TP MPI N°2
Différences finies appliquées à l'équation de la chaleur bidimensionnelle

1. Equation de la chaleur bidimensionnelle et la résolution numérique

Considérons l'équation de la chaleur suivante :

$$\begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) - c\dfrac{\partial^2 u}{\partial x^2}(t,x,y) - c\dfrac{\partial^2 u}{\partial y^2}(t,x,y) = 0, \quad \text{pour } t \in]0,+\infty[\text{ et } x,y \in]0,1[\\ u(0,x,y) = u_0(x,y), \quad \text{condition initiale} \\ u(t,0,0) = u(t,1,0) = u(t,0,1) = u(t,1,1) = 0, \quad \text{conditions limites} \end{cases}$$

u(t,x,y) représente la température du point *(x,y)* au temps *t*. u_0 est la répartition initiale de température (exemple : la Figure 1b) et la température du bord reste à *0* au cours du temps. *c* est une constante.

On cherche les valeurs approchées de *u* sur les points du maillage 2D régulier du domaine *[0,1]* x *[0,1]*. Les points du maillage sont en (x_i, y_j) avec $x_i = ih_x$ et $y_j = jh_y$, *i,j=0, 1, …, n+1* (voir la Figure 1a). La méthode de différences finies avec le schéma numérique explicite (schéma d'Euler explicite) suivant sera utilisée pour calculer l'évolution de la température dans le temps.

$$\begin{cases} \dfrac{u_{ij}^{k+1} - u_{ij}^{k}}{h_t} - c\dfrac{u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}}{(h_x)^2} - c\dfrac{u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k}}{(h_y)^2} = 0, \quad \text{i}, j \in [1,n] \\ u_{0,j}^{k} = u_{n,j}^{k} = u_{i,0}^{k} = u_{i,n}^{k} = 0 \quad \text{pour i}, j \in [1,n] \end{cases}$$

i.e. $u_{ij}^{k+1} = u_{ij}^{k} + c_x\left(u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}\right) + c_y\left(u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k}\right)$

Stabilité du schéma : le schéma d'Euler explicite est stable sous condition β<1/2, où $\beta = \max\{c_x, c_y\} = \max\{ch_t / h_x^2, ch_t / h_y^2\}$.

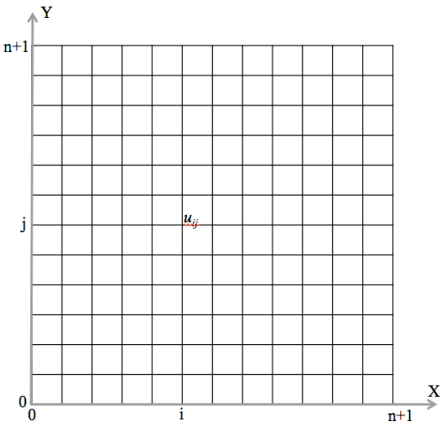
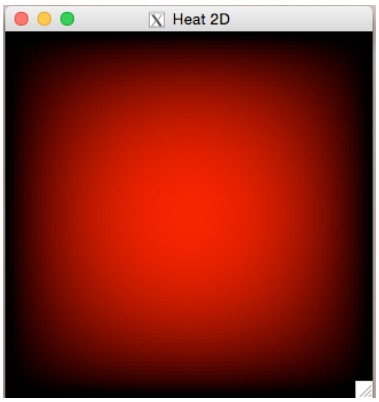


Figure 1 a. Maillage du domaine de calcul



b. u_0 fourni par `initData()`, *n=78*

2. Programme séquentiel

Le programme séquentiel fourni contient une fonction d'initialisation de u_0 (`initData()`), une fonction `updateNLigs()` qui calcule l'évolution de la température sur les lignes spécifiées et les fonctions de sauvegarde de résultats dans un fichier (`saveData()` crée un nouveau fichier et `appendData()` ajoute en fin d'un fichier existant). On peut utiliser ces fonctions pour la sauvegarde des résultats, mais également pour leur vérification et leur comparaison du programme parallèle au programme séquentiel. La taille du maillage est égale à 80 par défaut. Elle peut être modifiée en exécution (avec 2 arguments sur la ligne de commande d'exécution).

Les paramètres impliqués dans le schéma d'Euler explicite doivent respecter la condition $\beta<1/2$, qui garantisse la stabilité de celui-ci. En programmation, les coefficients des quatre voisins directs sont implantés dans un masque de taille `3 x 3`. Un exemple est donné par la Figure 2, où $c_x = c_y = 0.1$, ce qui donne `MASK[]={0.0, 0.1, 0.0, 0.1, 0.6, 0.1, 0.0, 0.1, 0.0}`.

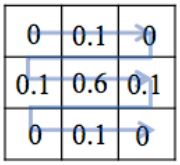


Figure 2. Masque des coefficients pour l'évolution de température

Deux versions du programme séquentiel sont fournies : l'une sans affichage graphique (`heat2D.c`) et l'autre avec (`heat2DXlib.zip`). La version graphique permet une vérification visuelle de l'évolution de température, mais sa parallélisation est plus délicate, donc plutôt réservée à ceux qui maitrise bien la programmation.

3. Parallélisation - Décomposition en blocs de lignes – Schéma Maître-Esclaves

La décomposion du domaine se fait par blocs de lignes. Nous allons utiliser le schéma de maître-esclaves pour la parallélisation. Le maître initialise u_0, distribue sans duplication les données initiales aux esclaves et récolte les résultats à chaque étape et les sauvegarde dans un fichier ou les affiche dans une fenêtre graphique. Les esclaves réceptionnent les données initiales, communiquent entre eux les données du bord et font évoluer la température de leurs sous-domaines puis remontent leurs résultats à chaque étape du calcul.

La Figure 3 montre la répartition du calcul avec 3 processus. La température évolue uniquement sur les points intérieurs du maillage. Celle du bord reste à 0. Le Proc 0 est le maître, il ne participe pas au calcul.

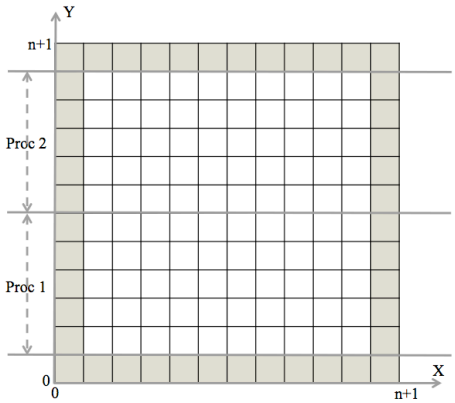


Figure 3. Répartition du calcul avec 3 processus

La communication entre les processus sera en premier temps réalisée à l'aide de la communication point-à-point.

En deuxième temps, les communications collectives seront utilisées pour la distribution de u_0 et la collecte des résultats. **Attention :** pour faciliter l'utilisation de la communication collective, le maître participe aussi au calcul de l'évolution de température.

Expérimenter la mesure de performance de votre programme parallèle selon la même procédure vue en TP d'OpenMP, avec n suffisamment grand. Les fonctions de mesure du temps sont similaires :

- *double MPI_Wtime(void) ;* donne le temps en micro/nano secondes à partir d'un point donné. *double MPI_Wtick(void) ;* donne la résolution de l'horloge.
- Une synchronisation de tous les processus (à l'aide de *MPI_Barrier*) est nécessaire avant le 1er mesure du temps, afin que les processus démarrent leur mesure en même temps et que les temps d'exécution de tous les processus soient comparables.