Méthodes de décomposition de domaines

Martin J. Gander [*] Laurence Halpern [†]

24 avril 2012

English Abstract : Domain Decomposition Methods

More and more complicated physical phenomena are nowadays modeled by partial differential equations. These equations are then discretized and solved approximately on a computer. Processing speeds and memory sizes have steadily increased over the last decades, but finally reached a few years ago their long predicted plateau which can not be overcome any more with the same technology. Hence the classical single processor unit has been replaced by multi-core processors in desktop computers in order to further increase performance, and by large scale parallel computers with distributed memory and many processors for industrial simulations (the biggest current computer is a FUJITSU with 548352 processors and peak performance of 10 petaflops). In order to use these large scale parallel computers effectively, new parallel algorithms had to be developed. Domain decomposition methods are naturally adapted to run in parallel, and especially suited for complex modeling problems. They allow users to structure their problem and domain on which it is posed naturally by forming subdomains. For each subdomain, both the data (geometry, coefficients) and the computations are then handled by a dedicated processor of the parallel computer, and they communicate among each other by sending and receiving so called transmission conditions between subdomains. Domain decomposition methods are however much more than just a parallelization tool : they permit for example also the modeling and coupling of different physics in different subdomains. This monograph gives a detailed overview over the main existing classes of domain decomposition methods developed over the course of time. In order to guide the reader, we chose a simple model problem for which we can show in all detail the main concepts and mechanisms, and we illustrate them each time with small Matlab programs.

Keywords : Domain Decomposition, Schwarz Methods, Schur Methods, FETI, Neumann-Neumann, Waveform Relaxation, Parareal Algorithm

Résumé

Des phénomènes physiques de plus en plus complexes sont modélisés par des systèmes d'équations aux dérivées partielles. Ces équations sont elles-mêmes *discrétisées* pour être résolues par ordinateur. Si les progrès technologiques ont constamment amélioré les performances des processeurs, ces performances atteignent aujourd'hui un plateau en terme de mémoire et de puissance de calcul. Les ordinateurs monoprocesseurs ont laissé la place aux quadricœurs pour les ordinateurs de bureau, aux ordinateurs parallèles à mémoire distribuée pour les calculs actuels qui nécessitent de calculer en parallèle sur un grand nombre de processeurs (le plus gros ordinateur actuel est un FUJITSU formé de 548 352 processeurs, qui développe une puissance de calcul de 10 petaflops). Pour utiliser au mieux ces *calculateurs parallèles* formés d'un grand nombre de processeurs, il fallait inventer de nouveaux *algorithmes parallèles*. Les méthodes de décomposition de domaine sont naturellement adaptées à cette

[*]Section de Mathématiques. Université de Genève. 2-4 rue du Lièvre, CP 64, CH-1211 Genève.
[†]LAGA et CNRS UMR7539. Université Paris 13. Avenue J.B. Clément, 93430 Villetaneuse. FRANCE

problématique, pour des systèmes complexes issus de la modélisation. Elles permettent de structurer et gérer le domaine géométrique en sous-domaines. Pour chaque sous-domaine, les données du problème (géométrie, coefficients) et les calculs sont dédiés à un *nœud* du multiprocesseur, et la communication entre les nœuds reproduit divers types de transmission entre les sous-domaines. Les méthodes de décomposition de domaines sont encore bien plus qu'un outil de calcul parallèle. Elles permettent de modéliser le couplage entre différentes physiques, mais ceci est une autre histoire... Cet article se propose de donner un aperçu des diverses méthodes de décomposition de domaines, inventées au fil du temps dans divers contextes. Pour guider le lecteur, nous nous donnerons un problème modèle simple sur lequel nous introduirons les principaux concepts et mécanismes, chaque fois illustrés par des petits programmes Matlab.

Mots-clés : Décomposition de domaines, Méthodes de Schwarz, Méthodes de Schur, FETI, Neumann-Neumann, Relaxation d'ondes, Algorithme Pararéel

Table des matières

1 Introduction

Tous les problèmes d'ingénierie aujourd'hui sont résolus en parallèle sur des ordinateurs composés de centaines, voire des milliers d'ordinateurs. Cet article se propose d'exposer les méthodes de décomposition de domaines susceptibles de s'appliquer à ces nouveaux outils. Emile Picard nous enseigne dans [36] que pour comprendre une théorie, il est bon d'avoir en tête un problème modèle :

> Les méthodes d'approximation dont nous faisons usage sont théoriquement susceptibles de s'appliquer à toute équation, mais elles ne deviennent vraiment intéressantes pour l'étude des propriétés des fonctions définies par les équations différentielles que si l'on ne reste pas dans les généralités et si l'on envisage certaines classes d'équations.

Nous choisirons donc dans tout cet exposé un fil conducteur, l'équation de la chaleur

$$\partial_t u - \Delta u = f, \tag{1.1}$$

représentant les variations en temps et en espace de la température d'un corps emplissant le domaine Ω, soumis à une source de chaleur f (qui sera appelée second membre), avec une température initiale donnée dans tout le domaine, et des conditions aux limites sur le bord du domaine $\partial\Omega$, par exemple de Dirichlet (la température est fixée), soit $u = g$. $\partial_t u$ est la dérivée en temps de u, Δ est l'opérateur de Laplace, $\Delta u = \partial_{11} u + \partial_{22} u + \partial_{33} u$. Pour calculer sur un ordinateur une solution approchée de cette équation, on peut commencer par une *semi-discrétisation en temps*. Le schéma le plus simple est le schéma d'Euler implicite (voir le dossier AF 1 220 des Techniques de l'Ingénieur). Partageons l'intervalle de temps $[0, T]$ en sous-intervalles $[t_n, t_{n+1}]$ de longueur Δt. Notons $u_n(x)$ l'approximation de u à l'instant t_n au point x, calculée par la formule de récurrence

$$\frac{1}{\Delta t} u_{n+1} - \Delta u_{n+1} = \frac{1}{\Delta t} u_n + f_{n+1},$$

où f_{n+1} représente $f(x, t_{n+1})$. Pour passer du temps t_n au temps t_{n+1}, il faut donc résoudre l'*équation elliptique*

$$(\eta - \Delta) u = f$$

dans le domaine Ω, où f est maintenant une fonction indépendante du temps.

La discrétisation en espace de cette équation par une méthode de type éléments finis ou volumes finis, mène à un système linéaire (voir les dossiers AF 500 et AF503 des Techniques de l'Ingénieur). Lorsque la taille du domaine de calcul est très grande, ou la discrétisation très fine, la taille du système linéaire excède les capacités de stockage et de calcul d'un seul ordinateur, si puissant soit-il. L'idée la plus simple pour remédier à ce problème est de décomposer le système linéaire en sous-systèmes, dont chacun est suffisamment petit pour être résolu très rapidement sur un nœud d'un système d'ordinateurs (*divide et impera*). Cela peut se faire au niveau purement informatique, mais il est plus fructueux de revenir en amont et de développer une stratégie au niveau du problème mathématique. Cette démarche est souvent réclamée par la géométrie elle-même (assemblage de structures par exemple). Le domaine de calcul est alors partagé en sous-domaines, chacun assigné à un nœud de la grappe de calcul. Les échanges entres les sous-domaines sont effectués par des conditions de transmission, et traduits par des échanges entre les processeurs. La résolution du problème de départ est alors réalisée en itérant entre les sous-domaines, et les sous-domaines peuvent même être en espace-temps.

Toutes ces méthodes sont des méthodes de décomposition de domaines. Elles ont pour fondateur H.A. Schwarz qui en écrivit une première version en 1870 [41]. Elles ont donné lieu à une intense activité scientifique depuis l'avènement des calculateurs parallèles. Elles sont utilisées pour des calculs de pneumatiques, d'automobiles, de structures sismiques, de navette spatiale, de reconnaissance de forme, d'environnement, de météorologie, d'astrophysique, de médecine, et tant d'autres.

Leur champ d'utilisation est même plus large : si par exemple le modèle a des propriétés physiques différentes dans différentes parties du domaine, les méthodes de décomposition de domaine sont un outil naturel pour leur traitement. Mentionnons par exemple la jonction d'une poutre et d'une plaque, le couplage entre l'océan et l'atmosphère.

2 Historique

2.1 Méthode de Schwarz

Les méthodes de Schwarz trouvent leur origine dans un algorithme alterné imaginé par H.A. Schwarz pour démontrer l'existence de fonctions harmoniques (*i.e.* solutions deux fois continuement dérivables de l'équation de Laplace $\Delta u = 0$) dans un domaine composite, avec une valeur au bord prescrite g. Il commence par considérer un domaine composé d'un disque T_1 et d'un carré T_2 (voir figure 2.1 à gauche), domaines pour lesquels une solution explicite au moyen de séries de Fourier existe.

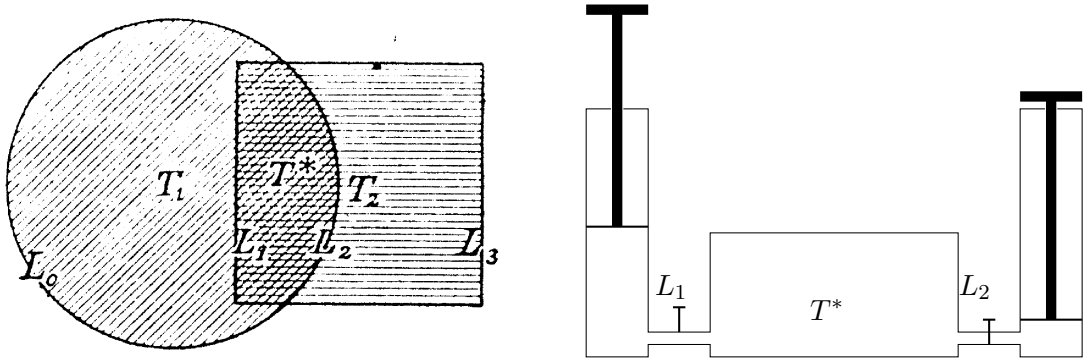


Figure 2.1 – Figure originale de 1870 pour l'algorithme alterné de H.A. Schwarz, et son interprétation physique

Les frontières respectives de ces deux domaines sont notées ∂T_1 et ∂T_2. La partie de ∂T_1 située dans T_2 est notée L_2, le complémentaire est L_0. La partie de ∂T_2 située dans T_1 est notée L_1, le complémentaire est L_3. Le domaine global T est l'union des deux domaines T_1 et T_2, sa frontière ∂T est constituée de L_0 et L_3. Pour montrer que le problème de Dirichlet

$$\Delta u = 0 \quad \text{dans } T, \quad u = g \quad \text{sur } \partial T, \tag{2.1}$$

a une solution, Schwarz introduit une suite de solutions de problèmes de Dirichlet sur les domaines T_i alternativement. Pour illustrer son propos, il introduit une pompe à air où chaque T_i joue le rôle d'un cylindre, T^* est le réservoir intermédiaire, L_1 et L_2 sont les soupapes (voir figure 2.1 à droite). L'algorithme est initialisé par une donnée constante sur L_2, $\underline{u} = \inf_{L_0} g$. La première étape définit une fonction u_1 harmonique dans T_1, égale à g sur L_0, et à $\underline{u}$ sur L_2.

Le principe du maximum (voir par exemple [14]) affirme qu'une fonction harmonique dans un domaine atteint son maximum sur le bord du domaine. De plus, si le domaine est connexe, et si le maximum est atteint en un point intérieur au domaine, la fonction est constante. En remplaçant la fonction par son opposé, on obtient les mêmes énoncés en remplaçant "maximum" par "minimum". Cet outil fondamental de l'analyse est utilisé par Schwarz pour montrer la convergence du processus, couplé à un lemme qui en découle, que nous énonçons ici sous la forme donnée dans [29] pour des géométries plus générales.

Lemme 2.1 *Soit w une fonction harmonique sur T_1, continue sauf en $\partial T_1 \cap \partial T_2$, prenant la valeur 0 sur L_0 et 1 sur L_2. Alors le maximum q_1 de w sur L_1 est strictement compris entre 0 et 1.*

La seule difficulté dans ce lemme est le traitement des points de jonction entre les frontières de T_1 et T_2 (voir [29]). Le même résultat est valable dans T_2 avec une constante q_2.

Par le principe du maximum, u_1 prend ses valeurs dans T_1, et en particulier sur L_1, entre $\underline{u}$ et $\overline{u} = \sup_{L_0} g$. À partir de u_1, Schwarz construit une fonction u_2 harmonique dans T_2, fixée à la valeur g sur L_3, et assujettie sur L_1 à la valeur de u_1 nouvellement calculée. De nouveau, u_2 prend ses valeurs

entre $\underline{u}$ et $\overline{u}$. Revenant maintenant dans T_1, il construit u_3, qui prend la valeur g sur L_0, et coïncide avec u_2 sur L_2. Par suite $u_3 - u_1$ est également harmonique dans T_1, s'annule sur L_0, et coïncide avec $u_2 - \underline{u}$ sur L_2. Cette dernière fonction étant positive, par le principe du maximum $u_3 - u_1$ est positive dans T_1, et plus petite que le maximum de $u_2 - \underline{u}$ sur L_2, donc que $G = \overline{u} - \underline{u}$. Par le lemme 2.1 appliqué à $(u_3 - u_1)/G$, la valeur de $u_3 - u_1$ sur L_1 est bornée par $q_1 G$. En itérant le processus, Schwarz résout alternativement pour $n \geq 0$ les problèmes

$$\Delta u_{2n+1} = 0 \text{ dans } T_1, \qquad u_{2n+1} = g \text{ sur } L_0, \qquad u_{2n+1} = u_{2n} \text{ sur } L_2, \tag{2.2}$$

$$\Delta u_{2n+2} = 0 \text{ dans } T_2, \qquad u_{2n+2} = g \text{ sur } L_3, \qquad u_{2n+2} = u_{2n+1} \text{ sur } L_1. \tag{2.3}$$

En appliquant le lemme 2.1 aux différences successives des itérées paires et impaires, qui sont nulles sur les bord extérieurs L_0 et L_3, il en déduit que

$$0 < u_{2n+1} - u_{2n-1} \leq G(q_1 q_2)^{n-1} \text{ sur } L_1, \qquad 0 < u_{2n+2} - u_{2n} \leq G(q_1 q_2)^{n-1} q_1 \text{ sur } L_2.$$

Il peut alors définir deux fonctions u et u' par les séries géométriques paires et impaires (convergentes !)

$$\begin{aligned} u &= u_1 + (u_3 - u_1) + (u_5 - u_3) + \dots \\ u' &= u_2 + (u_4 - u_2) + (u_6 - u_4) + \dots . \end{aligned}$$

Ces fonctions sont définies dans T_1 et T_2 respectivement, elles sont toutes deux harmoniques dans T^* et coïncident sur L_1 et L_2. Elles sont donc égales dans T^* par le principe du maximum, et il a défini ainsi une fonction harmonique sur tout le domaine $T = T_1 \cup T_2$.

Près de 100 ans plus tard, Miller propose un analogue numérique à l'algorithme de Schwarz, reposant sur l'utilisation de séries de Fourier discrètes pour le calcul rapide dans des domaines simples comme des rectangles et des disques [34]. Chaque itération de l'algorithme est alors peu coûteuse.

Dans une suite de trois articles aux trois premiers congrès internationaux en décomposition de domaines [28, 29, 30], P.L. Lions propose une extension magistrale des travaux de Schwarz, adaptée au calcul parallèle sur ordinateurs. Il introduit d'abord une forme parallèle de l'algorithme où la condition de transmission sur L_1 $u_{2n+2} = u_{2n+1}$ est remplacée par $u_{2n+2} = u_{2n-1}$ (pour une écriture moderne voir (3.2)). Lions étend également la preuve de convergence de Schwarz à un nombre quelconque de sous-domaines avec recouvrement, et il propose une nouvelle preuve de convergence au moyen d'opérateurs de projection dans un espace de Hilbert. Notons ici que même l'algorithme de Schwarz alterné peut être exécuté en parallèle dans le cas de plus de deux sous-domaines : il suffit de colorer les domaines de différentes couleurs de façon à ce que deux sous-domaines de même couleur ne se touchent pas. Ainsi tous les sous-domaines de même couleur peuvent être calculés en parallèle, avant de prendre la couleur suivante.

Dans cette série de communications aux premiers congrès décomposition de domaines, P.L. Lions propose également d'autres conditions de transmission entre les sous-domaines. Il établit ainsi la possibilité d'utiliser des sous-domaines sans recouvrement. Ce travail a ouvert de nouvelles perspectives, créant un lien avec les méthodes de sous-structuration, et permettant plus tard la construction d'algorithmes optimisés, avec ou sans recouvrement. Pour obtenir géométriquement une décomposition avec recouvrement en sous-domaines Ω_i, il est commode de partir d'une décomposition sans recouvrement $\widetilde{\Omega}_i$, et d'étendre chaque sous-domaine par une couche d'épaisseur ϵ (voir figure 2.2).

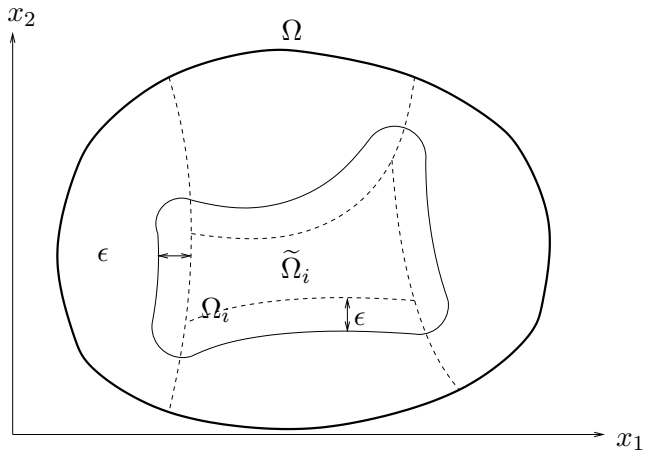


Figure 2.2 – Construction d'une décomposition avec recouvrement en dimension 2

Avec l'avènement des calculateurs parallèles, la méthode de Schwarz a connu un renouveau d'intérêt, et de nombreux développements [5, 42, 38, 43]. Les méthodes de Schwarz seront étudiées en détail dans le chapitre 3.

2.2 Méthode de Schur

Les méthodes de Schur, historiquement aussi appelées méthodes de sous-structuration, ont été introduites par Przemieniecki en 1963 [37], dans le contexte des calculs en aéronautique.

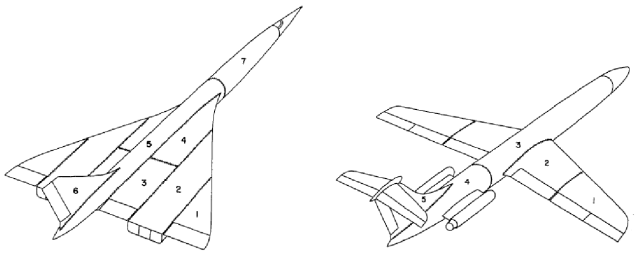


Figure 2.3 – La première décomposition de domaine sans recouvrement pour des calculs en aéronautiques par Przemieniecki

Partons d'une discrétisation par éléments finis de la structure décrite figure 2.3. Dans la notation originale de Przemieniecki, P est le vecteur des forces extérieures, K la matrice de rigidité, et le vecteur des déplacements U est solution du système linéaire

$$KU = P.$$

La structure est naturellement partitionnée en *sous-structures*, en introduisant des frontières intérieures, ce qui explique le nom historique de méthode de sous-structuration. Le vecteur U_i représente l'ensemble des degrés de liberté intérieurs aux sous-domaines, et le vecteur U_b l'ensemble des degrés de liberté correspondant aux frontières intérieures. La matrice K est également partitionnée par blocs, et le système se réécrit

$$K \begin{bmatrix} U_b \\ U_i \end{bmatrix} := \begin{bmatrix} K_{bb} & K_{bi} \\ K_{ib} & K_{ii} \end{bmatrix} \begin{bmatrix} U_b \\ U_i \end{bmatrix} = \begin{bmatrix} P_b \\ P_i \end{bmatrix}.$$

Przemieniecki motive physiquement son approche en décomposant les forces appliquées P et la solution

recherchée U en deux composantes

$$P = P^{(\alpha)} + P^{(\beta)} = \begin{bmatrix} P_b^{(\alpha)} \\ P_i \end{bmatrix} + \begin{bmatrix} P_b^{(\beta)} \\ 0 \end{bmatrix}, \tag{2.4}$$

$$U = U^{(\alpha)} + U^{(\beta)} = \begin{bmatrix} 0 \\ U_i^{(\alpha)} \end{bmatrix} + \begin{bmatrix} U_b \\ U_i^{(\beta)} \end{bmatrix}. \tag{2.5}$$

Par linéarité il obtient deux systèmes

$$(\alpha): \begin{bmatrix} K_{bb} & K_{bi} \\ K_{ib} & K_{ii} \end{bmatrix} \begin{bmatrix} 0 \\ U_i^{(\alpha)} \end{bmatrix} = \begin{bmatrix} P_b^{(\alpha)} \\ P_i \end{bmatrix} \quad (\beta): \begin{bmatrix} K_{bb} & K_{bi} \\ K_{ib} & K_{ii} \end{bmatrix} \begin{bmatrix} U_b \\ U_i^{(\beta)} \end{bmatrix} = \begin{bmatrix} P_b^{(\beta)} \\ 0 \end{bmatrix},$$

qui se réécrivent

$$(\alpha): \begin{cases} K_{bi}U_i^{(\alpha)} = P_b^{(\alpha)}, \\ K_{ii}U_i^{(\alpha)} = P_i, \end{cases} \qquad (\beta): \begin{cases} K_{bb}U_b + K_{bi}U_i^{(\beta)} = P_b^{(\beta)}, \\ K_{ib}U_b + K_{ii}U_i^{(\beta)} = 0. \end{cases}$$

Connaissant les forces P_i intérieures aux sous-structures, il résout d'abord la deuxième équation du système (α), ce qui correspond physiquement à calculer les déplacements associés à ces forces, en fixant à zéro les déplacements sur les interfaces

$$U_i^{(\alpha)} = K_{ii}^{-1} P_i.$$

En reportant $U_i^{(\alpha)}$ dans la première équation du système (α), il calcule les forces $P_b^{(\alpha)}$ produites sur les interfaces par ces déplacements

$$P_b^{(\alpha)} = K_{bi}K_{ii}^{-1}P_i,$$

puis les forces restantes agissant sur les interfaces

$$P_b^{(\beta)} := P_b - P_b^{(\alpha)} = P_b - K_{bi}K_{ii}^{-1}P_i\,,$$

justifiant ainsi la décomposition (2.4). Le système (β) calcule maintenant la réponse des structures aux forces $P_b^{(\beta)}$ s'exerçant sur les interfaces. La deuxième équation de (β) permet d'exprimer les déplacement internes $U_i^{(\beta)}$ en fonction de U_b,

$$U_i^{(\beta)} = -K_{ii}^{-1}K_{ib}U_b,$$

et en reportant dans la première, Przemieniecki obtient le *système d'interface*

$$(K_{bb} - K_{bi}K_{ii}^{-1}K_{ib})U_b = P_b - K_{bi}K_{ii}^{-1}P_i, \tag{2.6}$$

qui donne la valeur U_b sur la frontière

$$U_b = (K_{bb} - K_{bi}K_{ii}^{-1}K_{ib})^{-1}(P_b - K_{bi}K_{ii}^{-1}P_i). \tag{2.7}$$

Les déplacements internes dans chaque sous-domaine sont alors obtenus en sommant $U_i^{(\beta)}$ et $U_i^{(\alpha)}$.

En fait la matrice K_{ii} est une matrice diagonale par blocs, chaque bloc représentant une des sous-structures. Donc, au lieu d'inverser la grande matrice K, Przemieniecki imagine d'inverser les plus petites matrices de rigidité de chaque sous-structure, puis la plus petite matrice $S = K_{bb} - K_{bi}K_{ii}^{-1}K_{ib}$ qui représente le couplage par l'interface, et dont la taille est une dimension inférieure à celle de K. La matrice S est la matrice du complément de Schur des sous-domaines . Cette dénomination a été introduite par Emilie Haynsworth dans [23] d'après le lemme du déterminant d'Issai Schur, voir [46] pour un historique du complément de Schur.

La méthode que nous venons de décrire, avec la résolution algébrique des problèmes dans les sous-domaines, est répertoriée dans la classe des "méthodes de décomposition de domaine de type Schur". Ce nom est plus précis que le nom historique de sous-structuration, car toutes les méthodes de décomposition de domaine, y compris les méthodes de Schwarz, peuvent être écrites sous forme sous-structurée en éliminant les variables intérieures. C'est ce que nous verrons aux chapitres 3 et 4.

Les méthodes de Schur peuvent être employées de façon algébrique, en oubliant la physique contenue dans la présentation de Przemieniecki. La solution du système d'interface donnée explicitement dans (2.7) est souvent obtenue en appliquant une méthode iterative au système d'interface (2.6). Il n'est alors même pas nécessaire de former ce système explicitement : le produit matrice vecteur à effectuer à chaque itération revient à résoudre des problèmes aux limites dans les sous-domaines, représentés par les blocs de la matrice K_{ii} [5, 42, 38], ce que l'on peut faire à l'aide d'une méthode directe, ou de nouveau itérative. Ces méthodes seront décrites en détail dans la section 4.

2.3 Méthodes de relaxation d'onde

Ces algorithmes s'appliquent au calcul en parallèle de la solution $\boldsymbol{y}\colon [0,T] \to \mathbb{R}^d$ du système d'équations différentielles ordinaires

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(t,\boldsymbol{y}), \text{ avec la condition initiale } \boldsymbol{y}(0) = \boldsymbol{y}_0. \tag{2.8}$$

Ils s'appuient sur la méthode des approximations successives ou de point fixe de Picard [35] pour démontrer l'existence d'une solution à (2.8). Cette méthode définit une suite vectorielle $\boldsymbol{y}^n$ par

$$\frac{d\boldsymbol{y}^{n+1}}{dt} = \boldsymbol{f}(t,\boldsymbol{y}^n), \text{ avec la condition initiale } \boldsymbol{y}^{n+1}(0) = \boldsymbol{y}_0.$$

La première itération $\boldsymbol{y}^0$ est la fonction constante égale à $\boldsymbol{y}_0$. Cette dernière équation s'intègre sous la forme

$$\boldsymbol{y}^{n+1}(t) = \boldsymbol{y}_0 + \int_0^t \boldsymbol{f}(s,\boldsymbol{y}^n(s))\,ds. \tag{2.9}$$

On définit une condition de Lipschitz uniforme pour $\boldsymbol{f}$ (voir le dossier AF 652 des Techniques de l'Ingénieur)

$\forall t \in [0,T],\ \forall (\boldsymbol{y},\boldsymbol{z}) \in (\mathbb{R}^d)^2, \quad \|\boldsymbol{f}(t,\boldsymbol{y}) - \boldsymbol{f}(t,\boldsymbol{z})\| \leq L\|\boldsymbol{y}-\boldsymbol{z}\|\,, \quad$ L est la constante de Lipschitz de $\boldsymbol{f}$.

Théorème 2.2 *Si f est continue par rapport à ses deux variables, et uniformément lipschitzienne par rapport à y de constante L, la suite est convergente, et l'erreur est donnée par avec l'estimation superlinéaire de [26],*

$$\|\boldsymbol{y} - \boldsymbol{y}^n\| \leq \frac{(LT)^n}{n!}\|\boldsymbol{y} - \boldsymbol{y}^0\|, \tag{2.10}$$

où $||\cdot||$ est la norme du maximum sur $[0,T]$.

Démonstration Montrons par récurrence que pour tout n

$$\|\boldsymbol{y}^{n+1}(t) - \boldsymbol{y}^n(t)\| \leq \frac{(Lt)^n}{n!} \max_{[0,T]} \|\boldsymbol{y}^1 - \boldsymbol{y}^0\|.$$

Pour cela calculons par la formule intégrale

$$\begin{aligned}
\boldsymbol{y}^{n+2}(t) - \boldsymbol{y}^{n+1}(t) &= \int_0^t (\boldsymbol{f}(s, \boldsymbol{y}^{n+1}(s)) - \boldsymbol{f}(s, \boldsymbol{y}^n(s)))\, ds, \\
\|\boldsymbol{y}^{n+2}(t) - \boldsymbol{y}^{n+1}(t)\| &\leq \int_0^t \|\boldsymbol{f}(s, \boldsymbol{y}^{n+1}(s)) - \boldsymbol{f}(s, \boldsymbol{y}^n(s))\|\, ds \\
&\leq \int_0^t L\|\boldsymbol{y}^{n+1}(s) - \boldsymbol{y}^n(s)\|\, ds \text{ puisque } \boldsymbol{f} \text{ est lipschitzienne} \\
&\leq \max_{[0,T]} \|\boldsymbol{y}^1 - \boldsymbol{y}^0\| \int_0^t L\frac{(Ls)^n}{n!}\, ds \text{ par l'hypothèse de récurrence} \\
&\leq \max_{[0,T]} \|\boldsymbol{y}^1 - \boldsymbol{y}^0\| \frac{(Lt)^{n+1}}{(n+1)!} \text{ par intégration.} \qquad (2.11)
\end{aligned}$$

Notons que $\boldsymbol{y}^1 - \boldsymbol{y}^0$ est bornée, car

$$\|\boldsymbol{y}^1(t) - \boldsymbol{y}^0\| \leq \int_0^t \|\boldsymbol{f}(s, \boldsymbol{y}_0)\|\, ds \leq T \max_{s\in[0,T]} \|\boldsymbol{f}(s, \boldsymbol{y}_0)\|.$$

De l'estimation (2.11), nous déduisons que la suite y^n est de Cauchy. Pour tout n et tout p,

$$\begin{aligned}
\|\boldsymbol{y}^{n+p}(t) - \boldsymbol{y}^n(t)\| &\leq \sum_{k=1}^{p} \|\boldsymbol{y}^{n+k}(t) - \boldsymbol{y}^{n+k-1}(t)\| \\
&\leq \max_{[0,T]} \|\boldsymbol{y}^1 - \boldsymbol{y}^0\| \sum_{k=1}^{p} \frac{(Lt)^{n+k-1}}{(n+k-1)!} \\
&= \max_{[0,T]} \|\boldsymbol{y}^1 - \boldsymbol{y}^0\| \sum_{j=n}^{n+p-1} \frac{(Lt)^j}{j!}.
\end{aligned}$$

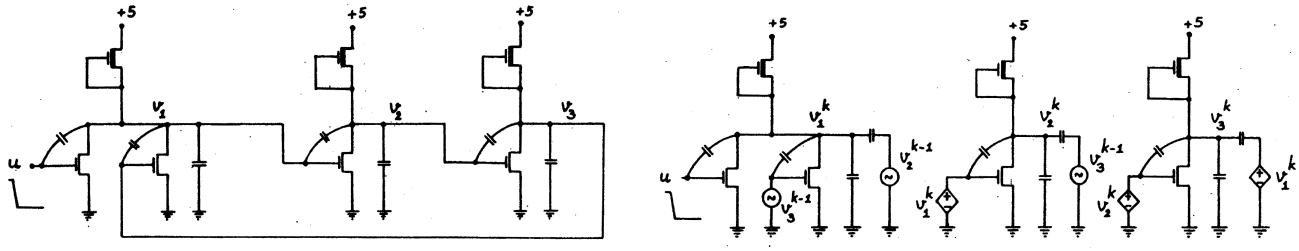
Cette dernière somme est une somme de Cauchy pour la série uniformément convergente $\sum \frac{(Lt)^j}{j!} = exp(Lt)$. Elle tend donc vers 0 et la suite est uniformément de Cauchy, elle converge vers une fonction $\tilde{y}$. Par continuité $\boldsymbol{f}(t, \boldsymbol{y}^n(t))$ tend uniformément vers $\boldsymbol{f}(t, \boldsymbol{y}(t))$. Passant à la limite dans (2.9), on peut en déduire que $\tilde{\boldsymbol{y}}$ et $\boldsymbol{y}$ coincident sur $[0, T]$. L'estimation (2.10) est obtenue par récurrence comme précédemment à partir de l'identité $\boldsymbol{y}^{n+1}(t) - \boldsymbol{y}(t) = \int_0^t (\boldsymbol{f}(s, \boldsymbol{y}^n(s)) - \boldsymbol{f}(s, \boldsymbol{y}(s)))\, ds$. ■

Remarque 2.1 *On parle de* convergence superlinéaire *: dans les premières itérations, si $LT < 1$, la convergence est dominée par $(LT)^n$, elle est donc linéaire. Par contre après quelques itérations, le terme $1/n!$ prend le pas et la convergence devient très rapide, même si $LT > 1$.*

Cet algorithme a été réinventé sous une forme plus générale un siècle plus tard par Lelarasmee, Ruehli and Sangiovanni-Vincentelli [25] pour simuler des circuits à très grande échelle (VLSI). Le modèle original traité en détail dans leur publication est l'oscillateur MOS en anneau décrit figure 2.4a. Son comportement est régi par un système d'équations différentielles portant sur les tensions v_1, v_2, v_3, qui s'écrit de manière abstraite

$$\frac{dv_1}{dt} = f_1(v_1, v_2, v_3), \quad \frac{dv_2}{dt} = f_2(v_1, v_2, v_3), \quad \frac{dv_3}{dt} = f_3(v_1, v_2, v_3).$$

Ils décident de couper leurs circuits en parties cohérentes pouvant être traités sur un seul processeur et de les coller. Le processus itératif est visible sur la figure 2.4b. Les trois équations différentielles des



(a) Circuit complet (b) Circuit partitionné

Figure 2.4 – Exemple original de 1982 pour les méthodes de relaxation d'ondes

sous-circuits sont résolues à l'étape k, avec comme données le potentiel d'entrée u, et les potentiels calculés à l'étape précédente :

$$\frac{dv_1^{k+1}}{dt} = f_1(v_1^{k+1}, v_2^k, v_3^k), \quad \frac{dv_2^{k+1}}{dt} = f_2(v_1^k, v_2^{k+1}, v_3^k), \quad \frac{dv_3^{k+1}}{dt} = f_3(v_1^k, v_2^k, v_3^{k+1}),$$

et ce jusqu'à convergence. Cet algorithme fait écho à l'algorithme de Jacobi pour la résolution de systèmes linéaires, qui fait partie de la famille des algorithmes de relaxation. Les représentations des signaux sur les oscilloscopes étant appelés waveform en anglais, ces algorithmes furent appelés waveform relaxation algorithms (ou relaxation d'onde en français). L'identification avec des méthodes de décomposition de domaines pour des équations aux dérivées partielles fut faite par Bjørhus [2] pour des systèmes hyperboliques du premier ordre, et par Gander et Stuart pour des équations paraboliques comme l'équation de la chaleur [20]. C'est ainsi qu'ont été créées les "Schwarz waveform relaxation methods", ou en français les méthodes de Schwarz relaxation d'onde.

2.4 Plan de l'exposé

Reprenons l'équation elliptique obtenue par discrétisation en temps de l'équation de la chaleur

$$(\eta - \Delta)u = f. \tag{2.12}$$

Dans tout l'exposé, cette équation est considérée dans un domaine Ω régulier : le segment $[0,1]$ dans le cas monodimensionnel, le carré $[0,1]\times[0,1]$ dans le cas bidimensionnel. La fonction f est une fonction continue sur Ω, appelée second membre de l'équation. Des résultats d'analyse généraux montrent que l'équation (2.12) a une solution unique de classe $\mathcal{C}^2$, qui s'annule sur la frontière $\partial\Omega$ de Ω. Plus précisément, nous détaillerons en dimension 1, sur l'équation

$$-\frac{d^2u}{dx^2} + \eta\, u = f \tag{2.13}$$

tous les algorithmes dont la présentation a été faite dans ce chapitre historique : Schwarz et ses variantes au chapitre 3, Schur et ses variantes au chapitre 4. Nous introduirons aussi en dimension 2 la notion fondamentale de conditionnement. Pour chacune de ces méthodes, nous donnerons un script Matlab, et représenterons la solution obtenue. Nous comparerons également les qualités de convergence de ces méthodes. Nous étudierons ensuite les propriétés de ces algorithmes lorsque le domaine de calcul est partagé en un grand nombre de sous-domaines. C'est la notion de scalabilité qui sera étudiée en détail au chapitre 5.

Ensuite, dans la section 6, nous nous intéresserons aux problèmes d'évolution. Nous commencerons par présenter une version évolutive de la méthode de Schwarz, la méthode de Schwarz relaxation d'onde, permettant de coupler des grilles en temps différentes dans les sous-domaines. Ensuite nous aborderons la question du parallélisme en temps, en présentant l'algorithme pararéel. Nous finirons par un exemple complet de parallélisme en espace et en temps.

3 Méthodes de Schwarz

Nous exposerons d'abord dans le cas monodimensionnel de l'équation (2.13) les méthodes de Schwarz "historiques", présentées par H.A. Schwarz (méthode alternée) puis P.L. Lions (méthode parallèle). Ensuite nous les discrétiserons, et interpréterons les algorithmes discrets obtenus en terme d'algorithme de relaxation. Puis nous présenterons les algorithmes discrets, en particulier la méthode de Schwarz additive de M. Dryja et O. Widlund [10, 11], qui constitue une invention majeure. Dans le paragraphe suivant, nous interpréterons ces méthodes de Schwarz comme des préconditionneurs pour la résolution d'un système linéaire. Ce système porte soit sur les inconnues internes, soit sur les inconnues d'interface. Nous étudierons alors en dimension 2 le conditionnement de ces systèmes linéaires préconditionnés.

3.1 Méthodes de Schwarz alternée et parallèle

Nous cherchons ici à calculer itérativement la solution du problème (2.13) qui prend des valeurs données g_g et g_d en 0 et 1. Le domaine de calcul $\Omega = [0,1]$ est divisé en deux sous-domaines $\Omega_1 = [0,\beta]$ et $\Omega_2 = [\alpha,1]$. La frontière de Ω_1 située dans Ω_2 est $\Gamma_1 = \{\beta\}$, et symétriquement $\Gamma_2 = \{\alpha\}$. Les domaines se recouvrent sur une distance $\delta = \beta - \alpha$.

Dans la *méthode de Schwarz alternée*, une suite (u_1^n, u_2^n) pour $n \geq 0$ est constituée en résolvant alternativement l'équation (2.13) dans Ω_1 et Ω_2, en se fixant les valeurs sur le bord du recouvrement au moyen de l'itération précédente, ce qui s'écrit :

$$\begin{array}{rclcrcl} -\dfrac{d^2u_1^{n+1}}{dx^2} + \eta\, u_1^{n+1} &=& f \text{ dans } \Omega_1, & \quad & -\dfrac{d^2u_2^{n+1}}{dx^2} + \eta\, u_2^{n+1} &=& f \text{ dans } \Omega_2, \\ u_1^{n+1}(0) &=& g_g, & & u_2^{n+1}(1) &=& g_d, \\ u_1^{n+1}(\beta) &=& u_2^n(\beta), & & u_2^{n+1}(\alpha) &=& u_1^{n+1}(\alpha). \end{array} \tag{3.1}$$

L'algorithme est initialisée par $g \in \mathbb{R}$, avec la convention $u_2^0(\beta) \equiv g$, c'est-à-dire que u_1^1 est calculé avec la condition $u_1^1(\beta) = g$.

Dans la *méthode de Schwarz parallèle* [28], les calculs dans Ω_1 et Ω_2 se font en parallèle :

$$\begin{array}{rclcrcl} -\dfrac{d^2\tilde{u}_1^{n+1}}{dx^2} + \eta\, \tilde{u}_1^{n+1} &=& f \text{ dans } \Omega_1, & \quad & -\dfrac{d^2\tilde{u}_2^{n+1}}{dx^2} + \eta\, \tilde{u}_2^{n+1} &=& f \text{ dans } \Omega_2, \\ \tilde{u}_1^{n+1}(0) &=& g_g, & & \tilde{u}_2^{n+1}(1) &=& g_d, \\ \tilde{u}_1^{n+1}(\beta) &=& \tilde{u}_2^n(\beta), & & \tilde{u}_2^{n+1}(\alpha) &=& \tilde{u}_1^n(\alpha). \end{array} \tag{3.2}$$

Il faut alors se donner deux valeurs d'initialisation g_1 et g_2.

La figure 3.1 représente la solution de (2.13) dans le cas modèle qui nous accompagnera tout au long de notre étude. L'exemple que nous avons choisi reproduit la distribution de température dans un barreau de longueur 1, soumis à une source de chaleur sur une partie de sa longueur, et dont la température est fixée aux deux extrémités. Sur la figure 3.2 sont représentées les itérées des algorithmes de Schwarz alterné et parallèle.

Théorème 3.1 *Pour tout $\eta \geq 0$, les algorithmes de Schwarz alterné et parallèle en dimension 1 appliqués à l'équation* (2.13) *sont convergents.*

Démonstration La démonstration originale de Schwarz peut être mise en œuvre simplement ici, mais nous préférons donner une nouvelle démonstration, qui fait intervenir l'importante notion de facteur de convergence. Commençons par l'algorithme alterné (3.1). Par linéarité, les erreurs $e_i^n = u_i^n - u$ sont solution des mêmes équations dans les sous-domaines avec un second membre f et des données aux bords g_g et g_d nuls. Nous pouvons résoudre explicitement les équations dans les sous-domaines, au

moyen de la fonction sinus hyperbolique $\mathrm{sh}x = (e^x - e^{-x})/2$ pour $\eta > 0$, à une constante multiplicative a_i^n près :

$$\begin{array}{ll} \text{pour } \eta > 0, & e_1^n = a_1^n \ \mathrm{sh}(\sqrt{\eta}\, x), \quad e_2^n = a_2^n \ \mathrm{sh}(\sqrt{\eta}\,(1-x)), \\ \text{pour } \eta = 0, & e_1^n = a_1^n \ x, \qquad\qquad\quad e_2^n = a_2^n \ (1-x). \end{array}$$

A la première itération, a_1^1 est déterminé par la condition $u_1^1(\beta) = g$, donc $e_1^1(\beta) = g - u(\beta)$:

$$\begin{cases} \text{pour } \eta > 0, & a_1^1 \ \mathrm{sh}(\sqrt{\eta}\ \beta) = g - u(\beta), \\ \text{pour } \eta = 0, & a_1^1 \ \beta = g - u(\beta). \end{cases}$$

Les conditions de transmission $e_1^{n+1}(\beta) = e_2^n(\beta)$ et $e_2^{n+1}(\alpha) = e_1^{n+1}(\alpha)$ donnent ensuite une formule de récurrence pour déterminer les coefficients a_i^n :

$$\begin{cases} \text{pour } \eta > 0, & a_1^{n+1} \ \mathrm{sh}(\sqrt{\eta}\ \beta) = a_2^n \ \mathrm{sh}(\sqrt{\eta}\,(1-\beta)), \quad a_2^{n+1} \ \mathrm{sh}(\sqrt{\eta}\,(1-\alpha)) = a_1^{n+1} \ \mathrm{sh}(\sqrt{\eta}\ \alpha), \\ \text{pour } \eta = 0, & a_1^{n+1}\, \beta = a_2^n \ (1-\beta), \quad a_2^{n+1} \ (1-\alpha) = a_1^{n+1}\alpha. \end{cases}$$

Posons

$$\rho_1 = \frac{\mathrm{sh}(\sqrt{\eta}\,(1-\beta))}{\mathrm{sh}(\sqrt{\eta}\ \beta)}, \quad \rho_2 = \frac{\mathrm{sh}(\sqrt{\eta}\,\alpha)}{\mathrm{sh}(\sqrt{\eta}\,(1-\alpha))}. \tag{3.3}$$

Ces formules sont aussi valables pour $\eta = 0$ par passage à la limite. Réécrivons la relation de récurrence sous la forme

$$a_1^{n+1} = \rho_1 a_2^n, \quad a_2^{n+1} = \rho_2\, a_1^{n+1}, \qquad \text{ou encore } a_i^{n+1} = \rho_1\rho_2\, a_i^n.$$

Les suites a_1^n et a_2^n sont des suites géométriques de raison $\rho = \rho_1\rho_2$, qui est aussi appelé facteur de convergence de la méthode. La fonction sinus hyperbolique étant croissante, et $\alpha < \beta$, nous avons $\mathrm{sh}(\sqrt{\eta}\ \alpha) < \mathrm{sh}(\sqrt{\eta}\ \beta)$ et $\mathrm{sh}(\sqrt{\eta}\,(1-\beta)) < \mathrm{sh}(\sqrt{\eta}\,(1-\alpha))$. Donc ρ est positif et strictement plus petit que 1. Les coefficients a_i^n sont alors donnés par

$$a_1^{n+1} = \rho^n a_1^1, \quad a_2^{n+1} = \rho_2\rho^n a_1^1. \tag{3.4}$$

Les fonctions u_i^n vérifient en chaque point de Ω_i :

$$u_i^{n+1}(x) - u(x) = \rho(u_i^n(x) - u(x)) = \rho^n(u_i^1(x) - u(x)).$$

A chaque étape, l'erreur en chaque point est multipliée par ρ. Dans le domaine Ω_i, la suite u_i^n converge donc uniformément vers u, et la convergence est linéaire.

Dans le cas parallèle, on a une relation analogue entre les coefficients $\tilde{a}_i^n$ de $\tilde{u}_i^n$:

$$\tilde{a}_1^{n+1} = \rho_1\tilde{a}_2^n, \quad \tilde{a}_2^{n+1} = \rho_2\tilde{a}_1^n, \qquad \text{ou encore } \tilde{a}_i^{n+1} = \rho\, \tilde{a}_i^{n-1},$$

si bien que $\tilde{a}_i^{2n+1} = \rho^n\, \tilde{a}_i^1$. Les itérées paires et impaires de $\tilde{u}_i^n$ convergent linéairement avec un facteur de convergence égal à ρ. ■

Remarque 3.1 *Si l'on pose $g = g_1$, on obtient $\tilde{u}_1^{2n-1} = u_1^n$, puis $\tilde{u}_2^{2n} = u_2^n$. Deux étapes de l'algorithme parallèle équivalent donc à une étape de l'algorithme alterné, ce que montre la figure 3.2.*

Remarque 3.2 *La convergence de la suite est d'autant plus rapide que ρ est petit. Ceci est réalisé lorsque η est grand, auquel cas ρ est équivalent à $exp(-2\sqrt{\eta}\ \delta)$. C'est le facteur de convergence que l'on obtiendrait aussi si l'on considérait les domaines Ω_i comme semi-infinis. Ce résultat montre aussi que les algorithmes de Schwarz sont d'autant plus rapides que le recouvrement δ est grand.*

3.2 Méthodes de Schwarz alternée et parallèle discrétisées

L'intervalle $[0,1]$ est partagé en $J+1$ sous-intervalles de longueur h. Les nœuds de la discrétisation sont les $x_j = jh$ pour $0 \leq j \leq J+1$. Le schéma aux différences finies associé à (2.13), où u_j est une approximation de $u(x_j)$ et f_j une approximation de $f(x_j)$, s'écrit

$$-\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + \eta\, u_j = f_j, \quad 1 \leq j \leq J.$$

A ces J équations s'ajoutent les valeurs aux deux extrémités $u_0 = g_g$ et $u_{J+1} = g_d$, pour obtenir un système linéaire portant sur les inconnues $\boldsymbol{u} = (u_1, \cdots, u_J)^T$ aux nœuds internes, que l'on peut écrire sous forme matricielle

$$A\boldsymbol{u} = \boldsymbol{f}, \quad A = \begin{pmatrix} \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & \\ -\frac{1}{h^2} & \frac{2}{h^2}+\eta & \ddots & \\ & \ddots & \ddots & -\frac{1}{h^2} \\ & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} f_1 + \frac{1}{h^2} g_g \\ f_2 \\ \vdots \\ f_{J-1} \\ f_J + \frac{1}{h^2} g_d \end{pmatrix}. \tag{3.5}$$

La matrice A s'écrit en format *sparse* en Matlab au moyen de la routine `spdiags` :

```
function A=A1d(eta,a,b,J)
% A1D one dimensional finite difference approximation
%   A=A1d(eta,a,b,J) computes a sparse finite difference approximation
%   of the one dimensional operator eta-Delta on the domain
%   Omega=(a,b) using J interior points

h=(b-a)/(J+1); e=ones(J,1);
A=spdiags([-e/h^2 (eta+2/h^2)*e -e/h^2],[-1 0 1],J,J);
```

Le script suivant détaille la résolution en Matlab du problème discret avec conditions aux limites non homogènes. Le système linéaire est résolu par la routine "\" de Matlab, qui est basée sur une méthode de Gauss, optimisée pour des matrices creuses. Notons que, la matrice A étant symétrique définie positive, nous pourrions également choisir comme solveur du système linéaire un algorithme itératif, le gradient conjugué, qui est extrêmement efficace, surtout pour les matrices creuses de grande taille (voir le dossier AF 488 des Techniques de l'Ingénieur).

```
function u=Solve1d(f,eta,a,b,gg,gd)
% SOLVE1D solves eta-Delta in 1d using finite differences
%   u=Solve1d(f,eta,a,b,gg,gd) solves the one dimensional equation
%   (eta-Delta)u=f on the domain Omega=(a,b) with Dirichlet boundary
%   conditions u=gg at x=a and u=gd at x=b using a finite
%   difference approximation with length(f) interior grid points

J=length(f);
A=A1d(eta,a,b,J);          % construct 1d finite difference operator
h=(b-a)/(J+1);
f(1)=f(1)+gg/h^2;          % add boundary conditions into rhs
f(end)=f(end)+gd/h^2;
u=A\f;
u=[gg;u;gd];               % add boundary values to solution
```

L'exemple que nous avons choisi reproduit la distribution de température dans un barreau de longueur 1, soumis à une source de chaleur sur une partie de sa longueur (le segment [0.4 0.7]), et dont la température est fixée aux deux extrémités. Le code Matlab `Bar.m` ci-dessous appelle le sous-programme de résolution `Solve1d` pour ces données et produit la courbe représentée figure 3.1.

```
eta=0;J=20;                % J number of interior mesh points
x=0:1/(J+1):1;             % finite difference mesh, including boundary
f=zeros(J,1);              % source term zero, except for a
f(x>0.4 & x<0.7)=5;        % heater in this position
gg=0.1; gd=0;              % put warm wall on the left, cold on the right
u=Solve1d(f,eta,0,1,gg,gd);
plot(x,u,'-'); xlabel('x'); ylabel('solution');
```

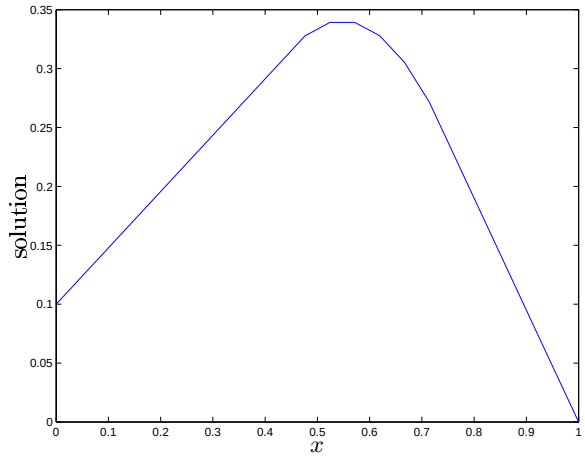


Figure 3.1 – Exemple de résolution de l'équation (2.13) par différences finies

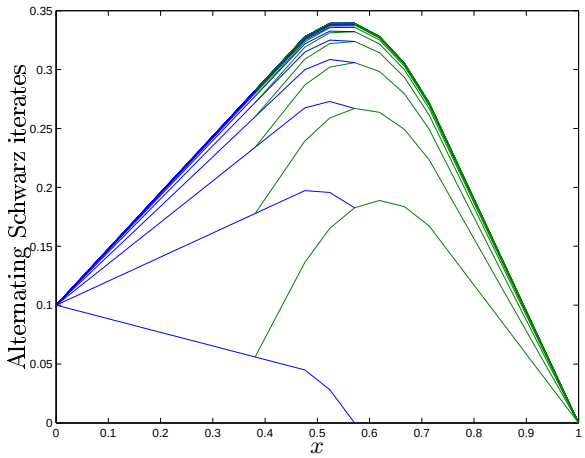
Pour discrétiser l'algorithme de Schwarz alterné, le point α est repéré par son indice a et β par $b = a + d$ où d représente le recouvrement, $\delta = d\,h$. Les points $x_1, \cdots, x_J$ sont intérieurs à Ω, les points $x_1, \cdots, x_{b-1}$ sont intérieurs à Ω_1, tandis que les points $x_{a+1}, \cdots, x_J$ sont intérieurs à Ω_2. La discrétisation de l'algorithme de Schwarz alterné (3.1) s'écrit

$$\begin{aligned}
&-\frac{(u_1^{n+1})_{j+1} - 2(u_1^{n+1})_j + (u_1^{n+1})_{j-1}}{h^2} + \eta\,(u_1^{n+1})_j = f_j, \quad 1 \le j \le b-1,\ (u_1^{n+1})_b = (u_2^n)_b, \\
&-\frac{(u_2^{n+1})_{j+1} - 2(u_2^{n+1})_j + (u_2^{n+1})_{j-1}}{h^2} + \eta\,(u_2^{n+1})_j = f_j, \quad a+1 \le j \le J, \quad (u_2^{n+1})_a = (u_1^{n+1})_a,
\end{aligned} \tag{3.6}$$

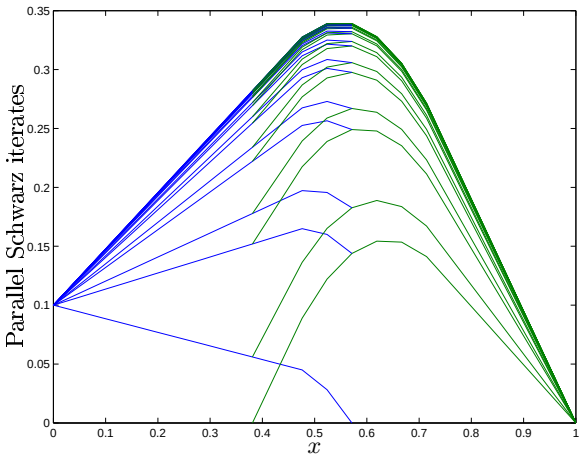
avec les conditions aux limites extérieures $(u_1^{n+1})_0 = g_g$ et $(u_2^n)_{J+1} = g_d$. Le script suivant réalise l'algorithme (3.6) (la première ligne de ce script, "`Bar;`", exécute les commandes de l'exemple précédent, qui doivent se trouver dans le fichier `Bar.m`) :

```
Bar;                          % to include problem parameters
a=8; d=4;                     % subdomain decomposition
f1=f(1:a+d-1); f2=f(a+1:J);   % subdomain source terms
u1=[gg; zeros(a+d,1)];        % zero initial guess, except boundary value
u2=[zeros(J-a+1,1); gd];
x1=x(1:a+d+1); x2=x(a+1:end); % finite difference meshes
for i=1:20                    % Alternating Schwarz iteration
  u1=Solve1d(f1,eta,x1(1),x1(end),gg,u2(d+1));
  u2=Solve1d(f2,eta,x2(1),x2(end),u1(end-d),gd);
  plot(x1,u1,'-',x2,u2,'-'); xlabel('x');
  ylabel('Alternating Schwarz iterates');
  hold on; pause
end
hold off
```

Exécuté en Matlab, il produit la suite de courbes représentée figure 3.2a.



(a) Algorithme alterné



(b) Algorithme parallèle

Figure 3.2 – Exemple de résolution de l'équation (2.13) par méthode de Schwarz discrétisée par différences finies

Pour obtenir l'algorithme parallèle de Schwarz, et les résultats de la figure 3.2b, il faut modifier la boucle précédente en

```
u1old=u1;
u1=Solve1d(f1,eta,x1(1),x1(end),gg,u2(d+1));
u2=Solve1d(f2,eta,x2(1),x2(end),u1old(end-d),gd);
plot(x1,u1,'-',x2,u2,'-'); xlabel('x');
ylabel('Parallel Schwarz iterates');
```

Interprétation algébrique Nous allons maintenant écrire l'algorithme (3.6) sous forme d'un algorithme algébrique portant sur les vecteurs $\boldsymbol{u}_1^n = ((u_1^n)_1, \cdots, (u_1^n)_{b-1})^T$ et $\boldsymbol{u}_2^n = ((u_2^n)_{a+1}, \cdots, (u_2^n)_J)^T$. La matrice A peut être décomposée par blocs sous la forme

$$A = \left(\begin{array}{cccc|ccc} \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & & & & \\ -\frac{1}{h^2} & \frac{2}{h^2}+\eta & \ddots & & & & \\ & \ddots & \ddots & -\frac{1}{h^2} & & & \\ & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & \\ \hline & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & -\frac{1}{h^2} & \\ & & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & \ddots \\ & & & & & \ddots & -\frac{1}{h^2} \\ & & & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta \end{array}\right). \tag{3.7}$$

Introduisons deux décompositions de ce type :

$$A = \begin{pmatrix} A_1 & B_1 \\ C_1 & D_1 \end{pmatrix} = \begin{pmatrix} D_2 & C_2 \\ B_2 & A_2 \end{pmatrix}, \tag{3.8}$$

où A_1 est de taille $(b-1)\times(b-1)$, et D_2 de taille $a\times a$, et donc A_2 est de taille $(J-a)\times(J-a)$. Les matrices A_1 et D_2 coïncident lorsque $b=a+1$, c'est-à-dire $d=1$. Le recouvrement géométrique est alors minimal, et le recouvrement algébrique est vide. Les matrices A_1 et A_2 sont les matrices de l'opérateur $\eta-\Delta$ discrétisé, avec données de Dirichlet homogène sur les bords, elles sont donc inversibles.

Les matrices B_i sont complétées par des zéros en

$$\tilde{B}_1 = [0_{b-1,d-1}\ B_1], \quad \tilde{B}_2 = [B_2\ 0_{J-a,d-1}].$$

Ainsi $\tilde{B}_1$ est une matrice $(b-1)\times(J-a)$ qui à un vecteur défini sur Ω_2 associe un vecteur défini sur Ω_1, prolongé par 0 hors de Ω_2. De même $\tilde{B}_2$ est une matrice $(J-a)\times(b-1)$ qui à un vecteur défini sur Ω_1 associe un vecteur défini sur Ω_2, prolongé par 0 hors de Ω_1. Avec ces notations, l'algorithme alterné (3.6) prend la forme

$$A_1\boldsymbol{u}_1^{n+1} = \boldsymbol{f}_1 - \tilde{B}_1\boldsymbol{u}_2^n\ , \quad A_2\boldsymbol{u}_2^{n+1} = \boldsymbol{f}_2 - \tilde{B}_2\boldsymbol{u}_1^{n+1}\ , \tag{3.9}$$

qui n'est autre qu'une méthode de Gauss-Seidel par blocs

$$\begin{pmatrix} A_1 & 0 \\ \tilde{B}_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n+1} = \begin{pmatrix} 0 & -\tilde{B}_1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} + \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix} \tag{3.10}$$

pour le système augmenté

$$\tilde{A}\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{f}}\ : \quad \begin{pmatrix} A_1 & \tilde{B}_1 \\ \tilde{B}_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{3.11}$$

Notons que lorsque le recouvrement est minimal, la matrice augmentée coïncide avec la matrice A de départ.

L'algorithme de Schwarz parallèle discrétisé peut aussi être exprimé sous forme algébrique :

$$A_1\boldsymbol{u}_1^{n+1} = \boldsymbol{f}_1 - \tilde{B}_1\boldsymbol{u}_2^n, \qquad A_2\boldsymbol{u}_2^{n+1} = \boldsymbol{f}_2 - \tilde{B}_2\boldsymbol{u}_1^n, \tag{3.12}$$

qui est cette fois une méthode de Jacobi par blocs pour le système augmenté (3.11), i.e.

$$\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n+1} = \begin{pmatrix} 0 & -\tilde{B}_1 \\ -\tilde{B}_2 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} + \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{3.13}$$

Une itération de l'algorithme se comprend de la manière suivante : $\tilde{B}_2\boldsymbol{u}_1^n$ est le vecteur de taille $J-a$ dont toutes les composantes sont nulles sauf la première qui vaut $-\dfrac{1}{h^2}(u_1^n)_a$. De même $\tilde{B}_1\boldsymbol{u}_2$ est le vecteur de taille $b-1$ dont toutes les composantes sont nulles sauf la dernière qui vaut $-\dfrac{1}{h^2}(u_2^n)_b$. Maintenant $A_1^{-1}(\boldsymbol{f}_1-\tilde{B}_1\boldsymbol{u}_2)$ représente la résolution du problème de Dirichlet dans Ω_1, avec second membre $\boldsymbol{f}_1$, et donnée à la limite en b égale à $(u_2^n)_b$. De même $A_2^{-1}(\boldsymbol{f}_2-\tilde{B}_2\boldsymbol{u}_1)$ représente la résolution du problème de Dirichlet dans Ω_2, avec second membre $\boldsymbol{f}_2$, et donnée à la limite en a égale à $(u_1^n)_a$. Si bien qu'une itération de l'algorithme revient à résoudre un problème aux limites dans chaque sous-domaine.

Notons que, bien que la matrice A soit symétrique, la matrice augmentée ne l'est pas, car les matrices $\tilde{B}_2^T$ et $\tilde{B}_1$ sont différentes sauf si le recouvrement est minimal, i.e. $d=1$ (en effet le seul terme non nul dans $\tilde{B}_1$ est $(\tilde{B}_1)_{b-1,d}$, alors que $(\tilde{B}_2^T)_{b-1,d} = (\tilde{B}_2)_{d,b-1} = 0$ si $d\neq 1$). Ceci interdit l'utilisation de la méthode du gradient conjugué pour résoudre le système préconditionné par la méthode de Schwarz parallèle,

$$\begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} A_1 & \tilde{B}_1 \\ \tilde{B}_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}, \tag{3.14}$$

car la méthode du gradient conjugué ne peut s'appliquer qu'à une matrice symétrique définie positive préconditionnée par matrice symétrique définie positive. Toutefois des méthodes de Krylov ont été développées pour les systèmes non symétriques, et ces méthodes, malgré leur coût supérieur à celui de la méthode du gradient conjugué et l'absence d'une théorie de convergence complète, sont plus performantes que les méthodes simples de Jacobi ou Gauss-Seidel par blocs (par exemple GMRES, QMR, BiCGStab, voir l'excellent livre d'analyse numérique matricielle de Yousef Saad [40] ou le dossier AF 488 des Techniques de l'Ingénieur). D'autre part d'autres algorithmes de décomposition de domaine ont été développés pour rétablir la symétrie, comme nous allons le voir dans le paragraphe suivant.

3.3 Méthodes de Schwarz discrètes : AS, MS et RAS

Pour comprendre l'invention majeure que constitue la méthode de Schwarz additive (AS), revenons à la méthode de Schwarz parallèle discrétisée écrite sous forme de Jacobi par blocs en (3.13). Dans le cas où $d = 1$, $\tilde{B}_i$ n'est autre que B_i, et l'itération est identique à

$$\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n+1} = \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} + \begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \left(\boldsymbol{f} - A \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}^{n} \right). \tag{3.15}$$

Cette écriture nous permet dans ce cas d'interpréter l'algorithme de Schwarz parallèle discrétisé (3.13) comme une méthode itérative pour le *système préconditionné*

$$\begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} A_1 & B_1 \\ B_2 & A_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{3.16}$$

Si la matrice A est symétrique et définie positive, le *préconditioneur* $\left(\begin{smallmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{smallmatrix}\right)$ l'est aussi, et (3.16) peut être résolu par la méthode du gradient conjugué.

Introduisons maintenant les matrices de restriction

$$R_1 = [I_{b-1} \quad 0_{b-1,J-b+1}], \quad R_2 = [0_{J-a,a} \quad I_{J-a}]. \tag{3.17}$$

Le préconditionneur s'écrit au moyen de ces matrices de restriction :

$$\begin{pmatrix} A_1^{-1} & 0 \\ 0 & A_2^{-1} \end{pmatrix} = \sum_{i=1}^{2} R_i^T A_i^{-1} R_i \ .$$

Puisque le vecteur $(\boldsymbol{u}_1, \boldsymbol{u}_2)$ avec recouvrement minimal $d = 1$ n'est autre que le vecteur global $\boldsymbol{u}$, nous en déduisons une nouvelle forme pour l'algorithme de Schwarz parallèle discrétisé (3.13) dans le cas où $d = 1$, *i.e.* (3.15) :

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \sum_{i=1}^{2} R_i^T A_i^{-1} R_i \ (\boldsymbol{f} - A\boldsymbol{u}^n). \tag{3.18}$$

Cet algorithme a encore un sens pour un recouvrement quelconque, mais il n'est pas très utile, comme le montre le contre-exemple suivant :

Théorème 3.2 *Si le recouvrement n'est pas minimal, $d > 1$, l'algorithme* (3.18) *appliqué à la matrice de différences finies* (3.5) *n'est pas convergent : il existe une donnée initiale $\boldsymbol{u}^0$ telle que l'algorithme oscille entre $\boldsymbol{u}^0$ et $-\boldsymbol{u}^0$.*

Démonstration Décomposons la matrice A de deux façons comme en (3.8). A chaque itération, le vecteur $\boldsymbol{u}^n$ est décomposé en $(\boldsymbol{u}_{11}^n, \boldsymbol{u}_{12}^n)$ en accord avec la première décomposition de A, et en $(\boldsymbol{u}_{21}^n, \boldsymbol{u}_{22}^n)$ en accord avec la deuxième décomposition. Le second membre $\boldsymbol{f}$ est décomposé de la même façon. Nous avons alors :

$$R_1 A\boldsymbol{u}^n = [A_1 \ B_1]\boldsymbol{u}^n = A_1\boldsymbol{u}_{11}^n + B_1\boldsymbol{u}_{12}^n, \quad R_2 A\boldsymbol{u}^n = [B_2 \ A_2]\boldsymbol{u}^n = B_2\boldsymbol{u}_{21}^n + A_2\boldsymbol{u}_{22}^n \ .$$

Calculons maintenant

$$R_1^T A_1^{-1} R_1 A \boldsymbol{u}^n = R_1^T \boldsymbol{u}_{11}^n + R_1^T A_1^{-1} B_1 \boldsymbol{u}_{12}^n,$$
$$R_2^T A_2^{-1} R_2 A \boldsymbol{u}^n = R_2^T \boldsymbol{u}_{22}^n + R_2^T A_2^{-1} B_2 \boldsymbol{u}_{21}^n \ .$$

L'équation (3.18) s'écrit alors

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \begin{pmatrix} \boldsymbol{u}_{11}^n \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ \boldsymbol{u}_{22}^n \end{pmatrix} - \begin{pmatrix} A_1^{-1} B_1 \boldsymbol{u}_{12}^n \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ A_2^{-1} B_2 \boldsymbol{u}_{21}^n \end{pmatrix} + \begin{pmatrix} A_1^{-1} \boldsymbol{f}_1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ A_2^{-1} \boldsymbol{f}_2 \end{pmatrix}.$$

Regardons maintenant la convergence de l'algorithme. Pour cela choisissons un second membre $\boldsymbol{f}$ nul, et pour un indice j compris strictement entre a et $a+d$, une donnée initiale $\boldsymbol{u}^0 = \boldsymbol{e}_j$, le j-ème vecteur de la base canonique de $\mathbb{R}^J$. Les seuls termes non nuls à droite de l'équation précédente sont les trois premiers, chacun vaut $\boldsymbol{u}^0$, si bien que $\boldsymbol{u}^1 = -\boldsymbol{u}^0$. En itérant l'argument, on voit que l'algorithme oscille entre $\boldsymbol{u}^0$ et $-\boldsymbol{u}^0$. ■

Le script Matlab suivant `BarAS.m` réalise les itérations de la méthode (3.18) pour le même exemple que précédemment. Les itérations sont représentées figure 3.3. Le défaut de convergence dans le recouvrement apparaît nettement.

```
Bar;                                    % to include problem parameters
a=8; d=4;                               % subdomain decomposition
h=1/(J+1);
f(1)=f(1)+gg/h^2; f(end)=f(end)+gd/h^2; % add boundary conditions into rhs
A=A1d(eta,0,1,J);                       % construct finite difference operator
R1=[speye(a+d-1) sparse(a+d-1,J-a-d+1)];
R2=[sparse(J-a,a) speye(J-a)];
A1=R1*A*R1'; A2=R2*A*R2';
u=zeros(J,1);
for i=1:20
  r=f-A*u;
  u=u+(R1'*(A1\(R1*r))+R2'*(A2\(R2*r)));
  plot(x,[gg;u;gd],'-'); xlabel('x'); ylabel('Additive Schwarz stationnary iterates');
  hold on; pause
  ri(i)=norm(r);                        % keep residual for plotting later
end
hold off
```

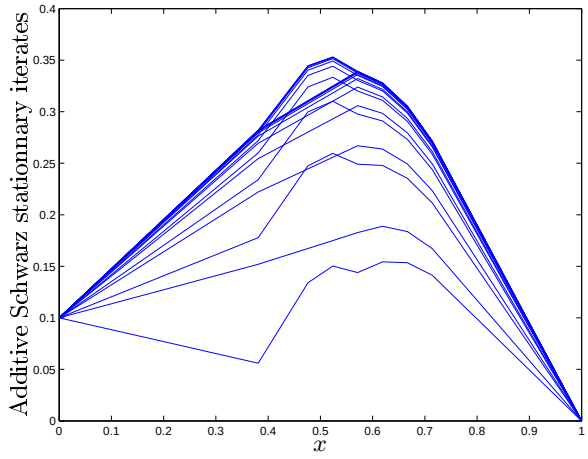


Figure 3.3 – Essai de résolution du système (3.5) par l'algorithme (3.18)

La véritable **méthode de Schwarz additive ou AS** est basée sur l'itération (3.18), mais considérée comme préconditionneur. Elle consiste à résoudre le *système préconditionné* obtenu à la limite

$$M_{AS}^{-1} A\boldsymbol{u} := \sum_{i=1}^{2} R_i^T A_i^{-1} R_i A\boldsymbol{u} = \sum_{i=1}^{2} R_i^T A_i^{-1} R_i \boldsymbol{f}. \tag{3.19}$$

Lorsque A est symétrique, le préconditionneur de Schwarz additif $M_{AS}^{-1} = \sum_i R_i^T A_i^{-1} R_i$ est symétrique, ce qui permet la résolution par l'algorithme du gradient conjugué, voir paragraphe 3.4. C'est la seule méthode de Schwarz connue à ce jour qui ait cette propriété, mais elle est obtenue au prix de la perte de convergence dans la version itérative (3.18) pour les modes présents dans le recouvrement. Cet algorithme est devenu l'algorithme de Schwarz le plus communément utilisé pour les problèmes symétriques, couplé avec une grille grossière, nous en parlerons au chapitre 5. Pour une approche différente qui utilise du recouvrement et reste symétrique, voir [9].

La méthode de *Schwarz multiplicative ou MS* (voir [5]) est la version séquentielle de la méthode de Schwarz additive. Pour notre exemple, elle s'écrit

$$\begin{aligned} \boldsymbol{u}^{n+\frac{1}{2}} &= \boldsymbol{u}^n + R_1^T A_1^{-1} R_1 (\boldsymbol{f} - A\boldsymbol{u}^n), \\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n+\frac{1}{2}} + R_2^T A_2^{-1} R_2 (\boldsymbol{f} - A\boldsymbol{u}^{n+\frac{1}{2}}) \ . \end{aligned} \tag{3.20}$$

Pour l'implémenter en Matlab, il suffit de remplacer dans la boucle du programme précédent le calcul de `r` et `u` par

```
r=f-A*u; u=u+R1'*(A1\(R1*r));
r=f-A*u; u=u+R2'*(A2\(R2*r));
```

ce qui produit les itérations tracées sur la figure 3.4.

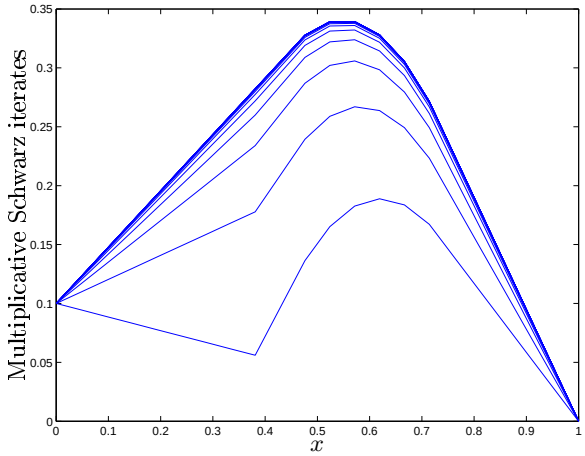


Figure 3.4 – Exemple de la méthode de Schwarz multiplicative

Il est intéressant de constater que les modes oscillants dans le recouvrement ont disparu : l'algorithme multiplicatif itératif est convergent. Ceci peut être démontré d'une manière rigoureuse pour des décompositions très générales, voir [16] où l'on montre également que cette méthode est équivalente à la méthode de Schwarz alternée discrétisée. Mais par contre le système préconditionné associé est non-symétrique, puisqu'il correspond à une méthode de Gauss-Seidel par blocs, comme nous l'avons vu, et ne peut donc pas être résolu par un gradient conjugué.

En 1998, Cai et Sarkis [4] introduisirent une nouvelle méthode discrète, l'algorithme de *Schwarz additif restreint ou RAS*, défini par

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \sum_{i=1}^{2} \widetilde{R}_i^T A_i^{-1} R_i (\boldsymbol{f} - A\boldsymbol{u}^n). \tag{3.21}$$

Les deux nouvelles matrices de restriction $\widetilde{R}_i$ sont obtenues en changeant des 1 en 0 dans les R_i, de façon à correspondre à une décomposition sans recouvrement, i.e. $\widetilde{R}_1^T R_1 + \widetilde{R}_2^T R_2 = I$. Le principe est décrit figure 3.5,

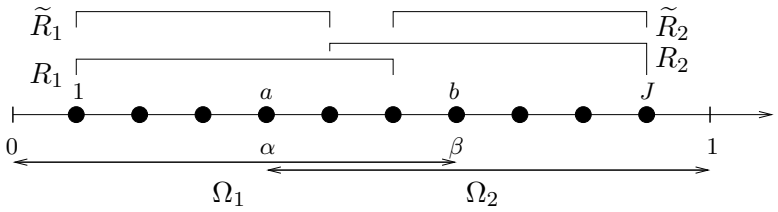


Figure 3.5 – Définition des $\widetilde{R}_i$ en dimension 1

et les matrices $\widetilde{R}_i$ sont de même taille que les R_i, avec

$$\widetilde{R}_1 = \begin{bmatrix} I_{a+[\frac{d}{2}]} & 0 \\ 0 & 0 \end{bmatrix}, \quad \widetilde{R}_2 = \begin{bmatrix} 0 & 0 \\ 0 & I_{J-(a+[\frac{d}{2}])} \end{bmatrix},$$

où $[x]$ désigne la partie entière de x. Cette modification a comme conséquence de supprimer l'erreur commise dans le recouvrement par l'itération (3.18) de Schwarz additif, et la méthode est convergente et équivalente à la méthode de Schwarz parallèle discrétisée, voir [13, 16].

Pour obtenir `BarRAS.m`, implémentation en Matlab de la méthode RAS, il suffit d'ajouter dans `BarAS.m` le calcul des opérateurs $\widetilde{R}_i$ par les commandes

```
m=a+ceil(d/2);                          % construct the Rtilde operators
R1t=R1;R1t(m:a+d-1,m:a+d-1)=0;
R2t=R2;R2t(1:m-a-1,a+1:m-1)=0;
```

et de les remplacer dans la formule pour `u`. On voit clairement sur les courbes de la figure 3.6 que l'algorithme converge maintenant dans le recouvrement, mais les itérées sont discontinues.

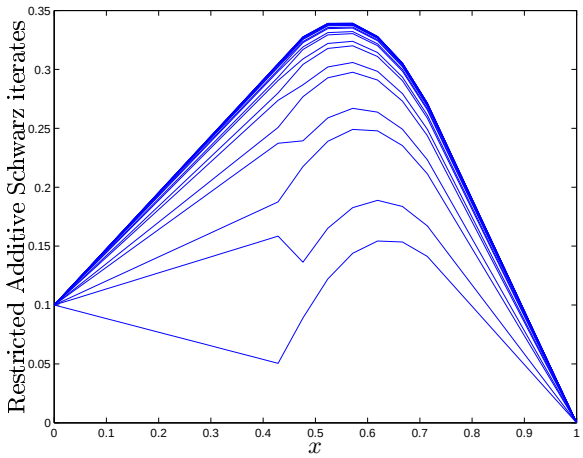


Figure 3.6 – Exemple de résolution de l'équation (2.13) par la méthode de Schwarz additive restreinte

Malheureusement cette modification qui améliore la performance de AS détruit la symétrie du préconditionneur, car $M_{RAS}^{-1} := \sum_{i=1}^{2} \widetilde{R}_i^T A_i^{-1} R_i$ n'est clairement pas symétrique. Cependant, parmi les méthodes de Schwarz, RAS est devenu le standard pour les problèmes non-symétriques.

Remarque 3.3 *On pourrait également construire une méthode de Schwarz multiplicative restreinte, mais cela changerait peu les performances, puisque Schwarz multiplicatif est déjà convergent dans le recouvrement.*

Tous les algorithmes de Schwarz discrets calculent une approximation globale de la solution, tandis que les algorithmes discrétisés (3.9) et (3.12) calculent des approximations de $\boldsymbol{u}_1$ et $\boldsymbol{u}_2$, dans les sous-domaines. Il est cependant facile de reconstituer une solution globale à partir de ces approximations : introduisons les matrices de partition de l'unité X_i pour $i = 1, 2$:

$$X_1 = \begin{pmatrix} I_a & 0 & 0 \\ 0 & M_{d-1} & 0 \\ 0 & 0 & 0_{J-a+d+1} \end{pmatrix} R_1^T, \quad X_2 = \begin{pmatrix} 0_a & 0 & 0 \\ 0 & I - M_{d-1} & 0 \\ 0 & 0 & I_{J-a+d+1} \end{pmatrix} R_2^T,$$

où la matrice M est par exemple diagonale avec $M_{ii} = (a+d-i)/d$. Une approximation discrète $\boldsymbol{u}$ de u aux points de grille est alors définie par

$$\boldsymbol{u} = X_1 \boldsymbol{u}_1 + X_2 \boldsymbol{u}_2.$$

Cette construction est également possible au niveau continu, au moyen d'une partition de l'unité, c'est-à-dire deux fonctions χ_i pour $i = 1, 2$ indéfiniment différentiable sur Ω, de somme égale à 1, à support dans $\overline{\Omega_i}$. L'approximation globale est alors donnée par

$$u = \chi_1 u_1 + \chi_2 u_2.$$

La matrice M ci-dessus est une discrétisation affine de χ_1.

3.4 Méthodes de Schwarz considérées comme préconditionneur

Une des motivations les plus fortes pour l'utilisation des méthodes de décomposition de domaines est la taille des systèmes en jeu. En effet lorsque la taille du système croit, les méthodes itératives traditionnelles deviennent de plus en plus lentes. La qualité de la méthode est mesurée par le conditionnement (voir [40] et le dossier AF485 des Techniques de l'Ingénieur).

Conditionnement et gradient conjugué Le conditionnement de la résolution d'un système linéaire $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ est mesuré par le *conditionnement* de la matrice $\mathcal{A}$. Ce dernier, relatif à la norme euclidienne notée $\|\cdot\|$, est défini par la formule

$$\kappa(\mathcal{A}) = \|\mathcal{A}\| \, \|\mathcal{A}^{-1}\|.$$

La matrice A des différences finies en dimension 1 (donnée par (3.5)) est symétrique définie positive. Sa norme est donc égale à son rayon spectral, c'est-à-dire sa plus grande valeur propre. Il en va de même de son inverse. Les valeurs propres de A sont

$$\lambda_j^d = \eta + \frac{4}{h^2} \sin^2 \frac{j\pi\, h}{2}, \quad 1 \leq j \leq J. \tag{3.22}$$

Le conditionnement de A est donc donné par

$$\kappa(A) = \frac{\max_{1 \leq j \leq J} \lambda_j^d}{\min_{1 \leq j \leq J} \lambda_j^d} = \frac{\lambda_J^d}{\lambda_1^d} = \frac{\eta + \dfrac{4}{h^2} \sin^2 \dfrac{J\pi\, h}{2}}{\eta + \dfrac{4}{h^2} \sin^2 \dfrac{\pi\, h}{2}}.$$

Lorsque h est petit, $\lambda_J^d \sim \frac{4}{h^2}$, tandis que $\lambda_1^d \sim \eta + \pi^2$. Notons que les valeurs propres du problème continu (2.13) avec des conditions aux limites de Dirichlet homogène sont égales à $\lambda_j^c = \eta + j^2\pi^2$ pour tout j entier naturel. On a donc pour de faibles valeurs de h, $\lambda_1^d \sim \lambda_1^c$ et $\lambda_J^d \sim \lambda^c_{\frac{2}{\pi h}}$. La gamme de

fréquences du problème continu correspondant à des modes propres du problème discret est $(1, \frac{2}{\pi h})$. Maintenant le conditionnement de la matrice A pour h petit est asymptotiquement égal à

$$\kappa(A) \sim \frac{4}{\eta + \pi^2} \frac{1}{h^2}. \tag{3.23}$$

Il est bien connu que l'algorithme de Richardson, défini par

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \lambda(\boldsymbol{b} - \mathcal{A}\boldsymbol{x}^n), \tag{3.24}$$

converge linéairement pour des matrices symétriques définies positives, avec un facteur de convergence égal à

$$\frac{\kappa(\mathcal{A}) - 1}{\kappa(\mathcal{A}) + 1},$$

si l'on utilise un paramètre de relaxation optimal

$$\lambda^* = \frac{2}{\lambda_{\min}(\mathcal{A}) + \lambda_{\min}(\mathcal{A})}.$$

Dans le cas des différences finies en dimension 1, avec l'estimation (3.23), le facteur de convergence est équivalent à $1 - \frac{\sqrt{\eta+\pi^2}}{2} h^2$. Pour l'algorithme du gradient conjugué, le facteur de convergence est égal à

$$\frac{\sqrt{\kappa(\mathcal{A})} - 1}{\sqrt{\kappa(\mathcal{A})} + 1},$$

et pour la matrice A des différences finies en dimension 1, cette quantité est équivalente à $1 - \sqrt{\eta + \pi^2}\, h$, ce qui améliore de façon frappante la convergence. En effet la convergence est d'autant plus lente que le facteur de convergence est proche de 1, et lorsque h est petit, $1 - \mathcal{O}(h^2)$ est beaucoup plus proche de 1 que $1 - \mathcal{O}(h)$. Ce résultat est aussi valable en dimension supérieure, comme nous le verrons plus loin en dimension 2 dans la formule (3.31). Pour l'améliorer encore, il faudrait rendre le conditionnement le moins dépendant de h possible. Cela peut être obtenu en préconditionnant le système, c'est-à-dire en remplaçant le système $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ par $\mathcal{M}^{-1}\mathcal{A}\boldsymbol{x} = \mathcal{M}^{-1}\boldsymbol{b}$ où la matrice $\mathcal{M}$ est une matrice facile à inverser, telle que le conditionnement de $\mathcal{M}^{-1}\mathcal{A}$ soit meilleur que celui de $\mathcal{A}$. Parmi beaucoup de choix possibles, les méthodes de Schwarz que nous avons présentées plus haut sont d'excellentes candidats parallèles.

Préconditionnement en volume Nous avons vu que le préconditionneur de Schwarz additif présenté en (3.19) est symétrique pour les problèmes symétriques. Le système préconditionné peut alors être résolu par le même algorithme que le problème de départ, le gradient conjugué, qui est l'algorithme le plus performant pour les problèmes symétriques. Sur la figure 3.7 nous voyons comment, à partir de l'algorithme itératif (3.18) qui ne converge pas dans le recouvrement, nous sommes arrivés à une convergence très rapide avec le gradient conjugué. En abscisse est représenté le numéro n de l'itération, en ordonnée la norme euclidienne du résidu $\boldsymbol{f} - A\boldsymbol{u}_n$. Le script Matlab correspondant est inséré ci-dessous.

```
BarAS;                                          % first part like in BarAS.m
Mfun=@(x) R1'*(A1\(R1*x))+R2'*(A2\(R2*x));
[u,fl,r,it,rcg]=pcg(A,f,1e-6,10,Mfun);
semilogy(0:length(ri)-1,ri,'-o',0:length(rcg)-1,rcg,'-+');
xlabel('iteration'); ylabel('residual');
legend('Iterative','CG');
```

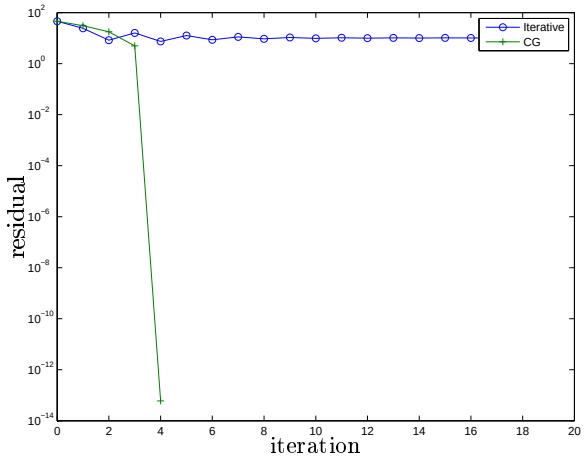


Figure 3.7 – Schwarz additif utilisé comme préconditionneur comparé à la méthode itérative

L'algorithme du gradient conjugué converge à la quatrième itération. En effet la matrice du système linéaire préconditionné issu de l'algorithme de Schwarz s'écrit sous la forme $(I - M)$ où M est de rang $r = 3$ pour notre exemple, car dans ce cas monodimensionnel le problème d'interface résolu est de taille 2, comme nous le verrons plus tard, et Schwarz additif a encore un mode dans le recouvrement. Or on sait que pour une telle matrice, les méthodes de Krylov convergent en $r + 1$ itérations.

L'algorithme de Schwarz additif restreint peut aussi être utilisé comme préconditionneur. Les matrices R_i à gauche sont alors remplacées par les matrices $\tilde{R}_i$. Le préconditionneur obtenu n'est plus symétrique, la méthode du gradient conjugué ne semble pas converger, comme le montre la figure 3.8. On doit alors faire appel à un algorithme de Krylov adapté aux problèmes non symétriques, par exemple GMRES voir [40]. Le script Matlab est inséré ci-dessous.

```
BarRAS;                                          % first part like BarRAS.m
Mfun=@(x) R1t'*(A1\(R1*x))+R2t'*(A2\(R2*x));
[u,fl,r,it,rcg]=pcg(A,f,1e-6,20,Mfun);
[u,fl,r,it,rgmres]=gmres(A,f,[],1e-6,10,Mfun);
rgmres=rgmres*ri(1);                             % plot absolute residual
semilogy(0:length(ri)-1,ri,'-o',0:length(rcg)-1,rcg,'-+',0:length(rgmres)-1,rgmres,'-d');
xlabel('iteration')
ylabel('residual')
legend('Iterative','CG','GMRES')
```

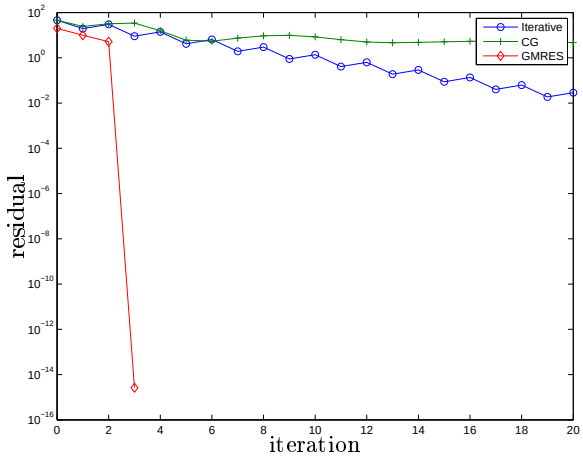


Figure 3.8 – Schwarz additif restreint utilisé comme préconditionneur pour le gradient conjugué (CG) ou GMRES, comparé à la méthode itérative

Ici la méthode de Krylov GMRES converge à la troisième itération, une de moins que pour la méthode de Schwarz additif avec le gradient conjugué, car dans la matrice $I - M$ du système préconditionné, la matrice M est maintenant de rang 2.

Préconditionnement par sous-structuration Une autre façon d'accélérer les méthodes de Schwarz itératives par des méthodes de Krylov est de les interpréter comme des algorithmes portant sur les variables d'interface, ici u_a et u_b. Pour cela, définissons les opérateurs de trace discrets

$$\begin{aligned} G_1 : \mathbb{R}^{a+d-1} &\to \mathbb{R}, & (u_1, \ldots, u_a, \ldots, u_{b-1}) &\mapsto u_a, \\ G_2 : \mathbb{R}^{J-a} &\to \mathbb{R}, & (u_{a+1}, \ldots, u_b, \ldots, u_J) &\mapsto u_b, \end{aligned}$$

et les relèvements

$$\begin{aligned} E_1 : \mathbb{R} &\to \mathbb{R}^{b-1}, & u_b &\mapsto (0, \ldots, 0, u_b), \\ E_2 : \mathbb{R} &\to \mathbb{R}^{J-a}, & u_a &\mapsto (u_a, 0, \ldots, 0). \end{aligned}$$

Appliquons à l'itération de Schwarz parallèle (3.13) l'opérateur de trace $\left(\begin{smallmatrix} G_1 & 0 \\ 0 & G_2 \end{smallmatrix}\right)$. Puisque

$$\tilde{B}_1 \boldsymbol{u}_2 = -\frac{1}{h^2} E_1 (u_2)_b \ , \quad \tilde{B}_2 \boldsymbol{u}_1 = -\frac{1}{h^2} E_2 (u_1)_a \ ,$$

nous pouvons écrire un algorithme portant sur les seules inconnues d'interface $(u_1^n)_a$ et $(u_2^n)_b$:

$$\begin{aligned} (u_1^{n+1})_a &= \frac{1}{h^2} G_1 \, A_1^{-1} \, E_1 (u_2^n)_b + G_1 \, A_1^{-1} \, \boldsymbol{f}_1, \\ (u_2^{n+1})_b &= \frac{1}{h^2} G_2 \, A_2^{-1} \, E_2 (u_1^n)_a + G_2 \, A_2^{-1} \, \boldsymbol{f}_2 \ . \end{aligned}$$

L'algorithme de Schwarz parallèle se réécrit donc sous forme d'un algorithme portant uniquement sur les inconnues d'interface $(g_1^n, g_2^n) = ((u_1^n)_a, (u_2^n)_b)$:

$$\begin{pmatrix} g_1^{n+1} \\ g_2^{n+1} \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} 0 & G_1 \, A_1^{-1} \, E_1 \\ G_2 \, A_2^{-1} \, E_2 & 0 \end{pmatrix} \begin{pmatrix} g_1^n \\ g_2^n \end{pmatrix} + \begin{pmatrix} G_1 \, A_1^{-1} \, \boldsymbol{f}_1 \\ G_2 \, A_2^{-1} \, \boldsymbol{f}_2 \end{pmatrix}. \tag{3.25}$$

Cet algorithme peut être vu, de nouveau, comme un algorithme de Jacobi par blocs (ou de Richardson, car la diagonale est l'identité) pour le système

$$\begin{pmatrix} I & -\frac{1}{h^2} G_1 \, A_1^{-1} \, E_1 \\ -\frac{1}{h^2} G_2 \, A_2^{-1} \, E_2 & I \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} G_1 \, A_1^{-1} \, \boldsymbol{f}_1 \\ G_2 \, A_2^{-1} \, \boldsymbol{f}_2 \end{pmatrix}. \tag{3.26}$$

Ce dernier système peut alors être résolu par un algorithme de Krylov. Le script suivant contient les algorithmes de Schwarz et Krylov en version sous-structurée.

```
Bar;                                    % to include problem parameters
a=8; d=4;                               % subdomain decomposition
f1=f(1:a+d-1); f2=f(a+1:J);             % subdomain source terms
x1=x(1:a+d+1); x2=x(a+1:end);           % finite difference meshes
G1=zeros(1,a+d+1);G1(end-d)=1;          % construct substructured system:
G2=zeros(1,J-a+2);G2(d+1)=1;            %
b=[G1*Solve1d(f1,eta,x1(1),x1(end),gg,0);% T*g=b, g unknowns at interfaces
   G2*Solve1d(f2,eta,x2(1),x2(end),0,gd)];
T=@(g) [g(1)-G1*Solve1d(zeros(size(f1)),eta,x1(1),x1(end),0,g(2));
        g(2)-G2*Solve1d(zeros(size(f2)),eta,x2(1),x2(end),g(1),0)];
g=[0;0];                                % zero initial guess
ri(1)=norm(b-T(g));                     % initial residual
for i=1:20                              % classial Schwarz iteration
  g=g-T(g)+b;                           % gn=(I-T)*g+b
  ri(i+1)=norm(b-T(g));                 % keep residual for plotting
end
[g,fl,r,it,rk]=gmres(T,b);              % use Krylov solver
rk(end+1)=norm(b-T(g));                 % add residual of result
semilogy(0:length(ri)-1,ri,'-o',0:length(rk)-1,rk,'-+');
xlabel('iteration'); ylabel('residual'); legend('Iterative','Krylov')
```

La figure 3.9 compare la convergence des deux algorithmes, itératif ou préconditionneur pour Krylov, pour un recouvrement de 4 points de grille.

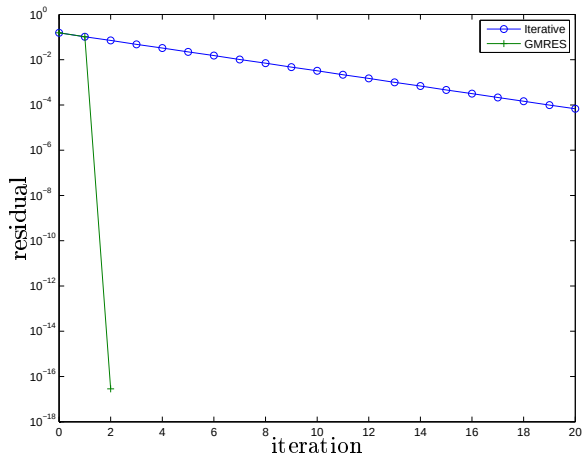


Figure 3.9 – Système sous-structuré résolu par méthode de Krylov (GMRES) et itérative

Le problème sous-structuré est un système de deux équations à deux inconnues g_1 et g_2. La méthode de Krylov converge donc en deux itérations.

Remarque 3.4 ***1.*** *Dans le cas où la matrice A est symétrique définie positive, le préconditionneur AS dans (3.19) l'est aussi, ce qui permet d'utiliser la méthode du gradient conjugué, qui est la méthode de Krylov de choix pour les matrices symétriques définies positives, car elle converge d'une manière optimale avec une récurrence courte. Il n'en est plus de même pour le préconditionneur RAS, voir (3.21), et le problème d'interface* (3.26)*. Cet avantage de AS disparaît lorsque le problème de départ n'est pas symétrique (équation d'advection-diffusion par exemple), ou pas défini positif (équation de Helmholtz par exemple).*

2. *En dimension $d \geq 1$, si le nombre de points dans chaque direction est de l'ordre de J, la dimension du système d'interface* (3.26) *est J^{d-1} tandis que celle du système complet préconditionné en volume (par exemple (3.19) pour AS) est J^d. Nous voyons maintenant que la taille du système d'interface* (3.26) *le rend plus avantageux dans l'utilisation d'une méthode de Krylov, où il faut stocker*

les directions de descente. Ceci est vrai en particulier si l'on utilise la méthode GMRES, où il faut stocker toutes les directions de descente.

3. *Il est possible d'utiliser des solveurs approchés pour la résolution dans les sous-domaines dans les formulations non sous-structurées, i.e. les méthodes de Schwarz discrètes, car la solution du problème de départ n'en est pas affectée. On voit cela clairement par exemple dans la formulation de Schwarz additif (3.19) : si l'on remplace les matrices A_i^{-1} par des approximations, la solution $\boldsymbol{u}$ de (3.19) ne change pas, car le système préconditionné est équivalent au système de départ. Par contre l'utilisation de solveurs approchés dans les formulations sous-structurées des méthodes de Schwarz changent la solution, comme on le voit clairement dans* (3.26) *: si l'on remplace maintenant les matrices A_i^{-1} par des approximations, la matrice du système d'interface change, et donc également la solution (g_1, g_2) à l'interface.*

La formulation sous-structurée des méthodes de Schwarz peut également être décrite sur le problème continu. Pour g_1 et g_2 dans $\mathbb{R}$, définissons $\mathcal{U}_1^{\mathcal{D}}(g_2, g_g, f)$ et $\mathcal{U}_2^{\mathcal{D}}(g_1, g_d, f)$ solutions des problèmes aux limites

$$\begin{aligned} &-\frac{d^2u_1}{dx^2} + \eta\, u_1 = f \text{ dans } \Omega_1, && -\frac{d^2u_2}{dx^2} + \eta\, u_2 = f \text{ dans } \Omega_2, \\ &u_1(0) = g_g \ , \ u_1(\beta) = g_2, && u_2(\alpha) = g_1 \ , \ u_2(1) = g_d. \end{aligned} \tag{3.27}$$

On a donc pour la méthode alternée de Schwarz

$$u_1^{n+1} = \mathcal{U}_1^{\mathcal{D}}(u_2^n(\beta), g_g, f), \quad u_2^{n+1} = \mathcal{U}_2^{\mathcal{D}}(u_1^{n+1}(\alpha), g_d, f),$$

et le problème en variable d'interface s'écrit au moyen des opérateurs de trace $\gamma_1 : u \to u(\alpha)$ et $\gamma_2 : u \to u(\beta)$:

$$u_1^{n+1}(\alpha) = \gamma_1\, \mathcal{U}_1^{\mathcal{D}}(u_2^n(\beta), g_g, f), \quad u_2^{n+1}(\beta) = \gamma_2\, \mathcal{U}_2^{\mathcal{D}}(u_1^{n+1}(\alpha), g_d, f),$$

ou encore, en notant $g_2^n = u_2^n(\beta)$ et $g_1^n = u_1^n(\alpha)$,

$$g_1^{n+1} = \gamma_1\, \mathcal{U}_1^{\mathcal{D}}(g_2^n, g_g, f), \quad g_2^{n+1} = \gamma_2\, \mathcal{U}_2^{\mathcal{D}}(g_1^{n+1}, g_d, f),$$

ce qui donne matriciellement l'itération

$$\begin{pmatrix} 1 & 0 \\ -\gamma_2\, \mathcal{U}_2^{\mathcal{D}}(\cdot, 0, 0) & 1 \end{pmatrix} \begin{pmatrix} g_1^{n+1} \\ g_2^{n+1} \end{pmatrix} = \begin{pmatrix} 0 & \gamma_1\, \mathcal{U}_1^{\mathcal{D}}(\cdot, 0, 0) \\ 0 & 0 \end{pmatrix} \begin{pmatrix} g_1^n \\ g_2^n \end{pmatrix} + \begin{pmatrix} \gamma_1\, \mathcal{U}_1^{\mathcal{D}}(0, g_g, f) \\ \gamma_2\, \mathcal{U}_2^{\mathcal{D}}(0, g_d, f) \end{pmatrix}.$$

Nous reconnaissons un algorithme de Gauss-Seidel pour la résolution du problème d'interface

$$\begin{pmatrix} 1 & -\gamma_1\, \mathcal{U}_1^{\mathcal{D}}(\cdot, 0, 0) \\ -\gamma_2\, \mathcal{U}_2^{\mathcal{D}}(\cdot, 0, 0) & 1 \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} \gamma_1\, \mathcal{U}_1^{\mathcal{D}}(0, g_g, f) \\ \gamma_2\, \mathcal{U}_2^{\mathcal{D}}(0, g_d, f) \end{pmatrix}. \tag{3.28}$$

Avec les notations précédentes, l'algorithme de Schwarz parallèle s'écrit aussi en formulation sous-structurée à l'interface,

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} g_1^{n+1} \\ g_2^{n+1} \end{pmatrix} = \begin{pmatrix} 0 & \gamma_1\, \mathcal{U}_1^{\mathcal{D}}(\cdot, 0, 0) \\ \gamma_2\, \mathcal{U}_2^{\mathcal{D}}(\cdot, 0, 0) & 0 \end{pmatrix} \begin{pmatrix} g_1^n \\ g_2^n \end{pmatrix} + \begin{pmatrix} \gamma_1\, \mathcal{U}_1^{\mathcal{D}}(0, g_g, f) \\ \gamma_2\, \mathcal{U}_2^{\mathcal{D}}(0, g_d, f) \end{pmatrix}.$$

ce qui est un algorithme de Jacobi pour la résolution du problème d'interface (3.28) .

Rayon spectral et conditionnement en dimension 2 En dimension 1, le problème sous-structuré en variables d'interface ne comporte que deux inconnues, et les méthodes de Krylov convergent alors en au plus deux iterations. Pour obtenir un problème d'interface intéressant, il faut passer en dimension 2. Considérons donc l'équation $\eta u - \Delta u = f$ dans le rectangle $\Omega = [0,1] \times [0,1]$. Le domaine est partagé en deux sous-domaines constitués de deux rectangles $\Omega_1 = [0,\beta] \times [0,1]$ et $\Omega_2 = [\alpha,1] \times [0,1]$. Les méthodes de Schwarz alternée et parallèle gardent la même forme, avec les conditions aux limites portant maintenant sur toute l'étendue des segments $\Gamma_1 = \{\beta\} \times [0,1]$ et $\Gamma_2 = \{\alpha\} \times [0,1]$. Les erreurs peuvent être développées en séries de sinus dans la variable y :

$$e_i^n(x,y) = \sum_{k=0}^{\infty} \hat{e}_i^n(x,k) \sin k\pi y.$$

L'équation $\eta e_i^n - \Delta e_i^n = 0$ devient après développement en série de sinus l'équation monodimensionnelle (2.13), où η est remplacé par $\eta + k^2\pi^2$. Cette équation est résolue comme dans la démonstration du théorème 3.1, et l'erreur dans le sous-domaine i à l'étape n est alors donnée par

$$e_i^n(x,y) = \sum_k \hat{e}_i^n(k) \mathrm{sh}\sqrt{\eta + k^2\pi^2}\, x \sin k\pi y.$$

Pour l'algorithme de Schwarz parallèle par exemple, on obtient

$$\hat{e}_1^{n+1}(k) = \rho_1(k)\hat{e}_2^n(k), \quad \hat{e}_2^{n+1}(k) = \rho_2(k)\hat{e}_1^n(k),$$

avec

$$\rho_1(k) = \frac{\mathrm{sh}(\sqrt{\eta + k^2\pi^2}\,(1-\beta))}{\mathrm{sh}(\sqrt{\eta + k^2\pi^2}\ \beta)}, \quad \rho_2(k) = \frac{\mathrm{sh}(\sqrt{\eta + k^2\pi^2}\,\alpha)}{\mathrm{sh}(\sqrt{\eta + k^2\pi^2}\,(1-\alpha))}. \tag{3.29}$$

On obtient de nouveau une formule de récurrence sur une double itération :

$$\hat{e}_i^{n+1}(k) = \rho^{(\mathcal{D})}(k)\hat{e}_i^{n-1}(k), \quad \text{avec } \rho^{(\mathcal{D})}(k) = \rho_1(k)\rho_2(k)\ . \tag{3.30}$$

Théorème 3.3 *L'algorithme de Schwarz parallèle en dimension 2 est convergent. Avec un recouvrement $\delta := \beta - \alpha$, le facteur de convergence $\rho^{(\mathcal{D})}(k)$ satisfait*

$$\sup_k \rho^{(\mathcal{D})}(k) = 1 - \mathcal{O}(\delta).$$

Démonstration Comme en dimension 1, il est facile de voir que $\rho^{(\mathcal{D})}(k)$ est strictement compris entre 0 et 1. Mieux encore, un calcul direct montre que la fonction $k \mapsto \rho^{(\mathcal{D})}(k)$ est décroissante. Elle atteint son maximum en $k = 1$:

$$\rho^{(\mathcal{D})}(k) \leq \rho^{(\mathcal{D})}(1) < 1\ .$$

Étudions par exemple la suite e_1^{2n}. Puisque, pour tout k, $\hat{e}_1^{2n}(k) = (\rho^{(\mathcal{D})}(k))^n \hat{e}_1^0(k)$, nous pouvons écrire

$$e_1^{2n}(x,y) = \sum_k (\rho^{(\mathcal{D})}(k))^n \hat{e}_1^0(k) \mathrm{sh}\sqrt{\eta + k^2\pi^2}\, x \sin k\pi y,$$

et la norme L^2 de e_1^{2n} peut être calculée explicitement :

$$\begin{aligned} \|e_1^{2n}\|_{\Omega_1}^2 = \textstyle\int_{\Omega_1} |e_1^{2n}(x,y)|^2\, dxdy \quad &= \textstyle\sum_k (\rho^{(\mathcal{D})}(k))^{2n} |\hat{e}_1^0(k)|^2 \int_{\Omega_1} \mathrm{sh}^2\sqrt{\eta + k^2\pi^2}\, x \sin^2 k\pi y\, dxdy \\ &\leq (\rho^{(\mathcal{D})}(1))^{2n} \textstyle\sum_k |\hat{e}_1^0(k)|^2 \int_{\Omega_1} \mathrm{sh}^2\sqrt{\eta + k^2\pi^2}\, x \sin^2 k\pi y\, dxdy \\ &= (\rho^{(\mathcal{D})}(1))^{2n} \|e_1^0\|_{\Omega_1}^2, \end{aligned}$$

et nous obtenons finalement

$$\|e_1^{2n}\|_{\Omega_1} \leq (\rho^{(\mathcal{D})}(1))^n \|e_1^0\|_{\Omega_1}.$$

Puisque $\rho^{(\mathcal{D})}(1) < 1$, la suite e_1^{2n} tend vers 0 dans $L^2(\Omega_1)$, et donc la suite u_1^{2n} tend vers u. Il en est de même pour les itérées paires, et pour l'autre sous-domaine.

Le facteur $\rho^{(\mathcal{D})}(1)$ mesure la vitesse de convergence de la suite. Il est intéressant de connaître son comportement lorsque le recouvrement tend vers 0. Pour cela effectuons un développement limité de $\rho^{(\mathcal{D})}(1)$ pour δ petit en remplaçant β par $\alpha + \delta$. Commençons par ρ_1 :

$$\begin{aligned}
\rho_1(1) &= \frac{\mathrm{sh}(\sqrt{\eta+\pi^2}\,(1-\alpha-\delta))}{\mathrm{sh}(\sqrt{\eta+\pi^2}\,(\alpha+\delta))} \\
&= \frac{\mathrm{sh}(\sqrt{\eta+\pi^2}\,(1-\alpha))\mathrm{ch}(\sqrt{\eta+\pi^2}\,\delta)-\mathrm{ch}(\sqrt{\eta+\pi^2}\,(1-\alpha))\mathrm{sh}(\sqrt{\eta+\pi^2}\,\delta)}{\mathrm{sh}(\sqrt{\eta+\pi^2}\,\alpha)\mathrm{ch}(\sqrt{\eta+\pi^2}\,\delta))+\mathrm{ch}(\sqrt{\eta+\pi^2}\,\alpha)\mathrm{sh}(\sqrt{\eta+\pi^2}\,\delta))} \\
&= \frac{\mathrm{sh}(\sqrt{\eta+\pi^2}\,(1-\alpha))-\delta\sqrt{\eta+\pi^2}\,\mathrm{ch}(\sqrt{\eta+\pi^2}\,(1-\alpha))+\mathcal{O}(\delta^2)}{\mathrm{sh}(\sqrt{\eta+\pi^2}\,\alpha)+\sqrt{\eta+\pi^2}\,\mathrm{ch}(\sqrt{\eta+\pi^2}\,\alpha)+\mathcal{O}(\delta^2)} \\
&= \frac{\mathrm{sh}(\sqrt{\eta+\pi^2}\,(1-\alpha))}{\mathrm{sh}(\sqrt{\eta+\pi^2}\,\alpha)}\ \frac{1-\delta\sqrt{\eta+\pi^2}\coth(\sqrt{\eta+\pi^2}\,(1-\alpha))+\mathcal{O}(\delta^2)}{1+\delta\sqrt{\eta+\pi^2}\coth(\sqrt{\eta+\pi^2}\,\alpha)+\mathcal{O}(\delta^2)} \\
&= \frac{1}{\rho_2(1)}\ \left(1-\delta\sqrt{\eta+\pi^2}\,(\coth(\sqrt{\eta+\pi^2}\,(1-\alpha))+\coth(\sqrt{\eta+\pi^2}\,\alpha))+\mathcal{O}(\delta^2)\right),
\end{aligned}$$

si bien que

$$\rho^{(\mathcal{D})}(1) = \rho_1(1)\rho_2(1) = 1-\delta\sqrt{\eta+\pi^2}\,(\coth(\sqrt{\eta+\pi^2}\,(1-\alpha))+\coth(\sqrt{\eta+\pi^2}\,\alpha))+\mathcal{O}(\delta^2),$$

ce qui établit le développement limité du théorème. ■

Comme en dimension 1, cet algorithme correspond à un algorithme de Jacobi pour le problème d'interface (3.28).

On se donne maintenant une grille (i,j), isotrope pour simplifier, c'est-à-dire $h_x = h_y = h$, et le schéma aux différences finies pour $u_{i,j} \sim u(x_i, y_j)$,

$$-\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2} - \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h^2} + \eta\, u_{i,j} = f_{i,j},$$

avec les conditions aux limites $u_{0,j} = (g_g)_j$ et $u_{J+1,j} = (g_d)_j$ pour $0 \le j \le J+1$, et $u_{i,0} = u_{i,J+1} = 0$ pour $0 \le i \le J+1$. Voici notre code Matlab qui construit la matrice associée en dimension deux à partir de la matrice `A1d` en dimension 1, à l'aide du produit tensoriel, ou produit de Kronecker.

```
function A=A2d(eta,Jx,Jy);
% A2D finite difference approximation of eta-Delta in 2d
%   A=A2d(eta,Jx,Jy); constructs the finite difference approximation to
%   eta-Delta on a Jx x Jy grid with spacing h=1/(Jy+1) and homogeneous
%   Dirichlet conditions all around

h=1/(Jy+1); Dxx=A1d(eta,0,h*(Jx+1),Jx); Dyy=A1d(0,0,1,Jy);
A=kron(speye(size(Dxx)),Dyy)+kron(Dxx,speye(size(Dyy)));
```

Les valeurs propres de la matrice A associée sont

$$\lambda^d_{j,k} = \eta + \frac{4}{h^2}(\sin^2\frac{j\pi\,h}{2} + \sin^2\frac{k\pi\,h}{2}) = \lambda^d_j + \lambda^d_k - \eta, \quad 1 \le j,k \le J,$$

où λ^d_j est définie dans (3.22) et contient η, ce qui explique pourquoi il faut enlever de nouveau un η après la somme de l'expression à droite. Le conditionnement de la matrice A est donné par

$$\kappa(A) = \frac{2\lambda^d_J - \eta}{2\lambda^d_1 - \eta} \sim \frac{2}{\eta + 2\pi^2}\,\frac{1}{h^2}. \tag{3.31}$$

Voici une implémentation en dimension 2 de notre solveur en Matlab :

```
function u=Solve2d(f,eta,ai,bi,gg,gd)
% SOLVE2D solves 2d eta-Delta using a finite difference approximation
%   u=Solve2d(f,eta,a,b,gg,gd) solves the two dimensional equation
%   (eta-Delta)u=f on the domain Omega=(ai*h,bi*h)x(0,1) with
%   Dirichlet boundary conditions u=gg at x=ai*h and u=gd at x=bi*h
%   and u=0 at y=0 and y=1 using a finite difference approximation
%   with interior grid points (bi-ai) times length(gg) using the same
%   mesh size h=1/(length(gg)+1) in both x and y.

nx=bi-ai-1;
ny=length(gg);
h=1/(length(gg)+1);
A=A2d(eta,nx,ny);
f(1:ny,1)=f(1:ny,1)+gg/h^2;          % add boundary conditions into rhs
f(1:ny,end)=f(1:ny,end)+gd/h^2;
u=A\f(:);
u=reshape(u,ny,nx);
u=[gg u gd];                         % add boundary values to solution
```

Nous l'utilisons maintenant pour calculer un exemple modèle, mais déjà intéressant : la distribution de température dans une pièce chauffée par un poêle situé en son centre (représenté par le second membre f). Les trois murs extérieurs sont froids, le mur ouest est chauffé par une pièce voisine, et contient une porte. Le petit programme Matlab `Room.m` ci-dessous effectue ce calcul, et trace le résultat figure 3.10.

```
eta=0; J=20;              % number of interior mesh points in x and y direction
x=0:1/(J+1):1; y=x;       % finite difference mesh, including boundary
f=zeros(J,J);             % source term does not include boundary
xi=x(2:end-1); yi=xi;
f([yi>0.4 & yi<0.6],[xi>0.4 & xi<0.6])=50;
gg=0.3*ones(J,1); gg(yi>0.5 & yi<0.9)=1;
gd=zeros(J,1);
u=Solve2d(f,eta,0,J+1,gg,gd);
u=[zeros(1,J+2);u;zeros(1,J+2)];
mesh(x(1:end),y,u); xlabel('x'); ylabel('y'); zlabel('solution');
```

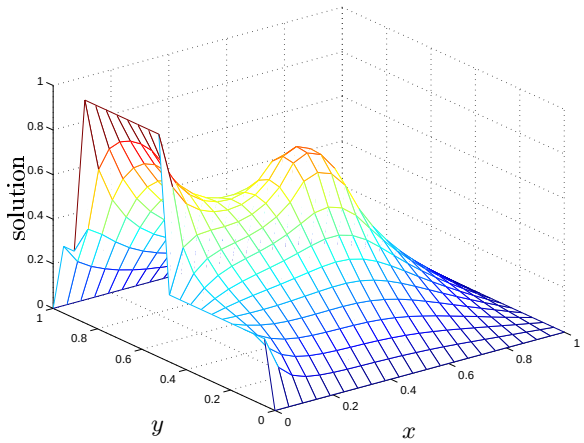


Figure 3.10 – Chauffage d'une pièce par un poêle

Voici maintenant le programme Matlab de la méthode de Schwarz parallèle discrétisée en dimension 2. Le recouvrement est de $d = 4$, soit $\beta - \alpha = 4h$. Les 6 premières itérations sont représentées figure 3.11.

```
Room;                                  % include problem parameters
a=8; d=4;                              % decomposition
f1=f(:,1:a+d-1); f2=f(:,a+1:end);
u1=[gg zeros(J,a+d)];                  % zero initial guess, except boundary value
u2=[zeros(J,J-a+1) gd];
h=1/(J+1); y=0:h:1;                    % finite difference meshes
x1=0:h:(a+d)*h; x2=a*h:h:1;
z1=zeros(1,a+d+1);z2=zeros(1,J-a+2); % for plotting purposes
for i=1:20
  u1n=Solve2d(f1,eta,0,a+d,gg,u2(:,d+1));
  u2n=Solve2d(f2,eta,a,J+1,u1(:,end-d),gd);
  u1=u1n;
  u2=u2n;
  mesh(x1,y,[z1;u1n;z1]); hold on;mesh(x2,y,[z2;u2n;z2]);hold off
  xlabel('x'); ylabel('y'); zlabel('Schwarz iterates');
  pause
end
```

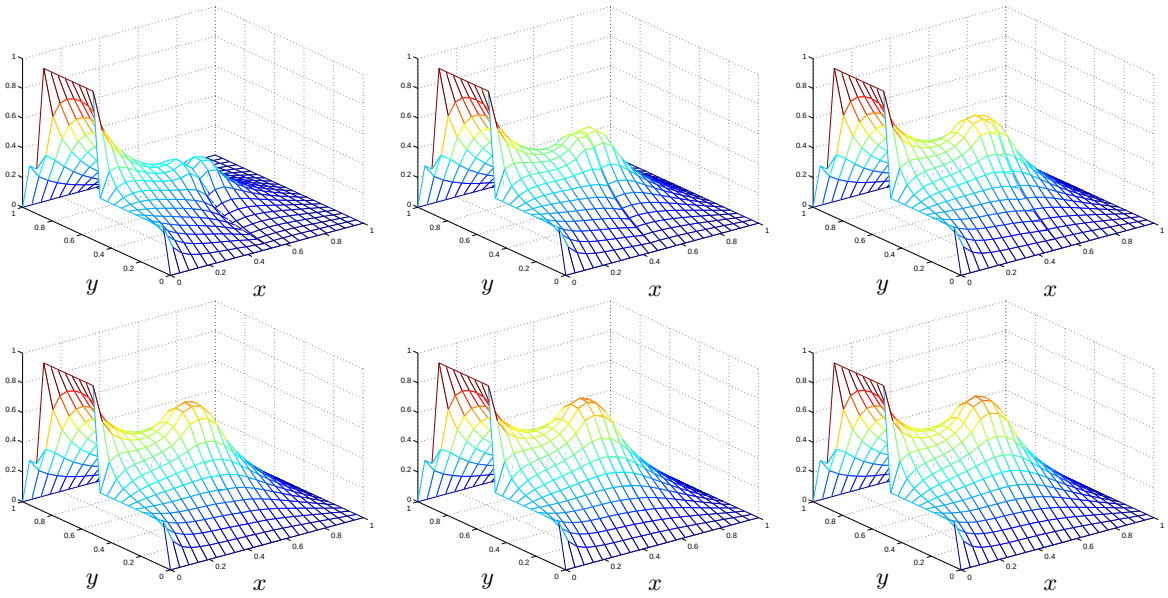


Figure 3.11 – Résolution du problème de chauffage par l'algorithme de Schwarz parallèle : 6 premières itérations

Si l'on essaie d'utiliser la méthode de Schwarz additive stationnaire (3.18), on constate, comme en dimension 1, des oscillations dans le recouvrement, représentées sur la figure 3.12.

```
Room;
a=8; d=4;                              % decomposition
h=1/(J+1);
f=f(:);
f(1:J)=f(1:J)+gg/h^2;                  % add boundary conditions into rhs
f(end-J+1:end)=f(end-J+1:end)+gd/h^2;
A=A2d(eta,J,J);
R1=[speye(J*(a+d-1)) sparse(J*(a+d-1),J*(J-a-d+1))];
R2=[sparse(J*(J-a),J*a) speye(J*(J-a))];
A1=R1*A*R1';
A2=R2*A*R2';
u=zeros(J*J,1);
x=0:h:1; y=x;                          % finite difference mesh, including boundary
```

```
for i=1:20
  r=f-A*u;
  u=u+(R1'*(A1\(R1*r))+R2'*(A2\(R2*r)));
  mesh(x,y,[zeros(1,J+2);gg reshape(u,J,J) gd;zeros(1,J+2)]);
  xlabel('x');  ylabel('y'); zlabel('Additive Schwarz iterates');
  pause
end
```

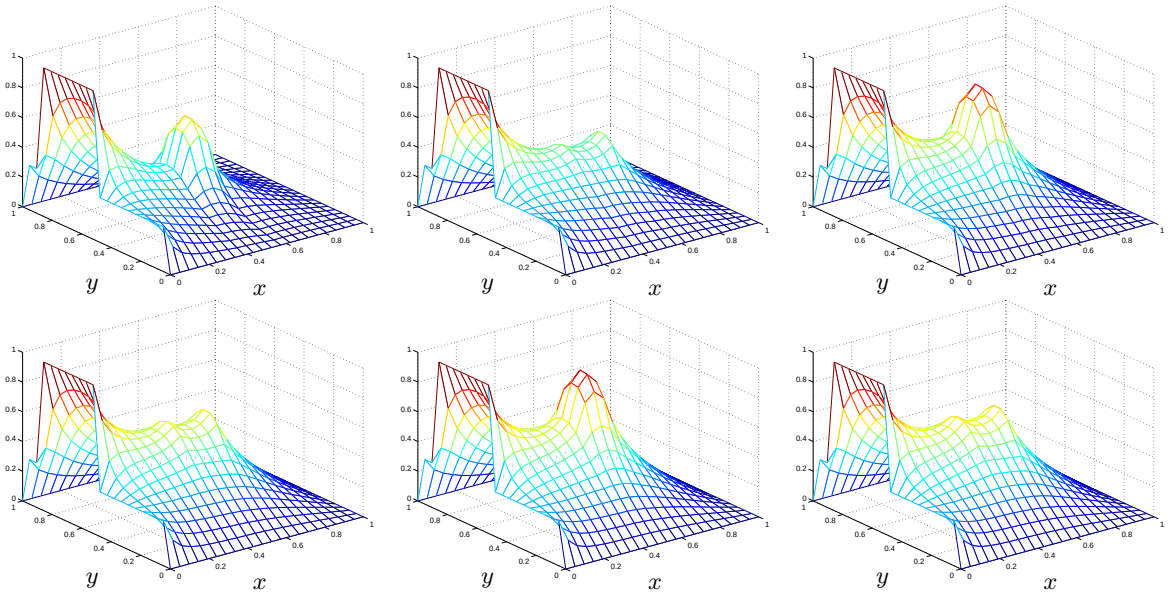


Figure 3.12 – Essai de résolution avec Schwarz additif, mettant en évidence le mode oscillatoire dans le recouvrement

Nous abordons maintenant les formulations de Schwarz considéré comme préconditionneur. Nous ne donnons ici que les résultats graphiques, les scripts sont analogues aux formulations en dimension 1.

D'abord les calculs en sous-structuration : nous comparons la résolution du système correspondant à (3.26) en dimension deux par méthode itérative ou méthode de Krylov. Les deux courbes de convergence sont représentées figure 3.13. Elles montrent bien le gain qu'apporte l'utilisation de l'algorithme de Krylov.

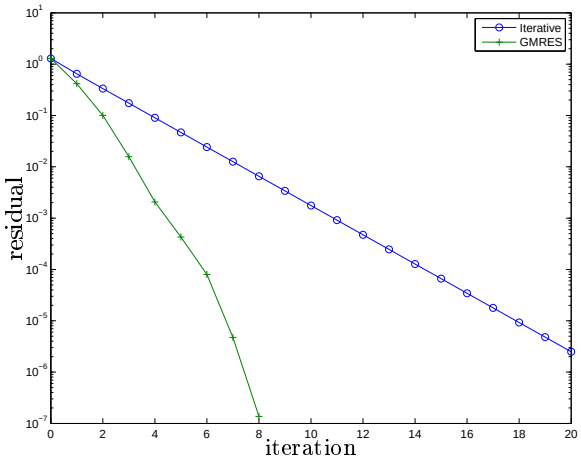


Figure 3.13 – Résolution du problème sous-structuré (3.26) en dimension 2 par méthode itérative ou GMRES

Pour le même cas test, avec la méthode de Schwarz additive (3.19), les résultats sont représentés figure 3.14. Ici, comme le problème est symétrique, l'algorithme du gradient conjugué peut être utilisé. Par contre, comme en dimension 1, l'itération stationnaire ne converge pas.

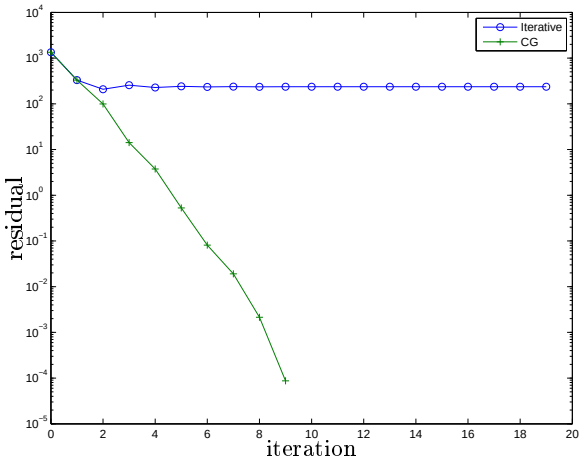


Figure 3.14 – Résolution en dimension 2 par Schwarz additif : version itérative ou gradient conjugué

Pour finir, nous avons représenté figure 3.15 les calculs correspondant à l'algorithme de Schwarz restreint (RAS). Nous comparons ici 3 méthodes : itérative, gradient conjugué, et GMRES qui est adapté aux matrices non symétriques.

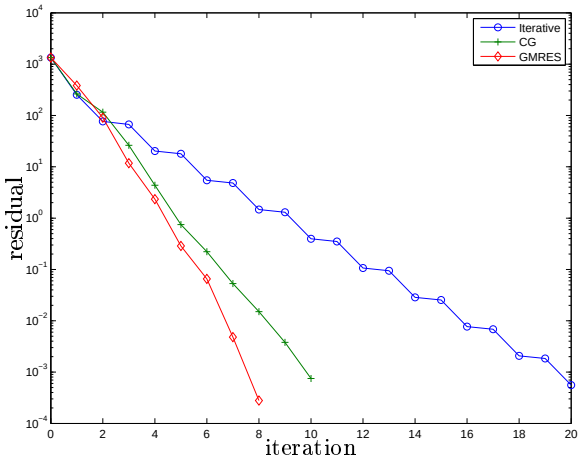


Figure 3.15 – Résolution en dimension 2 par Schwarz additif restreint : version itérative, gradient conjugué ou GMRES

Dans cet exemple, nous voyons que pour RAS, la méthode du gradient conjugué converge plutôt bien, compte-tenu du fait que RAS est un préconditionneur non-symétrique. Ce n'était pas le cas en dimension un. La méthode GMRES donne une convergence encore un peu meilleure, ce qui est logique dans la mesure où elle est adaptée aux préconditionneurs non-symétriques, cependant elle a un coût plus élevé que la méthode du gradient conjugué (voir [40]).

Nous représentons maintenant les spectres des matrices préconditionnées $M_{AS}^{-1}A$ et $M_{RAS}^{-1}A$, et nous étudions leur comportement lorsque le pas h tend vers 0. Rappelons que si le spectre de la matrice préconditionnée est proche de 1, un algorithme de Krylov pour la résolution du système est en général rapide. D'abord nous considérons dans la figure 3.16 le cas où le recouvrement est maintenu constant. De gauche à droite le pas est divisé par deux à chaque fois. Il y a donc de plus en plus de points dans le recouvrement.

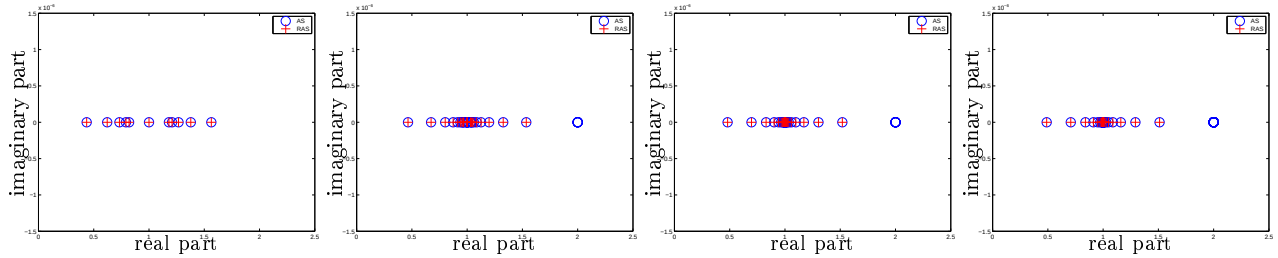


Figure 3.16 – Comparaison des spectres de AS and RAS comme préconditionneurs, avec recouvrement δ constant, indépendant du pas du maillage

Les spectres de AS et de RAS sont tous deux contenus dans le segment $[0, 2]$, et présentent peu de différences. Dans le cas d'un recouvrement minimal à gauche, les deux méthodes sont identiques, mais dès qu'il y a des points de grille dans le recouvrement, le mode non convergent de AS apparaît comme associé à la valeur propre supplémentaire de valeur 2. On voit aussi que si l'on raffine le maillage, l'intervalle du spectre reste inchangé. Ceci illustre le fait que ces méthodes convergent indépendamment de h lorsque le recouvrement est indépendant de h, voir théorème 3.3.

Dans la série de courbes de la figure 3.17, nous fixons maintenant le recouvrement à 2 points de grille, *i.e.* $\delta = 2h$, et nous faisons varier h comme dans la figure précédente. La taille du recouvrement tend donc vers 0 avec h.

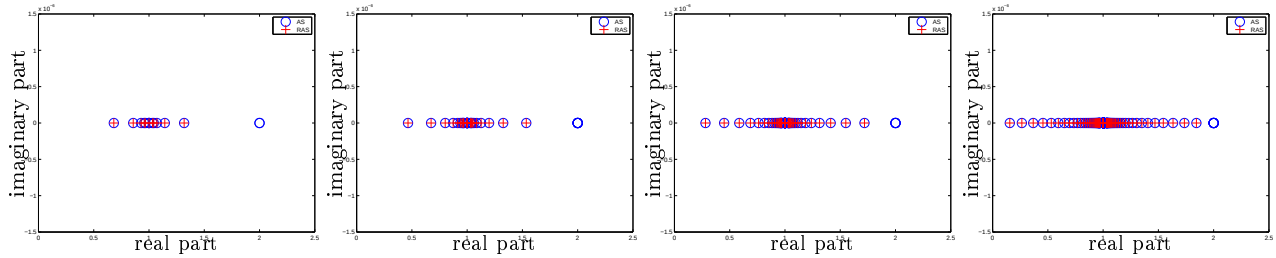


Figure 3.17 – Comparaison des spectres de AS et RAS comme prconditioneurs, avec recouvrement dépendant du pas de maillage, $\delta = 2h$

De nouveau, les spectres de AS et RAS sont très similaires, sauf pour les modes non-convergents de AS. Mais cette fois les spectres tendent à remplir tout l'intervalle $[0, 2]$ lorsque h diminue : les méthodes de Krylov utilisant ces préconditionneurs deviennent alors plus lents lorsque l'on raffine le maillage et que le recouvrement est de l'ordre de h.

3.5 Méthodes de Schwarz optimisées

Les méthodes de Schwarz optimisées établissent un pont entre les méthodes de Schwarz classiques, qui fonctionnent avec du recouvrement, et les méthodes de Schur que nous verrons plus tard, et qui n'utilisent pas de recouvrement. Elles permettent d'améliorer nettement les facteurs de convergence à la fois des méthodes stationnaires et des méthodes de Krylov, et peuvent être utilisées sans recouvrement, pour une introduction, voir [15]. Nous nous plaçons dès maintenant en dimension 2. Dans [28], Pierre-Louis Lions propose de remplacer les conditions de transmission portant sur la valeur de la fonction par des conditions de type Robin (ou encore Fourier en thermique), ce qui donne l'algorithme suivant :

$$\begin{aligned} \eta\, u_1^n - \Delta u_1^n &= f \quad \text{dans } \Omega_1, \\ u_1^n(0,\cdot) &= g_g, \\ (\partial_x + p) u_1^n(\beta,\cdot) &= (\partial_x + p) u_2^{n-1}(\beta,\cdot), \end{aligned} \tag{3.32a}$$

$$\begin{aligned} \eta\, u_2^n - \Delta u_2^n &= f \quad \text{dans } \Omega_2, \\ (\partial_x - p) u_2^n(\alpha,\cdot) &= (\partial_x - p) u_1^{n-1}(\alpha,\cdot), \\ u_2^n(1,\cdot) &= g_d. \end{aligned} \tag{3.32b}$$

Pour tout $p > 0$, les problèmes aux limites dans les sous-domaines sont bien posés : il existe une et une seule solution. On peut mettre les problèmes sous forme variationnelle, et les traiter par une méthode d'éléments finis. On peut aussi les résoudre par différences finies, c'est ce que nous faisons dans le script ci-dessous, où nous avons utilise une modification de notre procedure `Solve` pour tenir compte de conditions de Robin sur les bords à gauche et à droite :

```
function u=Solve2dR(f,eta,ai,bi,gg,gd,p1,p2)
% SOLVE2DR solves 2d eta-Delta using Robin conditions
%   u=Solve2dR(f,eta,a,b,gg,gd,n,p1,p2) solves the two dimensional
%   equation (eta-Delta)u=f on the domain Omega=(ai*h,bi*h)x(0,1) with
%   Robin boundary conditions (dn+p1)u=gg at x=ai*h and (dn+p2)u=gd at
%   x=bi*h and u=0 at y=0 and y=1 using a finite difference
%   approximation with interior grid points (bi-ai) times length(gg)
%   using the same mesh size h=1/(length(gg)+1) in both x and y.

nx=bi-ai+1;
ny=length(gg);
h=1/(length(gg)+1);
A=A2d(eta,nx,ny);
A(1:ny,1:ny)=A(1:ny,1:ny)/2+p1/h*speye(ny);
A(end-ny+1:end,end-ny+1:end)=A(end-ny+1:end,end-ny+1:end)/2+p2/h*speye(ny);
```

```
f(1:ny,1)=f(1:ny,1)/2+gg/h;          % add boundary conditions into rhs
f(1:ny,end)=f(1:ny,end)/2+gd/h;
u=A\f(:);
u=reshape(u,ny,nx);
```

Nous pouvons maintenant résoudre un problème encore plus réaliste de thermique dans une pièce, avec un mur isolant à l'est, modélisé par une condition de Robin. Avec le petit script Matlab suivant nommé `RoomR`, nous obtenons la distribution de température représentée figure 3.18.

```
eta=0;
J=20;                    % number of interior mesh points in x and y direction
x=0:1/(J+1):1;y=x;       % finite difference mesh, including boundary
f=zeros(J,J+2);          % source term includes Robin boundary
xi=x(2:end-1);yi=xi;
f([yi>0.4 & yi<0.6],[xi>0.4 & xi<0.6])=50;
gd=zeros(J,1);
gg=0.3*ones(J,1);
gg(yi>0.5 & yi<0.9)=1;
pe=1e5;                  % 1e5 to emulate a Dirichlet condition
pin=1;                   % 1 to emulate insulation
u=Solve2dR(f,eta,0,J+1,gg*pe,gd,pe,pin);
u=[zeros(1,J+2);u;zeros(1,J+2)];
mesh(x(1:end),y,u); xlabel('x'); ylabel('y'); zlabel('solution');
```

Remarquons dans ce code que la condition de Dirichlet à l'ouest est réalisée avec une condition de Robin de paramètre 10^5.

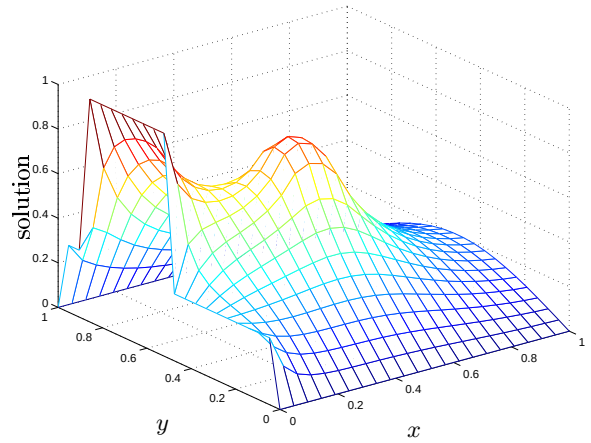


Figure 3.18 – Exemple de distribution de température avec une condition de Robin à l'est, représentant un mur isolé

La température sur le mur est n'est plus nulle comme dans l'exemple précédent figure 3.10, mais varie en fonction de la température intérieure.

La difficulté supplémentaire dans l'algorithme de Schwarz optimisé réside dans le choix du paramètre p. Il est calculé de la manière suivante. Supposons pour simplifier que la taille des sous-domaines est infinie dans la direction x. Nous pouvons reprendre l'analyse développée dans le théorème 3.3. Elle est un peu plus simple, puisque nous pouvons écrire

$$e_1^n(x,y) = \sum_k \hat{e}_1^n(k) e^{\sqrt{\eta+k^2\pi^2}\,x} \sin k\pi y,\ \ e_2^n(x,y) = \sum_k \hat{e}_2^n(k) e^{-\sqrt{\eta+k^2\pi^2}\,x} \sin k\pi y\ .$$

Les coefficients $\hat{e}_i^n(k)$ sont maintenant donnés par $\hat{e}_i^n(k) = \rho^{(\mathcal{R})}\hat{e}_i^{n-2}(k)$, avec

$$\rho^{(\mathcal{R})}(k,p) = \rho^{(\mathcal{D})}(k) \left(\frac{p-\sqrt{\eta+k^2\pi^2}}{p+\sqrt{\eta+k^2\pi^2}}\right)^2, \quad \rho^{(\mathcal{D})}(k) = e^{-2\sqrt{\eta+k^2\pi^2}\,\delta}.$$

Ici $\rho^{(\mathcal{D})}$ est le facteur de convergence de l'algorithme classique, il mesure l'importance du recouvrement. Le facteur $\rho^{(\mathcal{R})}$ peut être minimisé en choisissant judicieusement le paramètre p, voir [15], c'est-à-dire en résolvant le problème de min-max

$$\min_{p>0} \max_{1\le k\le J} \left|\frac{p-\sqrt{\eta+k^2\pi^2}}{p+\sqrt{\eta+k^2\pi^2}}\right|^2 e^{-2\sqrt{\eta+k^2\pi^2}\,\delta}.$$

Une étude approfondie de ce problème par des procédés classiques d'analyse (voir [15]) montre qu'il a une solution unique p^*, qui est donnée par les formules asymptotiques

Cas avec recouvrement : Si $\delta > 0$ et J grand, le paramètre p^* optimisé et le facteur de convergence correspondant ont un comportement asymptotique donné par

$$p^* \sim \left(\frac{\pi^2+\eta}{2\delta}\right)^{\frac{1}{3}}, \quad \max_{1\le k\le J} |\rho^{(\mathcal{R})}(k,p^*)| \sim 1 - 4\left(2\delta\sqrt{\pi^2+\eta}\right)^{\frac{1}{3}} = 1-\mathcal{O}(\delta^{1/3}).$$

Ces formules impliquent que si le recouvrement dépend de h, par exemple $\delta = \mathcal{O}(h)$, le facteur de convergence est asymptotiquement en $1-\mathcal{O}(h^{1/3})$.

Cas sans recouvrement : Si $\delta = 0$, le résultat de l'optimisation est donné par

$$p^* = \left((\pi^2+\eta)(\pi^2J^2+\eta)\right)^{1/4}, \quad \max_{1\le k\le J} |\rho^{(\mathcal{R})}(k,p^*)| = \left(\frac{\sqrt[4]{\pi^2J^2+\eta}-\sqrt[4]{\pi^2+\eta}}{\sqrt[4]{\pi^2J^2+\eta}+\sqrt[4]{\pi^2+\eta}}\right)^2 = 1-\mathcal{O}(J^{-1/2}).$$

Puisque $(J+1)h = 1$, nous avons maintenant un taux de convergence égal à $1-\mathcal{O}(h^{1/2})$.

Ces résultats doivent être mis en regard du facteur de convergence pour l'algorithme de Schwarz classique qui est $1-\mathcal{O}(h)$: la vitesse de convergence dépend plus faiblement du pas du maillage, et donc de la taille de la matrice. Voici une implémentation en Matlab de cette méthode de Schwarz optimisée :

```
RoomR;
ue=u;
a=8;d=4;
f1=f(:,1:a+d+1); f2=f(:,a+1:end);
u1=zeros(J,a+d+1);u2=zeros(J,J-a+2); % zero initial guess
h=1/(J+1);
x1=0:h:(a+d)*h; x2=a*h:h:1; y=0:h:1; % finite difference meshes
z1=zeros(1,a+d+1);z2=zeros(1,J-a+2); % for plotting purposes
p=((pi^2+eta)/(d*h))^(1/3);
e=ones(J,1);                         % construct normal derivatives
Na=[speye(J) -spdiags([-e (eta*h^2+4)*e -e]/2,[-1 0 1],J,J)]/h;
Nb=[-spdiags([-e (eta*h^2+4)*e -e]/2,[-1 0 1],J,J) speye(J)]/h;
for i=1:20
  tb=Nb*[u2(:,d+1);u2(:,d+2)]+f2(:,d+1)*h/2+p*u2(:,d+1);
  ta=Na*[u1(:,end-d-1);u1(:,end-d)]+f1(:,end-d)*h/2+p*u1(:,end-d);
  u1n=Solve2dR(f1,eta,0,a+d,pe*gg,tb,pe,p);
  u2n=Solve2dR(f2,eta,a,J+1,ta,gd,p,pin);
  u1=u1n;
  u2=u2n;
```

```
  mesh(x1,y,[z1;u1n;z1]); hold on;mesh(x2,y,[z2;u2n;z2]); hold off
  xlabel('x'); ylabel('y'); zlabel('Optimized Schwarz iterates');
  pause
end
```

Pour le cas test décrit sur la figure 3.18, nous représentons sur la figure 3.19 les trois premières itérations de cette méthode avec une valeur de p donnée par la formule asymptotique ci-dessus, soit $p = 3.7281$. On voit bien que la convergence est beaucoup plus rapide que pour les méthodes de Schwarz classiques illustrées figure 3.11 : la première itération est déjà une excellente approximation de la solution, et la troisième itération est indiscernable à l'œil nu de la solution convergée.

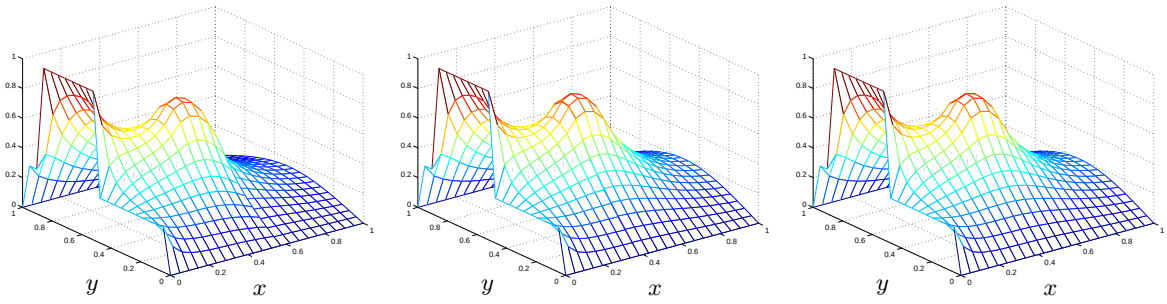


Figure 3.19 – Résolution du problème de chauffage figure 3.18 par la méthode de Schwarz optimisée

Comme pour les méthodes de Schwarz précédentes, nous allons mettre l'algorithme sous forme d'un algorithme de Jacobi par blocs pour le problème d'interface portant sur $g_2 = (\partial_x + p)(u_2)(\beta, \cdot)$ et $g_1 = (\partial_x - p)(u_1)(\alpha, \cdot)$. Introduisons les problèmes

$$\begin{array}{ll} \eta\, u_1 - \Delta u_1 = f \text{ dans } \Omega_1, & \eta\, u_2 - \Delta u_2 = f \text{ dans } \Omega_2, \\ u_1(0,\cdot) = g_g \ , \ (\partial_x u_1 + p u_1)(\beta,\cdot) = g_2, & (\partial_x u_2 - p u_2)(\alpha,\cdot) = g_1 \ , \ u_2(1,\cdot) = g_d. \end{array} \tag{3.33}$$

Pour $p \geq 0$, ces deux problèmes ont une solution unique, que nous appellerons $\mathcal{U}_1^{\mathcal{R}}(g_2, g_g, f)$ et $\mathcal{U}_2^{\mathcal{R}}(g_1, g_d, f)$. Supposons $g_1^n = (\partial_x - p)(u_1^n)(\alpha, \cdot)$ et $g_2^n = (\partial_x + p)(u_2^n)(\beta, \cdot)$ calculés à l'étape n. Alors une étape de l'algorithme de Schwarz optimisé s'écrit

$$g_1^{n+1} = ((\partial_x - p)\mathcal{U}_1^{\mathcal{R}}(g_2^n, g_d, f))(\alpha, \cdot), \quad g_2^{n+1} = ((\partial_x + p)\mathcal{U}_2^{\mathcal{R}}(g_1^n, g_g, f))(\beta, \cdot).$$

Il nous est maintenant facile d'écrire le problème d'interface sous forme de système linéaire

$$\begin{pmatrix} I & -(\partial_x - p)\mathcal{U}_1^{\mathcal{R}}(\cdot, 0, 0)(\alpha) \\ -(\partial_x + p)\mathcal{U}_2^{\mathcal{R}}(\cdot, 0, 0)(\beta) & I \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} (\partial_x - p)\mathcal{U}_1^{\mathcal{R}}(0, g_g, f)(\alpha) \\ (\partial_x + p)\mathcal{U}_2^{\mathcal{R}}(0, g_d, f)(\beta) \end{pmatrix}, \tag{3.34}$$

et de le traiter par l'algorithme GMRES. Sur la figure 3.20, nous traçons l'historique de convergence de l'algorithme sous-structuré itératif d'une part, accéléré par la méthode GMRES (donc où on résout le système sous-structuré (3.34) par GMRES), et les résultats correspondants pour la méthode de Schwarz classique de la figure 3.13.

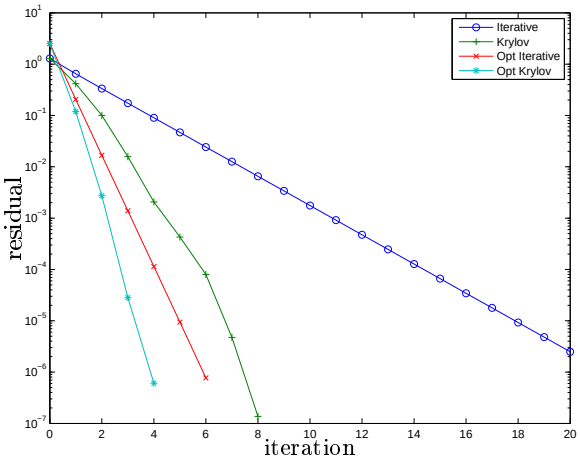


Figure 3.20 – Comparaison de Schwarz optimisé comme algorithme itératif et comme préconditionneur, et pour référence les résultats correspondants de la méthode de Schwarz classique. Historique de convergence

En comparant ces courbes de convergence, nous constatons deux choses : la méthode de Schwarz optimisée converge maintenant plus vite que la méthode de Schwarz classique accélérée par GMRES. Si l'on accélère maintenant la méthode de Schwarz optimisée par GMRES, on gagne encore en nombre d'itérations, mais le gain est moins important que pour la méthode de Schwarz classique : la méthode de Schwarz optimisée est dejà un très bon solveur itératif, même sans accélération par Krylov.

4 Méthodes de Schur

Historiquement les méthodes de Schur ont souvent été appelées méthodes de sous-structuration. Néanmoins nous avons vu que les méthodes de Schwarz peuvent aussi s'écrire sous forme sous-structurée, c'est pourquoi nous préférons utiliser la terminologie de *méthodes de Schur*. Il en existe deux variantes principales, la méthode de Schur primal, et la méthode de Schur dual. Nous verrons que les deux méthodes sont intimement liées, et chacune constitue un excellent préconditionneur pour l'autre.

4.1 Méthode de Schur primal

Suivons la démarche de Przemieniecki esquissée au paragraphe 2.2, avec les notations modernes. Reprenons le système linéaire (3.5) en dimension 1, $A\boldsymbol{u} = \boldsymbol{f}$, en écrivant d'abord les inconnues de $\Omega_1 =]0, \alpha[$, $\boldsymbol{u}_1 = (u_1, \cdots, u_{a-1})^T$, puis celles de $\Gamma = \{\alpha\}$, $u_\Gamma = u_a$, puis celles de $\Omega_2 =]\alpha, 1[$, $\boldsymbol{u}_2 = (u_{a+1}, \cdots, u_J)^T$. Ces trois blocs constituent le vecteur

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_1 \\ u_\Gamma \\ \boldsymbol{u}_2 \end{pmatrix} .$$

Le second membre $\boldsymbol{f}$ est décomposé de la même façon, et la matrice A est décomposée par blocs en

$$A = \left(\begin{array}{ccccc|c|ccccc}
\frac{2}{h^2}+\eta & -\frac{1}{h^2} & & & & & & & & & \\
-\frac{1}{h^2} & \frac{2}{h^2}+\eta & \ddots & & & & & & & & \\
 & \ddots & \ddots & \frac{1}{h^2} & & & & & & & \\
 & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & & & & & \\
\hline
 & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & & & & \\
\hline
 & & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & -\frac{1}{h^2} & & & & \\
 & & & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & \ddots & & & \\
 & & & & & & \ddots & \ddots & -\frac{1}{h^2} & & \\
 & & & & & & & -\frac{1}{h^2} & \frac{2}{h^2}+\eta & &
\end{array}\right),$$

soit sous forme compacte

$$A = \begin{pmatrix} A_{11} & A_{1\Gamma} & 0 \\ A_{\Gamma 1} & A_{\Gamma\Gamma} & A_{\Gamma 2} \\ 0 & A_{2\Gamma} & A_{22} \end{pmatrix}.$$

Les matrices A_{11} et A_{22} ont la même forme que A mais sont de taille respective $a-1$ et $J-a$, elles sont inversibles. $A_{\Gamma\Gamma}$ représente les échanges entre les points de Γ. En dimension 1 c'est une matrice scalaire, en dimension 2 ce sera une matrice bande. Les matrices $A_{i\Gamma}$ représentent le couplage entre les points intérieurs au domaine Ω_i et la frontière commune $\{\alpha\}$, et $A_{\Gamma i} = A_{i\Gamma}^T$ puisque la matrice A est symétrique. En dimension 1 ce sont des vecteurs. Les blocs concernant Γ sont donnés par

$$A_{\Gamma 1} = -\frac{1}{h^2}(0,\dots,0,1), \quad A_{\Gamma 2} = -\frac{1}{h^2}(1,0,\dots,0), \quad A_{\Gamma\Gamma} = \frac{2}{h^2}+\eta\ .$$

Le problème se réécrit par blocs

$$\begin{aligned} A_{11}\boldsymbol{u}_1 + A_{1\Gamma}u_\Gamma &= \boldsymbol{f}_1, \\ A_{22}\boldsymbol{u}_2 + A_{2\Gamma}u_\Gamma &= \boldsymbol{f}_2, \\ A_{\Gamma 1}\boldsymbol{u}_1 + A_{\Gamma 2}\boldsymbol{u}_2 + A_{\Gamma\Gamma}u_\Gamma &= f_\Gamma. \end{aligned} \tag{4.1}$$

Nous pouvons extraire $\boldsymbol{u}_1$ et $\boldsymbol{u}_2$ des deux premières équations,

$$\boldsymbol{u}_1 = A_{11}^{-1}(\boldsymbol{f}_1 - A_{1\Gamma}u_\Gamma), \quad \boldsymbol{u}_2 = A_{22}^{-1}(\boldsymbol{f}_2 - A_{2\Gamma}u_\Gamma), \tag{4.2}$$

et les reporter dans la troisième. Nous obtenons alors une équation (scalaire en dimension un) portant sur u_Γ,

$$(A_{\Gamma\Gamma} - A_{\Gamma 1}A_{11}^{-1}A_{1\Gamma} - A_{\Gamma 2}A_{22}^{-1}A_{2\Gamma})u_\Gamma = f_\Gamma - A_{\Gamma 1}A_{11}^{-1}\boldsymbol{f}_1 - A_{\Gamma 2}A_{22}^{-1}\boldsymbol{f}_2 := \tilde{f}_\Gamma\ . \tag{4.3}$$

Nous reconnaissons le système (2.7) de Przemienecki, nous l'appellerons formulation *Schur primal*. La matrice $S_P = A_{\Gamma\Gamma} - A_{\Gamma 1}A_{11}^{-1}A_{1\Gamma} - A_{\Gamma 2}A_{22}^{-1}A_{2\Gamma}$ est la *matrice du complément de Schur* introduite dans l'historique. Puisque le système (4.1) a une solution unique, la matrice S_P est inversible. En contraste avec l'algorithme de Schwarz, la méthode de Schur introduit un nouveau système, plus petit (puisque sa taille est égale au nombre de nœuds sur l'interface), mais ne fournit pas d'algorithme pour le résoudre. Le système (4.3) peut être formé et résolu par une méthode directe, ce qui était historiquement souvent le cas. Les matrices A_{ii} admettent chacune une décomposition $L_{ii}U_{ii}$, ces calculs peuvent être faits en

parallèle et sont regroupés pour calculer la matrice S_P. Ce système peut aussi être résolu par une méthode itérative, de préférence une méthode de Krylov. Dans ce cas, on ne forme pas la matrice S_P, puisqu'une itération revient à un produit matrice vecteur, qui correspond à une résolution par sous-domaine.

La méthode de Schur est par définition une méthode discrète, mais pour avoir une vision plus globale, nous allons en déduire une formulation continue. Pour cela, explicitons la troisième équation de (4.1) :

$$-\frac{1}{h^2}(u_1)_{a-1} + (\frac{2}{h^2} + \eta)u_\Gamma - \frac{1}{h^2}(u_2)_1 = f_\Gamma. \tag{4.4}$$

Étendons le vecteur $\boldsymbol{u}_1$ par $(u_1)_a = u_\Gamma$ et $\boldsymbol{u}_2$ par $(u_2)_0 = u_\Gamma$. Lorsque le pas h tend vers 0, $\boldsymbol{u}_1$ tend vers u_1, solution de l'équation continue (2.13) dans Ω_1, avec $u_1(0) = g_g$ et $u_1(\alpha) = u_\Gamma$, et de même pour $\boldsymbol{u}_2$. L'équation précédente se réécrit

$$-\frac{1}{h^2}u_1(\alpha - h) + (\frac{2}{h^2} + \eta)u_\Gamma - \frac{1}{h^2}u_2(\alpha + h) = f(\alpha) + \mathcal{O}(h^2),$$

ou encore

$$\frac{1}{h^2}(u_1(\alpha) - u_1(\alpha - h)) - \frac{1}{h^2}(u_2(\alpha + h) - u_2(\alpha)) + \eta u_\Gamma = f(\alpha) + \mathcal{O}(h^2). \tag{4.5}$$

Calculons par développements limités

$$u_1(\alpha - h) = u_1(\alpha) - h\frac{du_1}{dx}(\alpha) + \frac{h^2}{2}\frac{d^2u_1}{dx^2}(\alpha) + o(h^2),$$

et utilisons l'équation différentielle sur u_1 pour obtenir

$$u_1(\alpha - h) = u_1(\alpha) - h\frac{du_1}{dx}(\alpha) + \frac{h^2}{2}(\eta u_1(\alpha) - f(\alpha)) + o(h^2),$$

ce qui fournit l'approximation

$$\frac{du_1}{dx}(\alpha) = \frac{u_1(\alpha) - u_1(\alpha - h)}{h} + \frac{h}{2}(\eta u_1(\alpha) - f(\alpha)) + o(h). \tag{4.6}$$

De même

$$\frac{du_2}{dx}(\alpha) = \frac{u_2(\alpha + h) - u_2(\alpha)}{h} - \frac{h}{2}(\eta u_2(\alpha) - f(\alpha)) + o(h). \tag{4.7}$$

Ces deux relations nous indiquent que

$$\frac{u_1(\alpha) - u_1(\alpha - h)}{h^2} - \frac{u_2(\alpha + h) - u_2(\alpha)}{h^2} + \eta u_\Gamma - f(\alpha) = \frac{1}{h}\left(\frac{du_1}{dx}(\alpha) - \frac{du_2}{dx}(\alpha)\right) + o(1).$$

Ainsi (4.4) est la discrétisation de la condition de couplage

$$\frac{du_2}{dx}(\alpha) - \frac{du_1}{dx}(\alpha) = 0.$$

Par conséquent, le système (4.1) est une version discrète de la formulation continue

$$\begin{aligned} -\frac{d^2u_1}{dx^2} + \eta u_1 &= f \text{ dans } \Omega_1, \\ -\frac{d^2u_2}{dx^2} + \eta u_2 &= f \text{ dans } \Omega_2, \\ u_1(\alpha) = u_2(\alpha), \qquad & \frac{du_1}{dx}(\alpha) = \frac{du_2}{dx}(\alpha). \end{aligned} \tag{4.8}$$

C'est un système de deux équations différentielles dans les sous-domaines, couplées sur la frontière commune par l'égalité des fonctions et de leurs dérivées. Nous allons en déduire la formulation *Schur primal continue*, en appliquant sur le problème continu la démarche précédente. Pour cela nous définissons l'opérateur de Dirichlet-Neumann ou de Steklov-Poincaré comme suit. Reprenons les notations du paragraphe 3.4 en dimension 1, avec $\delta = 0$. Pour une donnée g, nous calculons la solution du problème aux limites dans Ω_i avec donnée g en $x = \alpha$ notée u_i, puis nous définissons les opérateurs $\mathcal{S}_i^{\mathcal{DN}}$ par

$$\mathcal{S}_1^{\mathcal{DN}}(g, g_g, f) = \frac{du_1}{dx}(\alpha), \quad \mathcal{S}_2^{\mathcal{DN}}(g, g_d, f) = \frac{du_2}{dx}(\alpha). \tag{4.9}$$

Les opérateurs $\mathcal{S}_i^{\mathcal{DN}}$ sont affines en la variable g, $\mathcal{S}_1^{\mathcal{DN}}(g, g_g, f) = \mathcal{S}_1^{\mathcal{DN}}(g, 0) + \mathcal{S}_1^{\mathcal{DN}}(0, g_g, f)$ et de même pour $\mathcal{S}_2^{\mathcal{DN}}$, si bien que l'équation de continuité des dérivées à l'interface dans (4.8) se réécrit, en posant $g = u_1(\alpha) = u_2(\alpha)$,

$$\mathcal{S}_P\, g := \mathcal{S}_1^{\mathcal{DN}}(g, 0, 0) - \mathcal{S}_2^{\mathcal{DN}}(g, 0, 0) = -\mathcal{S}_1^{\mathcal{DN}}(0, g_g, f) + \mathcal{S}_2^{\mathcal{DN}}(0, g_d, f). \tag{4.10}$$

Le système linéaire ci-dessus peut être résolu soit par méthode directe, soit par méthode itérative. En dimension 1, cela ne représente qu'une équation scalaire, mais en dimension supérieure, c'est un système qui couple toutes les inconnues à l'interface. Avant d'en expliciter une méthode de résolution, nous allons calculer le conditionnement du système en dimension 2. Pour cela développons g en séries de sinus dans la variable y :

$$g(y) = \sum_{k=1}^{+\infty} \hat{g}(k)\, \sin k\pi y.$$

Les images $\mathcal{S}_i^{\mathcal{DN}}(g, 0, 0)$ se décomposent de la même façon, et les coefficients sont donnés par :

$$\begin{aligned} \widehat{\mathcal{S}_1^{\mathcal{DN}}(g, 0, 0)}(k) &= \sqrt{\eta + k^2\pi^2}\, \coth(\sqrt{\eta + k^2\pi^2}\, \alpha)\, \hat{g}(k), \\ \widehat{\mathcal{S}_2^{\mathcal{DN}}(g, 0, 0)}(k) &= -\sqrt{\eta + k^2\pi^2}\, \coth(\sqrt{\eta + k^2\pi^2}\, (1-\alpha))\, \hat{g}(k). \end{aligned} \tag{4.11}$$

Le symbole de l'opérateur $\mathcal{S}_P$ (défini par $\widehat{\mathcal{S}_P g}(k) = \widehat{\mathcal{S}_P}(k)\hat{g}(k)$) est donc

$$\widehat{\mathcal{S}}_P(k) = \sqrt{\eta + k^2\pi^2}\, (\coth(\sqrt{\eta + k^2\pi^2}\, \alpha) + \coth(\sqrt{\eta + k^2\pi^2}\, (1-\alpha))).$$

La fonction $\widehat{\mathcal{S}}_P$ est croissante. Le conditionnement de $\mathcal{S}_P$ discrétisé sur la grille de taille J peut donc être estimé par

$$\mathcal{K}(\mathcal{S}_P) := \frac{\max\limits_{1 \le k \le J} (\widehat{\mathcal{S}}_P(k))}{\min\limits_{1 \le k \le J} (\widehat{\mathcal{S}}_P(k))} = \frac{\widehat{\mathcal{S}}_P(J)}{\widehat{\mathcal{S}}_P(1)},$$

et son comportement asymptotique en fonction de h est facile à calculer. Il est donné par

$$\begin{aligned} \mathcal{K}(\mathcal{S}_P) &= \frac{2\, \sqrt{\eta + J^2\pi^2}}{\sqrt{\eta + \pi^2}\, (\coth(\sqrt{\eta + \pi^2}\, \alpha) + \coth(\sqrt{\eta + \pi^2}\, (1-\alpha)))} (1 + \mathcal{O}(e^{-\sqrt{\eta + J^2\pi^2}\alpha})) \\ &= \mathcal{O}(J) = \mathcal{O}(h^{-1}) \text{ puisque } (J+1)h = 1. \end{aligned}$$

Rappelons que le conditionnement du problème de résolution du système linéaire de matrice (3.5) est $\mathcal{O}(h^{-2})$: la formulation sous forme de problème d'interface améliore donc le conditionnement d'un ordre de grandeur. Nous verrons plus tard sur la figure 4.2 une comparaison numérique des conditionnements sur notre problème modèle de la figure 3.10.

Voici maintenant un script Matlab de résolution de ce problème modèle par la méthode du complément de Schur primal. Le problème d'interface est résolu par la méthode du gradient conjugué, car il est symétrique pour une matrice de départ symétrique. La courbe de convergence est représentée figure 4.1.

```
Room;
a=10;                                   % decomposition
h=1/(J+1);
f=f(:);
f(1:J)=f(1:J)+gg/h^2;                   % add boundary conditions into rhs
f(end-J+1:end)=f(end-J+1:end)+gd/h^2;
A=A2d(eta,J,J);

A11=A(1:J*(a-1),1:J*(a-1));             % form block decomposition
AGG=A(J*(a-1)+1:J*a,J*(a-1)+1:J*a);
A22=A(J*a+1:J*J,J*a+1:J*J);
A1G=A(1:J*(a-1),J*(a-1)+1:J*a);
AG1=A(J*(a-1)+1:J*a,1:J*(a-1));
A2G=A(J*a+1:J*J,J*(a-1)+1:J*a);
AG2=A(J*(a-1)+1:J*a,J*a+1:J*J);

f1=f(1:J*(a-1));
fG=f(J*(a-1)+1:J*a);
f2=f(J*a+1:J*J);

Afun=@(x) AGG*x-AG1*(A11\(A1G*x))-AG2*(A22\(A2G*x));
ft=fG-AG1*(A11\f1)-AG2*(A22\f2);
[uG,fl,r,it,rcg]=pcg(Afun,ft,1e-6,100);
u1=A11\(f1-A1G*uG); u2=A22\(f2-A2G*uG);    % reconstruct global solution
u=[zeros(1,J+2);gg reshape([u1;uG;u2],J,J) gd;zeros(1,J+2)];

mesh(0:h:1,0:h:1,u); xlabel('x'); ylabel('y')
pause

semilogy(0:length(rcg)-1,rcg,'-+');
xlabel('iteration'); ylabel('residual'); legend('Primal Schur with CG')
```

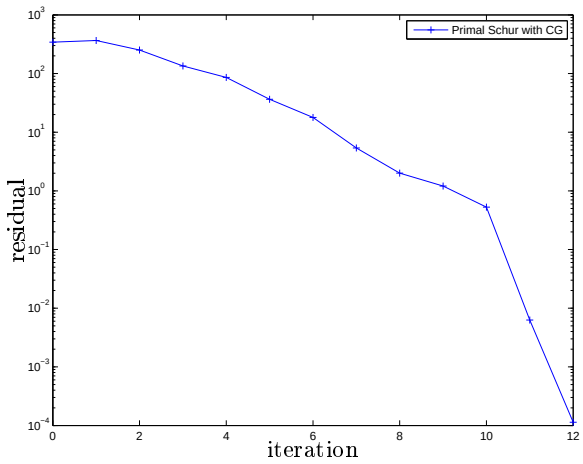


Figure 4.1 – Convergence de la méthode de Schur primal pour le problème modèle bidimensionnel de la figure 3.10

Considérons de nouveau le système (4.8). Dans la méthode précédente on considère que u_1 et u_2 ont la même valeur g sur l'interface, et on calcule le saut des dérivées en ce même point en fonction de g. Annuler cette quantité fournit l'équation sur g à résoudre.

On peut imaginer de faire l'inverse, c'est ce que l'on appelle la méthode de *Schur dual* que nous présentons maintenant.

4.2 Méthode de Schur dual

Donnons-nous un réel g, et calculons les analogues de (3.27), à ceci près que les conditions sur l'interface sont maintenant des conditions de Neumann :

$$\begin{array}{ll} -\dfrac{d^2u_1}{dx^2}+\eta\,u_1=f \text{ dans } \Omega_1, & -\dfrac{d^2u_2}{dx^2}+\eta\,u_2=f \text{ dans } \Omega_2,\\ u_1(0)=g_g\ ,\ \dfrac{du_1}{dx}(\alpha)=g, & \dfrac{du_2}{dx}(\alpha)=g\ ,\ u_2(1)=g_d. \end{array} \tag{4.12}$$

Les équations précédentes définissent une solution unique dans chaque sous-domaine, elles sont notées $u_1=U_1^{\mathcal{N}}(g,g_g,f)$ et $u_2=U_2^{\mathcal{N}}(g,g_d,f)$.

Remarque 4.1 *Lorsque $\eta=0$, l'existence et l'unicité dans les sous-domaines proviennent de la condition de Dirichlet sur un bord, qui permet d'utiliser l'inégalité de Poincaré. Dans le cas de trois sous-domaines, le problème du milieu est un problème de Neumann "pur", qui réclame une relation de compatibilité pour en assurer l'existence. L'unicité est alors obtenue modulo les constantes, ce qui fut au début considéré comme un défaut de la méthode. Une étape astucieuse dans le cadre des méthodes de FETI fut par la suite d'utiliser ces constantes libres dans chaque sous-domaine pour former une composante grille grossière naturelle, ce qui contribua grandement àla célébrité des méthodes FETI, que nous exposerons au paragraphe 4.3.*

Nous allons déduire maintenant la formulation *Schur dual continue*. Pour cela définissons les opérateurs de Neumann-Dirichlet

$$\mathcal{S}_1^{\mathcal{ND}}(g,g_g,f)=U_1^{\mathcal{N}}(g,g_g,f)(\alpha),\quad \mathcal{S}_2^{\mathcal{ND}}(g,g_d,f)=U_2^{\mathcal{N}}(g,g_d,f)(\alpha)\ . \tag{4.13}$$

L'équation $u_1(\alpha)=u_2(\alpha)$ prend la forme d'une équation linéaire portant sur la variable d'interface g:

$$\mathcal{S}_D\,g:=\mathcal{S}_1^{\mathcal{ND}}(g,0,0)-\mathcal{S}_2^{\mathcal{ND}}(g,0,0)=-\mathcal{S}_1^{\mathcal{ND}}(0,g_g,f)+\mathcal{S}_2^{\mathcal{ND}}(0,g_d,f). \tag{4.14}$$

Etudions maintenant ce système en dimension 2. Il est symétrique, et sa résolution par la méthode du gradient conjugué requiert à chaque étape la résolution de deux problèmes avec conditions aux limites de Neumann. Développons g en série de sinus dans la variable y comme précédemment. Les images $\mathcal{S}_i^{\mathcal{DN}}$ se décomposent de la même façon, avec

$$\begin{aligned} \widehat{\mathcal{S}_1^{\mathcal{ND}}(g,0,0)}(k)&=\frac{\mathrm{th}(\sqrt{\eta+k^2\pi^2}\,\alpha)}{\sqrt{\eta+k^2\pi^2}}\ \hat{g}(k),\\ \widehat{\mathcal{S}_2^{\mathcal{ND}}(g,0,0)}(k)&=-\frac{\mathrm{th}(\sqrt{\eta+k^2\pi^2}\,(1-\alpha))}{\sqrt{\eta+k^2\pi^2}}\ \hat{g}(k). \end{aligned} \tag{4.15}$$

Le symbole de l'opérateur $\mathcal{S}_D$ est donc

$$\widehat{\mathcal{S}}_D(k)=\frac{1}{\sqrt{\eta+k^2\pi^2}}(\mathrm{th}(\sqrt{\eta+k^2\pi^2}\,\alpha)+\mathrm{th}(\sqrt{\eta+k^2\pi^2}\,(1-\alpha))).$$

Cette fois la fonction $\widehat{\mathcal{S}}_D$ est décroissante. Le conditionnement de $\widehat{\mathcal{S}}_D$ sur la grille de taille J peut être estimé par

$$\begin{aligned} \mathcal{K}(\mathcal{S}_D) &= \frac{\widehat{\mathcal{S}}_D(1)}{\widehat{\mathcal{S}}_D(J)}=\frac{\mathrm{th}(\sqrt{\eta+\pi^2}\,\alpha)+\mathrm{th}(\sqrt{\eta+\pi^2}\,(1-\alpha))}{2\sqrt{\eta+\pi^2}}\sqrt{\eta+J^2\pi^2}(1+\mathcal{O}(e^{-\sqrt{\eta+J^2\pi^2}\alpha}))\\ &= \mathcal{O}(J)=\mathcal{O}(h^{-1}). \end{aligned}$$

Le conditionnement est du même ordre que celui de Schur primal. Voici un exemple en Matlab qui permet de comparer en dimension 2 le conditionnement de la matrice A aux conditionnements des matrices des méthodes de Schur primal et Schur dual.

```
for j=1:4
  J=10*2^(j-1); a=5*2^(j-1);
  eta=0;
  h(j)=1/(J+1);
  A=A2d(eta,J,J);

  A11=A(1:J*(a-1),1:J*(a-1));        % form block decomposition
  AGG=A(J*(a-1)+1:J*a,J*(a-1)+1:J*a);
  A22=A(J*a+1:J*J,J*a+1:J*J);
  A1G=A(1:J*(a-1),J*(a-1)+1:J*a);
  AG1=A(J*(a-1)+1:J*a,1:J*(a-1));
  A2G=A(J*a+1:J*J,J*(a-1)+1:J*a);
  AG2=A(J*(a-1)+1:J*a,J*a+1:J*J);

  A1=[A11 A1G
      AG1 AGG/2];
  A2=[AGG/2 AG2
      A2G A22];

  R1=[zeros(size(AG1)) speye(size(AGG))];
  R2=[speye(size(AGG)) zeros(size(AG2))];

  Acond(j)=condest(A);
  SPcond(j)=condest(AGG-AG1*(A11\A1G)-AG2*(A22\A2G));
  SDcond(j)=condest(R1*(A1\R1')+R2*(A2\R2'));
end;
loglog(h,Acond,'--b',h,SPcond,'--r',h,SDcond,'-.k',h,1./h.^2,'-b',h,1./h,'-r');
xlabel('mesh size h')
ylabel('condition number')
legend('A','Primal Schur','Dual Schur','h^{-1}','h^{-2}')
```

Les résultats sont portés sur la figure 4.2, où nous traçons les conditionnements de la matrice A, des matrices $\mathcal{S}_D$ et $\mathcal{S}_P$, en fonction du pas du maillage h. Les résultats sont en accord avec les estimations théoriques.

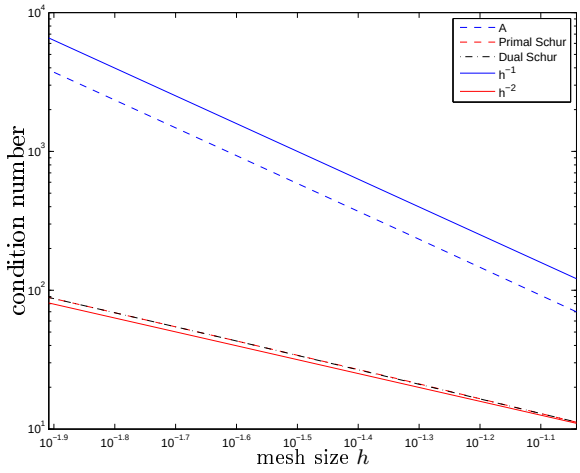


Figure 4.2 – Comparaison en dimension 2 des conditionnements du problème de départ et des méthodes de Schur primal et Schur dual

Écrivons maintenant en dimension 1 la formulation discrète associée à la méthode de Schur dual, avec les notations du paragraphe 4.1. Pour cela nous écrivons l'équation habituelle en tous les points

intérieurs, et nous ajoutons sur l'interface commun une relation obtenue au moyen de (4.6,4.7) :

$$g = \frac{(u_1)_a - (u_1)_{a-1}}{h} + \frac{h}{2}(\eta(u_1)_a - f_a) = \frac{(u_2)_{a+1} - (u_2)_a}{h} - \frac{h}{2}(\eta(u_2)_a - f_a),$$

ce qui donne un système global

$$\begin{array}{l} -\dfrac{(u_1)_{j+1} - 2(u_1)_j + (u_1)_{j-1}}{h^2} + \eta\,(u_1)_j = (f_1)_j, \quad 1 \le j \le a-1, \\ -\dfrac{1}{h^2}(u_1)_{a-1} + \dfrac{1}{2}(\eta + \dfrac{2}{h^2})(u_1)_a = \dfrac{1}{h}g + \dfrac{1}{2}f_a, \\ -\dfrac{(u_2)_{j+1} - 2(u_2)_j + (u_2)_{j-1}}{h^2} + \eta\,(u_2)_j = (f_2)_j, \quad a+1 \le j \le J, \\ -\dfrac{1}{h^2}(u_2)_{a+1} + \dfrac{1}{2}(\eta + \dfrac{2}{h^2})(u_2)_a = -\dfrac{1}{h}g + \dfrac{1}{2}f_a. \end{array} \tag{4.16}$$

Ce système s'écrit sous forme matricielle :

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & \frac{1}{2}A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ (u_1)_a \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \frac{1}{2}f_a + \frac{1}{h}g \end{pmatrix}, \tag{4.17}$$

$$\begin{pmatrix} \frac{1}{2}A_{\Gamma\Gamma} & A_{\Gamma 2} \\ A_{2\Gamma} & A_{22} \end{pmatrix} \begin{pmatrix} (u_2)_a \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}f_a - \frac{1}{h}g \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{4.18}$$

Définissons les opérateurs de trace discrets $\widetilde{G}_i$ (distincts des G_i définis au paragraphe 3.4 dans le cas avec recouvrement)

$$\begin{array}{lll} \widetilde{G}_1 : \mathbb{R}^a & \to \mathbb{R}, & (u_1, \ldots, u_a)^T \mapsto u_a, \\ \widetilde{G}_2 : \mathbb{R}^{J-a+1} & \to \mathbb{R}, & (u_a, \ldots, u_J)^T \mapsto u_a. \end{array}$$

Extrayons $(u_1)_a$ de (4.17) et $(u_2)_a$ de (4.18),

$$(u_1)_a = \widetilde{G}_1 \begin{pmatrix} \boldsymbol{u}_1 \\ (u_1)_a \end{pmatrix} = \widetilde{G}_1 \begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & \frac{1}{2}A_{\Gamma\Gamma} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{f}_1 \\ \frac{1}{2}f_a + \frac{1}{h}g \end{pmatrix}, \tag{4.19}$$

$$(u_2)_a = \widetilde{G}_2 \begin{pmatrix} (u_2)_a \\ \boldsymbol{u}_2 \end{pmatrix} = \widetilde{G}_2 \begin{pmatrix} \frac{1}{2}A_{\Gamma\Gamma} & A_{\Gamma 2} \\ A_{2\Gamma} & A_{22} \end{pmatrix}^{-1} \begin{pmatrix} \frac{1}{2}f_a - \frac{1}{h}g \\ \boldsymbol{f}_2 \end{pmatrix}. \tag{4.20}$$

En écrivant que ces deux expressions sont égales, nous obtenons une équation portant sur g :

$$\widetilde{G}_1 \begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & \frac{1}{2}A_{\Gamma\Gamma} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{f}_1 \\ \frac{1}{2}f_a + \frac{1}{h}g \end{pmatrix} - \widetilde{G}_2 \begin{pmatrix} \frac{1}{2}A_{\Gamma\Gamma} & A_{\Gamma 2} \\ A_{2\Gamma} & A_{22} \end{pmatrix}^{-1} \begin{pmatrix} \frac{1}{2}f_a - \frac{1}{h}g \\ \boldsymbol{f}_2 \end{pmatrix} = 0, \tag{4.21}$$

et par linéarité

$$\begin{array}{l} \left(\widetilde{G}_1 \begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & \frac{1}{2}A_{\Gamma\Gamma} \end{pmatrix}^{-1} \widetilde{G}_1^T + \widetilde{G}_2 \begin{pmatrix} \frac{1}{2}A_{\Gamma\Gamma} & A_{\Gamma 2} \\ A_{2\Gamma} & A_{22} \end{pmatrix}^{-1} \widetilde{G}_2^T \right) g = \\ -h\,\widetilde{G}_1 \begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & \frac{1}{2}A_{\Gamma\Gamma} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{f}_1 \\ \frac{1}{2}f_a \end{pmatrix} + h\,\widetilde{G}_2 \begin{pmatrix} \frac{1}{2}A_{\Gamma\Gamma} & A_{\Gamma 2} \\ A_{2\Gamma} & A_{22} \end{pmatrix}^{-1} \begin{pmatrix} \frac{1}{2}f_a \\ \boldsymbol{f}_2 \end{pmatrix}. \end{array} \tag{4.22}$$

Voici une implementation de la méthode Schur dual en Matlab, avec résolution du système par l'algorithme du gradient conjugué. La courbe de convergence est représentée figure 4.3.

```
Room;
a=10;                                % decomposition
h=1/(J+1);
f=f(:);
f(1:J)=f(1:J)+gg/h^2;                % add boundary conditions into rhs
```

```
f(end-J+1:end)=f(end-J+1:end)+gd/h^2;
A=A2d(eta,J,J);

A11=A(1:J*(a-1),1:J*(a-1));        % form block decomposition
AGG=A(J*(a-1)+1:J*a,J*(a-1)+1:J*a);
A22=A(J*a+1:J*J,J*a+1:J*J);
A1G=A(1:J*(a-1),J*(a-1)+1:J*a);
AG1=A(J*(a-1)+1:J*a,1:J*(a-1));
A2G=A(J*a+1:J*J,J*(a-1)+1:J*a);
AG2=A(J*(a-1)+1:J*a,J*a+1:J*J);

f1=f(1:J*(a-1));
fG=f(J*(a-1)+1:J*a);
f2=f(J*a+1:J*J);

A1=[A11 A1G
    AG1 AGG/2];
A2=[AGG/2 AG2
    A2G A22];

G1=[zeros(size(AG1)) speye(size(AGG))];
G2=[speye(size(AGG)) zeros(size(AG2))];

Afun=@(x) G1*(A1\(G1'*x))+G2*(A2\(G2'*x));
ft=-G1*(A1\[f1;fG/2])+G2*(A2\[fG/2;f2]);
[la,fl,r,it,rcg]=pcg(Afun,ft,1e-6,100);
u1=A1\(G1'*la+[f1;fG/2]);
u2=A2\(-G2'*la+[fG/2;f2]);
u=[zeros(1,J+2);gg reshape([u1;u2(J+1:end)],J,J) gd;zeros(1,J+2)];

mesh(0:h:1,0:h:1,u); xlabel('x'); ylabel('y');
pause

semilogy(0:length(rcg)-1,rcg,'-+');
xlabel('iteration'); ylabel('residual'); legend('Dual Schur with CG')
```

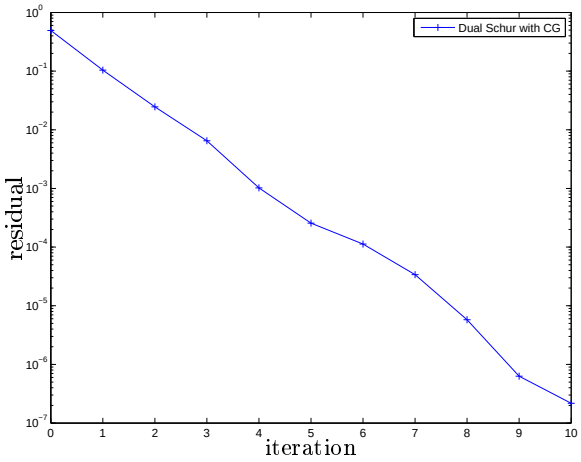


Figure 4.3 – Exemple de la méthode de Schur dual

4.3 FETI et Neumann-Neumann

La méthode FETI (Finite Element Tearing and Interconnect) de Farhat et Roux est à la base une méthode de Schur dual, mais dans sa formulation originale elle s'écrit sous forme variationnelle (voir les premiers exposés à la conférence internationale sur les méthodes de décomposition de domaines `http://www.ddm.org/conferences.html`). La solution du problème $-\Delta u = f$ dans Ω, nulle au bord de Ω, réalise le minimum de la fonctionnelle quadratique

$$J(v) = \frac{1}{2}\int_\Omega |\nabla v|^2\,dx - \int_\Omega fv\,dx$$

sur l'espace V des fonctions, nulles au bord de Ω, dont le gradient est de carré intégrable sur Ω. Décomposons maintenant J sur les sous-domaines Ω_i sans recouvrement,

$$J_1(v) = \frac{1}{2}\int_{\Omega_1} |\nabla v|^2\,dx - \int_{\Omega_1} fv\,dx, \quad J_2(v) = \frac{1}{2}\int_{\Omega_2} |\nabla v|^2\,dx - \int_{\Omega_2} fv\,dx, \quad J(v) = J_1(v) + J_2(v),$$

et minimisons sur l'espace des couples de fonctions (v_1, v_2) telles que le gradient de v_i est de carré intégrable sur Ω_i, et $v_1 = v_2$ sur l'interface commun Γ. C'est un problème de minimisation sous contraintes égalités, que l'on peut écrire au moyen du lagrangien

$$\mathcal{L}(v_1, v_2, h) = J_1(v_1) + J_2(v_2) + \int_\Gamma h(v_2 - v_1)\,ds.$$

Le minimum de J est atteint au point (u_1, u_2, g) où toutes les dérivées partielles du lagrangien sont nulles :

$$\partial_{v_i}\mathcal{L}(u_1, u_2, g) = 0,\ i = 1, 2 \text{ et } \partial_h\mathcal{L}(u_1, u_2, g) = 0.$$

Les dérivées de ces fonctions quadratiques ou linéaires sont simples à calculer, par exemple pour la dérivée par rapport à la variable v_1 en développant $\mathcal{L}(u_1 + v_1, u_2, g) - \mathcal{L}(u_1, u_2, g)$ et en négligeant le terme $J_1(v_1)$. On obtient

$$\partial_{v_i}\mathcal{L}(u_1, u_2, g)\cdot v_i = \int_{\Omega_i} \nabla u_i \cdot \nabla v_i\,dx - \int_{\Omega_i} fv_i\,dx,\ i = 1, 2 \text{ et } \partial_g\mathcal{L}(u_1, u_2, g)\cdot h = \int_\Gamma h(u_2 - u_1)\,ds\ .$$

La nullité des dérivées peut donc se traduire par

$$\begin{aligned}
&\forall v_1 \in V_1,\ \int_{\Omega_1} \nabla u_1 \nabla v_1\,dx - \int_{\Omega_1} fv_1\,dx - \int_\Gamma gv_1\,ds = 0,\\
&\forall v_2 \in V_2,\ \int_{\Omega_2} \nabla u_2 \nabla v_2\,dx - \int_{\Omega_2} fv_2\,dx - \int_\Gamma gv_2\,ds = 0,\\
&\forall h \text{ défini sur } \Gamma,\ \int_\Gamma h(u_2 - u_1)\,ds = 0.
\end{aligned} \tag{4.23}$$

Les deux premières équations du système (4.23) constituent les formulations variationnelles en dimension 2 des équations du système (4.12) avec donnée de Neumann g sur l'interface. La troisième équation est la forme faible de l'équation de continuité $u_1 = u_2$ sur Γ. La résolution des deux premières équations et l'insertion dans la troisième donne la formulation variationnelle de la méthode de Schur dual (4.14). Elle est souvent résolue par une méthode d'éléments finis, ce qui mène à un système discret similaire à (4.22).

La méthode FETI contient deux ingrédients supplémentaires. Nous avons déjà évoqué la grille grossière naturelle à la remarque 4.1. Nous y reviendrons au chapitre 5. Le deuxième ingrédient est lié au préconditionnement. Nous avons vu que les méthodes de Schur primal et dual ont un conditionnement

similaire. Mais il existe un lien beaucoup plus important entre ces deux méthodes : regardons leurs symboles $\hat{\mathcal{S}}_P$ et $\hat{\mathcal{S}}_D$. Leur produit est égal à

$$\hat{\mathcal{S}}_P\,\hat{\mathcal{S}}_D = (\coth(\sqrt{\eta + k^2\pi^2}\,\alpha) + \coth(\sqrt{\eta + k^2\pi^2}\,(1-\alpha)))(\mathrm{th}(\sqrt{\eta + k^2\pi^2}\,\alpha) + \mathrm{th}(\sqrt{\eta + k^2\pi^2}\,(1-\alpha)))$$

qui est une fonction décroissante comprise entre 4 et $(\coth(\sqrt{\eta}\,\alpha) + \coth(\sqrt{\eta}\,(1-\alpha)))(\mathrm{th}(\sqrt{\eta}\,\alpha) + \mathrm{th}(\sqrt{\eta}\,(1-\alpha)))$. Le conditionnement de $\mathcal{S}_P\mathcal{S}_D$ *est donc indépendant de* h : la méthode de Schur primal est un préconditionneur idéal pour la méthode de Schur dual (et réciproquement), obtenant ainsi un conditionnement indépendant du maillage. Plus précisément, au lieu de résoudre par une méthode itérative le système de Schur dual (4.14) avec un conditionnement $O(\frac{1}{h})$, il est beaucoup plus avantageux de résoudre le système préconditionné

$$(\mathcal{S}_1^{\mathcal{DN}} - \mathcal{S}_2^{\mathcal{DN}})(\mathcal{S}_1^{\mathcal{ND}}(g,0,0) - \mathcal{S}_2^{\mathcal{ND}}(g,0,0),0,0) = (\mathcal{S}_1^{\mathcal{DN}} - \mathcal{S}_2^{\mathcal{DN}})(-\mathcal{S}_1^{\mathcal{ND}}(0,g_g,f) + \mathcal{S}_2^{\mathcal{ND}}(0,g_d,f),0,0), \tag{4.24}$$

dont le conditionnement est $O(1)$, par la méthode de Krylov. Joint à la grille grossière naturelle, c'est la méthode FETI.

On peut évidemment inverser les deux processus, c'est-à-dire préconditionner Schur primal par Schur dual, ce qui a été présenté dans la littérature sous le nom de méthode de Neumann-Neumann, voir [3]. Elle peut aussi être implémentée avec une grille grossière naturelle, et prend alors le nom de *Balancing Neumann-Neumann*.

Nous représentons sur la figure 4.4 les spectres de la matrice A de départ, des méthodes de Schur primal et dual, de la méthode de Schur primal préconditionnée par la méthode de Schur dual et vice versa, en fonction du pas du maillage. Dans cette représentation, les spectres sont tous réels, et nous utilisons l'axe y uniquement pour ne pas dessiner les spectres les uns sur les autres. [1]

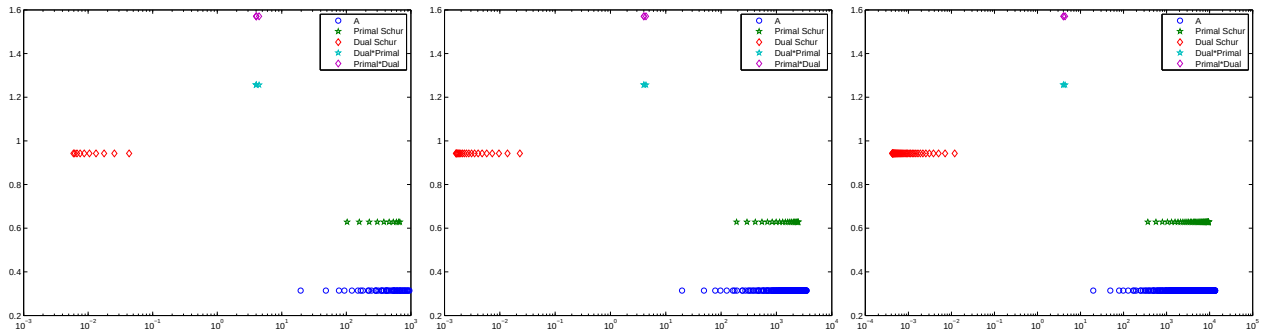


Figure 4.4 – Spectres de A, Schur primal, Schur dual, Schur primal préconditionné par Schur dual and vice versa en fonction du pas du maillage. Les spectres sont réels, l'axe y est seulement utilisé pour les distinguer

Il est intéressant de voir comment le spectre de la méthode de Schur primal se décale vers l'infini, comme le spectre de la matrice de départ, et en contraste le spectre de la méthode de Schur dual s'approche de zéro. Les deux méthodes ont un conditionnement comparable, et nettement meilleur que celui de la matrice de départ, comme le montrent les trois premières colonnes du tableau ci-dessous, pour diverses valeurs du nombre de points de grille J.

1. Pour ces simulations, nous avons choisi de décaler légèrement l'interface du milieu du segment, car si l'interface est au milieu du domaine, ce qui donne une décomposition géométrique symétrique, la méthode de Schur primal est exactement l'inverse de la méthode de Schur dual, et ainsi le spectre de Schur primal préconditionné par Schur dual est réduit au point 1 (et vice versa).

J	A	Schur Primal	Schur Dual	Dual-Primal	Primal-Dual
10	48.37	6.55	7.28	1.11	1.11
20	178.06	13.04	14.31	1.10	1.10
40	680.62	25.91	28.26	1.09	1.09

Tableau 4.1 – Conditionnement des matrices des systèmes linéaires

Par contre pour les méthodes préconditionnées, le spectre reste concentré et le conditionnement est proche de 1, indépendamment du pas du maillage, ce qui montre que la méthode de Schur primal est un préconditionneur idéal pour la méthode de Schur dual, et vice-versa. Cette propriété peut être démontrée pour des décompositions beaucoup plus générales que les deux sous-domaines utilisés ici, voir par exemple [43].

4.4 Méthodes de Dirichlet-Neumann et Neumann-Dirichlet

Au cours de l'histoire des méthodes de décomposition de domaines, le système (4.8) a été également résolu par des méthodes itératives stationnaires. Dans l'algorithme de Dirichlet-Neumann, une suite (g^n, u_1^n, u_2^n) est définie par l'algorithme alterné, dont une étape est donnée par

$$\begin{aligned} \eta\, u_1^{n+1} - \Delta {u_1}^{n+1} &= f \text{ dans } \Omega_1, & \eta\, u_2^{n+1} - \Delta {u_2}^{n+1} &= f \text{ dans } \Omega_2, \\ u_1^{n+1}(0,\cdot) &= g_g, & u_2^{n+1}(1,\cdot) &= g_d, \\ u_1^{n+1}(\alpha,\cdot) &= g^n, & \frac{du_2^{n+1}}{dx}(\alpha,\cdot) &= \frac{du_1^{n+1}}{dx}(\alpha,\cdot). \end{aligned}$$

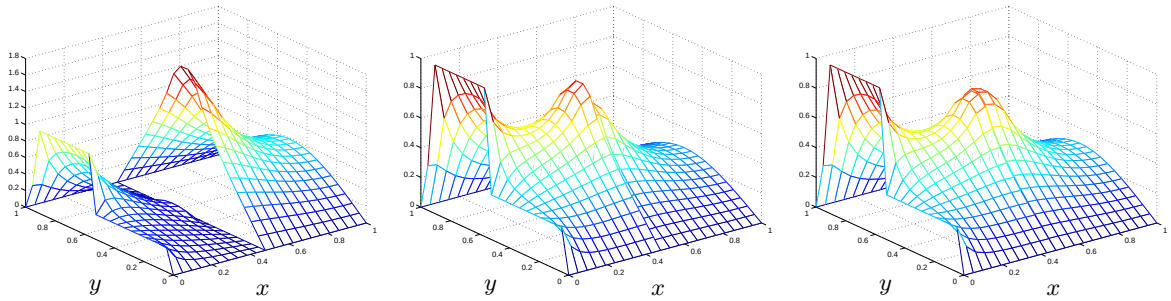
$$g^{n+1} = \theta u_2^{n+1}(\alpha,\cdot) + (1-\theta) g^n.$$

Voici une implémentation en Matlab de la méthode de Dirichlet-Neumann en dimension 2, sur l'exemple de la figure 3.18 :

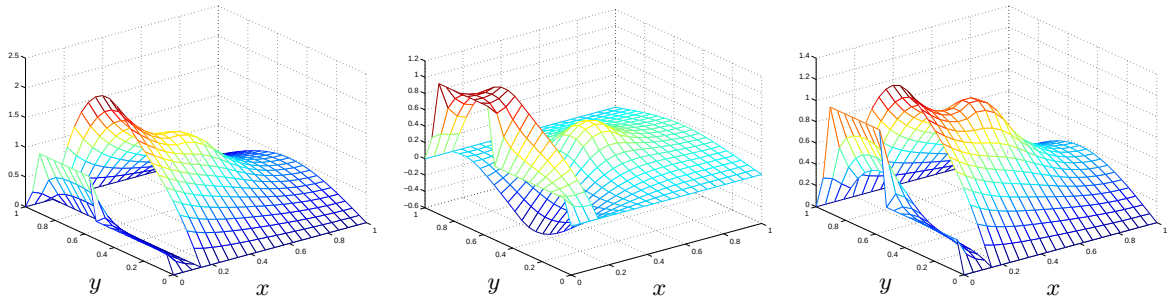
```
eta=0; J=20;                          % number of interior mesh points
x=0:1/(J+1):1; y=x;                   % finite difference mesh, including boundary
f=zeros(J,J+2);                       % source term, include right boundary
f([y>0.4 & y<0.6],[x>0.4 & x<0.6])=50;
gg=0.3*ones(J,1); gg(y>0.5 & y<0.9)=1;
gd=zeros(J,1);

a=10;                                 % decomposition
f1=f(:,2:a);
f2=f(:,a+1:end);
g=zeros(J,1);                         % zero initial guess
h=1/(J+1);
x1=0:h:a*h;                           % finite difference meshes
x2=a*h:h:1;
y=0:h:1;
z1=zeros(1,a+1);z2=zeros(1,J-a+2); % for plotting purposes
th=0.5;                               % relaxation parameter
e=ones(J,1);                          % construct normal derivative
pe=1e12;
Na=[speye(J) -spdiags([-e (eta*h^2+4)*e -e]/2,[-1 0 1],J,J)]/h;
ue=Solve2dR(f,eta,0,J+1,gg*pe,gd,pe,0);
for i=1:20
  u1=Solve2d(f1,eta,0,a,gg,g);
  ta=Na*[u1(:,end-1);u1(:,end)]+f2(:,1)*h/2;
  u2=Solve2dR(f2,eta,a,J+1,ta,gd,0,0);
  g=th*u2(:,1)+(1-th)*g;
```

```
  mesh(x1,y,[z1;u1;z1]); hold on;mesh(x2,y,[z2;u2;z2]);hold off
  xlabel('x'); ylabel('y'); zlabel('Dirichlet Neumann iterates');
  pause
end
```

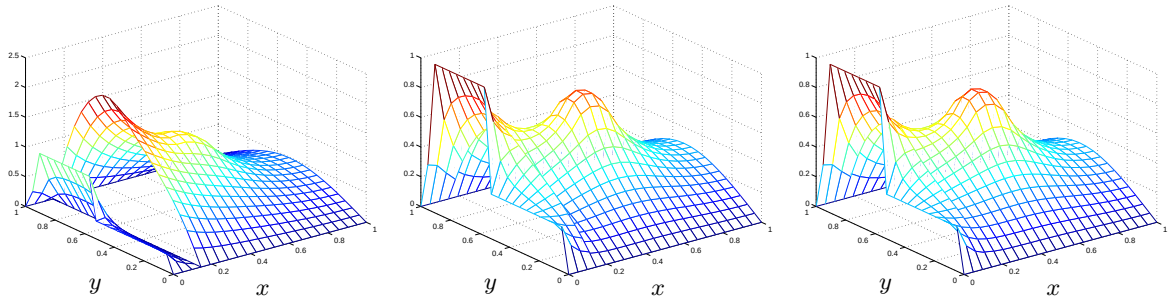
Nous montrons sur la figure 4.5 que le choix du *paramètre de relaxation* θ est crucial, et le meilleur choix dépend malheureusement de la position de l'interface. Si les sous-domaines et le problème sont symétriques, le choix $\theta = \frac{1}{2}$ est optimal, comme on peut le voir sur la première ligne de la figure 4.5. Par contre, lorsque l'interface est plus à gauche en $a = 3$, la méthode converge très mal pour ce choix de θ et peut même diverger (deuxième ligne). Il faut choisir θ plus petit pour rétablir la convergence optimale (troisième ligne).



Interface en $\alpha = 0.5$, $\theta = 0.5$



Interface en $\alpha = 0.15$, $\theta = 0.5$



Interface en $\alpha = 0.15$, $\theta = 0.3$

Figure 4.5 – Méthode de Dirichlet-Neumann : représentation des trois premières itérations de gauche à droite, pour trois couples de valeurs (α, θ)

En termes d'opérateurs de Dirichlet-Neumann et Neumann-Dirichlet, l'algorithme s'écrit

$$g^{n+1} = g^n + \theta \left(\mathcal{S}_2^{\mathcal{ND}}(\mathcal{S}_1^{\mathcal{DN}}(g^n, g_g, f), g_d, f) - g^n \right).$$

C'est un algorithme de Richardson de paramètre θ pour la résolution du système linéaire d'inconnue g,

$$\mathcal{S}_2^{\mathcal{ND}}(\mathcal{S}_1^{\mathcal{DN}}(g, g_g, f), g_d, f) = g.$$

Nous utilisons maintenant le fait que $\mathcal{S}_2^{\mathcal{ND}}(\mathcal{S}_2^{\mathcal{DN}}(g,g_d,f),g_d,f)=g$ pour toutes fonctions f, g_d et g, pour écrire le système linéaire sous la forme

$$\mathcal{S}_2^{\mathcal{ND}}(\mathcal{S}_1^{\mathcal{DN}}(g,g_g,f)-\mathcal{S}_2^{\mathcal{DN}}(g,g_d,f),g_d,f)=0. \tag{4.25}$$

C'est donc un algorithme de Richardson pour la résolution du problème de Schur primal (4.10) préconditionné par $\mathcal{S}_2^{\mathcal{ND}}$.

Pour l'algorithme de Neumann-Dirichlet, une suite (g^n,u_1^n,u_2^n) est définie par

$$\begin{array}{rclcrcl}
\eta\, u_1^{n+1}-\Delta {u_1}^{n+1} &=& f \text{ dans } \Omega_1, & \qquad & \eta\, u_2^{n+1}-\Delta {u_2}^{n+1} &=& f \text{ dans } \Omega_2,\\
u_1^{n+1}(0,\cdot) &=& g_g, & & u_2^{n+1}(1,\cdot) &=& g_d,\\
\dfrac{du_1^{n+1}}{dx}(\alpha,\cdot) &=& g^n, & & u_2^{n+1}(\alpha,\cdot) &=& u_1^{n+1}(\alpha,\cdot).
\end{array}$$

$$g^{n+1}=\theta\frac{du_2^{n+1}}{dx}(\alpha,\cdot)+(1-\theta)g^n.$$

En termes d'opérateurs de Dirichlet-Neumann et Neumann-Dirichlet, cet algorithme s'écrit

$$g^{n+1}=g^n+\theta\left(\mathcal{S}_2^{\mathcal{DN}}(\mathcal{S}_1^{\mathcal{ND}}(g^n,g_g,f),g_d,f)-g^n\right)$$

ce qui est de nouveau un algorithme de Richardson pour la résolution du système linéaire pour g :

$$\mathcal{S}_2^{\mathcal{DN}}(\mathcal{S}_1^{\mathcal{ND}}(g,g_g,f),g_d,f)=g,$$

ou encore de

$$\mathcal{S}_2^{\mathcal{DN}}(\mathcal{S}_1^{\mathcal{ND}}(g,g_g,f)-\mathcal{S}_2^{\mathcal{ND}}(g,g_d,f),g_d,f)=0. \tag{4.26}$$

C'est maintenant un algorithme de Richardson pour la résolution du problème de Schur dual (4.14) préconditionné par $\mathcal{S}_2^{\mathcal{DN}}$.

Une décomposition en modes de Fourier comme pour l'analyse des méthodes de Schur montre que la condition des deux systèmes (4.25) et (4.26), ne dépend pas du pas du maillage h.

5 Préconditionneur grille grossière

Pour exposer les différentes méthodes dans un cadre simple, nous avons considéré dans les chapitres précédents 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 ce chapitre.

5.1 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 5.1. Nous supposerons que seuls deux sous-domaines successifs se touchent, c'est-à-dire que pour tout i, $\beta_{i-1}<\alpha_{i+1}$.

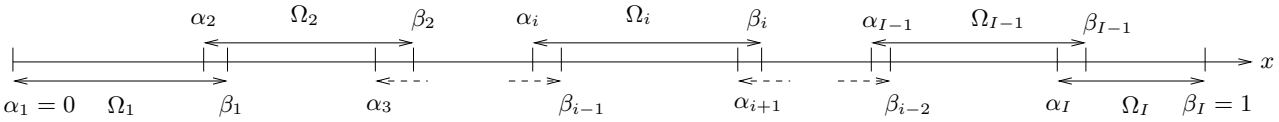


Figure 5.1 – Décomposition en sous-domaines

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

$$\begin{aligned} \partial_{xx} u_i^n &= 0 \quad \text{dans } \Omega_i, \\ u_i^n(\alpha_i) = u_{i-1}^{n-1}(\alpha_i), &\quad u_i^n(\beta_i) = u_{i+1}^{n-1}(\beta_i), \end{aligned} \tag{5.1}$$

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 5.2, où sont tracées les courbes de convergence pour 2, 4, 8 et 16 sous-domaines. Les tests seront les mêmes tout au long de ce chapitre :

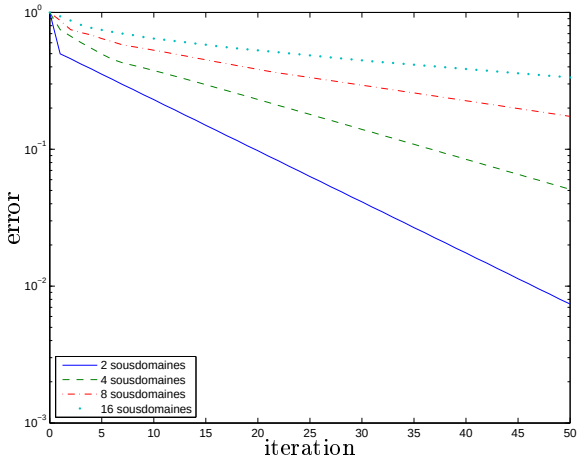
Cas test

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.

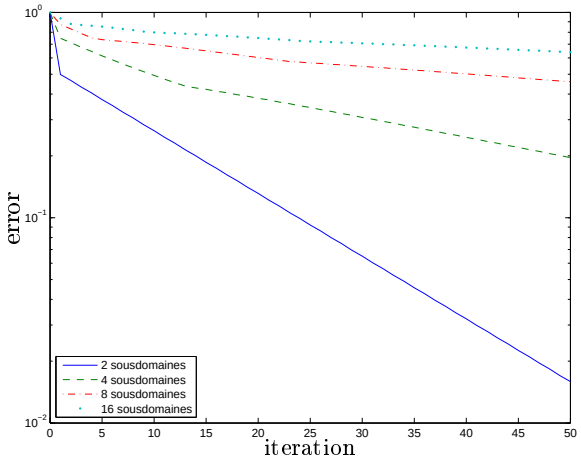
Sur les deux figures du bas, nous étudions la scalabilité faible : lorsque le nombre de sous-domaines augmente $(2, 4, 8, 16)$, la taille du problème global augmente en proportion $(J = 511, 1023, 2047, 4095)$.

Sur les figures de gauche le recouvrement entre les sous-domaines est constant égal à $h_0 = 20/(J_0+1)$.

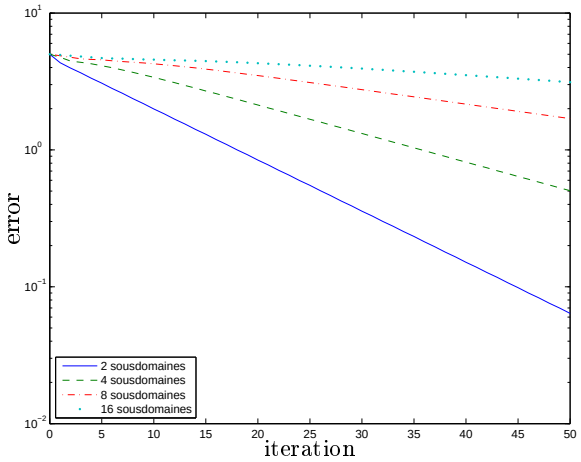
Sur les figures de droite le recouvrement diminue en proportion du nombre de sous-domaines.



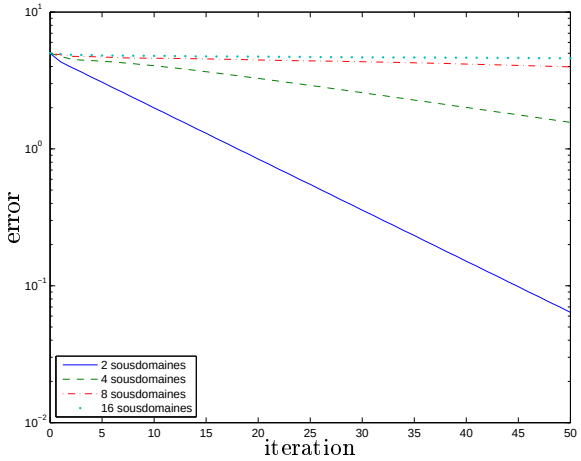
(a) Etude de scalabilité forte : δ de taille constante



(b) Etude de scalabilité forte : δ diminue avec H



(c) Etude de scalabilité faible : δ de taille constante



(d) Etude de scalabilité faible : δ diminue avec H

Figure 5.2 – Résolution du système (2.13) avec un nombre croissant 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.

5.2 Explications intuitives et mathématiques

Pourquoi la méthode ralentit-elle lorsque l'on ajoute des sous-domaines ? L'explication est visible sur la figure 5.3, 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.

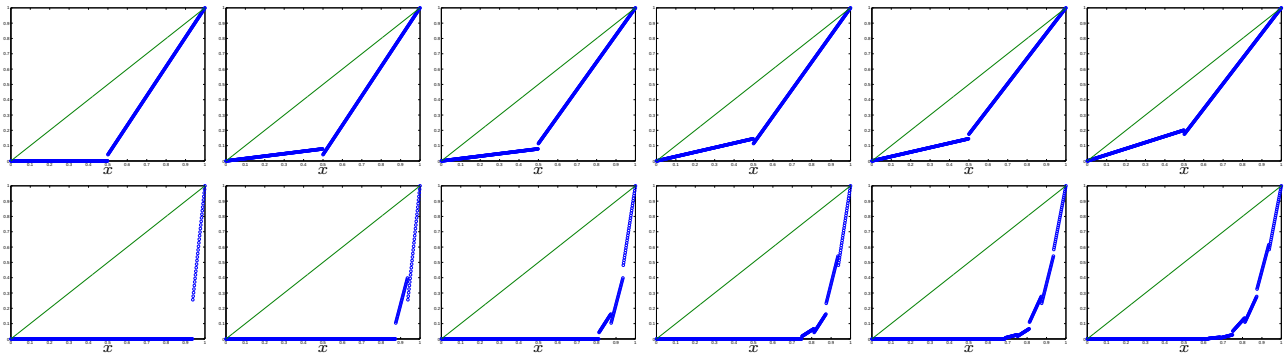


Figure 5.3 – Premières itérations d'une méthode de Schwarz parallèle : deux sous-domaines sur la première ligne, 16 sous-domaines sur la deuxième

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

$$\begin{pmatrix} e_1^n(\alpha_2) \\ e_2^n(\beta_1) \\ e_2^n(\alpha_3) \\ e_3^n(\beta_2) \\ e_3^n(\alpha_4) \\ \vdots \\ e_{I-1}^n(\beta_{I-2}) \\ e_{I-1}^n(\alpha_I) \\ e_I^n(\beta_{I-1}) \end{pmatrix} = \begin{pmatrix} 0 & b_1^+ & 0 & & & & & & \\ a_2^- & 0 & 0 & b_2^- & & & & & \\ a_2^+ & 0 & 0 & b_2^+ & 0 & & & & \\ & 0 & a_3^- & 0 & 0 & b_3^- & & & \\ & & a_3^+ & 0 & 0 & b_3^+ & 0 & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & & 0 & a_{I-1}^- & 0 & 0 & b_{I-1}^- \\ & & & & & a_{I-1}^+ & 0 & 0 & b_{I-1}^+ \\ & & & & & & 0 & a_I^- & 0 \end{pmatrix} \begin{pmatrix} e_1^{n-1}(\alpha_2) \\ e_2^{n-1}(\beta_1) \\ e_2^{n-1}(\alpha_3) \\ e_3^{n-1}(\beta_2) \\ e_3^{n-1}(\alpha_4) \\ \vdots \\ e_{I-1}^{n-1}(\beta_{I-2}) \\ e_{I-1}^{n-1}(\alpha_I) \\ e_I^{n-1}(\beta_{I-1}) \end{pmatrix}. \qquad (5.2)$$

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

anti-diagonal par blocs

$$\left(\begin{array}{c} e_1^n(\alpha_2) \\ e_3^n(\beta_2) \\ e_3^n(\alpha_4) \\ \vdots \\ e_{I-1}^n(\beta_{I-2}) \\ e_{I-1}^n(\alpha_I) \\ \hline e_2^n(\beta_1) \\ e_2^n(\alpha_3) \\ e_4^n(\beta_3) \\ e_4^n(\alpha_5) \\ \vdots \\ e_I^n(\beta_{I-1}) \end{array}\right) = \left(\begin{array}{cccccc|cccccc} & & & & & & b_1^+ & & & & & \\ & & & & & & & a_3^- & b_3^- & & & \\ & & & & & & & a_3^+ & b_3^+ & & & \\ & & & & & & & & & \ddots & & \\ & & & & & & & & & & a_{I-1}^- & b_{I-1}^- \\ & & & & & & & & & & a_{I-1}^+ & b_{I-1}^+ \\ \hline a_2^- & b_2^- & & & & & & & & & & \\ a_2^+ & b_2^+ & & & & & & & & & & \\ & & a_4^- & b_4^- & & & & & & & & \\ & & a_4^+ & b_4^+ & & & & & & & & \\ & & & & \ddots & & & & & & & \\ & & & & & a_I^- & & & & & & \end{array}\right) \left(\begin{array}{c} e_1^{n-1}(\alpha_2) \\ e_3^{n-1}(\beta_2) \\ e_3^{n-1}(\alpha_4) \\ \vdots \\ e_{I-1}^{n-1}(\beta_{I-2}) \\ e_{I-1}^{n-1}(\alpha_I) \\ \hline e_2^{n-1}(\beta_1) \\ e_2^{n-1}(\alpha_3) \\ e_4^{n-1}(\beta_3) \\ e_4^{n-1}(\alpha_5) \\ \vdots \\ e_I^{n-1}(\beta_{I-1}) \end{array}\right), \tag{5.3}$$

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

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

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

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

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 5.1 *Pour une décomposition régulière, la norme euclidienne de A_H^2 est bornée par*

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

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

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

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

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

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

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

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

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

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

$$\|A_H^2\|_2 \leq (a+b)^2 - 4ab\sin^2\big(\frac{\pi}{2I}\big) = 1 - 4b(1-b)\sin^2\big(\frac{\pi}{2I}\big) \ \sim 1 - b(1-b)\big(\frac{\pi}{I}\big)^2.$$

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

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

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

$$\begin{aligned} \|\lambda\boldsymbol{x} + b^2F\boldsymbol{x}\|_2^2 &= \|(\lambda x_1, \lambda x_2 + b^2x_4, \lambda x_3 + b^2x_1, \dots)\|_2^2 \\ &= \lambda^2\sum_{j=1}^{I-1} x_j^2 + b^4\sum_{j\neq 2, I-1} x_j^2 + 2\lambda\, b^2\sum_{j=1}^{I-3} x_jx_{j+2}. \end{aligned}$$

Grâce à l'égalité

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

nous pouvons calculer le premier terme

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

Calculons maintenant le troisième terme

$$\begin{aligned} \sum_{j=1}^{I-3} x_jx_{j+2} &= \sum_{j=1}^{I-3}\sin\,\frac{j\pi}{I}\sin\,\frac{(j+2)\pi}{I} \\ &= \frac{1}{2}\sum_{j=1}^{I-3}(\cos\,\frac{2\pi}{I} - \cos\,\frac{2(j+1)\pi}{I}) \\ &= \frac{1}{2}\big((I-3)\cos\,\frac{2\pi}{I} - \sum_{j=2}^{I-2}\cos\,\frac{2j\pi}{I}\big)) \\ &= \frac{1}{2}\big((I-3)\cos\,\frac{2\pi}{I} - (-1-\cos\,\frac{2\pi}{I} - \cos\,\frac{2(I-1)\pi}{I})\big) \\ &= \frac{1}{2}\big((I-3)\cos\,\frac{2\pi}{I} - (-1-2\cos\,\frac{2\pi}{I}\big) \\ &= \frac{1}{2}\big((I-1)\cos\,\frac{2\pi}{I} + 1\big). \end{aligned}$$

Nous pouvons maintenant calculer

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

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

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

et

$$\frac{\|\lambda \boldsymbol{x} + b^2 F\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|_2^2} = M_1 + \frac{2}{I}M_2.$$

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 plus haut,

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

puis

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

et donc

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

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

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

et de noter que les deux quantités contenant $\boldsymbol{x}$ sont égales. En effet

$$\begin{array}{rl} \|\lambda \boldsymbol{x} + b^2 F'\boldsymbol{x}\|_2^2 & = \|(\lambda x_1 + b^2 x_3, \lambda x_2, \lambda x_3 + b^2 x_5, \dots)\|_2^2 \\ & = \lambda^2 \displaystyle\sum_{j=1}^{I-1} x_j^2 + b^4 \sum_{j \neq 1, I-2} x_j^2 + 2\lambda\, b^2 \sum_{j=1}^{I-3} x_j x_{j+2}\;, \end{array}$$

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

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

■

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.

5.3 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 multi-grille. On parle alors de méthode à deux niveaux, l'étape n est décomposée comme suit :

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

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 $R_{ij} = E_{ji}/\sum_k E_{ki}$. 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 grid 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
  R=E'; R=diag(1./sum(R'))*R;              % restriction
end;
Ac=R*A*E;                                  % Galerkin coarse matrix
```

Nous pouvons maintenant utiliser ces composantes dans le programme Matlab suivant, qui réalise en dimension un 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 5.2 et 5.3, et avec lequel nous testons maintenant l'effet de l'ajout d'une grille grossière.

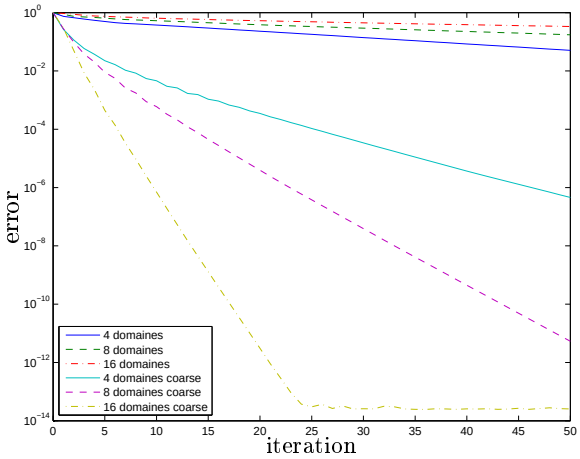
```
function [u,err]=SolveDD(f,eta,a,b,gg,gd,Ii,d,u0,CC,N)
% SOLVEDD solves 1d model problem with domain decomposition
%   u=SolveDDPrivate(f,eta,a,b,gg,gd,Ii,d,u0,CC,N); solves
%   (eta-dxx)u=f on the 1d domain (a,b) with Dirichlet conditions u=gg
%   and u=gd on the equidistant grid defined by the length of the rhs
```

```
%   f using a Schwarz algorithm. Subdomains are defined by the indices
%   Ii(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,Ii(2:end-1)-d]; bi=[Ii(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(Ii); [E,R,Ac]=CoarseOperators(Im,A);
elseif CC==2
  Im=CoarseOpt(Ii); [E,R,Ac]=CoarseOperators(Im,A);
end;

for n=1:N                                   % Schwarz iterations
  uo=u;
  udd=u;
  r=ft-A*uo(2:end-1);
  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;
```

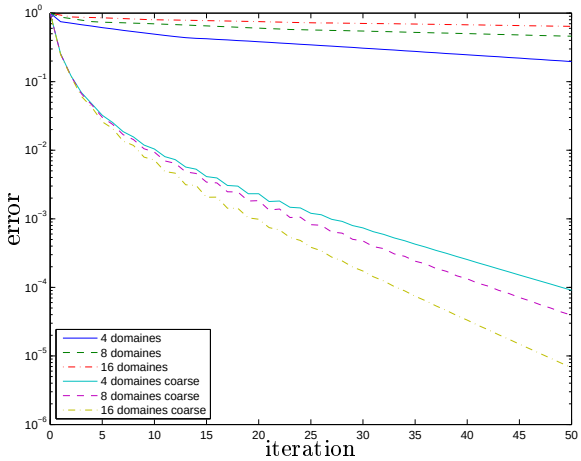
Nous utilisons maintenant ce code pour réaliser les tests décrits page 52. Nous choisissons d'abord les nœuds de la grille grossière de façon classique (*cf* [43]), c'est-à-dire aux centres des sous-domaines :

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

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

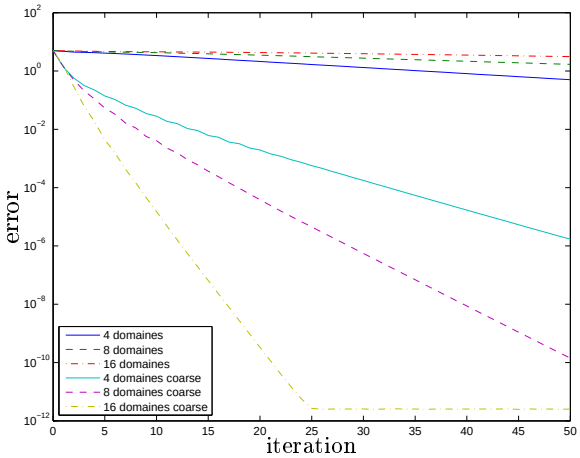
Les résultats sont présentés sur la figure 5.4.



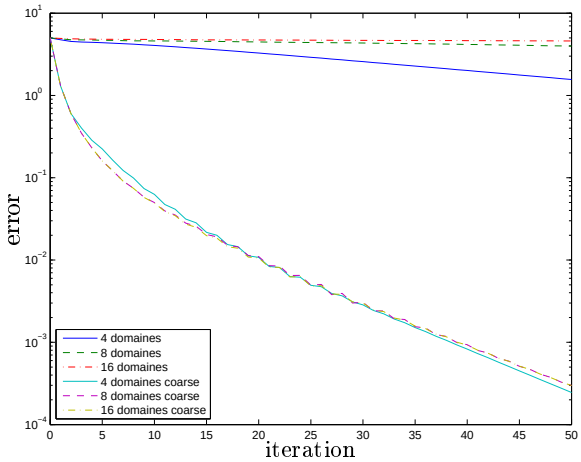
(a) Etude de scalabilité forte : δ de taille constante



(b) Etude de scalabilité forte : δ diminue avec H



(c) Etude de scalabilité faible : δ de taille constante



(d) Etude de scalabilité faible : δ diminue avec H

Figure 5.4 – Résolution du système (2.13) avec un nombre croissant de sous-domaines, sans et avec grille grossière (coarse) classique

Nous voyons que dans le cas d'un recouvrement de taille constante (figures de gauche), 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 (figures de droite), la convergence est maintenant 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 5.2 *Le conditionnement du système préconditionné par la méthode de Schwarz additif avec grille grossière est majoré par*

$$C(1 + \frac{H}{\delta}) \tag{5.4}$$

où C est une constante indépendante de H et δ.

Démonstration Voir [43]. ■

Ce résultat théorique explique les comportements décrits sur la figure 5.4. En effet, si dans l'estimation (5.4), 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 5.5 les premières itérations de la méthode de Schwarz avec grille grossière pour 16 sous-domaines.

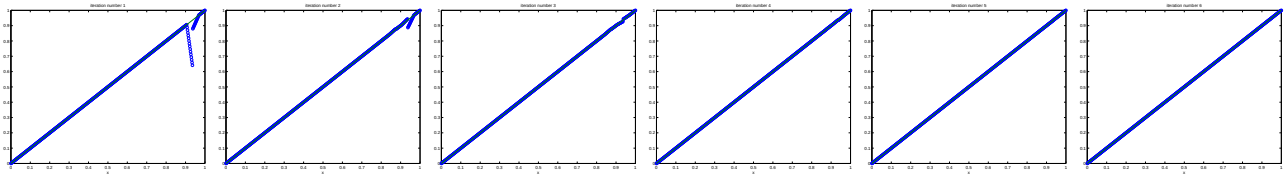
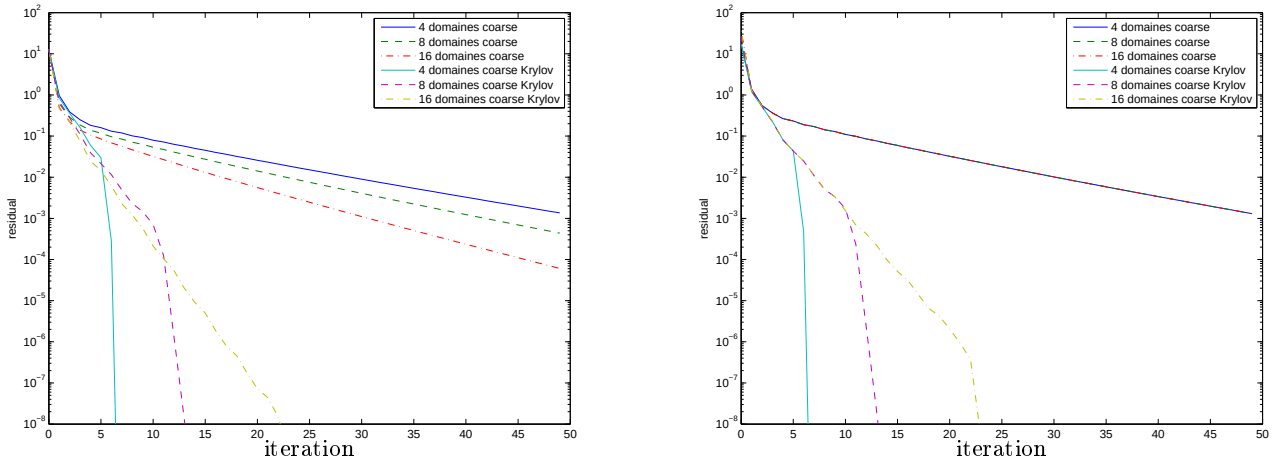


Figure 5.5 – Premières itérations d'une méthode de Schwarz avec 16 sous-domaines et une correction grille grossière : la convergence est maintenant plus rapide

En comparaison avec la figure 5.3, 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 sur la dernière interface, comme on le voit sur la première itération de la figure 5.3. 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 suivante, nous comparons l'effet de l'ajout de la méthode de Krylov dans le cas où δ diminue avec H.



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

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

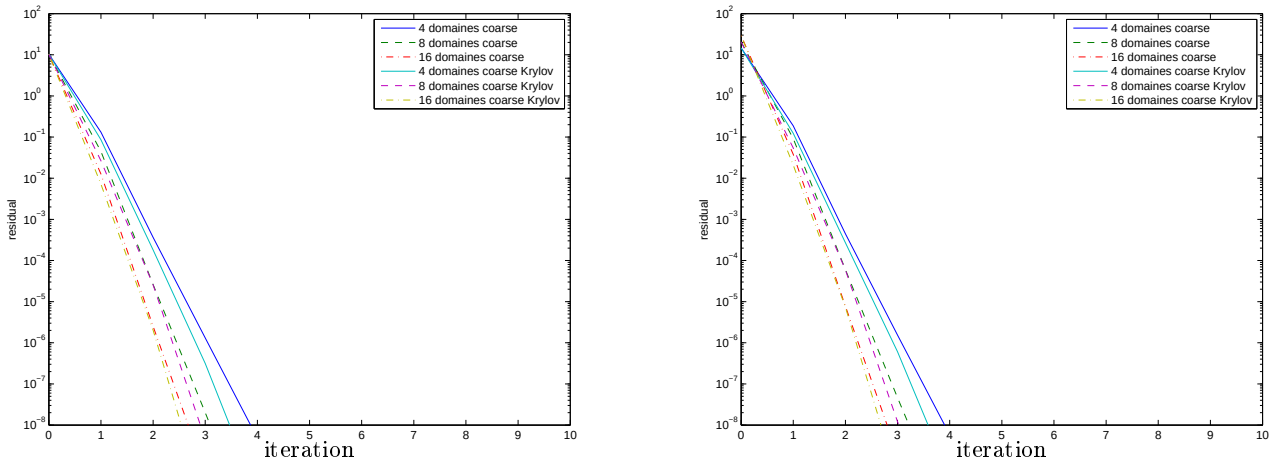
Figure 5.6 – Résolution du système (2.13) avec un nombre croissant de sous-domaines, avec grille grossière classique, avec ou sans algorithme de Krylov

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 page 23.

Revenons maintenant à la question de régularité soulevée par l'étude de la figure 5.3. 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 page 21, 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 5.5. 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-dessous définit cette nouvelle grille grossière, et la figure suivante illustre les propriétés de convergence de l'algorithme avec cette grille grossière.

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

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



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

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

Figure 5.7 – Résolution du système (2.13) avec un nombre croissant de sous-domaines, avec grille grossière adaptée aux discontinuités, avec ou sans algorithme de Krylov

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 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 3.20.

Remarque 5.1 *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 5.8 une comparaison des deux méthodes présentées.

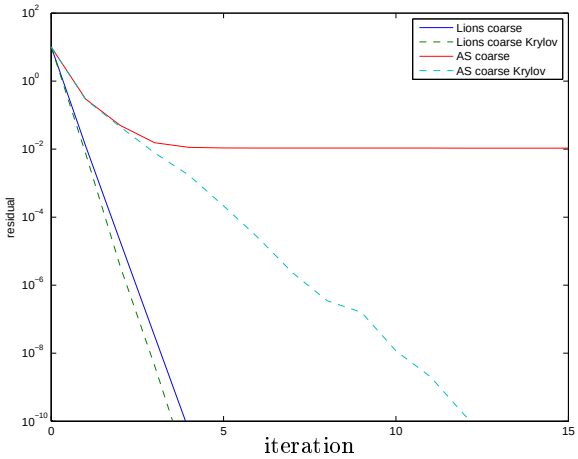


Figure 5.8 – Comparaison de la méthode Schwarz additif avec grille grossière classique avec la méthode de Schwarz parallèle de Lions et grille grossière optimisée, avec ou sans Krylov

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 4.1 : 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 5.2, voir [12, 32] pour Neumann-Neumann et [33] pour FETI.

Théorème 5.3 *Le conditionnement du système FETI ou balancing Neumann-Neumann est majoré par*

$$C(1+\log(\frac{H}{h}))^2 \tag{5.5}$$

où C est une constante indépendante de H et h.

Démonstration Voir [43]. ■

6 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 (cf début de section 2.4). Celle-ci pourra être résolue par une des méthodes de décomposition de domaines 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 les méthodes optimisées développées par la suite, dans [18] par exemple. Ces méthodes font l'objet de la première partie de ce chapitre.

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.

6.1 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*

$$\begin{array}{rcll} \partial_t u - \partial_{xx} u & = & f & \text{dans } \Omega \times [0,T], \\ u(\cdot,0) & = & u_0 & \text{dans } \Omega, \\ u(0,\cdot) & = & g_g, & \\ u(1,\cdot) & = & g_d, & \end{array} \tag{6.1}$$

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 with
%   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-dessous effectue cette simulation. L'évolution de la solution est représentée sur la figure 6.1.

```
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))]);
```

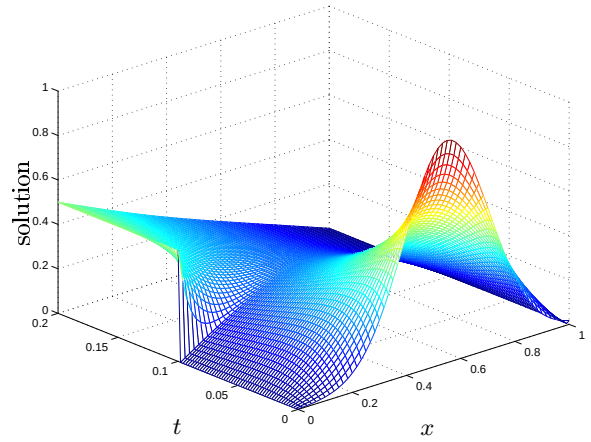


Figure 6.1 – Exemple d'une solution instationnaire de l'equation de la chaleur

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 $n = 1, 2, \ldots$

$$\begin{array}{rclll} \partial_t u_1^n - \partial_{xx} u_1^n & = & f & \text{dans } \Omega_1 \times [0,T], & \partial_t u_2^n - \partial_{xx} u_2^n = f \text{ dans } \Omega_2 \times [0,T], \\ u_1^n(\cdot,0) & = & u_0 & \text{dans } \Omega_1, & u_2^n(\cdot,0) = u_0 \text{ dans } \Omega_2, \\ u_1^n(0,\cdot) & = & g_g, & & u_2^n(1,\cdot) = g_d, \\ u_1^n(\beta,\cdot) & = & u_2^{n-1}(\beta,\cdot) & \text{sur } [0,T], & u_2^n(\alpha,\cdot) = u_1^{n-1}(\alpha,\cdot) \text{ sur } [0,T]. \end{array} \tag{6.2}$$

Les donnée 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 $\Gamma_i \times [0,T]$. 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 6.1 *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*

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

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 les mêmes conditions de transmission. Définissons un algorithme stationnaire, pour des fonctions $\tilde{e}_i^n$ de l'espace uniquement

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

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

$$0 \leq \tilde{e}_i^n \leq \tilde{e}_i^n{}_{|\Gamma_i}.$$

Nous allons montrer que pour tout (x,t) dans $\Omega_i \times [0,T]$, $|e_i^n(x,t)| \leq \tilde{e}_i^n(x)$. 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

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

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

$$\begin{aligned} \tilde{e}_1^n(\beta) = \sup_{t\in[0,T]} |e_1^n(\beta,t)| = \sup_{t\in[0,T]} |e_2^{n-1}(\beta,t)| \leq \tilde{e}_2^{n-1}(\beta) = \frac{1-\beta}{1-\alpha}\, \tilde{e}_2^{n-1}(\alpha), \\ \tilde{e}_2^{n-1}(\alpha) = \sup_{t\in[0,T]} |e_2^{n-1}(\alpha,t)| = \sup_{t\in[0,T]} |e_1^{n-2}(\alpha,t)| \leq \tilde{e}_1^{n-2}(\alpha) = \frac{\alpha}{\beta}\, \tilde{e}_1^{n-2}(\beta). \end{aligned}$$

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

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

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_i^{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 6.2 *Dans le cas de deux demi-droites infinies, $\Omega_1 =]-\infty, \beta]$ et $\Omega_2 = [\alpha, +\infty[$ avec $\delta = \beta - \alpha > 0$, les algorithmes de Schwarz relaxation d'onde alterné et parallèle convergent superlinéairement sur un intervalle de temps borné. Par exemple pour l'algorithme parallèle, on a dans le sous-domaine de gauche :*

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

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

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

$$\mathcal{L}f : s \mapsto \hat{f}(s) = \int_0^{+\infty} f(t) e^{-st}\, dt.$$

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

$$\begin{array}{rcll} s\hat{e}_1^n - \partial_{xx}\hat{e}_1^n &=& 0 & \text{dans } \Omega_1, \\ \hat{e}_1^n(\beta,s) &=& \hat{e}_2^{n-1}(\beta,s), & \end{array} \qquad \begin{array}{rcll} s\hat{e}_2^n - \partial_{xx}\hat{e}_2^n &=& 0 & \text{dans } \Omega_2, \\ \hat{e}_2^n(\alpha,s) &=& \hat{e}_1^{n-1}(\alpha,s). & \end{array} \tag{6.3}$$

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

$$\hat{e}_1^{2n}(\alpha,s) = (\rho^D(s))^n\ \hat{e}_1^0(\alpha,s), \quad \hat{e}_2^{2n}(\beta,s) = (\rho^D(s))^n\ \hat{e}_2^0(\beta,s), \tag{6.4}$$

où le facteur de convergence ρ^D est défini par

$$\rho^D(s) = e^{-2\delta\sqrt{s}}. \tag{6.5}$$

Les relations (6.4) 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 $(\rho^D(s))^n$, donnée par

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

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

$$|e_1^{2n}(\alpha,t)| = \left| \int_0^t e_1^0(\alpha,t-\tau) G(\tau)\, d\tau \right| \leq \sup_{t\in[0,T]} |e_1^0(\alpha,t)| \int_0^T G(t)\, dt = \sup_{t\in[0,T]} |e_1^0(\alpha,t)|\, \mathrm{erfc}(\frac{n\delta}{\sqrt{T}}).$$

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

Remarque 6.1 *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 2.2. Pour de grandes valeurs de n, la convergence de l'algorithme de Picard est estimée à l'aide de la formule de Stirling par*

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

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

$$\mathrm{erfc}\,(\frac{n\delta}{\sqrt{T}}) \sim \frac{\sqrt{T}}{\sqrt{\pi} n\delta} e^{-\frac{\delta^2}{T} n^2}.$$

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
a=28;d=4;
u1=[u0(1:a+d+1);gg zeros(M,a+d)];       % zero initial guess
u2=[u0(a+1:end);zeros(M,J-a-1) gd];
udd=[u1(:,1:a) u2];
err(1)=norm(u-udd,'inf');
x1=0:dx:(a+d)*dx; x2=a*dx:dx:1;         % finite difference meshes
for i=1:20
  u1=Heat(0,(a+d)*dx,u0(1:a+d+1),gg,u2(2:end,d+1),T);
  u2=Heat(a*dx,1,u0(a+1:end),u1(2:end,end-d),gd,T);
  udd=[u1(:,1:a) 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 6.2 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 $T = 0.2$.

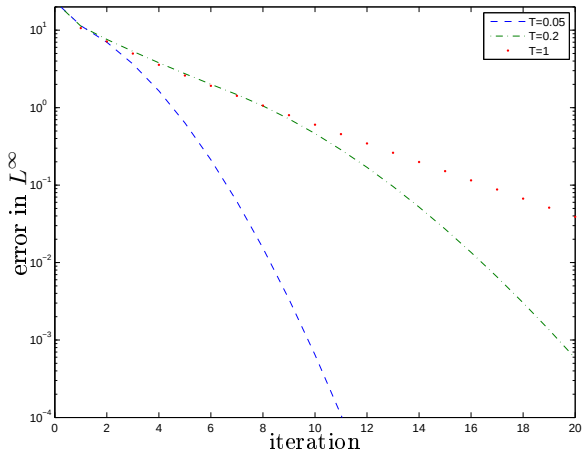


Figure 6.2 – Convergence linéaire et super-linéaire de l'algorithme de Schwarz relaxation d'onde

6.2 Méthodes de Schwarz relaxation d'onde optimisées

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 (6.2) sont alors remplacées par

$$\begin{array}{rcll} \partial_x u_1^n + p u_1^n & = & \partial_x u_2^{n-1} + p u_2^{n-1} & \text{sur } \Gamma_1 \times [0,T], \\ \partial_x u_2^n - p u_2^n & = & \partial_x u_1^{n-1} - p u_1^{n-1} & \text{sur } \Gamma_2 \times [0,T]. \end{array} \tag{6.6}$$

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

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

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 6.2. 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 6.2. Nous obtenons pour les itérées paires par exemple

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

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 deuxième 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, $|\rho^R(s,p)| < 1$.

La suite des valeurs de $\hat{e}_2^{2n}(\beta,s)$ converge donc vers 0. D'autre part, $|\hat{e}_2^{2n}(\beta,s)|$ est borné par $|\hat{e}_2^{0}(\beta,s)|$ 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 le même ouvrage), 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 6.2.

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

$$\begin{aligned}
\int_0^\alpha (\partial_t e_1^n(x,t)) e_1^n(x,t)\,dx - \int_0^\alpha (\partial_{xx} e_1^n(x,t)) e_1^n(x,t)\,dx = 0,\\
\int_\alpha^1 (\partial_t e_2^n(x,t)) e_2^n(x,t)\,dx - \int_\alpha^1 (\partial_{xx} e_2^n(x,t)) e_2^n(x,t)\,dx = 0,
\end{aligned}$$

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

$$\begin{aligned}
\frac{1}{2}\frac{d}{dt}\int_0^\alpha (e_1^n(x,t))^2\,dx + \int_0^\alpha (\partial_x e_1^n(x,t))^2\,dx - (\partial_x e_1^n(\alpha,t)) e_1^n(\alpha,t) = 0,\\
\frac{1}{2}\frac{d}{dt}\int_\alpha^1 (e_2^n(x,t))^2\,dx + \int_\alpha^1 (\partial_x e_2^n(x,t))^2\,dx + (\partial_x e_2^n(\alpha,t)) e_2^n(\alpha,t) = 0.
\end{aligned}$$

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

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

Utilisons maintenant les conditions de transmission (6.6) appliquées aux erreurs pour remplacer le membre de droite :

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

Ajoutons ces deux égalités pour obtenir

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

Nous constatons que les termes frontière 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 :

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

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

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

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

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

■

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 :

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

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'équioscillation inspirées par les résultats de Chebyshev et de la Vallée Poussin [7, 44]. Nous donnons dans la Table 6.1 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 $\Delta t = \mathcal{O}(\Delta x^2) = \mathcal{O}(\delta^2)$, tandis que dans le cas implicite, on choisit d'ordinaire $\Delta t = \mathcal{O}(\Delta x) = \mathcal{O}(\delta)$.

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 $\delta = \mathcal{O}(\Delta t)$	$\sqrt{2}(\omega_{min}\omega_{max})^{\frac{1}{4}}$	$1 - 2\sqrt{2}\left(\frac{\omega_{min}}{\omega_{max}}\right)^{\frac{1}{4}}$
Avec recouvrement $\delta = \mathcal{O}(\sqrt{\Delta t})$	$2(4\omega_{min})^{\frac{1}{3}}\delta^{-\frac{1}{3}}$	$1 - 4\left(\frac{\omega_{min}}{2}\right)^{\frac{1}{6}}\delta^{\frac{1}{3}}$

Tableau 6.1 – Résultats asymptotiques

Voici le solveur Matlab avec des conditions de Robin :

```
function u=HeatR(a,b,u0,gg,gd,T,p1,p2);
% HEATR solves the heat equation with Robin conditions
%   u=HeatR(a,b,u0,gg,gd,T,p1,p2); solves the heat equation
%   du/dt=d^2u/dx^2 on (a,b)x(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-a)/(J-1);
e=ones(J,1);                                % size of A
B=[-1/dx^2*e (1/dt+2/dx^2)*e -1/dx^2*e];
A=spdiags(B,[-1,0,1],J,J);
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-dessous, nous avons établi les courbes de convergence de la figure 6.3. Le paramètre optimisé est obtenu par la formule du tableau 6.1, 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.

```
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
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;
pe=1e5;                                 % simulate Dirichlet conditions
u=HeatR(0,1,u0,pe*gg,pe*gd,T,pe,pe); % compute reference solution

a=28;d=4;
u1=[u0(1:a+d+1);gg zeros(M,a+d)];       % zero initial guess
u2=[u0(a+1:end);zeros(M,J-a-1) gd];
udd=[u1(:,1:a) u2];
err(1)=norm(u-udd,'inf');
x1=0:dx:(a+d)*dx; x2=a*dx:dx:1;         % finite difference meshes
p=sqrt(2)*(pi/(2*T)*pi/dt)^0.25;        % optimized parameter
pe=1e5;                                 % emulate Dirichlet conditions
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,(a+d)*dx,u0(1:a+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(a*dx,1,u0(a+1:end),ta,pe*gd,T,p,pe);
  udd=[u1(:,1:a) u2];
  mesh(x,t,udd); xlabel('x'); ylabel('t');
  zlabel('Optimized Schwarz waveform relaxation iterates');
  pause
  err(i+1)=norm(u-udd,'inf');
end
```

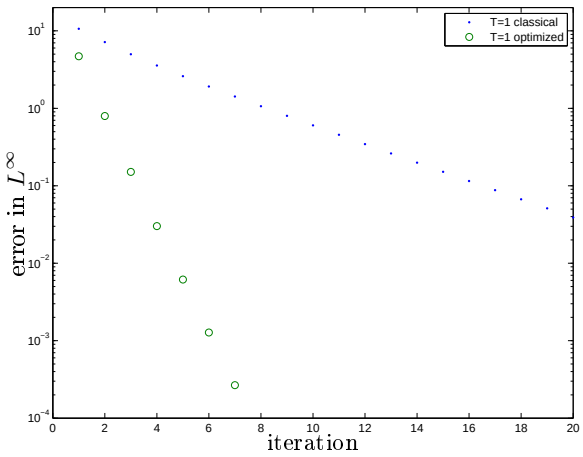


Figure 6.3 – Comparaison des méthodes de Schwarz relaxation d'onde classique et optimisée

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 6.2.

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.

6.3 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 au chapitre 5, 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 $n^o\, i$ 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*, destinée 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,

$$u'' = f(u), \quad u(0) = u^0, \quad u(T) = u^1. \tag{6.7}$$

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

$$u_0'' = f(u_0), \quad u_0(0) = u^0, \quad u_0'(0) = U. \tag{6.8}$$

Pour une fonction f lipschitzienne, pour toute valeur de U, le problème (6.8) a une solution unique, au moins localement (voir le dossier AF 1 220 des Techniques de l'Ingénieur). Notons-la $u_0(t; u^0, U)$. La solution u de (6.7) est la solution u_0 de (6.8) pour la valeur $\bar{U}$ de U définie par $u_0(1, u^0, \bar{U}) = u^1$. 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 (6.7) est bien posé, il se peut que le système dynamique (6.8) 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 $u : [0, T] \to \mathbb{R}^d$:

$$u' = f(u), \quad u(0) = u^0, \quad t \in [0, T]. \tag{6.9}$$

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 $[T_0 = 0, T_1]$, $\cdots, [T_I, T_{I+1} = T]$, 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

$$U_0 = u^0, \quad U_{i+1} = u_i(T_{i+1}; U_i), \quad i = 0, \ldots, I - 1. \tag{6.10}$$

Les $I + 1$ relations (6.10) forment un système non linéaire, que nous écrivons sous forme compacte $\mathcal{F}(\boldsymbol{U}) = 0$, avec

$$\boldsymbol{U} = \begin{pmatrix} U_0 \\ U_1 \\ \vdots \\ U_I \end{pmatrix}, \quad \mathcal{F}(\boldsymbol{U}) = \begin{pmatrix} u^0 - U_0 \\ u_0(T_1; U_0) - U_1 \\ \vdots \\ u_{I-1}(T_I; U_{I-1}) - U_I \end{pmatrix}.$$

Appliquons à ce système non linéaire la méthode de Newton

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

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

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

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

$$U_{i+1}^{n+1} = u_i(T_{i+1}; U_i^n) + \partial_2 u_i(T_{i+1}; U_i^n)(U_i^{n+1} - U_i^n). \tag{6.11}$$

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

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

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

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 $\boldsymbol{U}^0 = \{U_i^0\}$ 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

$$U_{i+1}^{n+1} = G(T_{i+1}, T_i, U_i^{n+1}) + F(T_{i+1}, T_i, U_i^n) - G(T_{i+1}, T_i, U_i^n). \tag{6.12}$$

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 (6.12). Cette simplicité d'utilisation explique en partie l'intérêt suscité par cette méthode.

Si l'on réécrit (6.12) sous la forme

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

et que l'on compare avec la méthode de tir multiple avec résolution par Newton (6.11), on découvre que l'algorithme pararéel est une méthode de tir multiple, où le solveur F est le calcul exact de la solution, et la différence $G(T_{i+1}, T_i, U_i^{n+1}) - G(T_{i+1}, T_i, U_i^n)$ est une approximation naturelle du jacobien $\partial_2 u_i(T_{i+1}; U_i^n)(U_i^{n+1} - U_i^n)$ 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 pararéel, 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;
err(1)=0;
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,:);
  err(1)=max([err(1) norm(ue(i*M/I,:)-U(i+1,:),'inf')]);
end;
G=U;
for n=1:I
  err(n+1)=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 6.4. Sur la première ligne nous voyons la solution approchée elle-même, tandis que sur la deuxième ligne sont tracées les erreurs.

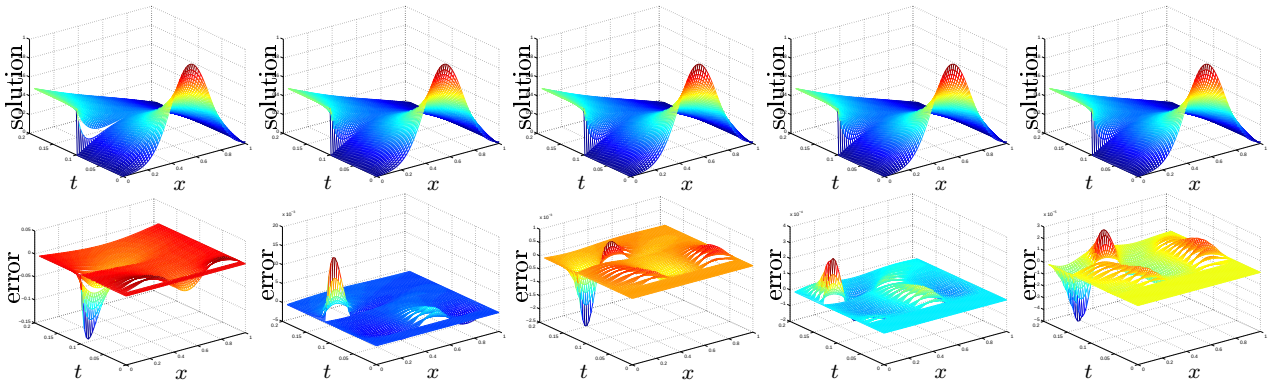


Figure 6.4 – Représentation des solutions et des erreurs pour l'algoritme pararéel

Sur la figure 6.5 nous montrons finalement que l'algorithme converge linéairement dans ce cas, comme il a été prouvé dans [21].

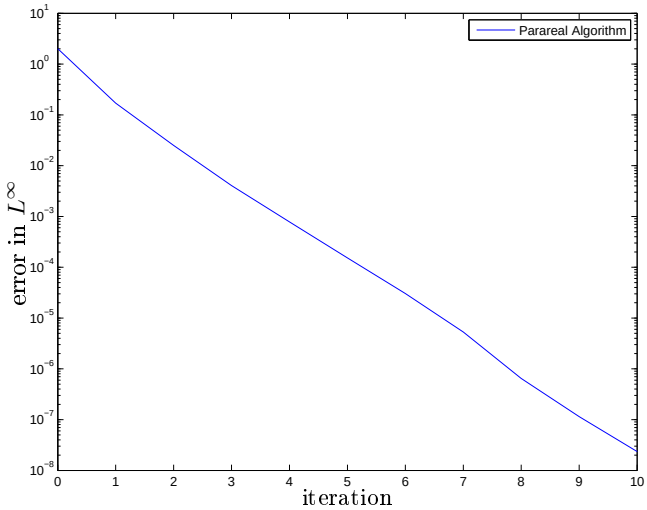


Figure 6.5 – Erreur L^∞ en fonction des itérations pour l'équation de la chaleur traitée par l'algorithme pararéel

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 6.5 que l'erreur 0.1256 est atteinte par l'algorithme pararéel 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 neglige le coût de la solution grossière et de la communication.

6.4 Parallélisation en espace et en temps

Pour finir cet exposé, nous proposons de réaliser un parallélisme espace-temps pour la résolution de l'équation de la chaleur (6.1), en couplant l'approche de Schwarz relaxation d'onde et l'approche pararéelle. Une façon naturelle de coupler les deux approches est de remplacer dans l'algorithme pararéel le solveur fin par un solveur de relaxation d'ondes, voir [31].

Nous présentons une stratégie différente, proposée dans [19]. Le domaine $\Omega \times [0,T]$ est décomposé en sous-domaines espace-temps $\Omega_{ji} = \Omega_j \times [T_i, T_{i+1}]$ représentés figure 6.6.

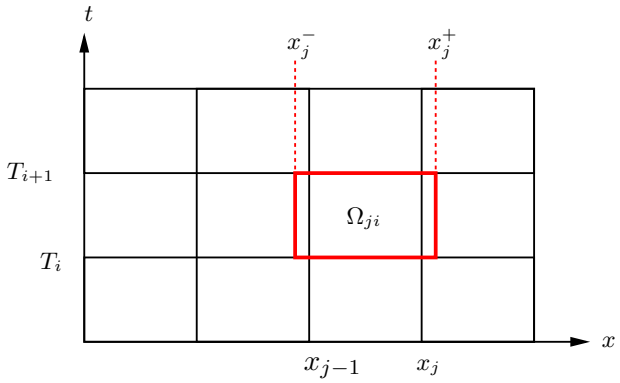


Figure 6.6 – Décomposition en espace-temps pour l'algorithme pararéel-Schwarz relaxation d'onde

Pour résoudre (6.1), nous introduisons dans chaque sous-domaine $\Omega_{j,i}$ un problème type dont la solution est notée $u_{j,i}$:

$$\begin{array}{rcll} \partial_t u_{j,i} - \partial_{xx} u_{j,i} & = & 0 & \text{dans } \Omega_{j,i}, \\ u_{j,i}(\cdot, T_i) & = & U_{j,i} & \text{dans } \Omega_j, \\ \mathcal{B}_j^- u_{j,i}(x_j^-, \cdot) & = & g_{j,i}^- & \text{dans } [T_i, T_{i+1}], \\ \mathcal{B}_j^+ u_{j,i}(x_j^+, \cdot) & = & g_{j,i}^+ & \text{dans } [T_i, T_{i+1}]. \end{array} \tag{6.13}$$

Ici $\mathcal{B}_j^{\pm}$ 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 6.2.

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 $g_{j,i}^- = \mathcal{B}_j^- u_{j-1,i}^0$ et $g_{j,i}^+ = \mathcal{B}_j^+ u_{j+1,i}^0$ sont données pour chaque Ω_{ji}. Nous calculons maintenant pour tout $n = 0, 1, 2, \ldots$

1. des approximations précises

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

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,

$$\begin{aligned} U_{j,i+1}^{n+1} &= u_{j,i}^{n+1}(\cdot, T_{j+1}) + G_{j,i}(U_{j,i}^{n+1}, \mathcal{B}_j^- u_{j-1,i}^{n+1}, \mathcal{B}_j^+ u_{j+1,i}^{n+1}) \\ &\quad - G_{j,i}(U_{j,i}^n, \mathcal{B}_j^- u_{j-1,i}^n, \mathcal{B}_j^+ u_{j+1,i}^n). \end{aligned}$$

Nous illustrons sur la figure 6.7 le comportement d'une première variante de l'algorithme avec des conditions de transmission de type Dirichlet.

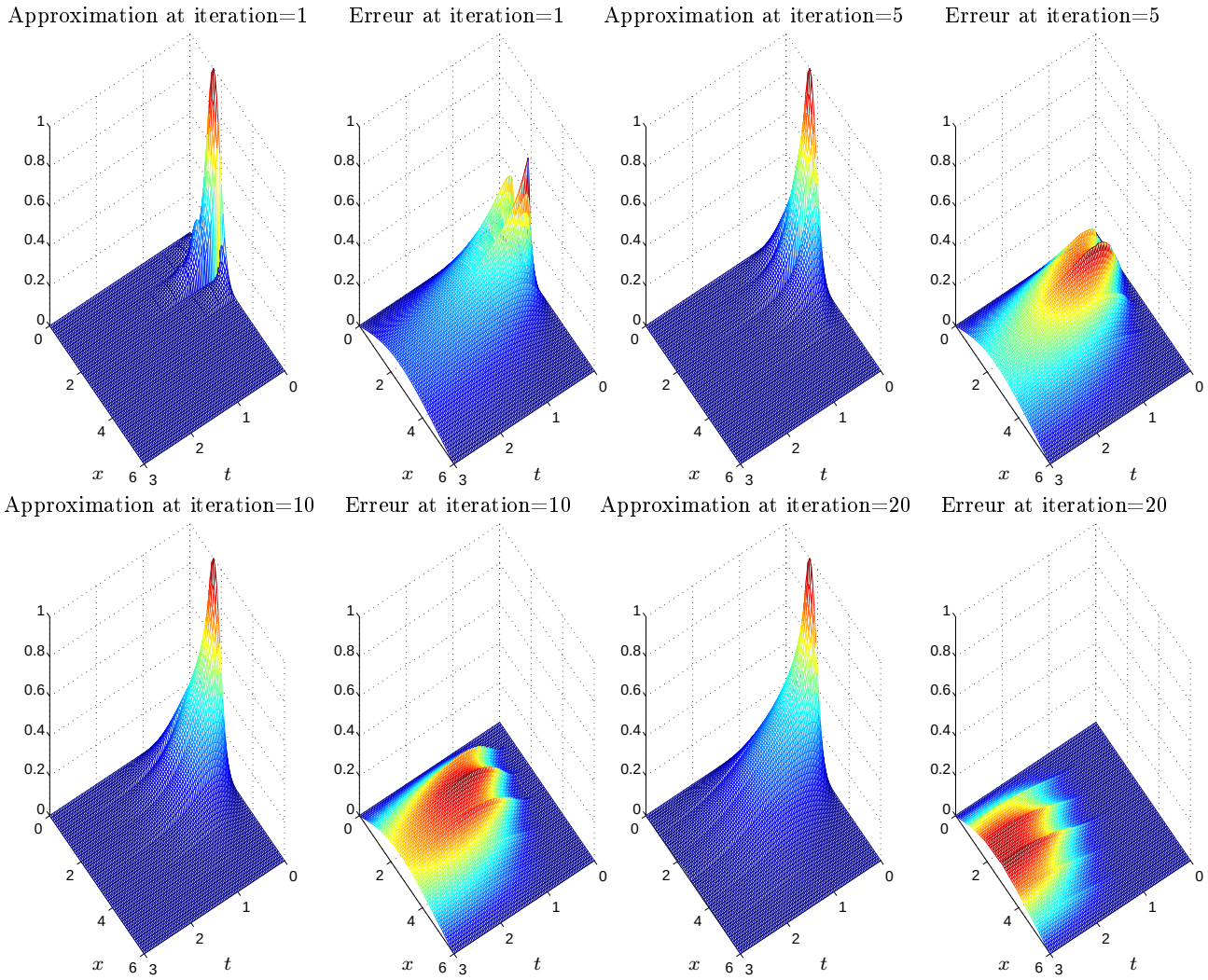


Figure 6.7 – Algorithme pararéel-Schwarz relaxation d'onde avec conditions de transmission de Dirichlet

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 6.2. Les courbes de convergence pour les deux méthodes sont tracées sur la figure 6.8.

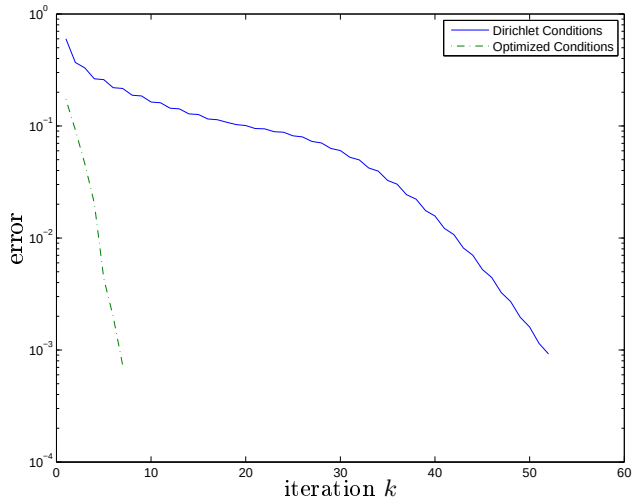


Figure 6.8 – Comparaison de l'algorithme couplé pararéel-Schwarz relaxation d'onde avec conditions de transmission de Dirichlet ou optimisées

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.

Références

[1] V. Arnold, *Équations différentielles ordinaires*, Editions MIR, Moscou, 1974.

[2] M. Bjorhus, *Semi-discrete subdomain iteration for hyperbolic systems*, Tech. Report 4, NTNU, 1995.

[3] J.-F. Bourgat, R. Glowinski, P. Le Tallec, and M. Vidrascu, *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, 1989, SIAM, pp. 3–16.

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

[5] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, in Acta Numerica 1994, Cambridge University Press, 1994, pp. 61–143.

[6] P. Chartier and B. Philippe, *A parallel shooting technique for solving dissipative ODEs*, Computing, 51 (1993), pp. 209–236.

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

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

[9] Q. V. Dinh, B. Mantel, J. Periaux, and R. Glowinski, *Approximate solution of the Navier-Stokes equations for incompressible viscous fluids, related domain decomposition methods*, vol. 1005 of Lect. notes in Math., Springer, 1983, pp. 46–86.

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

[11] M. Dryja and O. B. Widlund, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Report 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 (1995), pp. 121–155.

[13] E. Efstathiou and M. J. Gander, *Why Restricted Additive Schwarz converges faster than Additive Schwarz*, BIT Numerical Mathematics, 43 (2003), pp. 945–959.

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

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

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

[17] M. J. Gander and E. Hairer, *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, 2008, pp. 45–56.

[18] M. J. Gander, L. Halpern, and F. Nataf, *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] M. J. Gander, Y.-L. Jiang, and R.-J. Li, *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] M. J. Gander and A. M. Stuart, *Space-time continuous analysis of waveform relaxation for the heat equation*, SIAM J. Sci. Comput., 19 (1998), pp. 2014–2031.

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

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

[23] E. V. Haynsworth, *On the Schur complement*, Tech. Report 20, Basel Mathematical Notes, June 1968.

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

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

[26] E. Lindelöf, *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 (1894), pp. 117–128.

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

[28] P.-L. Lions, *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, 1988, SIAM, pp. 1–42.

[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, 1989, SIAM, pp. 47–70.

[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, 1990, SIAM, pp. 202–223.

[31] Y. Maday and G. Turinici, *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, 2003, pp. 441–448.

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

[33] J. Mandel and R. Tezaur, *Convergence of a substructuring method with Lagrange multipliers*, Numer. Math., 73 (1996), pp. 473–487.

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

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

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

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

[38] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford Science Publications, 1999.

[39] W. Rudin, *Real and Complex Analysis*, Mc Graw-Hill, 1966.

[40] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1996.

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

[42] B. F. Smith, P. E. Bjørstad, and W. Gropp, *Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.

[43] A. Toselli and O. Widlund, *Domain Decomposition Methods - Algorithms and Theory*, vol. 34 of Springer Series in Computational Mathematics, Springer, 2004.

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

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

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