TP Calcul par éléments finis et parallélisation

1 Préparation

1.1 Récupération des fichiers

Les fichiers sont à télécharger à l'adresse suivante :

Extrayez le fichier dans un répertoire dédié au TP.

1.2 Le maillage et les partitions

Le maillage de base dont on se servira dans tout le TP est écrit dans le fichier ***meshprogc.dat***a (fichier généré lors du dernier TP à partir de ***mesh.data*** par le programme ***data2tec***). Vous pouvez utiliser le partitionnement calculé par METIS (***dualformetis.dat.epart.X***) aussi bien que celui calculé par SCOTCH (***dualforscotch.map***).

Génération des fichiers de maillage partitionné

Les partitions calculées lors du TP précédent vont être utilisées pour répartir le maillage original dans plusieurs sous-maillages (1 fichier par sous-maillage). Pour ce faire, nous allons utiliser le programme ***Preprocess*** qui génère ces sous-fichiers à partir de la concaténation du maillage original et du partitionnement. Pour générer ce fichier ***mesh_for_progc.data*** :

— A partir du partitionnement METIS :

```
cat meshprogc.dat dualformetis.dat.epart.4 > mesh_for_progc.data
```

— A partir du partitionnement SCOTCH :

```
gcc -o fromscotch.exe fromscotch.c
./fromscotch.exe meshprogc.dat dualforscotch.map mesh_for_progc.data
```

Vous pouvez maintenant utiliser le programme ***Preprocess*** qui, à partir de ce fichier ***mesh_for_progc.data*** créé N fichiers ***Data00.In*** ... ***DataN.In*** (où dans l'exemple ci-dessous $N = 4$) :

```
gcc -o Preprocess.exe Preprocess.c
./Preprocess.exe 4 mesh_for_progc.data
```

2 Le code de calcul par éléments finis

2.1 Compilation et exécution

Le programme est compilé par la commande :

```
(CREMI) mpicc.openmpi -lm -o fem.exe FemPar.c
(ENSEIRB-MATMECA) mpicc -lm -o fem.exe FemPar.c
```

et exécuté sur 4 processeurs par la commande :

```
(CREMI) mpirun.openmpi -np 4 ./fem.exe
(ENSEIRB-MATMECA) mpirun -np 4 ./fem.exe
```

Ouvrir et analyser le fichier FemPar.c ; s'assurer de bien comprendre les principales structures de données du code :

Shared
Common
Neighbours
Node

2.2 Implémentation des fonctions parallèles

La communication entre process se fait dans le solveur, ici le Gradient Conjugué, et est restreinte aux sous-programmes : ***InnerProduct et Update.***

2.2.1 Produit vecteur-vecteur

La fonction ***InnerProduct(A1, A2, B1, B2)*** calcule le produit scalaire entre deux vecteurs : $A \cdot B = A1 \cdot B1 + A2 \cdot B2$. Ici nous avons décomposé le vecteur global A (resp. B) en deux parties :

— $A1$ pour l'ensemble des points à l'intérieur de la partition (points non partagés),

— $A2$ pour l'ensemble des points à la frontière avec une autre partition.

Réalisez l'implémentation de cette fonction pour qu'elle donne le bon résultat en parallèle. Attention de ne pas compter plusieurs fois les contributions des noeuds frontières entre les partitions.

2.2.2 Contribution des éléments aux noeuds

Les noeuds du maillage reçoivent des contributions des éléments les entourant. Par conséquent, les noeuds à la frontière entre deux processeurs doivent collecter les contributions d'éléments de plusieurs partitions. La fonction ***Update(A)*** effectue ce calcul.

Réalisez l'implémentation de cette fonction pour qu'elle donne le bon résultat en parallèle.

Pour ce faire, vous devez étudier les structures ***Common***, ***Neighbours*** et ***Node***.

Vérifiez que votre solution est robuste (qu'elle fonctionne dans tous les cas, avec une quantité de données importante, face à un « ralentissement » des échanges, etc.)

3 Validation : courbes de Speed-up et Efficacité du code EF parallèle

Comparez les courbes de Speed-up du code pour les partitions ***METIS*** et ***SCOTCH***.

3.1 Faire en sorte que le code fonctionne sur un process

Créez le fichier de maillage (input du programme FemPar.c) permettant l'usage d'un seul process. Equiper le code de manière à connaître le temps d'exécution. Le temps d'exécution du code sur 1 process constituera la charge totale.

3.2 Exécutez le code en faisant varier le nombre de process (partitions SCOTCH et METIS)

Pour ce faire, vous devez organiser correctement votre travail :

En créant un répertoire ***metis*** dans lequels vous aurez des sous-répertoires partition1, partition2, partition3, etc... Faites la même chose pour ***scotch***.