III Comment compiler un programme MATLAB

1- Introduction

Le compilateur MATLAB génère un code source C à partir d'une fonction MATLAB. Le code C généré par le compilateur MATLAB peut être:

- soit un code source C pour construire des fonctions MATLAB compilées ('MEX-files').
- soit un code source C pour combiner avec d'autres modules pour former une application externe qui puisse fonctionner sans MATLAB.

2- Pourquoi compiler des fonctions MATLAB ?

Il y a au moins trois bonnes raisons pour compiler une fonction MATLAB:

⇨ Pour augmenter la vitesse
⇨ Pour cacher le code source pour diffuser vos travaux et éviter que votre code soit modifié.
⇨ Pour créer une application indépendante de MATLAB.

Un code C compilé tourne plus vite que son équivalent en MATLAB car :

- Un programme compilé tourne plus vite qu'un programme interprété.
- Un code C contient des données de types plus simple. Pour MATLAB toutes les données sont des matrices (des tableaux).

MATLAB vérifie la taille des tableaux à chaque affectation d'élément ce que l'on peut eviter de faire en C.

MATLAB doit ré-allouer de la mémoire en cours d'exécution la ou en C on peut l'eviter.

La compilation d'une fonction MATLAB n'apportera pas forcement de gain si :

- la fonction MATLAB est fortement vectorisée.
- la fonction MATLAB passe beaucoup de temps à utiliser des fonctions internes MATLAB mathématiques (très rapide) et graphiques (très lent).

La compilation apportera des gains de temps si la fonction MATLAB :

- contient des boucles (for,while).
- contient des variables que le compilateur traduit en 'real' ou en 'integer'. ne travaille que sur des réelles.

3 - Création d'une fonction MATLAB compilée 'MEX-files

a) Mécanisme de base

Le compilateur MATLAB 'mcc' transforme une fonction MATLAB en code source C qui est ensuite compilé par l'outil 'cmex'. Celui-ci génère un code compilé (avec comme extension '.mexrs6') de la fonction qui sera automatiquement utilisé à la place de la fonction MATLAB correspondante.

Soit la fonction MATLAB suivante **'carre1'** qui va nous servir d'exemple:

---

function mat=carre1(n); % calcul test for i=1:n mat=sqrt(i)*i;

Pour compiler on tape sous MATLAB :

```
>> mcc carre1.m
```

On obtient un code source C 'carre1.c' et un exécutable 'carre1.mexrs6' qui sera utilisé à la place de 'carre1.m'. Ces deux fichiers se trouve dans le répertoire courant. L'utilisation de l'outil 'cmex' est transparent pour l'utilisateur.

Pour lancez la fonction on tape par exemple :

```
>> carre1(100)
```

Si l'on desire des informations sur la compilation l'option **'-v'** permet d'avoir toutes les informations en cours de la compilation.

```
>> mcc -v carre1.m
```

---

b) Optimisation

Pour optimiser une fonction MATLAB on peut supprimer :

la vérification du dimensionnement des tableaux: option **'-i'**. (pas de vérifcation en cas de debordement de tableau)

spécifier qu'il n'y a pas de variables du type complexe dans votre fonction, si c'est le cas bien sur ! option **'-r'**

Essayons ces options :

```
>> mcc -ri carre1
```

La compilation réussie mais cela ne veux pas forcement dire que le programme marche. Ici si on lance carre1, on obtient un message d'erreur vraiment méchant. Car en fait on a un redimensionnement de la matrice 'mat' à chaque itération ce qui est incompatible avec l'option '-i'.

Le pré-dimensionnement de tableau permet d'éviter ce problème. Avant tout il faut savoir qu'il permet d'obtenir des gains de performances assez considérables que ce soit pour une fonction compilée que pour une fonction interprétée.

En tenant compte de ces conseils voici '**carre2.m**', une version amélioré de **carre1.m**

function mat=carre2(n); % déclaration d'une matrice d'élément nul de taille 1,n mat=zeros(1,n); for i=1:n mat=sqrt(i)*i; end

Le tableau suivant montre les gains de performances obtenus suivant les options de compilations. Les calculs du temps d'exécution sont fait à l'aide des fonctions 'tic' et 'toc' pour un nombre d'itérations n=10000.

Nom du programme	Forme	Temps écoulées	Facteur de gain
carre1	interprété	3.800 secondes	1
mcc carre1	compilé	1.700 s	2.24
mcc -i carre1	compilé	erreur	-
mcc -r carre1	compilé	0.500 s	7.60
mcc -ri carre1	compilé	erreur	-
carre2	interprété	0.660 s	1
mcc carre2	compilé	0.590 s	1.12
mcc -i carre2	compilé	0.590 s	1.12
mcc -r carre2	compilé	0.012 s	55.00
mcc -ri carre2	compilé	0.0085 s	77.64

Si on compare le temps d'exécution de la fonction 'carre1' interprété sous MATLAB et la version optimisé et compilé 'carre2', celle-ci est 447 fois plus rapide que 'carre1'.

function mat=carre2(n); % déclaration d'une matrice d'élément nul de taille 1,n mat=zeros(1,n); for i=1:n mat=sqrt(i)*i; end

Le tableau suivant montre les gains de performances obtenus suivant les options de compilations. Les calculs du temps d'exécution sont fait à l'aide des fonctions 'tic' et 'toc' pour un nombre d'itérations n=10000.

Nom du programme	Forme	Temps écoulées	Facteur de gain
carre1	interprété	3.800 secondes	1
mcc carre1	compilé	1.700 s	2.24
mcc -i carre1	compilé	erreur	-
mcc -r carre1	compilé	0.500 s	7.60
mcc -ri carre1	compilé	erreur	-
carre2	interprété	0.660 s	1
mcc carre2	compilé	0.590 s	1.12
mcc -i carre2	compilé	0.590 s	1.12
mcc -r carre2	compilé	0.012 s	55.00
mcc -ri carre2	compilé	0.0085 s	77.64

Si on compare le temps d'éxecution de la fonction 'carre1' inteprété sous MATLAB et la version optimisé et compilé 'carre2', celle-ci est 447 fois plus rapide que 'carre1'.

---

b) Ecriture d'un programme principale en MATLAB

Pour éviter d'écrire un programme principale en C, on peut écrire la fonction **'main'** et la compiler. En C 'main' est le nom de la fonction principale. Par exemple :

---

function h=main % juste un calcul, n'affiche rien carre2(10000);

On compile les deux fonctions:

>> mcc -e main.m

>> mcc -e carre2.m lancement de la commande UNIX mccbuild depuis MATLAB >> !mccbuild -o main main.c carre2.m

On peut maintenant exécuter le programme 'main' sur n'importe quel IBM RS/6000 du Centre.

---

c) Ecriture d'une seule fonction MATLAB

Si on a une fonction qui n'a pas de paramètre en entrée, on peut la compiler sans avoir à créer un programme principale. Pour cela, il faut créer un programme 'main.c'

On procède ainsi, on copie sous UNIX la fonction 'fonction.m' dans 'main.m':

>> !cp fonction.m main.m On genère le code C 'main.c' qui comporte la fonction 'main'.

>> mcc -e main.m

Puis on génère l'exécutable sous UNIX

$ mccbuild -o monprog main.c

On peut maintenant utiliser le programme exécutable 'monprog'

---

d) Génération de fonctions objets

Le génération de fonctions objets '.o' est intéressante à deux titre :

Si on developpe une application assez importante, il est possible de ne compiler que les objets modifiés d'ou un gain de temps à la compilation.

On peut ne fournir à un autre utilisateur que les fonctions objets ce qui assure une bonne confidentialité (c'est du binaire)

Voici un exemple pour obtenir des fonctions objets:
Génération des programmes C sous MATLAB

>> mcc -e main.m

>> mcc -e fonction.m

Génération des programmes objet

>> cmex -c fonction.c

>> cmex -c main.c

Génération de l'exécutable sous UNIX

$ mccbuild -o prog main.o fonction.o

4. Limitations et Restrictions

Il y a quelques restrictions qui font que le compilateur MATLAB ne peut pas transformer tous les codes MATLAB en code C. Le compilateur MATLAB ne peut pas compiler :

un script MATLAB, il faut que ce soit une fonction MATLAB, souvent il suffit de rajouter 'function f=nomfonction' en première ligne.

un code contenant 'eval','input' ou 'load'.

un code contenant la variable 'ans'. un code qui utilise les matrices creuses. un code contenant des variables dont le nom comporte un '_'.

Il y a quelques différences d'interprétations du code entre le compilateur MATLAB et l'interpréteur.

Le compilateur MATLAB ne permet pas les opérations de comparaisons entre nombres complexes.

L'affectation d'un variable nul dans une autre variable ne sera pas correctement traité en C (exemple: a=[];x(i)=a)

Les arguments des fonctions 'zeros','ones','eyes' et 'rand' sont transformés en entier par le compilateur.

Restrictions supplémentaires concernant les applications externes
Les applications externes ne peuvent accéder :

aux fonctions d'E/S de MATLAB intéractives comme 'input'.

aux fonctions de debug MATLAB comme 'dbclear'. aux fonctions graphiques comme 'surf','plot','get' et 'set'. à la fonction 'save'.

aux fonctions SIMULINK

5 - Documentation

Pour plus de renseignements tapez sous MATLAB **'help mcc'** et**'help cmex'**.
Consultez la documentation MATLAB qui se nomme **"MATLAB Compiler User's Guide"** et **"MATLAB C Math library User's Guide"** disponible au Centre de Calcul.