L3 - Méthodes numériques

TP 3 - Algèbre linéaire

Support de TP

Les TPs visent à appliquer de manière concrète les notions vues en cours. Il est donc conseillé de venir avec vos supports de cours ou de les télécharger sur le web.

Un programme C type ainsi que le fichier Makefile permettant de le compiler est disponible sur le site du cours. Pour chacune des parties du TP, vous devez créer un programme "tp3_i.c" où i est le numéro de la section.

Les TPs notés doivent être soumis sur la page Jalon du cours dans les deux semaines suivant le TP. Vous devez soumettre un fichier zip contenant le code **compilable** avec un simple appel à `make` ainsi qu'un court rapport de 2 pages maximum en **PDF** (sans page de titre). Le rapport ne doit contenir aucune sortie terminal ou ligne de code. Vous devez par contre pour chaque section du TP rédiger une réflexion sur les notions utilisées et ce que vous avez appris.

1 Opérations de base

Cette section du TP vise à implémenter en C certaines multiplications matricielles vues dans l'exercice 1 (page 12) du cours d'algèbre linéaire.

Une bibliothèque d'algèbre linéaire est donnée dans le fichier `linalgutils.h`. Vous y trouverez toutes les fonctions vues en cours.

1. Ouvrir le fichier `tp3_0.c`, compiler et interpréter la sortie terminal. Enregistrer le fichier sous `tp3_1.c` que vous compléterez dans la suite.
2. Coder et afficher le résultat des opérations $\mathbf{u}^\top\mathbf{u}$ et $\mathbf{u}^\top\mathbf{v}$ (fonction `vdot`).
3. Calculer le vecteur $2\mathbf{u} + \mathbf{v}$ et l'afficher (fonction `vaxpy`).
4. Coder les multiplications matricielles $\mathbf{Au}$ et $\mathbf{Av}$ et afficher les résultats (fonction `mgemv`).
5. Coder les multiplications $\mathbf{Au}$ et $\mathbf{Av}$ en utilisant BLAS (fonction cblas `cblas_dgemv`).
6. Comparer les deux implémentations et mesurer la différence entre les deux solutions (fonction `err`).
7. Coder la multiplication $\mathbf{uu}^\top$ et afficher le résultat (fonction `mger`).

2 Résolution de système linéaire

Dans cette section nous allons chercher à résoudre un système linéaire de la forme $\mathbf{Ax} = \mathbf{b}$. Pour celà nous allons coder un pivot de Gauss et résoudre le système triangulaire puis comparer à l'implémentation de référence LAPACK.

1. Initialiser et afficher une matrice $\mathbf{A}_0$ et un vecteur $\mathbf{b}_0$ avec :

$$\mathbf{A}_0 = \begin{bmatrix} 2 & 1 & -1 \\ -3 & -1 & 2 \\ -2 & 1 & 2 \end{bmatrix}, \quad \mathbf{b}_0 = \begin{bmatrix} 8 \\ -11 \\ -3 \end{bmatrix}$$

2. Copier $\mathbf{A}_0$ et $\mathbf{b}_0$ dans une matrice $\mathbf{A}$ et un vecteur $\mathbf{b}$ et appliquer le pivot de Gauss vu en cours (fonction `pivot`). Imprimer la matrice triangulaire résultante $\mathbf{A}$ et le vecteur $\mathbf{b}$.
3. Comparer le résultat au pivot illustré page 45 du cours.

4. Maintenant que la matrice $\mathbf{A}$ est triangulaire supérieure, il faut résoudre le système. Pour ce faire, compléter la fonction `trisupsv` du fichier `linalgutils.h`. Vous pouvez vous inspirer de `triinfsv` qui résout un système pour une matrice triangulaire inférieure.
5. Afficher la solution $\mathbf{x}$ obtenue ainsi que $\mathbf{A}_0\mathbf{x}$. Comparer à $\mathbf{b}_0$.
6. Copier $\mathbf{A}_0$ et $\mathbf{b}_0$ dans une matrice $\mathbf{A}1$ et un vecteur $\mathbf{b}1$ et résoudre le système linéaire à l'aide de LAPACK (fonction `LAPACKE_dgesv`).
7. Comparer les deux solutions (`pivot`+`trisupsv` VS LAPACK).

3 Précision et temps de calcul

Dans cette section nous effectuerons une comparaison systématique des solutions de systèmes linéaire pour différentes tailles de problème.

1. Prendre $n = 10$.
2. Initialiser $\mathbf{A}_0, \mathbf{A}, \mathbf{A}_1$ de taille $n \times n$ et $\mathbf{b}_0, \mathbf{b}, \mathbf{b}_1$ de taille n à la valeur 1 pour toutes les composantes.
3. Initialiser $\mathbf{A}_0$ avec des valeurs aléatoires (fonction `mrand`) et copier les valeurs dans $\mathbf{A}$ et $\mathbf{A}1$.
4. Résoudre le système avec `pivot` et `trisupsv`. Afficher le temps de calcul (fonctions `tic` et `toc`).
5. Calculer l'erreur entre $\mathbf{b}_0$ et $\mathbf{A}_0\mathbf{x}$ et l'afficher (fonctions `mgemv` et `err`).
6. Résoudre le système en utilisant LAPACK (fonction `LAPACKE_dgesv`). Afficher le temps de calcul et l'erreur numérique (fonctions `mgemv` et `err`).
7. Changer la valeur de n pour 100, 500 et 1000. Conclusions ?

4 Evolution de la précision et du temps de calcul (bonus)

Le but est de tracer le temps de calcul et l'erreur de la résolution en fonction de la taille du système avec Gnuplot.

1. Pour une liste de valeurs de `n` allant de 50 à 500 par pas de 50. Effectuer les étapes de la section précédente dans une boucle.
2. Penser à libérer le mémoire à la fin de chaque itération pour les matrices et vecteurs initialisés dans la boucle (fonctions `free` et `mfree`).
3. Tracer l'évolution du temps de calcul et de l'erreur numérique pour chacune des méthodes avec Gnuplot.