

Cours CN5

Résolution d'un système d'équations algébriques linéaires

D.Malka

MPSI 2018-2019

Sommaire

- 1 Rappel : système linéaire
- 2 Résolution d'un système linéaire : pivot de Gauss
- 3 Implémentation « maison » du pivot de Gauss
- 4 Fonction prédéfinie de la bibliothèque `numpy`

Sommaire

- 1 Rappel : système linéaire
- 2 Résolution d'un système linéaire : pivot de Gauss
- 3 Implémentation « maison » du pivot de Gauss
- 4 Fonction prédéfinie de la bibliothèque `numpy`

Système linéaire

Exemple :

$$\begin{cases} 2x + 2y - 3z = 2 & L_1 \\ -2x - y - 3z = -5 & L_2 \\ 6x + 4y + 4z = 16 & L_3 \end{cases}$$

Représentation matricielle

Représentation matricielle :

$$AX = Y$$

avec :

$$X = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad Y = \begin{pmatrix} 2 \\ -5 \\ 16 \end{pmatrix}$$
$$A = \begin{pmatrix} 2 & 2 & -3 \\ -2 & -1 & -3 \\ 6 & 4 & 4 \end{pmatrix}$$

Sommaire

- 1 Rappel : système linéaire
- 2 Résolution d'un système linéaire : pivot de Gauss**
- 3 Implémentation « maison » du pivot de Gauss
- 4 Fonction prédéfinie de la bibliothèque `numpy`

Principe du pivot de Gauss

Soit, par exemple le système linéaire suivant :

$$\begin{cases} 2x + 2y - 3z = 2 & L_1 \\ -2x - y - 3z = -5 & L_2 \\ 6x + 4y + 4z = 16 & L_3 \end{cases}$$

But : déterminer un système triangulaire ■' (trivial à résoudre par substitutions successives) *équivalent* à ■.

L'algorithme du pivot de Gauss détermine ■' et la(les) solution(s) ou la non existence d'une solution de façon déterministe.

Principe du pivot de Gauss

Principe du pivot de Gauss

D'abord, triangulariser le système d'équations.

Puis, résoudre de façon triviale le système d'équation

Principe du pivot de Gauss

Triangularisation du système d'équations

- ▶ Réaliser des *transvections* sur les lignes L_i du système.
- ▶ Les *transvections* servent à éliminer l'occurrence d'une variables dans les lignes L_j .
- ▶ Elles sont réalisées à partir d'un *pivot*.
- ▶ Les *transvections* sur les lignes sont des combinaisons du type $L'_i = L_i + \lambda L_j$.
- ▶ Les *transvections* s'appuie le sur le choix d'un *pivot*
- ▶ Le *pivot* est un des coefficients de la ligne L_j utilisée pour réaliser les transvections.
- ▶ **Le *pivot* doit être non nul**

Principe du pivot de Gauss

Exemple

Exécutons l'algorithme du pivot de Gauss sur le système ■ suivant :

$$\begin{cases} 2x & +2y & -3z & = 2 & L_1 \\ -2x & -y & -3z & = -5 & L_2 \\ 6x & +4y & +4z & = 16 & L_3 \end{cases}$$

A chaque étape, le **pivot** est indiqué en **rouge**.

Principe du pivot de Gauss

Exemple

1^{ère} transvection :

$$\begin{cases} 2x & +2y & -3z & = 2 & L_1 \\ -2x & -y & -3z & = -5 & L_2 \\ 6x & +4y & +4z & = 16 & L_3 \end{cases}$$

Principe du pivot de Gauss

Exemple

1^{ère} transvection :

$$\left\{ \begin{array}{l} 2x \quad +2y \quad -3z \quad = 2 \quad L'_1 = L_1 \\ -2x \quad -y \quad -3z \quad = -5 \quad L'_2 = L_2 - \frac{-2}{2} L_1 = L_2 + L_1 \\ 6x \quad +4y \quad +4z \quad = 16 \quad L'_3 = L_3 - \frac{6}{2} L_1 = L_3 - 3L_1 \end{array} \right.$$

Principe du pivot de Gauss

Exemple

2^{ème} transvection :

$$\left\{ \begin{array}{lclcl} 2x & +2y & -3z & = 2 & L'_1 \\ & 1y & -6z & = -3 & L'_2 \\ & -2y & +13z & = 10 & L'_3 \end{array} \right.$$

Principe du pivot de Gauss

Exemple

2^{ème} transvection :

$$\left\{ \begin{array}{l} 2x + 2y - 3z = 2 \quad L_1'' = L_1' \\ \quad \quad \color{red}{1}y - 6z = -3 \quad L_2'' = L_2' \\ \quad \quad -2y + 13z = 10 \quad L_3'' = L_3' - \frac{-2}{\color{red}{1}}L_2' = L_3' - 2L_2' \end{array} \right.$$

Principe du pivot de Gauss

Exemple

Système triangulaire ■' équivalent à ■ :

$$\left\{ \begin{array}{rcll} 2x & +2y & -3z & = 2 & L_1'' \\ & y & -6z & = -3 & L_2'' \\ & & z & = 4 & L_3'' \end{array} \right.$$

Principe du pivot de Gauss

Exemple

Phase de remontée :

$$\left\{ \begin{array}{l} 2x + 2y - 3z = 2 \\ y = -3 + 6z = -3 + 24 = 21 \\ z = 4 \end{array} \right. \begin{array}{l} L_1'' \\ L_2'' \\ L_3'' \end{array}$$

Principe du pivot de Gauss

Exemple

Phase de remontée :

$$\left\{ \begin{array}{l} x = \frac{1}{2}(2 - 2y + 3z) = \frac{1}{2}(2 - 42 + 12) = -14 \quad L_1'' \\ y = 21 \quad L_2'' \\ z = 4 \quad L_3'' \end{array} \right.$$

Solution : $(-14, 21, 4)$.

Système de Cramer

Dans la suite, on se restreint à un système de Cramer ce qui garantit :

- ▶ **n équations pour n inconnues.**
- ▶ **Qu'il existe une solution.**
- ▶ **Que la solution est unique.**
- ▶ **Qu'on peut toujours trouver un pivot pour chaque variable**

On pourra être amené à échanger des lignes pour trouver le pivot.

Point de vue matricielle

D'un point de vue matricielle, l'algorithme de Gauss revient à :

- ▶ **Remplacer $AX = Y$ par $TX = Y'$**
- ▶ **Avec T matrice triangulaire**

$$T = \begin{pmatrix} 2 & 2 & -3 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{pmatrix}$$

- ▶ **Et Y' :**

$$Y' = \begin{pmatrix} 2 \\ -3 \\ 4 \end{pmatrix}$$

Sommaire

- 1 Rappel : système linéaire
- 2 Résolution d'un système linéaire : pivot de Gauss
- 3 Implémentation « maison » du pivot de Gauss**
- 4 Fonction prédéfinie de la bibliothèque `numpy`

L'algorithme en bref

Principe de l'algorithme du pivot de Gauss.

1. Triangulariser le système $AX = Y$
 - Choix du pivot a_{ij} (permutation éventuelles de lignes)
 - Transvection sur les lignes : $L_j \leftarrow L_j - \frac{a_{ji}}{a_{ii}} L_i$
2. Résoudre le système triangulaire équivalent à $TX = Y_p$ par substitutions successives.

L'algorithme en bref

Les fonctions nécessaires :

- 1 `triang(A, Y)` : triangularise le système $AX = Y$
 - `rech_pivot(A, j)` : recherche le pivot dans la colonne j
 - `permuter_ligne(A, i, j)` : permute les ligne i et j d'un vecteur ou d'une matrice
 - `transvect(i, j, c)` : réalise la combinaison linéaire : $L_j \leftarrow L_j - cL_i$
- 2 `remonte(T, Yp)` : résout le système triangulaire supérieure $TX = Y_p$
- 3 `resol(A, Y)` : résout le système de Cramer $AX = Y$

L'algorithme en détails

Deux implémentations :

- ▶ *Pivot naturel* : a_{ij} systématiquement
- ▶ *Pivot partiel* : a_{ji} le plus grand possible avec $j \geq i$.

Pivot naturel

Voir `pivot_gauss_naturel.py`.

- ▶ Risque de tomber sur un pivot nul \Rightarrow division par zéro !
- ▶ Ou très petit pour des matrices mal conditionnées \Rightarrow fait exploser les erreurs numériques (absorption ...).

Pivot partiel

Voir `pivot_gauss_partiel.py`.

On minimise les deux risques précédents en prenant systématiquement comme pivot le plus grand des coefficients matriciels de la colonne $j > i$. Ne garantit pas le résultat pour autant.

Naïvement, on pourrait se contenter de tester la nullité du pivot mais les comparaisons à zéro n'ont pas de sens avec les flottants ! L'arithmétique y est fautive. Ainsi $(1-1/3) - 2/3 == 0$ est faux !

Les erreurs numériques

- ▶ Erreurs d'arrondis initiales sur les données A et Y :
 - Problème mathématique : X tel que $AX = Y$?
 - Problème résolu : X tel que $A_1 X = Y_1$ avec $A_1 = A + \delta A$ et $Y_1 = Y + \delta Y$.
 - On peut supposer que $\frac{\delta Y}{Y}$ et $\frac{\delta A}{A}$ sont de l'ordre de l' ε – *machine* i.e. $2^{-52} \approx 2.10^{-16}$.
- ▶ Ces erreurs se propagent au cours des calculs.
- ▶ On appelle *conditionnement* d'un problème sa sensibilité aux petites variations sur les entrées.
- ▶ Pour des matrices bien conditionnées, l'erreur sur le résultat reste petite.
- ▶ Pour des matrices mal conditionnées, les erreurs deviennent critiques.
- ▶ Par exemple, l'algorithme peut renvoyer une solution à un problème qui n'en admet pas et réciproquement.

Complexité en temps

- ▶ Triangularisation : $O(n^3)$
- ▶ Remontée : $O(n^2)$

Finalement la complexité de l'algorithme est $O(n^3)$.

Acceptable pour $n < 10^5$.

Complexité en mémoire

Stockage de la matrice : $O(n^2)$

Ex : pour $n = 10^6$, la mémoire vive occupée est de l'ordre de 1 To !Rédhibitoire !

Sommaire

- 1 Rappel : système linéaire
- 2 Résolution d'un système linéaire : pivot de Gauss
- 3 Implémentation « maison » du pivot de Gauss
- 4 Fonction prédéfinie de la bibliothèque `numpy`

La fonction `linalg.solve` résout les systèmes linéaires $AX = Y$.

Sous-couche en C et Fortran très performante, très rapide.

Préférer cette fonction à une implémentation maison !