Séance 4 : résolution de systèmes linéaires par la méthode de Gauss (aspects théoriques et numériques)

Objectif : proposer un algorithme de résolution (peu coûteux et robuste) exacte ou approché, d'un système linéaire cramérien :

$$Ax = b \text{ avec } \begin{cases} A \in GL_n(\mathbb{R}) \\ b \in \mathbb{R}^n \end{cases}$$

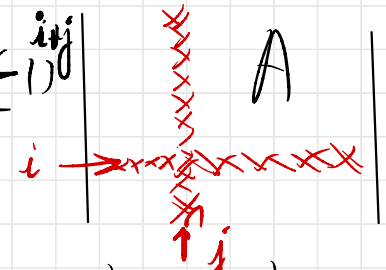
(la solution $x \in \mathbb{R}^n$ existe et est unique)

Ce problème se rencontre dans de nombreuses modélisations) éventuellement avec n grand.

On dispose, en théorie, de la méthode de Cramer vue précédemment :

$$x = \frac{1}{\det(A)} {}^t com(A) \cdot b$$

où $com(A)_{ij} = (-1)^{i+j}$



Cette méthode est exacte mais excessivement coûteuse : elle est basée sur :

* Le calcul d'un déterminant $n \times n$
* Le calcul de n^2 déterminants $(n-1)\times(n-1)$

soit environ :

$$\left.\begin{matrix} n! \text{ additions} \\ n!\,(n-1) \text{ multiplications} \end{matrix}\right\} + \left\{ n^2\,(n-1)! + n^2\,(n-1)!\,(n-2) \right\}$$

c'est à dire de l'ordre de

$$n^2 \cdot n! \text{ opérations } (+ \text{ et } \times)$$

Si $n = 100$, on doit effectuer environ

$$10^4 \cdot 100! \sim 10^4 \left(\frac{100}{e}\right)^{100} \sqrt{200\pi}$$

c'est à dire plus de 10^{104} opérations

Si on utilise un ordinateur effectuant 10^{12} opérations par seconde (1 Tflops), il faudrait 10^8 secondes, soit plus de 3 ans.... On ne peut utiliser cette méthode numériquement !!

Il est donc indispensable d'utiliser une autre méthode moins coûteuse, exacte ou approchée, et vérifier ensuite sa robustesse.

On présente ici la méthode (exacte) du pivot de Gauss

1) Description de la méthode

Le principe consiste à faire des opérations (échange, combinaison linéaire) sur les lignes du système linéaire, de telle sorte à se ramener à un système triangulaire du type

$$T x = \beta \quad \text{où}$$

$$T = \begin{pmatrix} * & \cdots & * \\ & \ddots & \vdots \\ 0 & & * \end{pmatrix}$$

La résolution d'un tel système s'effectue alors facilement par un principe de remontée : 3

$$\begin{cases} x_n = \dfrac{1}{t_{n,n}} b_n \\ x_{n-1} = \dfrac{1}{t_{n-1,n-1}} \left(b_{n-1} - t_{n-1,n} x_n \right) \\ \vdots \\ x_1 = \dfrac{1}{t_{1,1}} \left(b_1 - \dots \right) \end{cases}$$

Cette remontée nécessite de l'ordre 2 :

$$\begin{cases} \dfrac{(n-1)n}{2} & \text{additions} \\ \dfrac{n(n+1)}{2} & \text{multiplications} \end{cases}$$

soit n^2 opérations

(effectué en 10^{-8} s sur un ordinateur à 1 Tflops pour $n=100$)

La matrice triangulaire s'obtient en faisant apparaître progressivement des colonnes de "0" sous la diagonale de A.

On décrit en détail l'opération qui consiste à transformer le système initial :

$$Ax = b$$

en un système équivalent du type

$A_1 x = b_1$ où

pivot

$$A_1 = \begin{pmatrix} * & * & \cdots & * \\ 0 & & & \\ | & & A'_1 & \\ 0 & & & \end{pmatrix}, \quad b = \begin{pmatrix} * \\ b'_1 \end{pmatrix}$$

Pour cela, on définit la transformation

$$\begin{cases} A \longmapsto G(A) = A_1 \\ b \longmapsto G(b) = b_1 \end{cases}$$ en 2 étapes :

Etape 1 : choix du pivot par échange de lignes :

si $A = [a_{ij}]$, comme A est inversible

$\begin{pmatrix} a_{1,1} \\ \vdots \\ a_{n,1} \end{pmatrix} \neq 0$) on note

$$|a_{i_0,1}| = \max_{1 \le i \le n} (|a_{i,1}|)$$

pivot ($\neq 0$)

On échange alors les lignes 1 et i_0 de $\tilde{A}$ et $\tilde{b}$ ($\tilde{A}$ et $\tilde{b}$)

Etape 2 : opération sur les lignes L_2 à L_n pour faire apparaître un 0 en première position :

$$L_i \to L_i - \frac{\tilde{a}_{i,1}}{\tilde{a}_{1,1}} L_1$$

(à faire sur $\tilde{A}$ et $\tilde{b}$)

Cette opération de transformation de A en A_1 (et b en b_1) est repétée ensuite à A'_1 et b'_1 :

$$A'_1 = \begin{pmatrix} * & * & * & * \\ 0 & & & \\ | & & A'_2 & \\ 0 & & & \end{pmatrix}, \quad b'_1 = \begin{pmatrix} * \\ b'_2 \end{pmatrix}$$

puis successivement jusqu'à

$$A'_{n-2} = \begin{pmatrix} * & * \\ 0 & * \end{pmatrix}, \quad b'_{n-2} = \begin{pmatrix} * \\ * \end{pmatrix}$$

→ ceci est possible car A'_1 est inversible (avec le déterminant) et les résultats des étapes précédentes se conservent.

2) Justification mathématique

On vérifie qu'ainsi construite, la méthode donne un système équivalent.

* Échange de lignes : pour tout (i,j), $i \neq j$, on note T_{ij} la matrice de permutation :

$$T_{ij} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \\ \end{matrix}$$

(colonnes i et j indiquées au-dessus de la matrice)

Alors, $T_{ij} \in GL_n(\mathbb{K})$, $\det(T_{ij}) = -1$, et si $A \in \mathcal{M}_n(\mathbb{R})$,

$T_{ij}A$ = matrice A où lignes i et j échangées.

(multiplication à gauche)

* Combinaison linéaire (transvection)

si

$$E_1(\lambda_2, \dots, \lambda_n) = \begin{pmatrix} 1 & & & & 0 \\ \lambda_2 & \ddots & & & \\ \vdots & & \ddots & & \\ & 0 & & \ddots & \\ \lambda_n & & & & 1 \end{pmatrix}$$

alors

E_1 inversible, $\det(E_1) = 1$, et si $A \in \mathcal{M}_n(\mathbb{R})$)

$E_1(\lambda_2, \dots, \lambda_n) A$ = matrice A avec

$$\begin{cases} L_2 \to L_2 + \lambda_2 L_1 \\ \vdots \\ L_n \to L_n + \lambda_n L_1 \end{cases}$$

Ainsi,

$$A_1 = G(A) = E_1\left(-\frac{\tilde{a}_{21}}{\tilde{a}_{11}}, \dots, -\frac{\tilde{a}_{n1}}{\tilde{a}_{11}}\right) T_{1,1} A$$

$$b_1 = G(b) = \dots = \dots = b$$

Il s'agit donc bien d'un système équivalent.

Au final, $Tx = \beta$ est également un système équivalent à $Ax = b$.

Aucune condition autre que A inversible n'est nécessaire.

3) Coût et robustesse

Calcul du nombre d'opérations $(+, *)$ de la méthode (hors remontée) :

→ échange : $\ln i$ (en $n \log n$) négligeable.

* $A \to G(A)$:

1 division, n^2 mult., n^2 add. $\sim 2n^2$

* $A_1' \to G(A_1')$: $\sim 2(n-1)^2$

soit au total :

$$2\left(\sum_{k=1}^{n} k^2\right) = 2\,\frac{n(n+1)(2n+1)}{6}$$

$\sim 2n^3/3$ opérations

Le coût global de la méthode de Gauss est donc de l'ordre de $2n^3/3$ opérations (add. ou mult.)

En reprenant l'exemple $n=100$, le système est résolu en 10^{-6} s !!

On a pu montrer que l'exposant 3 n'est certes pas optimal ($\alpha_{opt} \approx 2{,}8$) mais aucune méthode n'est aussi "simple" que la méthode de Gauss. Seulement par certaines matrices particulières ($A \in \mathcal{S}_n^{++}(\mathbb{R})$), il est possible d'améliorer ce résultat de manière avantageuse (Cholesky).

* Etude de robustesse

Grâce à la stratégie de pivot, la méthode de Gauss est stable numériquement, c'est à dire que les approximations sur les opérations n'engendrent pas d'erreur trop importante sur le résultat final. (on évite les divisions par un pivot petit).

A l'inverse, si on choisit seulement n'importe quel pivot non nul,

La méthode de Gauss peut devenir instable : voir exemple TD2 :

$$A = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Si on suppose que l'ordinateur travaille avec un précision 10^{-3} :
(par exemple : $1,0003 \longrightarrow 1$)

* Gauss avec stratégie du pivot :

→ échange $\begin{pmatrix} 1 & 1 \\ 10^{-4} & 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

→ transvection : $(L_2 \leftarrow L_2 - 10^{-4} L_1)$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$1 - 10^{-4} \cdot 1$; $1 - 10^{-4} \cdot 2$

→ remontée :

$$OC = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

* Gauss sans stratégie de pivot

→ transvection :

$$\begin{pmatrix} 10^{-4} & 1 \\ 0 & -10^4 \end{pmatrix} \quad \begin{pmatrix} 1 \\ -10^4 \end{pmatrix}$$

$(L_2 \leftarrow L_2 - 10^4 L_1)$; $1 - 10^4 \cdot 1$; $2 - 10^4 \cdot 1$

remontée $\rightsquigarrow x = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ /

$(-9999 = -0{,}9999\ E^4 = -1E^4)$

* La solution exacte est :

$$x = \begin{pmatrix} \dfrac{1}{1-10^{-4}} \\ 2 - \dfrac{1}{1-10^{-4}} \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$\rightarrow$ la méthode sans stratégie de pivot échoue ...

4) Implémentation Python

(script à télécharger sur site web)

On observe :

$\rightarrow$ la vitesse de l'algorithme ($n = 100, \sim \ldots 1000$)

$\rightarrow$ la non robustesse en cas d'absence de stratégie de pivot (sur l'exemple précédent avec 10^{-18})

$\rightarrow$ l'échec du calcul ($n = 12$) par la matrice de Hilbert :

$$H = \left[\frac{1}{i+j-1} \right] \ (\in \mathcal{S}_n^{++})$$

5) Notion de conditionnement numérique du problème

On observe que le système :

$Ax = b$ avec

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \text{ et } b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$$

a pour solution exacte :

$u = {}^t(1, \dots, 1)$

alors que le système

$Ax = b'$ avec $b' = \begin{pmatrix} 32,1 \\ 22,9 \\ 33,1 \\ 30,9 \end{pmatrix}$

a pour solution exacte :

$$u = \begin{pmatrix} 9,2 \\ -12,6 \\ 4,5 \\ -1,1 \end{pmatrix}$$

On peut faire la même observation (forte variation de la solution exacte avec une petite modification des coefficients) avec la matrice de Hilbert.

$\neq 0,33333\dots 3$

$$H = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \vdots & & & & \\ \frac{1}{n} & \cdots & \cdots & \cdots & \frac{1}{2n-1} \end{pmatrix}$$

Avant tout calcul de solution, il est donc impératif d'estimer le conditionnement de la matrice vis à vis de la résolution d'un système linéaire.

Dans le cas présent, il existe une grandeur associée à la matrice qui mesure le conditionnement de A :

Def : soit $A \in GL_n(\mathbb{R})$ et $|||\ |||$ une norme subordonnée sur $\mathcal{M}_n(\mathbb{R})$:

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

(à voir ultérieurement)

On note $cond(A) = |||A|||.|||A^{-1}|||$ le conditionnement de A par $|||.|||$.

Proposition :

* $cond(A) \geq 1$, $cond(\lambda A) = cond(A)$
* si $Au = b$ et $Au = b'$ alors

$$\frac{\|u-u'\|}{\|u\|} \leq cond(A) \left(\frac{\|b-b'\|}{\|b\|}\right)$$

erreur — amplification

* si $Au = b$ et $A'u' = b$, alors

$$\frac{\|u-u'\|}{\|u\|} \leq cond(A) \frac{|||A-A'|||}{|||A|||} (1 + O(|||A-A'|||))$$

(preuve : voir poly algèbre linéaire)