La simulation numérique
Algèbre linéaire
Résolution des systèmes linéaires sous Python

Anis SAIED
Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul (IPEIN),
Université de Carthage,
Tunisie
anis_saied@hotmail.com

2025/26

1 Introduction

Dans ce chapitre, on se propose de programmer en `Python` une méthode de résolution de systèmes linéaires inversible de **n** équations à **n** inconnues S (Système de Cramer) de la forme :

$$\mathbf{S} = \begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \qquad\qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

Nous noterons ce système sous la forme matricielle : $Ax = b$
Où A , la matrice d'ordre **n** à coefficients $a_{i,j}$ avec $1 \leq (i,j) \leq n$; x le vecteur à déterminer ayant les coefficients x_i et b le vecteur à coefficients b_i avec $1 \leq i \leq n$.

$$\mathbf{A} = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \ldots & a_{nn} \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

2 Résolution du système linéaire par la méthode du pivot de Gauss

2.1 Description de la méthode

Le pivot de Gauss est une méthode qui peut s'appliquer sur des matrices ou sur des systèmes d'équations.

Le but de cette méthode est de transformer la matrice augmentée $M = (A|B)$ (qui a n lignes et $n+1$ colonnes) en une matrice augmentée équivalente $M' = (A'|B')$ où A' est une matrice triangulaire supérieure à coefficients diagonaux non nuls), puis de résoudre le système linéaire associé à cette nouvelle matrice augmentée. L'algorithme du pivot de Gauss se déroule donc en deux étapes :

2.1.1 Etape 1 : Triangularisation

Cette étape consiste à transformer progressivement la matrice A par opérations élémentaires sur les lignes en une matrice triangulaire supérieure par des itérations successives.

Si on désigne par L_i la $i^{ème}$ ligne du système linéaire, les opérations élémentaires sont :

- Permuter les lignes i et j de A ainsi que celle de b : $L_i \leftrightarrow L_j$
- Multiplier la ligne i par λ (un scalaire non nul) : $L_i \leftarrow \lambda L_i$
- Remplacer la ligne i par $L_i + \lambda L_j$: $L_i \leftarrow L_i + \lambda L_j$

Ces opérations effectuées dans n'importe quel ordre et en nombre quelconque, laissent invariant l'ensemble des solutions du système linéaire.

Le système devient alors triangulaire :

$$\left\{\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ & & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ & & & & \vdots & & & & \\ & & & & & & a_{nn}x_n & = & b_n \end{array}\right.$$

Essayer d'appliquer le principe qui suit sur ce système linéaire :

$$S = \left\{\begin{array}{ccccccc} x & - & 3y & - & 2z & = & 6 \\ 2x & - & 4y & - & 3z & = & 8 \\ -3x & + & 6y & + & 8z & = & -5 \end{array}\right.$$

Soit $M = (A|B)$:

$$\left(\begin{array}{ccc|c} 1 & -3 & -2 & 6 \\ 2 & -4 & -3 & 8 \\ -3 & 6 & 8 & -5 \end{array}\right)$$

Itération 1

- On cherche un coefficient a_{11} devant x_1 qui soit non nul, appelé *premier pivot*. Si ce coefficient apparaît en ligne i, on permute les lignes 1 et la ligne i.
- On élimine par la suite tous les coefficients devant x_1 à partir de la ligne 2 jusqu'à la ligne n en appliquant la relation $L_i \leftarrow L_i - \frac{a_{i1}}{a_{11}} L_1$ aux coefficients des lignes entre 2 à n. L_i représente les coefficients de la ligne i de A .

A la fin de cette itération la matrice apparaît comme suit :

$$\left(\begin{array}{ccccccc|c} a'_{11} & a'_{12} & \dots & \dots & \dots & a'_{1n} & & b'_1 \\ 0 & a'_{22} & \dots & \dots & \dots & a'_{2n} & & b'_2 \\ \vdots & a'_{32} & \ddots & & & \vdots & & \vdots \\ 0 & \vdots & \vdots & a'_{jj} & \dots & a'_{jn} & & b'_j \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ 0 & a'_{n2} & \dots & a'_{nj} & \dots & a'_{nn} & & b'_n \end{array}\right)$$

A l'itération suivante, la ligne 1 reste inchangée, et on réapplique le même processus à la sous matrice à partir de la ligne 2.

On définit donc un script qui à partir d'une matrice M (à n lignes et $n+1$ colonnes) construit une matrice équivalente $M' = (A'|B')$ où A' est carrée triangulaire supérieure. L'algorithme de triangularisation fait apparaître des zéros sous la diagonale, en procédant de haut vers le bas et de gauche à droite.

Si on note L_k la k^e ligne de la matrice M, pour $k \in [1; n]$, et m_{ij} le coefficient en i^e ligne et j^e colonne de M, pour $(i, j) \in [1, n] \times [1; n+1]$, l'algorithme s'écrit comme suit :

```
Pour k de 1 à n−1
  Si m_kk = 0
    Chercher un pivot m_pk ≠ 0 avec p ∈ [k+1; n]
    si un pivot m_pk ≠ 0 existe
       permuter les lignes L_k et L_p
    sinon
       Afficher "PIVOT NUL STOP"
    fin si
  Fin Si

  Si m_kk est différent de 0
    Pour i de k+1 à n
      d ← m_ik / m_kk
      Pour j de k à n+1
          m_ij ← m_ij − d m_kj
      Fin Pour
    Fin Pour
  fin si
Fin Pour
```

2.1.2 Etape 2 : Remontée

La deuxième étape de remontée consiste à résoudre, de bas en haut, le système triangulaire obtenu S', équivalent à S, dont $M' = (A'|B')$ est la matrice augmentée, qui est de la forme :

$$\mathbf{S'} = \begin{cases} a'_{11}x_1 + a'_{12}x_2 + \ldots + a'_{1n}x_n = b'_1 \\ a'_{22}x_2 + \ldots + a'_{2n}x_n = b'_2 \\ \ddots \quad \vdots \\ a'_{nn}x_n = b'_n \end{cases}$$

avec $a'_{11} \neq 0, \ldots, a'_{nn} \neq 0$.
Ce système est triangulaire, sa solution s'obtient par substitutions successives. On commence par la résolution de la dernière équation, à une seule inconnue, pour déterminer le $x_n = \frac{b'_n}{a'_{nn}}$ puis, on substitue x_n dans l'équation de la ligne $n-1$ pour déterminer x_{n-1} et ainsi de suite.

A chaque étape, on substitue aux inconnues d'une ligne les valeurs trouvées dans les lignes inférieures pour résoudre l'équation courante jusqu'à trouver toutes les solutions.

On calcule donc les valeurs de $x_1, \ldots, x_n$ par récurrence descendante.
Nous avons :

$$\begin{cases} x_n = \dfrac{b'_n}{a'_{nn}} \\ x_i = \dfrac{1}{a'_{ii}} \left(b'_i - \displaystyle\sum_{j=i+1}^{n} a'_{ij}xj \right) \end{cases} \qquad \forall i \in \{n-1, \ldots, 1\}$$

L'algorithme qui correspond à cette deuxième étape est donc le suivant, sachant que les matrices A' et B' sont connues et que $X[k]$ désigne la k^e composante de X :

$X = 0$ # Matrice unicolonne nulle

$X[n] \leftarrow \dfrac{b'_n}{a'_{nn}}$

Pour i de $n-1$ à 1 faire

  $somme \leftarrow b'_i$

  Pour j de $i+1$ à n faire

    $somme \leftarrow somme - a'_{ij} X[j]$

  Fin Pour

  $X[i] \leftarrow \dfrac{somme}{a'_{ii}}$

Fin Pour

Afficher X

2.2 Traduction dans le langage python

2.2.1 Implémentation de la méthode du pivot de Gauss en python

Hypothèses de travail :

- On peut résoudre toutes les parties en utilisant des listes et des listes de listes mais dans le cadre de ce travail on vous demande d'utiliser les tableaux numpy.
- Penser à travailler sur une copie de la matrice et du vecteur pour ne pas altérer les données d'origine. Modifier une donnée d'une extraction de tableau entraîne aussi une modification du tableau initial. Pour remédier à ce problème on peut utiliser la commande `np.copy()`

Travail à faire :

1. Ecrire une fonction python sans paramètres, appelée **taille**, qui permet de saisir et retourner un entier ≥ 2.
2. Ecrire une fonction python, nommée **saisie_Vect** qui permet la saisie des n coefficients réels d'un vecteur. Cette fonction doit retourner une matrice B unicolonne d'ordre $(n, 1)$.
3. Ecrire une fonction python, nommée **saisie_Mat** qui permet la saisie des coefficients réels d'une matrice carrée d'ordre n. Cette fonction doit retourner une matrice A carrée d'ordre (n, n).
4. Ecrire une fonction python, nommée **triangulaire**, qui prend en paramètres une matrice **M** d'ordre $(n, n+1)$ et qui rend triangulaire supérieure la matrice **M**.
5. Ecrire une fonction python, nommée **remontee**, qui prend en paramètres une matrice **T** d'ordre $(n, n+1)$ triangulaire supérieure et retourne un vecteur **X** solution du système triangulaire.
6. Ecrire une fonction python, nommée **pivot_Gauss**, qui prend en paramètres une matrice **A** et un vecteur **B** et retourne le vecteur **X**, solution du système.
7. Déterminer la matrice des inconnus X :

$$A = \begin{pmatrix} 2 & -5 & 1 & 3 \\ 4 & 7 & 8 & 2 \\ 3 & 1 & 1 & 6 \\ 4 & 1 & 7 & 9 \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \qquad B = \begin{pmatrix} 5 \\ 10 \\ 2 \\ 6 \end{pmatrix}$$

3 Déjà disponible en Python

Comme pour tous les algorithmes de ce chapitre, nous ne ferons que réimplémenter des fonctions déjà existantes en Python. Concernant les méthodes de résolution des systèmes linéaires, tout se trouve dans le module `numpy.linalg`. Il contient entre autres les fonctions suivantes :

`solve(A,B)` : Résoudre le système linéaire $AX = B$