

Informatique

Systemes linéaires

Pivot de Gauss

Cours

Pivot de Gauss	3
1.I. Contexte.....	3
1.II. Mise sous forme matricielle	3
1.III. Résolution classique sous Python.....	3
1.IV. Méthode de Gauss avec recherche partielle des pivots	4
1.IV.1 Exemple.....	4
1.IV.1.a Raisonnement sur le système.....	4
1.IV.1.b Raisonnement matriciel.....	5
1.IV.2 Méthode générale	6
1.IV.2.a Préliminaires.....	6
1.IV.2.b Notations	6
1.IV.2.c Algorithme de transformation.....	7
1.IV.2.d Algorithme de résolution	8
1.IV.2.d.i Résolution directe.....	8
1.IV.2.d.ii Diagonalisation de la matrice	8
1.IV.3 Complexité	9
1.IV.3.a Mise sous forme échelonnée	9
1.IV.3.b Résolution triangulaire	10
1.IV.3.c Bilan	10

Pivot de Gauss

1.I. Contexte

Supposons un problème dont les équations se mettent sous la forme d'un système linéaire à n équations et n inconnues inversible, c'est-à-dire admettant une solution unique, ou encore dit de Cramer.

1.II. Mise sous forme matricielle

Soit le système linéaire de n équations à m variables d'entrée $e_i(t)$ et n variables de sortie $s_i(t)$:

$$\begin{cases} a_{11}s_1(t) + a_{12}s_2(t) + \dots + a_{1n}s_n(t) = b_{11}e_1(t) + b_{12}e_2(t) + \dots + b_{1m}e_m(t) \\ a_{21}s_1(t) + a_{22}s_2(t) + \dots + a_{2n}s_n(t) = b_{21}e_1(t) + b_{22}e_2(t) + \dots + b_{2m}e_m(t) \\ \vdots \\ a_{n1}s_1(t) + a_{n2}s_2(t) + \dots + a_{nn}s_n(t) = b_{n1}e_1(t) + b_{n2}e_2(t) + \dots + b_{nm}e_m(t) \end{cases}$$

Avec a_{ij} et b_{ij} des coefficients constants.

On peut alors traduire ce système d'équations sous forme matricielle :

$$K_E E(t) = K_S S(t)$$

Avec :

$$E(t) = \begin{bmatrix} e_1(t) \\ e_2(t) \\ \vdots \\ e_n(t) \end{bmatrix} ; \quad K_E = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ & \vdots & & \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix}$$
$$S(t) = \begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_n(t) \end{bmatrix} ; \quad K_S = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ & \vdots & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Connaissant les entrées, l'objectif est alors de déterminer les sorties.

On peut donc se ramener à la résolution du système suivant : $K_S S(t) = B$

Avec $B = K_E E(t)$ un vecteur dans lequel tout est connu.

1.III. Résolution classique sous Python

Il est assez simple de résoudre directement ce système avec le module solve de numpy. Après avoir créé K et B du système $K \begin{bmatrix} x \\ y \\ z \end{bmatrix} = B$, il suffit d'écrire :

```
import numpy as np
x,y,z = np.linalg.solve(K,B)
```

1.IV. Méthode de Gauss avec recherche partielle des pivots

1.IV.1 Exemple

Soit le système suivant :

$$\begin{cases} x + y + 2z = 10 \\ x + 3y + 4z = 20 \\ x + 5y + z = 30 \end{cases}$$

Notre objectif est d'obtenir un système échelonné, c'est-à-dire dans lequel une première ligne contient une seule inconnue, une seconde ligne contient cette précédente inconnue et une autre, et ainsi de suite. Il sera alors aisé de déterminer chacune d'entre elles les unes après les autres.

1.IV.1.a Raisonnement sur le système

On appelle pivot une variable (x, y ou z dans notre exemple). La méthode du pivot de Gauss va alors consister à choisir successivement un groupe équation/pivot et à réaliser des opérations sur les lignes suivantes. Le choix est généralement arbitraire, ligne par ligne et variable après variable... Vous comprendrez comment traiter les cas particuliers où un pivot n'est pas présent dans une équation plus tard dans le cours (permutations de lignes ou colonnes).

Ainsi, pour notre exemple, nous choisissons x comme premier pivot dans la première équation :

$$\begin{cases} \color{red}{x} + y + 2z = 10 \\ x + 3y + 4z = 20 \\ x + 5y + z = 30 \end{cases}$$

On soustrait à chacune des lignes suivantes la première ligne afin d'y faire disparaître le pivot en multipliant si besoin par coefficient :

$$\begin{cases} \color{red}{x} + y + 2z = 10 \\ x - \color{red}{x} + 3y - \color{red}{y} + 4z - \color{red}{2z} = 20 - \color{red}{10} \\ x - \color{red}{x} + 5y - \color{red}{y} + z - \color{red}{2z} = 30 - \color{red}{10} \end{cases} \Leftrightarrow \begin{cases} x + y + 2z = 10 \\ 2y + 2z = 10 \\ 4y - z = 20 \end{cases}$$

On procède ainsi pour chaque nouveau système à partir de la ligne suivante. Ainsi, le nouveau pivot du second système est y :

$$\begin{cases} x + y + 2z = 10 \\ 2\color{red}{y} + 2z = 10 \\ 4y - \color{red}{2} * \color{red}{2y} - z - \color{red}{2} * \color{red}{2z} = 20 - \color{red}{2} * \color{red}{10} \end{cases} \Leftrightarrow \begin{cases} x + y + 2z = 10 \\ 2y + 2z = 10 \\ -5z = 0 \end{cases}$$

On peut alors résoudre le système car une équation donne une première inconnue puis chacune des autres permet de trouver une inconnue supplémentaire :

$$\begin{cases} x + y + 2z = 10 \\ 2y + 2z = 10 \\ -5z = 0 \end{cases} \Leftrightarrow \begin{cases} x + y = 10 \\ 2y = 10 \\ z = 0 \end{cases} \Leftrightarrow \begin{cases} x + 5 = 10 \\ y = 5 \\ z = 0 \end{cases} \Leftrightarrow \begin{cases} x = 5 \\ y = 5 \\ z = 0 \end{cases}$$

1.IV.1.b Raisonnement matriciel

$$\begin{cases} x + y + 2z = 10 \\ x + 3y + 4z = 20 \\ x + 5y + z = 30 \end{cases} \Leftrightarrow \begin{bmatrix} 1 & 1 & 2 \\ 1 & 3 & 4 \\ 1 & 5 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$

Les opérations successives sur le système se traduisent ainsi :

$$\begin{bmatrix} \mathbf{1} & 1 & 2 \\ 1 - \mathbf{1} & 3 - \mathbf{1} & 4 - \mathbf{2} \\ 1 - \mathbf{1} & 5 - \mathbf{1} & 1 - \mathbf{2} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 20 - \mathbf{10} \\ 30 - \mathbf{10} \end{bmatrix} \Leftrightarrow \begin{bmatrix} \mathbf{1} & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 4 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 20 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & \mathbf{2} & 2 \\ 0 & 4 - \mathbf{2 * 2} & -1 - \mathbf{2 * 2} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 20 - \mathbf{2 * 10} \end{bmatrix} \Leftrightarrow \begin{bmatrix} 1 & 1 & 2 \\ 0 & \mathbf{2} & 2 \\ 0 & 0 & -5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 0 & -5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 0 \end{bmatrix}$$

On voit qu'il est donc possible d'obtenir une matrice triangulaire, dite échelonnée, traduisant le même système linéaire que le système initial :

$$\begin{bmatrix} 1 & 1 & 2 \\ \mathbf{0} & 2 & 2 \\ \mathbf{0} & \mathbf{0} & -5 \end{bmatrix}$$

1.IV.2 Méthode générale

1.IV.2.a Préliminaires

Soit le système $K_S S(t) = B$ inversible. Appelons L_i la ligne i du système $K_S S(t) = B$ et C_i la colonne i de la matrice K_S . Notre objectif est de procéder à des opérations permettant d'arriver à un système échelonné, c'est-à-dire à une forme de matrice K_S dont un triangle (inférieur ou supérieur) est plein (diagonale incluse) et où le reste est vide.

Théorème de Gauss-Jordan : Tout système linéaire se ramène à un système échelonné équivalent en utilisant 3 types d'opérations élémentaires :

- Permutation de 2 équations : $L_i \leftrightarrow L_j$
- Permutation de l'ordre des inconnues : $C_i \leftrightarrow C_j$
- Remplacement d'une équation (transvection) : $L_i \leftarrow L_i + \lambda L_j$

Attention : ces opérations doivent être réalisées les unes après les autres, et non en même temps. En effet, il est alors possible de transformer le système et de ne plus avoir les mêmes solutions. Imaginons par exemple remplacer en même temps chaque ligne d'un système à n équations par la somme de toutes les lignes... On obtient alors n fois la même équation... Le système n'est plus inversible.

1.IV.2.b Notations

Soit la matrice $K_S = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$, le vecteur $B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ et le vecteur solution $S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix}$

1.IV.2.c Algorithme de transformation

Attention aux indices pris ci-dessous comme allant de 1 à n (0 à n-1 dans Python).

A la ligne $i \in [1, n - 1]$, le travail réalisé sur la matrice complète revient à travailler sur la sous-matrice

suivante : $\begin{bmatrix} a_{ii} & a_{i,i+1} & \dots & a_{in} \\ & \vdots & & \\ a_{ni} & a_{n,i+1} & \dots & a_{nn} \end{bmatrix}$. Alors :

- Si $a_{ii} \neq 0$: On remplace les lignes $L_j, j \in [i + 1, n]$ telles que : $L_j \leftarrow L_j - \frac{a_{ji}}{a_{ii}} L_i$ – C'est-à-dire qu'il faut modifier à la fois K_S et B . Cette étape s'appelle la « **transvection** »
- Si $a_{ii} = 0$:
 - o On cherche un coefficient a_{ji} non nul pour $j \in [i + 1, n]$, c'est-à-dire en dessous de a_{ii} (il existe forcément – cf. remarques). On appelle k la ligne retenue
 - o On permute les lignes i et k
 - o On procède comme proposé lorsque $a_{ii} \neq 0$

Remarques :

- Il est possible de procéder par un échange de colonnes, mais cela interverti l'ordre des inconnues dans le vecteur inconnu, ce qui rend la résolution automatique plus complexe
- Lors de la première étape, la matrice K_S étant inversible, la première colonne ne peut contenir que des 0. On trouvera donc un terme non nul.
- A chaque étape i , le système est de la forme :

$$\begin{bmatrix} a_{11} & a_{12} & \dots & \dots & \dots & \dots & a_{1n} \\ 0 & a_{22} & \dots & \dots & \dots & \dots & a_{2n} \\ 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & a_{ii} & a_{i,i+1} & \dots & a_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & a_{ni} & a_{n,i+1} & \dots & a_{nn} \end{bmatrix}$$

Chaque sous matrice $\begin{bmatrix} a_{ii} & a_{i,i+1} & \dots & a_{in} \\ & \vdots & & \\ a_{ni} & a_{n,i+1} & \dots & a_{nn} \end{bmatrix}$ est inversible puisqu'en face d'un bloc de 0 à sa

gauche, il y aura donc forcément un terme non nul dans sa première colonne.

- Numériquement, $a_{ii} \neq 0$ peut conduire à l'utilisation d'un pivot très proche de 0, ce qui peut conduire à de mauvais résultats. On préférera donc intervertir les lignes afin d'obtenir comme **pivot le terme de plus grande valeur absolue** parmi tous les choix possibles, même si $a_{ii} \neq 0$. On parle alors de **pivot partiel**

La matrice ainsi obtenue s'écrit sous la forme :

$$K'_S = \begin{bmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & a'_{nn} \end{bmatrix}$$

Tel que : $K'_S S(t) = B'$

1.IV.2.d Algorithme de résolution

1.IV.2.d.i Résolution directe

Ce qui suit n'est valable qu'en cas de permutations de lignes (et non de colonnes) dans la méthode précédente, sinon l'ordre des inconnues peut avoir changé.

- Ligne n :

$$\begin{aligned}a'_{nn}s_n &= b'_n \\ \Rightarrow s_n &= \frac{b'_n}{a'_{nn}}\end{aligned}$$

- Ligne $n - 1$:

$$\begin{aligned}a'_{n-1,n-1}s_{n-1} + a'_{n-1,n}s_n &= b'_{n-1} \\ \Rightarrow s_{n-1} &= \frac{b'_{n-1} - a'_{n-1,n}s_n}{a'_{n-1,n-1}} = \frac{b'_{n-1} - \sum_{k=n}^n a'_{n-1,k}s_k}{a'_{n-1,n-1}}\end{aligned}$$

- Ligne $n - 2$:

$$\begin{aligned}a'_{n-2,n-2}s_{n-2} + a'_{n-2,n-1}s_{n-1} + a'_{n-2,n}s_n &= b'_{n-2} \\ \Rightarrow s_{n-2} &= \frac{b'_{n-2} - a'_{n-2,n}s_n - a'_{n-2,n-1}s_{n-1}}{a'_{n-2,n-2}} = \frac{b'_{n-2} - \sum_{k=n-1}^n a'_{n-2,k}s_k}{a'_{n-2,n-2}}\end{aligned}$$

Donc, d'une manière générale, à la ligne i en résolvant de n à 1 :

$$s_i = \frac{b'_i - \sum_{k=i+1}^n a'_{i,k}s_k}{a'_{ii}}$$

1.IV.2.d.ii Diagonalisation de la matrice

Une seconde méthode consiste à appliquer les algorithmes élaborés lors de la transformation de la moitié inférieure gauche afin d'annuler les termes de la moitié supérieure droite, ce qui conduit à obtenir une matrice diagonale :

$$K'_S = \begin{bmatrix} a''_{11} & 0 & \dots & 0 \\ 0 & a''_{22} & \dots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & a''_{nn} \end{bmatrix}$$

Tel que :

$$K''_S S(t) = B''$$

On a alors directement les solutions :

$$s_i = \frac{b''_i}{a''_{ii}}$$

1.IV.3 Complexité

1.IV.3.a Mise sous forme échelonnée

En supposant dans un premier temps qu'il n'est pas nécessaire d'effectuer de permutations de lignes, la transformation du système général en système échelonné s'effectue en N étapes (modifications de cases), chacune associée à un nombre d'opérations mathématiques (+, -, *, /) fixe. Comme $O(n) = O(3n)$ par exemple, compter les « étapes » donnera la même complexité que de compter les opérations. Pour simplifier la suite, je parlerai quand même d'opérations à réaliser, mais ce ne sont donc pas les « opérations mathématiques ».

Pour chaque ligne i du système de 1 à $n - 1$, on effectue N_i opérations avec :

- $(n - i + 1)$ opérations à chaque ligne sur la matrice K & 1 opération sur le vecteur B , soit $(n - i + 2)$ opérations par ligne
- Ceci appliqué aux $(n - i)$ lignes du système (transvection des lignes en dessous du pivot)

$$N_i = \sum_{j=1}^{n-i} (n - i + 2) = (n - i)(n - i + 2)$$

Soit au total, pour tous les pivots :

$$\begin{aligned} N &= \sum_{i=1}^{n-1} N_i = \sum_{i=1}^{n-1} (n - i)(n - i + 2) = \sum_{i=1}^{n-1} (n^2 - ni + 2n - ni + i^2 - 2i) \\ &= \sum_{i=1}^{n-1} (n(n + 2) - 2(n + 1)i + i^2) = (n - 1)n(n + 2) - 2(n + 1) \sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1} i^2 \\ \sum_{i=1}^{n-1} i &= \frac{n(n - 1)}{2} \quad ; \quad \sum_{i=1}^{n-1} i^2 = \frac{(n - 1)n(2n - 1)}{6} \end{aligned}$$

Soit :

$$\begin{aligned} N &= (n - 1)n(n + 2) - 2(n + 1) \frac{n(n - 1)}{2} + \frac{(n - 1)n(2n - 1)}{6} \\ N &= (n - 1)n \left[(n + 2) - (n + 1) + \frac{(2n - 1)}{6} \right] = (n - 1)n \left[1 + \frac{(2n - 1)}{6} \right] \\ N &= \frac{(n - 1)n(2n + 5)}{6} \end{aligned}$$

La complexité de la transformation en système échelonné dans le meilleur des cas est en :

$$O(n^3)$$

A ce nombre, dans le pire des cas, il faut ajouter pour chaque ligne i un nombre d'opérations N'_i contenant :

- Un test sur les $(n - i)$ lignes en dessous : $\sum_{j=1}^{n-i} 1 = n - i$
- Un échange de 2 lignes (uniquement les termes non nuls) de $K(n - i + 1)$ et le terme de $B(1)$: $(n - i + 1) + 1 = n - i + 2$

On obtient :

$$N'_i = n - i + 2 + n - i = 2n - 2i + 2 = 2(n + 1) - 2i$$

Et donc un coût supplémentaire pour le système total de :

$$N' = \sum_{i=1}^{n-1} N'_i = \sum_{i=1}^{n-1} 2(n + 1) - 2i = \sum_{i=1}^{n-1} 2(n + 1) - 2 \sum_{i=1}^{n-1} i$$

$$N' = 2(n - 1)(n + 1) - 2 \frac{n(n - 1)}{2}$$

$$N' = 2(n - 1)(n + 1) - n(n - 1) = n(n - 1)(2n + 2 - n) = n(n - 1)(n + 2)$$

La nombre d'opérations dans le pire des cas étant la somme $N + N'$, la complexité globale reste inchangée, en $O(n^3)$.

1.IV.3.b Résolution triangulaire

Rappelons la formule :

$$s_i = \frac{b'_i - \sum_{k=i+1}^n a'_{i,k} s_k}{a'_{ii}}$$

Soit N''_i le nombre d'opérations pour le calcul de chaque inconnue (on ne peut se limiter au nombre de calculs, car le nombre d'opérations dépend de i). Il y a une différence, une division et une somme de $n - i$ termes, soit :

$$N''_i = 2 + n - i = 2 + n - i$$

$$N'' = \sum_{i=1}^n N''_i = \sum_{i=1}^n (2 + n - i) = n(2 + n) - \sum_{i=1}^n i = n(2 + n) - \frac{n(n - 1)}{2}$$

$$= \frac{2n(2 + n) - n(n - 1)}{2} = \frac{4n + 2n^2 - n^2 + n}{2} = \frac{5n + n^2}{2}$$

Soit finalement une complexité en $O(n^2)$, ce qui ne change pas la complexité de l'algorithme total.

1.IV.3.c Bilan

La méthode de résolution de systèmes linéaires par pivot de Gauss est de complexité $O(n^3)$ quel que soit le système inversible initial.