Algorithmique Numérique

saida.bouakaz@univ-lyon1.fr

Plan du Cours

- **Rappel sur les matrice**
  - Définitions
  - Opérations sur les matrices
  - Déterminant & méthode de cramer

- **Résolution de système linéaire**
  - Méthodes directes
    - Triangulation de Gauss
    - Décomposition LU
  - Méthodes itératives
    - Méthode de Jacobi
    - Méthode de Seidel
- **Racines de fonctions F(x)=0**
  - Introduction
  - Méthode de Newton
  - Méthode de la sécante
  - Méthode de dichotomie

- **Interpolation**
  - Interpolation linéaire et quadratique
  - Formule de Lagrange, polynôme de Newton,
  - Différences finis
  - Splines

- **Approximation polynomiale**
  - Méthode des moindres carrés, moindres carrés pondérées
  - Polynômes de Chebychev

- **Intégration numérique**
  - Introduction
  - Méthode des trapèzes
  - Méthode de Simpson
  - Méthodes améliorées

Chapitre 2
Résolution de systèmes linéaires

Résoudre : $A\ X = B$ avec A matrice $(n \times n)$;
$X = (x_1 \quad \dots \quad x_i \ \dots \quad x_n)^t$; $\quad B = (b_1 \quad \dots \quad b_i \ \dots \quad b_n)^t$

- Méthode du pivot de Gauss: basée sur la triangulation
- Méthode de factorisation : LU
- Méthodes itératives

Méthode de Gauss

Méthode de résolution directe

- Idée : transformer le système en un système triangulaire
- Comment : par une suite de combinaisons linaires entre les différentes lignes
- Spécificité : travaille sur la matrice élargie $[A \mid B]$
- Mode : on opère par étape
  - ► $AX = B \Leftrightarrow A^{(k)} = B^{(k)}$
  - ► Arrêt : matrice finale triangulaire.

Complexité

- Complexité de la résolution du système triangulaire en $O(n^2)$:
- Complexité de la triangulation en $O(n^3)$:

Méthode de Gauss

▶ $AX = B \Leftrightarrow A^{(k)} = B^k$

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

En utilisant la méthode de Gauss, l'objectif est de résoudre ce système en une fois.

en opposition aux méthodes itératives (prochain chapitre) pour lesquelles on répète des opérations jusqu'à ce que le résultat converge vers une solution.

Méthode de Gauss

1. Procédé du pivot avec normalisation de la diagonale

Le principe consiste à transformer le système $A\ X = B$

en un système triangulaire équivalent. Pour triangulariser, il faut éliminer des cœfficients sous la diagonale de la matrice. A l'étape i

$$\begin{bmatrix} a^*_{11} & a^*_{12} & \cdots & a^*_{1n} \\ & a^*_{22} & \cdots & a^*_{11} \\ & & \ddots & \vdots \\ & & & a^*_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b^*_1 \\ b^*_2 \\ \vdots \\ b^*_n \end{bmatrix}$$

La solution se calcule par remontée.

- La transformation de A en A^* se compose de deux étapes itérées n fois.

A l'étape *i* :

normalisation : on divise la ligne i par $a_{i;i}$ ($a_{i;i} \neq 0$ le *pivot*)

si $a_{ii} \neq 0$ pour obtenir $a_{ii} = 1$,

annulation sous la diagonale : pour $i + 1 = k \rightarrow n$,

on soustrait la ligne du pivot multipliée par a_{ki} à la ligne k pour obtenir $a_{ki} = 0$

Méthode de Gauss - form

Procédé du pivot sans normalisation de la diagonale

On garde le principe de transformer le système A X = B en un système équivalent.

On travaille tjrs avec la matrice élargie.

Première étape

On pose : $m_{i1} = \dfrac{a_{i1}}{a_{11}}$ pour $1 \leq i \leq n$ d'où

$$
\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\
 & & (a_{22} - \boxed{m_{21}}a_{12})x_2 & + & \cdots & + & (a_{2n} - \boxed{m_{21}}a_{1n})x_n & = & b_2 - \boxed{m_{21}}b_1 \\
 & & \vdots & & & & \vdots & & \\
 & & (a_{i2} - \boxed{m_{i1}}a_{12})x_2 & + & \cdots & + & (a_{in} - \boxed{m_{i1}}a_{1n})x_n & = & b_i - \boxed{m_{i1}}b_1 \\
 & & \vdots & & & & \vdots & & \\
 & & (a_{n2} - \boxed{m_{n1}}a_{12})x_2 & + & \cdots & + & (a_{nn} - \boxed{m_{n1}}a_{1n})x_n & = & b_n - \boxed{m_{n1}}b_1
\end{array}
$$

A l'issue de la première transformation, la matrice du nouveau système est

$$A^{(2)}X = B^{(2)}$$

$$A^{(2)} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \boxed{0} & a_{22} - \boxed{m_{21}}a_{12} & \cdots & a_{2n} - \boxed{m_{21}}a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{i2} - \boxed{m_{i1}}a_{12} & \cdots & a_{in} - \boxed{m_{i1}}a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2} - \boxed{m_{n1}}a_{12} & \cdots & a_{nn} - \boxed{m_{n1}}a_{1n} \end{pmatrix}$$

le second membre est

$$B^{(2)} = \begin{pmatrix} b_1 \\ b_2 - \boxed{m_{21}}b_1 \\ \vdots \\ b_i - \boxed{m_{i1}}b_1 \\ \vdots \\ b_n - \boxed{m_{n1}}b_1 \end{pmatrix}$$

Le nouveau système s'écrit :

$$A^{(k)}X = B^{(k)}$$

À l'étape k, on a

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

Cas générique : à l'étape k

Étape $(K\text{-}1)$: $A^{(K-1)} X = B^{(K-1)}$

$$A^{(k-1)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1k-1}^{(0)} & a_{1k}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2k-1}^{(1)} & a_{2k}^{(1)} & \cdots & a_{2n}^{(1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{k-1}^{(k-2)} & a_{k-1\,k}^{(k-2)} & \cdots & a_{k-1\,n}^{(k-2)} \\ 0 & 0 & \cdots & 0 & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & 0 & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{bmatrix} \qquad B^{(k-1)} = \begin{bmatrix} b_1^{(0)} \\ b_2^{(1)} \\ \vdots \\ b_{k-1}^{(k-2)} \\ b_k^{(k-1)} \\ \vdots \\ b_n^{(k-1)} \end{bmatrix}$$

Expression analytique On pose : $m_{ik} = \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \qquad i = k+1, \dots, n$

pour $\begin{cases} i = k+1, \dots, n \\ j = k+1, \dots, n \end{cases}$ on a $\begin{cases} a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \\ b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)} \end{cases}$

Par ligne : $l_{ij}^{(k+1)} = l_{ij}^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} l_{kj}^{(k)}$

Algorithme de triangulation sans normalisation de la diagonale

pour i de 1 à $n-1$ **faire**

  déterminer j tel que $a_{j\,i} = \sup\limits_{i \le k \le n} |a_{k\,i}|$

  permuter ligne i et ligne j sur A et sur B

  $L_i, L_j \leftarrow L_j, L_i$

  pour k de $i+1$ à n **faire**

    $L_k \leftarrow L_k - \dfrac{a_{k\,i}}{a_{i\,i}}.L_i$

  fin pour

fin pour

Pivot nul

- Si $a_{kk}^{(k)} = 0$ on cherche $a_{lk}^{(k)} \neq 0$ deux possibilités :
  • méthode du pivot partiel : permutation des lignes (ligne k et ligne l) méthode simple
  • méthode du pivot total : permutation des lignes et des colonnes méthode plus robuste

Ecrire l'algorithme avec pivot partiel sur les lignes

Méthode de Gauss

Pivot partiel sur les lignes

pour i de 1 à $n-1$ **faire**

  déterminer j tel que $a_{j\,i} = \sup_{i \le k \le n} |a_{k\,i}|$

  permuter ligne i et ligne j sur A et sur B

  $L_i, L_j \leftarrow L_j, L_i$

  pour k de $i+1$ à n **faire**

    $L_k \leftarrow L_k - \frac{a_{k\,i}}{a_{i\,i}}.L_i$

  fin pour

fin pour

- Algorithme de résolution (après triangulation de la matrice)

Ecrire l'algorithme

Pivot de gauss : technique pratique pour inverser une matrice

Technique : elle s'appuie sur : $A . A^{-1} = I$

- la matrice A et la matrice identité I sont juxtaposée (on parle de matrice augmentée $[A|I]$
- On applique une série de transformation aux ligne de façon à obtenir une matrice identité à la place de A, la matrice situé à droite sera la matrice inverse → $[A. A^{-1} | A^{-1} .I]$
- La méthode du pivot de gauss permet d'obtenir cette matrice

Méthode de factorisation *LU* (ou LR)

- Méthode : basée sur une factorisation A
- Le principe de cette méthode de recherche de solution consiste à décomposer

la matrice A sous forme d'un produit A = L . U

A=L .U → (L . U) X= B

A=L .U → L .(U X)= B si on pose UX=Y

AX= B $\Rightarrow \begin{cases} LY = B \\ UX = Y \end{cases}$

Méthode de factorisation *LU* (ou LR)

Si on peut décomposer la matrice A en le produit de 2 matrices A=L .U (ou A= L.R)

- L : Triangulaire inférieure (L pour Lower triangular matrix)
- U : Triangulaire supérieure (U pour Upper triangular matrix)

- $AX = B \Leftrightarrow (L.U)X = B \Leftrightarrow L.(UX) = B$
- On pose $UX = Y$ d'où $LY = B$

3 étapes :

1. Trouver les matrices L et U
2. Résolution du système $LY = B$ (L triangulaire inférieure)
3. Résolution du système $UX = Y$ (U triangulaire supérieure)

Remarque

LR : L pour Left triangular matrix et R pour Right triangular matrix

- L est une matrice triangulaire inférieure avec diagonale unité
- U est une matrice triangulaire supérieure.

- On utilisera la méthode LU lorsque l'on veut résoudre une famille de systèmes de la forme

$$A \, . \, X = B_i$$

- où seul le vecteur B_i (les données) varie, le modèle (matrice A) reste la même. le calcul de L et R est totalement indépendant de B

Comment déterminer L et U et quelle est la complexité de la décomposition (en ?? opérations).

- Deux méthodes :
  - décomposition de Gauss
  - Algorithme de Crout (identification)

Représentation matricielle de l'élimination de Gauss

$$AX = B \Rightarrow \begin{cases} LY = B \\ UX = Y \end{cases}$$

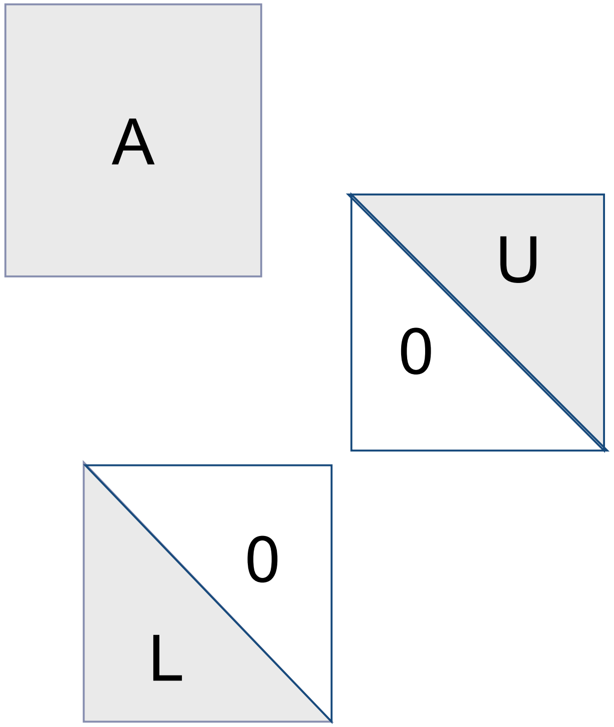
Rappelle : à chaque étape de l'algorithme de gauss...

$$\text{pour} \quad i = k+1,\ldots,n \quad \begin{cases} a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} \quad \text{pour } j = k+1,\ldots,n \\ b_i^{(k+1)} \leftarrow b_i^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)} \end{cases}$$

$$\text{notation matricielle : } A^{(k+1)} = M^{(k)} A^{(k)}; \quad b^{(k+1)} = M^{(k)} b^{(k)};$$

LU : principe

Il est si facile le résoudre un système « triangulaire » !



$$A = LU$$

$$Ax = b \Leftrightarrow \begin{cases} (1) \quad Ly = b \\ (2) \quad Ux = y \end{cases}$$

Comment construire Let U ?

idée :

reprendre l'étape de triangularisation de la méthode de Gauss

De Gauss à LU (ou LR)

Représentons une étape de la triangularisation par la multiplication de *A* par une matrice $M^{(k)}$

$$A^{(k+1)} = M^{(k)} A^{(k)} \qquad A^{(1)} = A \quad \text{et} \quad A^{(n)} = U$$

$$\boxed{\ell_{i,k} = \frac{a_{ik}}{a_{kk}} = -m_{i,k}}$$

$$\begin{cases} a_{ij} \leftarrow a_{ij} - \dfrac{a_{ik}}{a_{kk}} a_{kj} \\ b_i \leftarrow b_i - \dfrac{a_{ik}}{a_{kk}} b_k \end{cases}$$

$$M^{(k)} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \ddots & -\ell_{k+1,k} & \ddots & \vdots \\ 0 & \cdots & -\ell_{n,k} & 0 & 1 \end{pmatrix}$$

$$U = M^{(n-1)} \ldots M^{(k)} \ldots M^{(1)} A = MA$$

$$A = M^{-1} U = LU$$

$$\text{donc} \quad L = M^{-1}$$

LU : récapitulatif

Les matrices élémentaires $M^{(k)}$ sont **in**versibles
et leurs inverses sont les matrices $L^{(k)}$ triangulaires inférieures
telles que :

$$L^{(k)} = \begin{cases} l_{ii} = 1 & i = 1,n \\ l_{ik} = \ell_{ik} & i = k+1,n \\ l_{ij} = 0 & \text{sinon} \end{cases}$$

$$L^{(k)} = I - \left(M^{(k)} - I\right)$$

$$M^{(k)} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \ddots & -\ell_{k+1,k} & \ddots & \vdots \\ 0 & \cdots & -\ell_{n,k} & 0 & 1 \end{pmatrix}$$

$$L^{(k)} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ell_{k+1,k} & \ddots & \vdots \\ 0 & \cdots & \ell_{n,k} & 0 & 1 \end{pmatrix}$$

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

C'est la matrice l_{ik}

Exemple

$$A=\begin{pmatrix} 1 & 1 & -1 & 2 \\ -1 & 2 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 2 \end{pmatrix}$$

La décomposition de A=LU donne :

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ -1 & 2 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 2 \end{pmatrix} = \overbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & \frac{-1}{3} & 1 & 0 \\ 1 & \frac{-2}{3} & \frac{1}{2} & 1 \end{pmatrix}}^{L} \times \overbrace{\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 3 \end{pmatrix}}^{U}$$

Détail de la décomposition

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ -1 & 2 & 1 & 1 \\ 1 & 0 & 1 & -1 \\ 1 & -1 & 0 & 2 \end{pmatrix}$$

Pivot =1

lig2 ←lig2 -(-1)lig1

lig3 ←lig3 -(1)lig1

lig4 ←lig4 -(1)lig1

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & -1 & 2 & -3 \\ 0 & -2 & 1 & 0 \end{pmatrix}$$

Pivot =3

lig3 ←lig3 -(-1/3)lig2

lig4 ←lig4 -(-2/3)lig2

Etape 2

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

Pivot =2

lig4 ←lig4 -(1/2)lig3

$$\begin{pmatrix} 1 & 1 & -1 & 2 \\ 0 & 3 & 0 & 3 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

L’algorithme de décomposition

Fonction L,U = décompose(A)

```
pour k = 1 jusqu'à n − 1
    pivot ← a_kk          (* stratégie de pivot *)
    si pivot ≠ 0 alors
        ℓ_kk ← 1
        pour i = k + 1 jusqu'à n
            ℓ_ik ← a_ik / pivot
            pour j = k + 1 jusqu'à n
                a_ij ← a_ij − ℓ_ik a_kj
            fait
        fait
    sinon "problème"
fait
```

$$\begin{array}{l}
\text{pour } k = 1 \text{ jusqu'à } n-1 \\
\quad pivot \leftarrow a_{kk} \qquad \text{(* stratégie de pivot *)} \\
\quad \text{si } pivot \neq 0 \text{ alors} \\
\qquad \ell_{kk} \leftarrow 1 \\
\qquad \text{pour } i = k+1 \text{ jusqu'à } n \\
\qquad\quad \ell_{ik} \leftarrow \dfrac{a_{ik}}{pivot} \\
\qquad\quad \text{pour } j = k+1 \text{ jusqu'à } n \\
\qquad\qquad a_{ij} \leftarrow a_{ij} - \ell_{ik} a_{kj} \\
\qquad\quad \text{fait} \\
\qquad \text{fait} \\
\quad \text{sinon "problème"} \\
\text{fait}
\end{array}$$

Calcul des matrice L et U (ou L et R) par identification : Algorithme de Crout

Pour calculer L et U, il suffit de remarquer que

$$\begin{pmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{pmatrix} \times \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{pmatrix}$$

$$= \begin{pmatrix} \boxed{u_{1,1}} & \boxed{u_{1,2}} & \boxed{u_{1,3}} \\ \boxed{l_{2,1}}u_{1,1} & l_{2,1}u_{1,2} + \boxed{u_{2,2}} & l_{2,1}u_{1,3} + \boxed{u_{2,3}} \\ \boxed{l_{3,1}}u_{1,1} & l_{3,1}u_{1,2} + \boxed{l_{3,2}}u_{2,2} & l_{3,1}u_{1,3} + l_{3,2}u_{2,3} + \boxed{u_{3,3}} \end{pmatrix}$$

En prenant les équations obtenues dans le bons ordres (les colonnes de gauche à droite et les lignes de haut en bas) on remarque que l'on obtient un système à résoudre où à chaque étape, il n'y a qu'une seule inconnue.

$U_{11}=a_{11}$; $u_{12}=a_{12}$; $u_{13}=a_{13}$; $l_{21}=a_{21}/u_{11}$; $l_{31}= a_{31}/u_{11}$

$u_{22}=a_{22} - l_{21} u_{12}$; $l_{32}= (a_{31}-l_{31} u_{12}) /u_{22}$; $u_{33} = a_{33}-l_{31}u_{13} -l_{32}u_{23}$

Algorithme de Crout

```
pour j de 1 à n  faire
  pour i de 1 à j  faire          // Calcul des r_{i,j}
    r_{i,j} ← a_{i,j}
    pour k de 1 à i−1  faire
      r_{i,j} ← r_{i,j} − l_{i,k} r_{k,j}
    fin pour
  fin pour      pour i de j+1 à n  faire          // Calcul des l_{i,j}
    l_{i,j} ← a_{i,j}
    pour k de 1 à j−1  faire
      l_{i,j} ← l_{i,j} − l_{i,k} r_{k,j}
    fin pour
      l_{i,j} ← l_{i,j}/r_{j,j}
  fin pour
fin pour
```