TP d'informatique n°20

Pivot de Gauss

L'objectif du TP est de programmer et tester différentes méthodes pour résoudre numériquement des systèmes linéaires. Nous mettrons également en place des algorithmes utilisant le même principe de pivot de Gauss que pour la résolution de système.

Les différentes bibliothèques utilisées seront :

- math pour les fonctions cos, exp, sin et log (ou numpy pour ces mêmes fonctions)
- numpy.linalg.solve pour résoudre les systèmes
- time pour la fonction time.
- numpy pour les fonctions array, linspace et les fonctions usuelles chargé avec l'alias np.
- sympy pour les fonctions Matrix, et les méthodes det, inv, minorMatrix qui s'appliquent aux objets de ce type.
- matplotlib.pyplot pour plot, show…… chargé avec l'alias plt.

I) Entrainement

Résoudre à la main les systèmes suivants :

$$\begin{cases} 2x + y = 5 \\ 5x - 2y = -16 \end{cases} \quad \text{et} \quad \begin{cases} 2x + 2y - 3z = 2 \\ -2x - y - 3z = -5 \\ 6x + 4y + 4z = 16 \end{cases}$$

II) Mise en place du pivot de Gauss pour la résolution de systèmes

1) **R**eprendre les procédures du cours recherche-pivot, echange_lignes, transvections_lignes, resolution et calculer leurs complexités en fonction de la dimension des matrices auxquelles elles s'appliquent.

2) Tester le programme résolution sur les systèmes traités manuellement. Comparer avec les résultats obtenus avec la fonction solve de la librairie numpy.linalg

III) Calculs de déterminants

1) **E**n s'inspirant de la procédure "resolution" mettant en place le pivot de Gauss, écrire une fonction Det_Pivot qui prend pour argument une matrice carrée et qui retourne le déterminant de cette matrice en utilisant la méthode du pivot de Gauss. **C**alculer le nombre d'opérations nécessaires pour calculer le déterminant d'une matrice de taille (n,n) quelconque. Attention : cette procèdure doit fonctionner aussi pour les matrices non inversibles.

2) **E**crire une procédure Det_3 qui prend comme argument une matrice carrée de hauteur 3 et qui retourne son déterminant à l'aide de la formule de Sarrus.

Ecrire une procédure Det_4 qui prend comme argument une matrice (4,4) et qui retourne le déterminant de cette matrice en développant suivant la première ligne (cette procédure fait appel à la procédure Det_3 déjà vue). **C**alculer le nombre d'opérations nécessaires.

3) Imaginons que l'on ait écrit Det_5, Det_6, Det_7, Det_8 etc…, suivant le même modèle que Det_4, calculer le nombre u_{n+1} d'opérations nécessaires pour calculer un déterminant (n+1, n+1) en fonction du nombre u_n d'opérations nécessaires pour calculer un déterminant (n, n). En déduire u_n en fonction de n et comparer avec la complexité de Det_Pivot

IV) Calcul de l'inverse d'une matrice

1) **E**n s'inspirant de la procédure "resolution" mettant en place le pivot de Gauss, écrire une fonction Inverse_Pivot qui prend pour argument une matrice carrée et qui retourne l'inverse de cette matrice si elle est inversible (et un message d'erreur sinon) en utilisant la méthode du pivot total. **C**alculer le nombre d'opérations nécessaires pour calculer l'inverse d'une matrice de taille (n,n) supposée inversible.

2) **C**alculer à l'aide de votre procédure, les inverses des matrices :

$$\begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix}, \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \begin{pmatrix} 2 & 2 & -3 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 & -3 \\ -2 & -1 & -3 \\ 6 & 4 & 4 \end{pmatrix},$$

3) **C**alculer à l'aide de votre procédure, les inverses des matrices :

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \begin{pmatrix} 1 & 1/4 & 1 \\ 1 & 1/3 & 2 \\ 0 & 1 & 12 \end{pmatrix}, \begin{pmatrix} 1 & 1+10^{5} & 1 \\ 1 & 1+10^{-5} & 2 \\ 0 & 10^{5} & -1 \end{pmatrix} \begin{pmatrix} 1 & 1+10^{15} & 1 \\ 1 & 1+10^{-15} & 2 \\ 0 & 10^{15} & -1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},$$

4) **S**ur les exemples précédents, comparer le résultat obtenu avec ceux fournis d'une part par la fonction inv de la bibliothèque numpy.linalg et d'autre part par la méthode inv des objets de type Matrix dans le module sympy

5) La matrice de Virginie d'ordre n est $V_n = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$ Inverser V_2 et V_3 .

Comparer les temps nécessaires pour les inversions de V_n avec n dans {10, 20, 40, 80} pour votre procédure et celles des bibliothèques numpy et sympy.