

PIVOT DE GAUSS

1. Problématique

L'algorithme de Gauss sert à résoudre un système linéaire de n équations à p inconnues.

Un tel système s'écrit $AX = B$, où $A \in \mathcal{M}_{n,p}(\mathbb{R})$ est la matrice des coefficients, $X \in \mathcal{M}_{p,1}(\mathbb{R})$ est la matrice colonne des inconnues et $B \in \mathcal{M}_{n,1}(\mathbb{R})$ est la matrice colonne des résultats.

Cet algorithme exécute des opérations élémentaires sur les lignes des matrices A et B .

Dans ce cours nous nous plaçons dans le cas particulier suivant :

la matrice A est inversible, c'est à dire $A \in GL_n(\mathbb{R})$.

La système comporte donc n équations et n inconnues et admet une unique solution. On parle de système de *Cramer*.

2. Représentation matricielle et opérations élémentaires

Les notions de cette section ont été vues lors du TP `python` sur les matrices.

2.1 Représentation d'une matrice carrée

Une matrice se représente comme une liste de listes.

En mathématiques	En python
$A = \begin{pmatrix} 2 & 2 & -3 \\ -2 & -1 & -3 \\ 6 & 4 & 4 \end{pmatrix}$	
$A_{1,1} = 2, A_{3,3} = 4, A_{3,1} = 6.$	

Remarque. Attention, dans le système $AX = B$, la matrice colonne des résultats B sera représentée également comme une liste de liste, avec un indice double. En effet, elle doit subir les mêmes opérations élémentaires que la matrice A et les fonctions des opérations élémentaires utilisent les indices doubles.

Par exemple, on représentera $B = [[0], [0], [0]]$ pour un système homogène de 3 équations à 3 inconnues.

2.2 Opérations élémentaires

Les trois opérations élémentaires sur les lignes L_i d'une matrice sont

- les dilatations $L_i \leftarrow \mu L_i$, où μ est un réel non nul,
- les transvections $L_j \leftarrow L_j + \mu L_i$, où μ est un réel non nul,
- les permutations ou transpositions $L_i \leftrightarrow L_j$.

Seules les transvections et les permutations sont utilisées dans la programmation du pivot de Gauss.

On donne leurs algorithmes en `python` ci-dessous. Les indices utilisés sont ceux de `python`. Ils commencent à zéro.

A. Transvection

La fonction `transvection` prend en paramètres une matrice A , un indice de ligne j , un indice de ligne i et un réel non nul μ .

Elle modifie sur place la matrice A en effectuant $L_j \leftarrow L_j + \mu L_i$.

```
def transvection(A, j, i, mu):
    """ Lj ← Lj + mu Li """
    n=len(A[0]) #le nombre de colonnes parcourues par
    k
    for k in range(n):
        A[j][k] = A[j][k] + mu * A[i][k]
```

B. Permutation ou transposition

La fonction `permutation` prend en paramètres une matrice A , un indice de ligne i et un indice de ligne j .

Elle modifie sur place la matrice A en effectuant $L_i \leftrightarrow L_j$.

```
def permutation(A, i, j):
    """ Li ↔ Lj """
    A[i] , A[j] = A[j] , A[i]
```

3. Le pivot de Gauss dans un cas particulier

Avant de donner l'algorithme de Gauss général, on présente une version plus simple qui traite un cas particulier.

Hypothèse ($\star$). On suppose qu'à l'étape i , le pivot est l'élément $a_{ii} \neq 0$ de la ligne L_i . Il n'y a donc pas d'échange de lignes à faire, c'est à dire pas de permutation et on est assuré de pouvoir diviser par a_{ii} non nul.

Exemples.

1. Un système lié à la matrice A précédente vérifie ($\star$).
2. Un système lié à la matrice $\begin{pmatrix} 2 & 2 & -3 \\ -2 & -2 & -3 \\ 6 & 4 & 4 \end{pmatrix}$ ne vérifie pas ($\star$).

3.1 Mise sous forme triangulaire supérieure

L'algorithme suivant transforme une matrice A vérifiant l'hypothèse ($\star$) en une matrice triangulaire supérieure.

```
n = len(A)
for i in range(n-1):
    for j in range(i+1, n):
        mu = - A[j][i] / A[i][i]
        transvection(A, j, i, mu)
```

Exercice 1

1. Expliquer (et comprendre) ce que font les indices i et j . Commenter cet algorithme.
2. Estimer la complexité de cet algorithme en fonction de n .

4. Le pivot de Gauss dans le cas général

On se place maintenant dans le cas général précisé au début du chapitre. Ainsi, à l'étape i , l'élément a_{ii} de la ligne L_i peut poser problème si mathématiquement il est nul :

1. ceci n'est pas détectable car les comparaisons à zéro de flottants n'ont jamais de sens (voir le cours sur la représentation des nombres en machine),
2. il est alors représenté par un flottant très proche de zéro. Les transvections avec $\frac{a_{ji}}{a_{ii}}$ donneront de grands nombres flottants qui mathématiquement ne devaient pas intervenir dans les calculs.

Sachant que le système est de Cramer, on sait qu'un pivot existera et on décide d'utiliser comme pivot le **coefficient le plus élevé en valeur absolue** a_{ki} avec $k \geq i$ (c'est à dire a_{ki} est pris dans une ligne L_k sous L_i).

4.1 Recherche de pivots

On propose la fonction suivante qui prend en paramètre la matrice A et l'indice de ligne i et qui retourne l'indice $k \geq i$ tel que $|a_{ki}|$ soit maximale :

```
def recherche_pivot(A,i):
    """ retourne l'indice du plus grand élément
    en valeur absolue sous A[i][i] """
    n=len(A)
    indice_piv=i # la ligne du maximum provisoire
    for k in range(i+1,n):
        if abs(A[k][i])>abs(A[indice_piv][i]):
            indice_piv=k #un nouveau maximum ✓
    provisoire
    return(indice_piv)
```

Une fois cet indice $k \geq i$ déterminé,

1. s'il vaut i alors on utilise a_{ii} comme pivot
2. sinon on effectue la permutation des lignes L_i et L_k . Le **nouveau coefficient** a_{ii} obtenu après cet échange sert de pivot.

4.2 L'algorithme complet du pivot de Gauss général

Exercice 4 Recopier et compléter l'algorithme de Gauss général en python.

```
def Gauss(A0,B0):
    """ Résolution de A0.X=B0
    sous l'hypothèse A0 inversible """
    (A , B) = (copie_matrice(A0) , copie_matrice(B0))
    n=len(A)
    for i in range(n-1):
        # recherche l'indice k du pivot sous L[i]
        if k>i :
            # permuter les lignes L[i] et L[k]
            # mise sous forme triangulaire
```

```
# phase de remontée  
return(X)
```

5. Complexité

On s'intéresse maintenant à la complexité de l'algorithme du pivot de Gauss en fonction de n (la taille de la matrice carrée associée au système).

- **Recherche de pivot et mise sous forme triangulaire.**

Pour chaque indice de ligne i ,

- la recherche du pivot a une complexité linéaire en n (c'est à dire en $O(n)$),
- l'échange de lignes a aussi une complexité linéaire en n ,
- chaque transvection également et il y en a au plus n , d'où une complexité quadratique (c'est à dire en $O(n^2)$).

Or l'indice de ligne i parcourt $\{0, \dots, n-2\}$ d'où finalement une complexité en $O(n^3)$ pour la mise sous forme triangulaire.

- **Phase de remontée.** Cette phase contient deux boucles **for** imbriquées. Sa complexité est en $O(n^2)$.

- **Bilan.** Finalement, la complexité de l'algorithme du pivot de Gauss est en $O(n^3)$. Ce coût est principalement payé lors de la mise sous forme triangulaire.

6. Problèmes liés à cette résolution

6.1 Erreurs d'arrondis

La manipulation de flottants s'accompagne d'erreurs de calculs (voir chapitre sur la représentation des nombres en machine).

La recherche de pivot maximal en valeur absolue (appelée aussi méthode *du pivot partiel*) permet d'éviter l'utilisation de flottants trop proche de 0 et de leurs inverses trop grands. Mais cela n'interdit pas d'éventuelles mauvaises surprises (voir les matrices de Hilbert en TP).

6.2 Temps de calcul

Dans certains problèmes pratiques, n peut être de l'ordre de quelques milliards. Une résolution en $O(n^3)$ est absolument impossible en un temps raisonnable.

En effet il y aura de l'ordre de $(10^9)^3 = 10^{27}$ opérations sachant qu'un ordinateur personnel en effectue environ 10^9 par seconde ... faites le calcul !