TECHNIQUES
DE L'INGÉNIEUR

Réf. : **AF1376 V1**

Méthodes de décomposition de domaines - Extensions

Date de publication :
10 avril 2012

Cet article est issu de : **Sciences fondamentales | Mathématiques**

par **Martin J. GANDER, Laurence HALPERN**

Méthodes de décomposition de domaines

Extensions

par **Martin J. GANDER**
Professeur de mathématiques
Section de Mathématiques, Université de Genève

et **Laurence HALPERN**
Professeur de Mathématiques
Laboratoire Analyse, Géométrie et Applications, Université Paris 13

Il s'agit ici de la seconde partie de l'article Méthodes de décomposition de domaines. Notions de base.

1. Préconditionneur grille grossière

Pour exposer les différentes méthodes dans un cadre simple, nous avons considéré dans les sections précédentes des décompositions en deux sous-domaines. En réalité les ordinateurs parallèles ont des milliers, voire même des centaines de milliers de processeurs, et il faut décomposer les domaines en autant de sous-domaines pour bien paralléliser le processus de résolution. Mais toutes les méthodes itératives que nous avons vues perdent de leur efficacité lorsque le nombre de sous-domaines augmente. On dit que ces méthodes ne sont pas **scalables**. Il faut ajouter à toutes ces méthodes une nouvelle composante pour obtenir des méthodes scalables, et ceci est le sujet de cette section.

1.1 Problèmes de scalabilité

Il y a deux types de scalabilité pour un algorithme : la scalabilité forte et la scalabilité faible.

Scalabilité forte : pour un problème de taille fixée, le temps d'exécution est inversement proportionnel au nombre de processeurs.

Scalabilité faible : le temps d'exécution ne varie pas lorsque l'on augmente la taille du problème et le nombre de processeurs dans la même proportion.

Dans le cadre des méthodes de décomposition de domaines, la scalabilité est mesurée par rapport au nombre de sous-domaines. Le domaine $\Omega = [0, 1]$ est décomposé en I sous-domaines $\Omega_i = [\alpha_i, \beta_i]$ de taille H_i, avec recouvrement $\delta_i = \beta_i - \alpha_{i+1} > 0$, voir figure **1**. Nous supposerons que seuls deux sous-domaines successifs se touchent, c'est-à-dire que pour tout i, $\beta_{i-1} < \alpha_{i+1}$.

Étudions numériquement sur le problème modèle monodimensionnel avec $\eta = 0$ la scalabilité de la méthode de Schwarz parallèle. Pour l'équation $\partial_{xx}u = 0$ avec des conditions de Dirichlet sur les bords $u(0) = g_g$ et $u(1) = g_d$, l'algorithme de Schwarz parallèle calcule ainsi à chaque étape $n = 1, 2,\ldots$ une fonction u_i^n dans le domaine i par l'algorithme

$$\begin{aligned} &\partial_{xx}u_i^n = 0 \quad \text{dans } \Omega_i, \\ &u_i^n(\alpha_i) = u_{i-1}^{n-1}(\alpha_i), \quad u_i^n(\beta_i) = u_{i+1}^{n-1}(\beta_i), \end{aligned} \tag{1}$$

avec les données au bord physiques $u_1^n(\alpha_1) = g_g$ et $u_I^n(\beta_I) = g_d$.

Nous illustrons les performances de cet algorithme par rapport au nombre de sous-domaines sur la figure **2** où sont tracées les courbes de convergence pour 2, 4, 8 et 16 sous-domaines. Les tests seront les mêmes tout au long de cette section.

Cas test

Sur les deux figures du haut, nous étudions la scalabilité forte : la taille du problème global est fixée à $J_0 = 511$, et nous faisons varier le nombre de sous-domaines : 2, 4, 8, 16 sous-domaines.
Sur les deux figures du bas, nous étudions la scalabilité faible : lorsque le nombre de sous-domaines augmente (2, 4, 8, 16), la taille du problème global augmente en proportion (J = 511, 1023, 2047, 4095).
Sur les figures de gauche le recouvrement entre les sous-domaines est constant égal à $h_0 = 20/(J_0 + 1)$.
Sur les figures de droite le recouvrement diminue en proportion du nombre de sous-domaines.

Dans tous les cas, il apparaît clairement que la convergence se dégrade lorsque l'on augmente le nombre de sous-domaines. Puisque le but de la décomposition en sous-domaines est, soit d'obtenir une solution plus rapidement, soit de résoudre un problème de plus grande taille, on peut dire que ce but n'est pas atteint : en augmentant le nombre de sous-domaines, le nombre d'itérations augmente, le temps de calcul devient de plus en plus grand, ce qui rend la méthode inutilisable pour le calcul parallèle à grande échelle.

1.2 Explications intuitives et mathématiques

Pourquoi la méthode ralentit-elle lorsque l'on ajoute des sous-domaines ? L'explication est visible sur la figure **3**, où nous montrons les premières itérations d'une méthode de Schwarz parallèle pour le cas de deux sous-domaines sur la première ligne de graphiques, et 16 sous-domaines sur la deuxième ligne. La solution exacte du problème est la ligne droite. La donnée initiale de l'algorithme est 0 sur tous les bords des sous-domaines, sauf aux bords extérieurs. Dans le cas de deux sous-domaines, la deuxième itération commence à s'approcher de la solution cherchée sur les deux sous-domaines. Il en va tout autrement dans le cas de seize sous-domaines, où même après six itérations, les sous-domaines trop éloignés de la condition au bord droit ont encore comme approximation zéro, la donnée initiale choisie pour cette exemple. La transmission entre les sous-domaines étant locale, la condition au bord droit ne pourra être transmise au premier sous-domaine qu'après seize itérations.

Nous faisons maintenant l'analyse de cet exemple simple. De nouveau $e_i^n = u_i^n - u$ est l'erreur dans le domaine Ω_i à l'itération n. Elle est donnée explicitement par

$$e_i^n = \frac{1}{\beta_i - \alpha_i}\left(e_{i+1}^{n-1}(\beta_i)(x - \alpha_i) + e_{i-1}^{n-1}(\alpha_i)(\beta_i - x)\right).$$

Introduisons les coefficients

$$a_i^- = \frac{\beta_i - \beta_{i-1}}{\beta_i - \alpha_i},\ b_i^- = \frac{\beta_{i-1} - \alpha_i}{\beta_i - \alpha_i},\ a_i^+ = \frac{\beta_i - \alpha_{i+1}}{\beta_i - \alpha_i},\ b_i^+ = \frac{\alpha_{i+1} - \alpha_i}{\beta_i - \alpha_i},$$

qui s'écrivent au moyen des tailles caractéristiques de la décomposition

$$a_i^+ = \frac{\delta_i}{H_i},\ b_i^- = \frac{\delta_{i-1}}{H_i},\quad a_i^- = 1 - b_i^-,\ b_i^+ = 1 - a_i^+.$$

Nous écrivons alors les relations de récurrence

$$\begin{aligned} e_i^n(\beta_{i-1}) &= a_i^- e_{i-1}^{n-1}(\alpha_i) + b_i^- e_{i+1}^{n-1}(\beta_i), \\ e_i^n(\alpha_{i+1}) &= a_i^+ e_{i-1}^{n-1}(\alpha_i) + b_i^+ e_{i+1}^{n-1}(\beta_i). \end{aligned}$$

Elles se traduisent en un grand système portant sur les valeurs des erreurs à l'interface

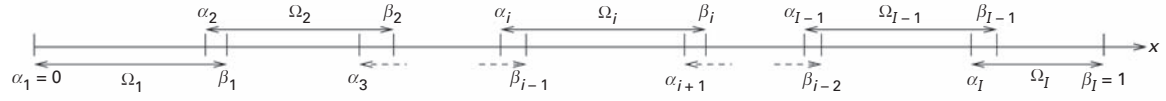
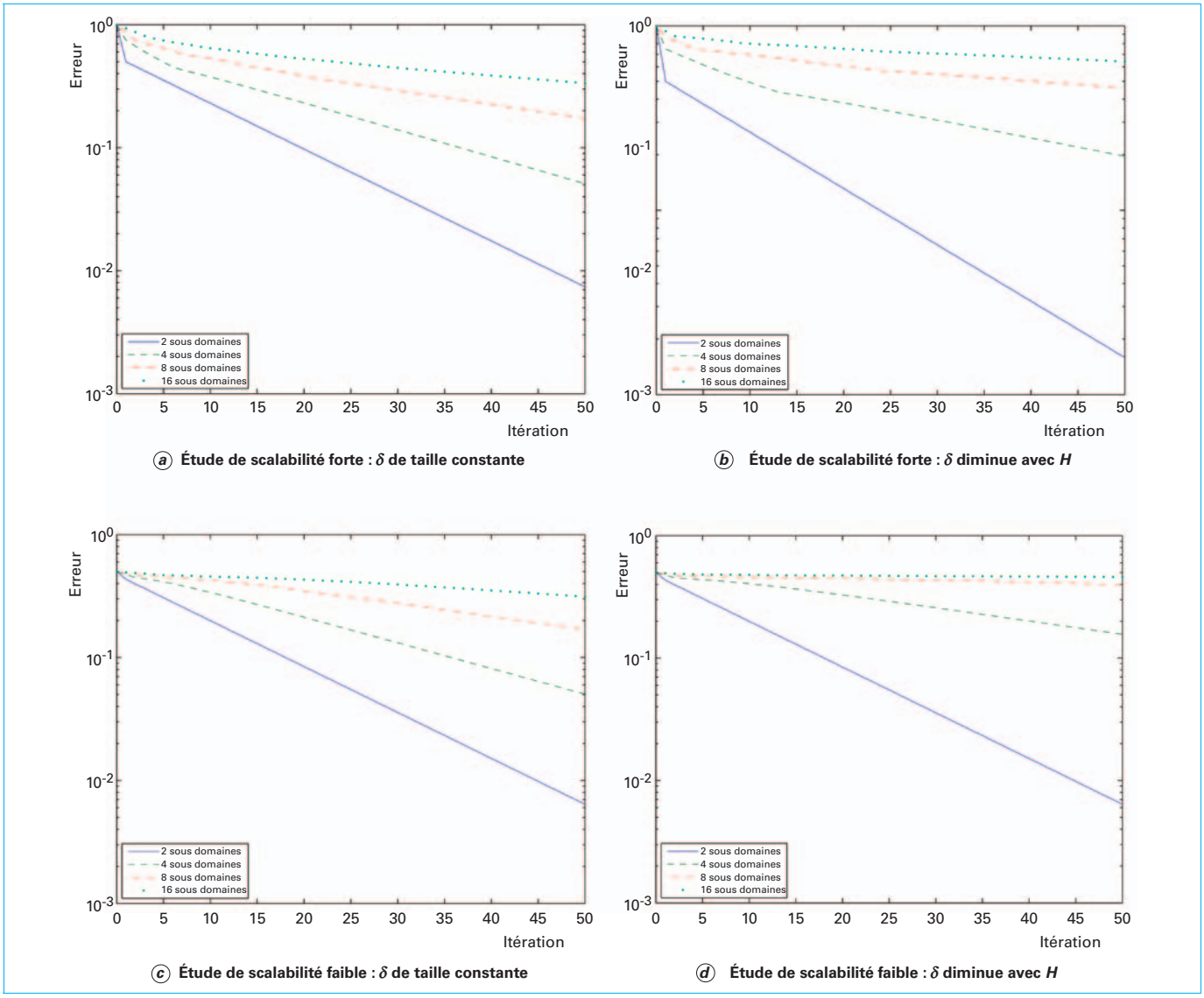


Figure 1 – Décomposition en sous-domaines



ⓐ Étude de scalabilité forte : δ de taille constante

ⓑ Étude de scalabilité forte : δ diminue avec H

ⓒ Étude de scalabilité faible : δ de taille constante

ⓓ Étude de scalabilité faible : δ diminue avec H

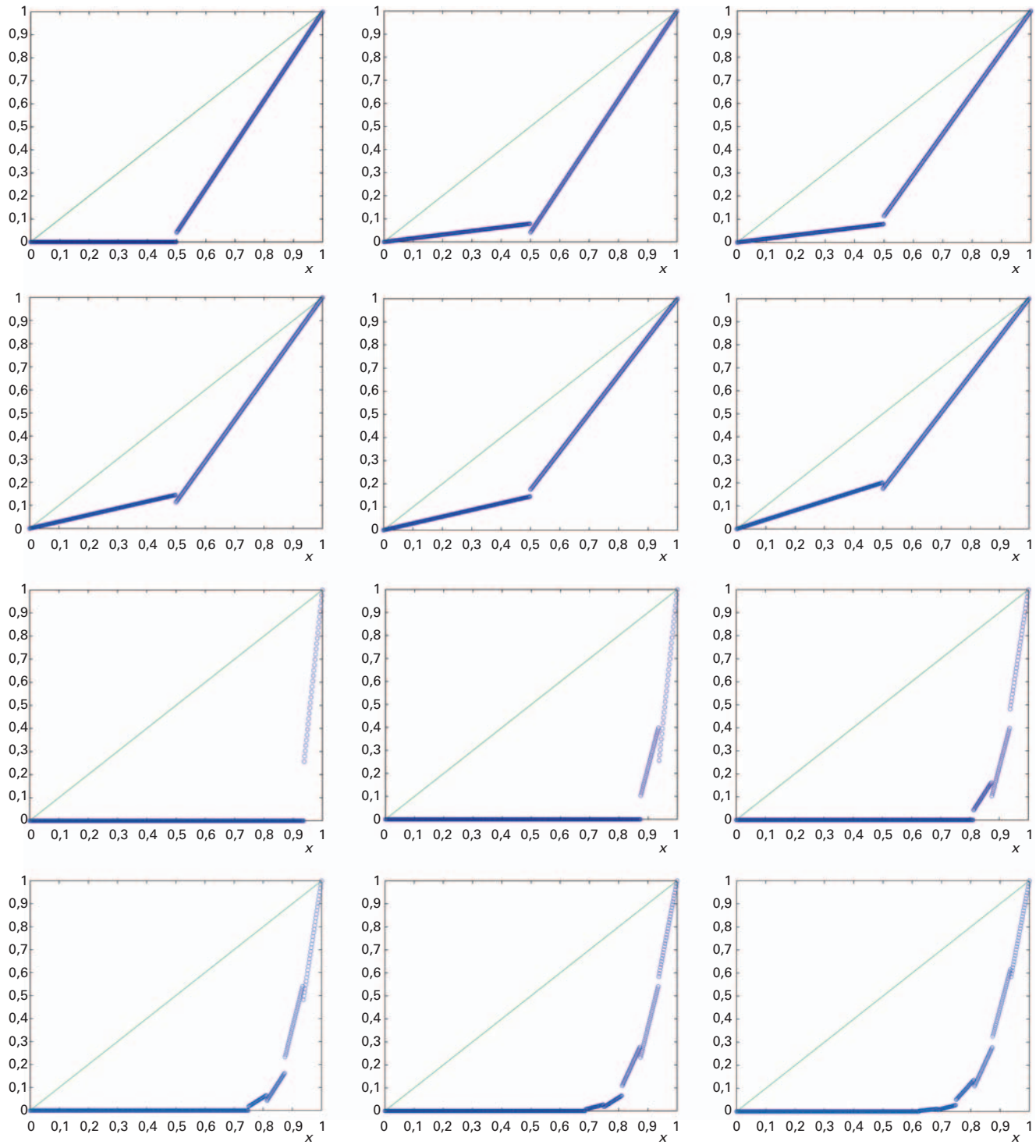
Figure 2 – Résolution du système (14) avec un nombre croissant de sous-domaines

$$\begin{pmatrix} e_1^n(\alpha_2) \\ e_2^n(\beta_1) \\ e_2^n(\alpha_3) \\ e_3^n(\beta_2) \\ e_3^n(\alpha_4) \\ \vdots \\ e_{I-1}^n(\beta_{I-2}) \\ e_{I-1}^n(\alpha_I) \\ e_I^n(\beta_{I-1}) \end{pmatrix} = \begin{pmatrix} 0 & b_1^+ & 0 & & & & & & \\ a_2^- & 0 & 0 & b_2^- & & & & & \\ a_2^+ & 0 & 0 & b_2^+ & 0 & & & & \\ & 0 & a_3^- & 0 & 0 & b_3^- & & & \\ & & a_3^+ & 0 & 0 & b_3^+ & 0 & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & & 0 & a_{I-1}^- & 0 & 0 & b_{I-1}^- \\ & & & & & a_{I-1}^+ & 0 & 0 & b_{I-1}^+ \\ & & & & & & 0 & a_I^- & 0 \end{pmatrix} \begin{pmatrix} e_1^{n-1}(\alpha_2) \\ e_2^{n-1}(\beta_1) \\ e_2^{n-1}(\alpha_3) \\ e_3^{n-1}(\beta_2) \\ e_3^{n-1}(\alpha_4) \\ \vdots \\ e_{I-1}^{n-1}(\beta_{I-2}) \\ e_{I-1}^{n-1}(\alpha_I) \\ e_I^{n-1}(\beta_{I-1}) \end{pmatrix} \quad (2)$$

Réordonnons les inconnues en supposant pour simplifier que I est pair. Nous obtenons le système antidiagonal par blocs

$$\begin{pmatrix} e_1^n(\alpha_2) \\ e_3^n(\beta_2) \\ e_3^n(\alpha_4) \\ \vdots \\ e_{I-1}^n(\beta_{I-2}) \\ e_{I-1}^n(\alpha_I) \\ \hline e_2^n(\beta_1) \\ e_2^n(\alpha_3) \\ e_4^n(\beta_3) \\ e_4^n(\alpha_5) \\ \vdots \\ e_I^n(\beta_{I-1}) \end{pmatrix} = \left(\begin{array}{ccccc|ccccc} & & & & & b_1^+ & & & & \\ & & & & & & a_3^- & b_3^- & & \\ & & & & & & a_3^+ & b_3^+ & & \\ & & & & & & & & a_{I-1}^- & b_{I-1}^- \\ & & & & & & & & a_{I-1}^+ & b_{I-1}^+ \\ \hline a_2^- & b_2^- & & & & & & & & \\ a_2^+ & b_2^+ & & & & & & & & \\ & & a_4^- & b_4^- & & & & & & \\ & & a_4^+ & b_4^+ & & & & & & \\ & & & \ddots & & & & & & \\ & & & & a_I^- & & & & & \end{array}\right) \begin{pmatrix} e_1^{n-1}(\alpha_2) \\ e_3^{n-1}(\beta_2) \\ e_3^{n-1}(\alpha_4) \\ \vdots \\ e_{I-1}^{n-1}(\beta_{I-2}) \\ e_{I-1}^{n-1}(\alpha_I) \\ \hline e_2^{n-1}(\beta_1) \\ e_2^{n-1}(\alpha_3) \\ e_4^{n-1}(\beta_3) \\ e_4^{n-1}(\alpha_5) \\ \vdots \\ e_I^{n-1}(\beta_{I-1}) \end{pmatrix} \quad (3)$$

dont nous noterons $A_H = \begin{pmatrix} 0 & A_1 \\ A_2 & 0 \end{pmatrix}$ la matrice. Sur ce système il est visible que les valeurs des erreurs dans les sous-domaines pairs



2 sous-domaines sur les 2 premières lignes, 16 sous-domaines sur les 2 dernières lignes.

Figure 3 – Premières itérations d'une méthode de Schwarz parallèle

dépendent seulement des erreurs dans les sous-domaines impairs, et vice versa. Il faut donc deux itérations de l'algorithme pour voir tous les domaines affectés par l'algorithme, et nous allons nous intéresser à la matrice A_H^2, qui est diagonale par blocs, car

$$A_H = \begin{pmatrix} 0 & A_1 \\ A_2 & 0 \end{pmatrix} \quad \Rightarrow \quad A_H^2 = \begin{pmatrix} A_1A_2 & 0 \\ 0 & A_2A_1 \end{pmatrix}.$$

Nous mesurons la qualité de l'algorithme par la norme euclidienne de A_H^2, que nous estimons maintenant. Rappelons que la norme euclidienne d'une matrice $\mathcal{A}$ est définie par

$$\|\mathcal{A}\|_2 = \sup_{x \neq 0} \frac{\|\mathcal{A}x\|_2}{\|x\|_2}.$$

La norme de $\mathcal{A}$ est aussi égale à la racine carrée du rayon spectral de $\mathcal{A}^T\mathcal{A}$ (c'est-à-dire le plus grand module des valeurs propres). Pour une matrice $\mathcal{A}$ symétrique, sa norme est égale à son rayon spectral. Pour simplifier les calculs, nous supposons que la décomposition est régulière, c'est-à-dire tous les intervalles de même longueur H, tous les recouvrements de même taille δ. Il ne reste alors plus que deux paramètres : la taille $H = 1/I$ du sous-domaine et $b = \delta/H$, puisque $b_i^- = a_i^+ := b = \delta/H$ et $a_i^- = b_i^+ = a = 1-b$.

Théorème 1.1 Pour une décomposition régulière, la norme euclidienne de A_H^2 est bornée par

$$\left\|A_H^2\right\|_2 \leq 1 - 4b(1-b)\sin^2\left(\frac{\pi H}{2}\right), \quad b = \frac{\delta}{H}.$$

De plus cette estimation est optimale asymptotiquement si b est constant et H tend vers 0 : il existe un vecteur y tel que

$$\frac{\left\|A_H^2 y\right\|_2}{\|y\|_2} \sim 1 - b(1-b)\left(1 + 2b(1+b)\right)(\pi H)^2.$$

Démonstration La norme de A_H^2 est égale au maximum des normes de A_1A_2 et A_2A_1. Le produit des deux matrices devient dans le cas régulier

$$A_1A_2 = \begin{pmatrix} a & & & & & \\ & a & b & & & \\ & b & a & \ddots & & \\ & & & & a & b \\ & & & & b & a \end{pmatrix} \begin{pmatrix} a & b & & & & \\ b & a & \ddots & & & \\ & & & a & b & \\ & & & b & a & \\ & & & & & a \end{pmatrix}$$

$$= \begin{pmatrix} a^2 & ab & & & & & & & & \\ ab & a^2 & ab & b^2 & & & & & & \\ b^2 & ab & a^2 & ab & 0 & & & & & \\ & 0 & ab & a^2 & ab & b^2 & & & & \\ & & b^2 & ab & a^2 & ab & 0 & & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & & b^2 & ab & a^2 & ab & 0 & \\ & & & & & 0 & ab & a^2 & ab & \\ & & & & & & b^2 & ab & a^2 & \end{pmatrix} = M + b^2F.$$

La matrice M est tridiagonale symétrique, sa norme est égale à son rayon spectral

$$\|M\|_2 = a^2 + 2ab\cos\left(\frac{\pi}{I}\right).$$

Nous noterons λ cette quantité. La matrice F est donnée par

$$F\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{I-2} \\ x_{I-1} \end{pmatrix} = \begin{pmatrix} 0 \\ x_4 \\ x_1 \\ x_6 \\ x_3 \\ \vdots \\ x_{I-2} \\ x_{I-5} \\ 0 \\ x_{I-3} \end{pmatrix}$$

si bien que

$$\frac{\|Fx\|_2^2}{\|x\|_2^2} = \frac{\sum_{i \neq 2, I-1} |x_i|^2}{\sum_i |x_i|^2},$$

et $\|F\|_2 = 1$. Nous pouvons en conclure que

$$\|A_1A_2\|_2 = \left\|b^2F + M\right\|_2 \leq b^2 + a^2 + 2ab\cos\left(\frac{\pi}{I}\right),$$

et il en est de même pour A_2A_1 qui s'écrit $M + b^2F$, avec

$$F'\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{I-2} \\ x_{I-1} \end{pmatrix} = \begin{pmatrix} x_3 \\ 0 \\ x_5 \\ x_2 \\ x_7 \\ x_4 \\ \vdots \\ x_{I-6} \\ x_{I-1} \\ x_{I-4} \\ 0 \end{pmatrix}$$

et $\|F'\|_2 = 1$. Compte-tenu de ce que $a + b = 1$, nous pouvons donc écrire $\left\|A_H^2\right\|_2 \leq (a+b)^2 - 4ab\sin^2\left(\frac{\pi}{2I}\right) = 1 - 4b(1-b)\sin^2\left(\frac{\pi}{2I}\right) \sim 1 - b(1-b)\left(\frac{\pi}{I}\right)^2$.

Pour se convaincre que cette majoration est optimale, calculons la norme euclidienne de $A_H^2\boldsymbol{y}$ où $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x} \end{pmatrix}$ et $\boldsymbol{x}$ est le vecteur propre de M associé à la plus grande valeur propre λ, dont les composantes sont donnés par $x_j = \sin\frac{j\pi}{I}$.

$$\frac{\left\|A_H^2\boldsymbol{y}\right\|_2^2}{\|\boldsymbol{y}\|_2^2} = \frac{I - 4 + (1-b)^2 + \left(1-b^2\right)^2 + \left(1-b(1-b)\right)^2}{I-1} = 1 - 4\frac{b}{I-1}(1-b)\left(1 + \frac{1}{2}b^2\right).$$

Calculons d'abord la norme euclidienne de $A_1A_2\boldsymbol{x} = M\boldsymbol{x} + b^2F\boldsymbol{x} = \lambda\boldsymbol{x} + b^2F\boldsymbol{x}$:

$$\left\|\lambda\boldsymbol{x} + b^2F\boldsymbol{x}\right\|_2^2 = \left\|\left(\lambda x_1, \lambda x_2 + b^2x_4, \lambda x_3 + b^2x_1, \ldots\right)\right\|_2^2$$
$$= \lambda^2\sum_{j=1}^{I-1} x_j^2 + b^4\sum_{j \neq 2, I-1} x_j^2 + 2\lambda b^2\sum_{j=1}^{I-3} x_j x_{j+2}.$$

Grâce à l'égalité

$$\sum_{j=0}^{I-1}\cos\frac{2j\pi}{I} = 0, \quad \text{donc} \quad \sum_{j=1}^{I-1}\cos\frac{2j\pi}{I} = -1,$$

nous pouvons calculer le premier terme

$$\sum_{j=1}^{I-1} x_j^2 = \sum_{j=1}^{I-1}\sin^2\frac{j\pi}{I} = \frac{1}{2}\sum_{j=1}^{I-1}\left(1 - \cos\frac{2j\pi}{I}\right) = \frac{I}{2}.$$

Calculons maintenant le troisième terme

$$\begin{aligned}\sum_{j=1}^{I-3} x_j x_{j+2} &= \sum_{j=1}^{I-3} \sin\frac{j\pi}{I}\sin\frac{(j+2)\pi}{I}\\ &= \frac{1}{2}\sum_{j=1}^{I-3}\left(\cos\frac{2\pi}{I}-\cos\frac{2(j+1)\pi}{I}\right)\\ &= \frac{1}{2}\left((I-3)\cos\frac{2\pi}{I}-\sum_{j=2}^{I-2}\cos\frac{2j\pi}{I}\right)\\ &= \frac{1}{2}\left((I-3)\cos\frac{2\pi}{I}-\left(-1-2\cos\frac{2\pi}{I}\right)\right)\\ &= \frac{1}{2}\left((I-1)\cos\frac{2\pi}{I}+1\right).\end{aligned}$$

Nous pouvons maintenant calculer

$$\left\|\lambda\boldsymbol{x}+b^2F\boldsymbol{x}\right\|_2^2=\frac{I}{2}\lambda^2+b^4\left(\frac{I}{2}-x_2^2-x_{I-1}^2\right)+\lambda b^2\left((I-1)\cos\frac{2\pi}{I}+1\right).$$

Développons cette dernière expression comme polynôme en I : $\left\|\lambda\boldsymbol{x}+b^2F\boldsymbol{x}\right\|_2^2=\frac{I}{2}M_1+M_2$, avec

$$\begin{aligned}M_1 &= \lambda^2+2\lambda b^2\cos\frac{2\pi}{I}+b^4,\\ M_2 &= -b^4\left(\sin^2\frac{\pi}{I}+\sin^2\frac{\pi}{I}\right)+\lambda b^2\left(1-\cos\frac{2\pi}{I}\right),\end{aligned}$$

et

$$\frac{\left\|\lambda x+b^2Fx\right\|_2^2}{\|x\|_2^2}=M_1+\frac{2}{I}M_2.$$

Lorsque $\theta=\pi/I$ est petit, λ est un $\mathcal{O}(1)$ et M_2 un $\mathcal{O}(\theta^2)$, si bien que pour développer à l'ordre 2 la quantité précédente, il suffit de développer M_1. Pour cela nous procédons comme précédemment

$$\lambda\sim a^2+2ab-ab\theta^2,\quad \lambda+b^2\sim 1-ab\theta^2,$$

puis

$$\begin{aligned}M_1 &= \left(\lambda+b^2\right)^2-2\lambda b^2\left(1-\cos\frac{2\pi}{I}\right),\\ &\sim\left(1-ab\theta^2\right)^2-4b^2\left(a^2+2ab\right)\theta^2,\\ &\sim 1-2ab\left(1+2b\left(a+2b\right)\right)\theta^2\\ &\sim 1-2ab\left(1+2b(1+b)\right)\theta^2,\end{aligned}$$

et donc

$$\frac{\left\|\lambda\boldsymbol{x}+b^2F\boldsymbol{x}\right\|_2^2}{\|\boldsymbol{x}\|_2^2}\sim 1-2ab\left(1+2b\left(1+b\right)\right)\theta^2.$$

Pour obtenir le résultat complet, il suffit maintenant d'écrire

$$\left\|A_H^2\boldsymbol{y}\right\|_2^2=\left\|\lambda\boldsymbol{x}+b^2F\boldsymbol{x}\right\|_2^2+\left\|\lambda\boldsymbol{x}+b^2F'\boldsymbol{x}\right\|_2^2,$$

et de noter que les deux quantités contenant x sont égales. En effet

$$\begin{aligned}\left\|\lambda\boldsymbol{x}+b^2F'\boldsymbol{x}\right\|_2^2 &= \left\|\left(\lambda x_1+b^2x_3,\ \lambda x_2,\ \lambda x_3+b^2x_5,\ldots\right)\right\|_2^2\\ &= \lambda^2\sum_{j=1}^{I-1}x_j^2+b^4\sum_{j\neq 1,I-2}x_j^2+2\lambda b^2\sum_{j=1}^{I-3}x_jx_{j+2},\end{aligned}$$

et $x_2 = x_{I-2}$ et $x_1 = x_{I-1}$. Donc

$$\frac{\left\|A_H^2\boldsymbol{y}\right\|_2^2}{\|\boldsymbol{y}\|_2^2}=\frac{\left\|\lambda\boldsymbol{x}+b^2F\boldsymbol{x}\right\|_2^2}{\|\boldsymbol{x}\|_2^2}\sim 1-2ab\left(1+2b\left(1+b\right)\right)\theta^2.$$ ■

Nous avons donc montré que la norme euclidienne de la matrice A_H^2 tend rapidement vers 1 lorsque le nombre de sous-domaines augmente, ce qui entraîne que la suite des solutions dans les sous-domaines converge de moins en moins vite. Il est nécessaire d'ajouter un mécanisme pour corriger ce problème, c'est l'ajout d'une **grille grossière**, que nous présentons maintenant.

1.3 Construction d'un solveur grille grossière. Méthode à deux niveaux

Nous avons vu que la motivation pour introduire un calcul grossier est de propager l'information rapidement entre tous les sous-domaines. Mais le calcul lui-même diffère suivant les méthodes et les auteurs (voir par exemple [43]). La grille grossière est constituée d'un ensemble de points dont nous discuterons la position par la suite. La façon la plus simple d'opérer est de calculer un résidu après chaque itération de l'algorithme et ensuite d'utiliser une équation de correction pour ce résidu sur la grille grossière, comme pour la méthode multigrille. On parle alors de méthode à deux niveaux, l'étape n est décomposée comme suit :

Calcul du résidu sur la grille fine :	$\boldsymbol{r}_n = \boldsymbol{f} - A\boldsymbol{u}_n$,
Restriction du résidu à la grille grossière :	$\boldsymbol{r}_c = R\boldsymbol{r}_n$,
Résolution de l'équation discrétisée sur la grille grossière :	$\boldsymbol{u}_c = A_c^{-1}\boldsymbol{r}_c$,
Extension de $\boldsymbol{u}_c$ à la grille fine et correction de $\boldsymbol{u}_n$:	$\boldsymbol{u}_n = \boldsymbol{u}_n + E\boldsymbol{u}_c$.

Ici, l'opérateur E représente l'extension d'un vecteur défini sur la grille grossière à la grille fine par interpolation affine, R la restriction sur la grille grossière d'un vecteur défini sur la grille fine, ses coefficients sont donnés par moyennisation, soit $R_{ij}=\frac{E_{ji}}{\sum_k E_{ki}}$. La matrice A_c est la matrice de la discrétisation du problème sur la grille grossière, calculée par la méthode de Galerkin, $A_c = RAE$. Pour une matrice A et une grille grossière données, le programme suivant calcule les matrices E, R et A_c.

```
function [E, R, Ac]=CoarseOperators(Im, A)
% COARSE compute coarse gride components in one dimension
%   [E, R, Ac]=CoarseOperators(Im, A), computes for a given coarse grid
%   Im and the discretization matrix A a linear extension operator E,
%   the corresponding restriction operator R, and the coarse system
%   matrix Ac.

J=size(A,1); I=length(Im);
R=sparse(I-1,J); E=sparse(J,I-1);
for i=1:I
  if i==1
    dx=1/Im(i); E(1:Im(i)-1,i)=dx:dx:1-dx;   % linear extension
  else
    dx=1/(Im(i)-Im(i-1)); E(Im(i-1):Im(i)-1,i)=0:dx:1-dx;
  end
  if i==I
    dx=1/(J-Im(i)+1); E(Im(i):J,i)=1:-dx:dx;
  else
    dx=1/(Im(i+1)-Im(i)); E(Im(i):Im(i+1)-1,i)=1:dx:dx;
  end
end;
R=E'; R=spdiags(1./sum(R')',0,I,I)*R;        % restriction
Ac=R*A*E;                                    % Galerkin coarse matrix
```

Nous pouvons maintenant utiliser ces composantes dans le programme Matlab suivant, qui réalise en dimension 1 une méthode de Schwarz sur un nombre arbitraire de sous-domaines de taille quelconque. Ce programme contient un paramètre CC qui permet de décider de l'utilisation ou non d'une grille grossière. C'est celui que nous avons utilisé pour réaliser les expériences des figures **2**

et **3**, et avec lequel nous testons maintenant l'effet de l'ajout d'une grille grossière.

```
function [u, err]=SolveDD(f, eta, a, b, gg, gd, li, d, u0, CC, N)
% SOLVEDD solves 1d model problem with domain decomposition
%    u=SolveDD(f, eta, a, b, gg, gd, li, d, u0, CC, N); solves (eta-dxx)u=f on
%    the 1d domain (a, b) with Dirichlet conditions u=gg and u=gd on the
%    equidistant grid defined by the length of the rhs f using a
%    Schwarz algorithm. Subdomains are defined by the indices li(j) of
%    a non-overlapping decomposition, enlarged by d mesh sizes in both
%    directions to obtain an overlapping one, using the initial guess
%    u0, doing N iterations. CC==0 uses no coarse grid, CC==1 a coarse
%    grid with nodes centered in the subdomains, and CC==2 an optimized
%    coarse grid for parallel Schwarz or RAS

ue=Solve1d(f, eta, a, b, gg, gd);                    % reference solution
u=[gg ; u0; gd];                                     % initial guess
J=length(f), h=(b-a)/(J+1); x=a:h:b;
ai=[1, li(2:end-1)-d]; bi=[li(2:end-1)+d, J];        % construct overlapping dec
A=A1d(eta, a, b, J);
err(1)=norm(u-ue, inf);
ft=f; ft(1)=ft(1)+1/h^2*gg; ft(end)=ft(end)+1/h^2*gd;
if CC==1                                             % compute coarse components
   Im=Coarse(li); [E, R, Ac]=Coarse0perators(Im, A);
elseif CC==2
   Im=Coarse0pt(li); [E, R, Ac]=Coarse0perators(Im, A);
end;

for n=1:N                                            % Schwarz iterations
   uo=u;
   for j=1:length(ai)                                % subdomain solves
      tmp=Solve1d(f(ai(j):bi(j)), eta, (ai(j)-1)*h, (bi(j)+1)*h, uo(ai(j)), uo(bi(j)+2));
      if j==1                                        % compose a global solution
         u(ai(j):bi(j)+2-d+1)=tmp(1:end-d+1);
      elseif j==length(ai)
         u(ai(j)+d+1:bi(j)+2)=tmp(1+d+1:end);
      else
         u(ai(j)+d+1:bi(j)+2-d+1)=tmp(1+d+1:end-d+1);
      end
   end;
   if CC                                             % coarse grid correction
      r=ft-A*u(2:end-1);
      uc=Ac\(R*r);
      u(2:end-1)=u(2:end-1)+E*uc;
   end;
   err(n+1)=norm(u-ue, inf);
   plot(x, u, 'o', x, ue, '-');
   title(['iteration number ' num2str(n)])
   xlabel('x')
end;
```

Nous utilisons maintenant ce code pour réaliser les tests décrits dans l'encadré Cas Test. Nous choisissons d'abord les nœuds de la grille grossière de façon classique (voir [43]), c'est-à-dire aux centres des sous-domaines :

```
function Im=Corse(li)
%   COARSE compute coarse grid location in one dimension
%   Im=Coarse(li) computes for a non overlapping domain decomposition
%   given by the vector of interface indices li coarses grid nodes
%   located in th center of subdomain

I=length(li)-1;
for i=1:I
   Im(i)=round((li(i+1)+li(i))/2);
end;
```

Les résultats sont présentés sur la figure **4**.

Nous voyons que dans le cas d'un recouvrement de taille constante (graphiques de gauche de la figure **4**), la grille grossière améliore nettement la convergence, et la rapidité augmente lorsque l'on augmente le nombre de sous-domaines. Lorsque la taille du recouvrement diminue avec la taille des sous-domaines (graphiques de droite de la figure **4**), la convergence est alors indépendante du nombre de sous-domaines.

Ces résultats illustrent la caractéristique principale des méthodes de Schwarz à deux niveaux, qui a été établie dans un cadre géométrique général en deux ou trois dimensions pour la méthode de Schwarz additive dans [11] :

Théorème 1.2 Le conditionnement du système préconditionné par la méthode de Schwarz additif avec grille grossière est majoré par

$$C\left(1+\frac{H}{\delta}\right) \qquad \textbf{(4)}$$

où C est une constante indépendante de H et δ.

Démonstration Voir [43].

Ce résultat théorique explique les comportements décrits sur la figure **5**. En effet, si dans l'estimation (4), δ est constant et H décroît, le conditionnement décroît et la convergence s'améliore, tandis que si δ tend vers 0 avec H, le conditionnement reste constant.

Nous montrons maintenant sur la figure **5** les premières itérations de la méthode de Schwarz avec grille grossière pour 16 sous-domaines.

En comparaison avec la figure **3**, il apparaît clairement que la grille grossière change fondamentalement le comportement de l'algorithme et la convergence est maintenant uniforme sur tous les sous-domaines. Une remarque s'impose pour les premières itérations, où l'on voit une nette discontinuité sur la droite du domaine : cela vient du fait que le premier résidu, avec la donnée initiale fixée à zéro sur les interfaces, atteint un maximum au dernier interface, comme on le voit sur la première itération de la figure **3**. Avec un point de grille grossière au milieu des sous-domaines et une interpolation affine, ce maximum ne peut pas être corrigé suffisamment efficacement.

Pour améliorer le comportement, nous ajoutons maintenant un algorithme de Krylov (voir [43]). Dans la figure **6**, nous comparons l'effet de l'ajout de la méthode de Krylov dans le cas où δ diminue avec H.

Comme dans le cas de deux sous-domaines, la résolution par Krylov accélère considérablement l'algorithme. Dans les premières itérations, la convergence est indépendante du nombre de sous-domaines, puis elle accélère brusquement après un nombre d'itérations lié au nombre de sous-domaines. Ceci relève du phénomène décrit au paragraphe 2.4 de [AF 1 375].

Revenons maintenant à la question de régularité soulevée par l'étude de la figure **3**. Par la nature des algorithmes, les itérations de Schwarz parallèle ne se recollent pas aux interfaces. L'approximation globale à l'étape n est définie soit avec une partition de l'unité comme nous l'avons décrit au paragraphe 2.4 de [AF 1 375], soit comme nous l'avons fait dans le code en recollant les solutions par sous-domaines aux centres des recouvrements (comme dans RAS). Elle est alors discontinue en ces points et le résidu associé est nul partout sauf en ces points. Une projection sur les fonctions affines sur la grille grossière dont les nœuds sont aux centres des sous-domaines va moyenner ce résidu et le distribuer sur la grille grossière. Nous en avons pointé les conséquences sur les premières itérations figure **5**. Nous pouvons faire une correction grille grossière beaucoup plus précise en choisissant deux points dans le recouvrement, de part et d'autre du point de discontinuité. Le programme Matlab ci-après définit cette nouvelle grille grossière et la figure **7** illustre les propriétés de convergence de l'algorithme avec cette grille grossière.

```
function Im=CoarseOpt(li)
%   COARSEOPT compute optimized coarse grid location in one dimension
%   Im=Coarse(li) computes for a non overlapping domain decomposition
%   given by the vector of interface indices li coarse grid nodes
%   located around the discontinuities of a parallel Schwarz method

for i=1:length(li)-2;          % coarse grid points in
   Im(2*i-1)=li(i+1)-1;        % the center of overlaps
   Im(2*i)=li(i+1);            % permitting discontinuities
end;
```

De l'étude de ces tracés nous pouvons tirer plusieurs conclusions. D'abord la convergence est de nouveau indépendante du nombre de sous-domaines, ensuite elle est beaucoup plus rapide qu'avec la grille grossière classique (remarquons dans la figure que l'échelle est différente de celle des cas précédents, puisque

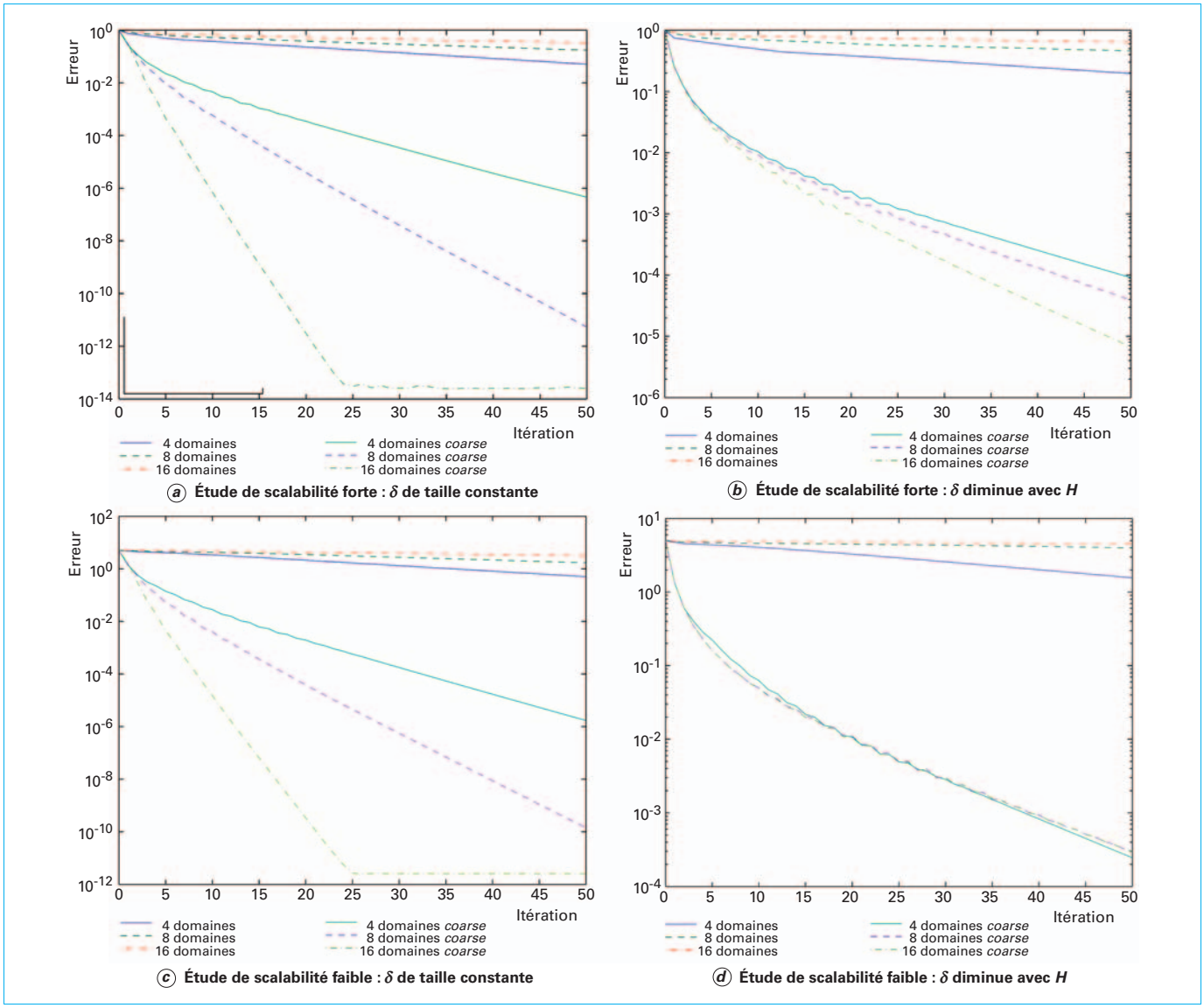


Figure 4 – Résolution du système (14) avec un nombre croissant de sous-domaines, sans et avec grille grossière (*coarse*) classique

nous n'avons tracé que 10 itérations). Par exemple dans l'étude de scalabilité forte, pour 16 sous-domaines, on atteint un résidu de 10^{-6} en deux itérations avec ou sans Krylov, alors qu'il en fallait 16 avec la grille classique et une méthode de Krylov. Sans Krylov, après 50 itérations, le résidu n'avait atteint que 10^{-4}. Il semble qu'avec ce choix de grille grossière, l'algorithme est si performant que l'ajout de Krylov n'est pas vraiment nécessaire. Nous avons fait une observation similaire au sujet de l'accélération de convergence produite par la méthode de Schwarz optimisée (voir figure **24** de [AF 1 375]).

Remarque 1.1 La correction de grille grossière optimisée est parfaite lorsque $\eta = 0$, car les itérées sont affines. Nous avons donc choisi, pour cette dernière expérience, $\eta = 5$.

Nous montrons enfin sur la figure **8** une comparaison des deux méthodes présentées.

Pour les méthodes FETI ou Neumann-Neumann, il y a un candidat naturel pour faire une correction grille grossière, comme nous l'avons déjà mentionné dans la remarque 3.1 de [AF 1 375] : pour les sous-domaines qui ne touchent pas le bord physique avec conditions de Dirichlet, la solution des problèmes avec conditions de Neumann ont une solution définie seulement à une constante près. On peut alors déterminer la constante qui reste libre dans le sous-domaine par un calcul grossier et faire une étude analogue à celle du théorème 2.2, voir [12] et [32] pour Neumann-Neumann et [33] pour FETI.

Théorème 1.3 Le conditionnement du système FETI ou *balancing Neumann-Neumann* est majoré par

$$C\left(1+\log\left(\frac{H}{h}\right)\right)^2, \qquad (5)$$

où C est une constante indépendante de H et h.

Démonstration Voir [43].

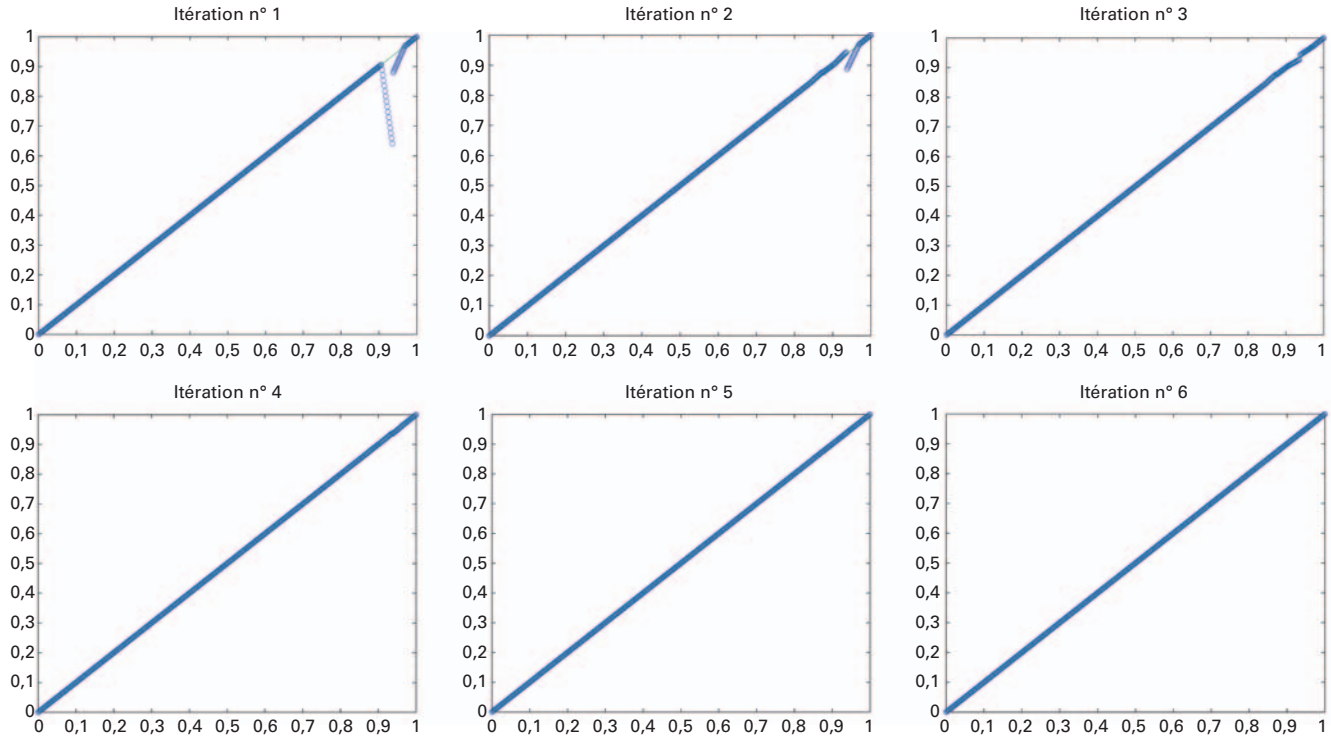
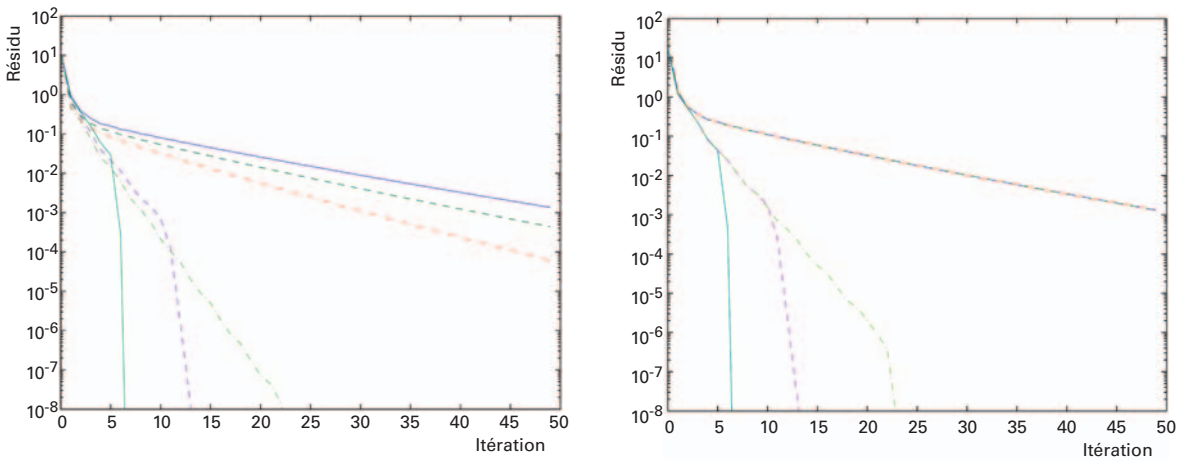


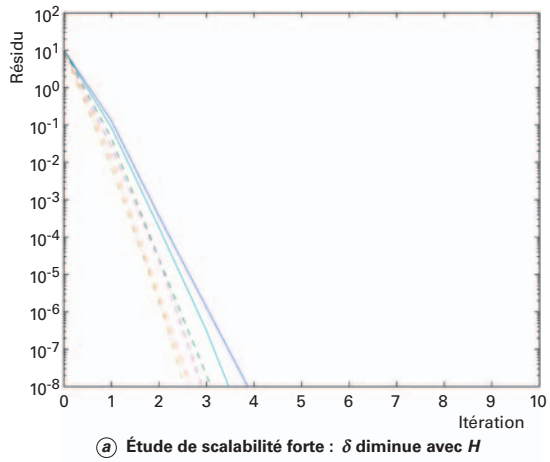
Figure 5 – Premières itérations d'une méthode de Schwarz avec 16 sous-domaines et une correction grille grossière



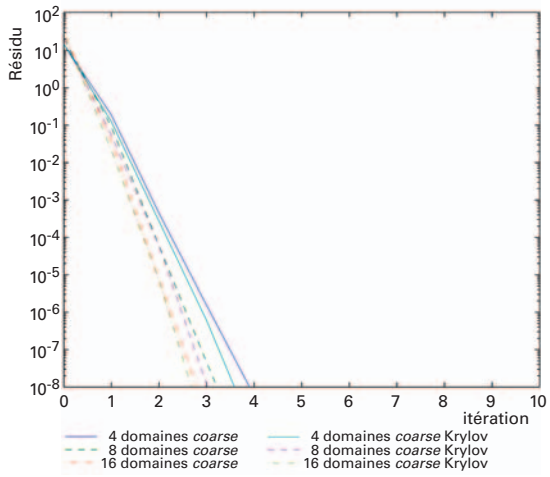
ⓐ **Étude de scalabilité forte : δ diminue avec H**

ⓑ **Étude de scalabilité faible : δ diminue avec H**

Figure 6 – Résolution du système (14) avec un nombre croissant de sous-domaines, avec grille grossière classique, avec ou sans algorithme de Krylov



(a) Étude de scalabilité forte : δ diminue avec H



(b) Étude de scalabilité faible : δ diminue avec H

Figure 7 – Résolution du système (14) avec un nombre croissant de sous-domaines, avec grille grossière adaptée aux discontinuités, avec ou sans algorithme de Krylov

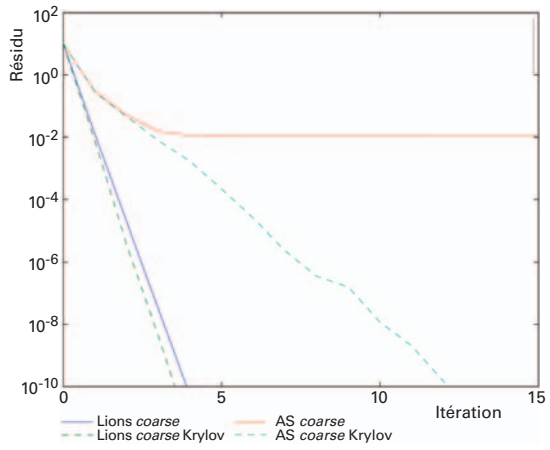


Figure 8 – Comparaison de la méthode Schwarz additif avec grille grossière classique avec la méthode de Schwarz parallèle de Lions et grille grossière optimisée, avec ou sans Krylov

2. Méthodes de parallélisation pour des problèmes en espace-temps

Revenons maintenant à un problème d'évolution en temps, de type chaleur, ondes, ou même Schrödinger. Pour calculer de façon approchée une solution en espace et en temps, on choisira de préférence un schéma implicite en temps dans le premier et le troisième cas, explicite dans le deuxième. Une discrétisation en temps uniforme sur le domaine produira alors dans le cas explicite une parallélisation naturelle, sans nécessité d'itérer entre les processeurs. Par contre pour un schéma implicite, on est amené à résoudre à chaque pas de temps une équation elliptique (voir début de section 1.4 de [AF 1 375]). Celle-ci pourra être résolue par une des méthodes de décomposition de domaine définies dans les sections précédentes. Dans la perspective du couplage de modèles, ou de raffinement local en temps, il peut cependant être intéressant de pouvoir itérer entre les sous-domaines sur un intervalle de temps composé de plusieurs pas de temps. Cela évite également d'avoir à communiquer entre les processeurs à chaque pas de temps, augmentant ainsi le ratio entre temps de calcul et temps de communication. Tout cela fait l'objet des algorithmes de Schwarz relaxation d'onde introduits dans [20] et des méthodes optimisées développées par la suite, dans [18] par exemple. Ces méthodes font l'objet de la première partie de cette section.

Avec l'utilisation des ordinateurs massivement parallèles, il peut arriver que le nombre de processeurs soit trop grand pour les nécessités de la parallélisation en espace. Un exemple extrême est la résolution d'une équation différentielle ordinaire scalaire. Le calcul d'une seule composante ne peut évidemment pas être parallélisé. Par contre, pour des calculs en temps très long (comme en astrophysique par exemple), il est nécessaire de paralléliser les calculs dans la direction du temps. Nous aborderons également cette question dans ce chapitre et conclurons avec un exemple de résolution par une méthode en espace-temps très générale.

2.1 Méthodes de Schwarz relaxation d'onde alternée et parallèle

Considérons de nouveau l'équation de la chaleur en dimension 1 d'espace sur l'intervalle $\Omega = [0, 1]$, et sur l'intervalle de temps $[0, T]$. Le **problème aux limites**

$$\begin{aligned} \partial_t u - \partial_{xx} u &= f && \text{dans } \Omega \times [0, T], \\ u(\cdot, 0) &= u_0 && \text{dans } \Omega, \\ u(0, \cdot) &= g_g, \\ u(1, \cdot) &= g_d, \end{aligned} \qquad \textbf{(6)}$$

a une solution unique $u(x.\ t)$. Voici une implémentation en Matlab d'un solveur pour ce problème, où nous avons choisi $f = 0$ pour simplifier :

```
function u = Heat(a, b, u0, gg, gd, T);
%   HEAT solves the heat equation in one dimension
%   u=Heat(a, b, u0, gg, gd, T); solves the heat equation du/dt = d^2u/dx^2
%   on (a, b)x(0, T) using finite difference and Backward Euler wich
%   initial condition u0 and Dirichlet boundary conditions gg and gd

J = length(u0) - 2;  M = length(gg);
u(1, :) = u0; u(2 : M + 1, 1) = gg; u(2 : M + 1, J + 2) = gd;
                                        %   add initial and boundary conditions
dt = T/M;  dx = (b - a)/(J + 1);
A = A1d(1/dt, a, b, J)
for i = 1 : M,                          %   time stepping
   temp = u(i, 2 : J + 1)'/dt;
   temp(1) = temp(1) + gg(i)/dx^2;      %   add the boundary values
   temp(J) = temp(J) + gd(i)/dx^2;
   u(i+1, 2 :J+1) = (A \ temp)';
end;
```

Pour obtenir une solution de référence, nous partons d'une donnée initiale gaussienne. Au temps initial nous éteignons le

chauffage au centre du barreau pour observer le refroidissement de l'objet, puis nous chauffons l'extrémité gauche à partir de l'instant t = 0,1 s. Le temps d'observation est de 0,2 s. Le petit programme Matlab BarTime.m ci-après effectue cette simulation. L'évolution de la solution est représentée sur la figure **9**.

```
T=0  2;
J=61; M=100;                          %  space and time discretization points
dx=1/(J-1); dt=T/M;
x=(0:dx:1); t=(0:dt:T)';              %  space and time meshes
u0=exp(-20*(0.55-x).^2);              %  initial condition
gg=zeros(M,1); gd=zeros(M,1);         %  Dirichlet boundary conditions
gg(t(2:end)>=0.1)=0.5;
u=Heat(0,1,u0,gg,gd,T);
mesh(x,t,u); xlabel('x'); ylabel('t'); zlabel('solution')
axis([0 1 0 T 0 max(max(u))]);
```

Reprenons la décomposition de Ω en $\Omega_1 = [0, \beta]$ et $\Omega_2 = [\alpha, 1]$, avec $\Gamma_1 = \{\beta\}$ et $\Gamma_2 = \{\alpha\}$, $\delta = \beta - \alpha$. L'algorithme de Schwarz relaxation d'onde parallèle s'écrit pour n = 1, 2,...

$$
\begin{array}{ll}
\partial_t u_1^n - \partial_{xx} u_1^n = f & \text{dans } \Omega_1 \times [0,T], \\
\partial_t u_2^n - \partial_{xx} u_2^n = f & \text{dans } \Omega_2 \times [0,T], \\
u_1^n(\cdot,0) = u_0 & \text{dans } \Omega_1, \\
u_2^n(\cdot,0) = u_0 & \text{dans } \Omega_2, \\
u_1^n(0,\cdot) = g_g, & \\
u_2^n(1,\cdot) = g_d, & \\
u_1^n(\beta,\cdot) = u_2^{n-1}(\beta,\cdot) & \text{sur } [0,T], \\
u_2^n(\alpha,\cdot) = u_1^{n-1}(\alpha,\cdot) & \text{sur } [0,T].
\end{array}
\qquad \textbf{(7)}
$$

Les données initiales de l'algorithme (à ne pas confondre avec la donnée initiale du problème d'évolution u_0) sont des fonctions du temps $g_i(t)$ qui servent de condition à la limite sur Γ_i x $[0,T]$. L'algorithme de Schwarz relaxation d'onde alterné s'obtient de la même façon en remplaçant la condition de transmission en $x = \alpha$ par $u_2^n = u_1^n$.

Théorème 2.1 Pour tout α et β tels que $\delta = \beta - \alpha > 0$, les algorithmes de Schwarz relaxation d'onde alterné et parallèle convergent. Pour i = 1 ou 2, on a pour l'algorithme parallèle l'estimation

$$\sup_{\substack{t\in[0,T]\\ x\in\Omega_i}} \left|u_i^{2n}(x,t) - u(x,t)\right| \leqslant \left(\frac{\alpha(1-\beta)}{\beta(1-\alpha)}\right)^n \sup_i \sup_{\substack{t\in[0,T]\\ x\in\Gamma_i}} \left|u_i^0(x,t) - u(x,t)\right|.$$

Démonstration Nous traitons le cas de l'algorithme parallèle et cette fois nous utilisons une preuve basée sur le principe du maximum. Notons de nouveau $e_i^n = u_i^n - u$ l'erreur dans le sous-domaine i à l'étape n, qui est maintenant une fonction de x et t. Par linéarité, les erreurs sont solutions des problèmes aux limites homogènes (*i.e.* avec $f \equiv 0$ et $u_0 \equiv 0$), et avec les mêmes conditions de transmission. Définissons un algorithme stationnaire, pour des fonctions $\tilde{e}_i^n$ de l'espace uniquement

$$
\begin{array}{ll}
\partial_{xx}\tilde{e}_1^n = 0 \quad \text{dans } \Omega_1, & \partial_{xx}\tilde{e}_2^n = 0 \quad \text{dans } \Omega_2, \\
\tilde{e}_1^n(0) = 0, & \tilde{e}_2^n(1) = 0, \\
\tilde{e}_1^n(\beta) = \sup_{t\in[0,T]} \left|e_1^n(\beta,t)\right|, & \tilde{e}_2^n(\alpha) = \sup_{t\in[0,T]} \left|e_2^n(\alpha,t)\right|.
\end{array}
$$

Fixons les données initiales de l'algorithme stationnaire par $\tilde{g}_i = \sup_{t\in[0,T]} \left|g_i(t) - u_{|\Gamma_i}(t)\right|$. Puisque les $\tilde{e}_i^n$ sont harmoniques, elles vérifient le principe du maximum :

$$0 \leqslant \tilde{e}_i^n \leqslant \tilde{e}_{i\,|\Gamma_i}^n.$$

Nous allons montrer que pour tout (x, t) dans Ω_i x $[0, T]$, $\left|e_i^n(x,t)\right| \leqslant \tilde{e}_i^n(x)$. Pour cela définissons les fonctions de x et t, $d_i^{n\pm} = \tilde{e}_i^n \pm e_i^n$. Par définition des $\tilde{e}_i^n$, les $d_i^{n\pm}$ sont positives ou nulles sur Γ_i et vérifient

$$
\begin{array}{llll}
\partial_t d_1^{n\pm} - \partial_{xx} d_1^{n\pm} = 0 & \text{dans } \Omega_1 \times [0,T], & \partial_t d_2^{n\pm} - \partial_{xx} d_2^{n\pm} = 0 & \text{dans } \Omega_2 \times [0,T], \\
d_1^{n\pm}(\cdot,0) \geqslant 0 & \text{dans } \Omega_1, & d_2^{n\pm}(\cdot,0) \geqslant 0 & \text{dans } \Omega_2, \\
d_1^{n\pm}(0,\cdot) = 0, & & d_2^{n\pm}(1,\cdot) = 0, & \\
d_1^{n\pm}(\beta,\cdot) \geqslant 0 & \text{sur } [0,T], & d_2^{n\pm}(\alpha,\cdot) \geqslant 0 & \text{sur } [0,T].
\end{array}
$$

Par le principe du maximum pour l'équation de la chaleur, nous en déduisons que $d_i^{n\pm}$ est positive ou nulle sur Ω_i x $[0, T]$, et donc que $\left|e_i^n\right| \leqslant \tilde{e}_i^n$ sur Ω_i x $[0, T]$. Nous pouvons alors écrire la suite d'égalités et inégalités

$$\tilde{e}_1^n(\beta) = \sup_{t\in[0,T]} \left|e_1^n(\beta,t)\right| = \sup_{t\in[0,T]} \left|e_2^{n-1}(\beta,t)\right| \leqslant \tilde{e}_2^{n-1}(\beta) = \frac{1-\beta}{1-\alpha}\tilde{e}_2^{n-1}(\alpha),$$

$$\tilde{e}_2^{n-1}(\alpha) = \sup_{t\in[0,T]} \left|e_2^{n-1}(\alpha,t)\right| = \sup_{t\in[0,T]} \left|e_2^{n-2}(\alpha,t)\right| \leqslant \tilde{e}_1^{n-2}(\alpha) = \frac{\alpha}{\beta}\tilde{e}_1^{n-2}(\beta).$$

Nous en déduisons que la suite $\tilde{e}_1^{2n}(\beta)$ est une suite géométrique,

$$\tilde{e}_1^{2n}(\beta) \leqslant \left(\frac{\alpha(1-\beta)}{\beta(1-\alpha)}\right)\tilde{e}_1^{2n-2}(\beta).$$

La raison est strictement inférieure à 1, cette suite converge vers 0, de même que $\tilde{e}_2^{2n}(\alpha)$. Par le principe du maximum les suites $\tilde{e}_i^{2n}$ tendent vers 0 uniformément dans leurs domaines respectifs, et il en est de même des suites e_i^{2n}. ■

La raison de la suite géométrique e_1^{2n} ne dépend pas du temps, la convergence des itérées est donc pour tout temps au moins linéaire. En fait, on peut estimer plus précisément la convergence sur un intervalle de temps borné :

Théorème 2.2 Dans le cas de deux demi-droites infinies, $\beta_1 =]-\infty, \beta]$ et $\Omega_2 = [\alpha, +\infty[$ avec $\delta = \beta - \alpha > 0$, les algorithmes de Schwarz relaxation d'onde alterné et parallèle convergent

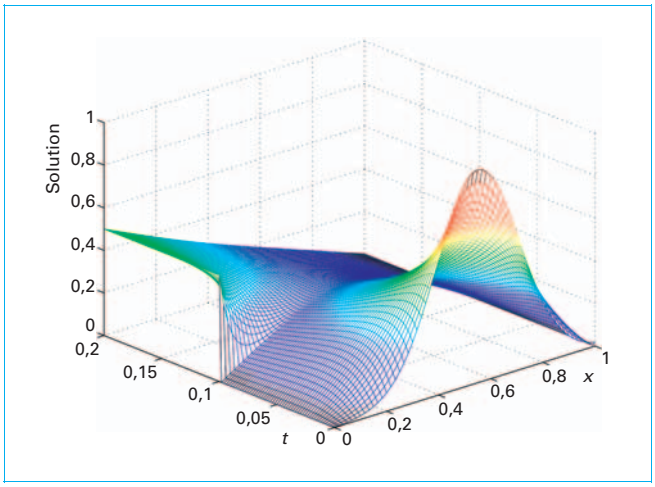


Figure 9 – Exemple d'une solution instationnaire de l'équation de la chaleur

superlinéairement sur un intervalle de temps borné. Par exemple pour l'algorithme parallèle, on a dans le sous-domaine de gauche :

$$\sup_{t\in[0,T]}\left|\left(u_1^{2n}-u\right)(\alpha,\cdot)\right| \leqslant \operatorname{erfc}\left(\frac{n\delta}{\sqrt{T}}\right)\sup_{t\in[0,T]}\left|\left(u_1^{0}-u\right)(\alpha,\cdot)\right|,$$

où la fonction d'erreur complémentaire notée erfc est définie par $\operatorname{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^{\infty} e^{-s^2}\mathrm{d}s$.

Démonstration La transformée de Laplace d'une fonction intégrable en temps est définie pour $\Re\mathrm{e}\, s>0$ par

$$\mathfrak{L}f : s \mapsto \tilde{f}(s)=\int_0^{+\infty} f(t)e^{-st}\mathrm{d}t.$$

En tant que fonctions de t, les erreurs e_i^n sont définies sur [0, T], nulles en 0. On peut les prolonger en des fonctions continues sur $\mathbb{R}_+$ et prolonger les équations sur tout $\mathbb{R}_+$. Leurs transformées de Laplace en temps $\hat{e}_i^n(x,s)$ sont alors solutions des problèmes

$$\begin{array}{ll} s\hat{e}_1^n-\partial_{xx}\hat{e}_1^n=0 \quad \text{dans } \Omega_1, & s\hat{e}_2^n-\partial_{xx}\hat{e}_2^n=0 \quad \text{dans } \Omega_2,\\ \hat{e}_1^n(\beta,s)=\hat{e}_2^{n-1}(\beta,s), & \hat{e}_2^n(\alpha,s)=\hat{e}_1^{n-1}(\alpha,s). \end{array} \qquad (8)$$

La solution générale de l'équation différentielle est une combinaison linéaire de $e^{\sqrt{s}x}$ et $e^{-\sqrt{s}x}$. Mais puisque nous cherchons des solutions bornées en la variable x, il ne reste que

$$\hat{e}_1^n(x,s)=\hat{e}_2^{n-1}(\beta,s)\,e^{\sqrt{s}(x-\beta)}, \quad \hat{e}_2^n(x,s)=\hat{e}_1^{n-1}(\alpha,s)\,e^{-\sqrt{s}(x-\alpha)}.$$

En évaluant aux interfaces sur deux itérations, nous obtenons par récurrence

$$\hat{e}_1^{2n}(\alpha,s)=\left(\rho^D(s)\right)^n\hat{e}_1^0(\alpha,s), \quad \hat{e}_2^{2n}(\beta,s)=\left(\rho^D(s)\right)^n\hat{e}_2^0(\beta,s), \qquad (9)$$

où le facteur de convergence ρ^D est défini par

$$\rho^D(s)=e^{-2\delta\sqrt{s}}. \qquad (10)$$

Les relations (9) s'interprètent en terme de produit de convolution par transformée de Laplace inverse :

$$e_1^{2n}(\alpha,\cdot)=G*e_1^0(\alpha,\cdot), \quad e_2^{2n}(\beta,\cdot)=G*e_2^0(\beta,\cdot),$$

où G est la transformée de Laplace inverse de $(\rho^D(s))^n$, donné par

$$G(t)=\frac{n\delta}{\sqrt{\pi t^3}}e^{-\frac{n^2\delta^2}{t}}.$$

Nous pouvons alors écrire, pour tout $t\in[0,T]$,

$$\left|e_1^{2n}(\alpha,t)\right|=\left|\int_0^t e_1^0(\alpha,t-\tau)G(\tau)\mathrm{d}\tau\right| \leqslant \sup_{t\in[0,T]}\left|e_1^0(\alpha,t)\right|\int_0^T G(t)\mathrm{d}t$$

$$=\sup_{t\in[0,T]}\left|e_1^0(\alpha,t)\right|\operatorname{erfc}\left(\frac{n\delta}{\sqrt{T}}\right).$$

La fonction erfc tend vers 0 à l'infini, la suite $e_1^{2n}(\alpha,-)$ tend donc uniformément vers 0. ■

Remarque 2.1 Il est intéressant de comparer le comportement de la méthode de Schwarz relaxation d'onde avec celui de la méthode de relaxation d'ondes pour une équation différentielle ordinaire, donné par le théorème 1.2. Pour de grandes valeurs de n, la convergence de l'algorithme de Picard est estimée à l'aide de la formule de Stirling par

$$\frac{(LT)^n}{n!}\sim\frac{(LTe)^n}{\sqrt{2\pi n}}e^{-n\log n},$$

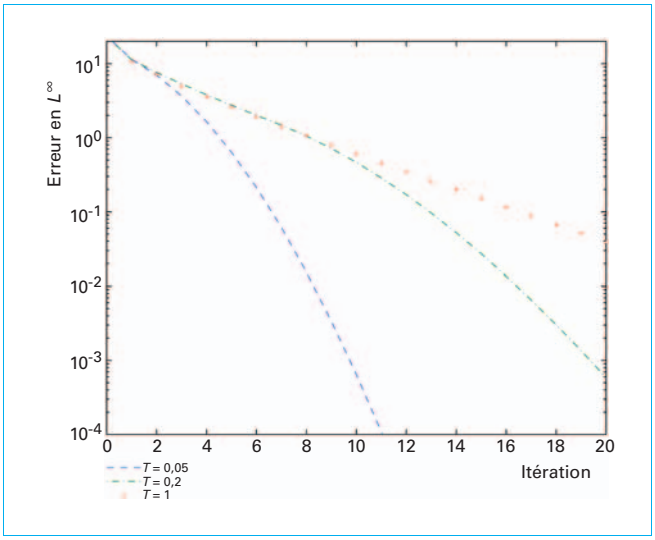


Figure 10 – Convergence linéaire et superlinéaire de l'algorithme de Schwarz relaxation d'onde

tandis que celle de l'algorithme de Schwarz relaxation d'ondes est donnée par

$$\operatorname{erfc}\left(\frac{n\delta}{\sqrt{T}}\right)\sim\frac{\sqrt{T}}{\sqrt{\pi}n\delta}e^{-\frac{\delta^2}{T}n^2}.$$

Il suffit de comparer les termes dominants en exponentielle pour constater que la convergence de l'algorithme Schwarz relaxation d'onde est meilleure.

Voici un script en Matlab pour tester cet algorithme :

```
BarTime ;                                   %  compute reference solution
α=28 ; d=4 ;
u1=[u0(1:α+d+1) ; gg zeros(M, α+d)] ;       %  zero initial guess
u2=[u0(α+1:end) ; zeros(M, J-α-1) gd] ;
udd=[u1(:, 1:α) u2] ;
err(1)=norm(u-udd, 'inf') ;
x1=0:dx:(α+d)*dx ;   x2=α*dx:dx:1 ;         %  finite difference meshes
for i=1:20
   u1=Heat(0, (α+d)*dx, u0(1:α+d+1), gg, u2(2:end, d+1), T) ;
   u2=Heat(α*dx, 1, u0(α+1:end), u1(2:end, end-d), gd, T) ;
   udd=[u1(:, 1:α)   u2] ;
   mesh(x, t, udd) ; xlabel('x') ; ylabel('t') ;
   zlabel('Schwarz waveform relaxation iterates') ;
   pause
   err(i+1)=norm(u-udd, 'inf') ;
end
```

La figure **10** représente les courbes de convergence de l'algorithme pour le problème de référence, avec les temps finaux T = 0,05, T = 0,2, et T = 1. On voit bien la convergence linéaire pour T = 1 « grand », la convergence superlinéaire pour T = 0,05 « petit » et un régime mixte, linéaire suivi de superlinéaire pour le temps intermédiaire T = 0,2.

2.2 Méthodes de Schwarz relaxation d'onde optimisés

Comme dans le cas des équations elliptiques, la convergence peut être beaucoup améliorée par l'introduction de conditions de transmission de type Robin. Dans le cas parallèle, les conditions de transmission dans le système (7) sont alors remplacées par

$$\begin{array}{ll} \partial_x u_1^n+pu_1^n=\partial_x u_2^{n-1}+pu_2^{n-1} & \text{sur } \Gamma_1\times[0,T],\\ \partial_x u_2^n+pu_2^n=\partial_x u_1^{n-1}-pu_1^{n-1} & \text{sur } \Gamma_2\times[0,T]. \end{array} \qquad (11)$$

L'algorithme est initialisé par les conditions aux limites

$$\partial_x u_2^0 + p u_2^0 = g_2 \text{ sur } \Gamma_1 \times [0,T], \quad \partial_x u_1^0 + p u_1^0 = g_1 \text{ sur } \Gamma_2 \times [0,T].$$

Théorème 2.3 Pour tout $p > 0$, les algorithmes de Schwarz relaxation d'onde alterné ou parallèle avec conditions de Robin convergent avec ou sans recouvrement :

$$\lim_{n \to +\infty} \sup_{t \in [0,T]} \int_{\Omega_i} \left(u_i^n(x,t) - u(x,t)\right)^2 \mathrm{d}x = 0.$$

Démonstration Pour simplifier la présentation, les preuves sont effectuées dans le cas de deux demi-droites infinies pour l'algorithme parallèle comme dans le théorème 2.2. Nous écrivons les équations portant sur les erreurs e_i^n, et effectuons une transformation de Laplace en temps comme dans la preuve du théorème 2.2. Nous obtenons pour les itérées paires par exemple

$$\hat{e}_1^{2n}(\alpha, s) = \left(\rho^R(s,p)\right)^n \hat{e}_1^0(\alpha, s), \quad \hat{e}_2^{2n}(\beta, s) = \left(\rho^R(s,p)\right)^n \hat{e}_2^0(\beta, s),$$

avec

$$\rho^R(s,p) = \left(\frac{\sqrt{s} - p}{\sqrt{s} + p}\right)^2 e^{-2\delta\sqrt{s}}.$$

La première partie du facteur de convergence vient de la condition de Robin, dans la seconde nous reconnaissons le facteur $\rho^D(s)$ qui définit l'algorithme de Schwarz classique et qui ne dépend que du recouvrement. Il est facile de voir que pour $p > 0$, pour tout s de partie réelle positive, $\left|\rho^R(s,p)\right| < 1$.

La suite des valeurs de $\hat{e}_2^{2n}(\beta, s)$ converge donc vers 0. D'autre part, $\left|\hat{e}_2^{2n}(\beta, s)\right|$ est borné par $\left|\hat{e}_2^0(\beta, s)\right|$ qui est de carré intégrable dans le demi-plan $\Re e\, s > 0$. Par le théorème de convergence dominée dans $L^2(\mathbb{C})$ (voir [39]), la suite des fonctions $\hat{e}_2^{2n}(\beta, \cdot)$ converge vers 0 dans $L^2(\mathbb{C})$. Par l'identité de Parseval (voir [39]), la suite de fonctions $e_2^{2n}(\beta, \cdot)$ converge également vers 0, dans $L^2(0, T)$. La fin de la démonstration est analogue à celle du théorème 2.2.

Cet argument est valable avec ou sans recouvrement. Néanmoins nous allons produire maintenant dans le cas où $\alpha = \beta$ un joli argument énergétique dû à Després [8]. Multiplions l'équation $\partial_t e_i^n - \partial_{xx} e_i^n = 0$ par e_i^n et intégrons sur Ω_i. Il vient

$$\int_0^\alpha \left(\partial_t e_1^n(x,t)\right) e_1^n(x,t)\,\mathrm{d}x - \int_0^\alpha \left(\partial_{xx} e_1^n(x,t)\right) e_1^n(x,t)\,\mathrm{d}x = 0,$$
$$\int_\alpha^1 \left(\partial_t e_2^n(x,t)\right) e_2^n(x,t)\,\mathrm{d}x - \int_\alpha^1 \left(\partial_{xx} e_2^n(x,t)\right) e_2^n(x,t)\,\mathrm{d}x = 0,$$

ce qui implique après intégration par parties et en sortant les dérivées en temps des intégrales :

$$\frac{1}{2}\frac{d}{dt}\int_0^\alpha \left(e_1^n(x,t)\right)^2 \mathrm{d}x + \int_0^\alpha \left(\partial_x e_1^n(x,t)\right)^2 \mathrm{d}x - \left(\partial_x e_1^n(\alpha,t)\right) e_1^n(\alpha,t) = 0,$$
$$\frac{1}{2}\frac{d}{dt}\int_\alpha^1 \left(e_2^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(\partial_x e_2^n(x,t)\right)^2 \mathrm{d}x + \left(\partial_x e_2^n(\alpha,t)\right) e_2^n(\alpha,t) = 0.$$

Nous utilisons pour le terme frontière l'égalité $ab = \frac{1}{4p}\left((a+pb)^2 - (a-pb)^2\right)$ et obtenons

$$\frac{1}{2}\frac{d}{dt}\int_0^\alpha \left(e_1^n(x,t)\right)^2 \mathrm{d}x + \int_0^\alpha \left(\partial_x e_1^n(x,t)\right)^2 \mathrm{d}x + \frac{1}{4p}\left(\partial_x e_1^n(\alpha,t) - p e_1^n(\alpha,t)\right)^2$$
$$= \frac{1}{4p}\left(\partial_x e_1^n(\alpha,t) + p e_1^n(\alpha,t)\right)^2,$$
$$\frac{1}{2}\frac{d}{dt}\int_\alpha^1 \left(e_2^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(\partial_x e_2^n(x,t)\right)^2 \mathrm{d}x + \frac{1}{4p}\left(\partial_x e_2^n(\alpha,t) + p e_2^n(\alpha,t)\right)^2$$
$$= \frac{1}{4p}\left(\partial_x e_2^n(\alpha,t) - p e_2^n(\alpha,t)\right)^2.$$

Utilisons maintenant les conditions de transmission (11) appliquées aux erreurs pour remplacer le membre de droite :

$$\frac{1}{2}\frac{d}{dt}\int_0^\alpha \left(e_1^n(x,t)\right)^2 \mathrm{d}x + \int_0^\alpha \left(\partial_x e_1^n(x,t)\right)^2 \mathrm{d}x + \frac{1}{4p}\left(\partial_x e_1^n(\alpha,t) - p e_1^n(\alpha,t)\right)^2$$
$$= \frac{1}{4p}\left(\partial_x e_2^{n-1}(\alpha,t) + p e_1^{n-1}(\alpha,t)\right)^2,$$
$$\frac{1}{2}\frac{d}{dt}\int_\alpha^1 \left(e_2^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(\partial_x e_2^n(x,t)\right)^2 \mathrm{d}x + \frac{1}{4p}\left(\partial_x e_2^n(\alpha,t) + p e_2^n(\alpha,t)\right)^2$$
$$= \frac{1}{4p}\left(\partial_x e_2^{n-1}(\alpha,t) - p e_2^{n-1}(\alpha,t)\right)^2.$$

Ajoutons ces deux égalités pour obtenir

$$\frac{1}{2}\frac{d}{dt}\left(\int_0^\alpha \left(e_1^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(e_2^n(x,t)\right)^2 \mathrm{d}x\right) + \int_0^\alpha \left(\partial_x e_1^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(\partial_x e_2^n(x,t)\right)^2 \mathrm{d}x$$
$$+ \frac{1}{4p}\left[\left(\partial_x e_1^n(\alpha,t) - p e_1^n(\alpha,t)\right)^2 + \left(\partial_x e_2^n(\alpha,t) + p e_2^n(\alpha,t)\right)^2\right]$$
$$= \frac{1}{4p}\left[\left(\partial_x e_1^{n-1}(\alpha,t) - p e_1^{n-1}(\alpha,t)\right)^2 + \left(\partial_x e_2^{n-1}(\alpha,t) + p e_2^{n-1}(\alpha,t)\right)^2\right].$$

Nous constatons que les termes frontières à droite et à gauche du signe égal sont les mêmes décalés d'un indice. Si bien qu'en sommant ces égalités entre 1 et N, tous les termes disparaissent sauf les deux extrêmes :

$$\sum_{n=1}^N \frac{1}{2}\frac{d}{dt}\left(\int_0^\alpha \left(e_1^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(e_2^n(x,t)\right)^2 \mathrm{d}x\right) + \sum_{n=1}^N \left(\int_0^\alpha \left(\partial_x e_1^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(\partial_x e_2^n(x,t)\right)^2 \mathrm{d}x\right)$$
$$+ \frac{1}{4p}\left(\left(\partial_x e_1^N(\alpha,t) - p e_1^N(\alpha,t)\right)^2 + \left(\partial_x e_2^N(\alpha,t) + p e_2^N(\alpha,t)\right)^2\right)$$
$$= \frac{1}{4p}\left(\left(\partial_x e_1^0(\alpha,t) - p e_1^0(\alpha,t)\right)^2 + \left(\partial_x e_2^0(\alpha,t) + p e_2^0(\alpha,t)\right)^2\right).$$

Intégrons sur $(0, t)$, et en utilisant le fait que les conditions initiales sont nulles pour l'erreur, nous obtenons

$$\sum_{n=1}^N \frac{1}{2}\left(\int_0^\alpha \left(e_1^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(e_2^n(x,t)\right)^2 \mathrm{d}x\right)$$
$$+ \sum_{n=1}^N \int_0^t \left(\int_0^\alpha \left(\partial_x e_1^n(x,s)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(\partial_x e_2^n(x,s)\right)^2 \mathrm{d}x\right)\mathrm{d}s$$
$$\leqslant \int_0^t \left(\left(g_1(s) - (\partial_x u - pu)(\alpha,s)\right)^2 + \left(g_2(s) - (\partial_x u - pu)(\alpha,s)\right)^2\right)\mathrm{d}s.$$

Pour tout t dans l'intervalle de temps $[0, T]$, lorsque N tend vers l'infini, les séries à termes positifs du membre de gauche sont donc convergentes, par conséquent leur terme général tend vers 0, et

$$\sup_{t \in [0,T]} \left[\int_0^\alpha \left(e_1^n(x,t)\right)^2 \mathrm{d}x + \int_\alpha^1 \left(e_2^n(x,t)\right)^2 \mathrm{d}x\right] \to 0 \text{ quand } n \to +\infty. \quad \blacksquare$$

Tableau 1 – Résultats asymptotiques

Méthode	*p* optimal	maximum du facteur de convergence
Sans recouvrement	$(\omega_{\min}\omega_{\max})^{\frac{1}{4}}$	$1-2\sqrt{2}\left(\frac{\omega_{\min}}{\omega_{\max}}\right)^{\frac{1}{4}}$
Avec recouvrement $\delta = \mathbb{O}(\Delta t)$	$\sqrt{2}(\omega_{\min}\omega_{\max})^{\frac{1}{4}}$	$1-2\sqrt{2}\left(\frac{\omega_{\min}}{\omega_{\max}}\right)^{\frac{1}{4}}$
Avec recouvrement $\delta = \mathbb{O}\left(\sqrt{\Delta t}\right)$	$2(4\omega_{\min})^{\frac{1}{3}}\delta^{-\frac{1}{3}}$	$1-4\left(\frac{\omega_{\min}}{2}\right)^{\frac{1}{6}}\delta^{\frac{1}{3}}$

Comme pour le problème stationnaire, le paramètre *p* peut être choisi de façon à minimiser le facteur de convergence sur les valeurs de *s* purement imaginaires :

$$\sup_{\omega\in[\omega_{\min},\omega_{\max}]}\left|\rho^R(i\omega, p)\right|.$$

Ici les fréquences extrêmes sont estimées par $\omega_{\min} = \frac{\pi}{2T}$ et $\omega_{\max} = \pi/\Delta t$, où Δt est le pas de discrétisation en temps. C'est de nouveau un problème de meilleure approximation au sens de Chebyshev, qui peut être complètement résolu par les méthodes d'équi-oscillation inspirées par les résultats de Chebyshev et de la Vallée Poussin [7] [44]. Nous donnons dans le tableau **1** les valeurs asymptotiques des paramètres. Il y a maintenant deux paramètres asymptotiques : la fréquence maximale en temps ω_{max} est grande et le recouvrement δ, de l'ordre de quelques pas d'espace, est petit. Les pas de temps et d'espace sont liés. Pour un schéma explicite, une condition de stabilité impose que $\Delta t = \mathbb{O}\left(\Delta x^2\right) = \mathbb{O}\left(\delta^2\right)$, tandis que dans le cas implicite, on choisit d'ordinaire $\Delta t = \mathbb{O}(\Delta x) = \mathbb{O}(\delta)$.

Voici le solveur Matlab avec des conditions de Robin :

```
function u=HeatR(α, b, u0, gg, gd, T, p1, p2);
%   HEATR solves the heat equation with Robin conditions
%   u=HeatR(α, b, u0, gg, gd, T, p1, p2);solves the heat equation
%   du/dt=d^2u/dx^2 on (α, b)  (0, T) using finite difference and
%   Backward Euler with the initial condition u0 and Robin boundary
%   condition gg and gd with parametres p1 and p2.

J=length(u0); M=length(gg)
u(1,:)=u0;
dt=T/M; dx=(b-α)/(J-1);
A = Ald(1/dt, α-dx, b+dx, J);          %   Ald constructs interior matrix
A(1, 1) = 1/2/dt+(1+p1*dx)/dx^2;       %   for the Robin conditions
A(J, J) = 1/2/dt+(1+p2*dx)/dx^2;
for i=1:M,
   temp=u(i,1:J)'/dt;
   temp(1)= temp(1)/2+gg(i)/dx;        %   add the boundary values
   temp(J)=temp(J)/2+gd(i)/dx;
   u(i+1,1:J) = (A\temp)';
end;
```

Avec le programme de test en Matlab ci-après, nous avons établi les courbes de convergence de la figure **11**. Le paramètre optimisé est obtenu par la formule du tableau **1**, soit $p^* = 6{,}6655$. Comme précédemment, nous utilisons une condition de Robin avec un paramètre très grand pour émuler une condition de Dirichlet sur le bord extérieur.

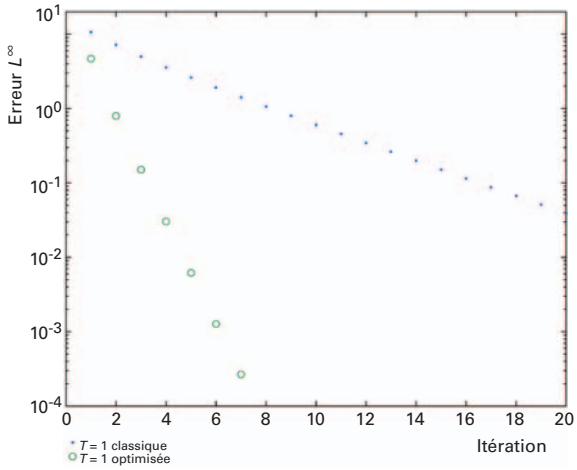


Figure 11 – Comparaison des méthodes de Schwarz relaxation d'onde classique et optimisée

```
T=1;
J=61; M=100;                    %   space and time discretization points
dx=1/(J-1); dt=T/M;
x=(0:dx:1); t=(0:dt:T)';        %   space and time meshes
uo=exp(-20*(0.55-x)^2);         %   initial condition
gg=zeros(M,1); gd=zeros(M,1);   %   Dirichlet boundary conditions
gg(t(2:end)>=0.1)=0.5;
pe=1e5;                         %   emulate Dirichlet conditions
u=HeatR(0, 1, u0, pe*gg, pe*gd, T, pe, pe);     %   compute reference solution

α=28; d=4;
u1=[u0(1:α+d+1); gg zeros(M, α+d)];             %   zero initial guess
u2=[u0(α+1:end); zeros(M, J-α-1)gd];
udd=[u1(:,1:α)u2];
err(1)=norm(u-udd,'inf');
x1=0:dx:(α+d)*dx;   x2=α*dx:dx:1;               %   finite difference meshes
p=sqrt(2)*(pi/(2*T)*pi/dt)^0.25;                %   optimized parameter
for i=1:20
   tb=dx*(1/2/dt*u2(1:end-1, d+1)-(1/2/dt+(1-p*dx)/dx^2)*u2(2:end, d+1)
                                 +1/dx^2*u2(2:end, d+2));
   u1=HeatR(0, (α+d)*dx, u0(1:α+d+1), pe*gg, tb, T, pe, p);
   ta=dx*(1/2/dt*u1(1:end-1, end-d)-(1/2/dt+(1-p*dx)/dx^2)*u1(2:end, end-d)
                                 +1/dx^2*u1(2:end, end-d-1));
   u2=HeatR(α*dx, 1, u0(α+1:end), ta, pe*gd, T, p, pe);
   udd=[u1(:,1:α)u2];
   mesh(x, t, udd); xlabel('x'); ylabel('t');
   zlabel('Optimized Schwarz waveform relaxation iterates');
   pause
   err(i+1)=norm(u-udd,'inf');
end
```

Il apparaît clairement qu'avec la condition de Robin optimisée, on peut nettement accélérer la convergence, même sur un intervalle de grande taille, $T = 1$. On obtient ainsi une vitesse de convergence qui était seulement atteignable sur des intervalles de temps très courts, voir figure **10**.

Un dernier commentaire important s'impose ici pour les méthodes de Schwarz relaxation d'onde : on n'a pas besoin d'une grille grossière, même avec beaucoup de sous-domaines, si l'on calcule sur une fenêtre en temps courte, car, dans ce cas, la condition initiale est plus importante que les conditions aux interfaces et la condition initiale est juste, voir [22]. Par contre, si l'on calcule sur un intervalle de temps long, il faut ajouter une composante grille grossière, comme pour les méthodes de décomposition de domaine pour des problèmes stationnaires, pour obtenir une méthode scalable. L'étude de cette question est en cours.

2.3 Parallélisation en temps : l'algorithme pararéel

Avec l'avènement des calculateurs parallèles à grande échelle composés de centaines de milliers de processeurs, le besoin de parallélisation est devenu tel que souvent la parallélisation en espace ne suffit plus pour utiliser tous les nœuds du calculateur. Comme nous l'avons vu à la section 4, si l'on ajoute une grille grossière, les méthodes de décomposition de domaines deviennent scalables, mais seulement tant que les sous-problèmes locaux sont de taille suffisante (le temps de communication entre les processeurs reste négligeable devant le temps de calcul dans le processeur). Il paraît donc intéressant de pouvoir utiliser le parallélisme aussi dans la direction du temps. Cependant, par le principe de causalité, la dimension du temps ne peut pas être traitée comme les dimensions d'espace. Soit à résoudre l'équation $u' = f(t, u(t))$ sur l'intervalle $[0, T]$, avec une donnée initiale $u(0) = u^0$. Imaginons une décomposition de l'intervalle $[0, T]$ en fenêtres $[T_i, T_{i+1}]$. Les valeurs de u dans la fenêtre numéro i ne dépendent pas de ses valeurs dans les fenêtres ultérieures. De plus un calcul dans cette fenêtre n'a de sens que si la donnée initiale en T_i est connue avec suffisamment de précision. On imagine donc assez bien qu'il faut un mécanisme global pour paralléliser en temps de façon convaincante. Ce mécanisme est connu sous le nom de **méthode de tir**, destiné à résoudre une équation différentielle ordinaire du deuxième ordre en dimension 1, avec données aux deux extrémités [24] [45].

Une variante de cette méthode, dont l'application aux équations aux dérivées partielles a reçu récemment beaucoup d'attention, est l'**algorithme pararéel** [27]. L'idée de base, même si la méthode a été présentée très différemment dans [27], en utilisant la théorie de contrôle virtuel, vient des méthodes de tir (voir l'analyse dans [21] [17]). Soit une équation différentielle ordinaire du second ordre avec des données aux limites aux temps 0 et T,

$$u'' = f(u), \quad u(0) = u^0, \quad u(T) = u^1. \tag{12}$$

La méthode de tir introduit le **problème de Cauchy** avec la donnée initiale u^0 et une vitesse initiale U donnée :

$$u_0'' = f(u_0), \quad u_0(0) = u^0, \quad u_0'(0) = U. \tag{13}$$

Pour une fonction f lipschitzienne, pour toute valeur de U, le problème (13) a une solution unique, au moins localement ([AF 1 220]). Notons la $u_0(t\,;\,u^0, U)$. La solution u de (12) est la solution u_0 de (13) pour la valeur $\bar{U}$ de U définie par $u^0\left(1, u^0, \bar{U}\right) = u^1$. Dans notre exemple, on voit bien que U représente une pente, donc un angle de tir, ce qui explique le nom de la méthode. L'équation non linéaire $u_0(1, u^0, U) = u^1$ pour l'inconnue U peut être résolue par une méthode de Newton. Notons que même lorsque le problème aux limites (12) est bien posé, il se peut que le système dynamique (13) possède des solutions exponentiellement croissantes. Il se peut aussi qu'il ne possède pas de solution sur tout l'intervalle désiré. Ce problème peut être partiellement surmonté par la méthode de tir multiples [24] [45].

Revenons à l'équation différentielle ordinaire du premier ordre pour la fonction $u : [0, T] \to \mathbb{R}^d$:

$$u' = f(u), \quad u(0) = u^0, \quad t \in [0, T]. \tag{14}$$

Pour cette équation, il n'y a pas d'horizon à atteindre, mais la méthode de tirs multiples permet de créer des horizons intermédiaires. Partageons l'intervalle $[0, 1]$ en $I + 1$ sous-intervalles $[T_0 = 0, T_1], \ldots, [T_I, T_I{+}1 = T]$, et donnons-nous $I + 1$ valeurs $U_0, \ldots, U_I$. Introduisons les problèmes de Cauchy dans les sous-intervalles (qui peuvent être résolus en parallèle)

$$u_i' = f(u_i), \quad u_i(T_i) = U_i.$$

La solution de ce problème est notée $u_i(t\,;\,U_i)$. La donnée initiale et la continuité de la solution aux temps T_i imposent les contraintes

$$U_0 = u^0, \quad U_{i+1} = u_i(T_{i-1}\,;\,U_i), \quad i = 0, \ldots, I-1. \tag{15}$$

Les $I + 1$ relations (15) forment un système non linéaire, que nous écrivons sous forme compacte $\mathfrak{F}(U) = 0$, avec

$$\boldsymbol{U} = \begin{pmatrix} U_0 \\ U_1 \\ \vdots \\ U_I \end{pmatrix}, \quad \mathfrak{F}(\boldsymbol{U}) = \begin{pmatrix} u^0 - U_0 \\ u^0(T_1\,;\,U_0) - U_1 \\ \vdots \\ u_{I-1}(T_I\,;\,U_{I-1}) - U_I \end{pmatrix}.$$

Appliquons à ce système non linéaire la méthode de Newton

$$\mathfrak{F}'\left(\boldsymbol{U}^n\right)\left(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n\right) + \mathfrak{F}\left(\boldsymbol{U}^n\right) = 0.$$

Calculons la dérivée de $\mathfrak{F}$:

$$\mathfrak{F}'(\boldsymbol{U})\tilde{\boldsymbol{U}} = \begin{pmatrix} 0 \\ \partial_2 u_0(T_1\,;\,U_0)\tilde{U}_0 \\ \vdots \\ \partial_2 U_{I-1}(T_I\,;\,U_{I-1})\tilde{U}_{I-1} \end{pmatrix} - \tilde{U}.$$

L'itération de Newton est donc donnée pour tout i par

$$U_{i+1}^{n+1} = u_i\left(T_{i+1}\,;\,U_i^n\right) + \partial_2 u_i\left(T_{i+1}\,;\,U_i^n\right)\left(U_i^{n+1} - U_i^n\right). \tag{16}$$

Pratiquement, la dérivée de u_i par rapport à la valeur initiale est calculée par (voir [1])

$\partial_2 u_i(T_{i+1}\,;\,U_i)\tilde{U}_i = \tilde{u}_i\left(T_{i+1}\,;\,\tilde{U}_i\right)$, où $\tilde{u}_i$ est la solution du problème linéaire

$$\tilde{u}_i' = \partial_2 f\left(t, u_i(t)\right)\tilde{u}_i, \quad \tilde{u}_i(T_i) = \tilde{U}_i.$$

Cet algorithme de Newton converge quadratiquement lorsque l'initialisation de l'algorithme est déjà assez proche de la solution, comme l'ont établi Chartier et Philippe dans [6]. Notons que la résolution des équations obtenues par la méthode de tir peut être effectuée par des méthodes quasi-Newton (type Broyden ou epsilon-algorithme) qui ne nécessitent pas le calcul, toujours difficile, du Jacobien.

L'algorithme pararéel proposé par Lions, Maday, et Turinici en 2001 [27] est très similaire à une méthode de tir. Il est défini à l'aide de deux opérateurs :

1. $G(T_{i+1}, T_i, U)$ qui donne une approximation grossière (et bon marché) de $u(T_{i+1})$ avec condition initiale $u(T_i) = U$.

2. $F(T_{i+1}, T_i, U)$ qui donne une approximation précise (et onéreuse) de la solution $u(T_{i+1})$ avec condition initiale $u(T_i) = U$.

L'algorithme démarre avec une approximation initiale $\boldsymbol{U}^0 = \left\{U_i^0\right\}$ aux instants $T_0, T_1, \ldots, T_I$, calculée par exemple avec G, et fait ensuite pour $n = 0, 1, \ldots$ l'itération de correction

$$U_{i+1}^{n+1} = G\left(T_{i+1}, T_i, U_i^{n+1}\right) + F\left(T_{i+1}, T_i, U_i^n\right) - G\left(T_{i+1}, T_i, U_i^n\right). \tag{17}$$

Dans cette itération, la partie onéreuse du calcul, représentée par F, peut être effectuée en parallèle, et seule la partie bon marché G est séquentielle. La méthode est très simple à utiliser : il suffit d'un seul solveur, qu'on appelle soit avec une haute résolution, soit avec une faible résolution, sur des sous-intervalles en temps, et on combine le résultat comme indiqué dans la formule (17). Cette simplicité d'utilisation explique en partie l'intérêt suscité par cette méthode.

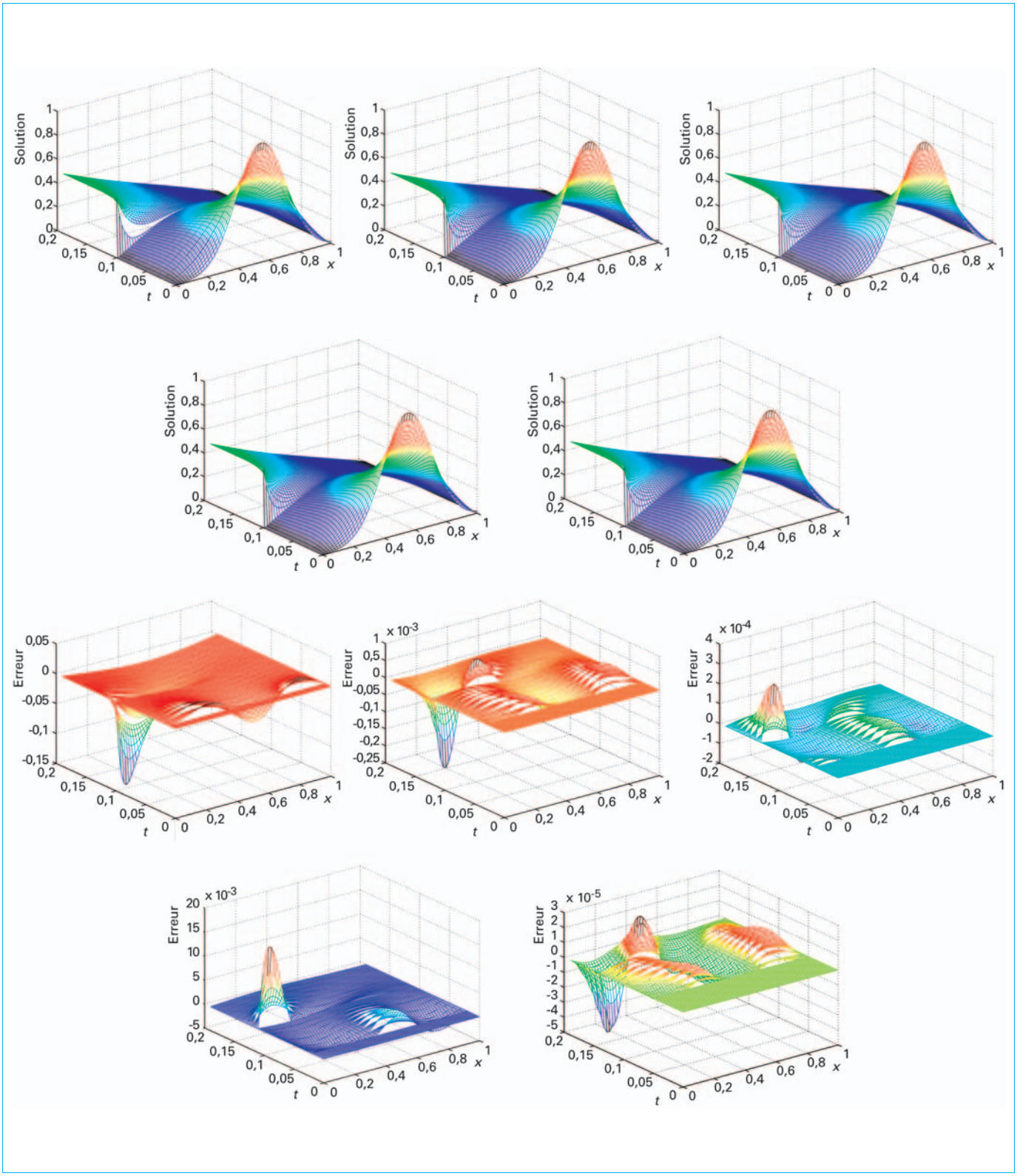


Figure 12 – Représentation des solutions et des erreurs pour l'algoritme pararéel

Si l'on réécrit (17) sous la forme

$$U_{i+1}^{n+1} = F\left(T_{i+1}, T_i, U_i^n\right) + G\left(T_{i+1}, T_i, U_i^{n+1}\right) - G\left(T_{i+1}, T_i, U_i^n\right),$$

et que l'on compare avec la méthode de tir multiple avec résolution par une méthode de quasi-Newton (16), on découvre que l'algorithme parareél est une méthode de tir multiple, où le solveur F est le calcul exact de la solution et la différence $G\left(T_{i+1}, T_i, U_i^{n+1}\right) - G\left(T_{i+1}, T_i, U_i^n\right)$ est une approximation naturelle du jacobien $\partial_2 u_i\left(T_{i+1}; U_i^n\right)\left(U_i^{n+1} - U_i^n\right)$ sur la grille grossière des T_i, voir aussi [21]. On peut démontrer que cette méthode converge superlinéairement vers la solution et, sous certaines hypothèses, la solution peut être calculée plus rapidement en parallèle qu'avec un seul processeur, voir aussi [17].

Voici une implémentation simple de l'algorithme parareél, en utilisant le programme Heat.m précédent, pour résoudre le même problème modèle, mais maintenant en parallélisant seulement en temps :

```
BarTime ;                              % compute reference solution
ue=u ;
I=20 ;                                 % number of coarse time points
DT=T/I ;
U(1,:)=u0 ; F(1,:)=u0 ;
for i=1:I-1                            % compute initial coarse
   u=Heat(0,1,U(i,:),gg(i*M/I),gd(i*M/I),DT) ;
   U(i+1,:)=u(end,:) ;
end ;
G=U ;
for n=1:I
   err(n)=0 ;
   for i=1:I-1                         % can start loops at n
      tf=(i-1)*M/I+1:i*M/I ;
      u=Heat(0,1,U(i,:),gg(tf),gd(tf),DT) ;
      F(i+1,:)=u(end,:) ;
      mesh(x,t([tf tf(end)+1]),u) ;
      err(n)=max([err(n) norm(ue([tf tf(end)+1],:)-u,'inf')]) ;
      hold on
   end ;
   hold off ;
   xlabel('x') ; ylabel('t') ; zlabel('solution') ;
   pause
   for i=1:I-1
      u=Heat(0,1,U(i,:),gg(i*M/I),gd(i*M/I),DT) ;
      U(i+1,:)=F(i+1,:)+u(end,:)-G(i+1,:) ;
      G(i+1,:)=u(end,:) ;
   end ;
end ;
```

Les itérations successives sont tracées figure **12**. Dans la première série nous voyons la solution approchée elle-même, tandis que dans la seconde série sont tracées les erreurs.

Sur la figure **13** nous montrons finalement que l'algorithme converge linéairement dans ce cas, comme il a été prouvé dans [21].

Pour avoir une idée du nombre d'itérations effectivement nécessaires dans cet exemple, on peut calculer l'erreur d'approximation du schéma : on calcule avec le programme BarTime.m une solution de référence $\boldsymbol{u}_r$ sur un maillage beaucoup plus fin, et qui est considérée comme la solution exacte. L'erreur du schéma est mesurée par la norme (maximum en temps et en espace par exemple) de la différence entre $\boldsymbol{u}$ et $\boldsymbol{u}_r$ sur la grille de $\boldsymbol{u}$. La commande suivante de Matlab réalise cette erreur :

```
> max(max(abs(ur(1:4:end,1:4:end)-ue)))
ans =
    0.1256
```

On voit alors sur la figure **13** que l'erreur 0,1256 est atteinte par l'algorithme parareél en 2 itérations. Si l'on utilisait 20 processeurs, on pourrait alors calculer cette solution 10 fois plus vite en parallélisant en temps qu'avec un seul processeur, si l'on néglige le coût de la solution grossière et de la communication.

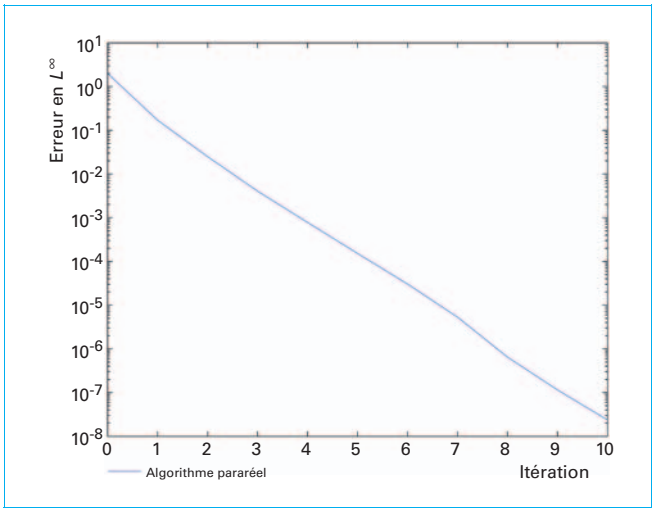


Figure 13 – Erreur L^∞ en fonction des itérations pour l'équation de la chaleur traitée par l'algorithme parareél

2.4 Parallélisation en espace et en temps

Pour finir cet exposé, nous nous proposons de réaliser un parallélisme espace-temps pour la résolution de l'équation de la chaleur (6), en couplant l'approche de Schwarz relaxation d'onde et l'approche parareélle. Une façon naturelle de coupler les deux approches est de remplacer dans l'algorithme parareél le solveur fin par un solveur de relaxation d'ondes, voir [31].

Nous présentons une stratégie différente, proposée dans [19]. Le domaine Ω x $[0,T]$ est décomposé en sous-domaines espace-temps $\Omega_{ji} = \Omega_j$ x $[T_i,\ T_{i+1}]$ représentés figure **14**.

Pour résoudre (6), nous introduisons dans chaque sous-domaine $\Omega_{j,i}$ un problème type dont la solution est notée $u_{j,i}$:

$$\begin{aligned} \partial_t u_{j,i} - \partial_{xx} u_{j,i} &= 0 \quad \text{dans } \Omega_{j,i}, \\ u_{j,i}(\cdot, T_i) &= U_{j,i} \quad \text{dans } \Omega_j, \\ \mathfrak{B}_j^- u_{j,i}\left(x_j^-, \cdot\right) &= g_{j,i}^- \quad \text{dans } [T_i, T_{i+1}], \\ \mathfrak{B}_j^+ u_{j,i}\left(x_j^+, \cdot\right) &= g_{j,i}^+ \quad \text{dans } [T_i, T_{i+1}]. \end{aligned} \tag{18}$$

Ici $\mathfrak{B}_j^+$ représente un opérateur sur la frontière, comme l'identité pour une condition de Dirichlet, une dérivée en x pour une condition de Neumann, ou encore une combinaison des deux pour une condition de Robin comme au paragraphe 2.2.

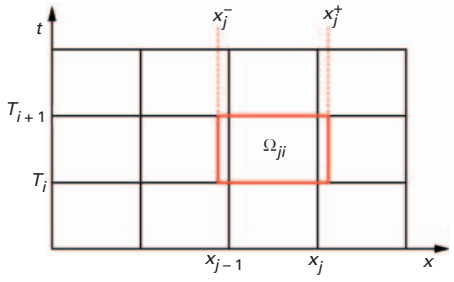


Figure 14 – Décomposition en espace-temps pour l'algorithme parareél-Schwarz relaxation d'onde

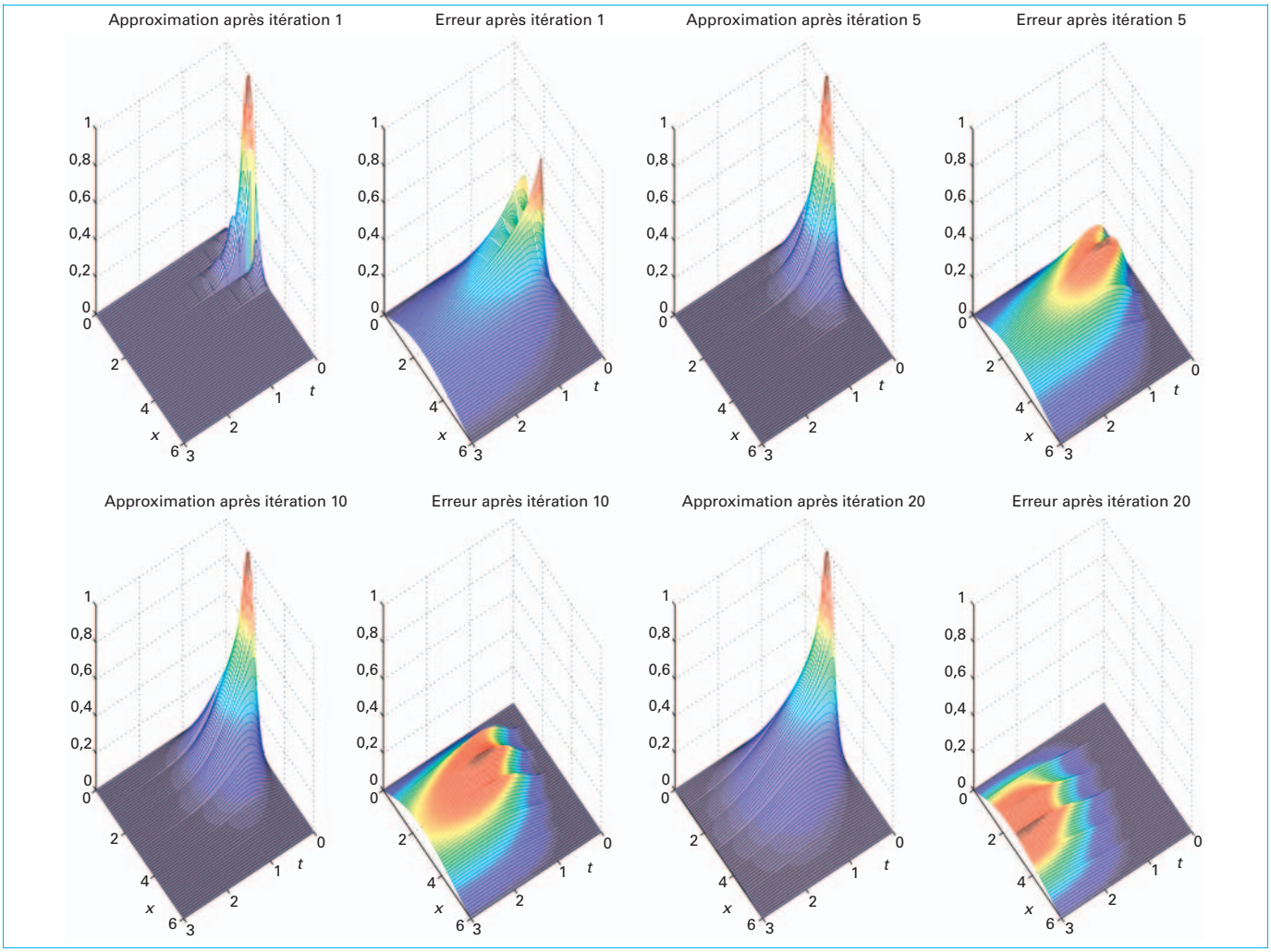


Figure 15 – Algorithme pararéel-Schwarz relaxation d'onde avec conditions de transmission de Dirichlet

Il faut maintenant calculer d'une manière itérative des conditions initiales et des conditions de transmission de plus en plus précises pour chaque sous-domaine en espace-temps. Pour cela, l'algorithme pararéel-Schwarz-relaxation d'onde approche les conditions initiales par une méthode pararéelle et les conditions de transmission par une méthode de Schwarz relaxation d'onde.

Nous proposons la stratégie suivante : des conditions initiales $U_{j,i}^0(x)$ au temps T_i et des conditions aux limites $g_{j,i}^- = \mathfrak{B}_j^- u_{j-1,i}^0$ et $g_{j,i}^+ = \mathfrak{B}_j^+ u_{j+1,i}^0$ sont données pour chaque Ω_{ji}. Nous calculons maintenant pour tout $n = 0, 1, 2,\ldots$

1. des approximations précises

$$u_{j,i}^{n+1}(x,t) := F_{j,i}\left(U_{j,i}^n, \mathfrak{B}_j^- u_{j-1,i}^n, \mathfrak{B}_j^+ u_{j+1,i}^n\right)$$

dans tous les sous-domaines espace-temps Ω_{ji}, en parallèle ;

2. Pour $i = 0, 1,\ldots$, de nouvelles conditions initiales, en utilisant une correction pararéelle en temps,

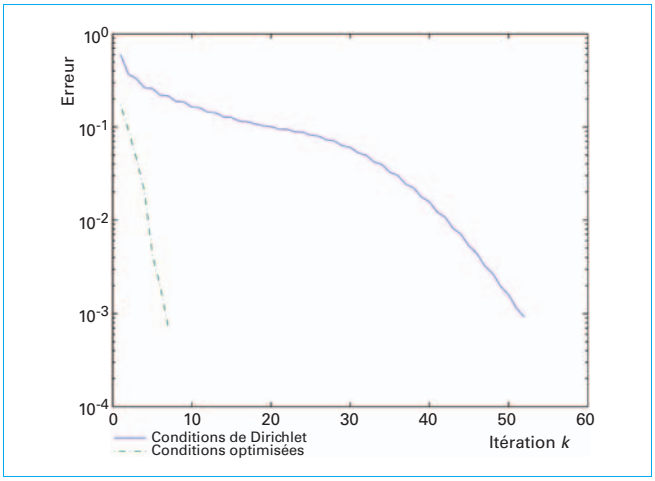


Figure 16 – Comparaison de l'algorithme couplé pararéel-Schwarz relaxation d'onde avec conditions de transmission de Dirichlet ou optimisées

$$U_{j,i+1}^{n+1} = u_{j,i}^{n+1}\left(\cdot, T_{j+1}\right) + G_{j,i}\left(U_{j,i}^{n+1}, \mathfrak{B}_j^- u_{j-1,i}^{n+1}, \mathfrak{B}_j^+ u_{j+1,i}^{n+1}\right)$$
$$- G_{j,i}\left(U_{j,i}^{n}, \mathfrak{B}_j^- u_{j-1,i}^{n}, \mathfrak{B}_j^+ u_{j+1,i}^{n}\right).$$

Nous illustrons sur la figure **15** le comportement d'une première variante de l'algorithme avec des conditions de transmission de type Dirichlet.

Nous voyons sur cette figure que l'algorithme est plus efficace dans la direction du temps que dans la direction spatiale. Cette performance moindre est due aux conditions de transmission de Dirichlet qui ne sont pas très efficaces. Elle peut être améliorée en utilisant les algorithmes Schwarz relaxation d'onde optimisés présentés au paragraphe 2.2. Les courbes de convergence pour les deux méthodes sont tracées sur la figure **16**.

Même dans le contexte plus complexe d'un découpage espace-temps, couplé à un algorithme pararéel, l'algorithme de Schwarz optimisé surclasse nettement l'algorithme classique.

Méthodes de décomposition de domaines

par **Martin J. GANDER**
Professeur de mathématiques
Section de Mathématiques, Université de Genève

et **Laurence HALPERN**
Professeur de Mathématiques
Laboratoire Analyse, Géométrie et Applications, Université Paris 13

Sources bibliographiques

[1] ARNOLD (V.). – *Équations différentielles ordinaires*. Éditions MIR, Moscou (1974).

[2] BJORHUS (M.). – *Semi-discrete subdomain iteration for hyperbolic systems*. Tech. Rep. 4, NTNU (1995).

[3] BOURGAT (J.-F.), GLOWINSKI (R.), LE TALLEC (P.) et VIDRASCU (M.). – *Variational formulation and algorithm for trace operator in domain decomposition calculations*. in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, PA, SIAM, pp. 3-16 (1989).

[4] CAI (X.-C.) et SARKIS (M.). – *A restricted additive Schwarz preconditioner for general sparse linear systems*. SIAM Journal on Scientific Computing, 21, pp. 239-247 (1999).

[5] CHAN (T.F.) et MATHEW (T.P.). – *Domain decomposition algorithms*. in Acta Numerica 1994. Cambridge University Press, pp. 61-143 (1994).

[6] CHARTIER (P.) et PHILIPPE (B.). – *A parallel shooting technique for solving dissipative ODEs*. Computing, 51, pp. 209-236 (1993).

[7] CHENEY (E.W.). – *Introduction to Approximation Theory*. McGraw-Hill Book Co., New York (1966).

[8] DESPRÉS (B.). – *Méthodes de décomposition de domaines pour les problèmes de propagation d'ondes en régime harmonique*. PhD thesis, Université Paris IX Dauphine (1991).

[9] DINH (Q.V.), MANTEL (B.), PERIAUX (J.) et GLOWINSKI (R.). – *Approximate solution of the Navier-Stokes equations for incompressible viscous fluids, related domain decomposition methods*. vol. 1005 of Lect. notes in Math., Springer, pp. 46-86 (1983).

[10] DRYJA (M.). – *A capacitance matrix method for Dirichlet problem on polygon region*. Numer. Math., 39, pp. 51-64 (1982).

[11] DRYJA (M.) et WIDLUND (O.B.). – *An additive variant of the Schwarz alternating method for the case of many subregions*. Tech. Rep. 339, also Ultracomputer Note 131, Department of Computer Science, Courant Institute (1987).

[12] —. – *Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems*. Comm. Pure Appl. Math., 48, pp. 121-155 (1995).

[13] EFSTATHIOU (E.) et GANDER (M.J.). – *Why Restricted Additive Schwarz converges faster than Additive Schwarz*. BIT Numerical Mathematics, 43, pp. 945-959 (2003).

[14] EVANS (L.C.). – *Partial differential equations*. vol. 19 of Graduate studies in Mathematics, AMS (1998).

[15] GANDER (M.J.). – *Optimized Schwarz methods*. SIAM J. Numer, Anal., 44, pp. 699-731 (2006).

[16] —. – *Schwarz methods over the course of time*. Electron. Trans. Numer. Anal, 31, pp. 228-255 (2008).

[17] GANDER (M.J.) et HAIRER (E.). – *Nonlinear convergence analysis for the parareal algorithm*. in Domain Decomposition Methods in Science and Engineering XVII, O. B. Widlund and D. E. Keyes, eds., vol. 60 of Lecture Notes in Computational Science and Engineering, Springer, pp. 45-56 (2008).

[18] GANDER (M.J.), HALPERN (L.) et NATAF (F.). – *Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation*. in Eleventh international Conference of Domain Decomposition Methods, C.-H. Lai, P. Bjørstad, M. Cross, and O. Widlund, eds., ddm. org (1999).

[19] GANDER (M.J.), JIANG (Y.-L.) et LI (R.-J.). – *Parareal schwarz waveform relaxation methods*. in Domain Decomposition Methods in Science and Engineering XX, O. B. Widlund and D. E. Keyes, eds., Lecture Notes in Computational Science and Engineering, Springer, submitted (2011).

[20] GANDER (M.J.) et STUART (A.M.). – *Space-time continuous analysis of waveform relaxation for the heat equation*. SIAM J. Sci. Comput., 19, pp. 2014-2031 (1998).

[21] GANDER (M.J.) et VANDEWALLE (S.). – *Analysis of the parareal time-parallel time-integration method*. SIAM J. Sci. Comput., 29, pp. 556-578 (2007).

[22] GANDER (M.J.) et ZHAO (H.). – *Overlapping Schwarz waveform relaxation for the heat equation in n-dimensions*. BIT, 42, pp. 779-795 (2002).

[23] HAYNSWORTH (E.V.). – *On the Schur complement*. Tech. Rep. 20, Basel Mathematical Notes (June 1968).

[24] KELLER (H.B.). – *Numerical Solution for Two-Point Boundary-Value Problems*. Dover Publications, Inc., New York (1992).

[25] LELARASMEE (E.), RUEHLI (A.E.) et SANGIOVANNI-VINCENTELLI (A. L.). – *The waveform relaxation method for time-domain analysis of large scale integrated circuits*. IEEE Trans. on CAD of IC and Syst., 1, pp. 131-145 (1982).

[26] LINDELOF (E.). – *Sur l'application des méthodes d'approximations successives à l'étude des intégrales réelles des équations différentielles ordinaires*. Journal de Mathématiques Pures et Appliquées, 10, pp. 117-128 (1894).

[27] LIONS (J.-L.), MADAY (Y.) et TURINICI (G.). – *A parareal in time discretization of PDE's*. C. R. Acad. Sci. Paris, Serie I, 332, pp. 661-668 (2001).

[28] LIONS (P.-L.). – *On the Schwarz alternating method I*. in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Philadelphia, PA, SIAM, pp. 1-42 (1988).

[29] —. – *On the Schwarz alternating method II : Stochastic interpretation and orders properties*. in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, PA, SIAM, pp. 47-70 (1989).

[30] —. – *On the Schwarz alternating method III : A variant for nonoverlapping subdomains*. in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, held in Houston, Texas, March 20-22, 1989, T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, PA, SIAM, pp. 202-223 (1990).

[31] MADAY (Y.) et TURINICI (G.). – *The parareal in time iterative solver : a further direction to parallel implementation*. in Proceedings of the 15th international domain decomposition conference, R. Kornhuber, R. H. W. Hoppe, J. Périaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer LNCSE, pp. 441-448 (2003).

[32] MANDEL (J.) et BREZINA (M.). – *Balancing domain decomposition for problems with large jumps in coefficients*. Math. Comp., 65, pp. 1387-1401 (1996).

[33] MANDEL (J.) et TEZAUR (R.). – *Convergence of a substructuring method with Lagrange multipliers*. Numer. Math., 73, pp. 473-487 (1996).

[34] MILLER (K.). – *Numerical analogs to the Schwarz alternating procedure*. Numer. Math., 7, pp. 91-103 (1965).

[35] PICARD (E.). – *Mémoire sur la théorie des équations aux dérivée partielles et la méthodes des approximations successives*. Journal de Mathématiques Pures et Appliquées, 6, pp. 145-210 (1890).

[36] —. – *Sur l'application des méthodes d'approximations successives à l'étude de certaines équations différentielles ordinaires*. Journal de Mathématiques Pures et Appliquées, 9, pp. 217-271 (1893).

[37] PRZEMIENIECKI (J.S.). – *Matrix structural analysis of substructures*. Am. Inst. Aero. Astro. J., 1, pp. 138-147 (1963).

[38] QUARTERONI (A.) et VALLI (A.). – *Domain Decomposition Methods for Partial Differential Equations*. Oxford Science Publications (1999).

[39] RUDIN (W.). – *Real and Complex Analysis*. Mc Graw-Hill (1966).

[40] SAAD (Y.). – *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company (1996).

[41] SCHWARZ (H.A.). – *Über einen Grenzübergang durch alternierendes Verfahren*. Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15, pp. 272-286 (1870).

[42] SMITH (B.F.), BJØRSTAD (P.E.) et GROPP (W.). – *Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press (1996).

[43] TOSELLI (A.) et WIDLUND (O.). – *Domain Decomposition Methods – Algorithms and Theory*. vol. 34 of Springer Series in Computational Mathematics, Springer (2004).

[44] TRAN (M.B.). – *Schwarz domain decomposition methods for linear and nonlinear problems*. PhD thesis, Université Paris 13 (September 2011).

[45] WEISS (R.). – *Convergence of shooting methods*. BIT, 13, pp. 470-475 (1973).

[46] ZHANG (F.). – *The Schur complement and its applications*. vol. 4 of Numerical Methods and Algorithms, Springer (December 2005).

À lire également dans nos bases

CABANE (R.). – *Méthodes numériques en algèbre linéaire*. [AF 485] Mathématiques pour l'ingénieur (1995).

MEURANT (G.). – *Méthodes de Krylov pour la résolution du systèmes linéaires*. [AF 488] Mathématiques pour l'ingénieur (2007).

SPITERI (P.). – *Méthode des différences finies pour les EDP stationnaires*. [AF 500] Mathématiques pour l'ingénieur (2002).

SPITERI (P.). – *Approche variationnelle pour la méthode des éléments finis*. [AF 503] Mathématiques pour l'ingénieur (2003).

BREZINSKI (C.). – *Méthodes numériques de base - Analyse numérique*. [AF 1 220] Mathématiques pour l'ingénieur (2006).

Sites Internet

Premiers exposés de la conférence internationale sur les méthodes de décomposition de domaines — http://www.ddm.org/conferences.html