

SYSTÈMES LINÉAIRES : MÉTHODE DU PIVOT DE GAUSS

On souhaite résoudre numériquement un système linéaire carré, de n équations et n inconnues :

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \dots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + \dots + a_{2,n}x_n = b_2 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + a_{n,3}x_3 + \dots + a_{n,n}x_n = b_n \end{cases}$$

On se place dans le cas d'un système de Cramer (une unique solution).

On va utiliser la méthode du pivot de Gauss

I. Principe

- On met le système sous forme triangulaire en utilisant des transvections (opérations sur les lignes de la forme $L_j \leftarrow L_j + \lambda L_i$).
- On résout le système triangulaire obtenu (remontée).

II. Implémentation

On représente le système par :

- la matrice **A** de ses coefficients (liste des listes des coefficients de chaque ligne).
- la liste **B** représentant le second membre

Pour pouvoir aisément manipuler les lignes de **A**, on utilise plutôt la structure **array** du module **numpy**.

Par exemple, le système

$$\begin{cases} 2x + y + z = 3 \\ x + 2y + z = 1 \\ 2x + y + 4z = 9 \end{cases}$$

sera représenté par :

```
A = np.array([[2., 1., 1.], [1., 2., 1.], [2., 1., 4.]])
B = np.array([3., 1., 9.])
```

Pour limiter l'instabilité numérique, on choisit systématiquement le pivot de plus grande valeur absolue (en échangeant deux lignes).

(programme page suivante)

III. Complexité

- Coût des recherches de pivot : n puis $n - 1$ puis ... puis 2 donc en n^2
- Coût des transvections :
 - $(n - 1)$ puis $(n - 2)$ puis .. puis 1 transvections
 - Coût d'une transvection : opération sur des **array** de taille n donc en n
 - **Coût total des transvections en n^3**
- Coût des éventuels échanges de lignes :
 - Au plus $n - 1$ échanges
 - Coût d'un échange avec la structure utilisée : constant
 - Coût dans le pire des cas en n (n^2 si modification coefficient par coefficient).
- Coût de la remontée
 - 0 puis 1 puis ... puis $n - 1$ soustractions/multiplications
 - n divisions
 - Coût total en n^2

Finalement, l'algorithme du pivot de Gauss pour résoudre un système de Cramer de n équations à n inconnues a une complexité en n^3 .

Programme

```
import numpy as np

def pivot(A, B):
    n = len(A)
    # Mise sous forme triangulaire
    for i in range(n-1):
        # Choix du pivot : plus grande valeur absolue
        m = A[i][i]
        imax = i # indice ligne pivot
        for j in range(i+1, n):
            if abs(A[j][i]) > m:
                m = abs(A[j][i])
                imax = j
        # Echange des lignes i et imax
        A[i], A[imax] = A[imax], A[i]
        B[i], B[imax] = B[imax], B[i]
        # Manipulation des lignes
        for j in range(i+1, n):
            c = A[j][i] / A[i][i]
            A[j] = A[j] - c*A[i]
            B[j] = B[j] - c*B[i]
    # Remontée
    X=np.array([0. for i in range(n)])
    for i in range(n-1,-1,-1):
        X[i]= B[i]
        for j in range(i+1, n):
            X[i] = X[i] - A[i][j]*X[j]
        X[i]= X[i] / A[i,i]
    return X
```