Partitionnement et décomposition de domaine en calcul parallèle

Utilisation de MPI, Metis, Scotch, Paraview et Medit

Rapport de projet

Lasvenes Rémi & Pannetier Valentin.

Université de Bordeaux - ENSEIRB MATMECA / Bordeaux INP- 28 janvier 2021
Master Informatique - option Calcul Intensif et Sciences des Données & Master Modélisation Numérique et Calcul Haute Performance Année 2020-2021

Table des matières

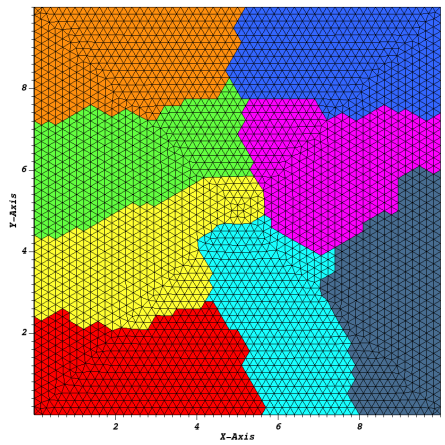
1 Équilibrage de charge

Il existe principalement deux types de maillages lorsqu'on s'intéresse à de la simulation pour du calcul numérique ; les **maillages structurés**, et les **maillages non-structurés**, tous les deux présentant avantages et inconvénients, qu'il convient d'utiliser en fonction du problème que l'on traite.

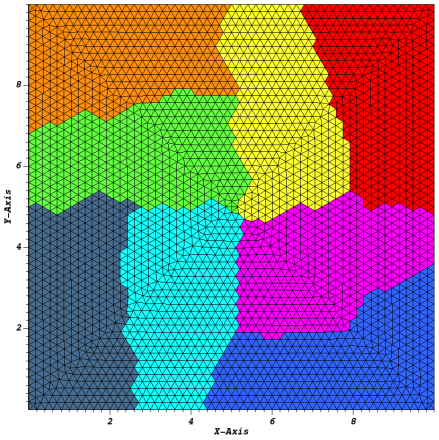
Maillages structurés Ce sont des maillages qui sont assez simples dans leur mise en place, assez peu volumineux, si on veut faire une modification sur une partie du maillage, c'est assez simple à mettre en place. De la même manière, lorsque l'on veut communiquer avec nos voisins, il est plus facile de déterminer ce qu'on veut envoyer. L'accès aux voisins est aussi facilité. En revanche, cela peut être compliqué à mettre en place lorsqu'on étudie un phénomène avec une géométrie complexe.

Maillages non-structurés Ils conviennent parfaitement pour des géométries complexes, et contiennent aussi moins de points par rapport à un maillage structuré. Pour leur génération, il existe plusieurs outils/algorithmes permettant d'automatiser ce processus. Néanmoins, en ce qui concerne les données à envoyer au sein d'une maille, cela est plus difficile, puisqu'elles vont avoir des formes plus ou moins grandes, le temps de communication sera alors irrégulier.

Exemple avec Metis et Scotch Pour comparer les différents algorithmes de partitionnement qui existent, voici un exemple sur des maillages vu pendant le cours, l'un obtenu avec Metis (figure 1a), et l'autre obtenu avec Scotch (figure 1b).



(a) Exemple de partitionnement avec *Metis* en utilisant 8 processus.



(b) Exemple de partitionnement avec *Scotch* en utilisant 8 processus.

FIGURE 1 – Différents partitionnement de maillages avec *Metis* et *Scotch*.

1.1 Éléments finis (EF)

Afin de réaliser les différents speed-up avec le code Éléments Finis, il y a toute une partie de pré-traitement, entre le moment où on part du maillage, et le moment où on partitionne ce même maillage en sous-domaines pour résoudre notre problème. La figure 2 illustre ce traitement.

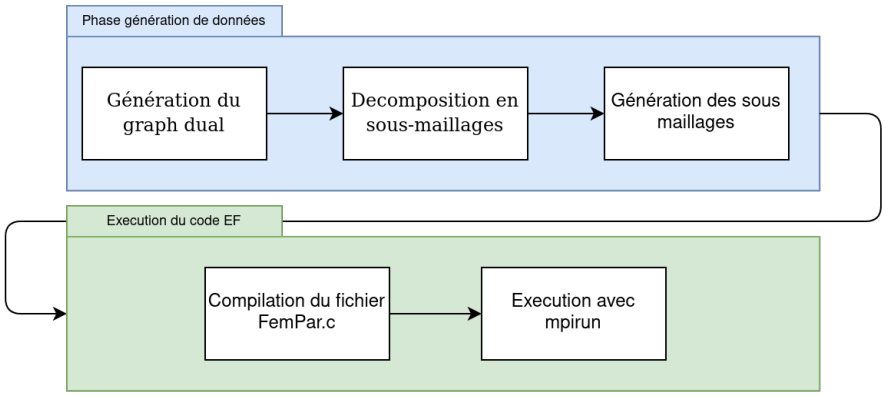
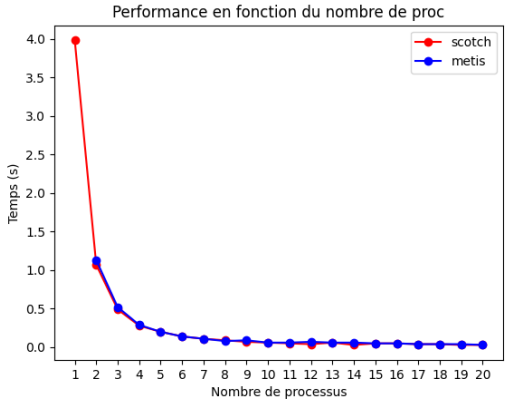


FIGURE 2 – Processus de génération des sous-maillages et d'exécution du code EF.

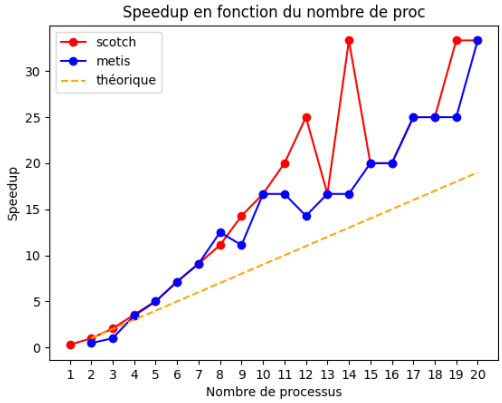
Il a été rajouté dans le code EF un *timer* mesurant les performances en temps du gradient conjugué, qui sera ensuite écrit dans un fichier, c'est de cette manière que nous allons pouvoir calculer le speed-up par la suite.

Remarque : lors de la partie découpage en sous-maillages, **Metis** génère une erreur si on essaie de le faire pour *une* partition seulement ($k = 1$), mais ce n'est pas le cas avec **Scotch**.

1.2 Courbes et performances



(a) Performances obtenues (en *seconde*) sur *plafrim* avec un maillage initiale de 5000 nœuds.



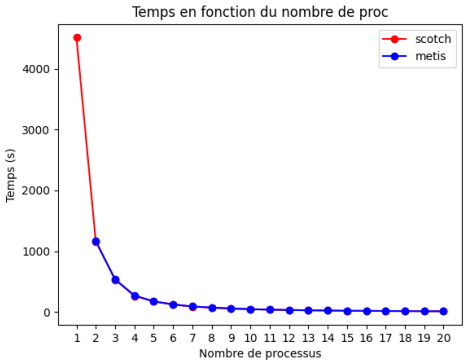
(b) Speed-up obtenu pour un maillage de 5000 nœuds.

FIGURE 3 – Courbes de performances et de speed-up avec un partitionnement Metis et Scotch.

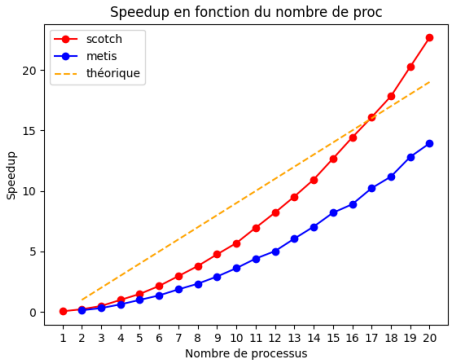
On remarque que sur la courbe de la figure 3a, le temps d'exécution diminue au fur et à mesure que l'on augmente le nombre de processus, ce qui est un comportement attendu. En revanche ce qui l'est moins, c'est que lorsqu'on calcule le speed-up [1], on se retrouve avec une valeur bien supérieure au speed-up théorique [2]. Ce qui peut expliquer cela, c'est le fait que le maillage de base soit si petit, et qu'on se retrouve avec la plupart des valeurs du calcul stockées dans le cache.
Nous avons donc testé sur un maillage bien plus grand (90000 nœuds), afin de voir si on retrouvait les mêmes observations que précédemment.
Les figures 3a et 4a ont la même allure, et le temps passé à calculer semble proportionnel au nombre de processus. Pour ce qui est du speed-up sur le maillage de taille 90000 (figure 4a et 4b), nous avons pris comme valeur de référence le temps d'exécution pour 4 processus (et non 1), afin d'obtenir des valeurs

1. Calculé en fonction du temps et du nombre de processus, avec comme valeur de référence le temps pris sur 2 processus
2. Calculé par rapport au temps de référence ; avec N processus on est censé aller N fois plus vite que le temps de référence (au plus)

plus proches de la réalité. Il se trouve que nous observons le même phénomène que précédemment, ce qui est assez étrange vu la taille du problème en entrée du programme.



(a) Performances obtenues (en *seconde*) sur *plafrim* avec un maillage initial de 90000 nœuds.



(b) Speedup obtenu pour un maillage de 90000 nœuds.

FIGURE 4 – Courbes de performances et de speed-up avec un partitionnement Metis et Scotch

Parallélisme En ce qui concerne le parallélisme effectué dans le code, il se situe au niveau de la fonction `InnerProduct()`, où chaque processus va faire son produit scalaire localement, pour ensuite faire une réduction sur toutes les partitions et les sommer. Également, la fonction `Update()` comprend des directives MPI, pour l'envoi et la réception des données partagées avec les processeurs voisins, ce qui se traduit par des directives `MPI_Isend()` et `MPI_Irecv()` afin de ne bloquer ni l'envoi, ni la réception de données.

Environnement Pour ce qui est de l'environnement de calcul utilisé, nous avons utilisé Plafrim, plus précisément les nœuds mistral, en se mettant seul sur un nœud à chaque fois (option `--exclusive`) sur un nombre de cœurs variant de 1 à 20.

2 Décomposition de domaine de Schwarz sur maillage cartésien

Dans cette partie nous étudierons la mise en œuvre de l'algorithme de Schwarz additif pour une équation bien connue et facile à discrétiser : l'équation de Poisson. Il est important de souligner que toutes les remarques que nous ferons dans la suite sont propres à ce problème et à la taille du stencil pour obtenir une erreur d'ordre 2 : il est facile de remarquer que le problème est tout autre lorsque nous avons des stencils plus grands (ne serait-ce que pour obtenir de l'ordre 4 pour le même problème).

2.1 Domaine de calcul et équation

À l'origine nous devions étudier l'équation de la chaleur, mais comme la seule différence par rapport au problème de Poisson réside dans l'ajout d'une boucle temporelle, nous avons décidé de nous concentrer uniquement sur le cas stationnaire

$$\begin{cases} -\Delta u = f & \text{dans } \Omega \\ u = g & \text{sur } \partial\Omega \end{cases}, \tag{1}$$

pour un ouvert Ω de $\mathbb{R}^2$ et f, g deux fonctions de $\Omega \to \mathbb{R}$.

La condition de Dirichlet sur le bord de Ω pourrait être aisément remplacée par une condition de Neumann sur une partie de ce bord et nous aurions toujours un problème bien posé. Ce cas n'est pas étudié ici, mais est très simple à mettre en œuvre pour un stencil de taille 5 sur un carré. Concernant Ω lui-même nous avons choisi un domaine carré, de préférence $[0,1]x[0,1]$ pour mener nos simulations

car c'est un domaine qui se prête à la discrétisation en maillage structuré.

Deux problèmes seront résolus par la suite

$$f_1(x, y) = 2x(1 - x) + 2y(1 - y) \qquad \text{et} \qquad g_1 \equiv 0 \tag{$\star$}$$

de solution analytique $u_1(x, y) = xy(1 - x)(1 - y)$

et

$$f_2(x, y) = \sin(x) + \cos(y) \qquad \text{et} \qquad g_2 = f_2 \tag{$\star\star$}$$

de solution analytique $u_2 = -f_2$.

2.2 Décomposition de domaine Schwarz additif

2.2.1 Décomposition de domaine

La décomposition de domaine consiste à partitionner le domaine de calcul Ω en plusieurs morceaux, les *partitions*. Le but est d'équirépartir le nombre de degrés de liberté entre chaque domaine nouvellement créé, ainsi que de résoudre en *parallèle* le même problème avec des tailles moindres par rapport au domaine de calcul initial. Ces résolutions sont alors découplées et peuvent s'effectuer *en même temps*. C'est ici que le calcul parallèle prend tout son sens.

Un seul défaut demeure pourtant : il faut **harmoniser** les valeurs entre les partitions puisqu'à l'échelle du domaine entier, un degré de liberté ne peut avoir qu'une seule valeur et la réunion des différentes solutions trouvées doit correspondre à l'unique solution valable sur le domaine Ω tout entier (le problème initial étant bien posé).

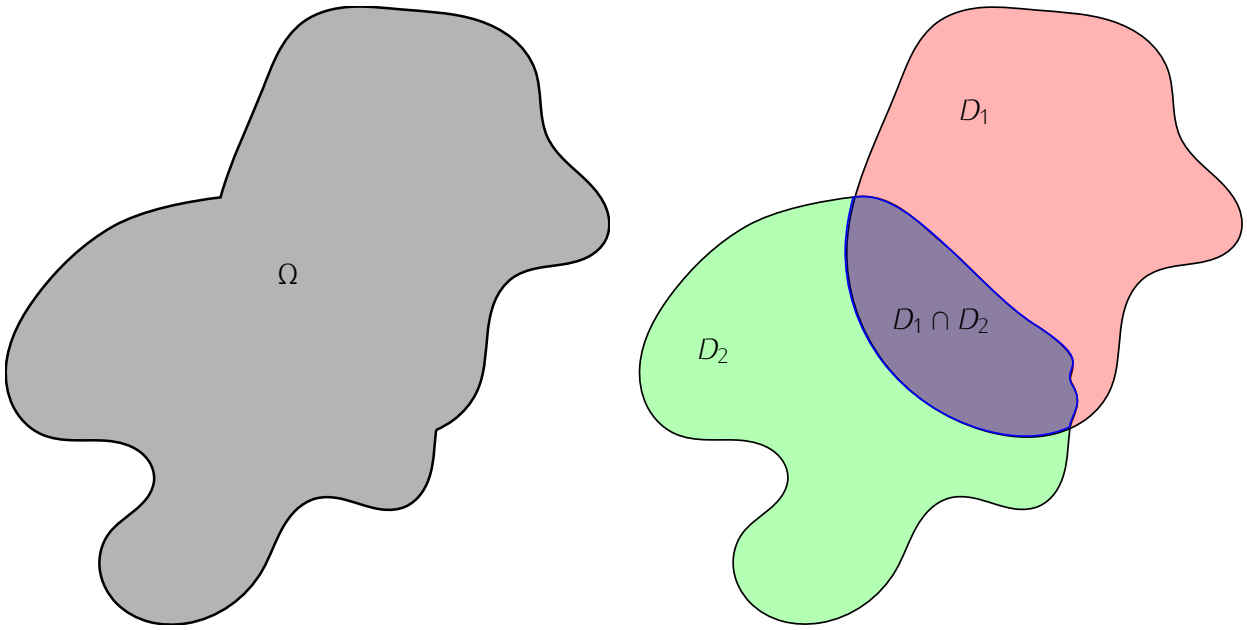


Figure 5 – Exemple de décomposition avec recouvrement d'un domaine Ω quelconque.

La décomposition de domaine avec recouvrement apporte une solution à cet enjeu. Le but est de partitionner le domaine en créant des zones de recouvrement plus ou moins grandes appartenant à au moins 2 partitions. Ces zones de recouvrement, dont la taille est en pratique variable, constituent le cœur même de cette **harmonisation**. Elles sont des zones d'échanges de valeurs entre partitions et permettent une

transmission, une sorte de liaison de continuité, entre deux domaines. Il va sans dire que la résolution du problème à l'échelle du domaine initial Ω ne peut pas se faire en une seule fois.

Si notre domaine est partitionné en deux zones D_1 et D_2, comme dans la figure 5, la zone d'échanges doit avertir D_1 des valeurs calculées sur D_2 et inversement. Un procédé itératif se dégage alors, le but étant d'atteindre un état *stationnaire* de sorte que les valeurs reçues par D_1 sur D_2 ne *modifient plus* les valeurs des degrés de libertés de D_2 (et inversement). Nous venons de dégager un critère de *convergence*.

2.2.2 La méthode de Schwarz additif

La méthode de Schwarz additif (il existe d'autres méthodes qui ne seront pas développées ici) est la mécanique de la zone d'échange. C'est une méthode itérative dans le sens où elle demande de résoudre plusieurs fois l'équation sur chaque partition et d'échanger des valeurs. Elle est basée sur l'échange de conditions aux bords que nous qualifierons de **conditions aux bords virtuels** [3].
Plusieurs conditions aux bords peuvent être envoyées, la plus évidente étant celle de type Dirichlet. Si nous reprenons notre figure 5 et que nous notons le recouvrement $R = D_1 \cap D_2$, alors nous nous ramenons aux deux problèmes suivants

$$\begin{cases} -\Delta u_{[1]} = f_{[1]} & \text{dans } D_1 \\ u_{[1]} = g & \text{sur } \partial\Omega \cap \partial D_1 \\ u_{[1]} = u_{[2]} & \text{sur } \Gamma_1 := \partial R \cap D_2 \end{cases} \tag{2}$$

$$\begin{cases} -\Delta u_{[2]} = f_{[2]} & \text{dans } D_2 \\ u_{[2]} = g & \text{sur } \partial\Omega \cap \partial D_2 \\ u_{[2]} = u_{[1]} & \text{sur } \Gamma_2 := \partial R \cap D_1 \end{cases} \tag{3}$$

Nous obtenons un algorithme naïf de résolution

Algorithme 1 : Schwarz additif sur D_1

$u_{[2],\Gamma_1} = 0$;
Calculer $u_{[1],\partial\Omega}$;
tant que *critère* **faire**
  Imposer $u_{[1],\partial\Omega}$ et $u_{[2],\Gamma_1}$;
  Résoudre $-\Delta u_{[1]} = f_{[1]}$;
  Envoyer à $\boxed{2}$ $u_{[1],\Gamma_2}$;
  Recevoir de $\boxed{2}$ $u_{[2],\Gamma_1}$;
fin

Algorithme 1 : Schwarz additif sur D_2

$u_{[1],\Gamma_2} = 0$;
Calculer $u_{[2],\partial\Omega}$;
tant que *critère* **faire**
  Imposer $u_{[2],\partial\Omega}$ et $u_{[1],\Gamma_2}$;
  Résoudre $-\Delta u_{[2]} = f_{[2]}$;
  Envoyer à $\boxed{1}$ $u_{[2],\Gamma_1}$;
  Recevoir de $\boxed{1}$ $u_{[1],\Gamma_2}$;
fin

Une illustration de la méthode est donné en suivant

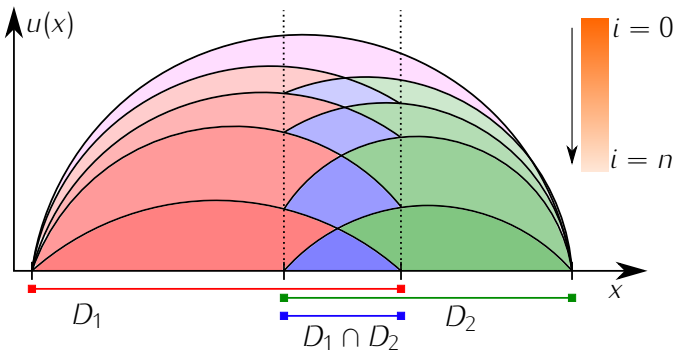


FIGURE 6 – Principe des itérations de la méthode de Schwarz additif en 1D (conditions de Dirichlet).

3. bords entre entre deux partitions en incluant la zone de recouvrement.

Échanger des conditions de Dirichlet fonctionne pour notre problème (1) mais nous pouvons remarquer qu'il est indépendant de l'équation résolue. Ce type de conditions aux bords fonctionnerait pour n'importe quelles équations.

2.2.3 Condition de type Robin

Théorie Regardons donc avec un peu plus de finesse ce qu'il se passe sur un bord virtuel $\Gamma \subset \Omega$ (certains préféreront sans doute employer ici la notion d'interface qui est assez parlante). Nous définissons D_1 et D_2 comme une partition de Ω au sens mathématique du terme, ie $\partial D_1 \cap \partial D_2 := \Gamma$. Nous considérons ici (1) après relèvement (ie $u = 0$ sur $\partial\Omega$). Soit $\varphi \in \mathcal{C}_c^\infty[\Omega]$ une distribution sur Ω, la formulation variationnelle est

$$-\langle \Delta u \mid \varphi \rangle_\Omega = \langle f \mid \varphi \rangle_\Omega$$

En notant $u_{[1]} := u$ sur D_1 et $u_{[2]} := u$ sur D_2, alors

$$\Longrightarrow -\left\langle \Delta u_{[1]} \mid \varphi \right\rangle_{D_1} - \left\langle \Delta u_{[2]} \mid \varphi \right\rangle_{D_2} = \langle f \mid \varphi \rangle_\Omega$$

En appliquant les formules de Green

$$\langle \Delta u \mid \varphi \rangle_\Omega = \langle \nabla u \cdot n \mid \varphi \rangle_{\partial\Omega} - \langle \nabla u \mid \nabla \varphi \rangle_\Omega \qquad \text{et} \qquad \langle \nabla u \mid \nabla \varphi \rangle_\Omega = \langle u \mid \nabla \varphi \cdot n \rangle_{\partial\Omega} - \langle u \mid \Delta \varphi \rangle_\Omega ,$$

nous obtenons

$$\begin{aligned} \Longrightarrow &-\left\langle \nabla u_{[1]} \cdot n_{D_1} \mid \varphi \right\rangle_{\partial D_1} + \left\langle u_{[1]} \mid \nabla \varphi \cdot n_{D_1} \right\rangle_{\partial D_1} - \left\langle u_{[1]} \mid \Delta \varphi \right\rangle_{D_1} \\ &-\left\langle \nabla u_{[2]} \cdot n_{D_2} \mid \varphi \right\rangle_{\partial D_2} + \left\langle u_{[2]} \mid \nabla \varphi \cdot n_{D_2} \right\rangle_{\partial D_2} - \left\langle u_{[2]} \mid \Delta \varphi \right\rangle_{D_2} = \langle f \mid \varphi \rangle_\Omega \end{aligned}$$

et comme $\varphi \equiv 0$ sur $\partial\Omega$ et en posant $n := n_{D_1} = -n_{D_2}$ sur Γ alors

$$\Longrightarrow -\left\langle \left(\nabla u_{[1]} - \nabla u_{[2]}\right) \cdot n \mid \varphi \right\rangle_\Gamma + \left\langle u_{[1]} - u_{[2]} \mid \nabla \varphi \cdot n \right\rangle_\Gamma - \left\langle u_{[1]} \mid \Delta \varphi \right\rangle_{D_1} - \left\langle u_{[2]} \mid \Delta \varphi \right\rangle_{D_2} = \langle f \mid \varphi \rangle_\Omega$$

Par définition du partitionnement nous souhaitons que

$$\langle u \mid \Delta \varphi \rangle_\Omega = \left\langle u_{[1]} \mid \Delta \varphi \right\rangle_{D_1} + \left\langle u_{[2]} \mid \Delta \varphi \right\rangle_{D_2}$$

(rappelons qu'ici il n'y a pas de recouvrement), donc si nous réappliquons deux fois les formules de Green sur Ω tout entier, alors

$$\left\langle -\left(\nabla_n u_{[1]} - \nabla_n u_{[2]}\right) \mid \varphi \right\rangle_\Gamma + \left\langle u_{[1]} - u_{[2]} \mid \nabla_n \varphi \right\rangle_\Gamma = \langle \Delta u + f \mid \varphi \rangle_\Omega \overset{\text{équation (1)}}{=} 0$$

En notant $[\kappa] := \kappa_{[2]} - \kappa_{[1]}$ le saut à l'interface pour une quantité κ, l'égalité précédente est exactement

$$\langle [\nabla_n u] \mid \varphi \rangle_\Gamma - \langle [u] \mid \nabla \varphi \cdot n \rangle_\Gamma = 0$$

Nous pouvons remarquer que si nous imposons un saut nul sur u et son gradient normal, ie $[u] = 0$ et $[\nabla_n u] = 0$, alors $\alpha[\nabla_n u] - \beta[u] = 0$, c'est-à-dire

$$\alpha \nabla_n u_{[1]} + \beta u_{[1]} = \alpha \nabla_n u_{[2]} + \beta u_{[2]}$$

Cette relation constitue une condition de bord virtuel dite de Robin.

Il est important de noter à ce stade qu'une condition de bord virtuel de type Dirichlet nécessite un recouvrement et non un simple partage de bord. En effet, imaginons que nos partitions D_1 et D_2 sont telles que $D_1 \cap D_2 = \Gamma$ soit un hyperplan de Ω (donc de mesure nulle avec la mesure sur Ω). Alors (2) devient

$$\begin{cases} -\Delta u_{[1]} = f_{[1]} & \text{dans } D_1 \\ u_{[1]} = g & \text{sur } \partial\Omega \cap \partial D_1 \\ u_{[1]} = u_{[2]} \equiv u_{[1]} & \text{sur } \Gamma \end{cases}$$

et nous pouvons remarquer que la condition de bord $u_{[1]}$ n'est plus mis-à-jour.

Discrétisation d'une condition de type Robin A contrario, une condition de type Robin ne semble pas demander une telle considération. Cela vient sans douter, de la façon qu'on peut imaginer imposer une condition de type Robin. Regardons donc la façon de discrétiser cette condition. Soit $n = \left[n_x, n_y\right]$ la normale sortante à D_1 sur le bord virtuel Γ, et g la valeur à imposer. Alors la condition de Robin est

$$\alpha \nabla_n u + \beta u = g \qquad\qquad \alpha, \beta \in \mathbb{R}. \tag{4}$$

Nous considérons ici l'exemple d'un bord virtuel *aligné* avec la grille de discrétisation, dans le code un bord non aligné avec la grille aura dans ses coins des conditions de type Dirichlet par défaut pour typiquement éviter les doubles conditions en un point :

$$\alpha n_x \partial_x u + \beta u = g_1 \qquad \alpha n_y \partial_y u + \beta u = g_2$$

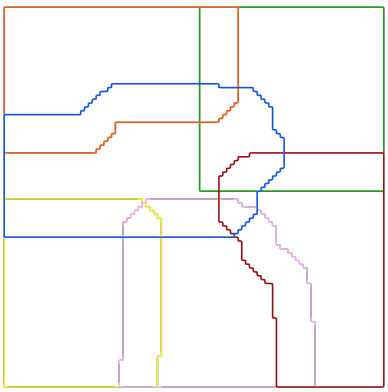
Nous rappelons que nous avons discrétisé l'équation avec un stencil de 5 points pour obtenir une erreur d'ordre 2. Il nous faut alors discrétiser la condition au bord avec le même ordre : nous utiliserons donc un schéma centré. Nous notons u_G le "ghost" point, u_{NG} le point "not-ghost" et u_H est le point où nous voulons imposer la condition g_H

$$\alpha \frac{u_G - u_{NG}}{2d} + \beta u_H = g_H \qquad \Longleftrightarrow \qquad u_G = u_{NG} - \frac{2d\beta}{\alpha} u_H + \frac{2d}{\alpha} g_H$$

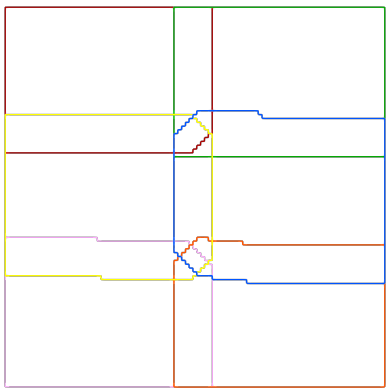
2.3 Le recouvrement en pratique

2.3.1 Les différentes techniques de partitionnement

Nous partons d'un domaine carré, précédemment maillé (en pratique le code nécessite un format de maillage ".lyra" généré par l'exécutable **generator-lyra**). Plusieurs techniques de partitionnement sont alors possibles en plus de la condition aux bords virtuels souhaité. Nous faisons appel à deux bibliothèque bien connues des partitionneurs : Metis [4] et Scotch [5]. Nous avons implémenté quatre autres méthodes : en colonnes, en lignes, en damier et par équilibrage de diagramme de Voronoï.

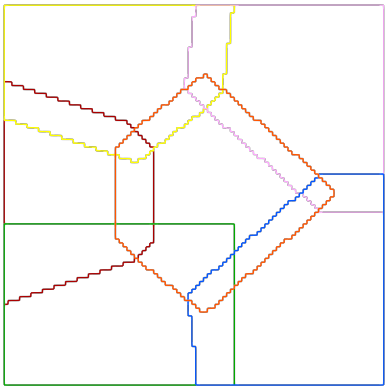


(a) Metis

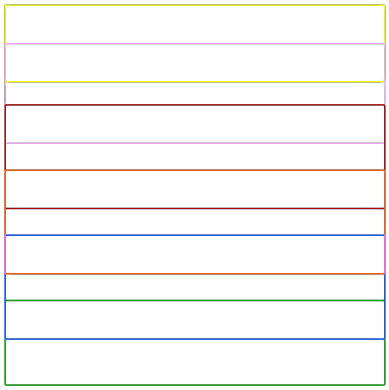


(b) Scotch

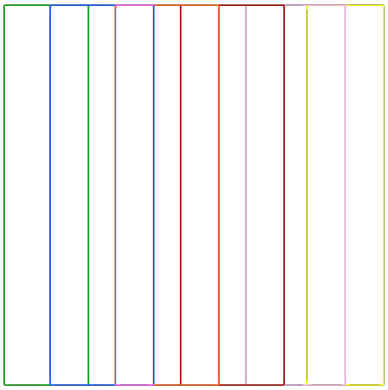
4. Metis : http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
5. Scotch : https://www.labri.fr/perso/pelegrin/scotch/



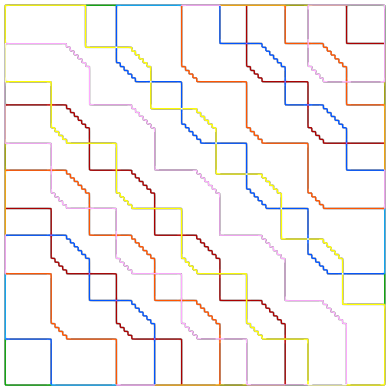
(c) Voronoï



(d) Row



(e) Column



(f) Board

FIGURE 7 – Les différents types de partitionnement, exemple 6 partitions, taille du recouvrement 5

En pratique le recouvrement est établi après partitionnement en trois étapes :

1. Nous partitionnons le graphe constitué des $n_x \times n_y$ points (n_x et n_y étant le nombre de points dans chaque direction sur le maillage initial). À ce stade chaque point appartient à une et une seule partition.

2. Un premier passage de la fonction d'agrandissement permet d'agrandir les partitions en les étendant sur les partitions voisines dont le numéro est plus grand. À l'issu de ceci toutes les partitions partagent leurs bords virtuels avec d'autres partitions.

3. Ensuite nous itérons la fonction d'agrandissement autant de fois que la taille de recouvrement souhaité : pour chaque point de chaque partition nous colorions les voisins de la couleur du processeur courant.

À l'issu de ces trois étapes nous obtenons des partitions de notre domaine Ω avec le recouvrement voulu. Par exemple, la figure 7 explicite les différents algorithmes de partitionnement pour une même taille de recouvrement. Nous pouvons pointer dès maintenant l'importance du partitionnement : autant du point de vue des communications que du nombres d'itérations nécessaires pour résoudre notre problème initiale (1) sous le critère dégagé précédemment.

Voronoï La méthode de partitionnement par équilibrage du diagramme de Voronoï est sans doute la moins claire des méthodes présentées. Nous avons implémenté l'équilibrage par déplacement des

générateurs (N_p générateurs pour N_p partitions) vers la moyenne de tous les points de la partition. Ensuite nous déduisons à quelle zone appartient le point par la recherche du générateur le plus proche. S'il y a un problème à souligner c'est bien celui-ci, cette recherche est faite à l'aide de la norme ℓ^2 et non en terme de distance d'arêtes. Si nous voulions améliorer cette méthode il faudrait changer ceci. Il est important de noter pour la suite que Metis utilise un algorithme de *k-mean* qui est basé sur l'algorithme de Lloyd, le cœur même de notre équilibrage.

2.3.2 Sans superposition de recouvrements

Nous nous intéressons maintenant à la mise en œuvre pratique d'un recouvrement et d'échanges de conditions aux bords virtuels. Dans le code nous avons choisi de faire des recouvrements de taille impaires dans chaque direction (ceci est une condition directe de l'algorithme de recouvrement présenté à la sous-section précédente). Nous illustrons l'échange de valeurs en une dimension par le schéma suivant

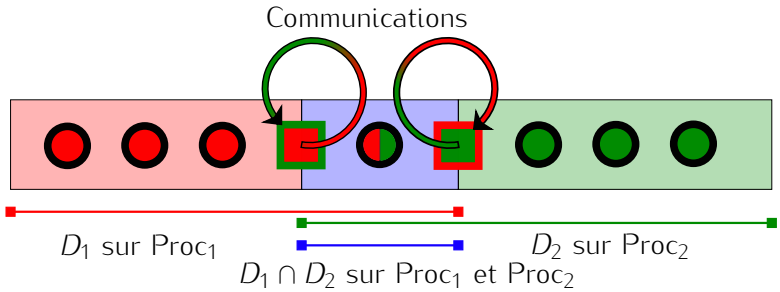


Figure 8 – Communication de conditions aux bords virtuels.

Ce schéma comporte des cercles : ce sont des points qui ne demandent pas de traitement particulier, leur(s) couleur(s) désigne le processeur auquel ils appartiennent. Par exemple, le point colorié en rouge et vert est **partagé** par les deux processeurs rouge et vert. Le schéma comporte aussi deux carrés : ce sont des points d'échanges. Les couleurs désigne aussi les processeurs auxquels appartient le point. La couleur intérieure désigne la couleur du processeur qui *envoie* la condition au bord et la couleur extérieure désigne la couleur du processeur qui *reçoit* la condition au bord. Nous faisons un récapitulatif sur le graphe suivant, c'est la version simplifiée de ce qui se passe dans le code mais l'idée reste la même.

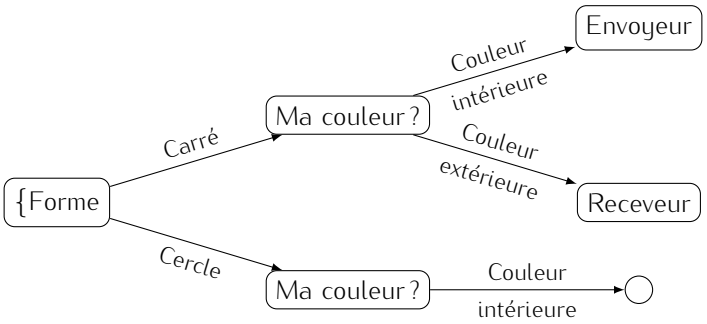


Figure 9 – Arbre de catégorisation sans superposition de recouvrement.

2.3.3 Avec superposition de recouvrements

Un cas pathologique est apparu au cours de nos tests numériques et de nos différentes techniques de partitionnement. Que faire quand au moins deux processeurs tombent dans la case "envoyeur" de l'arbre de décision précédent pour un même point ? L'exemple est illustré sur la figure ci-après

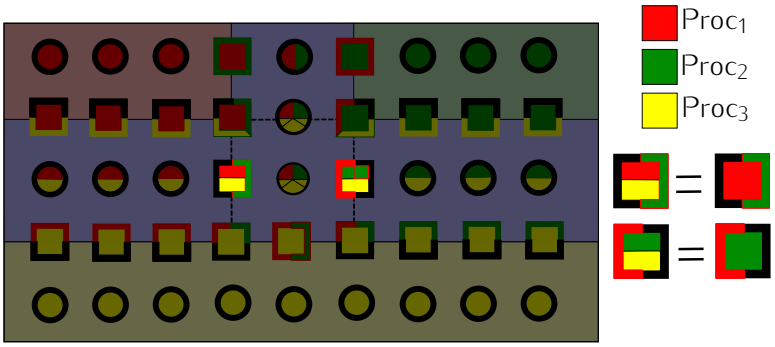


FIGURE 10 – La règle du plus jeune envoyeur.

La règle la plus simple à mettre en œuvre nécéssite un ordre entre les différentes partitions et donc entre les différents numéros de processeurs pour être valide. La règle, intuitive, qui en découle, que nous nommerons la règle du plus jeune envoyeur est la suivante

Je suis un ***envoyeur*** *si aucun des processeurs avant moi n'est un* ***envoyeur***. (5)

L'arbre de décision est modifié en conséquence

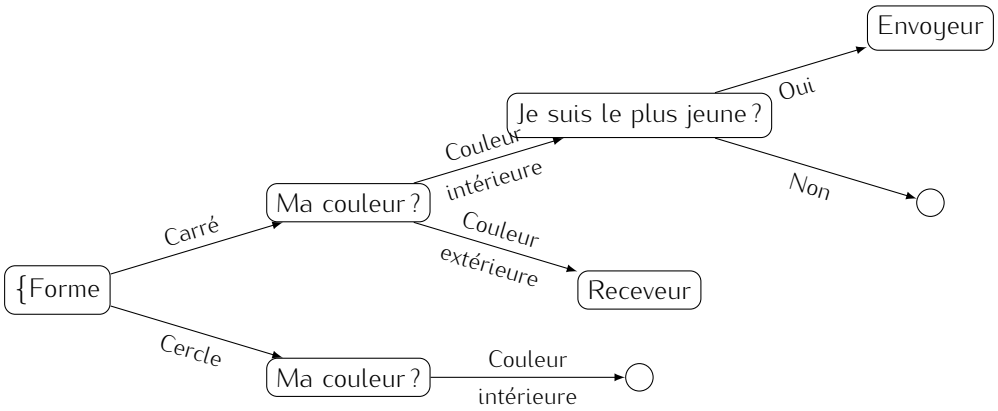


FIGURE 11 – Arbre de catégorisation version "plus jeune envoyeur".

Cette technique, bien que simpliste, permet de définir un **unique processeur envoyeur** et une liste de **processeurs receveurs** par point. Nous rappelons ici qu'une condition de type Robin dans un coin de la partition est automatiquement transformé en condition de Dirichlet (voir la section précédente).

2.3.4 Le format ".lyra"

Nous présentons ici le format de maillage (aisément modifiable) dont nous avons précédemment parlé et qui est la passerelle entre les différents exécutables de la bibliothèque **Lyra**.

```
number-of-points
coor-x coor-y coor-z global-num tag tag-infos \\ for each point
...
number-of-cells
vertex-1 vertex-2 vertex-3 vertex-4 \\ for each cell
...
```

TABLE 1 – Le format ".lyra"

où

- `coor-x coor-y coor-z` sont les coordonnées du point
- `global-num` est le numéro global du point sur le maillage initial (avant partitionnement), ce numéro est essentiel car il servira dans l'ordre des réceptions/envois avec MPI [6]
- `tag` est un nombre au format binaire représentant tous les tags du point. Voici les tags possibles

Tag	Valeur	Digit	Sur cette partition, je suis
PT_NONE	0x0	1	un simple point
PT_SHARED	0x1	1	partagé avec une autre partition
PT_PHYSICAL	0x2	2	un point de bord physique
PT_VIRTUAL	0x4	3	un point de bord virtuel
PT_SEND	0x8	4	un point envoyeur
PT_RECEIVE	0x10	5	un point receveur
PT_DIRICHLET	0x20	6	un point où il faut imposer une condition de type Dirichlet
PT_NEUMANN	0x40	7	un point où il faut imposer une condition de type Neumann
PT_ROBIN = PT_DIRICHLET | PT_NEUMANN	0x60	6 & 7	un point où il faut imposer une condition de type Robin

TABLE 2 – L'énumération `PTAG`

Par exemple, le tag `1101101` signifie `PT_ROBIN | PT_SEND | PT_VIRTUAL | PT_SHARED`.

- `tag-infos` sont les informations additionnelles nécessaires aux communications, elles sont donc uniquement déterminé par les tags `PT_SEND` et `PT_RECEIVE` :
  - Si `PT_SEND` alors il faut `number-of-receivers [id-proc dir-normal-outward]...` où `dir-normal-outward` vaut

    ```
    {LEFT = 0, RIGHT = 1, UP = 2, BOTTOM = 4, NOT_SET = 5}
    ```

    et sera utilisé pour la condition de type Robin.
  - Si `PT_RECEIVE` alors il faut uniquement `id-proc-sender`

2.4 Résultats

Nous présentons dans cette section divers résultats obtenus. Dans un premier temps rappelons les deux définitions utilisées en calcul parallèle pour déterminer la *performance* d'un code parallélisé par rapport au code séquentiel. Nous introduisons les notations time_1 désignant le temps série et time_{N_p} le temps parallèle en utilisant N_p processus.

$$\text{Speed-up}\left(N_p\right) = \frac{\text{charge totale}}{\max\limits_{\text{procs}\in N_p} \text{charge}} \approx \frac{\text{time}_1}{\text{time}_{N_p}} \leq N_p$$

$$\text{Efficacité}\left(N_p\right) = \frac{\text{Speed-up}\left(N_p\right)}{N_p} \approx \frac{\text{time}_1}{N_p \times \text{time}_{N_p}} \leq 1$$

où N_p est le nombre de processeurs utilisés. Nous ferons toujours la distinction entre la définition théorique et le résultat obtenu en pratique.
Nous introduisons aussi ce que nous appellerons le speed-up et l'efficacité normalisés à α processus

6. MPI : https://www.mpich.org/

comme :

$$\text{Speed-up-}\alpha\left(N_p\right) = \frac{\alpha \times \text{Speed-up}\left(\alpha\right)}{\text{Speed-up}\left(N_p\right)} \approx \frac{\alpha \times \text{time}_\alpha}{\text{time}_{N_p}} \leq N_p$$

$$\text{Efficacité-}\alpha\left(N_p\right) = \frac{\text{Speed-up-}\alpha\left(N_p\right)}{N_p} \approx \frac{\alpha \times \text{time}_\alpha}{N_p \times \text{time}_{N_p}} \leq 1$$

Par ailleurs, les différentes simulations ont été effectuées sur PlaFRIM avec différents scripts. Le solveur Lyra a été configuré comme suit

- le gradient conjugué et le bigradient conjugué stabilisé à une tolérance de 10^{-10},
- les itérations de Schwarz s'arrêtent lorsque le maximum de la différence entre deux solutions numériques est de 10^{-3}. Cette valeur est réduite et sommée sur chaque partition.

2.4.1 L'effet du partitionnement

Avant de comparer plus en profondeur le speed-up et l'efficacité de notre code, nous proposons de regarder succinctement l'effet des différents choix que nous avons fait lors du partitionnement.

La règle du plus jeune envoyeur (5) n'est pas étudié ici même si nous sommes conscient que cette **règle** est purement empirique et qu'il se peut qu'un choix plus stratégique se dégage au cours d'explorations : cela dépendrait certainement de la taille du recouvrement et du problème étudié.

La conversion de Robin à Dirichlet concernant les coins a un impact non négligeable au vu de nos simulations (l'effet Dirichlet-Robin est expliqué dans la suite). Regardons donc le taux de conversion selon la méthode de partitionnement choisi. Pour cet exemple nous avons utilisé un maillage de 200×200 que nous avons partitionné avec plusieurs méthodes avec une taille de recouvrement de 5 mailles (à multiplier par deux selon l'algorithme établi précédemment). Il est important de préciser qu'à partir de 20 partitions nombre de méthodes produisent des domaines entièrement partagés : par exemple une division en 20 lignes produit des partitions avant recouvrement de taille 10 et il est inévitable que le recouvrement à droite rejoigne celui de gauche avec notre taille prescrite.

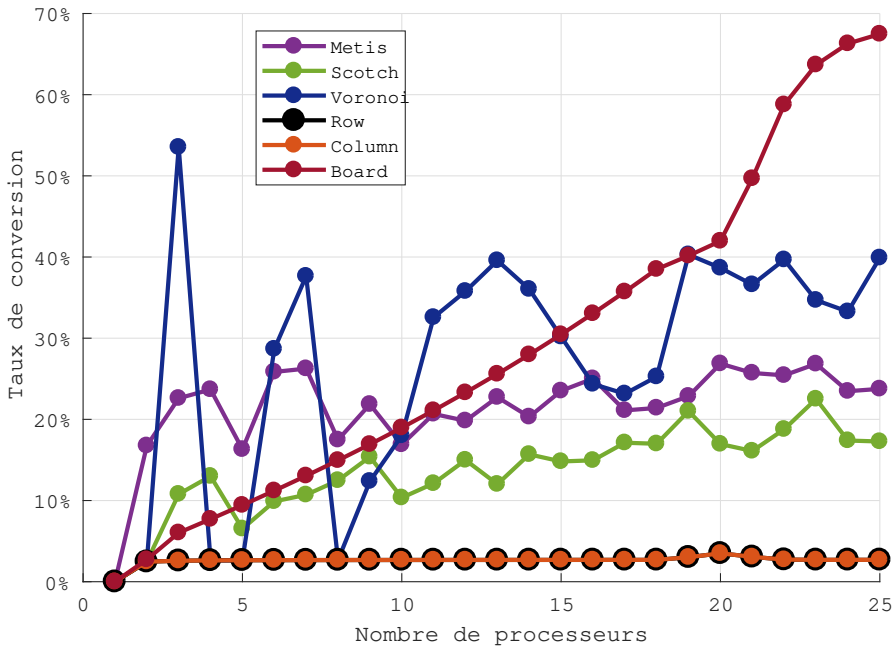


FIGURE 12 – Taux de conversion de Robin vers Dirichlet.

Plusieurs observations sont à faire ici :

- le partitionnement en lignes ou colonnes semble être la méthode qui transforme le moins de Robin en Dirichlet et cela même quand le nombre de partitions augmente : ceci s'explique car ces méthodes suivent la discrétisation (topographie) du maillage, nous dirons ici que *le partitionnement est fidèle au maillage*.
- Scotch semble donner des partitions plus fidèle que Metis même si pour ces deux méthodes nous restons en dessous de la barre des 30% ce qui est compréhensible car autant l'un que l'autre ont une **véritable** implémentation de partitionnement de graphe avec des méthodes qui ont fait leurs preuves.
- le partitionnement selon un l'uniformisation diagramme de Voronoï semble être la plus aléatoire et cela s'explique tout naturellement par rapport au *nombre de partitions demandées*. Nous pourrions aller chercher du côté des carrés parfaits pour trouver une explication plus théorique.
- Le partitionnement en damier semble être linéaire jusqu'à atteindre le nombre de partitions fatidique exposé précédemment. Au cours de nos tests nous avons remarqué que cette droite semble être de moins en moins pentue à mesure que le nombre de points de discrétisation augmente, cette méthode semble être un bon compromis à celle de Voronoï par exemple.

Regardons maintenant toutes ces méthodes d'un point de vue plus informatique : la taille des messages à faire transiter (le nombre de communications nécessaires). En effet le temps de communication (a fortiori la taille des messages) est une donnée importante à prendre en compte. Cet taille correspond en pratique au nombre de points définissant les interfaces entre les partitions.

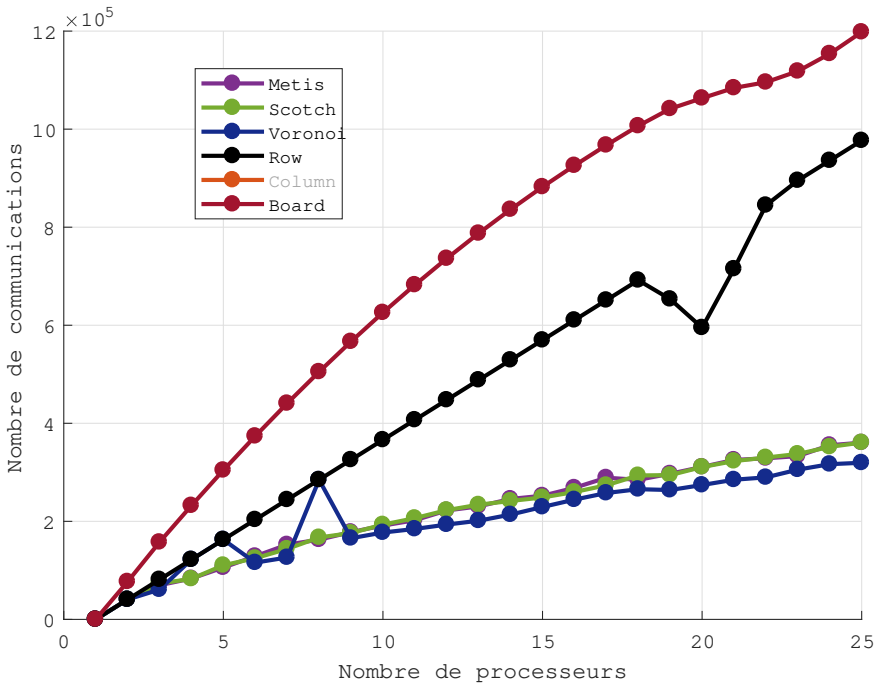


Figure 13 – Nombre de communications nécessaires à une itération de Schwarz maillage 200 × 200.

La méthode de Voronoï est ici plus que surprenante : elle demande pratiquement autant de communication qu'un partitionnement fidèle au maillage. Le partitionnement en damier est sans conteste la méthode qui demande le plus de communications, tandis que partitionner en ligne ou colonne semble moins favorable tout à coup.
Au vu des deux comparaisons il semble qu'il vaille mieux partitionner avec Metis ou Scotch. Il nous faut maintenant valider toutes ces remarques à l'aide des indicateurs de performances établis en début de section.

2.4.2 Résolution 300×300, recouvrement de taille 5

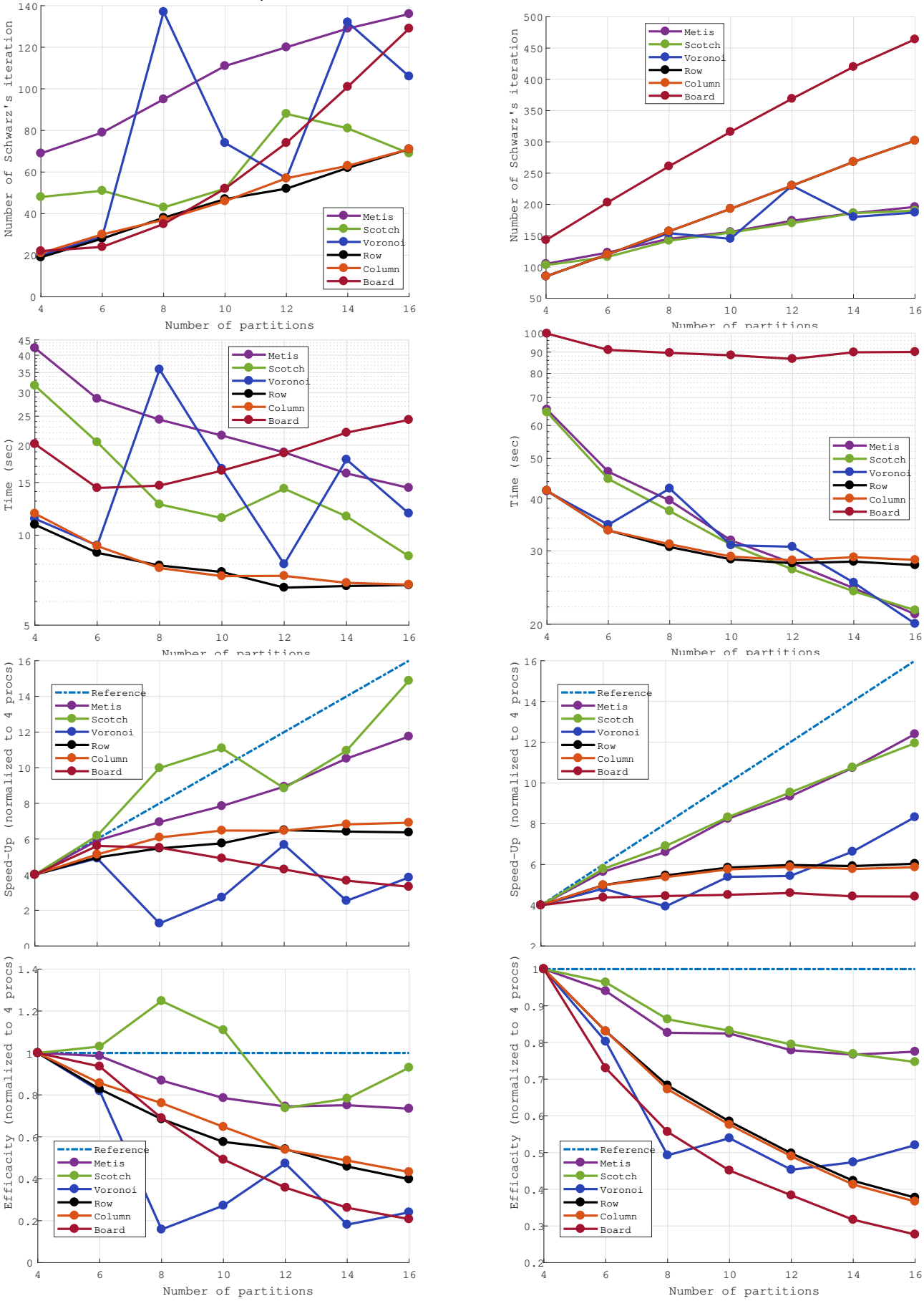
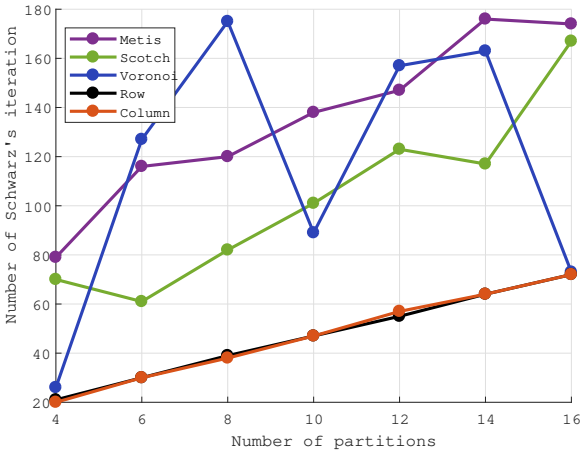
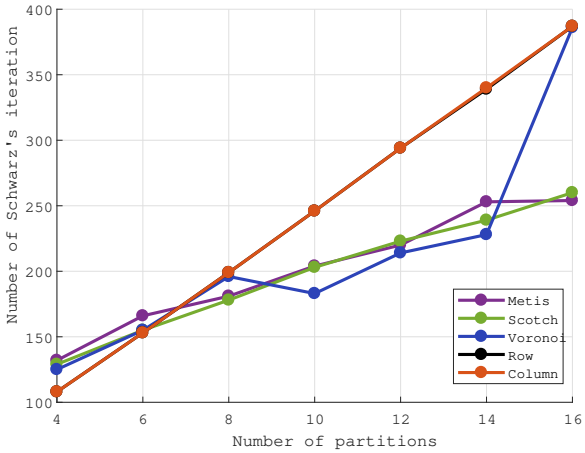


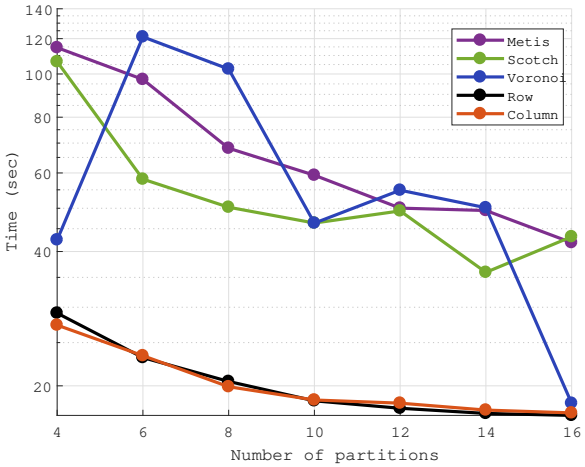
FIGURE 14 – Analyse du 300×300 avec un recouvrement de taille 5. À droite Robin et à gauche Dirichlet.

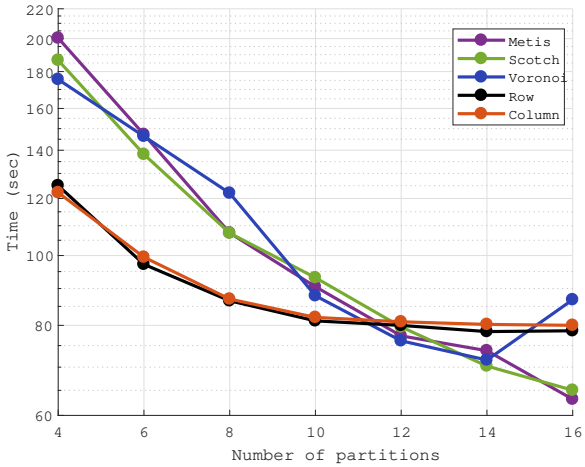
Dans cet exemple de résolution nous avons utilisé tous les types de partitionnement disponibles. La méthode d'équilibrage type Voronoï semble celle qui donne les résultats les plus aléatoires pour des conditions de Robin, tandis que le partitionnement en damier est celui qui prend le plus de temps et le plus d'itérations avec des conditions de Dirichlet. La comparaison entre les conditions aux bords virtuels de Dirichlet et Robin est sans appel : utiliser les conditions de Robin divise par deux les temps de calcul globalement. Dans tous les cas, ce sont Metis et Scotch qui se distinguent le plus en terme d'efficacité ; Scotch allant jusqu'à une efficacité de 120% pour des conditions de Robin !
Regardons maintenant ce que nos tests donnent lorsque nous augmentons le nombre de points sur le domaine (ie la charge pour chaque partition, presque deux fois plus).

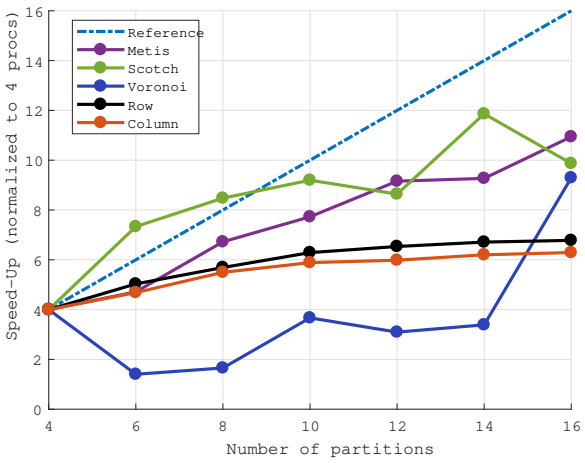
2.4.3 Résolution 400×400, recouvrement de taille 5

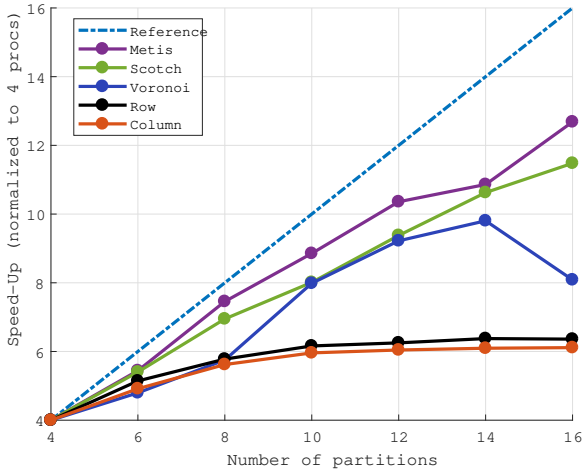












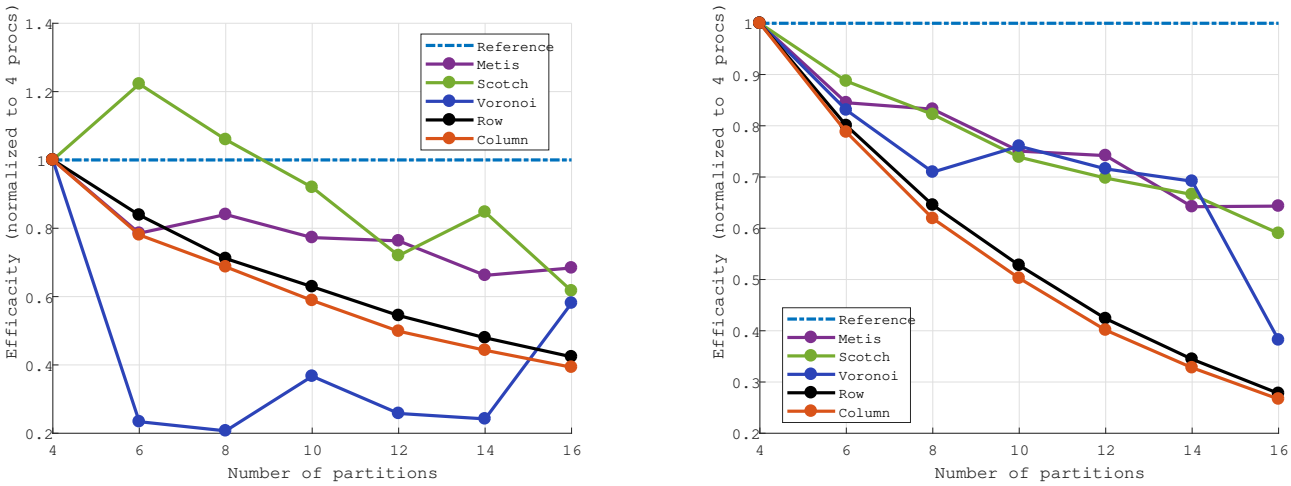
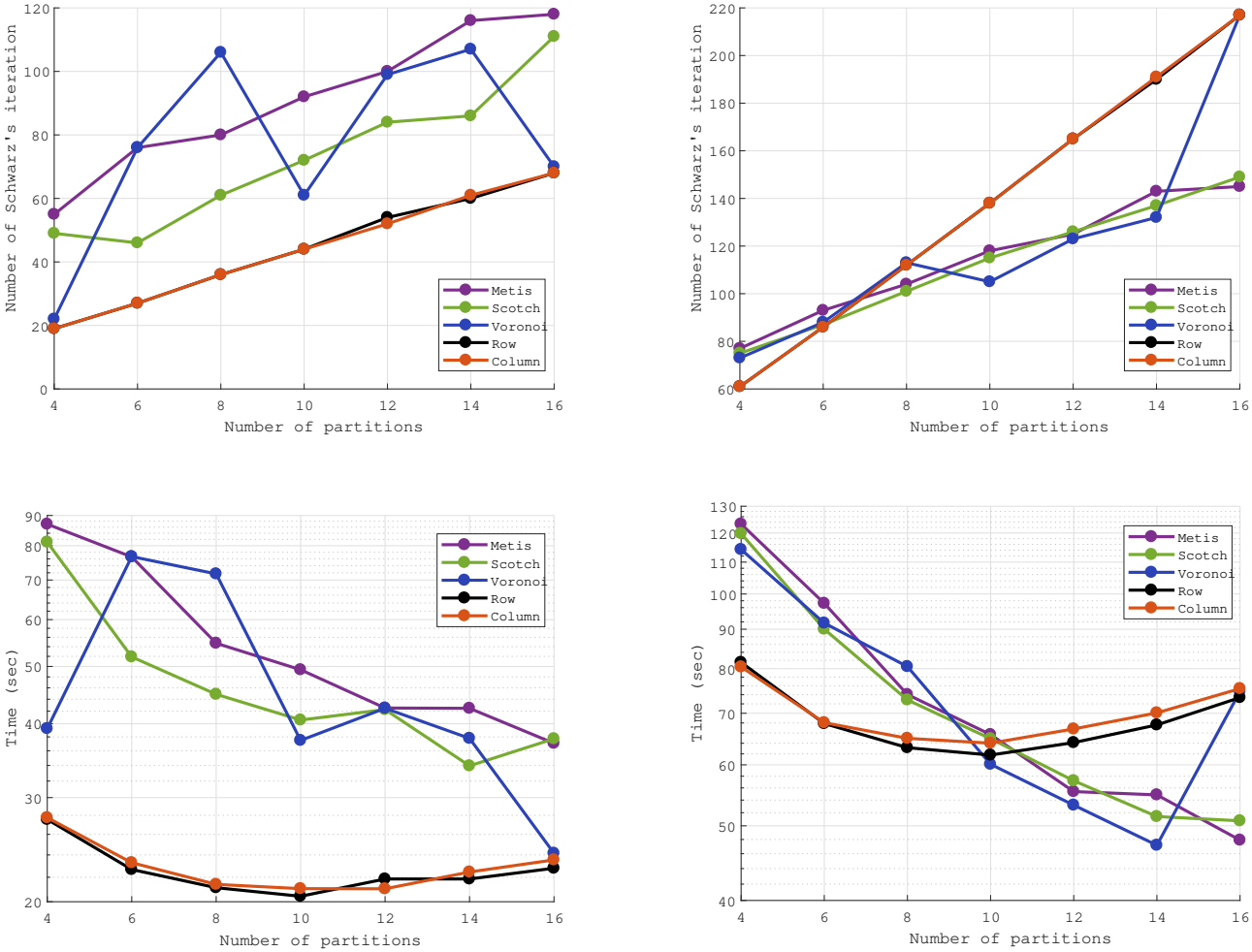


FIGURE 15 – Analyse du cas 400 × 400 avec un recouvrement de taille 5. À droite Robin et à gauche Dirichlet.

Dans ces tests nous avons volontairement omis le partitionnement en damier qui induit des temps de calcul vraiment plus longs comme nous l'avons montré dans la sous-section précédente. Cette partie mets en évidence l'importance de la taille du recouvrement rapportée à la taille totale. Globalement tous les temps de calculs sont augmentés et le nombre d'itérations aussi. Tout le reste est sensiblement pareil que dans l'exemple précédent.

2.4.4 Résolution 400 × 400, recouvrement de taille 10



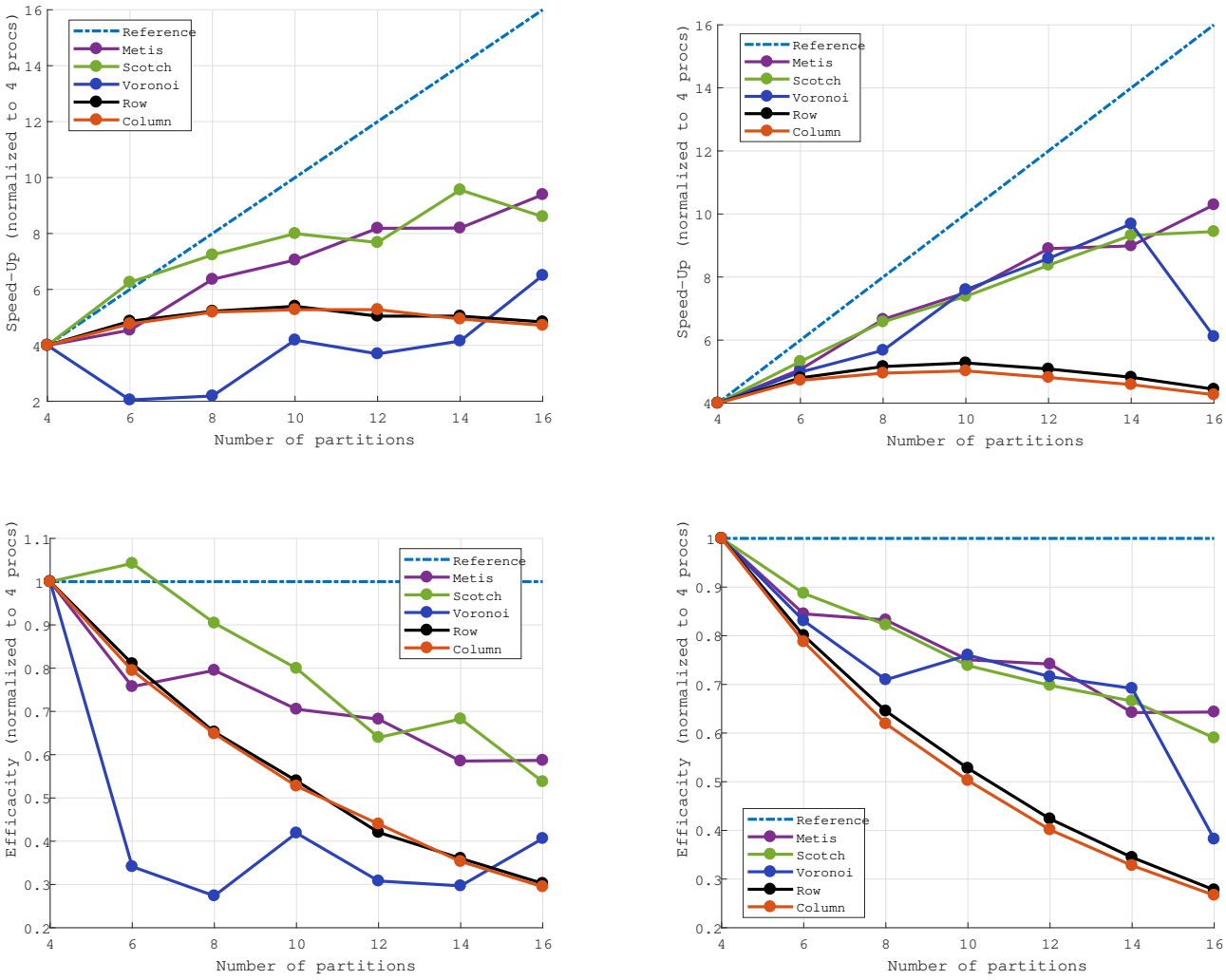


FIGURE 16 – Analyse du cas 400 × 400 avec un recouvrement de taille 10. À droite Robin et à gauche Dirichlet.

Cette partie porte sur la taille du recouvrement, les principales différences à pointer du doigt sont les temps de calculs et le nombre d'itérations nécessaires qui diminuent fortement.

2.4.5 Erreur et solution numérique : 200 × 200 - recouvrement 5 - 4 partitions avec Metis

Nous proposons quelques rendus d'erreur pour le problème (⋆⋆). Pour se faire nous avons utilisé un maillage 200 × 200 partitionné en 4 avec Metis, un recouvrement de taille 5 et des conditions de Robin.

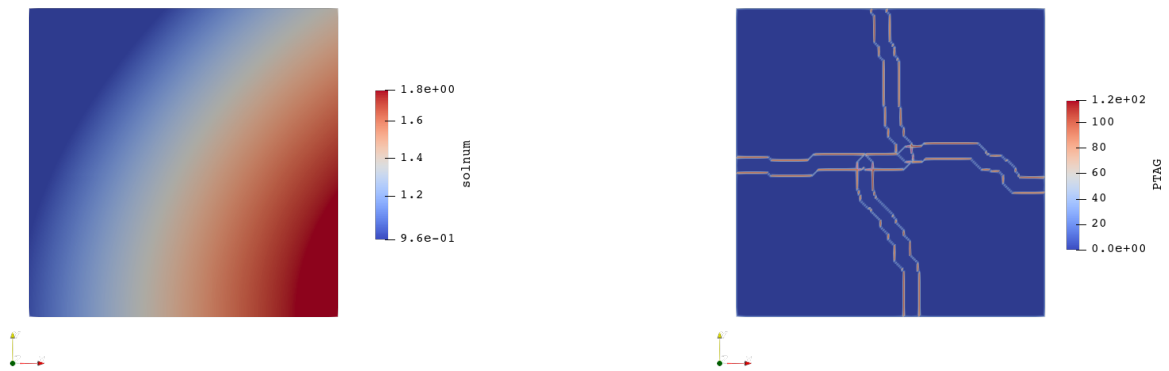


FIGURE 17 – Solution analytique à gauche et tags de partitionnement à droite.

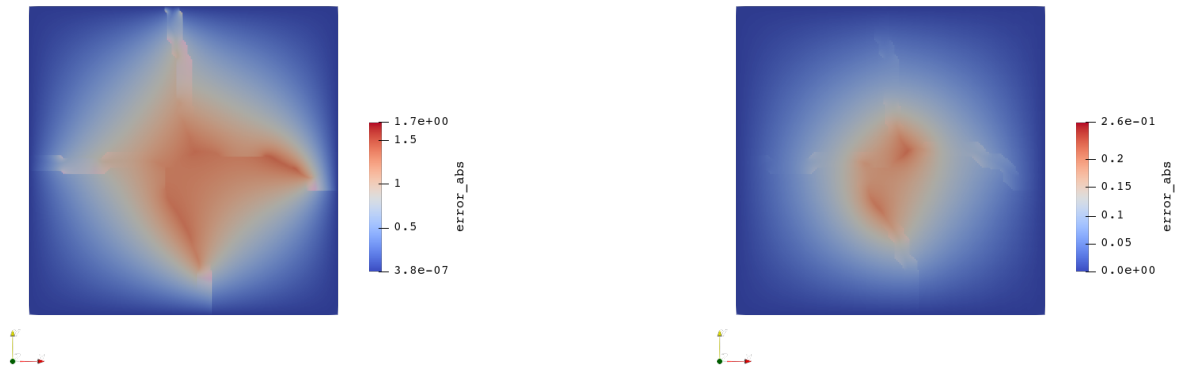


FIGURE 18 – Évolution de l'erreur en valeur absolue point à point aux itérations 1 et 11 sur 37.

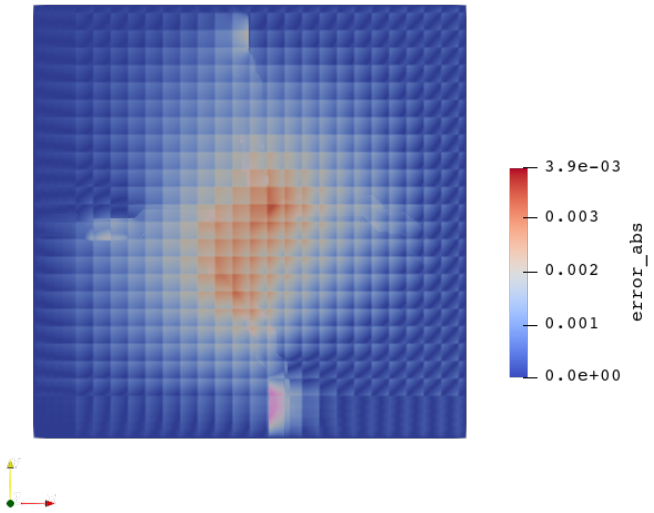


FIGURE 19 – Erreur en valeur absolue point à point à l'itération 30 sur 37.

L'erreur décroît bien au cours des itérations de Schwarz, donc la solution numérique converge petit à petit vers la solution analytique. La seule remarque à faire ici est sur le comportement visuel de l'erreur à l'itération 30 sur 37 : une grille semble apparaître et elle ne suit pas le maillage de base et encore moins les partitions. Nous ne sommes pas capable malheureusement d'expliquer le phénomène bien que celui-ci soit présent pour tous les types de partitionnement que nous avons testé. Un début d'explication peut être néanmoins tenté : il se peut que ce soit un comportement inhérent à notre solveur bigradient conjugué stabilisé mais nous n'en savons pas plus.

2.5 Conclusion

Pour conclure cette partie nous mettrons en lumière l'intérêt tout particulier des partitionnements Metis et Scotch. Ils sont un bon compromis selon tous nos tests lorsque que nous les allions à des conditions de Robin, autant du point de vue de la conversion Robin vers Dirichlet, que du temps de calcul ou de l'efficacité.
Implémenter la suite de code Lyra nous a permis de réfléchir plus en profondeur sur les différentes méthodes de partitionnement, sur MPI en lui-même et sur l'utilisation de PlaFRIM.
Il reste nombre de points à approfondir, notamment l'utilisation de communications non-bloquantes.

Table des figures