Projet de programmation CHP

Décomposition de domaine :
Additive Schwarz Method

MISTRAL Baptiste

SANS Damien

LEDERER Victor

Filière MATMECA - 3A CHP

Enseignante : BEAUGENDRE Héloise

1 Partie I : Équilibre de Charge

Les maillages

En modélisation, les maillages forment une partie essentielle du problème. Il s'agit d'un élément de support qui va permettre de calculer les solutions, valider un code et représenter le plus fidèlement possible un phénomène, en utilisant des méthodes de calcul pour résoudre numériquement des équations aux dérivées partielles (EDP) ou ordinaires (EDO). Ces maillages peuvent être réguliers avec des pavages structurés ou non structurés en fonction de la complexité de la géométrie ou encore des besoins de l'utilisateur en terme de précision ou de représentabilité.

Les difficultés liés à ces types de maillage sont les approximations de la solution obtenus en chaque point (ou centre de cellule), ainsi que le calcul des schémas numériques en ces points en fonction de la méthode utilisée (utilisation des normales aux cellules, centres des cellules..).

Pour générer des maillages, il y a plusieurs possibilités : si le maillage est de type cartésien régulier avec une géométrie simple, alors on peut calculer artificiellement les mailles et le calcul des solutions découle directement d'un schéma numérique à l'aide d'un pas d'espace. Très souvent dans ce cas, l'ensemble des mailles ont la même géométrie cartésienne : on parle de **maillage structuré**. Si la géométrie devient plus complexe, alors il devient nécessaire d'utiliser des logiciels de maillage (mailleurs) pour pouvoir disposer de fichiers contenant des informations sur le maillage et la géométrie définie en amont (ex : GMSH, EMC2,SCOTCH,METIS..etc). Suivant certains cas, les cellules n'ont pas toutes la même géométrie. On parle alors de **maillage non-structuré**. Les fichiers générés par les mailleurs sont utilisés en lecture d'un code pour pouvoir implémenter les schémas numériques correspondant au problème que l'on souhaite résoudre.
On donne l'exemple d'un maillage de canal généré par GMSH pour une simulation d'un code de Volumes finis 2D :
De la même manière en 3D, on a l'exemple d'un maillage de cylindre utilisé pour un calcul d'écoulement de type Poiseuille par calcul d'éléments finis (MEDIT) :

Le parallélisme

Lorsque l'on souhaite réaliser un code parallèle, il s'agit de répartir les données équitablement (suivant un processus d'équilibrage de charge) entre les différents processeurs. De ce fait, il est nécessaire que chaque processeur n'utilise que la partie du maillage qui lui est destinée. On utilise alors des logiciels de partitionnement de maillage qui vont permettre de générer un fichier de maillage par processeur. Á titre d'exemple, on peut fournir les solution d'un code de résolution de domaine par la méthode des éléments finis. Ce programme permet de résoudre le problème elliptique $-\vec{\nabla} \cdot (\vec{\nabla} u) = 0$ avec $u(x, y) = x + y$ sur $\partial\Omega$. Ce problème est résolu avec la méthode des éléments finis en résolvant le système matriciel en se basant sur la formulation variationnelle du problème. Le code utilise l'API MPI pour partager les informations sur les matrices entre les processeurs. A ce code sont reliés deux modules de partitionnement METIS et SCOTCH pour pouvoir générer les maillages partitionnés sur les différents logiciels.
Le code est construit en langage C et on donne les résultats obtenus avec le partitionneur Metis affichée avec Visit (à titre d'exemple) pour 1 processeur (cf figure (1)) et la construction complète de la solution sur l'ensemble du maillage (cf figure (2)) :

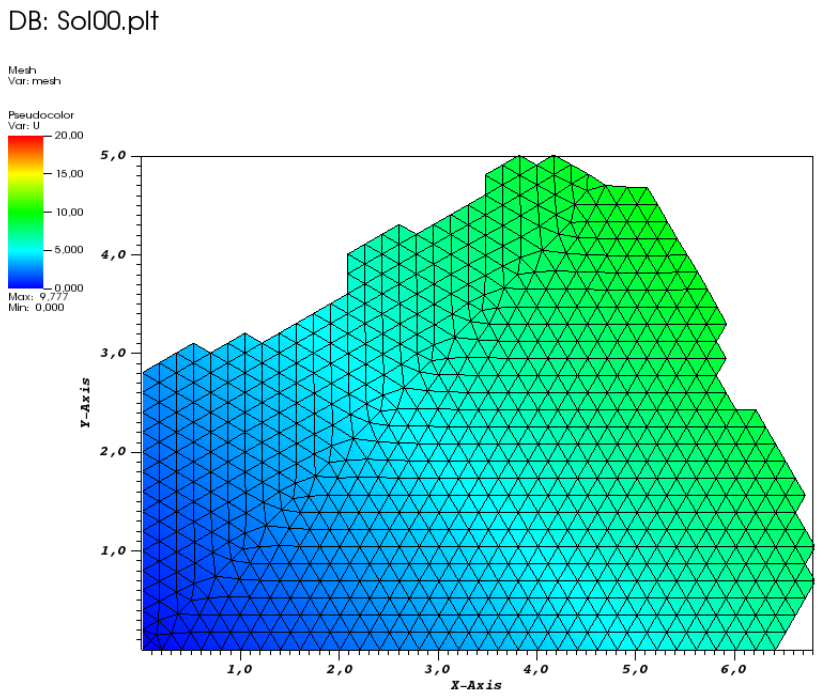

FIGURE 1 – Représentation de la solution du calcul obtenue par le processeur 0 sur 4 (5 processeurs au total).

D'après la thèse de Cédric CHEVALIER `https://www.labri.fr/perso/pelegrin/papers/these_cedric.pdf`, SCOTCH et METIS utilise des méthodes multi-niveaux parallèle (page.21). Ces méthodes sont composées de trois étapes, dont l'une d'elle est correspond à l'application d'une méthode heuristique pour trouver le partitionnement optimal (minimisation des interfaces et équilibrage de charge). La solution regroupée sur les différents processeurs puis rassemblés sur le maillage fournissent un résultat cohérent et complet, comme on peut le voir sur la représentation ci-dessous :

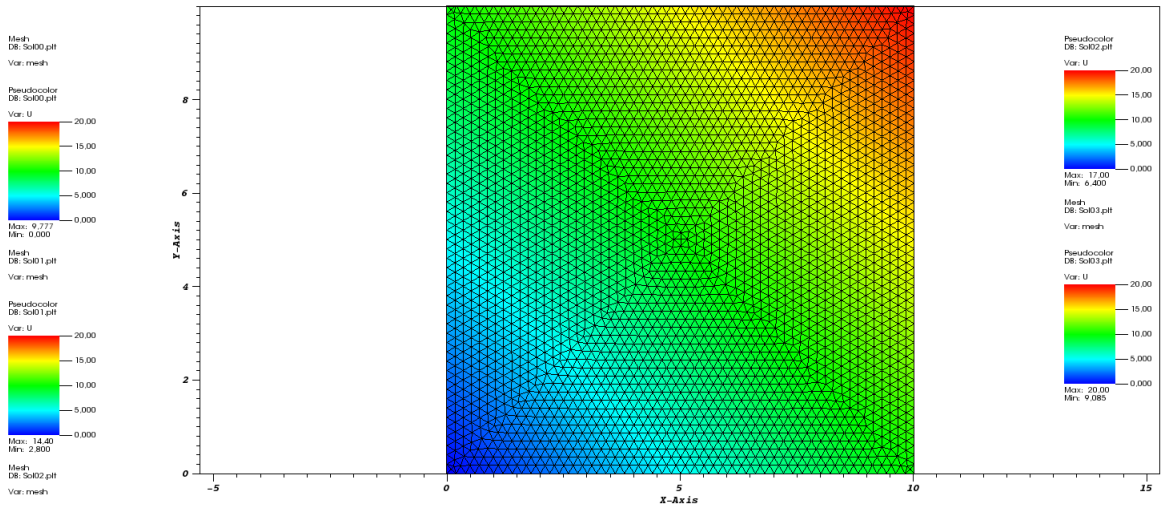

FIGURE 2 – Représentation de la solution globale du système sur le domaine obtenue avec 5 processeurs.

Pour illustrer le partitionnement de maillage, nous avons comparer sur les figures (3) et (4) la distribution de la charge par processeurs (Np=4) avec et sans partionnement pour un maillage (109300 triangles) qui sera utilisé lors d'un calcul d'éléments finis sur le problème du *Point Stress Criterion*(cf partie 2.2 :`https://www.hindawi.com/journals/isrn/2012/689386/`) En Regardant les figures (5) et (6), on constate que répartition de la charge est avec partitionnement minimise la connectivité entre les domaines et donc les messages entre les processus, tant dis que sans partitionnement les domaines

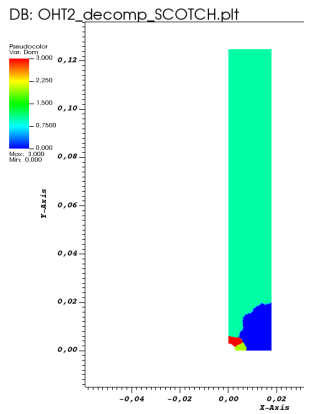

FIGURE 3 – Avec Partitionnement

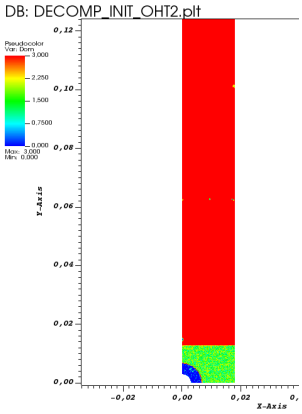

FIGURE 4 – Sans partitionnement

sont mélangés. Dans les deux cas la charge est répartie équitablement.

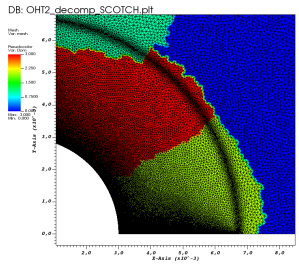

FIGURE 5 – Avec Partitionnement, Zoom

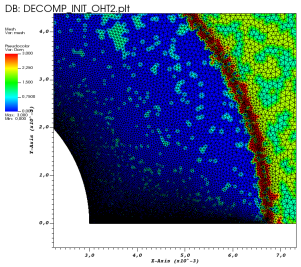

FIGURE 6 – Sans partitionnement, Zoom

Ainsi, lorsque l'on utilise un partitionneur de maillage pour réaliser du parallélisme sur un maillage, il est nécessaire de réaliser des communications entre les différents processeurs dans le code. En effet comme chauqe processeur ne connaîtra que la partie du maillage qui lui est propre, la continuité du résultat dépendra des échanges des informations aux frontières en imposant, par exemple, des conditions aux limites pour chaque domaine. Il est possible d'imposer une valeur, une dérivée ou une combinaison des deux. C'est précisément ce que nous allons étudier au cours de ce projet en procédant à une décomposition de domaine avec la méthode de Schwarz sur un maillage cartésien régulier.

2 Partie II : Mise en oeuvre d'une méthode de décomposition de domaine de type Schwarz sur un maillage cartésien régulier

2.1 Résolution de l'équation de conduction instationnaire

On se place dans le domaine $[0, L_x] \times [0, L_y]$ de R^2 dans lequel on résout l'équation de conduction instationnaire suivante :

$$\begin{aligned} \partial_t u(x,y,t) - D\Delta u(x,y,t) &= f(x,y,t) \\ u|_{\Gamma_0} &= g(x,y,t) \\ u|_{\Gamma_1} &= h(x,y,t) \end{aligned} \tag{1}$$

Avec $\Gamma_0 = \Gamma_G \bigcup \Gamma_D$ et $\Gamma_1 = \Gamma_H \bigcup \Gamma_B$ tels que $\partial\Omega = \Gamma_0 \bigcup \Gamma_1$.
L'objectif de ce projet est de résoudre cette équation sur un maillage cartésien régulier avec la méthode des différences finies en utilisant la méthode de décomposition de domaine. Pour valider les différentes méthodes proposées, on dispose de solutions analytiques (stationnaires ou non) qui permettront de comparer les résultats avec l'implémentation.

2.2 Conditions de bord et second membre pour les cas de validation

Pour pouvoir effectuer des cas de validation, on prendra les grandeurs suivantes : $L_x = L_y = 1$ et $D = 1$. On donne les expressions des solutions analytiques qui permettront de réaliser les comparaisons ainsi que les calculs d'erreur dans la suite :

Cas test 1 (Solution stationnaire)

$$\begin{array}{ccc} f = 2(x - x^2 + y - y^2) & & \text{avec } g = 0 \text{ et } h = 0 \\ u_e(x, y) & = & x(1-x)y(1-y). \end{array} \tag{2}$$

Cas test 2 (Solution stationnaire)

$$\begin{array}{ccc} f = sin(x) + cos(y) & & \text{avec } g = h = f \\ u_e(x, y) & = & sin(x) + cos(y). \end{array} \tag{3}$$

Cas test 3 (Solution instationnaire périodique)

$$f = e^{-\left(x-\frac{L_x}{2}\right)^2} e^{-\left(y-\frac{L_y}{2}\right)^2} cos(\tfrac{\pi}{2}t) \qquad \text{avec } g = 0 \text{ et } h = 0 \tag{4}$$

Pas de solution analytique.

2.3 Méthode de décomposition avec l'algorithme de Schwarz additif

La résolution numérique d'un problème par la méthode de décomposition avec l'algorithme de Schwarz additif consiste à diviser la résolution entre différents processeurs en partageant le domaine Ω global du problème entre ces différents processeurs, de telle sorte que chaque nouveau sous-domaine "déborde" sur ses sous-domaines voisins. Le débordement consiste à ce que tous les processeurs voisins possèdent une part de domaine de calcul en commun. C'est l'overlap ! Cet overlap permet à ce que deux processeurs voisins puissent se transmettre respectivement leurs conditions aux limites internes provenant de la solution numérique à l'itération en temps n pour le calcul de celle au temps t^{n+1}. Étant donné que à chaque nouvelle itération en temps les conditions aux limites internes de chaque processeur sont définies par la solution numérique de ses voisins à l'itération en temps précédente, celles-ci ne sont pas exactement celles qui proviendraient de la solution exacte. Il est donc nécessaire, pour chaque itération en temps, d'effectuer une boucle de *transmission de conditions aux limites/résolution* jusqu'à que la solution globale à cette itération n'évolue plus, et donc qu'elle ait convergé. C'est la boucle de Schwarz !

2.3.1 Discrétisation du domaine

Le segment $[0, L_x]$ est discrétisé en $N_x + 2$ points et le segment $[0, L_y]$ en $N_y + 2$ points, ce qui signifie que la discrétisation du domaine Ω global est discrétisée $(N_x + 2) \times (N_y + 2)$ points distincts.
On note Np le nombre de processus actifs et $CHARGE\_TOT$ la charge totale correspondant au nombre de points de calcul de la solution sur le **domaine intérieur**. Cette charge doit être distribuée équitablement entre les Np processus, pour cela on définit it_1 et it_N les bornes de travail des processus sur la numérotation globale des noeuds de calcul. On souhaite aussi connaitre l'intervalle de travail induit par it_1 et it_N des unités de calcul sur la discrétisation en X et en Y. En rappelant que $it_1 = P_{i1j1}$ et $it_N = P_{iNjN}$, on a la charge suivante en X : $[|S1, S2|]$ avec $i1 = S1$ et $iN = S2$ et la charge suivant Y est constante est et prise égale à N_y. Le mode de répartition est le suivant :

Répartition mode 1 : Elle effectue d'abord une répartion sur l'espace X en fonction de Np et N_x. Elle en déduit la charge globale par processeurs $[|it_1, it_N|]$ et fixe pour tout les processeurs la charge N_y en Y . Cela peut induire une différence maximale de charge globale de Ny entre deux processus et elle ne peut être utilisée que pour $Np \leq N_x$. A la fin de cette répartition, chaque processus a son propre domaine défini par la charge suivant X : $[|S1, S2|]$, la charge suivant Y : N_y et la charge locale sur la numérotation globale : $[|it_1, it_N|]$. Pour la suite la notion d'indice de ligne fait référence à la charge suivant X :$[|S1, S2|]$.

***Overlap* Dirichlet :** Pour le cas des conditions de Dirichlet sur les frontières immergées, *overlap* est un entier tel que la nouvelle charge sur le domaine intérieur soit :

$$S1 = S1 - (overlap - 1) \; et \; it1 = it1 - (overlap - 1) * N_y$$

$$S2 = S2 + (overlap - 1) \; et \; itN = itN + (overlap - 1) * N_y$$

***Overlap* Robin :** Pour le cas des conditions du type *Robin* la nouvelle charge sur le domaine intérieur se calcul via la charge calculée pour les conditions de Dirichlet soit :

$$S1 = S1 - 1 \; et \; it1 = it1 = it1 - N_y$$

$$S2 = S2 + 1 \; et \; itN = itN + N_y$$

Les processus 0 et $N_p - 1$ n'actualiserons respectivement que (S2,itN) et (S1,it) dans les deux types de condition. La figure (7) résume le principe de décomposition de domaine.

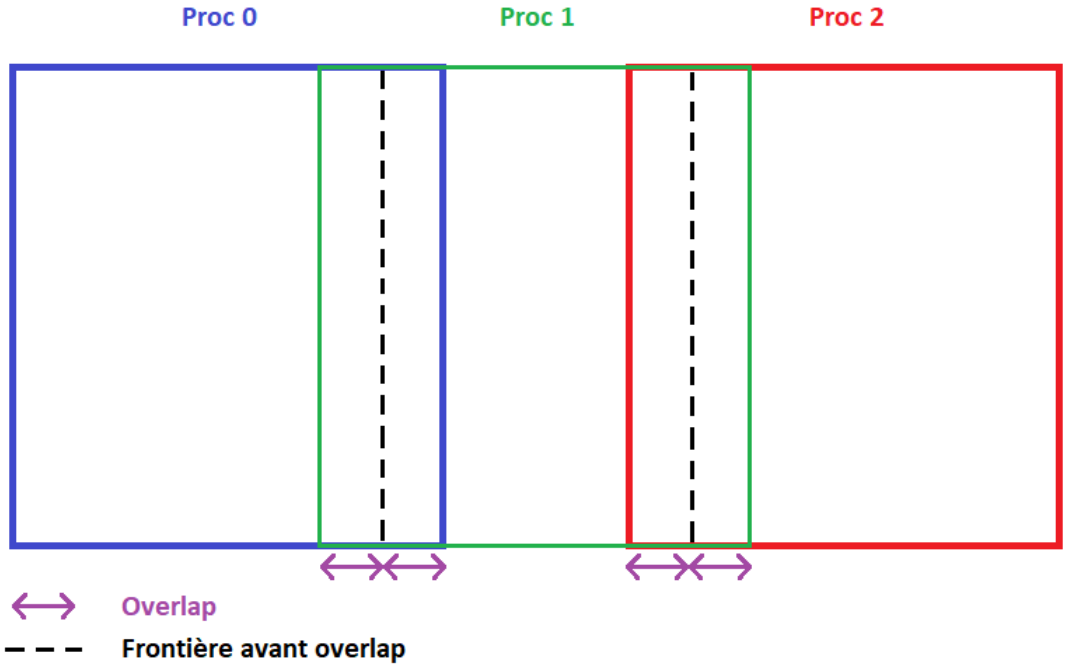

FIGURE 7 – Schéma de décomposition de domaine avec 3 processeurs pour l'algorithme de Schwarz additif.

2.3.2 Conditions de transmission de type Dirichlet

La condition de transmission de type Dirichlet consiste simplement à ce qu'un processeur transmette à ses voisins, au début du processus de calcul de la solution à une itération en temps, la solution numérique en leurs frontières de domaine respectives, que le processeur en question possède dans son domaine et qu'il a calculé à l'itération en temps précédente. Naturellement, il reçoit donc aussi, de ses voisins, la solution numérique de l'itération en temps précédente en ses frontières de domaine.

Considérant le processeur **N** qui possède une frontière immergée d'indice de ligne i dans le domaine du processeur **N+1**, la condition de transmission de type Dirichlet se traduit par :

$$u_{i,j}^{k+1,\mathbf{N}} \quad = \quad u_{i,j}^{k,\mathbf{N+1}} \quad \forall j \in [1, N_y - 1] \quad \forall k \in [1, N_t - 1] \tag{5}$$

Communications nécessaires et gestion du recouvrement

Chaque processeur doit donc transmettre autant de lignes de solution discrétisées qu'il a de frontières immergées dans les domaines de ses processeurs voisins. Et il en reçoit, bien entendu, autant. De ce fait, le premier et le dernier processeurs utilisés doivent envoyer et recevoir une ligne de solution discrétisée et tous les autres processeurs doivent en envoyer et en recevoir deux. Par exemple, dans le cas représenté dans la FIGURE 7, les processeurs 0 et 2 envoient et reçoivent une ligne et le 1 deux.

Résultats et performance de la méthode

On affiche les Speed-Up obtenus :

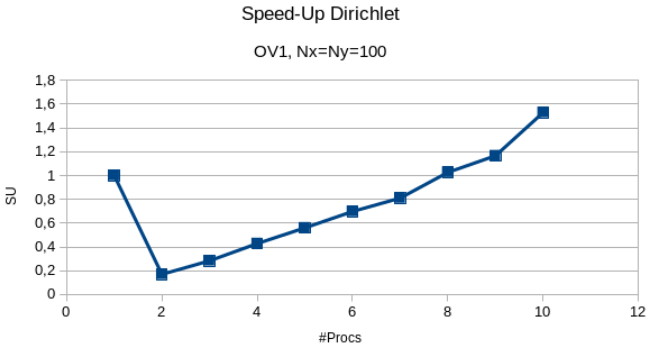

FIGURE 8 – Speed-up avec Dirichlet avec OV=1,Nx=Ny=100.

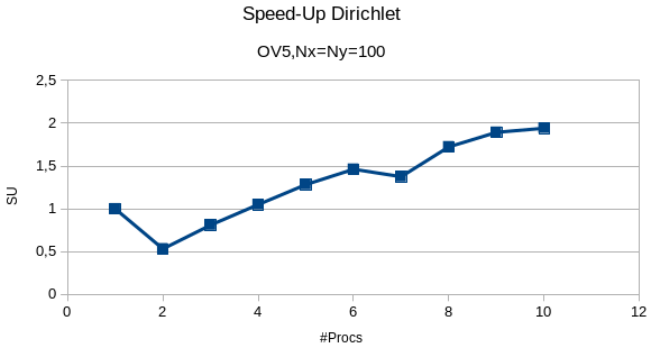

FIGURE 9 – Speed-up avec Dirichlet avec OV=5,Nx=Ny=100.

On affiche les efficacités obtenues :

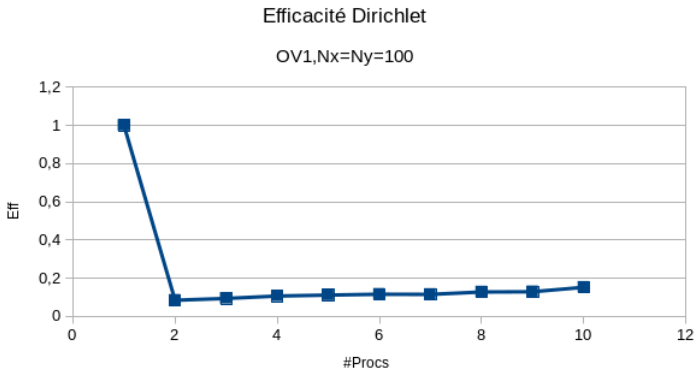

FIGURE 10 – Efficacités avec Dirichlet avec OV=1,Nx=Ny=100.

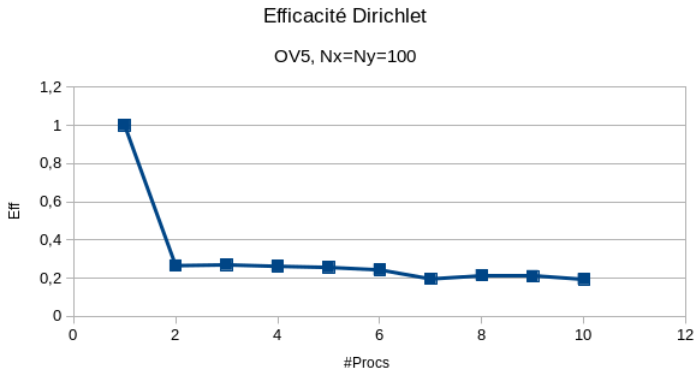

FIGURE 11 – Efficacités avec Dirichlet avec OV=5,Nx=Ny=100.

Et enfin ,on affiche les erreurs obtenues en norme 2 :

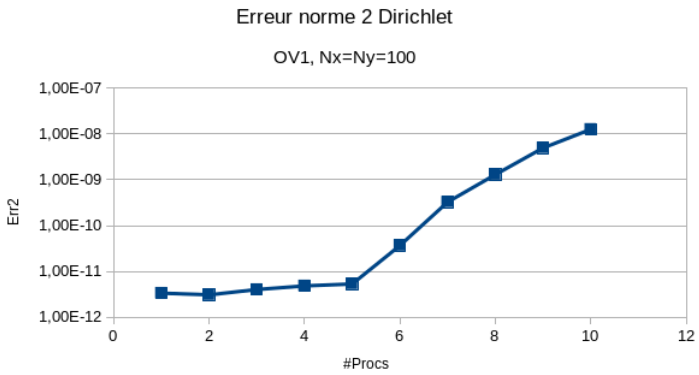

FIGURE 12 – Erreurs avec Dirichlet avec OV=1,Nx=Ny=100.

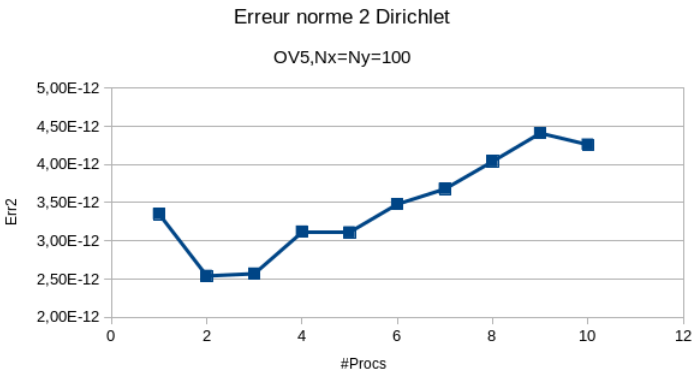

FIGURE 13 – Erreurs avec Dirichlet avec OV=5,Nx=Ny=100.

2.3.3 Conditions de transmission de type Dirichlet-Neumann

La condition de transmission de type Dirichlet-Neumann a pour but d'augmenter la vitesse de convergence de la solution numérique au cours du processus itératif de Schwarz. Ceci résulte donc à diminuer le nombre d'itérations de la boucle correspondante, surtout à la première itération en temps pour laquelle les conditions aux limites internes sont tout d'abord définies manuellement. En effet, puisque qu'elles sont définies par nous manuellement pour la première itération de Schwarz de la première itération en temps, elles sont donc beaucoup plus éloignées des vraies conditions dans ce cas. Alors que, pour les itérations en temps suivantes, elles sont définies à partir de la solution calculée précédemment et sont donc, dans ce cas, bien plus proches des vraies. La condition de transmission de type Dirichlet-Neumann permet donc de diminuer la quantité de calculs et le temps de simulation relatifs à la vitesse de convergence du processus itératif de Schwarz. Contrairement à celle de Dirichlet, elle ne consiste pas seulement en une condition d'égalité entre la solution numérique aux frontières immergées des différents domaines et celle correspondante provenant des processeurs voisins respectifs. Il y a en plus une condition d'égalité de dérivé en ces mêmes frontières immergées. C'est celle-ci qui doit augmenter la vitesse de convergence de la solution numérique au cours du processus itératif de Schwarz. Pour la condition de Dirichlet-Neumann, les frontières immergées de chaque processeur sont comprises dans leur domaine de calcul.

Considérant le processeur **N+1** qui possède une frontière immergée d'indice de ligne i dans le domaine du processeur **N**, la condition de transmission de type Dirichlet-Neumann (cas du débordement à gauche) se traduit par :

$$\begin{array}{ccc} \alpha \times u_{i,j}^{k+1,\mathbf{N+1}} + \beta \times \left(\frac{u_{i,j}^{k+1,\mathbf{N+1}} - u_{i-1,j}^{k+1,\mathbf{N+1}}}{\Delta x}\right) & = & \alpha \times u_{i,j}^{k,\mathbf{N}} + \beta \times \left(\frac{u_{i+1,j}^{k,\mathbf{N}} - u_{i,j}^{k,\mathbf{N}}}{\Delta x}\right) \\ \forall j \in [1, N_y - 1] & \forall k \in [1, N_t - 1] & \alpha, \beta \in R^* \end{array} \tag{6}$$

L'application de cette condition se traduit par une modification de la matrice et du second membre du problème pour chaque processeur.

Application au cas du débordement à gauche

L'injection de la condition de Dirichlet-Neumann (équation (6)) à partir du terme $u_{i-1,j}^{k+1,\mathbf{N+1}}$ de l'équation discrétisée du problème (équation (1)) suivante :

$$
\begin{aligned}
u_{i,j}^{k+1,\mathbf{N+1}} - \frac{D\Delta t}{\Delta x^2}\left[u_{i+1,j}^{k+1,\mathbf{N+1}} - 2u_{i,j}^{k+1,\mathbf{N+1}} + u_{i-1,j}^{k+1,\mathbf{N+1}}\right] \\
-\frac{D\Delta t}{\Delta y^2}\left[u_{i,j+1}^{k+1,\mathbf{N+1}} - 2u_{i,j}^{k+1,\mathbf{N+1}} + u_{i,j-1}^{k+1,\mathbf{N+1}}\right] \\
= u_{i,j}^{k,\mathbf{N+1}} + \Delta t \times F_{i,j}^{k}
\end{aligned}
\tag{7}
$$

Tous calculs faits, on obtient :

$$
\begin{aligned}
u_{i,j}^{k+1,\mathbf{N+1}}\left[1 + \frac{D\Delta t}{\Delta x^2} + 2\frac{D\Delta t}{\Delta y^2} + \frac{D\Delta t}{\Delta x}\frac{\alpha}{\beta}\right] \\
-\frac{D\Delta t}{\Delta x^2}u_{i+1,j}^{k+1,\mathbf{N+1}} - \frac{D\Delta t}{\Delta y^2}\left[u_{i,j+1}^{k+1,\mathbf{N+1}} + u_{i,j-1}^{k+1,\mathbf{N+1}}\right] \\
= u_{i,j}^{k,\mathbf{N+1}} + \Delta t \times F_{i,j}^{k} + \frac{D\Delta t}{\Delta x}\left[\frac{\alpha}{\beta}u_{i,j}^{k,\mathbf{N}} - \frac{1}{\Delta x}(u_{i+1,j}^{k,\mathbf{N}} - u_{i,j}^{k,\mathbf{N}})\right]
\end{aligned}
\tag{8}
$$

On remarque donc que le terme de la diagonale pour les $N_y - 1$ premières lignes de la matrice du processeur **N+1** devient $1 + \frac{D\Delta t}{\Delta x^2} + 2\frac{D\Delta t}{\Delta y^2} + \frac{D\Delta t}{\Delta x}\frac{\alpha}{\beta}$. De plus, les $N_y - 1$ premières valeurs du second terme deviennent $u_{i,j}^{k,\mathbf{N+1}} + \Delta t \times F_{i,j}^{k} + \frac{D\Delta t}{\Delta x}\left[\frac{\alpha}{\beta}u_{i,j}^{k,\mathbf{N}} - \frac{1}{\Delta x}(u_{i+1,j}^{k,\mathbf{N}} - u_{i,j}^{k,\mathbf{N}})\right]$.

Le cas du débordement à droite est analogue à celui à gauche. Ce sont cette fois le terme de la diagonale pour les $N_y - 1$ dernières lignes de la matrice et les $N_y - 1$ dernières valeurs du second terme qui sont modifiés. La valeur du terme de la diagonale dans ce cas est le même que pour le cas à gauche. Considérant le processeur **N** qui possède une frontière immergée d'indice i dans le domaine du processeur **N+1**, les $N_y - 1$ dernières valeurs du second terme deviennent $u_{i,j}^{k,\mathbf{N}} + \Delta t \times F_{i,j}^{k} + \frac{D\Delta t}{\Delta x}\left[\frac{\alpha}{\beta}u_{i,j}^{k,\mathbf{N+1}} - \frac{1}{\Delta x}(u_{i-1,j}^{k,\mathbf{N+1}} - u_{i,j}^{k,\mathbf{N+1}})\right]$.

Communications nécessaires et gestion du recouvrement

La condition de transmission de type Dirichlet-Neumann nécessite le double de la quantité de communications de celle de Dirichlet. Ceci se vérifie avec l'équation de la condition de transmission de type Dirichlet-Neumann (équation (6)) qui implique deux termes de deux lignes successives du processeur voisin. De ce fait, chaque processeur doit donc transmettre deux fois plus de lignes de solution discrétisées qu'il a de frontières immergées dans les domaines de ses processeurs voisins. Et il en reçoit, bien entendu, autant. De ce fait, le premier et le dernier processeurs utilisés doivent envoyer et recevoir deux lignes de solution discrétisée et tous les autres doivent en envoyer et en recevoir quatre. Par exemple, dans le cas représenté dans la FIGURE 7, les processeurs 0 et 2 envoient et reçoivent deux lignes et le 1 quatre.

Résultats et performance de la méthode

Les valeurs des coefficients des conditions de transmission de Dirichlet-Neumann utilisées sont : $\alpha = \beta = 0.5$. Ces valeurs semblent être le meilleur choix en terme de performances.

On affiche les Speed-Up obtenus :

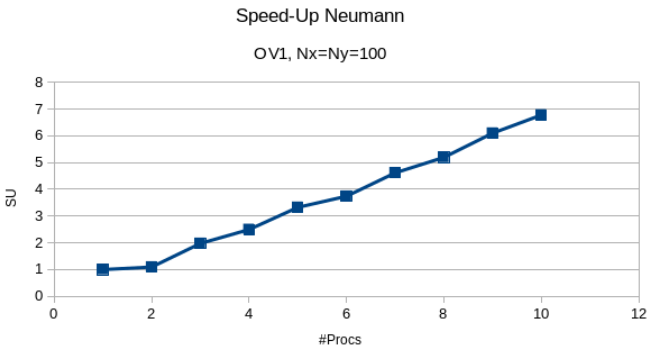

FIGURE 14 – Speed-up avec Neumann avec OV=1,Nx=Ny=100.

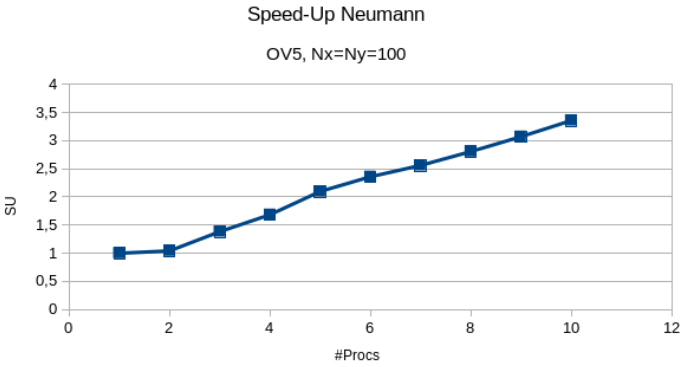

FIGURE 15 – Speed-up avec Neumann avec OV=5,Nx=Ny=100.

On affiche les efficacités obtenues :

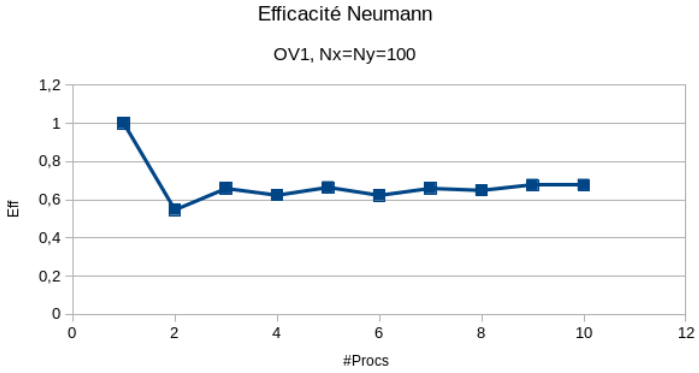

FIGURE 16 – Efficacités avec Neumann avec OV=1,Nx=Ny=100.

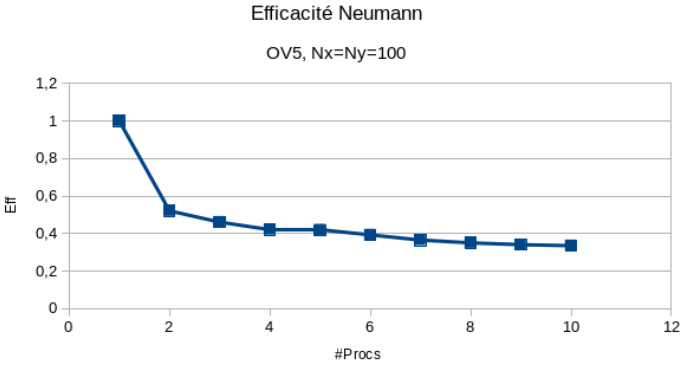

FIGURE 17 – Efficacités avec Neumann avec OV=5,Nx=Ny=100.

Et enfin ,on affiche les erreurs obtenues :

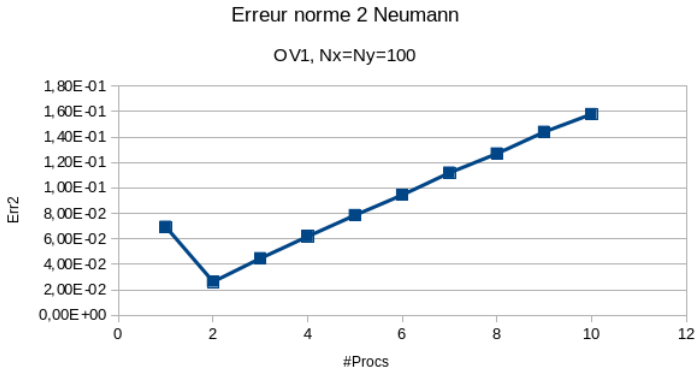

FIGURE 18 – Erreurs avec Neumann avec OV=1,Nx=Ny=100.

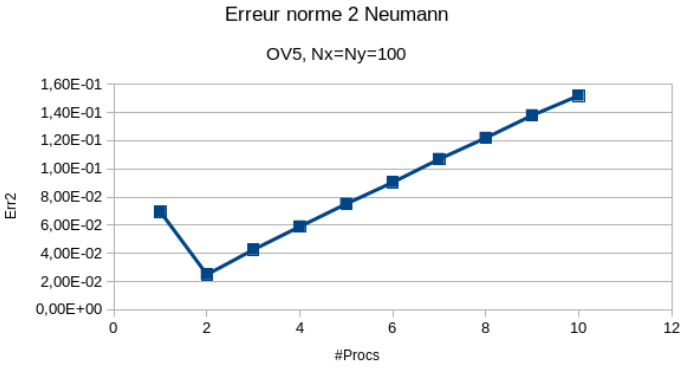

FIGURE 19 – Erreurs avec Neumann avec OV=5,Nx=Ny=100.

3 Discussion des résultats

Les résultats obtenus proviennent de simulation effectuées avec la plate-forme de calcul bordelaise PlaFRIM. Il semblerait qu'un temps de latence du à l'exécution en parallèle des programmes impacte

légèrement les performances.

Si l'on compare les courbes de résultats correspondants aux conditions de transmission de Dirichlet (voir 2.3.2) avec celles correspondants aux conditions de transmission de Dirichlet-Neumann (voir 2.3.3), on constate, à partir des courbes de speed-up et d'efficacité, que celles de Dirichlet-Neumann sont plus performantes. En effet, l'augmentation de vitesse de convergence du processus itératif de Schwarz par celles-ci permet d'obtenir de meilleurs résultats en terme de parallélisation. Par exemple, pour les conditions suivantes : $OV = 1$, $N_x = N_y = 100$ et $Np = 4$, on obtient pour la boucle de Schwarz de la première itération en temps, un nombre d'itérations maximal de 961 et un temps de calcul maximal de $3.84s$ parmi tous les processus avec la condition de Dirichlet alors que, avec la condition de Dirichlet-Neumann, on obtient un nombre d'itérations maximal de 114 et un temps de calcul maximal de $3.25s$ parmi tous les processus.

Cependant, les erreurs obtenues dans le cas de Dirichlet-Neumann sont bien moins bonnes en prennant $\alpha = 0.5$(condition sur u) et $\beta = 0.5$ (condition sur la dérivée de u) que celles obtenues dans le cas de Dirichlet. De plus, l'augmentation de l'overlap implique, dans le cas de Dirichlet, un diminution de l'erreur non-négligeable qui ne se retrouve pas dans le cas de Dirichlet-Neumann. Si l'on modifie les coefficients de la condition de Dirichlet-neumann (voir équation (6)) de telle sorte que β (prendre $1E-6$) soit négligeable devant α (prendre 1) alors on constate que l'on retombe sur les erreurs obtenues dans le cas de Dirichlet et du séquentiel. Il semblerait donc que c'est la modification du système par la condition de Neumann qui implique de moins bonnes erreurs.

4 Conclusion

Dans cette étude, nous avons étudié la résolution de l'équation de la chaleur 2D avec la méthode des différences finies en utilisant la décomposition de Schwarz additif. Nous avons vu que cette méthode permettait bien d'effectuer des calculs en parallèle en divisant le domaine en fonction du nombre de processus et que le sens de la découpe n'était pas unique. En effet, on aurait pu utiliser une découpe sur y ou encore x et y, et la méthode aurait été similaire. Il a aussi été vu qu'il était possible d'imposer différents types de conditions aux limites pour la méthode de Scharz : soit imposer une valeur constante (Dirichlet), soit imposer un gradient normal avec une valeur (Robin). Les résultats montrent que la méthode dite "Dirichlet" est plus efficace en ce qui concerne la précision du résultat mais a les valeurs d'efficacité les plus faibles. Cependant, la méthode "Robin" offre de meilleurs résultats en terme de parallélisme et fournit des temps CPU inférieurs. Même si l'objectif de la décomposition de domaine est d'offrir un temps CPU inférieur tout en conservant une erreur d'approximation faible, aucune des deux méthodes étudiées ne propose de résultats satisfaisants. De ce fait, on peut conclure que dans le cas étudié, il est plus efficace de paralléliser le solveur sur l'ensemble du domaine plutôt que de répartir le domaine sur les différents processus. De plus, la modification du domaine de calcul fait changer la structure de la matrice du système et impose le choix du solveur : il doit être plus robuste et polyvalent en fonction des conditions aux limites entre les domaines. Pour améliorer les speed-up, on peut faire des communications non bloquantes tel qu'un processus ayant envoyé son message puisse commencer à remplir son second membre avec les informations qu'il dispose à ce moment et le compléter lorsqu'il reçoit son message des autre processus. Comme amélioration, on peut aussi proposer d'augmenter l'ordre de précision du schéma différences finies pour comparer l'efficacité de la méthode, ou encore changer de méthode d'approximation de la solution (volumes finis, éléments finis...). De plus, il est possible que la décomposition soit plus efficace dans le cadre d'un changement de géométrie. Il serait alors intéressant d'étudier les résultats de la méthode avec un maillage non structuré qui prend en compte le changement de forme : on pourrait alors proposer un partitionnement de domaine avec les partitionneurs SCOTCH et/ou METIS vus en Partie I.