Simulation Numérique 4 :

Méthode de Gauss

Le but de ce chapitre est de résoudre des problèmes discrets multidimensionnels linéaires conduisant à la résolution d'un système linéaire inversible (ou de Cramer) par la méthode du pivot de Gauss avec recherche partielle du pivot.

I Rappels sur la méthodes de Gauss

On résout un système $Ax = b$ par la méthode du pivot de Gauss, $A \in \mathcal{M}_n(\mathbb{K})$ supposée inversible (système de Cramer).

La méthode présentée sert aussi à déterminer A^{-1} en effectuant les opérations en parallèle sur I_n au lieu de b.

On commence par chercher un pivot (non nul, bien sûr – ⚠ pour les matrices de flottants, il ne faut donc pas que les cœfficients soient trop petits...) dans la première colonne (il y en a au moins un car A est inversible).

Dans les opérations suivantes, il faudra diviser par ce pivot. Plus il est petit, plus cela risque de créer des problèmes numériques (avec les flottants). On a donc tout intérêt dans la pratique à la choisir le plus grand possible.

Conformément aux convention informatiques (et non mathématique), on numérote les éléments à partir de 0.

Echelonnement(A,b)

Pour j *allant de* 0 *à* $n-2$ **Faire**
- Chercher $i \in [\![j, n-1]\!]$ tel que $|a_{i,j}|$ est maximal
- $L_i \leftrightarrow L_j$ dans A et b
- **Pour** k *de* $j+1$ *à* $n-1$ **Faire**
  - $L_k \leftarrow L_k - \dfrac{a_{k,j}}{a_{j,j}} L_j$ dans A et b
- **FinPour**

FinPour

Invariant de sortie de la j^{e} étape : A est de la forme

$$\begin{pmatrix} * & \dots & * & * & \dots & \dots & * \\ 0 & \ddots & \vdots & \vdots & & & \vdots \\ \vdots & \ddots & * & \vdots & & & \vdots \\ \vdots & & 0 & \vdots & & & \vdots \\ \vdots & & \vdots & \vdots & & & \vdots \\ 0 & \dots & 0 & * & \dots & & * \end{pmatrix}$$

(Le premier bloc étant de taille $j+1$)

Complexité : Pour chaque $j \in [\![0, n-1]\!]$,

- Recherche du pivot partiel : $O(n)$ comparaisons,
- Échange de lignes : $O(n)$ affectations,
- Transvections : $O(n)$ pour au plus n transvections, soit $O\left(n^2\right)$.

Soit en tout $O\left(n^3\right)$ opérations élémentaires.

On a alors un système de matrice triangulaires supérieure, avec des cœfficients non nuls sur la diagonale.

On termine la résolution pour se ramener à I_n afin de lire la solution dans le second membre.

⚠ : risque d'instabilité numérique.

FinResolution(A,b)

Pour j *allant de* $n-1$ *à* 0 **Faire**
  $L_j \leftarrow \frac{1}{a_{j,j}} L_j$ dans A et b
  Pour k *de* 0 *à* $j-1$ **Faire**
    $L_k \leftarrow L_k - a_{k,j} L_j$ dans A et b
  FinPour
FinPour

Complexité : À nouveau $O\left(n^3\right)$ opérations élémentaires.
Invariant de sortie : A est de la forme

$$\begin{pmatrix} * & \dots & * & 0 & \dots & 0 \\ & \ddots & \vdots & \vdots & & \vdots \\ & & * & 0 & & \vdots \\ & & & 1 & \ddots & \vdots \\ & (0) & & & \ddots & 0 \\ & & & & & 1 \end{pmatrix}$$

(Le deuxième bloc étant de taille $n-j$)

À la fin de l'algorithme, b contient la solution à notre problème (qui est unique).

II IMPLÉMENTATION EN PYTHON

```python
import numpy as np
from time import time

# Opérations élémentaires

def dilatation(A, i, alpha):
    """L[i] <-- alpha*L[i]"""
    A[i, :] *= alpha


def dilatation2(A, i, alpha):
    """L[i] <-- alpha * L[i]"""
    for k in range(A.shape[1]):
        A[i,k] *= alpha # Attention aux types

def echange(A, i, j):
    """L[i] <--> L[j]"""
    L=A[i, :].copy()
    A[i, :] = A[j, :]
    A[j, :] = L

def echange2(A, i, j):
    """L[i] <--> L[j]"""
    for k in range(A.shape[1]):
        A[i, k], A[j, k] = A[j, k], A[i, k]


def transvection(A, i, j, mu):
    """L[i] <- L[i]+mu*L[j]"""
    A[i, :] += mu * A[j, :]

def pivot_partiel(A, j):
    """Recherche de l'indice tel que |A[k,j]| soit maximal pour k>j"""
    indice = j
    for k in range(j + 1, A.shape[0]):
        if abs(A[k, j]) > abs(A[indice, j]):
            indice = k
    return indice
```


DEF

```
def pivot(A, b):    """A doit être une matrice (carrée) inversible, b \
 est soit un
    vecteur soit une matrice.
    Si b est un vecteur, l'agorithme renvoie la solution
    du système A x = b.
    Si b est la matrice identité, l'agorithme renvoie l'inverse
    de A."""

    n, p = A.shape
    assert n == p # A doit être carrée.

    # création la matrice C=(A b) avec conversion en flottants pour
    # éviter les problèmes de type.
    C = np.concatenate((A.astype(float), b.astype(float)), axis=1)

    # Echelonnement
    for j in range(n - 1):
        k = pivot_partiel(C, j)
        if k != j: echange(C, j, k) # Si le pivot est déjà à sa place,
                                    # inutile de faire l'échange.
        for i in range(j + 1, n):
            assert C[j, j] != 0 # A doit être inversible.
            mu = - C[i, j] / C[j, j]
            transvection(C, i, j, mu)

    # Les coefficients diagonaux sont changés en 1.
    for i in range(n): dilatation(C, i, 1 / C[i, i])

    # Fin de la résolution.
    for j in range(n - 1, 0, -1):
        for i in range(j): transvection(C, i, j, -C[i,j])

    # Extraction de la solution
    return C[:, n:]
```

III PROBLÈMES NUMÉRIQUES LIÉS À LA RÉSOLUTION

On remarque qu'avec certaines matrices, une petite perturbation du second membre et de la matrice donne des résultats très éloignés.

C'est le cas par exemple pour le matrice $H_n = \left(\frac{1}{i+j-1}\right)$ de Hilbert.

On peut démontrer que l'erreur peut-être contrôlée par le **conditionnement** de la matrice du système.

Sans entrer dans les détails, il s'écrit $c(A) = |||A||| \times \left|\left|\left|A^{-1}\right|\right|\right|$ où

$$|||A||| = \sup_{x\in\mathbb{R}^n, x\neq 0} \frac{\|Ax\|}{\|x\|} \quad \text{et} \quad \|x\| = \sqrt{x_1^2 + \cdots + x_n^2}.$$

On peut démontrer que si A est symétrique réelle, de valeurs propres telles que $|\lambda_1| < \cdots < |\lambda_n|$, alors $c(A) = \frac{|\lambda_n|}{|\lambda_1|}$.

On a toujours $c(A) \geqslant 1$.

On peut démontrer que si on perturbe le second membre du système, alors on obtient comme solution $A(x+\delta x) = b + \delta b$ où $Ax = b$ tel que

$$\frac{\|\delta x\|}{\|x\|} \leqslant c(A)\frac{\|\delta b\|}{\|b\|}.$$

Ainsi, plus $c(A)$ est proche de 1, moins les imprécisions sont amplifiées.