Factorisation LU

TP2 - Mathématiques en technologie de l'information

Antoine Sutter
Julien Seemüller

Supervisé par : Dr. Eggenberg

06.05.2020

Index

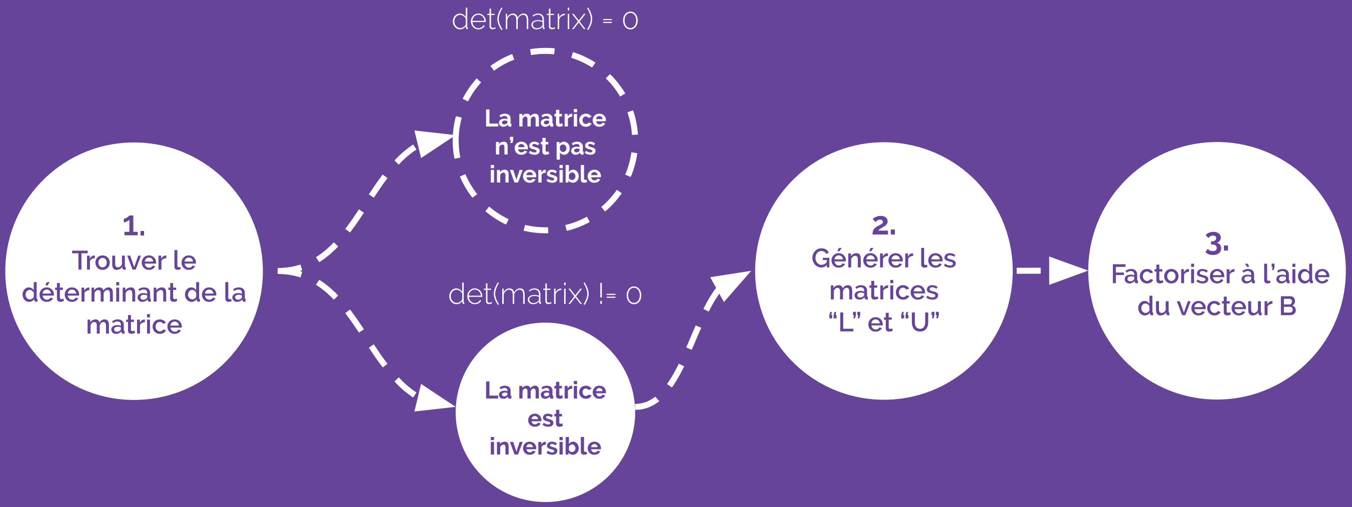
- Introduction et objectif
- Raisonnement
- Comment trouver “x” ?
- Détail sur algorithmes
- Présentation des résultats
- Difficultés rencontrées
- Améliorations envisageable

Objectif

résoudre un **systèmes d'équations** à "n" inconnues
a l'aide de **matrices** et la **factorisation LU**

Utilisation de **Python 3** et de la librairie **numpy**

Raisonnement



Comment trouver “x” ?

$$
\begin{pmatrix}
77 \cdot r_1 + 339 \cdot r_2 + [...] \\
95 \cdot r_1 + 415 \cdot r_2 + [...] \\
7 \cdot r_1 + 250 \cdot r_2 + [...] \\
80 \cdot r_1 + 350 \cdot r_2 + [...] \\
89 \cdot r_1 + 391 \cdot r_2 + [...] \\
28 \cdot r_1 + 123 \cdot r_2 + [...] \\
62 \cdot r_1 + 274 \cdot r_2 + [...] \\
36 \cdot r_1 + 160 \cdot r_2 + [...] \\
20 \cdot r_1 + 89 \cdot r_2 + [...] \\
6 \cdot r_1 + 27 \cdot r_2 + [...]
\end{pmatrix}
*
\begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix}
=
\begin{pmatrix} 61 \\ -74 \\ -28 \\ -73 \\ -3 \\ 27 \\ -94 \\ -11 \\ -89 \\ 76 \end{pmatrix}
$$

$$\mathbf{A}_{10x10} \qquad \vec{\mathbf{x}} \qquad \vec{\mathbf{b}}$$

Comment trouver “x” ?

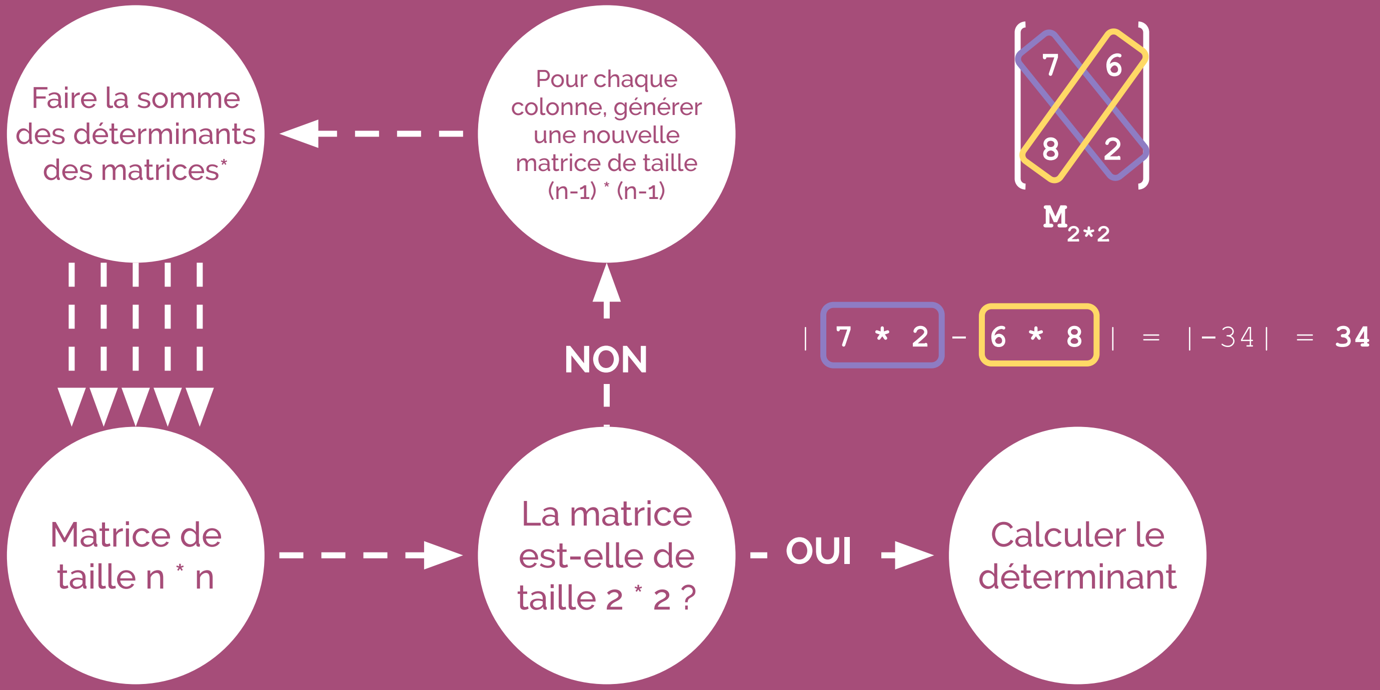
$$A_{10x10} = L * U \quad => \quad L * U * \vec{x} = \vec{b}$$

$$L * \underbrace{\left(U * \vec{x} \right)}_{\vec{y}} = \vec{b} \quad => \quad U * \vec{x} = \vec{y}$$

Donc ...

on trouve **x** avec **U * x = y**

Déterminant



Performance

- Calculer le déterminant <u>est plus lent</u> que de créer les matrices L et U
- On peut trouver le déterminant de manière plus efficace grâce à la technique suivante :

$$\underset{\mathbf{U}}{\begin{bmatrix} \underline{\mathbf{3}} & 9 & 4 \\ 0 & \underline{\mathbf{2}} & 5 \\ 0 & 0 & \underline{\mathbf{2}} \end{bmatrix}} \Rightarrow \quad 3 * 2 * 2 = \mathbf{12}$$

On multiplie simplement **chaque élément** de la **diagonale de “U”**
Le **déterminant** de la matrice **“A”** est **“12”**

Pivot de Gauss (trouver L & U)

On initialise deux matrices :

$L = I_{NxN}$

$U = A_{NxN}$

On va effectuer des **opérations sur "U"** pour essayer de la **transformer** en matrice **"upper"** (zéros partout sauf triangle supérieur)

Toutes les **opérations effectuées** sont **stockées** et répercutées dans **L**

De manière à ce que

$$A = U * L$$

soit **toujours vrai**

Pivot de Gauss (trouver L & U)

On parcourt la partie **inférieure à la diagonale** du tableau **U**

$$\begin{bmatrix} 4 & 3 & 2 \\ 2 & 4 & 0 \\ 1 & 2 & 3 \end{bmatrix}$$

U

On crée le coefficient **R** a l'aide de la **diagonale au dessus du chiffre actuel**

On **inverse le signe** de ce coefficient, puis on **stocke dans L**

$$\begin{bmatrix} 4 & 3 & 2 \\ 2 & 4 & 0 \\ 1 & 2 & 3 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{4} & 1 & 0 \end{bmatrix}$$

U **L**

R = -1 / 4

On **applique ce coefficient** a U à notre position

De cette manière, on obtient un "0"

$$\begin{bmatrix} 4 & 3 & 2 \\ 2 & 4 & 0 \\ 0 & 2 & 3 \end{bmatrix}$$

U

$1 + -\frac{1}{4} * 4 = 0$

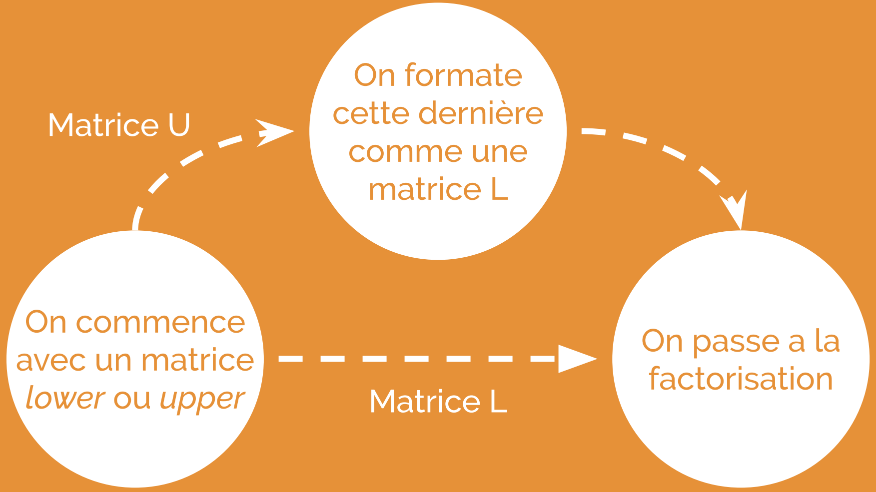
On **répercute ce changement** sur le reste de la matrice **U**

De manière à ce que

A = U * L

soit **toujours vrai**

Résoudre les équations L et U



Une matrice *lower* (L) contient toujours la solution de x_1 au même emplacement

Dans notre exemple : $x_1 = 8/4$ **donc** $x_1 = 2$

$$\Rightarrow \underbrace{\begin{bmatrix} 4 & 0 & 0 \\ 1 & 2 & 0 \\ 2 & 3 & 4 \end{bmatrix}}_{L_{3x3}} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} 8 \\ 3 \\ 5 \end{bmatrix}}_{\vec{b}}$$

Il ne reste qu'à **injecter** x_1 dans **la ligne suivante** et continuer jusqu'à la fin de la matrice

Présentation des résultats

$$
\underbrace{\begin{pmatrix}
77 \cdot r_1 + 339 \cdot r_2 + [...] \\
95 \cdot r_1 + 415 \cdot r_2 + [...] \\
7 \cdot r_1 + 250 \cdot r_2 + [...] \\
80 \cdot r_1 + 350 \cdot r_2 + [...] \\
89 \cdot r_1 + 391 \cdot r_2 + [...] \\
28 \cdot r_1 + 123 \cdot r_2 + [...] \\
62 \cdot r_1 + 274 \cdot r_2 + [...] \\
36 \cdot r_1 + 160 \cdot r_2 + [...] \\
20 \cdot r_1 + 89 \cdot r_2 + [...] \\
6 \cdot r_1 + 27 \cdot r_2 + [...]
\end{pmatrix}}_{\mathbf{A}_{10x10}}
*
\underbrace{\begin{bmatrix}
93,53 \\ 235,21 \\ -59,33 \\ 161,12 \\ -1,57 \\ -139,29 \\ 83,74 \\ 16,70 \\ -57,94 \\ -130,08
\end{bmatrix}}_{\vec{x}}
=
\underbrace{\begin{bmatrix}
61 \\ -74 \\ -28 \\ -73 \\ -3 \\ 27 \\ -94 \\ -11 \\ -89 \\ 76
\end{bmatrix}}_{\vec{b}}
$$

Difficultés rencontrées

- Utilisation implicite des types int / float de python
- Retourner le déterminant en valeur absolue
- Transformation d’une matrice Upper en matrice Lower

Améliorations envisageable

- Calcul intelligent du déterminant
  - On sélectionne la ligne ayant le plus de zéros pour éviter certains calculs
- Diagonalisation de la matrice
- Éviter de calculer le déterminant et tester si division par 0

Questions ?