

Modèles et techniques en programmation parallèle hybride et multi-cœurs

Projet 2 (2ème version)

Marc Tajchman

CEA - DEN/DM2S/STMF/LDEI

15/1/2026

Projet 2

On part de deux codes qui calculent une solution approchée du problème suivant :

Chercher $u: (x, t) \mapsto u(x, t)$, où $x \in \Omega = [0, 1]^3$ et $t \geq 0$, qui vérifie :

$$\frac{\partial u}{\partial t} = \Delta u + f(x, t)$$

$$u(x, 0) = g(x) \quad x \in \Omega$$

$$u(x, t) = g(x) \quad x \in \partial\Omega, t > 0$$

où f et g sont des fonctions données.

On utilise des différences finies pour approcher les dérivées partielles et on découpe Ω en $n_0 \times n_1 \times n_2$ subdivisions.

Codes de départ

Récupérer et décompresser le fichier

[Projet2_incomplet.tar.gz](#).

Cette archive contient 3 sous-répertoires :

- ▶ la version séquentielle (PoissonSeq),
- ▶ une version incomplète, parallélisée avec Cuda (PoissonCuda),
- ▶ une version incomplète, parallélisée avec Kokkos (PoissonKokkos), (le 3ème sous-répertoire, contenant la version Kokkos, sera fourni la semaine du 26 au 30 janvier 2026)

La version séquentielle sert de référence.

Le but du TP est de compléter les versions Cuda et Kokkos.

Dans la version Cuda, la partie du code à modifier se trouve dans les fichiers:

```
PoissonCuda/src/cuda/iteration.cu  
PoissonCuda/src/cuda/user.cu
```

Pour `iteration.cu`, on prendra les lignes 69 à 79 de `scheme.cxx` comme modèle.

Pour `user.cu`, on prendra comme modèle le fichier `user.cxx`

Un exemple de code écrit en cuda pour initialiser le tableau de valeurs :

- ▶ la fonction `init` entre les lignes 68 à 78 et
 - ▶ le noyau cuda `initValue` entre les lignes 51 à 66
- du fichier `PoissonCuda/src/cuda/values.cu`

On fournit une archive compressée

[Projet2_incomplet_2.tar.gz](#) qui contient le 3ème répertoire avec la version Kokkos du code.

Il faut la compléter en modifiant les 4 fichiers suivants dans le sous-répertoire src: [main.cxx](#), [scheme.cxx](#), [user.hxx](#) et [values.cxx](#) (des commentaires indiquent les parties à modifier ou à compléter).

On trouvera aussi un sous-répertoire kokkos qui contient le code source d'une version récente de Kokkos (5.0.1).

Enfin un script python [build.py](#) compile Kokkos et le code source du projet en trois versions (séquentielle, OpenMP et Cuda).

Il faut utiliser soit la machine rhum à l'Ensta, soit votre machine personnelle, à condition

- ▶ que cette machine contienne une carte graphique compatible (ç-à-d. une carte Nvidia), et
- ▶ que le toolkit Cuda soit installé
(<https://developer.nvidia.com/cuda-downloads>)

Pour compiler et exécuter

- ▶ dans le répertoire PoissonSeq, les 2 commandes sont :

```
python build.py  
./install/release/PoissonSeq
```

- ▶ dans le répertoire PoissonCuda, les 2 commandes sont :

```
python build.py  
./install/release/PoissonCuda
```

- ▶ dans le répertoire PoissonKokkos, les 2 commandes sont :

```
python build.py  
./install/release/PoissonKokkos
```

Envoyez votre code source, ce qu'affiche la commande `nvidia-smi`, et les copies de ce qui est affiché à l'écran à l'exécution des codes (dans une archive compressée) par mail à marc.tajchman@cea.fr **avant le 8/2/2026**.

Recommandation:

*Ne pas envoyer les fichiers créés par la compilation
(répertoires build et install)*