♦ 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 $i \in [\![2, n]\!]$, 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 :

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

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 $i \in [\![1, n]\!]$, 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 $[\![1, n]\!]$. 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 α de x_1 est non nul et on échange L_1 et L_p ;
- on utilise le coefficient α (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 β_i est le coefficient de x_1 dans L_i (si $\beta_i = 0$, il n'y a donc rien à faire) pour tout $i \in [\![2, n]\!]$;
- On réitère le procédé sur le système formé par les $n-1$ dernières équations avec x_2 (ou à défaut x_3, x_4...) 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 $i \in [\![1, n]\!]$ où α_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 $\left(-2\,;\frac{7}{5}\,;\frac{2}{5}\right)$.

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

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

Solution.

$$(S_6) \iff \begin{cases} x+y+2z=-1 & L_1 \\ -3y-2z=-2 & L_2 \leftarrow L_2-2L_1 \\ -3y-4z=2 & L_3 \leftarrow L_3-4L_1 \end{cases} \iff \begin{cases} x+y+2z=-1 & L_1 \\ -3y-2z=-2 & L_2 \\ -2z=4 & L_3 \leftarrow L_3-L_2 \end{cases}$$

$$\iff \begin{cases} x+y+2z=-1 \\ -3y=2z-2 \\ z=-2 \end{cases} \iff \begin{cases} x+y+2z=-1 \\ -3y=-6 \\ z=-2 \end{cases} \iff \begin{cases} x+y+2z=-1 \\ y=2 \\ z=-2 \end{cases}$$

$$\iff \begin{cases} x=-y-2z-1 \\ y=2 \\ z=-2 \end{cases} \iff \begin{cases} x=1 \\ y=2 \\ z=-2 \end{cases}$$

Ainsi, l'unique solution de (S_6) est $(1\,;2\,;-2)$.

$$(S_7) \iff \begin{cases} -x+2y+z=5 & L_1 \leftrightarrow L_2 \\ 2x-3y+4z=-3 & L_2 \leftrightarrow L_1 \\ 4x-5y+14z=1 & L_3 \end{cases} \iff \begin{cases} -x+2y+z=5 & L_1 \\ y+6z=7 & L_2 \leftarrow L_2+2L_1 \\ 3y+18z=21 & L_3 \leftarrow L_3+4L_1 \end{cases}$$

$$\iff \begin{cases} -x+2y+z=5 & L_1 \\ y+6z=7 & L_2 \leftarrow L_2+2L_1 \\ 0=0 & L_3 \leftarrow L_3-3L_2 \end{cases} \iff \begin{cases} -x=-2(-6z+7)-z+5 \\ y=-6z+7 \end{cases}$$

$$\iff \begin{cases} -x=11z-9 \\ y=-6z+7 \end{cases} \iff \begin{cases} x=-11z+9 \\ y=-6z+7 \end{cases}$$

Ainsi, l'ensemble des solutions de (S_7) est $\{(-11z+9\,;-6z+7\,;z) \mid z \in \mathbb{R}\}$.

$$(S_8) \iff \begin{cases} x-y-z=4 & L_1 \\ -y+4z=-7 & L_2 \leftarrow L_2-2L_1 \\ -3y+12z=-19 & L_3 \Leftarrow L_3-5L_1 \end{cases} \iff \begin{cases} x-y-z=4 & L_1 \\ -y+4z=-7 & L_2 \\ 0=2 & L_3 \leftarrow L_3-3L_2 \end{cases}$$

On obtient une égalité fausse sur la ligne 3 donc l'ensemble des solutions de (S_8) est $\varnothing$.

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

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

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)
```

Solution.

- `A[0]` renvoie la liste `[1, 1, 2]`
- `A[0][0]` renvoie l'entier `1`
- `A[1][3]` renvoie un message d'erreur car `A[1]` est une liste de 3 éléments numérotés de 0 à 2 donc `A[1][3]` n'existe pas.
- `len(A)` renvoie `3`
- `len(A[2])` renvoie `3`
- `B[-1]` renvoie la liste `[-2]`
- `B[1][0]` renvoie l'entier `-4`
- `len(B)` renvoie `3`

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 liste de listes `A` et `B` représentant respectivement une matrice $A \in \mathscr{M}_n(\mathbb{R})$ et une matrice colonne $B \in \mathscr{M}_{n,1}(\mathbb{R})$ 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`.

Solution.

```
def matrice_augmentee(A,B):
  M = []
  for k in range(len(A)):
    M.append(A[k] + B[k])
  return M
```

2) Opérations élémentaires

Question 3. Écrire une fonction `echange_lignes` prenant en arguments une liste de listes `M` représentant une matrice $M \in \mathscr{M}_{n,p}(\mathbb{R})$ 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]].`

Solution.

```
def echange_lignes(M,i,j):
  M[i],M[j]=M[j],M[i]
  return M
```

Question 4. Écrire une fonction `multiplie_ligne` prenant en arguments une liste de listes `M` représentant une matrice $M \in \mathscr{M}_{n,p}(\mathbb{R})$ 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]].`

Solution.

```
def multiplie_ligne(M,i,t):
  for k in range(len(M[i])):
    M[i][k]= t*M[i][k]
  return M
```

Question 5. Écrire une fonction `combine` prenant en arguments une liste de listes `M` représentant une matrice $M \in \mathscr{M}_{n,p}(\mathbb{R})$ 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

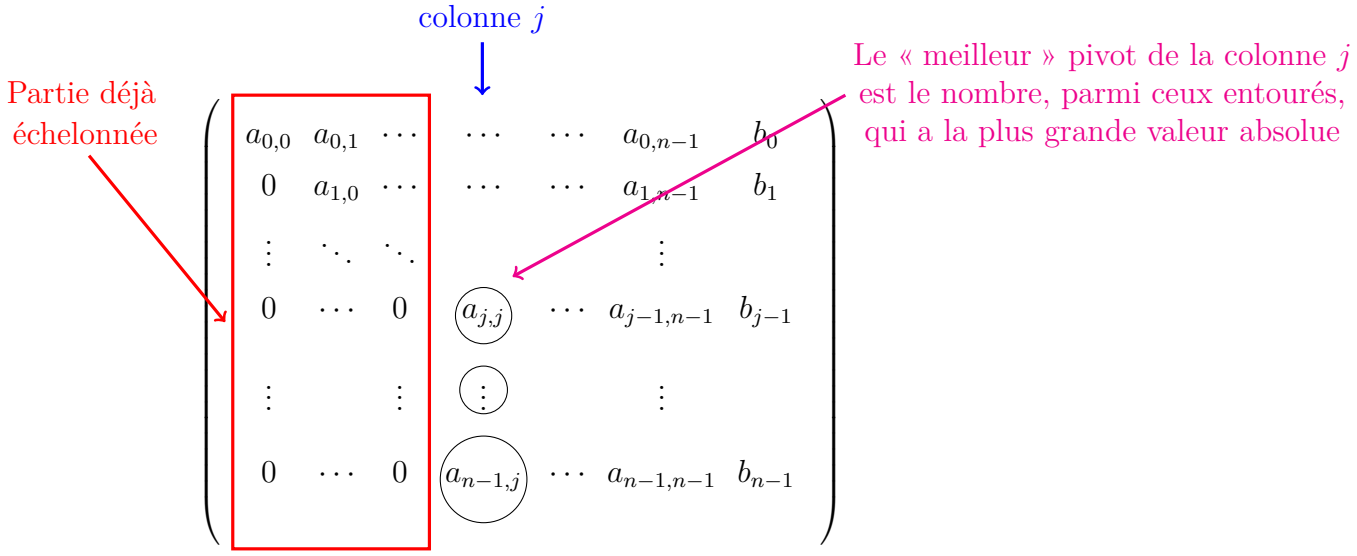
`[[1, 1, 2, -1], [0, -3, -2, -2], [4, 1, 4, -2]].`

Solution.

```
def combine(M,i,j,t):
  for k in range(len(M[i])):
    M[i][k] = M[i][k] + t*M[j][k]
  return M
```

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 $M = (M_{k,\ell}) \in \mathscr{M}_{n,p}(\mathbb{R})$ 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

$$\forall i \in [\![j, n-1]\!], \quad |M_{m,j}| \geqslant |M_{i,j}|\,.$$

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

Solution.

```
def cherche_pivot(M,j):
  m = j
  for k in range(j+1, len(M)):
    if abs(M[k][j]) > abs(M[m][j]):
      m = k
  return m
```

Question 7. On souhaite écrire une fonction `echelonnement` qui prend en arguments deux listes de listes `A` et `B` représentant respectivement une matrice $A \in \mathscr{M}_n(\mathbb{R})$ et une matrice colonne $B \in \mathscr{M}_{n,1}(\mathbb{R})$ 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]]
```

Solution.

```
def echelonnement(A,B):
  M = matrice_augmentee(A,B)
  n = len(A)
  for i in range(n):
    ligne_pivot = cherche_pivot(M,i)
    echange_lignes(M,i,ligne_pivot)
    for k in range(i+1,n):
      coeff = M[k][i]/M[i][i]
      combine(M,k,i,-coeff)
  return M
```

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])
```

Solution. On obtient à l'affichage les termes de liste `L` énumérés en partant de la fin.

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

Solution.

```
def resolution(A,B):
    n = len(A)
    M = echelonnement(A, B)
    X = [[0] for i in range(n)]
    for i in range(n-1, -1, -1):
        s = 0
        for j in range(i+1,n):
            s = s + M[i][j] * X[j][0]
        X[i] = [(M[i][n]-s)/M[i][i]]
    return X
```

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

Solution. Pour (S_7), on obtient un message d'erreur due à une tentative de division par 0. C'est cohérent car on a vu dans l'exemple 11 n'est pas un système de Cramer et que son rang est 2. Ainsi, lors de l'échelonnement, on va obtenir une ligne ne contenant pas de pivot (tous les coefficients sont nuls) et la fonction `resolution` va donc utiliser 0 comme « pivot », ce qui n'est pas possible.

Pour (S_8), on obtient la liste de listes

```
[[-7505999378950826.0], [-6004799503160662.0], [-1501199875790167.2]]
```

ce qui n'est pas cohérent puisque (S_8) n'est pas de solution. Le problème vient des erreurs d'arrondis et met en lumière les limites du calcul numérique en Python.

5) Autres exemples

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

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

Solution.

$$(S_9)\iff\begin{cases}4x-5y+3z=-4 & L_1\\ -\frac{13}{4}y-\frac{13}{4}z=0 & L_2\leftarrow L_2+\frac{5}{4}L_1\\ \frac{3}{2}y+\frac{7}{2}z=-2 & L_3\leftarrow L_3-\frac{3}{2}L_1\end{cases}\iff\begin{cases}4x-5y+3z=-4 & L_1\\ -\frac{13}{4}y-\frac{13}{4}z=0 & L_2\\ 2z=-2 & L_3\leftarrow L_3+\frac{6}{13}L_1\end{cases}$$
$$\iff\begin{cases}4x-5y+3z=-4\\ y=-z\\ z=-1\end{cases}\iff\begin{cases}4x=5y-3z-4\\ y=1\\ z=-1\end{cases}\iff\begin{cases}4x=4\\ y=1\\ z=-1\end{cases}\iff\begin{cases}x=1\\ y=1\\ z=-1\end{cases}$$

Ainsi, l'unique solution de (S_9) est $(1\,;1\,;-1)$. On vérifie qu'on obtient bien le même résultat à l'aide de la fonction `resolution`.

$$(S_{10})\iff\begin{cases}7x-12y+8z=7 & L_1\\ \frac{50}{7}y+\frac{11}{7}z=19 & L_2\leftarrow L_2-\frac{3}{7}L_1\\ \frac{111}{7}y-\frac{144}{7}z=-30 & L_3\leftarrow L_3-\frac{11}{7}L_1\end{cases}\iff\begin{cases}7x-12y+8z=7 & L_1\\ \frac{50}{7}y+\frac{11}{7}z=19 & L_2\\ -\frac{1203}{50}z=-\frac{3609}{50} & L_3\leftarrow L_3-\frac{111}{50}L_1\end{cases}$$
$$\iff\begin{cases}7x-12y+8z=7\\ \frac{50}{7}y=19-\frac{11}{7}z\\ z=3\end{cases}\iff\begin{cases}7x-12y+8z=7\\ \frac{50}{7}y=\frac{100}{7}\\ z=3\end{cases}\iff\begin{cases}7x=12y-8z+7\\ y=2\\ z=3\end{cases}$$
$$\iff\begin{cases}7x=7\\ y=2\\ z=3\end{cases}\iff\begin{cases}x=1\\ y=2\\ z=3\end{cases}$$

Ainsi, l'unique solution de (S_{10}) est $(1\,;2\,;3)$. On vérifie qu'on obtient bien le même résultat à l'aide de la fonction `resolution`. Précisément, on obtient

```
[[1.0], [1.9999999999999996], [3.0]]
```

l'imprécision de la seconde valeur étant dû au erreur d'arrondi du calcul en flottant.

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 naturel non nuls `n` et `p` et qui renvoie une liste de listes `A` représentant une matrice $A \in \mathscr{M}_{n,p}(\mathbb{R})$ 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 importer en utilisant l'instruction

```
from random import *
```

Solution.

```
from random import *

def matrice_aleatoire(n,p):
  M = [[random() for i in range(p)] for j in range(n)]
  return M
```

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
```

Solution. La fonction `temps` calcule le temps de résolution par la méthode du pivot d'un système de n équations à n inconnues dont les coefficients sont des nombres flottants aléatoires entre 0 et 1.

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`.

Solution.

```
def temps_moyen(n,m):
  T = 0
  for k in range(m):
    T += temps(n)
  return T/m
```

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 la 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 $m = 50$.

Solution. Notons $T(n)$ le temps moyen de résolution d'un système de n équations à n inconnues par la méthode du pivot de Gauss. Pour n assez grand, on a $T(n) \approx n^3$ donc $T(2n) \approx (2n)^3 = 8n^3$ donc $T(2n)/T(n) \approx 8$.

Ceci se confirme expérimentalement. En effet,

- en exécutant `print(temps_moyen(200,10)/temps_moyen(100,10))`, on a obtenu

  `8.411251550032349`

- en exécutant `print(temps_moyen(200,20)/temps_moyen(100,20))`, on a obtenu

  `8.239061921974065`

- en exécutant `print(temps_moyen(200,50)/temps_moyen(100,50))`, on a obtenu

  `8.09534974090762`.

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.

Solution.

```
def cherche_pivot(M,j):
  m = j
  for k in range(j, len(M)):
    if abs(M[k][j]) == 1:
      return k
    elif abs(M[k][j]) > abs(M[m][j]):
      m = k
  return m
```

En utilisant la fonction `resolution` pour le système (S_8), on obtient une erreur de type « tentative de division par 0 », ce qui n'était pas le cas avec la version initiale de cette fonction (voir la **Question 10.**). Ici, le fait de choisir le pivot 1 plutôt que le plus grand pivot en valeur absolue à éviter des approximations lors de division et donner une résultat cohérent.

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.

Solution.

```
def reduction(M):
  n = len(M)
  for i in range(n):
    multiplie_ligne(i, 1/M[i][i])
  return M
```

Question 18. Écrire une fonction `resolution_par_elimination` qui prend en arguments deux listes de listes `A` et `B` représentant respectivement une matrice $A \in \mathscr{M}_n(\mathbb{R})$ et une matrice $B \in \mathscr{M}_{n,1}(\mathbb{R})$ 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.

Solution.

```
def resolution_par_elimination(A,B):
  n = len(A)
  M = echelonnement(A,B)
  N = reduction(M)
  for j in range(1,n):
    for i in range(j):
      combine(N,i,j,-N[i][j])
  X = []
  for i in range(n):
    X.append([N[i][n]])
  return X
```

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 $A \in \mathscr{M}_n(\mathbb{R})$ et qui renvoie le rang de la matrice A.

Solution.

```
def rang(A):
  n = len(A)
  for i in range(n):
    ligne_pivot = cherche_pivot(A,i)
    echange_lignes(A,i,ligne_pivot)
    if A[i][i] != 0:
      for k in range(i+1,n):
        coeff = A[k][i]/A[i][i]
        combine(A,k,i,-coeff)
  r = 0
  for i in range(n):
    if A[i][i] != 0:
      r += 1
  return r
```