TD-TP2 Méthodes Numériques
Polytech Grenoble, département INFO

Jean-François Méhaut

17 février 2020

L'objectif de ce TP2 est de développer une bibliothèque d'algèbre linéaire. Vous reprendrez une partie des fonctions de la bibliothèque BLAS (Basic Linear Algebra Subprograms). Cette bibliothèque est struturée en 3 ensembles de fonction.

— Les fonctions **BLAS1** qui effectuent des opérations vecteur-vecteur.
— Les fonctions **BLAS2** qui effectuent des opérations vecteur-matrice.
— Les fonctions **BLAS3** qui effectuent des opérations matrice-matrice.

Le TP2 de MN est à rendre avant le Dimanche 15/3/2020. Vous deposerez sur le moodle une archive contenant le code source de votre bivliothèque et un rapport décrivant le travail que vous avez effectué. Pour ce second TP de MN, vous donnerez des éléments d'analyse de performance de certaines fonctions de votre bibliothèque d'algèbre linéaire.

Les fonctions de votre bibliothèqye se feront avec 4 types de données :
— flottant, simple précision (s)
— double, double précision (d)
— complexe simple précision (c)
— complexe double précision (z)

Pour les tests et évaluations de performance, vous créerez un programme de test par type de fonction. Ce programme de test doit tester les fonctions avec les 4 types de données (float, double, complexe simple et complexe double.

1 Préparation du TP

Vous allez récupérer l'archive `TPBLAS.tar.gz` sur le moodle.

Vous allez ensuite exécuter les commandes suivantes `tar` et `make` pour compiler le code source qui vous est fourni..

```
$ tar xvfz TPBLAS.tar.gz
TPBLAS/
TPBLAS/doc/
TPBLAS/doc/sujet.txt
TPBLAS/src/
TPBLAS/src/copy.c
TPBLAS/src/swap.c
```

```
TPBLAS/src/Makefile
TPBLAS/src/complexe.c
TPBLAS/src/dot.c
TPBLAS/include/
TPBLAS/include/complexe.h
TPBLAS/include/complexe2.h
TPBLAS/include/mnblas.h
TPBLAS/lib/
TPBLAS/examples/
TPBLAS/examples/flop.c
TPBLAS/examples/flop.h
TPBLAS/examples/Makefile
TPBLAS/examples/test_dot.c
TPBLAS/examples/test_complexe.c
TPBLAS/examples/test_complexe2.c
$ cd TPBLAS/src
gcc -O2 -Wall -fPIC -O2 -fopenmp -I../include -c copy.c
gcc  -O2 -Wall -fPIC -O2 -fopenmp -I../include -c swap.c
gcc  -O2 -Wall -fPIC -O2 -fopenmp -I../include -c dot.c
gcc -O2 -Wall -fPIC -O2 -fopenmp -I../include -c complexe.c
rm -f libmnblas.a ../lib/libmnblas.a
ar -r libmnblas.a copy.o swap.o dot.o complexe.o
ar: creation de libmnblas.a
cp libmnblas.a ../lib
rm -f libmnblasdyn.so ../lib/libmnblasdyn.so
gcc -shared -o libmnblasdyn.so copy.o swap.o dot.o complexe.o
cp libmnblasdyn.so ../lib
$ cd ../examples
$ make
gcc -O2 -Wall -fPIC -O2 -fopenmp -I../include -c copy.c
gcc  -O2 -Wall -fPIC -O2 -fopenmp -I../include -c swap.c
gcc  -O2 -Wall -fPIC -O2 -fopenmp -I../include -c dot.c
gcc -O2 -Wall -fPIC -O2 -fopenmp -I../include -c complexe.c
rm -f libmnblas.a ../lib/libmnblas.a
ar -r libmnblas.a copy.o swap.o dot.o complexe.o
ar: creation de libmnblas.a
cp libmnblas.a ../lib
rm -f libmnblasdyn.so ../lib/libmnblasdyn.so
gcc -shared -o libmnblasdyn.so copy.o swap.o dot.o complexe.o
cp libmnblasdyn.so ../lib
$ cd ../examples
$ make
gcc -Wall -O2 -fPIC -I../include -c test_complexe2.c
gcc -Wall -O2 -fPIC -I../include -c flop.c
gcc -o test_complexe2 test_complexe2.o flop.o -fopenmp -L../lib -lmnblas
gcc -Wall -O2 -fPIC -I../include -c test_complexe.c
gcc -o test_complexe test_complexe.o flop.o -fopenmp -L../lib -lmnblas
gcc -Wall -O2 -fPIC -I../include -c test_dot.c
gcc -o test_dot test_dot.o flop.o -fopenmp -L../lib -lmnblas
gcc -o test_dot_dyn flop.o test_dot.o -fopenmp -L../lib -lmnblasdyn
```

```
$ test_dot
mncblas_sdot 0 : res = 131072.00 nombre de cycles: 2597044
sdot  131072 operations 0.131 GFLOP/s
mncblas_sdot 1 : res = 131072.00 nombre de cycles: 40118
sdot  131072 operations 8.509 GFLOP/s
mncblas_sdot 2 : res = 131072.00 nombre de cycles: 38444
sdot  131072 operations 8.880 GFLOP/s
mncblas_sdot 3 : res = 131072.00 nombre de cycles: 37872
sdot  131072 operations 9.015 GFLOP/s
mncblas_sdot 4 : res = 131072.00 nombre de cycles: 38108
sdot  131072 operations 8.959 GFLOP/s
$ test_dot_dyn
test_dot_dyn: error while loading shared libraries: libmnblasdyn.so:
cannot open shared object file: No such file or directory
$ export LD_LIBRARY_PATH=../lib
$ ldd test_dot_dyn
        linux-vdso.so.1 (0x00007fff7e3ce000)
        libmnblasdyn.so => ../lib/libmnblasdyn.so (0x00007f2d09031000)
        libgomp.so.1 =>
        /lib/x86_64-linux-gnu/libgomp.so.1 (0x00007f2d08fe3000)
        libc.so.6 =>
        /lib/x86_64-linux-gnu/libc.so.6 (0x00007f2d08df2000)
        libdl.so.2 =>
        /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f2d08dec000)
        libpthread.so.0 =>
        /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f2d08dc9000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f2d090bd000)

$ test_dot_dyn
mncblas_sdot 0 : res = 131072.00 nombre de cycles: 3936236
sdot  131072 operations 0.087 GFLOP/s
mncblas_sdot 1 : res = 131072.00 nombre de cycles: 42162
sdot  131072 operations 8.088 GFLOP/s
mncblas_sdot 2 : res = 131072.00 nombre de cycles: 40784
sdot  131072 operations 8.361 GFLOP/s
mncblas_sdot 3 : res = 131072.00 nombre de cycles: 40598
sdot  131072 operations 8.400 GFLOP/s
mncblas_sdot 4 : res = 131072.00 nombre de cycles: 40472
sdot  131072 operations 8.426 GFLOP/s
$
```

Cette archive contient donc les répertoires suivants :

- Le répertoire `src` contient les fichiers source de vos fonctions BLAS. Un fichier `Makefile` est fourni avec les premières fonctions BLAS1. Vous compléterez ce fichier avec les nouvelles fonctions BLAS1, BLAS2 et BLAS3.
- Le répertoire `include` contient le fichier `mnblas.h`. Les premières fonctions BLAS 1 sont déclarées, les autres sont commentées. Vous devrez les décommenter au fur et à mesure de l'avancement de votre travail.
- Le répertoire `lib` contient les librairies statique (`libmnblas.a`) et dynamique (`libmnblasdyn.so`).

Ces deux biliothèques sont générées après compilation et édition de liens des fichiers du répertoire `src`.

— Le répertoire `examples` contiendra des exemples de programmes utilisant votre bibliothèque BLAS. Il est conseillé d'y placer plusieurs programmes d'examples (un par fonction ou par type de fonction). Les versions des exécutables en statique et dynamique sont également générées. Ce répertoire contient un fichier `flop.c` qui permet d'afficher le nombre de flop par seconde d'une fonction.

La documentation de l'ensemble des fonctions BLAS se trouve sur le site Web suivant : `https://software.intel.com/en-us/mkl-developer-reference-c-blas-routines`

Les fonctions de votre bibliothèques effectueront des calculs sur des vecteurs et des matrices où les éléments peuvent être de 4 types :

— `float`, simple précision, **s**
— `double`, double précision, **d**
— complexe simple précision, **sc**
— complexe double précision, **dz**

2 Calculs sur des nombres complexes

Deux types complexe sont définis dans le fichier `include/complexe.h`. Il est défini avec une structure contenant la partie réelle et la partie imaginaire du nombre complexe.

```
typedef struct {
  float real ;
  float imaginary ;
} complexe_float_t ;

typedef struct {
  double real ;
  double imaginary ;
} complexe_double_t ;
...
...
```

Des fonctions d'addition, multiplication et division sur les nombres complexes doivent être réalisées.

1. Ecrivez les fonctions de calcul sur les nombres complexes qui sont dans le fichier `complexe.c`.
2. Complétez le fichier `test_complexe.c` pour tester et évaluer les performances des fonctions opérant sur les complexes. Le programme `test_complexe.c` utilise la fonction `calcul_flop` du fichier `flop.c`. Vous modfierez ce fichier avec la fréquence nominale de votre processeur. Un second fichier include pour les objects complexe est fourni (`complexe2.h`). Les déclarations des types complexes sont identiques à celles qui se trouvent dans `complexe.h`. La principale différence se situe au niveau des fonctions sont *inlinées*. L'extension inline est une optimisation du compilateur qui remplace un appel de fonction par le code de cette fonction. Cette optimisation vise à réduire le temps d'exécution ainsi que la consommation mémoire.

Toute fois, cette extension peut conduire à une augmentation de la taille du programme exécutable.

3. Complétez le fichier `complexe2.h` avec le code associé à l'ensemble des fonctions opérant sur les complexes.
4. Vous allez lancer l'exécution du programme `test_complexe2`. Vérifiez que les résultats sont bien identiques à ceux obtenus avec `test_complexe`.
5. Comparez les résultats obtenus avec les programmes `test_complexe` et `test_complexe2`. Une anomalie de mesure se produit sur la mesure des opérations du programme `test_complexe2`. Pouvez l'expliquer et la corriger ? Vous pouvez demander de l'aide à votre enseignant.

3 Fonctions BLAS1

Plusieurs fonctions BLAS1 doivent être implémentées. Vous trouverez sur le site Intel [1] une descrption de ces fonctions.

Les fonctions **swap** et **copy** sont un peu particulières car elles n'effectuent pas de calcul. La fonction **copy** copie le vecteur x dans le vecteur y. Il s'agit donc d'une opération de copie de données qui sont en mémoire. La fonction **swap** permute le contenu des vecteurs x et y. Vous calculerez la performance de ces fonctions en utiisant comme métrique des mégas octets par seconde.

Pour les autres fonctions BLAS1, vous calculerez la performance avec comme métrique le nombre d'opérations flottantes par seconde. Ce calcul est effectué par la fonction `flop` du fichier `flop.c`.

11. Implémentez les fonctions de **swap** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.
12. Implémentez les fonctions de **copy** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.
13. Implémentez les fonctions de **dot** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.
14. Implémentez les fonctions de **axpy** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.
15. Implémentez les fonctions de **asum** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.
16. Implémentez les fonctions de **iamin** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.
17. Implémentez les fonctions de **iamax** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.
18. Implémentez les fonctions de **nrm2** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.

4 Fonction BLAS2

19. Implémentez les fonctions de **gemv** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.

1. `https://software.intel.com/en-us/mkl-developer-reference-c-blas-routines`

5 Fonction BLAS3

20. Implémentez les fonctions de **gemm** opérant sur les 4 types de données. Écrivez un programme de test et d'évaluation de ces fonctions.