à une décomposition LU non de la matrice A mais de la matrice PA, où P est une matrice de permutation.

▷ Décomposition $PA = LU$

- On a vu que la décomposition $A = LU$ est soumise à des conditions sur la matrice inversible A.
- Si le pivot est nul, la décomposition est impossible, à moins qu'on puisse échanger les lignes.
- Si ce pivot est "presque nul", on procédera aussi par un échange de lignes.
- Ceci conduira à une décomposition LU non de la matrice A mais de la matrice PA, où P est une matrice de permutation.

▷ Décomposition $PA = LU$

- On a vu que la décomposition $A = LU$ est soumise à des conditions sur la matrice inversible A.
- Si le pivot est nul, la décomposition est impossible, à moins qu'on puisse échanger les lignes.
- Si ce pivot est "presque nul", on procédera aussi par un échange de lignes.
- Ceci conduira à une décomposition LU non de la matrice A mais de la matrice PA, où P est une matrice de permutation.

▷ Décomposition $PA = LU$

- On a vu que la décomposition $A = LU$ est soumise à des conditions sur la matrice inversible A.
- Si le pivot est nul, la décomposition est impossible, à moins qu'on puisse échanger les lignes.
- Si ce pivot est "presque nul", on procédera aussi par un échange de lignes.
- Ceci conduira à une décomposition LU non de la matrice A mais de la matrice PA, où P est une matrice de permutation.

▷ Décomposition $PA = LU$

- On a vu que la décomposition $A = LU$ est soumise à des conditions sur la matrice inversible A.
- Si le pivot est nul, la décomposition est impossible, à moins qu'on puisse échanger les lignes.
- Si ce pivot est "presque nul", on procédera aussi par un échange de lignes.
- Ceci conduira à une décomposition LU non de la matrice A mais de la matrice PA, où P est une matrice de permutation.

Proposition

- $\det(A) = \det(L)\det(U) = \prod_{k=1}^{n} u_{kk}$
- *Le calcul explicite de l'inverse d'une matrice peut être effectué en utilisant la factorisation LU comme suit. En notant X l'inverse d'une matrice régulière A, les vecteurs colonnes de X sont les solutions des systèmes linéaires*

$$Ax_i = e_i, \ \text{pour } i = 1, ..., n.$$

En supposant que $PA = LU$, où P est la matrice de changement de pivot partiel, on doit résoudre $2n$ systèmes triangulaires de la forme

$$Ly_i = Pe_i, \ Ux_i = y_i, \ \text{pour } i = 1, ..., n,$$

c'est-à-dire une suite de systèmes linéaires ayant la même matrice mais des seconds membres différents.

Exercice

Soit les systèmes linéaires

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad et \quad \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \end{pmatrix}$$

1. *Résoudre les systèmes linéaires par la méthode du pivot de GAUSS.*
2. *Factoriser la matrice A (sans utiliser la technique du pivot) et résoudre les systèmes linéaires.*
3. *Calculer le déterminant de A.*
4. *Calculer A^{-1}.*

- **Méthode de Cholesky**

Dans le cas des matrices symétriques, on a une décomposition LU où L et U sont transposées l'une de l'autre. Cela a un avantage pratique évident : on avait vu que le stockage d'une matrice symétrique $n \times n$ ne nécessite que $\frac{n(n+1)}{2}$ coefficients, et donc dans ce cas, la décomposition LU nécessite uniquement la sauvegarde de la matrice L c'est-à-dire la sauvegarde de $\frac{n(n+1)}{2}$ coefficients. On a alors la décomposition de Cholesky qui est décrite par le résultat suivant.

- **Méthode de Cholesky**

Dans le cas des matrices symétriques, on a une décomposition LU où L et U sont transposées l'une de l'autre. Cela a un avantage pratique évident : on avait vu que le stockage d'une matrice symétrique $n \times n$ ne nécessite que $\frac{n(n+1)}{2}$ coefficients, et donc dans ce cas, la décomposition LU nécessite uniquement la sauvegarde de la matrice L c'est-à-dire la sauvegarde de $\frac{n(n+1)}{2}$ coefficients. On a alors la décomposition de Cholesky qui est décrite par le résultat suivant.

Definition

Une matrice A, $n \times n$ est définie positive et on note $A \succ 0$ si chacun des mineurs principaux dominants de A est > 0.

Exemple

Soit $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$, *alors*

$$A \succ 0 \iff a_{11} > 0, \begin{vmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{vmatrix} > 0 \text{ et } \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{vmatrix} > 0.$$

Definition

Une matrice A, $n \times n$ est définie positive et on note $A \succ 0$ si chacun des mineurs principaux dominants de A est > 0.

Exemple

Soit $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$, *alors*

$$A \succ 0 \iff a_{11} > 0, \begin{vmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{vmatrix} > 0 \text{ et } \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{vmatrix} > 0.$$

Theorème

Soit A une matrice réelle symétrique définie positive. Il existe une unique matrice réelle B triangulaire inférieure, telle que tous ses éléments diagonaux soient positifs et qui vérifie : $A = BB^t$.

Exemple

Pour comprendre le principe voici un exemple dans le cas $n = 2$.

On prend $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$, alors puisque $A \succ 0$ on a $a > 0$. On effectue la décomposition LU de A et on obtient

$$A = LU = \begin{pmatrix} 1 & 0 \\ \frac{b}{a} & 1 \end{pmatrix} \begin{pmatrix} a & b \\ 0 & c - \frac{b^2}{a} \end{pmatrix}.$$

En extrayant la diagonale de U, on obtient :

$$A = LU = \begin{pmatrix} 1 & 0 \\ \frac{b}{a} & 1 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & c - \frac{b^2}{a} \end{pmatrix} \begin{pmatrix} 1 & \frac{b}{a} \\ 0 & 1 \end{pmatrix}.$$

Theorème

Soit A une matrice réelle symétrique définie positive. Il existe une unique matrice réelle B triangulaire inférieure, telle que tous ses éléments diagonaux soient positifs et qui vérifie : $A = BB^t$.

Exemple

Pour comprendre le principe voici un exemple dans le cas $n = 2$.

On prend $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$, alors puisque $A \succ 0$ on a $a > 0$. On effectue la décomposition LU de A et on obtient

$$A = LU = \begin{pmatrix} 1 & 0 \\ \frac{b}{a} & 1 \end{pmatrix} \begin{pmatrix} a & b \\ 0 & c - \frac{b^2}{a} \end{pmatrix}.$$

En extrayant la diagonale de U, on obtient :

$$A = LU = \begin{pmatrix} 1 & 0 \\ \frac{b}{a} & 1 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & c - \frac{b^2}{a} \end{pmatrix} \begin{pmatrix} 1 & \frac{b}{a} \\ 0 & 1 \end{pmatrix}.$$

Algorithme

for $j = 1$ **to** n **do**
  for $k = 1$ **to** $j - 1$ **do**
    $a_{jj} \leftarrow a_{jj} - a_{jk}^2$
  end for
  $a_{jj} \leftarrow \sqrt{a_{jj}}$
  for $i = j + 1$ **to** n **do**
    for $k = 1$ **to** $j - 1$ **do**
      $a_{ij} \leftarrow a_{ij} - a_{ik} a_{jk}$
    end for
    $a_{ij} \leftarrow \frac{a_{ij}}{a_{jj}}$
  end for
end for

Exemple

Soit $A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 \\ 1 & 5 & 14 & 14 \\ 1 & 5 & 14 & 15 \end{pmatrix}$, Effectuer la factorisation de Cholesky de la matrice A.

Remarque

Pour résoudre un système linéaire par cette méthode, il faut environ $\frac{n^3}{6}$ additions et multiplications, $\frac{n^2}{2}$ divisions et n extractions de racines carrées.

Exemple

pour $n = 50$ on a 22133 opérations à faire.
Si on utilise le même l'Intel Core i7-3770, il faudrait donc

$$\frac{22133}{2\times 10^{11}} \text{ secondes, soit } 1.11 \times 10^{-8} \text{ secondes.}$$

Remarque

Pour résoudre un système linéaire par cette méthode, il faut environ $\frac{n^3}{6}$ *additions et multiplications,* $\frac{n^2}{2}$ *divisions et n extractions de racines carrées.*

Exemple

pour $n = 50$ *on a 22133 opérations à faire.*
Si on utilise le même l'Intel Core i7-3770, il faudrait donc

$$\boxed{\frac{22133}{2\times 10^{11}} \text{ secondes, soit } 1.11 \times 10^{-8} \text{ secondes.}}$$

Proposition (Matrices non symétriques)

Soit $A \in M_n(\mathbb{R})$ inversible. On suppose que A est non symétrique. On cherche a résoudre le système $Ax = b$ par la méthode de Cholesky. Ceci est possible en remarquant que : $Ax = b \iff A^tAx = A^tb$ Il ne reste alors plus qu'à vérifier que A^tA est symétrique définie positive.

Méthodes itératives de résolution de $Ax = b$

Une alternative aux méthodes directes de type Gauss, LU ou Cholesky présentées dans la section précédente est l'utilisation de méthodes itératives. Dans ce type de méthodes on ne cherche pas à déterminer la solution du système linéaire $Ax = b$, mais à construire une suite de vecteurs $\{x_k\}_{k\in\mathbb{N}}$ qui converge vers la solution du problème.

Ces méthodes sont utilisées en particulier quand on a des matrices creuses, pour lesquelles les produits matrices-vecteurs ont un coût essentiellement proportionnel à la longueur des vecteurs. Si on a une suite $\{x_k\}_{k\in\mathbb{N}}$ qui converge rapidement, alors le coût global de l'algorithme est bien plus faible que celui d'une méthode directe.

Une alternative aux méthodes directes de type Gauss, LU ou Cholesky présentées dans la section précédente est l'utilisation de méthodes itératives. Dans ce type de méthodes on ne cherche pas à déterminer la solution du système linéaire $Ax = b$, mais à construire une suite de vecteurs $\{x_k\}_{k\in\mathbb{N}}$ qui converge vers la solution du problème.

Ces méthodes sont utilisées en particulier quand on a des matrices creuses, pour lesquelles les produits matrices-vecteurs ont un coût essentiellement proportionnel à la longueur des vecteurs. Si on a une suite $\{x_k\}_{k\in\mathbb{N}}$ qui converge rapidement, alors le coût global de l'algorithme est bien plus faible que celui d'une méthode directe.

Principe général

On veut résoudre le système suivant

$$\begin{aligned} Ax = b &\iff x + Ax - x = b \\ &\iff x = \underbrace{(I - A)}_{B}x + b. \end{aligned}$$

On est donc ramené à la recherche d'un point fixe de l'application

$x \longmapsto Bx + b$, pour cela on considère la suite définie pour $k \in \mathbb{N}$, par

$$\begin{cases} x_0 \text{ donné} \\ x_{k+1} = Bx_k + b. \end{cases} \tag{2}$$

La matrice B es appelée *la matrice de la méthode itératives*.

Principe général

On veut résoudre le système suivant

$$\begin{aligned} Ax = b &\iff x + Ax - x = b \\ &\iff x = \underbrace{(I - A)}_{B}x + b. \end{aligned}$$

On est donc ramené à la recherche d'un point fixe de l'application

$x \longmapsto Bx + b$, pour cela on considère la suite définie pour $k \in \mathbb{N}$, par

$$\begin{cases} x_0 \text{ donné} \\ x_{k+1} = Bx_k + b \end{cases} \tag{2}$$

La matrice B es appelée *la matrice de la méthode itératives*.

Principe général

On veut résoudre le système suivant

$$\begin{aligned} Ax = b &\iff x + Ax - x = b \\ &\iff x = \underbrace{(I - A)}_{B}x + b. \end{aligned}$$

On est donc ramené à la recherche d'un point fixe de l'application

$x \longmapsto Bx + b$, pour cela on considère la suite définie pour $k \in \mathbb{N}$, par

$$\begin{cases} x_0 \text{ donné} \\ x_{k+1} = Bx_k + b \end{cases} \tag{2}$$

La matrice B es appelée *la matrice de la méthode itératives*.

Principe général

On veut résoudre le système suivant

$$\begin{aligned} Ax = b &\iff x + Ax - x = b \\ &\iff x = \underbrace{(I - A)}_{B}x + b. \end{aligned}$$

On est donc ramené à la recherche d'un point fixe de l'application

$x \longmapsto Bx + b$, pour cela on considère la suite définie pour $k \in \mathbb{N}$, par

$$\begin{cases} x_0 \text{ donné} \\ x_{k+1} = Bx_k + b. \end{cases} \tag{2}$$

La matrice B es appelée *la matrice de la méthode itératives*.

Principe général

On veut résoudre le système suivant

$$\begin{aligned} Ax = b &\iff x + Ax - x = b \\ &\iff x = \underbrace{(I - A)}_{B}x + b. \end{aligned}$$

On est donc ramené à la recherche d'un point fixe de l'application

$x \longmapsto Bx + b$, pour cela on considère la suite définie pour $k \in \mathbb{N}$, par

$$\begin{cases} x_0 \text{ donné} \\ x_{k+1} = Bx_k + b. \end{cases} \tag{2}$$

La matrice B es appelée *la matrice de la méthode itératives*.

Definition

Le système itératif (2) est convergent si

$$\lim_{k \to +\infty} x_k = x, \quad \forall x_0.$$

Remarque

Posons $\epsilon_k = x_k - x$, *pour* $k = 0, 1, \ldots$

comme $x = Bx + b$ *et* $x_{k+1} = Bx_k + b$ *on a*

$\epsilon_k = Bx_{k-1} - Bx = B\epsilon_{k-1} = \cdots = B^k \epsilon_0.$

$$\lim_{k \to +\infty} x_k = x \iff \lim_{k \to +\infty} \epsilon_k = 0 \iff \lim_{k \to +\infty} B^k \epsilon_0 = 0.$$

Donc le système itératif (2) est convergent si $\lim_{k \to +\infty} B^k v = 0, \ \forall v$

ce qui équivaut à $\lim_{k \to +\infty} \|B^k v\| = 0, \ \forall v$ *pour toute norme vectorielle* $\|\cdot\|$.

Definition

Le système itératif (2) est convergent si

$$\lim_{k \to +\infty} x_k = x, \quad \forall x_0.$$

Remarque

Posons $\epsilon_k = x_k - x$, *pour* $k = 0, 1, \ldots$

comme $x = Bx + b$ *et* $x_{k+1} = Bx_k + b$ *on a*

$\epsilon_k = Bx_{k-1} - Bx = B\epsilon_{k-1} = \cdots = B^k \epsilon_0.$

$$\lim_{k \to +\infty} x_k = x \iff \lim_{k \to +\infty} \epsilon_k = 0 \iff \lim_{k \to +\infty} B^k \epsilon_0 = 0.$$

Donc le système itératif (2) est convergent si $\lim_{k \to +\infty} B^k v = 0, \ \forall v$

ce qui équivaut à $\lim_{k \to +\infty} \|B^k v\| = 0, \ \forall v$ *pour toute norme vectorielle* $\|\cdot\|$.

Convergence des méthodes itératives

Le résultat suivant donne le critère fondamental de convergence des méthodes itératives.

Theorème

Les trois conditions suivantes sont équivalentes

1. *Le système itérative (2) converge.*
2. $\rho(B) < 1$.
3. *On peut trouver une norme matricielle $\|\cdot\|$ telle que $\|B\| < 1$.*

Démonstration.

La preuve du théorème se déduit du théorème 3 du chapitre 1. □

Convergence des méthodes itératives

Le résultat suivant donne le critère fondamental de convergence des méthodes itératives.

Theorème

Les trois conditions suivantes sont équivalentes

1. *Le système itérative (2) converge.*
2. $\rho(B) < 1$.
3. *On peut trouver une norme matricielle $\|\cdot\|$ telle que $\|B\| < 1$.*

Démonstration.

La preuve du théorème se déduit du théorème 3 du chapitre 1. □

Convergence des méthodes itératives

Le résultat suivant donne le critère fondamental de convergence des méthodes itératives.

Theorème

Les trois conditions suivantes sont équivalentes

1. *Le système itérative (2) converge.*
2. $\rho(B) < 1$.
3. *On peut trouver une norme matricielle* $\|\cdot\|$ *telle que* $\|B\| < 1$.

Démonstration.

La preuve du théorème se déduit du théorème 3 du chapitre 1. □

Convergence des méthodes itératives

Le résultat suivant donne le critère fondamental de convergence des méthodes itératives.

Theorème

Les trois conditions suivantes sont équivalentes

1. *Le système itérative (2) converge.*
2. $\rho(B) < 1$.
3. *On peut trouver une norme matricielle* $\|\cdot\|$ *telle que* $\|B\| < 1$.

Démonstration.

La preuve du théorème se déduit du théorème 3 du chapitre 1. □

Les questions qui se posent alors sont les suivantes :

1. Comment choisir la matrice B pour que la méthode converge ?
2. Si deux matrices B_i conviennent quelle est la « meilleure », c'est-à-dire celle pour laquelle la méthode est la plus rapide ou demande le moins d'opérations etc ... ?

Les questions qui se posent alors sont les suivantes :

1. Comment choisir la matrice B pour que la méthode converge ?
2. Si deux matrices B_i conviennent quelle est la « meilleure », c'est-à-dire celle pour laquelle la méthode est la plus rapide ou demande le moins d'opérations etc ... ?

Les questions qui se posent alors sont les suivantes :

1. Comment choisir la matrice B pour que la méthode converge ?
2. Si deux matrices B_i conviennent quelle est la « meilleure », c'est-à-dire celle pour laquelle la méthode est la plus rapide ou demande le moins d'opérations etc ... ?

Les méthodes de Jacobi, de Gauss-Seidel et de relaxation

Nous allons présenter plusieurs façons d'obtenir une matrice B comme ci-dessus à partir de la matrice originale A du système $Ax = b$. Supposons que la matrice A puisse s'écrire $A = M - N$ où la matrice M est "facile" à inverser. Nous aurons alors

$$Ax = b \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b.$$

la matrice B sera donc $B = M^{-1}N$ mais en pratique nous verrons qu'on ne calcule pas M^{-1}. La méthode itérative associée est

$$\begin{cases} x_0 \text{ donné} \\ Mx_{k+1} = Nx_k + b. \end{cases} \tag{3}$$

Les méthodes de Jacobi, de Gauss-Seidel et de relaxation

Nous allons présenter plusieurs façons d'obtenir une matrice B comme ci-dessus à partir de la matrice originale A du système $Ax = b$. Supposons que la matrice A puisse s'écrire $A = M - N$ où la matrice M est "facile" à inverser. Nous aurons alors

$$Ax = b \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b.$$

la matrice B sera donc $B = M^{-1}N$ mais en pratique nous verrons qu'on ne calcule pas M^{-1}. La méthode itérative associée est

$$\begin{cases} x_0 \text{ donné} \\ Mx_{k+1} = Nx_k + b. \end{cases} \tag{3}$$

Les méthodes de Jacobi, de Gauss-Seidel et de relaxation

Nous allons présenter plusieurs façons d'obtenir une matrice B comme ci-dessus à partir de la matrice originale A du système $Ax = b$. Supposons que la matrice A puisse s'écrire $A = M - N$ où la matrice M est "facile" à inverser. Nous aurons alors

$$Ax = b \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b.$$

la matrice B sera donc $B = M^{-1}N$ mais en pratique nous verrons qu'on ne calcule pas M^{-1}. La méthode itérative associée est

$$\begin{cases} x_0 \text{ donné} \\ Mx_{k+1} = Nx_k + b. \end{cases} \tag{3}$$

Les méthodes de Jacobi, de Gauss-Seidel et de relaxation

Nous allons présenter plusieurs façons d'obtenir une matrice B comme ci-dessus à partir de la matrice originale A du système $Ax = b$. Supposons que la matrice A puisse s'écrire $A = M - N$ où la matrice M est "facile" à inverser. Nous aurons alors

$$Ax = b \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b.$$

la matrice B sera donc $B = M^{-1}N$ mais en pratique nous verrons qu'on ne calcule pas M^{-1}. La méthode itérative associée est

$$\begin{cases} x_0 \text{ donné} \\ Mx_{k+1} = Nx_k + b. \end{cases} \tag{3}$$

Les méthodes de Jacobi, de Gauss-Seidel et de relaxation

Nous allons présenter plusieurs façons d'obtenir une matrice B comme ci-dessus à partir de la matrice originale A du système $Ax = b$. Supposons que la matrice A puisse s'écrire $A = M - N$ où la matrice M est "facile" à inverser. Nous aurons alors

$$Ax = b \iff Mx = Nx + b \iff x = M^{-1}Nx + M^{-1}b.$$

la matrice B sera donc $B = M^{-1}N$ mais en pratique nous verrons qu'on ne calcule pas M^{-1}. La méthode itérative associée est

$$\begin{cases} x_0 \text{ donné} \\ Mx_{k+1} = Nx_k + b. \end{cases} \tag{3}$$

Il y a une infinité de choix possibles pour M et N vérifiant $A = M - N$, nous allons en donner deux à titre d'exemples. L'idée est bien sûr de choisir une matrice M particulièrement facile à inverser, par exemple diagonale, ou bien triangulaire inférieure. Ces deux choix correspondent respectivement à la méthode de Jacobi et à la méthode de Gauss-Seidel.

On décompose A en $A = D - E - F$ où D est la matrice diagonale contenant la diagonale de A, E est la partie triangulaire inférieure stricte (sans la diagonale) de $-A$ et F est la partie triangulaire supérieure stricte de $-A$.

Il y a une infinité de choix possibles pour M et N vérifiant $A = M - N$, nous allons en donner deux à titre d'exemples. L'idée est bien sûr de choisir une matrice M particulièrement facile à inverser, par exemple diagonale, ou bien triangulaire inférieure. Ces deux choix correspondent respectivement à la méthode de Jacobi et à la méthode de Gauss-Seidel.

On décompose A en $A = D - E - F$ où D est la matrice diagonale contenant la diagonale de A, E est la partie triangulaire inférieure stricte (sans la diagonale) de $-A$ et F est la partie triangulaire supérieure stricte de $-A$.

Exemple

Considérons la matrice

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix} \tag{4}$$

La décomposition de A sous la forme $A = D - E - F$ décrite ci-dessus s'écrit alors

$$A = \underbrace{\begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}}_{D} - \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}}_{E} - \underbrace{\begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{pmatrix}}_{F}$$

Exemple

Considérons la matrice

$$A = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix} \tag{4}$$

La décomposition de A sous la forme $A = D - E - F$ décrite ci-dessus

s'écrit alors

$$A = \underbrace{\begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}}_{D} - \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}}_{E} - \underbrace{\begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & 0 \end{pmatrix}}_{F}$$

→ La méthode de JACOBI

La méthode de Jacobi correspond à la décomposition

$$M = D \text{ (diagonale) et } N = E + F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ Dx^{(k+1)} = (E+F)x^{(k)} + b. \end{cases} \tag{5}$$

On suppose que D est inversible, c'est-à-dire $a_{ii} \neq 0, \ 1 \leq i \leq n$.

La matrice $J = D^{-1}(E+F) = I_n - D^{-1}A$ est appelée la matrice de Jacobi de (5).
On peut donc décrire la méthode de Jacobi par l'expression suivante,

$$x^{(k+1)} = D^{-1}(E+F)x^{(k)} + D^{-1}b. \tag{6}$$

→ La méthode de JACOBI

La méthode de Jacobi correspond à la décomposition

$$M = D \text{ (diagonale) et } N = E + F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ Dx^{(k+1)} = (E+F)x^{(k)} + b. \end{cases} \tag{5}$$

On suppose que D est inversible, c'est-à-dire $a_{ii} \neq 0, \ 1 \leq i \leq n$.

La matrice $J = D^{-1}(E+F) = I_n - D^{-1}A$ est appelée la matrice de Jacobi de (5).
On peut donc décrire la méthode de Jacobi par l'expression suivante,

$$x^{(k+1)} = D^{-1}(E+F)x^{(k)} + D^{-1}b. \tag{6}$$

→ La méthode de JACOBI

La méthode de Jacobi correspond à la décomposition

$$M = D \text{ (diagonale) et } N = E + F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ Dx^{(k+1)} = (E+F)x^{(k)} + b. \end{cases} \tag{5}$$

On suppose que D est inversible, c'est-à-dire $a_{ii} \neq 0,\ 1 \leq i \leq n$.

La matrice $J = D^{-1}(E+F) = I_n - D^{-1}A$ est appelée la matrice de Jacobi de (5).
On peut donc décrire la méthode de Jacobi par l'expression suivante,

$$x^{(k+1)} = D^{-1}(E+F)x^{(k)} + D^{-1}b. \tag{6}$$

Si on explicite (6), cela donne

$$x_i^{(k+1)} = \sum_{i \neq j} \frac{-a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}}. \tag{7}$$

Dans (7) on voit que, pour calculer $x^{(k+1)}$, on se sert uniquement des valeurs $x^{(k)}$ calculées à l'itération précédente. Par exemple, pour calculer $x_j^{(k+1)}$, on utilise $x_1^{(k)}$, ..., $x_n^{(k)}$. Cependant, si on suppose que le processus est convergent, lorsque l'on calcule $x_j^{(k)}$, on a déjà à sa disposition des valeurs approchées plus précises de $\bar{x}_1, ..., \bar{x}_{j-1}$, à savoir $x_1^{(k)}, ..., x_{j-1}^{(k)}$. L'idée de la méthode de **Gauss-Seidel** est donc d'utiliser à l'itération k toutes les nouvelles valeurs qui ont déjà été calculées à l'itération k.

Si on explicite (6), cela donne

$$x_i^{(k+1)} = \sum_{i \neq j} \frac{-a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}}. \tag{7}$$

Dans (7) on voit que, pour calculer $x^{(k+1)}$, on se sert uniquement des

valeurs $x^{(k)}$ calculées à l'itération précédente. Par exemple, pour calculer $x_j^{(k+1)}$, on utilise $x_1^{(k)}$, ..., $x_n^{(k)}$. Cependant, si on suppose que le processus est convergent, lorsque l'on calcule $x_j^{(k)}$, on a déjà à sa disposition des valeurs approchées plus précises de $\bar{x}_1, ..., \bar{x}_{j-1}$, à savoir $x_1^{(k)}, ..., x_{j-1}^{(k)}$.
L'idée de la méthode de **Gauss-Seidel** est donc d'utiliser à l'itération k toutes les nouvelles valeurs qui ont déjà été calculées à l'itération k.

→ La méthode de Gauss-Seidel

La méthode de Gauss-Seidel correspond à la décomposition

$$M = D - E \text{ (triangulaire inférieure) et } N = F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ (D - E)x_{k+1} = Fx_k + b. \end{cases} \tag{8}$$

On suppose que $a_{ii} \neq 0$, $1 \leq i \leq n$, alors la matrice $L_1 = (D - E)^{-1}F$ est la matrice de

Gauss-Seidel. Autrement dit, pour $k \geq 1$ on calcule les composantes de l'itérée $x^{(k+1)}$ à partir

des itérées de $x^{(k)}$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} (a_{ij} x_j^{(k+1)} - \sum_{j=i+1} (a_{ij} x_j^{(k)}) + b_i \right].$$

la mise en oeuvre de l'algorithme de Gauss-Seidel est en réalité plus simple. A chaque itération,

il n'est, en effet, pas nécessaire de sauvegarder les itérés précédents. Chaque nouvelle valeur

calculée est automatiquement réutilisée pour les calculs suivants.

→ La méthode de Gauss-Seidel

La méthode de Gauss-Seidel correspond à la décomposition

$$M = D - E \text{ (triangulaire inférieure) et } N = F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ (D - E)x_{k+1} = Fx_k + b. \end{cases} \tag{8}$$

On suppose que $a_{ii} \neq 0$, $1 \leq i \leq n$, alors la matrice $L_1 = (D - E)^{-1}F$ est la matrice de Gauss-Seidel. Autrement dit, pour $k \geq 1$ on calcule les composantes de l'itérée $x^{(k+1)}$ à partir des itérées de $x^{(k)}$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} (a_{ij} x_j^{(k+1)} - \sum_{j=i+1} (a_{ij} x_j^{(k)}) + b_i \right].$$

la mise en oeuvre de l'algorithme de Gauss-Seidel est en réalité plus simple. A chaque itération, il n'est, en effet, pas nécessaire de sauvegarder les itérés précédents. Chaque nouvelle valeur calculée est automatiquement réutilisée pour les calculs suivants.

→ La méthode de Gauss-Seidel

La méthode de Gauss-Seidel correspond à la décomposition

$$M = D - E \text{ (triangulaire inférieure) et } N = F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ (D - E)x_{k+1} = Fx_k + b. \end{cases} \tag{8}$$

On suppose que $a_{ii} \neq 0$, $1 \leq i \leq n$, alors la matrice $L_1 = (D - E)^{-1}F$ est la matrice de Gauss-Seidel. Autrement dit, pour $k \geq 1$ on calcule les composantes de l'itérée $x^{(k+1)}$ à partir des itérées de $x^{(k)}$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} (a_{ij} x_j^{(k+1)}) - \sum_{j=i+1}^{n} (a_{ij} x_j^{(k)}) + b_i \right].$$

la mise en oeuvre de l'algorithme de Gauss-Seidel est en réalité plus simple. A chaque itération, il n'est, en effet, pas nécessaire de sauvegarder les itérés précédents. Chaque nouvelle valeur calculée est automatiquement réutilisée pour les calculs suivants.

→ La méthode de Gauss-Seidel

La méthode de Gauss-Seidel correspond à la décomposition

$$M = D - E \text{ (triangulaire inférieure) et } N = F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ (D - E)x_{k+1} = Fx_k + b. \end{cases} \tag{8}$$

On suppose que $a_{ii} \neq 0$, $1 \leq i \leq n$, alors la matrice $L_1 = (D - E)^{-1}F$ est la matrice de Gauss-Seidel. Autrement dit, pour $k \geq 1$ on calcule les composantes de l'itérée $x^{(k+1)}$ à partir des itérées de $x^{(k)}$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} (a_{ij} x_j^{(k+1)}) - \sum_{j=i+1}^{n} (a_{ij} x_j^{(k)}) + b_i \right].$$

la mise en oeuvre de l'algorithme de Gauss-Seidel est en réalité plus simple. A chaque itération, il n'est, en effet, pas nécessaire de sauvegarder les itérées précédents. Chaque nouvelle valeur calculée est automatiquement réutilisée pour les calculs suivants.

→ La méthode de Gauss-Seidel

La méthode de Gauss-Seidel correspond à la décomposition

$$M = D - E \text{ (triangulaire inférieure) et } N = F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ (D - E)x_{k+1} = Fx_k + b. \end{cases} \tag{8}$$

On suppose que $a_{ii} \neq 0$, $1 \leq i \leq n$, alors la matrice $L_1 = (D - E)^{-1}F$ est la matrice de Gauss-Seidel. Autrement dit, pour $k \geq 1$ on calcule les composantes de l'itérée $x^{(k+1)}$ à partir des itérées de $x^{(k)}$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} (a_{ij} x_j^{(k+1)}) - \sum_{j=i+1}^{n} (a_{ij} x_j^{(k)}) + b_i \right].$$

la mise en oeuvre de l'algorithme de Gauss-Seidel est en réalité plus simple. A chaque itération, il n'est, en effet, pas nécessaire de sauvegarder les itérées précédents. Chaque nouvelle valeur calculée est automatiquement réutilisée pour les calculs suivants.

→ La méthode de Gauss-Seidel

La méthode de Gauss-Seidel correspond à la décomposition

$$M = D - E \text{ (triangulaire inférieure) et } N = F.$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ (D - E)x_{k+1} = Fx_k + b. \end{cases} \tag{8}$$

On suppose que $a_{ii} \neq 0$, $1 \leq i \leq n$, alors la matrice $L_1 = (D - E)^{-1}F$ est la matrice de Gauss-Seidel. Autrement dit, pour $k \geq 1$ on calcule les composantes de l'itérée $x^{(k+1)}$ à partir des itérées de $x^{(k)}$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} (a_{ij} x_j^{(k+1)}) - \sum_{j=i+1}^{n} (a_{ij} x_j^{(k)}) + b_i \right].$$

la mise en oeuvre de l'algorithme de Gauss-Seidel est en réalité plus simple. A chaque itération, il n'est, en effet, pas nécessaire de sauvegarder les itérées précédents. Chaque nouvelle valeur calculée est automatiquement réutilisée pour les calculs suivants.

→ La méthode de relaxation

On peut, dans certains cas, améliorer la convergence de l'algorithme de Gauss-Seidel en introduisant un paramètre que l'on pourra modifier en fonction de la matrice A de départ. Pour ce faire, on introduit un facteur $0 < \omega < 2$. La méthode de relaxation est une méthode entre la méthode de Jacobi et celle de Gauss-Seidel et correspond à la décomposition suivante, Pour $\omega \neq 0$, en posant

$$M = \frac{1}{\omega} D - E, \quad N = \left(\frac{1}{\omega} - 1 \right) D + F,$$

la matrice de la méthode est

$$L_\omega = \left(\frac{1}{\omega} D - E \right)^{-1} \left(F + \left(\frac{1}{\omega} - 1 \right) D \right).$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ \left(\frac{1}{\omega} D - E \right) x_{k+1} = \left(F + \left(\frac{1}{\omega} - 1 \right) D \right) x_k + b. \end{cases} \tag{9}$$

Il faut encore résoudre à chaque étape un système triangulaire inférieur où la matrice L_ω n'est pas plus compliquée que L_1. Si

- $\omega = 1$, on retrouve la méthode de Gauss-Seidel.
- $\omega > 1$, la méthode est dite de sur -relaxation.
- $\omega < 1$, la méthode est dite de sous -relaxation.

→ La méthode de relaxation

On peut, dans certains cas, améliorer la convergence de l'algorithme de Gauss-Seidel en introduisant un paramètre que l'on pourra modifier en fonction de la matrice A de départ. Pour ce faire, on introduit un facteur $0 < \omega < 2$. La méthode de relaxation est une méthode entre la méthode de Jacobi et celle de Gauss-Seidel et correspond à la décomposition suivante, Pour $\omega \neq 0$, en posant

$$M = \frac{1}{\omega} D - E, \quad N = \left(\frac{1-\omega}{\omega} \right) D + F,$$

la matrice de la méthode est

$$L_\omega = \left(\frac{1}{\omega} D - E \right)^{-1} \left(F + \left(\frac{1}{\omega} - 1 \right) D \right).$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ \left(\frac{1}{\omega} D - E \right) x_{k+1} = \left(F + \left(\frac{1}{\omega} - 1 \right) D \right) x_k + b. \end{cases} \tag{9}$$

Il faut encore résoudre à chaque étape un système triangulaire inférieur où la matrice L_ω n'est pas plus compliquée que L_1. Si

- $\omega = 1$, on retrouve la méthode de Gauss-Seidel.
- $\omega > 1$, la méthode est dite de sur -relaxation.
- $\omega < 1$, la méthode est dite de sous -relaxation.

→ La méthode de relaxation

On peut, dans certains cas, améliorer la convergence de l'algorithme de Gauss-Seidel en introduisant un paramètre que l'on pourra modifier en fonction de la matrice A de départ. Pour ce faire, on introduit un facteur $0 < \omega < 2$. La méthode de relaxation est une méthode entre la

méthode de Jacobi et celle de Gauss-Seidel et correspond à la décomposition suivante, Pour $\omega \neq 0$, en posant

$$M = \frac{1}{\omega} D - E, \quad N = \left(\frac{1-\omega}{\omega} \right) D + F,$$

la matrice de la méthode est

$$L_\omega = \left(\frac{1}{\omega} D - E \right)^{-1} \left(F + \left(\frac{1}{\omega} - 1 \right) D \right).$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ \left(\frac{1}{\omega} D - E \right) x_{k+1} = \left(F + \left(\frac{1}{\omega} - 1 \right) D \right) x_k + b. \end{cases} \tag{9}$$

Il faut encore résoudre à chaque étape un système triangulaire inférieur où la matrice L_ω n'est

pas plus compliquée que L_1. Si

- $\omega = 1$, on retrouve la méthode de Gauss-Seidel.
- $\omega > 1$, la méthode est dite de sur -relaxation.
- $\omega < 1$, la méthode est dite de sous -relaxation.

→ La méthode de relaxation

On peut, dans certains cas, améliorer la convergence de l'algorithme de Gauss-Seidel en introduisant un paramètre que l'on pourra modifier en fonction de la matrice A de départ. Pour ce faire, on introduit un facteur $0 < \omega < 2$. La méthode de relaxation est une méthode entre la méthode de Jacobi et celle de Gauss-Seidel et correspond à la décomposition suivante, Pour $\omega \neq 0$, en posant

$$M = \frac{1}{\omega} D - E, \quad N = \left(\frac{1 - \omega}{\omega} \right) D + F,$$

la matrice de la méthode est

$$L_\omega = \left(\frac{1}{\omega} D - E \right)^{-1} \left(F + \left(\frac{1}{\omega} - 1 \right) D \right).$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ \left(\frac{1}{\omega} D - E \right) x_{k+1} = \left(F + \left(\frac{1}{\omega} - 1 \right) D \right) x_k + b \end{cases} \tag{9}$$

Il faut encore résoudre à chaque étape un système triangulaire inférieur où la matrice L_ω n'est pas plus compliquée que L_1. Si

- $\omega = 1$, on retrouve la méthode de Gauss-Seidel.
- $\omega > 1$, la méthode est dite de sur -relaxation.
- $\omega < 1$, la méthode est dite de sous -relaxation.

→ La méthode de relaxation

On peut, dans certains cas, améliorer la convergence de l'algorithme de Gauss-Seidel en introduisant un paramètre que l'on pourra modifier en fonction de la matrice A de départ. Pour ce faire, on introduit un facteur $0 < \omega < 2$. La méthode de relaxation est une méthode entre la méthode de Jacobi et celle de Gauss-Seidel et correspond à la décomposition suivante, Pour $\omega \neq 0$, en posant

$$M = \frac{1}{\omega} D - E, \quad N = \left(\frac{1-\omega}{\omega} \right) D + F,$$

la matrice de la méthode est

$$L_\omega = \left(\frac{1}{\omega} D - E \right)^{-1} \left(F + \left(\frac{1}{\omega} - 1 \right) D \right).$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ \left(\frac{1}{\omega} D - E \right) x_{k+1} = \left(F + \left(\frac{1}{\omega} - 1 \right) D \right) x_k + b \end{cases} \tag{9}$$

Il faut encore résoudre à chaque étape un système triangulaire inférieur où la matrice L_ω n'est pas plus compliquée que L_1. Si

- $\omega = 1$, on retrouve la méthode de Gauss-Seidel.
- $\omega > 1$, la méthode est dite de sur -relaxation.
- $\omega < 1$, la méthode est dite de sous -relaxation.

→ La méthode de relaxation

On peut, dans certains cas, améliorer la convergence de l'algorithme de Gauss-Seidel en introduisant un paramètre que l'on pourra modifier en fonction de la matrice A de départ. Pour ce faire, on introduit un facteur $0 < \omega < 2$. La méthode de relaxation est une méthode entre la méthode de Jacobi et celle de Gauss-Seidel et correspond à la décomposition suivante, Pour $\omega \neq 0$, en posant

$$M = \frac{1}{\omega} D - E, \quad N = \left(\frac{1-\omega}{\omega} \right) D + F,$$

la matrice de la méthode est

$$L_\omega = \left(\frac{1}{\omega} D - E \right)^{-1} \left(F + \left(\frac{1}{\omega} - 1 \right) D \right).$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ \left(\frac{1}{\omega} D - E \right) x_{k+1} = \left(F + \left(\frac{1}{\omega} - 1 \right) D \right) x_k + b. \end{cases} \tag{9}$$

Il faut encore résoudre à chaque étape un système triangulaire inférieur où la matrice L_ω n'est pas plus compliquée que L_1. Si

- $\omega = 1$, on retrouve la méthode de Gauss-Seidel.
- $\omega > 1$, la méthode est dite de sur -relaxation.
- $\omega < 1$, la méthode est dite de sous -relaxation.

→ La méthode de relaxation

On peut, dans certains cas, améliorer la convergence de l'algorithme de Gauss-Seidel en introduisant un paramètre que l'on pourra modifier en fonction de la matrice A de départ. Pour ce faire, on introduit un facteur $0 < \omega < 2$. La méthode de relaxation est une méthode entre la méthode de Jacobi et celle de Gauss-Seidel et correspond à la décomposition suivante, Pour $\omega \neq 0$, en posant

$$M = \frac{1}{\omega} D - E, \quad N = \left(\frac{1-\omega}{\omega} \right) D + F,$$

la matrice de la méthode est

$$L_\omega = \left(\frac{1}{\omega} D - E \right)^{-1} \left(F + \left(\frac{1}{\omega} - 1 \right) D \right).$$

L'algorithme s'écrit alors

$$\begin{cases} x_0 \text{ donné} \\ \left(\frac{1}{\omega} D - E \right) x_{k+1} = \left(F + \left(\frac{1}{\omega} - 1 \right) D \right) x_k + b. \end{cases} \tag{9}$$

Il faut encore résoudre à chaque étape un système triangulaire inférieur où la matrice L_ω n'est pas plus compliquée que L_1. Si

- $\omega = 1$, on retrouve la méthode de Gauss-Seidel.
- $\omega > 1$, la méthode est dite de sur -relaxation.
- $\omega < 1$, la méthode est dite de sous -relaxation.

- **Convergence des méthodes itératives**

→ **Résultat général**

Theorème

Soit A une matrice symétrique définie positive. On suppose que

$$A = M - N, \text{ avec } M \text{ inversible.}$$

Si la matrice symétrique $M^t + N$ est définie positive alors $\rho(M^{-1}N) < 1$.

- **Convergence des méthodes itératives**

→ **Résultat général**

Theorème

Soit A une matrice symétrique définie positive. On suppose que

$$A = M - N, \text{ avec } M \text{ inversible.}$$

Si la matrice symétrique $M^t + N$ est définie positive alors $\rho(M^{-1}N) < 1$.

- **Convergence des méthodes itératives**

→ **Résultat général**

Theorème

Soit A une matrice symétrique définie positive. On suppose que

$$A = M - N, \text{ avec } M \text{ inversible.}$$

Si la matrice symétrique $M^t + N$ est définie positive alors $\rho(M^{-1}N) < 1$.

Démonstration.

- $M^t + N$ est symétrique :

$$M^t + N = A^t + N^t + N = A + N^t + N = M + N^t.$$

- Comme A est symétrique définie positive, l'application

$$v \in \mathbb{R}^n, \quad \|v\| = \langle v, Av \rangle^{\frac{1}{2}}$$

définit une norme sur $\mathbb{R}^n$. On considère alors la norme matricielle $\|\cdot\|$ induite par cette norme vectorielle

$$\|M^{-1}N\| = \|I - M^{-1}A\| = \sup_{\|v\|=1} \|v - M^{-1}Av\|.$$

Soit v un vecteur tel que $\|v\| = 1$. On pose $w = M^{-1}Av$,

$$\begin{aligned} \|v - w\|^2 = (v - w)^t A(v - w) &= 1 - w^t M^t w - w^t M w + w^t A w \\ &= 1 - w^t (M^t + M - A) w \\ &= 1 - w^t (M^t + N) w \end{aligned}$$

$v \neq 0 \Longrightarrow w = M^{-1}Av \neq 0 \Longrightarrow w^t(M^t + N)w > 0$ Donc si la matrice symétrique $M^t + N$ est définie positive, on a

$$\|v - w\|^2 = 1 - w^t(M^t + N)w < 1,$$

et donc $\rho(M^{-1}N) < 1$.

Démonstration.

- $M^t + N$ est symétrique :

$$M^t + N = A^t + N^t + N = A + N^t + N = M + N^t.$$

- Comme A est symétrique définie positive, l'application

$$v \in \mathbb{R}^n, \quad \|v\| = \langle v, Av \rangle^{\frac{1}{2}}$$

définit une norme sur $\mathbb{R}^n$. On considère alors la norme matricielle $\|\cdot\|$ induite par cette norme vectorielle

$$\|M^{-1}N\| = \|I - M^{-1}A\| = \sup_{\|v\|=1} \|v - M^{-1}Av\|.$$

Soit v un vecteur tel que $\|v\| = 1$. On pose $w = M^{-1}Av$,

$$\begin{aligned} \|v - w\|^2 = (v - w)^t A(v - w) &= 1 - w^t M^t w - w^t M w + w^t A w \\ &= 1 - w^t(M^t + M - A)w \\ &= 1 - w^t(M^t + N)w \end{aligned}$$

$v \neq 0 \Longrightarrow w = M^{-1}Av \neq 0 \Longrightarrow w^t(M^t + N)w > 0$ Donc si la matrice symétrique $M^t + N$ est définie positive, on a

$$\|v - w\|^2 = 1 - w^t(M^t + N)w < 1,$$

et donc $\rho(M^{-1}N) < 1$.

Proposition

On considère le système $Ax = b$. Si A est à diagonale strictement dominante, c'est-à-dire si

$$\sum_{i \neq j} |a_{ij}| < |a_{ii}|$$

pour tout $i = 1, ..., n$, alors la méthode de Jacobi converge pour tout point de départ $x^{(0)}$.

On peut montrer que cette condition suffit également pour assurer la convergence de l'algorithme de Gauss-Seidel.

Proposition

On considère le système $Ax = b$. Si A est à diagonale strictement dominante, c'est-à-dire si

$$\sum_{i \neq j} |a_{ij}| < |a_{ii}|$$

pour tout $i = 1, ..., n$, alors la méthode de Jacobi converge pour tout point de départ $x^{(0)}$.

On peut montrer que cette condition suffit également pour assurer la convergence de l'algorithme de Gauss-Seidel.

→ Convergence des méthodes de Gauss-Seidel et relaxation

Nous allons utiliser le résultat qui précède pour donner un critère de convergence pour la méthode de relaxation.

Theorème

Soit A une matrice symétrique définie positive.

$$0 < \omega < 2 \Longrightarrow \text{ La méthode de relaxation converge.}$$

Démonstration.

On applique le résultat précédent.
On a $A = M - N$ avec $M = \frac{1}{\omega}D - E$ et $N = N = \left(\frac{1-\omega}{\omega}\right)D + F$.

$$\begin{aligned} M^t + N &= \frac{1}{\omega}D - E^t + \left(\frac{1-\omega}{\omega}\right)D + F^t \\ &= \frac{1}{\omega}D + \left(\frac{1-\omega}{\omega}\right)D \\ &= \left(\frac{2-\omega}{\omega}\right)D \end{aligned}$$

$0 < \omega < 2 \Longrightarrow \left(\frac{2-\omega}{\omega}\right) > 0. \Longrightarrow \left(\frac{2-\omega}{\omega}\right)D$ est définie positive. Donc d'après le théorème (18) on a la convergence

→ Convergence des méthodes de Gauss-Seidel et relaxation

Nous allons utiliser le résultat qui précède pour donner un critère de convergence pour la méthode de relaxation.

Théorème

Soit A une matrice symétrique définie positive.

$$0 < \omega < 2 \Longrightarrow \text{ La méthode de relaxation converge.}$$

Démonstration.

On applique le résultat précédent.
On a $A = M - N$ avec $M = \frac{1}{\omega}D - E$ et $N = N = \left(\frac{1-\omega}{\omega}\right)D + F$.

$$\begin{aligned} M^t + N &= \frac{1}{\omega}D - E^t + \left(\frac{1-\omega}{\omega}\right)D + F^t \\ &= \frac{1}{\omega}D + \left(\frac{1-\omega}{\omega}\right)D \\ &= \left(\frac{2-\omega}{\omega}\right)D \end{aligned}$$

$0 < \omega < 2 \Longrightarrow \left(\frac{2-\omega}{\omega}\right) > 0. \Longrightarrow \left(\frac{2-\omega}{\omega}\right)D$ est définie positive. Donc d'après le théorème (18) on a la convergence.

→ Convergence des méthodes de Gauss-Seidel et relaxation

Nous allons utiliser le résultat qui précède pour donner un critère de convergence pour la méthode de relaxation.

Théorème

Soit A une matrice symétrique définie positive.

$$0 < \omega < 2 \Longrightarrow \text{ La méthode de relaxation converge.}$$

Démonstration.

On applique le résultat précédent.
On a $A = M - N$ avec $M = \frac{1}{\omega} D - E$ et $N = N = \left(\frac{1-\omega}{\omega}\right) D + F$.

$$\begin{aligned} M^t + N &= \frac{1}{\omega} D - E^t + \left(\frac{1-\omega}{\omega}\right) D + F^t \\ &= \frac{1}{\omega} D + \left(\frac{1-\omega}{\omega}\right) D \\ &= \left(\frac{2-\omega}{\omega}\right) D \end{aligned}$$

$0 < \omega < 2 \Longrightarrow \left(\frac{2-\omega}{\omega}\right) > 0. \Longrightarrow \left(\frac{2-\omega}{\omega}\right) D$ est définie positive. Donc d'après le théorème (18) on a la convergence

Le résultat ci-dessus est une condition suffisante de convergence. En fait c'est aussi une condition nécessaire comme le montre le résultat suivant :

Proposition

Soit A une matrice, alors pour tout $\omega \neq 0$, $\rho(L_\omega) \geq |\omega - 1|$.

Démonstration.

Soient $(\lambda_i)_i$ les valeurs propres de L_ω. Alors,

$$\prod_{i=1}^{n} \lambda_i = \det(L_\omega) = \frac{\det\left(\frac{1-\omega}{\omega} D + F\right)}{\det\left(\frac{1}{\omega} D - E\right)} = (1-\omega)^n.$$

$$\Longrightarrow \rho(L_\omega) \geq \left| \prod_{i=1}^{n} \lambda_i \right|^{\frac{1}{n}} = |1-\omega|.$$

□

Le résultat ci-dessus est une condition suffisante de convergence. En fait c'est aussi une condition nécessaire comme le montre le résultat suivant :

Proposition

Soit A une matrice, alors pour tout $\omega \neq 0$, $\rho(L_\omega) \geq |\omega - 1|$.

Démonstration.

Soient $(\lambda_i)_i$ les valeurs propres de L_ω. Alors,

$$\prod_{i=1}^{n} \lambda_i = \det(L_\omega) = \frac{\det\left(\frac{1-\omega}{\omega} D + F\right)}{\det\left(\frac{1}{\omega} D - E\right)} = (1-\omega)^n.$$

$$\Longrightarrow \rho(L_\omega) \geq \left| \prod_{i=1}^{n} \lambda_i \right|^{\frac{1}{n}} = |1-\omega|.$$

□

Le résultat ci-dessus est une condition suffisante de convergence. En fait c'est aussi une condition nécessaire comme le montre le résultat suivant :

Proposition

Soit A une matrice, alors pour tout $\omega \neq 0$, $\rho(L_\omega) \geq |\omega - 1|$.

Démonstration.

Soient $(\lambda_i)_i$ les valeurs propres de L_ω. Alors,

$$\prod_{i=1}^{n} \lambda_i = \det(L_\omega) = \frac{\det\left(\frac{1-\omega}{\omega} D + F\right)}{\det\left(\frac{1}{\omega} D - E\right)} = (1-\omega)^n.$$

$$\Longrightarrow \rho(L_\omega) \geq \left| \prod_{i=1}^{n} \lambda_i \right|^{\frac{1}{n}} = |1 - \omega|.$$

□

Corollaire

Si A est une matrice symétrique définie positive, alors

$$0 < \omega < 2 \Longleftrightarrow \text{La méthode de relaxation converge.}$$

Nous terminons cette section par un théorème de comparaison des méthodes de Jacobi et de Gauss-Seidel dans un cas particulier. Nous admettrons ce résultat.

Theorème

Soit A une matrice tridiagonale par blocs. Alors

$$\rho(L_1) = \rho(J)^2.$$

Les deux méthodes convergent ou divergent donc simultanément. La méthode de Gauss-Seidel va plus vite que la méthode de Jacobi.

Corollaire

Si A est une matrice symétrique définie positive, alors

$$0 < \omega < 2 \Longleftrightarrow \text{ La méthode de relaxation converge.}$$

Nous terminons cette section par un théorème de comparaison des méthodes de Jacobi et de Gauss-Seidel dans un cas particulier. Nous admettrons ce résultat.

Theorème

Soit A une matrice tridiagonale par blocs. Alors

$$\rho(L_1) = \rho(J)^2.$$

Les deux méthodes convergent ou divergent donc simultanément. La méthode de Gauss-Seidel va plus vite que la méthode de Jacobi.

Corollaire

Si A est une matrice symétrique définie positive, alors

$$0 < \omega < 2 \Longleftrightarrow \text{La méthode de relaxation converge.}$$

Nous terminons cette section par un théorème de comparaison des méthodes de Jacobi et de Gauss-Seidel dans un cas particulier. Nous admettrons ce résultat.

Theorème

Soit A une matrice tridiagonale par blocs. Alors

$$\rho(L_1) = \rho(J)^2.$$

Les deux méthodes convergent ou divergent donc simultanément. La méthode de Gauss-Seidel va plus vite que la méthode de Jacobi.