

Compte Rendu TP Méthode Numérique n°2

Bibliothèque BLAS

**Mottino Loris | Da Costa Tom | Verrier Martin**

Introduction :

Dans ce TP nous allons réaliser notre propre bibliothèque d’algèbre linéaire. Nous allons implémenter toutes les fonctions et structures de données nécessaires à cela, et ensuite tester notre bibliothèque.

Nous allons donc vous expliquer brièvement comment avons-nous implémenter notre bibliothèque avant de nous attarder sur la partie test de notre code.

Résultats de nos tests :

Tous nos tests sont structurés de la même manière :

* On inclut les prototypes des fonctions avec mnblas.h.
* On inclut le type complexe avec complexe.h.
* On inclut une bibliothèque de notre création avec pleins d'utilitaires pratiques et récurrent dans le test, comme :
  + Print de vecteur
  + Print de matrice
  + Générer des nombre aléatoire complexe / réel
  + Calcul flop
  + etc.
* Ensuite, on effectue des tests de "bon résultat", c’est-à-dire, voir si les fonctions implémentées sortent le résultat attendu. Pour chacun des 4 types (float, double, complexe\_float, complexe\_double), on effectue un calcul avec un/des vecteur/matrice initialisé(s) avec une valeur simple (1 ou 2) et on print le résultat. On regarde ensuite, visuellement si le résultat est bon. On refait alors la meme chose mais avec un/des vecteur/matrice avec des nombres aléatoires. Ensuite il faut que l’utilisateur vérifie le résultat.
* Ensuite, on effectue les tests pour connaitre les ratios de flop/s (ou les o/s). Les vecteurs ou matrices de la partie "bon résultat" sont trop petit, c’est pourquoi on libère les vecteurs précèdent et on les réalloue avec une taille (VECSIZE\_FLOPS (ou VECSIZE\_OS) beaucoup plus grande que VECSIZE\_RESULAT (petite taille pour pouvoir être lisible et afficher).

Pour chaque type, on commence par calculer les flop ou os de l'exécution d'une seule fonction et ça NB\_EXPE\_VISIBLE fois. Puis on calcul les flops ou os pour NB\_EXPE exécution de la fonction étudiée.

* Puis on libère le tout.

On va présenter ici le résultat du test de la fonction axpy, concernant les flops :

|  |  |
| --- | --- |
|  | float |
|  | ------------------------------------------------ |
|  | mnblas\_saxpy : 200000 operations 0.152 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_saxpy : 200000 operations 0.271 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_saxpy : 200000 operations 0.253 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_saxpy : 200000 operations 0.277 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_saxpy : 200000 operations 0.257 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_saxpy : 200000 operations 0.298 GFLOP/s |
|  | <---------------------------------------------------------------> |
|  | float sur NB\_EXPE (1000) |
|  | mnblas\_saxpy : 200000000 operations 0.608 GFLOP/s |
|  | <---------------------------------------------------------------> |
|  | double |
|  | ------------------------------------------------ |
|  | mnblas\_daxpy : 200000 operations 0.220 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_daxpy : 200000 operations 0.154 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_daxpy : 200000 operations 0.219 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_daxpy : 200000 operations 0.747 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_daxpy : 200000 operations 0.775 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_daxpy : 200000 operations 0.782 GFLOP/s |
|  | <---------------------------------------------------------------> |
|  | double sur NB\_EXPE (1000) |
|  | mnblas\_daxpy : 200000000 operations 0.623 GFLOP/s |
|  | <---------------------------------------------------------------> |
|  | complexe\_float\_t |
|  | ------------------------------------------------ |
|  | mnblas\_caxpy : 800000 operations 0.264 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_caxpy : 800000 operations 0.330 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_caxpy : 800000 operations 0.358 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_caxpy : 800000 operations 0.369 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_caxpy : 800000 operations 0.375 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_caxpy : 800000 operations 0.378 GFLOP/s |
|  | <---------------------------------------------------------------> |
|  | complexe\_float\_t sur NB\_EXPE (1000) |
|  | mnblas\_caxpy : 800000000 operations 0.348 GFLOP/s |
|  | <---------------------------------------------------------------> |
|  | complexe\_double\_t |
|  | ------------------------------------------------ |
|  | mnblas\_zaxpy : 800000 operations 0.323 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_zaxpy : 800000 operations 0.552 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_zaxpy : 800000 operations 0.593 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_zaxpy : 800000 operations 0.598 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_zaxpy : 800000 operations 0.513 GFLOP/s |
|  | ------------------------------------------------ |
|  | mnblas\_zaxpy : 800000 operations 0.384 GFLOP/s |
|  | <---------------------------------------------------------------> |
|  | complexe\_double\_t sur NB\_EXPE (1000) |
|  | mnblas\_zaxpy : 800000000 operations 0.472 GFLOP/s |

On observe que pour float et double la fonction effectue 200 000 opérations contre 800000 pour complexe\_float\_t et complexe\_double\_, soit 4 fois plus. Par ailleurs, on remarque que de manière générale le ratio de GFLOP / s, augmente au cours des tests, cela peut s’expliquer par la puissance allouée par le processeur à cette tache qui doit augmenter au cours des essaies. On remarque que ce ratio n’est pas constant mais que sur un grand nombre de tests (ici 1000) le ratio pour float et double est d’environ 0.6 GFLOP / s contre 0.4 pour complexe\_flaot\_t et complexe\_double\_t, soit une vitesse moindre pour un plus grand nombre d’opérations effectue par appel à la fonction.

Dans notre projet vous pourrez retrouver des fichiers tests pour chaque fonction de la bibliothèque comprenant les codes test et les sorties affichées.

Conclusion :

Lors de ce TP nous avons appris à manipuler des bibliothèques et à créer notre propre bibliothèque. Nous avons aussi confectionné des codes de tests et appris à les analyser afin de s’assurer du bon fonctionnement de notre code, mais également de mieux comprendre le traitement des opérations par l’ordinateur.