

# ◆ TP5 – Méthode du pivot de Gauss

## I. — Méthode du pivot de Gauss (Rappel)

### 1) Systèmes échelonnés et pivots

#### Définition 1

On dit qu'un système linéaire  $(S)$  de  $n$  équations est échelonné si, pour tout  $i \in \llbracket 2, n \rrbracket$ , la  $i$ -ème ligne de  $(S)$  commence par au moins  $i - 1$  coefficients nuls (en respectant l'ordre initial des inconnues).

*Remarque 2.* Ceci revient à dire qu'à chaque nouvelle ligne, il y a au moins une inconnue supplémentaire qui disparaît (dans l'ordre initial des inconnues).

**Exemple 3.** Les systèmes suivants sont échelonnés :

$$(S_1) \begin{cases} 3x + y = 2 \\ y = 1 \end{cases} \quad (S_2) \begin{cases} -x + y - z = 3 \\ 3y + 2z = 5 \\ 5z = 2 \end{cases} \quad (S_3) \begin{cases} 4x - z = 0 \\ 2y + z = 0 \end{cases} \quad (S_4) \begin{cases} x + y + z = 2 \\ y - z = 1 \\ 2z = 5 \\ 0 = 0 \end{cases}$$

En revanche, le système  $(S_5) \begin{cases} x - 2y + z = 2 \\ 5x + z = 1 \\ y = -1 \end{cases}$  n'est pas échelonné.

#### Définition 4

Si  $(S)$  est un système échelonné alors les pivots de  $(S)$  sont les coefficients non nuls apparaissant en tête de chaque ligne.

**Exemple 5.**  $(S_1)$  possède 2 pivots (3 et 1),  $(S_2)$  possède 3 pivots (-1, 3 et 5),  $(S_3)$  possède 2 pivots (4 et 2) et  $(S_4)$  possède 3 pivots (1, 1 et 2).

### 2) Méthode du pivot de Gauss

**Notation 6.** Si  $(S)$  est un système linéaire de  $n$  équations, on note, pour tout  $i \in \llbracket 1, n \rrbracket$ ,  $L_i$  la  $i$ -ème ligne de  $(S)$ .

#### Propriété 7. — Opérations élémentaires

Soit  $(S)$  un système linéaire de  $n$  équations et  $i$  et  $j$  deux éléments de  $\llbracket 1, n \rrbracket$ . On obtient un système équivalent à  $(S)$  si :

- on échange les lignes  $L_i$  et  $L_j$ , ce qu'on notera  $L_i \leftrightarrow L_j$  ;
- on remplace la ligne  $L_i$  par  $tL_i$  où  $t \in \mathbb{R}^*$ , ce qu'on notera  $L_i \leftarrow tL_i$  ;
- on remplace la ligne  $L_i$  par  $L_i + tL_j$  où  $t \in \mathbb{R}$ , ce qu'on notera  $L_i \leftarrow L_i + tL_j$ .

*Remarque 8.* Les opérations décrites dans la propriété précédente s'appellent les opérations élémentaires sur les lignes d'un système.

## Méthode 9 : Algorithme du pivot de Gauss

Tout système peut être transformé en un système échelonné équivalent par opérations élémentaires sur les lignes de la manière suivante :

- on note  $L_p$  une ligne du système dans laquelle le coefficient  $\alpha$  de  $x_1$  est non nul et on échange  $L_1$  et  $L_p$  ;
- on utilise le coefficient  $\alpha$  (qui est le pivot de la première ligne) pour éliminer  $x_1$  dans toutes les autres équations du système grâce aux opérations  $L_i \leftarrow L_i - \frac{\beta_i}{\alpha} L_1$  où  $\beta_i$  est le coefficient de  $x_1$  dans  $L_i$  (si  $\beta_i = 0$ , il n'y a donc rien à faire) pour tout  $i \in \llbracket 2, n \rrbracket$  ;
- On réitère le procédé sur le système formé par les  $n - 1$  dernières équations avec  $x_2$  (ou à défaut  $x_3, x_4 \dots$ ) et ainsi de suite.

*Remarque 10.* On a en fait utilisé seulement deux des trois opérations élémentaires. La troisième peut être utilisée si l'on souhaite de plus que les pivots soient tous égaux à 1. Pour cela, il suffit de réaliser l'opération  $L_i \leftarrow \frac{1}{\alpha_i} L_i$  pour tout  $i \in \llbracket 1, n \rrbracket$  où  $\alpha_i$  est le pivot de la  $i$ -ème ligne.

L'intérêt de transformer un système  $(S)$  en un système échelonné équivalent  $(S')$  réside dans le fait que la résolution de  $(S')$  est particulièrement simple par substitutions successives en remontant le système à partir de la dernière équation. Par exemple, si on considère le système  $(S_2)$  de l'exemple 5 alors :

- de la troisième équation, on déduit que  $z = \frac{2}{5}$  ;
- de la deuxième équation, on déduit que  $3y = 5 - 2 \times \frac{2}{5} = \frac{21}{5}$  donc  $y = \frac{7}{5}$  ;
- de la première équation, on déduit que  $-x = 3 - \frac{7}{5} + \frac{2}{5} = 2$  donc  $x = -2$ .

Ainsi, l'unique solution de  $(S_2)$  est  $(-2; \frac{7}{5}; \frac{2}{5})$ .

**Exemple 11.** Dans chacun des cas suivants, échelonner le système par la méthode du pivot de Gauss puis résoudre le système.

$$(S_6) \begin{cases} x + y + 2z = -1 \\ 2x - y + 2z = -4 \\ 4x + y + 4z = -2 \end{cases} \quad (S_7) \begin{cases} 2x - 3y + 4z = -3 \\ -x + 2y + z = 5 \\ 4x - 5y + 14z = 1 \end{cases} \quad (S_8) \begin{cases} x - y - z = 4 \\ 2x - 3y + 2z = 1 \\ 5x - 8y + 7z = 1 \end{cases}$$

## Définition 12

Soit  $(S)$  un système linéaire de  $n$  équations à  $p$  inconnues.

1. Le nombre de pivot dans la méthode de Gauss s'appelle le rang du système. On le note  $\text{rg}(S)$ .
2. Le système  $(S)$  possède une unique solution si et seulement si  $n = p = \text{rg}(S)$ . Dans ce cas, on dit que  $(S)$  est un système Cramer.

## II. — Programmation de la méthode pour résoudre un système de Cramer

On souhaite résoudre un système de Cramer de  $n$  équations à  $n$  inconnues. Pour être en adéquation avec la numérotation de Python qui commence à 0, on écrira un tel système

$$(S) \begin{cases} a_{0,0}x_0 + a_{0,1}x_1 + a_{0,2}x_2 + \cdots + a_{0,n-1}x_{n-1} = b_0 \\ a_{1,0}x_0 + a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n-1}x_{n-1} = b_1 \\ \vdots \\ a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 + \cdots + a_{n-1,n-1}x_{n-1} = b_{n-1} \end{cases}$$

L'écriture matricielle de  $(S)$  est  $AX = B$  avec

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} \\ a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n-1} \end{pmatrix} \quad X = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

### 1) Saisie des données

Pour représenter un système linéaire en Python, on va utiliser l'écriture matricielle et représenter une matrice de taille  $n \times p$  par une liste de  $n$  listes ayant chacune  $p$  éléments.

Par exemple, pour le système  $S_6$ , on a

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 2 & -1 & 2 \\ 4 & 1 & 4 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} -1 \\ -4 \\ -2 \end{pmatrix}$$

et on va représenter la matrice  $A$  par la liste de listes

$$A = [[1, 1, 2], [2, -1, 2], [4, 1, 4]]$$

et la matrice  $B$  par la liste de listes

$$B = [[-1], [-4], [-2]].$$

**Question 1.** Déterminer ce que renvoie chacune des instructions suivantes et vérifier ensuite à l'aide de l'ordinateur.

`A[0]`    `A[0][0]`    `A[1][3]`    `len(A)`    `len(A[2])`    `B[-1]`    `B[1][0]`    `len(B)`

Pour la programmation de la méthode du pivot, il est pratique de considérer la matrice augmentée  $C$  obtenue en ajoutant en dernière colonne de  $A$  les éléments de  $B$ . Ainsi, pour le système  $(S_6)$  la matrice  $C$  est

$$C = \begin{pmatrix} 1 & 1 & 2 & -1 \\ 2 & -1 & 2 & -4 \\ 4 & 1 & 4 & -2 \end{pmatrix}.$$

**Question 2.** Écrire une fonction `matrice_augmentee` qui prend en arguments deux listes de listes  $A$  et  $B$  représentant respectivement une matrice  $A \in \mathcal{M}_n(\mathbb{R})$  et une matrice colonne  $B \in \mathcal{M}_{n,1}(\mathbb{R})$  et qui renvoie la liste de listes représentant la matrice augmentée associée. On prendra garde à écrire une fonction sans effet de bord.

Vérifier ensuite que l'instruction `C = matrice_augmentee(A,B)` affecte bien

$$[[1, 1, 2, -1], [2, -1, 2, -4], [4, 1, 4, -2]]$$

à la variable  $C$ .

## 2) Opérations élémentaires

**Question 3.** Écrire une fonction `echange_lignes` prenant en arguments une liste de listes `M` représentant une matrice  $M \in \mathcal{M}_{n,p}(\mathbb{R})$  ainsi que deux entiers `i` et `j` compris entre 0 et  $n - 1$  et qui renvoie la liste de listes représentant la matrice obtenue à partir de  $M$  en échangeant les lignes  $i$  et  $j$  (les lignes étant numérotées de 0 à  $n - 1$ ).

Vérifier ensuite que `echange_lignes(C, 0, 2)` renvoie bien

$$[[4, 1, 4, -2], [2, -1, 2, -4], [1, 1, 2, -1]].$$

**Question 4.** Écrire une fonction `multiplie_ligne` prenant en arguments une liste de listes `M` représentant une matrice  $M \in \mathcal{M}_{n,p}(\mathbb{R})$  ainsi qu'un entier `i` compris entre 0 et  $n - 1$  et un nombre flottant `t` non nul et qui renvoie la liste de listes représentant la matrice obtenue à partir de  $M$  en remplaçant la ligne  $L_i$  par  $t \times L_i$  (les lignes étant numérotées de 0 à  $n - 1$ ).

Vérifier ensuite que `multiplie_ligne(C, 1, 3)` renvoie bien

$$[[1, 1, 2, -1], [6, -3, 6, -12], [4, 1, 4, -2]].$$

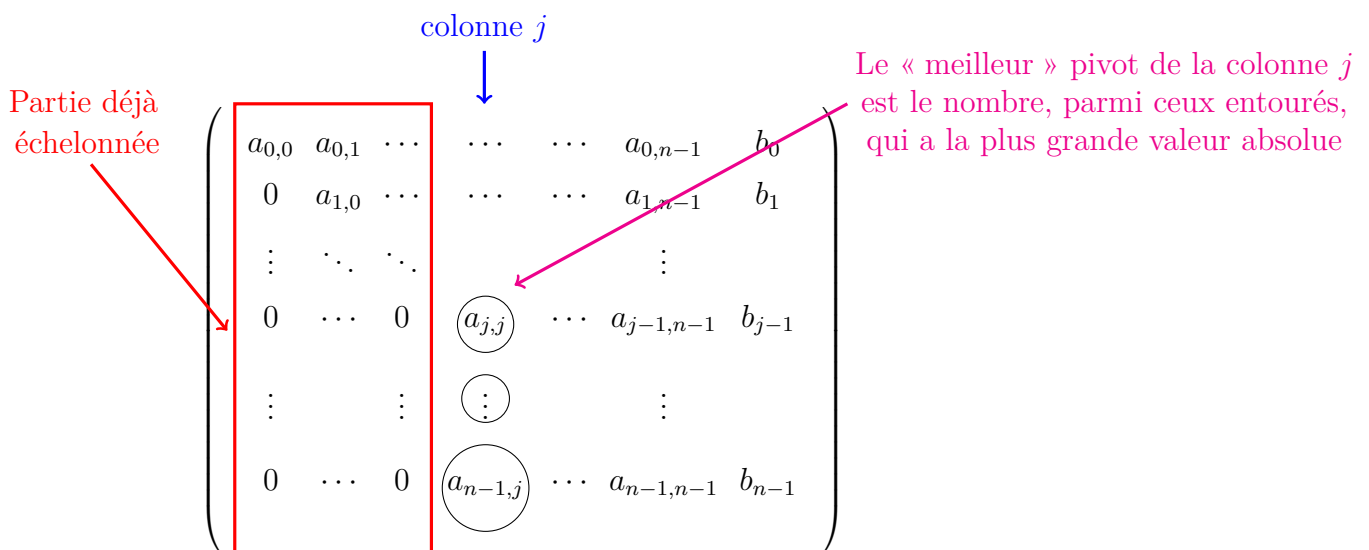
**Question 5.** Écrire une fonction `combine` prenant en arguments une liste de listes `M` représentant une matrice  $M \in \mathcal{M}_{n,p}(\mathbb{R})$  ainsi que deux entiers `i` et `j` compris entre 0 et  $n - 1$  et un nombre flottant `t` et qui renvoie la liste de listes représentant la matrice obtenue à partir de  $M$  en remplaçant la ligne  $L_i$  par  $L_i + t \times L_j$  (les lignes étant numérotées de 0 à  $n - 1$ ).

Vérifier ensuite que `combine(C, 1, 0, -2)` renvoie bien

$$[[1, 1, 2, -1], [0, -3, -2, -2], [4, 1, 4, -2]].$$

## 3) Échelonnement

D'un point de vue mathématique, on peut utiliser n'importe quel nombre non nul comme pivot dans une équation pour éliminer une inconnue dans les autres équations. Cependant, en Python, le calcul sur les nombres flottants se fait de façon approchée et des divisions par des petits nombres peuvent entraîner des erreurs d'arrondies qui, en se cumulant, donnent des résultats totalement faux. On a donc intérêt à éviter les pivots trop petits. Pour cela, on va choisir systématiquement le plus grand pivot possible en valeur absolue.



**Question 6.** Écrire une fonction `cherche_pivot` prenant en arguments une liste de listes `M` représentant une matrice  $M = (M_{k,\ell}) \in \mathcal{M}_{n,p}(\mathbb{R})$  et un entier `j` compris entre 0 et  $n - 1$  et qui renvoie un indice de ligne contenant le plus grand pivot en valeur absolue pour l'indice  $x_j$  c'est-à-dire un entier  $m$  compris entre `j` et  $n - 1$  tel que

$$\forall i \in \llbracket j, n - 1 \rrbracket, \quad |M_{m,j}| \geq |M_{i,j}|.$$

Vérifier que `cherche_pivot(C,0)` renvoie 2.

**Question 7.** On souhaite écrire une fonction `echelonnement` qui prend en arguments deux listes de listes `A` et `B` représentant respectivement une matrice  $A \in \mathcal{M}_n(\mathbb{R})$  et une matrice colonne  $B \in \mathcal{M}_{n,1}(\mathbb{R})$  et qui renvoie la matrice augmentée obtenue à l'issue de l'échelonnement par la méthode du pivot de Gauss.

Compléter le code suivant.

```
def echelonnement(A,B):
    M = matrice_augmentee(A,B)
    n = len(...) # nombre d'équations et d'inconnues
    for i in range(...):
        ligne_pivot = ... # détermination du "meilleur" pivot
        echange_lignes(..., ..., ...) # on échange la ligne i et la
        for k in range(..., ...): # ligne contenant le "meilleur" pivot
            coeff = M[k][i]/M[i][i]
            combine(M, ..., ..., ...) # élimination de x_i sur les
    return M # lignes L_k avec k>i
```

Vérifier que `echelonnement(A,B)` renvoie

`[[4, 1, 4, -2], [0.0, -1.5, 0.0, -3.0], [0.0, 0.0, 1.0, -2.0]]`

#### 4) Résolution du système

Une fois le système échelonné, on termine la résolution par substitution en remontant les équations à partir de la dernière. Pour cela, on va avoir besoin d'utiliser une boucle `for` avec un pas décroissant, contrairement à l'habitude. Ceci est possible avec la syntaxe

```
for k in range(a, b, -1)
```

qui prend toutes les valeurs de `a` à `b+1` avec un pas de 1 en décroissant.

**Question 8.** Déterminer ce qu'affiche le code suivant puis vérifier à l'aide de l'ordinateur.

```
L = [1, 2, 3, 4, 5, 6]
for k in range(5, -1, -1):
    print(L[k])
```

**Question 9.** On souhaite enfin écrire une fonction `resolution` prenant en arguments deux listes de listes `A` et `B` représentant respectivement une matrice  $A \in \mathcal{M}_n(\mathbb{R})$  et une matrice colonne  $B \in \mathcal{M}_{n,1}(\mathbb{R})$  et qui renvoie une liste de listes `X` représentant la matrice colonne  $X \in \mathcal{M}_{n,1}(\mathbb{R})$  solution de  $AX = B$ .

Compléter le code suivant.

```

def resolution(A,B):
    n = len(...) # nombre d'équations
    M = echelonnement(..., ...)
    X = [[...] for i in range(...)] # initialisation de X
    for i in range(..., ..., -1):
        s = 0
        for j in range(i+1,n):
            s = s + M[...] [...] * X[...] [...] # utilisation des inconnues
        X[i] = [(M[...] [...] - ...) / M[...] [...]] # déjà déterminées
    return X

```

Utiliser la fonction `resolution` pour vérifier la solution obtenue pour la système  $(S_6)$ .

**Question 10.** Utiliser la fonction `resolution` avec les systèmes  $(S_7)$  et  $(S_8)$ . Expliquer les résultats obtenus.

## 5) Autres exemples

**Question 11.** Résoudre à la main les systèmes suivants puis vérifier à l'aide de la fonction `resolution`.

$$(S_9) \begin{cases} 4x - 5y + 3z = -4 \\ -5x + 3y - 7z = 5 \\ 6x - 6y + 8z = -8 \end{cases} \quad (S_{10}) \begin{cases} 7x - 12y + 8z = 7 \\ 3x + 2y + 5z = 22 \\ 11x - 3y - 8z = -19 \end{cases} .$$

## III. — Prolongements

### 1) Complexité en temps

La méthode du pivot de Gauss ne dépend pas du nombre d'équations du système : elle fonctionne tout aussi bien pour un système de 2 équations à 2 inconnues que pour un système de 10 équations à 10 inconnues. Cependant, plus le nombre d'équations augmente, plus il y a de calculs à effectuer et plus la résolution va être longue. Le lien entre le nombre  $n$  d'équations et le nombre de calculs à effectuer (ou le temps mis pour les exécuter, qui lui est approximativement proportionnel) s'appelle la complexité en temps de l'algorithme. On peut démontrer que la complexité du pivot de Gauss est cubique, ce qui signifie que, lorsque  $n$  devient grand, le temps d'exécution est approximativement proportionnel à  $n^3$ .

**Question 12.** Écrire une fonction `matrice_aleatoire` qui prend en arguments deux entiers naturels non nuls  $n$  et  $p$  et qui renvoie une liste de listes  $A$  représentant une matrice  $A \in \mathcal{M}_{n,p}(\mathbb{R})$  dont les coefficients sont choisis au hasard et de façon équiprobable dans  $[0; 1[$

*Rappel.* Pour obtenir un flottant aléatoire entre 0 et 1, on utilise la fonction `random()` du module `random`. Il faut donc commencer par l'importer en utilisant l'instruction

```
from random import *
```

**Question 13.** Expliquer le rôle de la fonction suivante.

```
import time

def temps(n):
    A, B = matrice_aleatoire(n,n), matrice_aleatoire(n,1)
    t_initial = time.time()
    resolution(A,B)
    t_final = time.time()
    return t_final-t_initial
```

**Question 14.** Écrire une fonction `temps_moyen` prenant en arguments deux entiers naturels non nuls  $n$  et  $m$  et qui renvoie le temps moyen de résolution de  $m$  systèmes linéaires aléatoires de  $n$  équations à  $n$  inconnues à l'aide de la fonction `resolution`.

**Question 15.** Si on admet que lorsque  $n$  est suffisamment grand, le temps moyen de résolution d'un système de  $n$  équations à  $n$  inconnues par la méthode du pivot est proportionnel à  $n^3$ , quel est le rapport entre le temps moyen de résolution d'un système de  $2n$  équations à  $2n$  inconnues et le temps moyen de résolution d'un système de  $n$  équations à  $n$  inconnues?

Vérifier expérimentalement ce résultat en utilisant la fonction `temps_moyen` avec  $n = 100$  et  $m = 10$  puis  $m = 20$  puis  $m = 50$ .

## 2) Choix du pivot

On a dit précédemment que, d'un point de vue numérique, tous les pivots ne se valent pas et qu'il faut éviter les « petits » pivots. On a donc choisi de prendre, à chaque étape, le plus grand pivot possible en valeur absolue. Cependant, le problème des approximations vient des divisions et, en fait, de ce point de vue, les meilleurs pivots sont 1 et  $-1$ .

**Question 16.** Modifier la fonction `recherche_pivot` de façon à choisir 1 ou  $-1$  comme pivot si cela est possible et le plus grand pivot en valeur absolue sinon.

Utiliser alors la fonction `resolution` pour le système  $(S_8)$ . Que constate-t-on ?

## 3) Résolution par élimination

Une fois le système échelonné, on a terminé la résolution par substitution. Une autre méthode est possible en utilisant des opérations élémentaires pour éliminer les inconnues non seulement en dessous d'une ligne mais également au-dessus. Cela permet d'obtenir la solution du système sur la matrice augmentée à condition de partir d'un système échelonné réduit c'est-à-dire un système échelonné dans lequel tous les pivots sont égaux à 1.

**Question 17.** Écrire une fonction `reduction` qui prend en argument une liste de listes représentant la matrice augmentée obtenue à l'issue de l'échelonnement par la méthode du pivot et qui renvoie une matrice échelonnée équivalente dans laquelle tous les pivots sont égaux à 1.

**Question 18.** Écrire une fonction `resolution_par_elimination` qui prend en arguments deux listes de listes  $A$  et  $B$  représentant respectivement une matrice  $A \in \mathcal{M}_n(\mathbb{R})$  et une matrice  $B \in \mathcal{M}_{n,1}(\mathbb{R})$  et qui renvoie une liste de listes  $X$  représentant la matrice  $X$  solution de  $AX = B$  obtenue par élimination. On pourra utiliser les différentes fonctions définies précédemment.

## 4) Détermination du rang

**Question 19.** En s'inspirant de la fonction `echelonnement`, écrire une fonction `rang` prenant en argument une liste de listes  $A$  représentant une matrice  $A \in \mathcal{M}_n(\mathbb{R})$  et qui renvoie le rang de la matrice  $A$ .