Université Polytechnique HAUTS-DE-FRANCE

INSA

Année 2023 - 2024 SHPI 2

Méthodes numériques appliquées

TP 3

Pour cette séance, vous repartirez de votre carnet `Jupyter` du TP2

Les premières questions doivent être traitées dans l'ordre, jusqu'à 3 incluse. Ensuite, vous pourrez choisir.

Le thème est toujours l'algorithme du pivot de Gauss, pour la résolution des systèmes linéaires de la forme $A\,X = B$. Pour mémoire, son principe, rappelé au TP 2, est ci-dessous :

Voici un rappel du principe de l'algorithme de Gauss : en posant $A|B = (a_{i,j})_{i=1\cdots n, j=1\cdots n+1}$ et en notant L_i la i-ème ligne de $A|B$, le nombre k ci-dessous étant l'indice de colonne,
Pour $k = 1, \cdots, n-1$, faire

- trouver le plus petit $i \geqslant k$ tel que $a_{i,k} \neq 0$ (on cherche un pivot non nul)
- échanger L_i et L_k
- pour $l = k+1, \cdots, n$, faire {remplacer L_l par $L_l - \dfrac{a_{l,k}}{a_{k,k}} L_k$ } (on fait apparaître des 0 dans la k-ème colonne, sous le pivot)

On obtient ainsi une matrice triangulaire supérieure. On résout enfin le système « en remontant ».

1. Terminez la mise au point de votre algorithme commencé au TP 2, toujours dans le cas où A est une matrice carrée inversible. Il s'agit donc de créer une fonction `Python` prenant en paramètres la matrice A et le vecteur B, et retournant la solution X du système $A\,X = B$.
   Il faut toujours satisfaire les conditions suivantes :
   - respecter les pratiques de programmation conseillées, et rappelées dans `Moodle` ;
   - pour une utilisation ultérieure (question 5), les matrices passées en paramètre de votre programme ne doivent pas être modifiées à sa sortie.
2. Que donne votre programme pour
$$A = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \varepsilon = 10^{-20} \quad ?$$
   Que devrait-il donner ? S'il y a un problème, que proposez-vous pour y remédier ? Attention, à ce stade une réponse théorique suffit (donc dans une cellule de texte de `Jupyter`), le programme n'est pas demandé. Expliquez seulement la méthode proposée.
3. Même question pour
$$A = \begin{pmatrix} \varepsilon & 1 \\ \varepsilon & \varepsilon \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 1 \\ 2\varepsilon \end{pmatrix}$$
4. Déterminez *expérimentalement* le coût de votre algorithme (celui du 1) en fonction de la taille n (nombre de lignes ou de colonnes) de la matrice A : le coût est-il proportionnel à n ? à n^2 ? etc. Vous pouvez utiliser la commande `timeit`, déjà vue en TP, ou bien importer la bibliothèque `time` et utiliser sa fonction `perf_counter` : cette fonction (sans paramètre) retourne un « temps machine » non significatif en soi, mais en l'appelant avant et après l'exécution d'une commande, la différence des temps donne le temps d'exécution de la commande. Pour une valeur fixée de n, il s'agit donc de déterminer le temps *moyen* d'exécution de votre algorithme sur une matrice de taille n ; cela suppose de créer d'abord des matrices aléatoires (à cœfficients flottants) en nombre suffisant pour que la moyenne soit significative.
5. Comparez votre algorithme à la fonction `solve` de la bibliothèque `scipy.linalg` : en termes de coût (expérimental, toujours) mais aussi en termes d'efficacité pour les cas numériquement difficiles comme ceux des questions 2 et 3.

6. Programmez vos algorithmes du 2 et du 3.
7. Généralisez votre programme au cas d'un système non carré et d'une matrice non nécessairement inversible. Attention, il est indispensable ici de reprendre la méthode descendante... au début, car il faudra créer de nouvelles fonctions. Une liste de ces fonctions, indiquant ce que fait chacune, serait déjà une bonne chose.