Enseirb-Matmeca

— Projet de Calcul parallèle —

Résolution de l'équation de la chaleur par une méthode de décomposition de domaine

François-Régis Hammer Alix Kien Maëlle Praud

Encadré par

Héloïse Beaugendre

2021-2022

Table des matières

1 Objectifs du projet

L'objectif principal de ce projet est de résoudre l'équation de conduction instationnaire grâce à un code de calcul parallèle utilisant la méthode de décomposition de domaine.

Le système (1) est à résoudre dans le domaine $\Omega = [0, L_x] \times [0, L_y]$ de $\mathbb{R}^2$ illustré sur la FIGURE 1.

$$\begin{cases} \partial_t u(x,y,t) - D\Delta u(x,y,t) = f(x,y,t) \quad , (x,y) \in]0, L_x[\times]0, L_y[, t \geq 0 \\ u = g \quad \text{sur} \quad \Gamma_0 \\ u = h \quad \text{sur} \quad \Gamma_1 \end{cases} \tag{1}$$

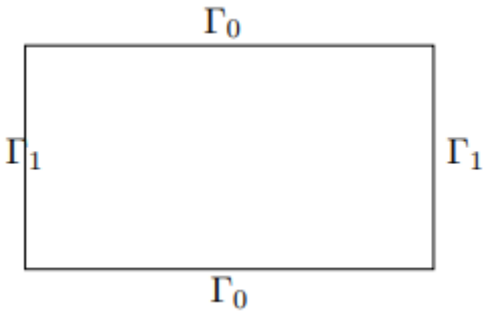

FIGURE 1 – Domaine de résolution

Les objectifs du projet sont multiples. En premier lieu, il s'agira de bien comprendre l'importance de l'équilibre de charge lors de l'élaboration d'un code de calcul parallèle. Dans un deuxième temps, la méthode de décomposition de domaine utilisé dans ce projet sera étudié. Il s'agit de la méthode dite de SCHWARZ additif sur un maillage cartésien régulier. Une attention particulière sera portée concernant la gestion du recouvrement mis en oeuvre dans la méthode de décomposition de domaine. De plus, la gestion des communications nécessaires au cours de l'algorithme de SCHWARZ fera l'objet d'une partie spécifique. Finalement les performances obtenues seront analysées en fonction du recouvrement mais aussi du nombre de sous-domaines ou encore des conditions de bords choisies, DIRICHLET ou ROBIN.

2 Élaboration d'un code de calcul parallèle

2.1 Équilibre de charge et maillages

Un maillage structuré est un maillage tel que le nombre de voisins de chaque sommet est constant. Donc pour un sommet donné il est assez simple de rechercher le bon nombre de voisins lors de la lecture du maillage. La lecture du maillage est primordiale pour la construction de schémas éléments finis et volumes finis donc pour le calcul scientifique en général.

Pour les maillages non structurés, l'intérêt est d'une part de permettre une meilleure représentation de la géométrie et d'autre part de faire varier la taille des mailles dans le domaine. Ces maillages sont utiles pour les géométries complexes mais leur lecture est plus difficile.

Dans le cadre du parallélisme informatique, l'un des enjeux est de répartir équitablement la charge de calcul du programme séquentiel sur chacun des processus. Dans le cas d'une résolution sur un maillage la problématique revient à répartir les mailles entre les processus. Cette répartition se fait soit avec une répartition totale soit de façon mixte. Pour la première, chaque maille est allouée à un unique processus et pour la seconde, les processus peuvent avoir des mailles en commun, on parle alors de recouvrement.

Pour les maillages structurés, le type d'élément est souvent le même partout. Il est alors assez aisé de répartir un nombre de maille équitable entre les processus. Pour le cas des maillages non structuré,

ils sont souvent plus complexes. Par exemple le type d'élément n'est pas forcément identique et cela peut rendre la répartition de la charge entre les processus plus difficile. De plus, le caractère spécifique de chaque noeud ou maille rend le calcul des charges beaucoup plus lourd.

Les logiciels METIS et SCOTCH permettent d'effectuer automatiquement un partitionnement à partir d'un maillage donné.

Ces deux logiciels sont basés sur la représentation d'un maillage au travers d'un graphe. Un graphe consiste à relier des noeuds par des arcs. Il y a deux types de graphe possibles : le graphe du maillage, qui prend pour noeuds les noeuds du maillage, et le graphe dual, qui prend les centres des mailles pour noeud. Le logiciel répartit ensuite les sous-domaines à partir de ces graphes.

METIS est basé sur la méthode du graphe de maillage et SCOTCH sur le graphe dual. Donc à partir d'un maillage identique et pour un même nombre de partitions, les deux logiciels n'effectuent pas le même partitionnement, comme cela est montré en FIGURE 2.

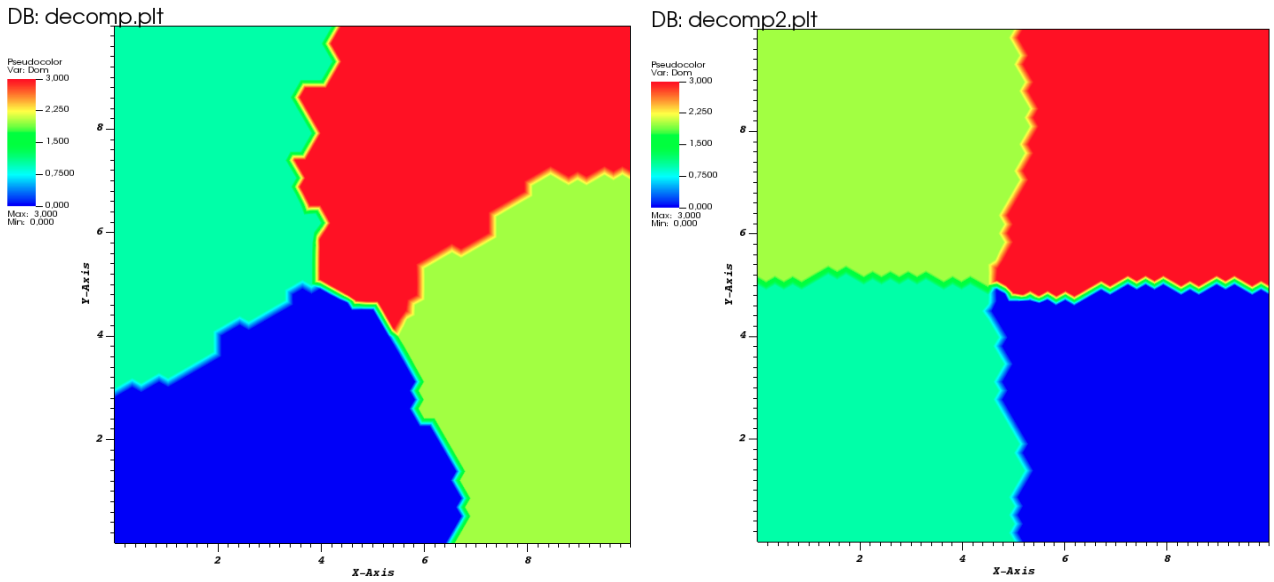

FIGURE 2 – Décomposition de domaine en 4 sous-domaines effectuée avec les logiciels METIS (à gauche) et SCOTCH (à droite).

2.2 Calcul du speed-up et de l'efficacité

Afin de déterminer les performances d'un code parallèle deux critères sont à prendre en compte : le speed-up et l'efficacité. Ces valeurs dépendent entre autres du maximum de la charge d'un processus, par une relation inversement proportionnelle. Ainsi, il est important d'avoir une charge totale la mieux répartie pour obtenir les meilleures performances.

Speed-up. Le speed-up est le rapport entre la charge totale d'un programme séquentiel et la charge du processus qui hérite du plus grand nombre de tâches.

$$S_p = \frac{\text{charge totale}}{max_{me}[charge(me)]} = N_p * \text{Eff} \tag{2}$$

C'est une valeur (positive) inférieure au nombre de processus N_p.

Efficacité. L'efficacité est le rapport entre le speed-up et le nombre de processus N_p :

$$\text{Eff} = \frac{\text{charge totale}}{N_p \times max_{me}[charge(me)]} \tag{3}$$

Il s'agit du taux d'efficacité moyen par processus. Il est donc compris entre 0 et 1.

2.3 Communications avec MPI

Dans le cadre de notre projet, un code séquentiel est parallélisé par échange de message, avec l'environnement MPI. C'est un schéma à mémoire distribuée et qui est très adapté pour les méthodes de décomposition de domaine. Chaque processus possède une partie du domaine global et effectue des échanges avec ces voisins.

Une problématique essentielle réside dans la part des communications par rapport aux calculs : si elle devient trop importante, les performances peuvent être dégradées au point de perdre l'intérêt du parallélisme. Ainsi, il faut minimiser les appels aux sous-programmes de communication dans les régions parallèles. Par ailleurs, la recopie d'un message dans un espace mémoire temporaire, ou buffering, est à éviter si possible. Enfin, il est aussi possible de recouvrir les communication par des calculs si le programme s'y prête bien, en utilisant des communications non-bloquantes.

Ainsi, en MPI, l'envoi d'un message peut se faire selon trois modes. Le mode `standard` laisse le choix à MPI d'effectuer ou non une recopie temporaire du message. En cas de recopie, l'envoi se termine lorsque la copie est effectuée, sinon, il se termine quand le message a été réceptionné.

Le mode `synchrone` permet de coupler l'envoi et la réception d'un message : l'envoi ne peut commencer que lorsque le processus récepteur est disponible pour communiquer. Les principaux avantages de ce mode sont de garantir la réception du message ainsi que son coût en ressources faible puisqu'il n'y a pas de recopie du message dans un buffer. L'inconvénient de cette méthode peut être le temps d'attente dans le cas où le processus récepteur n'est pas disponible.

Le mode `bufferisé` nécessite de programmer manuellement une recopie temporaire du message et l'envoi se termine à la recopie du message dans le buffer. L'avantage de cette méthode est qu'elle ne nécessite pas d'attendre le récepteur. Les principaux inconvénients de ce mode sont en premier lieu son coût en mémoire en raison des recopies dans les buffers, ainsi que la gestion manuelle des buffers d'envoi qui peut donc être mal réalisée.

Une communication peut être bloquante ou non-bloquante. Un appel est bloquant si l'espace mémoire servant à la communication peut être réutilisé immédiatement après la sortie de l'appel. Un appel non bloquant rend la main très rapidement, mais n'autorise pas la réutilisation immédiate de l'espace mémoire utilisé dans la communication. Il est alors nécessaire de s'assurer que la communication est bien terminée avant de l'utiliser à nouveau.

Dans le projet, le choix a été fait d'utiliser un mode de communication `synchrone` pour éviter la recopie dans un buffer mais aussi s'assurer que la réception a bien eu lieu. Par ailleurs, des communications bloquantes ont été choisies car il n'y avait pas de raisons particulières d'avoir recours à des communication non-bloquantes.

3 Décomposition de domaine – Schwarz additif

3.1 Discrétisation du problème

3.1.1 Schéma différences finies

Le but est de résoudre numériquement l'équation (1) par la méthode des différences finies en utilisant un schéma implicite en temps et un schéma centré en espace. Posons maintenant les notation suivantes :

$$
\begin{aligned}
&x_i = i\Delta x \quad \forall i = \{0, ..., N_x + 1\} \quad \text{avec } \Delta x = \frac{L_x}{N_x + 1},\\
&y_j = j\Delta y \quad \forall j = \{0, ..., N_y + 1\} \quad \text{avec } \Delta y = \frac{L_y}{N_y + 1},\\
&t_n = n\Delta t \quad n \geqslant 0 \quad \text{avec } \Delta t \text{ le pas de temps.}
\end{aligned}
$$

De cette manière, nous avons les approximations suivantes :

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j, t_n) \approx \frac{u(x_{i+1}, y_j, t_n) - 2u(x_i, y_j, t_n) + u(x_{i-1}, y_j, t_n)}{\Delta x^2}$$
$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j, t_n) \approx \frac{u(x_i, y_{j+1}, t_n) - 2u(x_i, y_j, t_n) + u(x_i, y_{j-1}, t_n)}{\Delta y^2}$$
$$\frac{\partial u}{\partial t}(x_i, y_j, t_n) \approx \frac{u(x_i, y_j, t_{n+1}) - u(x_i, y_j, t_n)}{\Delta t}.$$

On pose comme approximation :

$$u_{i,j}^n \simeq u(x_i, y_j, t_n).$$

Le schéma s'écrit sous forme factorisée $\forall i \in [\![1, N_x]\!], \forall j \in [\![1, N_y]\!]$:

$$\left[1 + \frac{2D\Delta t}{\Delta x^2} + \frac{2D\Delta t}{\Delta y^2}\right] u_{i,j}^{n+1} - \frac{D\Delta t}{\Delta x^2}\left(u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1}\right) - \frac{D\Delta t}{\Delta y^2}\left(u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}\right) = u_{i,j}^n + \Delta t f_{i,j}^{n+1} \quad (4)$$

A partir de cette factorisation l'écriture sous forme matricielle du problème est aisée.

3.1.2 Mise sous forme matricielle

La résolution du problème se fait par méthode implicite donc le problème revient à résoudre un système linéaire de la forme $AU_n^\star = F_n^\star$, où les vecteurs $U_n^\star$ et $F_n^\star$ sont tels que

$$U_n^\star = \begin{pmatrix} U_{1,n} \\ \vdots \\ U_{j,n} \\ \vdots \\ U_{Ny,n} \end{pmatrix} \quad \text{et} \quad F_n^\star = \Delta t \begin{pmatrix} F_{1,n} \\ \vdots \\ F_{j,n} \\ \vdots \\ F_{Ny,n} \end{pmatrix} + U_{n-1}^\star + \begin{pmatrix} H_n(y_1) + G_n(0) \\ H_n(y_2) \\ \vdots \\ H_n(y_{N_y-1}) \\ H_n(y_{N_y}) + G_n(L_y) \end{pmatrix}$$

Pour simplifier les écritures, les notations $u_{i,j}^n = u(x_i, y_j, t_n)$ et $f_{i,j}^n = f(x_i, y_j, t_n)$ sont posées. Les vecteurs $U_{j,n}$, $F_{j,n}$, $H_{j,n}$ et $G_n(y)$ sont de taille N_x et définis par

$$U_{j,n} = \begin{pmatrix} u_{1,j}^n \\ \vdots \\ u_{i,j}^n \\ \vdots \\ u_{Nx,j}^n \end{pmatrix} \quad F_{j,n} = \begin{pmatrix} f_{1,j}^n \\ \vdots \\ f_{i,j}^n \\ \vdots \\ f_{Nx,j}^n \end{pmatrix} \quad H_n(y) = \frac{D\Delta t}{\Delta x^2} \begin{pmatrix} h(0, y, t_n) \\ 0 \\ \vdots \\ 0 \\ h(L_x, y, t_n) \end{pmatrix} \quad G_n(y) = \frac{D\Delta t}{\Delta y^2} \begin{pmatrix} g(x_1, y, t_n) \\ \vdots \\ g(x_i, y, t_n) \\ \vdots \\ g(x_{N_x}, y, t_n) \end{pmatrix}$$

La matrice A, de taille $N_x \times N_y$, est de la forme

$$A = \begin{bmatrix} T & B & & & \\ B & T & B & & \\ & \ddots & \ddots & \ddots & \\ & & B & T & B \\ & & & B & T \end{bmatrix} \quad (5)$$

avec les matrices B et T, de taille $N_x \times N_x$, définies par

$$B = -\frac{D\Delta t}{\Delta y^2} I_{N_x} \quad \text{et} \quad T = \begin{bmatrix} \gamma & -\frac{D\Delta t}{\Delta x^2} & & & \\ -\frac{D\Delta t}{\Delta x^2} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -\frac{D\Delta t}{\Delta x^2} \\ & & & -\frac{D\Delta t}{\Delta x^2} & \gamma \end{bmatrix} \quad (6)$$

où I_n est la matrice identité de rang n et $\gamma = 1 + 2D\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)$.

Pour des raisons de lisibilité pour la suite, les notations $\delta_x = -\frac{D\Delta t}{\Delta x^2}$ et $\delta_y = -\frac{D\Delta t}{\Delta y^2}$ sont adoptées.

3.2 Principe de la décomposition de domaine

La décomposition de domaine est une méthode efficace pour résoudre les systèmes linéaires de taille importante. Le principe de cette méthode est de partitionner le domaine de calcul en sous-domaines. Cette décomposition peut se faire manuellement ou à l'aide d'un logiciel tel que METIS ou SCOTCH. La solution du problème physique n'est alors plus calculée sur le domaine en entier mais sur chaque sous-domaine. Ainsi, les systèmes linéaires à résoudre sont de tailles plus petites sur les sous-domaines. La réunion des solutions sur les sous-domaines permet de retrouver la solution globale recherchée.

La principale difficulté réside dans le lien à construire entre les sous-domaines. Il existe des méthodes de décomposition de domaine avec et sans recouvrement. Les méthodes avec recouvrement sont connues sous le nom de méthodes de Schwarz.

Dans la suite nous nous attacherons à élaborer une méthode de décomposition de domaine avec recouvrement pour résoudre le problème initial. Deux approches sont alors possibles. Soit tous les domaines sont résolus simultanément et les conditions de bords sont prises à l'itération n précédente. C'est la méthode dite additif. Soit les régions paires sont résolues en utilisant les conditions de bords à l'itération n précédente puis les régions impaires sont résolues en utilisant les conditions de bords de l'itération $n+1$ qui vient d'être calculée. C'est la méthode dite multiplicatif.

Nous nous concentrerons sur la méthode de Schwarz additif car c'est celle qui est la plus adaptée à une architecture de code parallèle.

3.3 Méthode de Schwarz additif

Pour simplifier la présentation de la méthode, un partitionnement en deux sous-domaines est considéré. Le domaine initial $\Omega = [0, L_x] \times [0, L_y]$ est séparé en deux sous-domaines $\Omega_1 = [0, L_{x1}] \times [0, L_y]$ et $\Omega_2 = [L_{x2}, L_x] \times [0, L_y]$. Ces deux sous-domaines possèdent h points en communs dans le maillage global : ce sont les points de recouvrement. Concrètement, dans notre cas, il y a un domaine à gauche (le domaine 1) et un domaine à droite (le domaine 2).

En premier lieu, les solutions $(U_0^\star)^{(1)}$ et $(U_0^\star)^{(2)}$ sont fixées, sur les sous-domaines 1 et 2 respectivement, de manière arbitraire, mais tels que ces vecteurs sont non nuls. Une itération de Schwarz est ensuite est ensuite effectuée, qui permet de calculer les solutions sur les sous-domaines. Puis, il faut vérifier que les points des sous-domaines 1 et 2 au niveau du recouvrement sont confondus. Si ce n'est pas le cas, l'itération est répétée en utilisant les solutions qui viennent d'être calculées, et cela jusqu'à convergence entre les solutions $(U_n^\star)^{(1)}$ et $(U_n^\star)^{(2)}$ au niveau des points du recouvrement.

3.4 Itération de Schwarz

Pour appliquer la méthode de décomposition de domaines avec recouvrement, il est nécessaire de réécrire le schéma numérique sur chaque sous domaine. Etant un schéma à 3 points, il nécessite une condition de bord à l'interface entre les sous-domaines qui permettent d'exprimer la solution en un point fantôme $N+1$ pour le domaine 1, et 0 pour le domaine 2. Pour réaliser cette jonction entre les sous-domaines en utilisant des points réels, la condition $\alpha \dfrac{\partial u}{\partial n} + \beta u$ est imposée à l'interface. Il s'agit de la condition de Robin. La condition de Dirichlet correspond au cas où $\alpha = 0$ et $\beta = 1$.

Posons les notations suivantes :

— N_k le nombre de points de discrétisation sur le domaine $k = 1, 2$, selon $O\vec{x}$;

— $x_i^{(k)}$ et $y_j^{(k)}$ les coordonnées des points de la discrétisation sur le domaine $k = 1, 2$;

— $u_{i,j}^{k,n}$ la solution u dans le domaine $k = 1, 2$, à l'itération n de la boucle de Schwarz, au point $(x_i^{(k)}, y_j^{(k)})$.

L'invariant suivant est vérifié :

$$N_x + h(n_d - 1) - \sum_{k=1}^{n_d} N_k = 0 \tag{7}$$

avec n_d le nombre de sous-domaines.
Les systèmes qui dérivent du système (1) initial sur les sous-domaines 1, à gauche, et 2, à droite, sont les systèmes (8) et (9), respectivement.

$$\begin{cases} \partial_t u_{i,j}^{1,n} - D\Delta u_{i,j}^{1,n} = f_{i,j}^n \quad , \quad i \in [\![1, N_1]\!], j \in [\![1, N_y]\!] \\ u_{i,0}^{1,n} = g(x_i^{(1)}, 0, t_n) \quad , \quad u_{i,0}^{1,n} = g(x_i^{(1)}, L_y, t_n) \\ u_{0,j}^{1,n} = h(0, y_j^{(1)}, t_n) \\ \alpha \dfrac{\partial u_{N_1,j}^{1,n}}{\partial n_1} + \beta u_{N_1,j}^{1,n} = \alpha \dfrac{\partial u_{h,j}^{2,n-1}}{\partial n_1} + \beta u_{h,j}^{2,n-1} \end{cases} \tag{8}$$

$$\begin{cases} \partial_t u_{i,j}^{2,n} - D\Delta u_{i,j}^{2,n} = f_{i,j}^n \quad , \quad i \in [\![1, N_2]\!], j \in [\![1, N_y]\!] \\ u_{i,0}^{2,n} = g(x_i^{(2)}, 0, t_n) \quad , \quad u_{i,N_y}^{1,n} = g(x_i^{(2)}, L_y, t_n) \\ u_{N_2,j}^{2,n} = h(L_x, y_j^{(2)}, t_n) \\ \alpha \dfrac{\partial u_{1,j}^{2,n}}{\partial n_2} + \beta u_{1,j}^{2,n} = \alpha \dfrac{\partial u_{N_1-h,j}^{1,n-1}}{\partial n_2} + \beta u_{N_1-h,j}^{1,n-1} \end{cases} \tag{9}$$

Les matrices $A^{(1)}$ et $A^{(2)}$ associées à ces systèmes linéaires sont de taille $N_1 \times N_y$ et $N_2 \times N_y$, respectivement, et elles conservent une forme tridiagonale par blocs. Le bloc diagonal $T^{(1)}$ (resp. $T^{(2)}$) et les bloc extra-diagonaux $B^{(1)}$ (resp. $B^{(2)}$) sont de taille $N_1 \times N_k$ (resp. $N_1 \times N_k$) et sont respectivement tridiagonal et diagonal.

3.4.1 Avec conditions de Dirichlet

La condition de Dirichlet correspond au cas où $\alpha = 0$ et $\beta = 1$.
Ainsi, les conditions de bord avec l'autre domaine deviennent $u_{N_1,j}^{1,n} = u_{h,j}^{2,n-1}$ pour le système (8) et $u_{1,j}^{2,n} = u_{N_1-h,j}^{1,n-1}$ pour le système (9), $(\forall j \in [\![1, N_y]\!])$.
Le système linéaire présenté en section 3.1.2 est changé ce qui se traduit par N_y lignes changées dans la matrice A et dans le second membre associé :

$$\begin{cases} (F_n^\star)_{N_1,j}^{(1)} = \dfrac{1}{\Delta x^2} u_{h,j}^{2,n-1} \\ (T^{(1)})_{N_1,j} = \dfrac{1}{\Delta x^2} \\ (B^{(1)})_{N_1,j} = 0 \end{cases} \quad \text{et} \quad \begin{cases} (F_n^\star)_{1,j}^{(2)} = \dfrac{1}{\Delta x^2} u_{N_1-h,j}^{1,n-1} \\ (T^{(2)})_{1,j} = \dfrac{1}{\Delta x^2} \\ (B^{(2)})_{1,j} = 0 \end{cases} \tag{10}$$

pour le sous-domaine 1 et le sous-domaine 2 respectivement, avec $j \in [\![1, N_y]\!]$

3.4.2 Avec conditions de Robin

Pour les conditions de Robin, il faut reprendre les schémas (8) et (9). Il faut faire attention à l'orientation de la normale : $n_1 = -n_2$. Par ailleurs, l'approximation du terme $\frac{\partial}{\partial n}\cdot$ fait apparaître une colonne fantôme d'indice $N_1 + 1$ pour le domaine 1 et d'indice 0 pour le domaine 2.

Domaine de gauche. La condition de Robin pour le schéma (8) s'écrit :

$$\begin{aligned} &\alpha\frac{u^{1,n}_{N_1+1,j}-u^{1,n}_{N_1,j}}{2\Delta x}+\beta u^{1,n}_{N_1-1,j}=\alpha\frac{u^{2,n-1}_{h+1,j}-u^{2,n-1}_{h-1,j}}{2\Delta x}+\beta u^{2,n-1}_{h,j}\\ \Rightarrow\quad &u^{1,n}_{N_1+1,j}=u^{1,n}_{N_1,j}+\frac{2\Delta x\beta}{\alpha}\left(u^{2,n-1}_{h,j}-u^{1,n}_{N_1-1,j}\right)+\left(u^{2,n-1}_{h+1,j}-u^{2,n-1}_{h-1,j}\right) \end{aligned}$$

$$\begin{aligned} \Rightarrow\quad &(\gamma+\eta)u^{1,n}_{N_1,j}+2\delta_x u^{1,n}_{i-1,j}+\delta_y\left(u^{1,n}_{N_1,j+1}+u^{1,n}_{N_1,j-1}\right)=\\ &u^{1,n-1}_{N_1,j}+\Delta t f^n_{N_1,j}-\delta_x\left(u^{2,n-1}_{h+1,j}-u^{2,n-1}_{h-1,j}\right)+\eta u^{2,n-1}_{h,j} \end{aligned} \tag{11}$$

avec $\eta=\dfrac{2D\Delta t}{\Delta x}\dfrac{\beta}{\alpha}$. La condition de Robin sur le domaine à gauche modifie donc la dernière ligne de la matrice T ainsi que les N_y lignes du second membre associées :

$$T^{(1)}=\begin{bmatrix} \gamma & \delta_x & & & \\ \delta_x & \gamma & \delta_x & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \delta_x \\ & & & 2\delta_x & \gamma+\eta \end{bmatrix} \quad \text{et} \quad B^{(1)}=\delta_y I_{N_1} \tag{12}$$

$$\text{et}\quad (F^\star_n)^{(1)}=F^\star_n-\delta_x\begin{pmatrix} u^{2,n-1}_{h+1,1}-u^{2,n-1}_{h-1,1}+\frac{2\Delta x\beta}{\alpha}u^{2,n-1}_{h,1}\\ 0\\ \vdots\\ 0\\ u^{2,n-1}_{h+1,j}-u^{2,n-1}_{h-1,j}+\frac{2\Delta x\beta}{\alpha}u^{2,n-1}_{h,j}\\ 0\\ \vdots\\ 0\\ u^{2,n-1}_{h+1,N_y}-u^{2,n-1}_{h-1,N_y}+\frac{2\Delta x\beta}{\alpha}u^{2,n-1}_{h,N_y}\\ 0\\ \vdots\\ 0 \end{pmatrix} \tag{13}$$

Domaine de droite. La condition de Robin pour le schéma (9) s'écrit :

$$\begin{aligned} &\alpha\frac{u^{2,n}_{0,j}-u^{2,n}_{2,j}}{2\Delta x}+\beta u^{2,n}_{1,j}=\alpha\frac{u^{1,n-1}_{N_1-h-1,j}-u^{1,n-1}_{N_1-h+1,j}}{2\Delta x}+\beta u^{1,n-1}_{N_1-h,j}\\ \Rightarrow\quad &u^{2,n}_{0,j}=u^{2,n}_{2,j}+\frac{2\Delta x\beta}{\alpha}\left(u^{1,n-1}_{N_1-h,j}-u^{2,n}_{1,j}\right)+\left(u^{1,n-1}_{N_1-h-1,j}-u^{1,n-1}_{N_1-h+1,j}\right) \end{aligned}$$

Donc l'équation du système correspondante se réécrit :

$$\begin{aligned} \Rightarrow\quad &(\gamma+\eta)u^{2,n}_{1,j}+2\delta_x u^{2,n}_{1,j}+\delta_y\left(u^{2,n}_{1,j+1}+u^{2,n}_{1,j-1}\right)=\\ &u^{1,n-1}_{2,j}+\Delta t f^n_{N_1-h,j}-\delta_x\left(u^{1,n-1}_{N_1-h-1,j}-u^{1,n-1}_{N_1-h+1,j}\right)+\eta u^{1,n-1}_{N_1-h,j} \end{aligned} \tag{14}$$

avec $\eta=\frac{2D\Delta t}{\Delta x}\frac{\beta}{\alpha}$. La condition de Robin sur le domaine à droite modifie donc la première ligne de la matrice T ainsi que les N_y lignes du second membre associées :

$$T^{(2)} = \begin{bmatrix} \gamma+\eta & 2\delta_x & & & \\ \delta_x & \gamma & \delta_x & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \delta_x \\ & & & \delta_x & \gamma \end{bmatrix} \quad \text{et} \quad B^{(2)} = \delta_y I_{N_2} \tag{15}$$

$$\text{et} \quad (F_n^\star)^{(2)} = F_n^\star - \delta_x \begin{pmatrix} 0 \\ \vdots \\ 0 \\ u^{1,n-1}_{N_1-h-1,1} - u^{1,n-1}_{N_1-h+1,1} + \frac{2\Delta x\beta}{\alpha} u^{1,n-1}_{N_1-h,1} \\ 0 \\ \vdots \\ 0 \\ u^{1,n-1}_{N_1-h-1,j} - u^{1,n-1}_{N_1-h+1,j} + \frac{2\Delta x\beta}{\alpha} u^{1,n-1}_{N_1-h,j} \\ 0 \\ \vdots \\ 0 \\ u^{1,n-1}_{N_1-h-1,N_y} - u^{1,n-1}_{N_1-h+1,N_y} + \frac{2\Delta x\beta}{\alpha} u^{1,n-1}_{N_1-h,N_y} \end{pmatrix} \tag{16}$$

3.5 Décomposition en plus de deux sous-domaines

Après avoir étudié un partitionnement en deux sous-domaines, il est possible d'étendre le principe à n_d sous-domaines. Les matrices $A^{(k)}$ sur chaque sous-domaines sont de taille $N_k \times N_y$, pour $k = 1, ..., n_d$. Les changements apportés sur les matrices $A^{(1)}$ et $A^{(n_d)}$ et les seconds membres $(F_n^\star)^{(1)}$ et $(F_n^\star)^{(n_d)}$ sont conservés. Il y a donc $k-2$ sous-domaines avec des conditions sur leurs bords gauche et droit qui dépendent des autres sous-domaines. Les matrices $A^{(k)}$ correspondantes sont aussi tridiagonales par blocs, avec les matrices $T^{(k)}$ et $B^{(k)}$ sur la diagonale et sur les extra-diagonales, respectivement. Ces matrices sont elles de taille $N_k \times N_k$, pour $k = 1, ..., n_d$.

Avec conditions de Dirichlet. La première ligne et la dernière ligne des matrices T et B sont modifiées :

$$T^{(k)} = \begin{bmatrix} \frac{1}{\Delta x^2} & 0 & & & \\ \delta_x & \gamma & \delta_x & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_x & \gamma & \delta_x \\ & & & 0 & \frac{1}{\Delta x^2} \end{bmatrix} \quad \text{et} \quad B^{(k)} = diag_{N_k}(0, \delta_y, ..., \delta_y, 0). \tag{17}$$

Avec conditions de Robin. Seule la matrice T est modifiée sur sa première et sa dernière ligne :

$$T^{(k)} = \begin{bmatrix} \gamma+\eta & 2\delta_x & & & \\ \delta_x & \gamma & \delta_x & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_x & \gamma & \delta_x \\ & & & 2\delta_x & \gamma+\eta \end{bmatrix} \quad \text{et} \quad B^{(k)} = \delta_y I_{N_k}. \tag{18}$$

4 Parallélisation MPI

Le nombre de processus considéré est noté N_p et le processus courant est appelé me. Le paramètre de recouvrement est noté h. Le vecteur solution local, soit sur chaque domaine/processus, est noté U_{loc}.

4.1 Répartition des charges

La répartition de charge entre les processus est réalisé grâce à la fonction `charge` déjà implémentée l'année dernière. Cette fonction renvoie pour une charge donnée, ici N_x le nombre de maille selon $O\vec{x}$, les indices $IBeg$ et $IEnd$ de résolution pour chaque processus. Pour prendre en compte le recouvrement, la fonction est complétée en fonction du processus courant. Sur le premier processsus, soit $me = 0$, la indices calculés ne sont pas modifiés. Sur les autres processus, l'indice $IBeg$ est décrémenté de la taille du recouvrement h.

4.2 Architecture générale

Chaque processus construit sa matrice et son second membre comme détaillé en 3.5. Les processus $me = 0$ et $m = N_p - 1$ construisent leurs matrices et leurs second membres comme détaillé en 3.3, agissant respectivement comme le domaine 1 et le domaine 2.

4.3 Communication entre les processus

4.3.1 Envoi des conditions de bord

En raison du recouvrement, les conditions aux limites sur les bords des domaines sont déterminés par les domaines adjacents. Ainsi, une communication des solutions à l'itération ite doit être effectuée entre des processus contiguës pour le calcul de leur solution à l'itération $ite + 1$, dans l'algorithme de Schwarz. Concrètement, seulement trois colonnes, de taille N_y chacune, sont à échanger entre processus : c'est ce qui est appelé stencil. La communication se fait donc avec des vecteurs de taille $3 \times N_y$. Les notations adoptées sont $mssgL$ pour les stencils à envoyer au processus $me - 1$, soit le sous-domaine de gauche, et $mssgR$ pour les stencils à envoyer processus $me + 1$, soit le sous-domaine de droite.

Il faut noter que le processus $me = 0$ ne communique qu'avec le processus $me + 1 = 1$ et que le processus $me = N_p - 1$ ne communique qu'avec le processus $me - 1 = N_p - 2$.

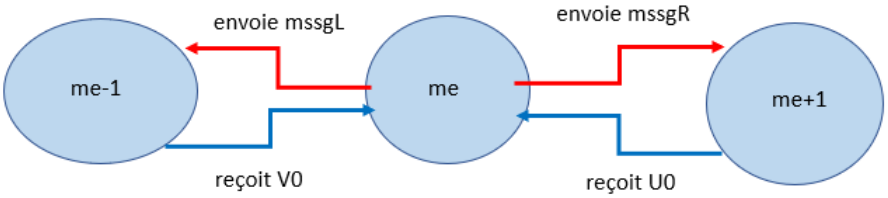

Figure 3 – Schéma des communications.

Pour chaque processus les indices des solutions à envoyer sont gérés en fonction du recouvrement de la manière suivante :

— à envoyer au processus $me + 1$
  - première colonne de $mssgR(me)$ correspond à la colonne d'indice $N - h - 2$ de U_{loc},
  - deuxième colonne de $mssgR(me)$ correspond à la colonne d'indice $N - h - 1$ de U_{loc},
  - troisième colonne de $mssgR(me)$ correspond à la colonne d'indice $N - h$ de U_{loc},

— à envoyer au processus $me - 1$
  - première colonne de $mssgL(me)$ correspond à la colonne d'indice $h - 1$ de U_{loc},

- deuxième colonne de $mssgL(me)$ correspond à la colonne d'indice h de U_{loc},
- troisième colonne de $mssgL(me)$ correspond à la colonne d'indice $h+1$ de U_{loc}.

Chaque processus reçoit donc sont ou ses stencils qui est ensuite stocké dans un vecteur U_0 pour le stencil reçu de $me+1$ et V_0 pour le stencil reçu de $me-1$, tous deux de taille $3 \times N_y$. Le stockage se fait en concordance avec les colonnes envoyées :

— à recevoir du processus $me+1$
  - première colonne de U_0 correspond à la première colonne de $mssgL(me-1)$,
  - deuxième colonne de U_0 correspond à la deuxième colonne de $mssgL(me-1)$,
  - troisième colonne de U_0 correspond à la deuxième colonne de $mssgL(me-1)$,

— à recevoir du processus $me-1$
  - première colonne de V_0 correspond à la première colonne de $mssgR(me+1)$,
  - deuxième colonne de V_0 correspond à la deuxième colonne de $mssgR(me+1)$,
  - troisième colonne de V_0 correspond à la deuxième colonne de $mssgR(me+1)$.

Le choix a été fait d'envoyer des stencils de solutions mais il aurait peut être été plus efficace d'envoyer plutôt des stencils en ayant d'abord effectué le calcul pour l'ajout dans le second membre.

4.3.2 Calcul de l'erreur

Par ailleurs, il faut que les processus calcule une erreur de Schwarz, qui est une condition d'arrêt de la boucle de l'algorithme de Schwarz. Ainsi, chaque processus sauf le premier doit envoyer sa solution sur la zone de recouvrement au processus précédent. Puis, tous les processus sauf le dernier calculent leur erreur de Schwarz. Le dernier processus reçoit alors du processus précédent l'erreur qui a été calculée.

Pour le calcul de l'erreur de Schwarz, la solution du processus me et du processus $me+1$ sur chaque point du recouvrement est comparée en valeur absolue. L'erreur choisie est le maximum de cette valeur absolue.

Il faut noter que la détermination de l'erreur de Schwarz n'est peut être pas la meilleure. Par ailleurs, chaque processus me regarde son erreur uniquement par rapport à la solution sur le processus $me+1$. Il aurait fallu, pour être plus cohérent, déterminer un calcul de l'erreur en prenant en compte le recouvrement à gauche et à droite.

5 Résultats obtenus

Concernant l'implémentation de la méthode, le code séquentiel de l'an dernier a été repris puis adapté pour effectué une méthode de décomposition de domaine, avec deux sous-domaines mais sur un seul processeurs. Ensuite, le code a été porté pour une architecture parallèle avec MPI, d'abord validant sur deux processeurs, puis en étendant pour un nombre de processeurs quelconque. Toute l'implémentation a été réalisée en prenant les conditions de Robin puis elle a été adaptée pour prendre en compte le cas particulier des conditions de Dirichlet.

5.1 Convergence des itérations de Schwarz

La Figure 4 présente l'erreur de Schwarz au cours des itérations successives méthode. Ces résultats ont été obtenus en utilisant une décomposition en deux sous-domaines, avec les conditions de Robin et les paramètres suivants : $N_x = 70$, $N_y = 70$, et un paramètre de recouvrement $h = 2$.

Sur les premières itérations, l'erreur est élevée à cause de l'initialisation qui est réalisée de façon arbitraire. Il faut plusieurs itérations pour que les solutions sur les sous-domaines se stabilisent puis

convergent. D'ailleurs, l'erreur peut avoir tendance à osciller avant de converger, comme cela est visible sur la FIGURE 4. Dans le cas de l'expérience montrée, l'algorithme a convergé après environ 80 itérations.

Cette étude nous a encouragé à choisir un nombre d'itérations maximum $N_x \times N_y$ pour s'assurer de la convergence de la solution.

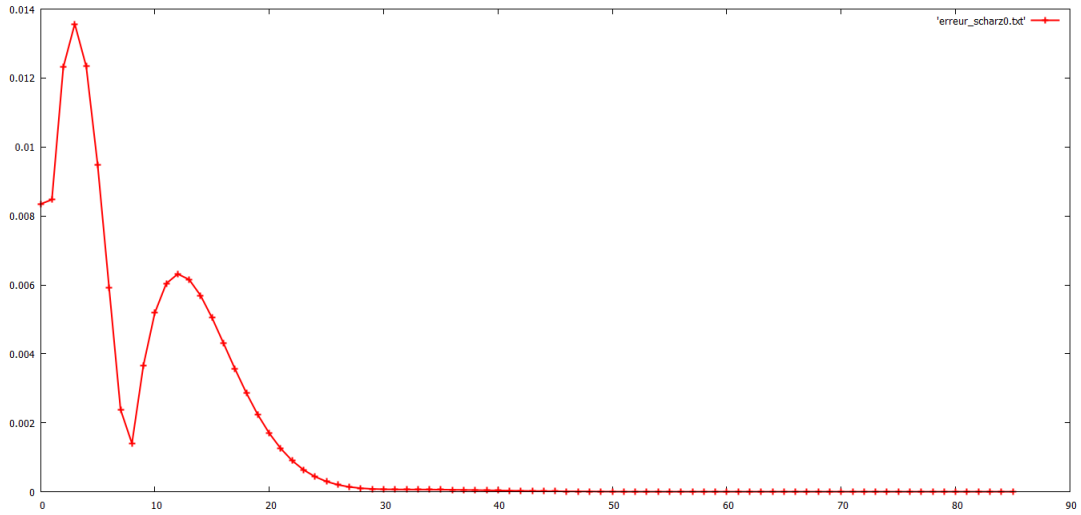

FIGURE 4 – Erreur de SCHWARZ en fonction du nombre d'itérations pour $N_x = 70$, $N_y = 70$ et $h = 2$, avec des conditions de DIRICHLET.

5.2 Validation

En vue de réaliser des études de performances, le code a pu être validé en utilisant deux cas stationnaires dont la solution exacte est connue :

$$f = 2(x - x^2 + y - y^2) \quad , \quad g = 0 \quad , \quad h = 0 \tag{19}$$

de solution exacte $u = x(1-x)y(1-y)$ et

$$f = \sin x + \cos y \quad , \quad g = \sin x + \cos y \quad , \quad h = \sin x + \cos y \tag{20}$$

de solution exacte $u = \cos x \sin y$.

Un cas instationnaire a aussi été implémenté mais non validé :

$$f = \exp\big(-(x - L_x/2)^2\big) \exp\big(-(y - L_y/2)^2\big) \cos\big(\frac{\pi}{2}t\big) \quad , \quad g = 0 \quad , \quad h = 1. \tag{21}$$

Pour construire notre code nous avons commencé par effectuer une décomposition de domaine sur deux sous-domaine en séquentiel. La précision de la solution obtenue par rapport à la solution exacte est validée. Nous avons de plus vérifié que l'erreur de SCHWARZ convergeait bien. Les résultats visuels pour ce cas de figure sont satisfaisants.

Pour le cas parallèle nous avons vérifié le code à partir d'un cas test et avec les résultats visuels. Pour la convergence de l'erreur de SCHWARZ malheureusement nous n'avons pas réussi à obtenir une convergence. Nous n'avons pas réussi à trouver le problème dans ce que nous avons codé pour la partie parallèle de toute évidence nous ne comparons pas les bonnes quantité car les résultats visuels nous prouvent que notre algorithme renvoie bien la bonne solution à la fin des itérations de SCHWARZ.

5.2.1 Condition de ROBIN

La FIGURE 5 présente la solution exacte avec les conditions initiales 19. Pour comparaison, la FIGURE 6 présente la superposition de la solution exacte et de la solution approchée par décomposition

de domaines avec 3 processeurs, en utilisant les conditions de Robin. Les paramètres utilisés pour pour cette approximation sont : $N_x = 70$, $N_y = 70$, et un paramètre de recouvrement $h = 2$.

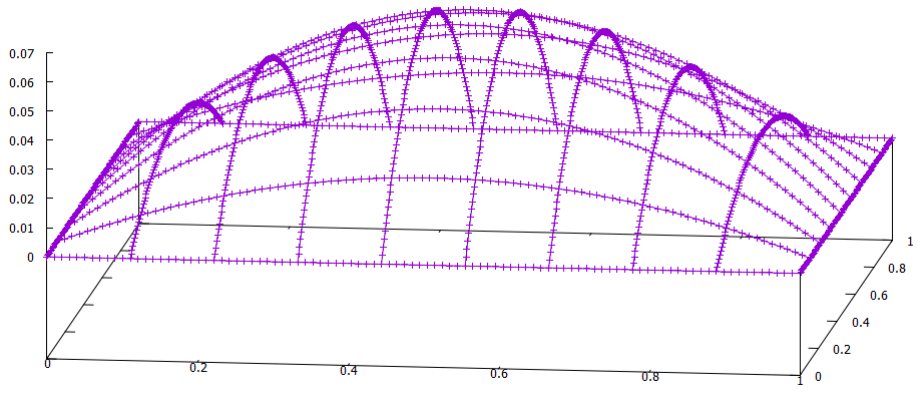

Figure 5 – Solution exacte pour le premier mode.

La solution exacte et celle approchée sont cohérentes et les recouvrements sont bien effectués. Les points sur les zones de recouvrements sont suffisamment proches même si pas tous confondus. Par ailleurs, la solution approchée a été calculée en utilisant 4, 6 et 10 processeurs et les résultats obtenus sont montrés dans l'Annexe A. Ces résultats restent conforment : la solution exacte est toujours bien approchée et les recouvrements bien réalisés.

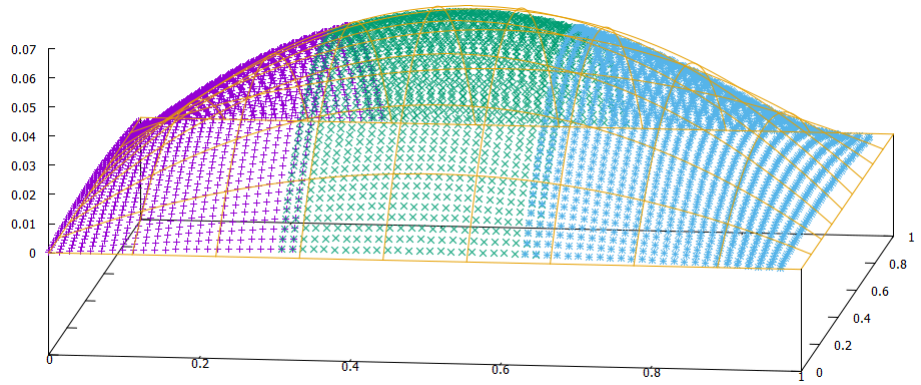

Figure 6 – Solution exacte et approchée sur 3 processeurs pour $N_x = 70$, $N_y = 70$ et $h = 2$, avec des conditions de Robin.

5.2.2 Condition de Dirichlet

La solution approchée avec les conditions de Dirichlet est présentée sur la Figure 7, en superpositiona vec la solution exacte 19. Les paramètres utilisés sont toujours les suivants : $N_x = 70$, $N_y = 70$, et un paramètre de recouvrement $h = 2$.

Comme précédemment, l'appréciation visuelle semble signifier que les résultats obtenus sont corrects. En revanche la précision de la solution plus importante avec les conditions mixtes qu'avec le cas particulier des conditions de Dirichlet.

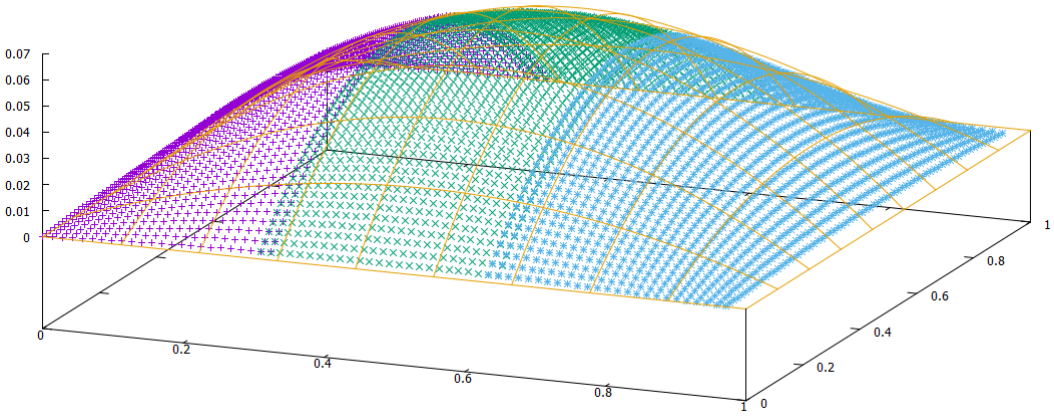

FIGURE 7 – Solution approchée sur 3 processeurs pour $N_x = 70$, $N_y = 70$ et $h = 2$, avec des conditions de DIRICHLET.

5.3 Influence du recouvrement

Le paramètre de recouvrement est central dans une méthode de décomposition de domaine. Des expérimentations ont été menées pour tenter de déterminer si un paramètre de recouvrement particulier était optimal. Les paramètres utilisés dans cette partie sont les suivants : $N_x = 50$ et $N_y = 50$, avec des conditions de ROBIN pour un partitionnement en deux sous-domaines.

La FIGURE 8 présente les temps de calculs obtenus en fonction du paramètres de recouvrement. En fixant le temps pour un recouvrement de $h = 2$ comme temps de référence, le graphe de droite est obtenu. Les mesures qui ont permis l'obtention de ces graphes sont regroupées dans la TABLE 1 de l'ANNEXE B.

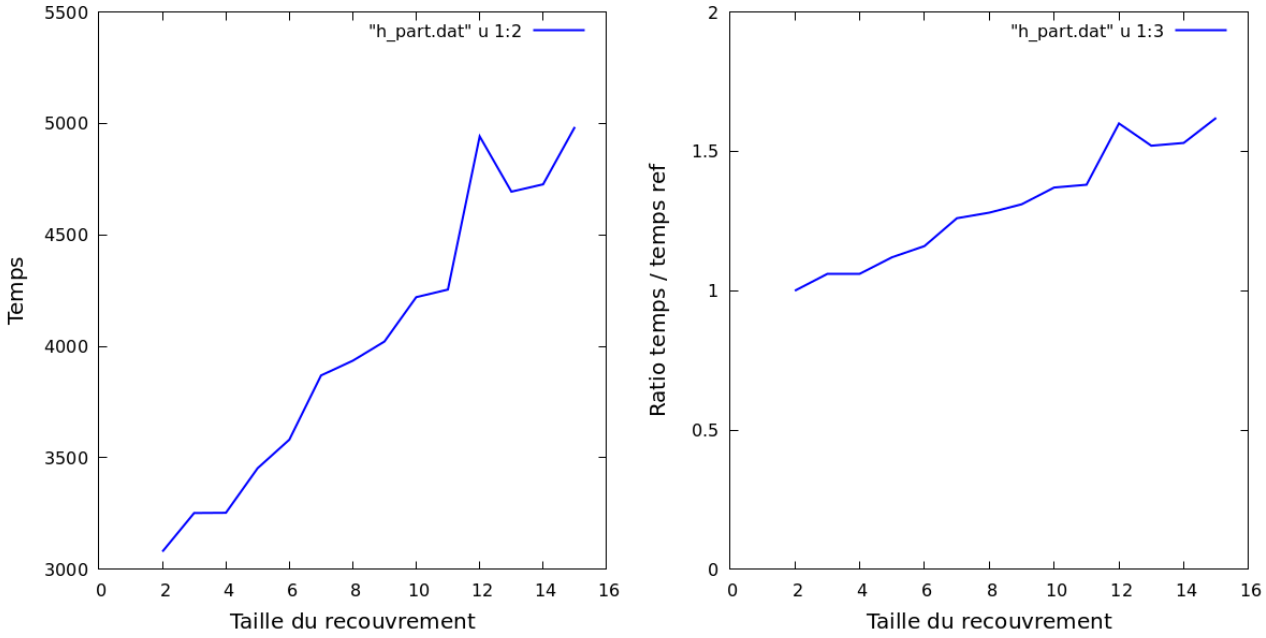

FIGURE 8 – Temps de calcul absolu (à gauche) et relatif (à droite) en fonction du recouvrement pour $N_x = 50$, $N_y = 50$, avec des conditions de ROBIN et une partition de 2.

Il faut remarquer que le temps de calcul n'est pas proportionnel à la taille du recouvrement. En

réalité, ce qui va être décisif dans le choix du paramètre de recouvrements est principalement le nombre de points de discrétisation. En effet, l'influence du recouvrement sur le temps de calcul du programme parallèle le plus important au moment du calcul de l'erreur de Schwarz, lorsque les processus doivent s'échanger $h \times N_y$ points.

5.4 Performance du code

La Figure 9 présente les courbes d'efficacité et de speed-up obtenus avec les paramètres suivants : $N_x = 100$, $N_y = 100$ et un recouvrement $h = 2$, avec des conditions de Robin. La courbe d'efficacité a été tracée grâce à l'équation (3) et la courbe de speed-up grâce à l'équation (2). Le résultat pour un processeur a été obtenu en utilisant le code séquentiel programmé l'an dernier. Les mesures qui ont permis l'obtention de ces graphes sont regroupées dans la Table 2 de l'Annexe B.

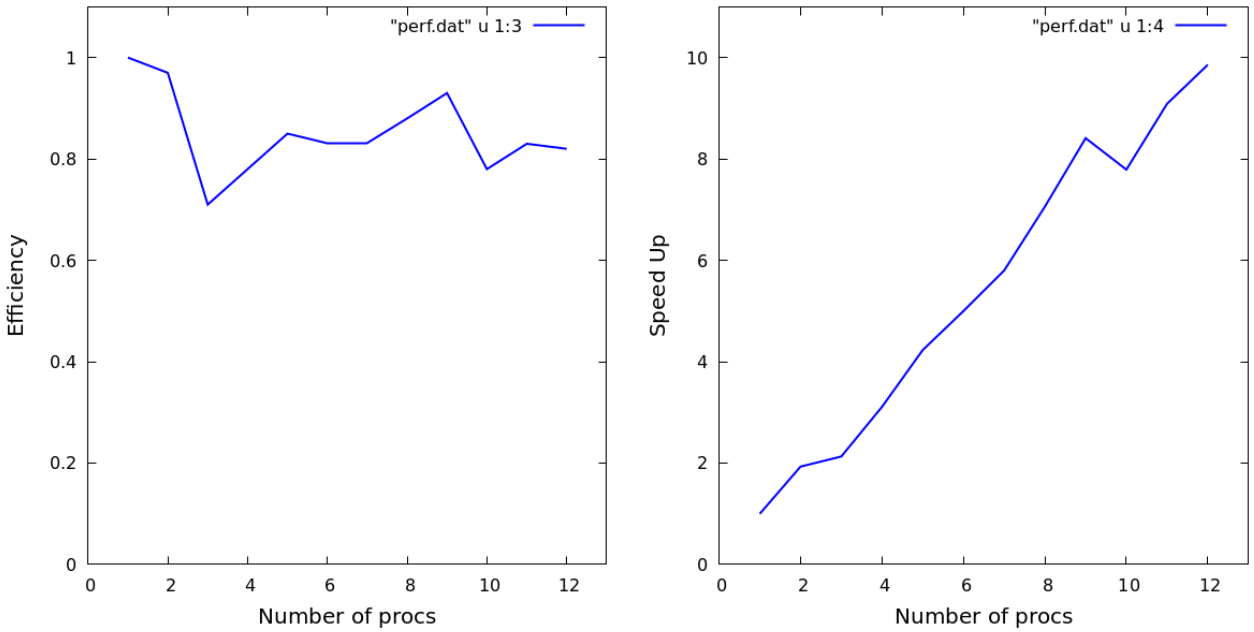

Figure 9 – Courbes d'efficacité (à gauche) et de speed-up (à droite) de notre programme parallèle pour $N_x = 100$, $N_y = 100$ et $h = 2$, avec des conditions de Robin.

6 Bilan du projet

Dans un premier temps, nous avons pu discuter de l'importance de l'équilibre de charge dans un code de calcul parallèle et notamment du lien avec le maillage utilisé pour discrétiser un problème. Nous avons pu voir deux manières de partitionner un maillage : en utilisant le graphe du maillage ou son graphe dual. Ces méthodes permettent d'équilibrer la charge théorique de calcul en équilibrant le nombre d'éléments par partition. Par ailleurs, si la charge de calcul est importante dans un code de calcul, la part des communications peut vite devenir importante dans un code parallèle si elles sont mal réalisées. Le choix a été fait d'utiliser communications synchrones et bloquantes. D'une part, l'encombrement mémoire est minimisé car aucune recopie des messages n'est effectuée dans un buffer. D'autre part, ce mode de communication permet de s'assurer que la réception a eu lieu.

Dans une deuxième partie la méthode de décomposition de domaine et en particulier celle de Schwarz additif a été détaillée, tant avec les conditions spécifiques de Dirichlet qu'avec les conditions de Robin. Dans une troisième partie, la programmation parallèle de notre code a été abordée en détaillant particulièrement les communications effectuées entre les processus.

Finalement, les résultats obtenus ont été présentés. Une étude sur la convergence de l'erreur de Schwarz nous a encouragé à choisir un nombre d'itérations maximum $N_x \times N_y$. Le programme a été validé pour plusieurs processeurs, sur les conditions de Dirichlet et de Robin, pour la solution exacte $u = x(1-x)y(1-y)$. Une étude sur l'influence du paramètre de recouvrement a été menée. Pour choisir sont paramètre de recouvrement, il faut en fait raisonner en fonction du nombre N_y de points de discrétisation selon $O\vec{y}$. Finalement, les performances du codes ont été résumée par les courbes d'efficacité et de speed-up tracées sur la Figure 9.

Annexes

A Résultats avec conditions de Robin

Les solutions obtenues par notre programme en utilisant un partitionnement sur 4 , 6, et 10 processeurs ont été tracées sur les Figures 10, 11 et 12. Les paramètres utilisés pour ces approximations sont : $N_x = 70$, $N_y = 70$, et $h = 2$. Pour rappel, la solution exacte pour ces expériences est : $u = x(1-x)y(1-y)$.

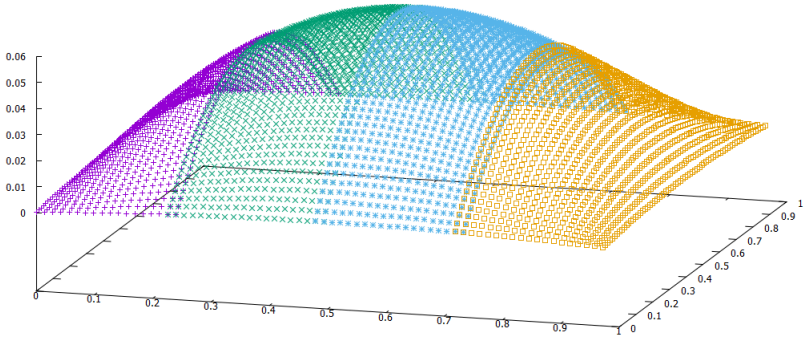

Figure 10 – Solution approchée sur 4 processeurs.

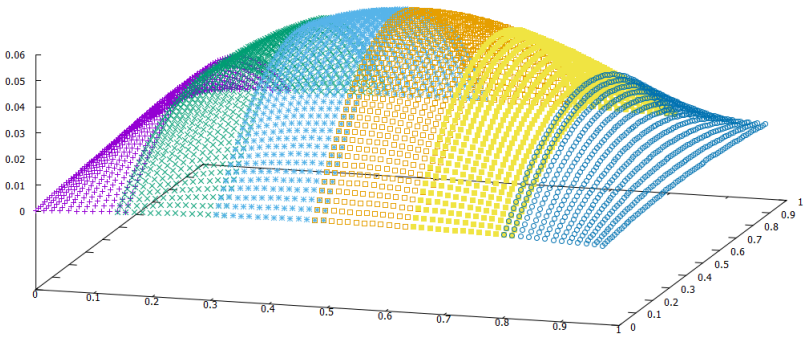

Figure 11 – Solution approchée sur 6 processeurs.

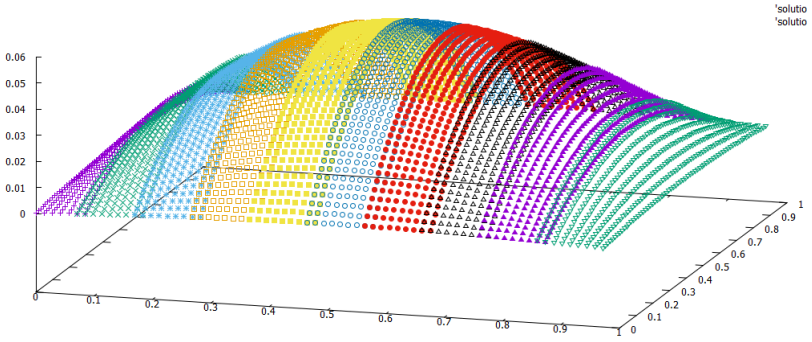

Figure 12 – Solution approchée sur 10 processeurs.

B Table des mesures

Cette annexe regroupes les mesures effectuées avec notre programme parallèle de décomposition de domaine. Les Tables 1 et 2 ont permis de créer les graphes des Figures 8 et 9. La machine utilisée pour obtenir ces résultats possède une configuration à 12 processeurs de modèle Intel(R) Xeon(R) E-2236 CPU @ 3.40GHz.

Paramètre de recouvrement	Temps
2	3080
3	3253
4	3254
5	3454
6	3582
7	3870
8	3936
9	4022
10	4221
11	4255
12	4942
13	4694
14	4727
15	4984

Table 1 – Mesures du temps d'exécution de notre en fonction du paramètre de recouvrement, pour un partitionnement de 2.

Nombre de processeurs	Temps
1	9522
2	9569
3	9521
4	9582
5	9565
6	9589
7	9575
8	9586
9	9507
10	9559
11	9511
12	9578

Table 2 – Mesures du temps d'exécution de notre programme en fonction du nombre de processeurs utilisés, pour un paramètre recouvrement de 2.