N° d'ordre : 2273

THÈSE

présentée
pour obtenir

LE TITRE DE DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

École doctorale : Informatique et Télécommunications
Spécialité : Programmation et Systèmes

par

Ming CHAU

Algorithmes Parallèles Asynchrones pour la Simulation Numérique

Soutenue le 3 Novembre 2005 devant le jury composé de :

M.	Henri-Claude BOISSON	Président
M.	Pierre SPITÉRI	Directeur de thèse
M.	Ronan GUIVARCH	Co-directeur de thèse
M.	Martin J. GANDER	Rapporteur
M.	Damien TROMEUR-DERVOUT	Rapporteur
M.	Michael CLIFTON	Examinateur
M.	Didier EL BAZ	Examinateur

Mots clés :

- Calcul scientifique
- Electrophorèse
- Parallélisme
- Algorithmes asynchrones
- Équations aux dérivées partielles
- Méthode alternée de Schwarz
- Problème de l'obstacle

Résumé :

En simulation numérique, la discrétisation des problèmes aux limites nous amène à résoudre des systèmes algébriques de grande dimension. Parmi les voies d'investigation et compte tenu de l'évolution actuelle des architectures des ordinateurs, la parallélisation des algorithmes est une solution naturelle pour résoudre ces problèmes. Or lorsqu'on exploite des calculateurs parallèles, les temps d'attente dus à la synchronisation entre les processus coopérants deviennent pénalisants ; cette perte de temps s'avère d'autant plus considérable en présence de déséquilibre de charge.

Les algorithmes parallèles asynchrones permettent d'envisager de minimiser les pertes de temps dus la synchronisation, sans faire appel aux techniques d'équilibrage de charge. Ce sont des algorithmes itératifs dans lesquels les composantes du vecteur itéré sont réactualisées en parallèle, dans un ordre arbitraire et sans synchronisation. Les restrictions imposées aux algorithmes sont très faibles. De plus, les modèles mathématiques qui décrivent ce type de méthode permettent de prendre en compte le maximum de flexibilité entre les processus et d'assurer, sous certaines hypothèses, la convergence des algorithmes itératifs.

Dans l'étude proposée, les modèles mathématiques ainsi que les théorèmes de convergence des itérations parallèles asynchrones classiques et avec communication flexible sont présentés dans un premier temps. Ensuite, nous exposons la parallélisation de l'algorithme de Schwarz à l'aide de la bibliothèque MPI (Message Passing Interface). Une étude de performance menée sur le serveur de calcul de l'IDRIS (Institut du Développement et des Ressources en Informatique Scientifique) permet de comparer les versions synchrones et asynchrones de l'algorithme parallèle dans le cadre de la résolution d'un problème de convection-diffusion tridimensionnel. Elle met en évidence les gains de temps obtenus grâce à l'asynchronisme. Enfin, sur le plan applicatif, nous nous sommes intéressés à des problèmes tridimensionnels tels que l'électrophorèse de zone à écoulement continu (dont le modèle mathématique résulte d'un couplage entre une équation de Navier-Stokes, une équation de convection-diffusion et une équation de Poisson généralisée) et le problème de l'obstacle (intervenant en mécanique et en mathématiques financières). Dans le cadre de ces applications, des études de performances ont également été menées.

Parallel Asynchronous Algorithms in Numerical Simulation

Keywords :

- Scientific computing
- Electrophoresis
- Parallelism
- Asynchronous algorithms
- Partial differential equations
- Alternating Schwarz method
- Obstacle problem

Abstract :

In numerical simulation, the discretization of boundary value problems lead to the solution of large sparse linear systems. Among the research topics and regard to the evolution of computer architectures, the parallelisation of the algorithms is a natural way to overcome the problems. However, the overhead due to the synchronization between the processors is the drawback of the use of parallel computers ; the waste of time is even more significant as the load is unbalanced.

Parallel asynchronous algorithms allow to minimize the overhead due to synchronisation, without using load balancing techniques. These iterative algorithms consist in updating the components of the iteration vector in a parallel way, without synchronization, in an arbitrary order. The restrictions imposed on these algorithms are very weak. Furthermore, the mathematical models that describe the considered algorithms take into account very flexible parallel computation schemes and ensure the convergence of the iterative algorithms, under some hypothesis.

The structure of the thesis is as follows. Firstly, mathematical models and convergence results of classical and flexible communication asynchronous iterations are presented. Then the implementation of parallel asynchronous Schwarz algorithm using MPI (Message Passing Interface) is exposed. The synchronous and asynchronous implementations of the algorithms are compared in the context of the solution of 3D convection-diffusion equations. The numerical experiments are carried out on the supercomputer of IDRIS (Institut du Développement et des Ressources en Informatique Scientifique). The benefits brought by asynchronism are shown. Finally, the algorithms are applied to the solution of 3D problems such as the continuous flow electrophoresis (which consists in coupling an incompressible Navier-Stokes equation with a convection-diffusion equation and a generalised Poisson equation) and the obstacle problem (which occurs in financial mathematics). Performance studies have also been carried out in the context of these applications.

Theory,
that funny thing . . .
John Coltrane

Remerciements

Je tiens à assurer de toute ma reconnaissance :

Mon directeur de thèse, Pierre Spitéri, professeur à l'ENSEEIHT, pour l'aide précieuse qu'il a m'apporté sur le plan de l'analyse numérique, pour son soutien constant et pour ses remarques pertinentes qui ont permis d'améliorer ce manuscrit.

Mon co-directeur de thèse, Ronan Guivarch, maître de conférence à l'ENSEEIHT, pour les conseils dont j'ai bénéficié concernant la programmation sur calculateur parallèle ainsi que pour la rédaction du manuscrit.

Henri-Claude Boisson, directeur de recherche CNRS à l'IMFT, qui m'a initié à la mécanique des fluides numérique. C'est grâce à ses qualités pédagogiques et à sa disponibilité que mes travaux ont pu avancer.

Didier El Baz, chargé de recherche CNRS au LAAS, pour m'avoir fait profiter de ses compétences et de ses connaissances sur les algorithmes asynchrones et pour avoir accepté de faire partie du jury.

Michael Clifton, directeur de recherche CNRS au LGC, pour m'avoir aidé à comprendre quelques subtilités du procédé d'électrophorèse de zone à écoulement continu. Son aide m'a permis de corriger des défauts dans le code du simulateur. Je suis honnoré par sa participation au jury.

Martin Gander, professeur à l'Université de Genève, et Damien Tromeur-Dervout, professeur à l'Université Lyon 1, pour l'intérêt qu'ils ont manifesté vis à vis de cette étude, et qui ont accepté d'être les rapporteurs de ce travail.

Je tiens également à remercier mes collègues, Thomas Haberkorn, Dorin Preda, Pierre Martinon, Romain Dujol et Ahmed Touhami, docteurs et doctorants matheux de l'ENSEEIHT-IRIT, pour l'ambiance chaleureuse qui a régné durant ces années.

Je remercie également l'IDRIS, pour les ressources de calcul qui m'ont été accordées ainsi que le support technique dont j'ai benéficié.

Enfin, je souhaite remercier l'ensemble du personnel enseignant, administratif et technique de l'ENSEEIHT, de l'INPT et de l'IRIT.

Table des matières

Liste des tableaux

Table des figures

Introduction générale

L'informatique est une science qui fournit des outils indispensables permettant de mener à bien des projets scientifiques, aussi bien dans le contexte académique qu'industriel. Nous nous intéressons dans cette étude à la simulation numérique qui est une démarche devenue incontournable dans de nombreux domaines scientifiques et industriels, notamment grâce à la puissance de calcul et à la capacité de stockage des ordinateurs actuels. C'est une démarche expérimentale qui consiste à calculer par ordinateur les caractéristiques d'un phénomène quelconque à partir des équations mathématiques qui le modélisent. Parmi les domaines où la simulation numérique intervient, nous pouvons citer l'avionique, le génie chimique, la mécanique des fluides, le nucléaire, la physique des plasmas, la météorologie, l'océanographie ou encore la biologie moléculaire.

La résolution numérique des équations qui modélisent les phénomènes est la principale source de difficulté des simulations en raison des temps de calcul prohibitifs. Lorsque les phénomènes sont régis par des équations aux dérivées partielles, les méthodes de discrétisation conduisent à la résolution d'équations algébriques dont la complexité et le nombre d'inconnues sont d'autant plus élevés que les exigences sur la précision des résultats sont contraignantes. Pour faire face aux besoins grandissants en ressource de calcul, le développement conjoint des architectures et des algorithmes parallèles est une voie d'investigation très explorée actuellement.

Contrairement aux applications parallèles telles que l'exploration de données (*data mining*), la résolution parallèle de systèmes algébriques fait intervenir des algorithmes nécessitant de nombreux échanges de donnée entre les différentes unités de calcul. Une parallélisation efficace est difficile à mettre en œuvre pour le type de problème qui nous concerne en raison des temps de latence induits par les accès à des données distantes. Ces accès représentent un handicap d'autant plus important lorsque les unités de calcul doivent se synchroniser durant les phases de communication. Le fait de devoir attendre qu'une donnée distante devienne disponible avant de poursuivre un calcul provoque nécessairement une dégradation des performances d'un algorithme parallèle. Le problème soulevé devient d'autant plus vrai que le nombre de processeurs augmente et que la fréquence des échanges est élevée. De plus, tout déséquilibre entre les charges de calcul attribuées aux différents processeurs contribue à diminuer l'efficacité de la parallélisation.

C'est dans l'optique de s'affranchir de ces problèmes de perte de performance

que les algorithmes parallèles asynchrones ont été développés. Ces méthodes asynchrones, applicables à la résolution de problèmes de point fixe, utilisent l'absence de synchronisation avec pour objectif de tirer parti du maximum de la puissance de calcul en supprimant les temps d'inactivité dues aux attentes bloquantes. Il en résulte que le calcul des composante du vecteur itéré est effectué en parallèle et sans aucun ordre a priori. Ce type d'algorithme a été expérimenté dès 1967 par J.L. Rosenfeld et leur convergence a été étudiée en 1969, dans le cadre de la résolution de systèmes linéaires, par D. Chazan et W. Miranker. Par la suite, les travaux de G. Baudet, J. Bahi, D.P. Bertsekas, D. El Baz, M.N. El Tarazi, A. Frommer, J.C. Miellou, P. Spitéri, D. Szyld, et J. Tsitsiklis ont permis d'établir des théorèmes de convergence dans le cadre non linéaire, avec des modèles de l'asynchronisme de plus en plus généraux, dans lesquels les dernières valeurs disponibles du vecteur itéré sont exploitées lors des accès aux données distantes. La liste précédente n'est pas exhaustive dans la mesure où d'autres auteurs non cités ici ont contribué à l'étude des méthodes asynchrones.

Dans le présent travail, nous nous sommes principalement intéressés à la mise en œuvre du parallélisme asynchrone dans le cadre de la résolution de problèmes aux limites linéaires et non linéaires définis dans des domaines tridimensionnels. En particulier, nous avons étudié la simulation parallèle d'un procédé électrochimique : l'électrophorèse de zone à écoulement continu. Ce procédé permet de séparer des mélanges d'espèces ioniques grâce à l'exploitation du phénomène d'électrophorèse dans un fluide en écoulement. Cette simulation fait donc intervenir une équation de Navier-Stokes, une équation de transport et une équation de potentiel électrique couplées entre elles. La résolution parallèle des systèmes algébriques issus des différentes équations fait appel à une méthode de sous-domaines avec recouvrement : l'algorithme parallèle de Schwarz. Les objectifs de cette étude sont d'une part de donner une méthode permettant d'appliquer un algorithme parallèle asynchrone pour la résolution d'un problème numérique concret et complexe, et d'autre part de montrer la pertinence de ce choix algorithmique en comparant les performances des versions synchrones et asynchrones du simulateur. Les essais ont été menés sur le serveur de calcul de l'IDRIS (Institut du Développement et des Ressources en Informatique Scientifique) [1] sur 2 à 16 processeurs.

L'algorithme de Schwarz qui a été implémenté a fait l'objet d'une étude de performance à part entière dans laquelle les versions synchrones et asynchrones de l'algorithme parallèle ont été testées sur 2 à 128 processeurs. À titre d'exemple, nous avons réalisé cette étude en résolvant un problème de convection-diffusion tridimensionnel. Ces essais ont pour but d'évaluer la scalabilité des méthodes asynchrones, dans l'optique de mettre au point des applications massivement parallèles. L'engagement des expérimentations dans cette voie est motivé par l'évolution des architectures parallèles et la croissance des besoins en ressources de calcul. Actuellement, le couplage de calculateurs parallèles à mémoire partagée par un réseau local à haute perfor-

[1] http ://www.idris.fr

mance est une technique très répandue pour augmenter le degré de parallélisme. Toutefois, les échanges de messages via le réseau sont bien plus lents que les transferts de donnée entre la mémoire partagée et la mémoire cache d'un processeur. Dans ces conditions, la synchronisation ne permet pas d'exploiter pleinement le potentiel offert par la mémoire partagée car la lenteur du réseau a des répercussions sur les performances globales. L'intérêt des algorithmes asynchrones est donc d'amortir le surcoût induit par les communications via le réseau.

Enfin, une étude distincte est également consacrée à la parallélisation asynchrone de l'algorithme de Richardson projeté, permettant de résoudre le problème de l'obstacle. C'est une étude à la fois théorique et expérimentale. En utilisant un résultat de convergence de M.N. El Tarazi, nous avons montré que l'algorithme de Richardson projeté converge sous une hypothèse plus faible que la condition classique de coercivité. Sur le plan expérimental, les versions synchrones et asynchrones de l'algorithme ont été testées sur 2 à 32 processeurs.

Ce mémoire se divise en 4 parties.

- Le chapitre 1 est consacré à un état de l'art sur la théorie des itérations parallèles synchrones et asynchrones. Nous présenterons en particulier le modèle classique et les modèles avec communication flexible qui autorisent plus de souplesse dans les accès aux données distantes.
- Le chapitre 2 est consacré à l'implémentation des versions synchrones et asynchrones de l'algorithme de Schwarz parallèle et à la mise en œuvre du concept de communication flexible. Les résultats des études de performance y seront exposés dans le cas de l'étude d'un problème de convection-diffusion tridimensionnel.
- Le chapitre 3 est consacré au traitement numérique de la simulation parallèle de l'électrophorèse de zone à écoulement continu et aux expérimentations sur calculateur parallèle. La discrétisation des équations ainsi que la mise en application des algorithmes asynchrones seront traitées.
- Le chapitre 4 est consacré à l'analyse et à la résolution numérique parallèle du problème de l'obstacle par l'algorithme de Richardson projeté. Sur le plan théorique en particulier, un lien avec les résultats présentés au chapitre 1 est effectué.

Chapitre 1

Algorithmes parallèles asynchrones

1.1 Introduction

Les algorithmes parallèles asynchrones présentent actuellement un grand intérêt avec le développement du parallélisme massif. À l'origine, ces méthodes itératives ont été introduites en 1969 par D. Chazan et W. Miranker [21] pour la résolution de systèmes linéaires. Cette étude a été généralisée en 1974 par J.C. Miellou [67] pour la résolution de systèmes algébriques non linéaires de grande taille. Une extension de cette dernière étude a également été poursuivie par G. Baudet en 1978 [10]. Dans ce type d'algorithme itératif, l'asynchronisme est pris en compte à l'aide de retards sur les blocs de composantes, chacun d'entre eux ayant des retards différents. De plus, les réactualisations du vecteur itéré sont effectuées sans synchronisation, ni ordre précis (*cf.* figure 1.1). Cependant, dans cette formulation classique, les communications interviennent uniquement à la fin de chaque réactualisation. Cet état de fait traduit un manque de souplesse dans les communications.

Les techniques d'analyse de la convergence des algorithmes parallèles asynchrones classiques les plus usuelles sont :

- les techniques de contraction pour une norme vectorielle adaptée [80], employées par D. Chazan et W. Miranker, J.C. Miellou durant ses premiers travaux et G. Baudet ; ces techniques ont aussi été utilisées dans [81, 20, 25, 5],
- les techniques de contraction sur un espace produit muni de la norme uniforme avec poids, employées par M.N. El Tarazi en 1982 [35, 36],
- les techniques d'ordre partiel (convergence monotone dans un espace partiellement ordonné), employées par J.C. Miellou en 1975 [68] ; ces techniques ont également été utilisées dans [58, 35, 37],
- les techniques d'ensembles emboîtés, employées par D.P. Bertsekas et J.N. Tsitsiklis en 1989 [15].

Récemment, l'analyse de la convergence des algorithmes asynchrones par des méthodes probabilistes a été introduite par J.C. Strikwerda [97].

En 1998, J.C. Miellou, D. El Baz et P. Spitéri ont introduit une nouvelle

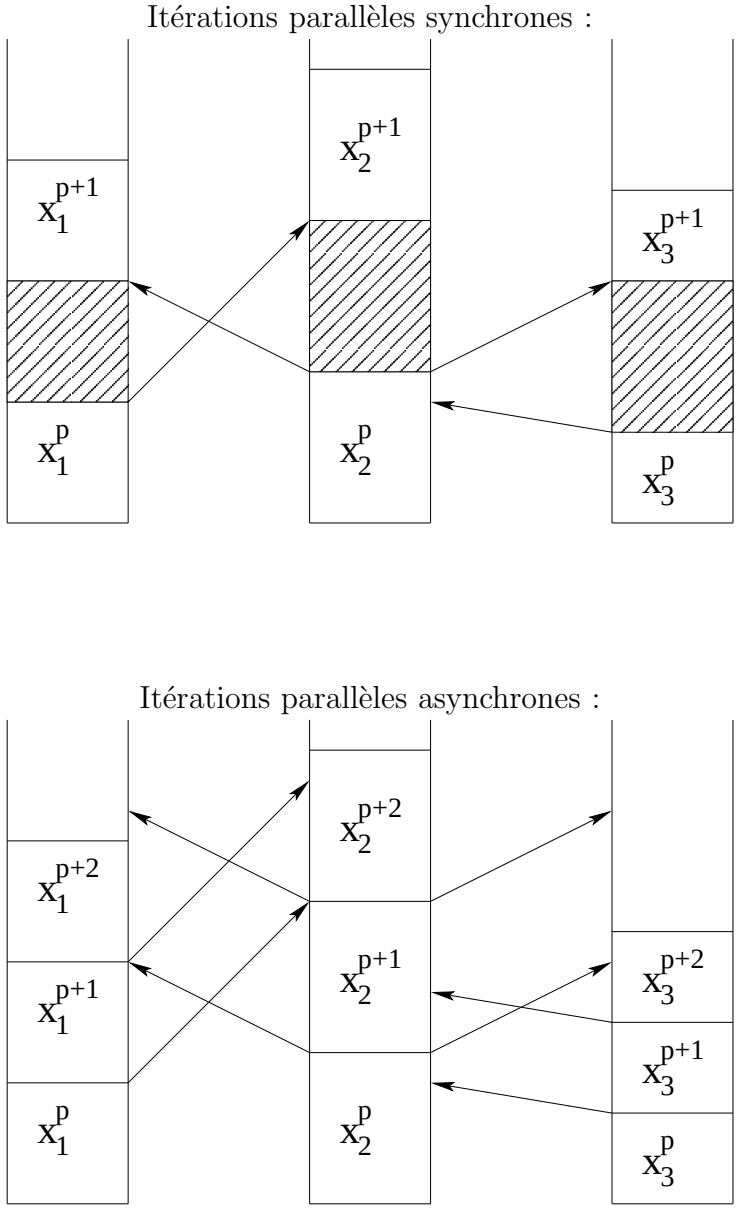
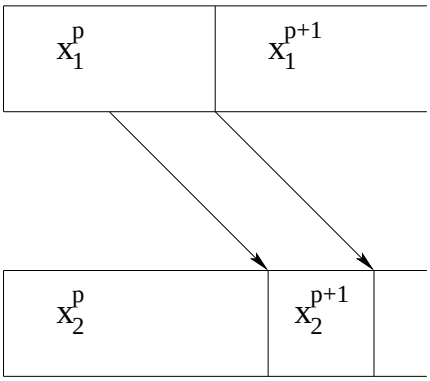


FIG. 1.1 – Itérations parallèles synchrones et asynchrones

classe de méthode : les algorithmes itératifs parallèles asynchrones avec communication flexible [71]. Ces algorithmes présentent la particularité d'intégrer étroitement les aspects communications aux aspects calculs. L'originalité par rapport aux algorithmes asynchrones classiques réside dans le fait que les communications interviennent pendant les réactualisations du vecteur itéré et non à l'issue de chaque itération. De plus, les réactualisations sont effectuées grâce aux valeurs partielles des blocs composantes du vecteur itéré (*cf.* figure 1.2). Il en résulte que ces méthodes

Exploitation de données provenant de calculs en cours de réalisation :



Prise en compte des messages reçus pendant les calculs :

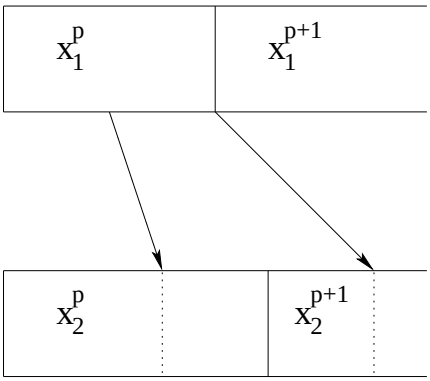


FIG. 1.2 – Itérations parallèles asynchrones avec communication flexible

parallèles sont très générales. Les seules contraintes qui leur sont imposées étant :

- aucune composante du vecteur itéré ne cesse d'être réactualisée de manière définitive lors de la résolution numérique ;
- des valeurs plus récentes du vecteur itéré doivent être utilisées au fur et à mesure que les calculs progressent.

L'analyse de la convergence de ces méthodes a été effectuée par des techniques d'ordre partiel. En 2004, D. EL BAZ, A. FROMMER et P. SPITÉRI ont proposé une nouvelle

formulation des algorithmes parallèles asynchrones avec communication flexible pour laquelle l'analyse de convergence est effectuée par des techniques de contraction [33].

Sur le plan applicatif, les méthodes parallèles asynchrones ont permis de résoudre de nombreux problèmes, modélisés par différents types de formulation :

- les équations aux dérivées partielles fortement non linéaires, parmi lesquels nous pouvons citer les problèmes de diffusion non linéaires et de convection-diffusion non linéaires, le problème d'obstacle intervenant en mathématique financière et en mécanique, ainsi que l'équation d'Hamilton-Jacobi-Bellmann intervenant en traitement d'image [88, 89, 12, 39, 48, 49, 50, 51, 54, 95, 92, 96, 94, 2],
- les problèmes linéaires ou non linéaires, simple-flots ou multi-flots, qui trouvent leurs applications dans la distribution d'eau ou de gaz, l'acheminement de produits, l'allocation de ressources, la communication dans les réseaux informatiques ou le transport [14, 18, 29, 34],
- la résolution de grands systèmes linéaires creux dans le domaine de la modélisation de grands systèmes à l'aide de chaînes de Markov [13, 30, 59],
- la résolution de grands systèmes linéaires par les méthodes de *multisplitting* (introduites par D.P. O'Leary et R.E. White [75]) [102, 16, 8, 6, 43, 17, 45, 7, 96].

Ces méthodes asynchrones peuvent aussi s'appliquer à la résolution de problèmes algébro-différentiels [4, 65], à la programmation dynamique [99, 100], à la résolution de l'équation de Boltzmann [55], ainsi qu'à bien d'autres types de problème. Notons que les algorithmes asynchrones peuvent également converger dans le cas de la résolution de certains systèmes linéaires singuliers [74, 41].

Dans le présent chapitre, nous présenterons dans la section 1.2 le modèle classique des algorithmes asynchrones et nous rappellerons les résultats de convergence associés, basés sur les techniques de contraction. Dans les sections 1.3 et 1.4, nous présenterons deux modèles d'itérations asynchrones avec communication flexible. Les approches par les techniques d'ordre partiel et de contraction y seront exposées. Des exemples simples issus de la discrétisation d'équations aux dérivées partielles illustreront chaque cas de figure. Enfin, la connexion entre les algorithmes asynchrones et la méthode de Schwarz sera traitée dans la section 1.5. Nous renvoyons à [69, 40, 44, 46, 47] pour des présentations détaillées de la méthode.

1.2 Algorithmes parallèles asynchrones classiques

1.2.1 Modèle mathématique

Soit E un espace de Banach, considéré comme un produit fini d'espaces de Banach :

$$E = \prod_{i=1}^{\beta} E_i \tag{1.1}$$

où β est un entier naturel. La décomposition de tout vecteur $X \in E$ s'écrit :

$$X = (x_1, \ldots, x_i, \ldots, x_\beta) \tag{1.2}$$

où $\forall i \in \{1, \ldots, \beta\}$, $x_i \in E_i$.

Chaque espace E_i est muni d'une norme notée $|\,.\,|_i$. La norme vectorielle canonique $\mathrm{q}(\,.\,)$ de tout $X \in E$ est le vecteur positif de $\mathbb{R}^\beta$ défini comme suit :

$$\forall X \in E,\ \mathrm{q}(X) = (|x_1|_1, \ldots, |x_i|_i, \ldots, |x_\beta|_\beta). \tag{1.3}$$

Soit F une application de $\mathcal{D}(F) \subset E$ à valeurs dans $\mathcal{D}(F)$, telle que $\mathcal{D}(F) \neq \emptyset$. On s'intéresse alors au problème de point fixe

$$X^* = F(X^*),\ X^* \in \mathcal{D}(F). \tag{1.4}$$

Compte tenu de la décomposition de l'espace E, l'application F se décompose de la manière suivante :

$$F(X) = (F_1(X), \ldots, F_i(X), \ldots, F_\beta(X)). \tag{1.5}$$

Chaque composante F_i est une application de $\mathcal{D}(F)$ à valeurs dans $E_i \cap \mathcal{D}(F)$.

Nous introduisons à présent les éléments permettant de modéliser le parallélisme ainsi que le comportement chaotique des algorithmes asynchrones.

Définition 1.1 (Stratégie)
Une stratégie $\mathcal{S}$ est une suite $(s(p))_{p\in\mathbb{N}}$ de parties non vides de $\{1, \ldots, \beta\}$ vérifiant :

$$\forall i \in \{1, \ldots, \beta\},\ \{p \in \mathbb{N} \mid i \in s(p)\} \text{ est un ensemble infini.} \tag{1.6}$$

Définition 1.2 (Suite de retards)
Une suite de retards $\mathcal{R}$ est une suite $(r(p))_{p\in\mathbb{N}}$ où :

$$\forall p \in \mathbb{N},\ r(p) = (r_1(p), \ldots, r_i(p), \ldots, r_\beta(p)) \in \mathbb{N}^\beta \tag{1.7}$$

et telle que pour tout $i \in \{1, \ldots, \beta\}$, l'application $\rho : \mathbb{N} \to \mathbb{N}^\beta$, de composantes $\rho_i : p \mapsto p - r_i(p)$ vérifie :

$$\forall p \in \mathbb{N},\ 0 \leq \rho_i(p) \leq p \tag{1.8}$$

$$\lim_{p\to\infty} (\rho_i(p)) = +\infty. \tag{1.9}$$

Compte tenu de ces définitions, les algorithmes parallèles asynchrones peuvent se formuler de la manière suivante [21, 67, 10] :

Définition 1.3 (Algorithme parallèle asynchrone classique)
Soient $X^0 \in \mathcal{D}(F)$ un vecteur quelconque, $\mathcal{S}$ une stratégie et $\mathcal{R}$ une suite de retards. Un algorithme parallèle asynchrone classique construit récursivement une suite d'itérés $(X^p)_{p\in\mathbb{N}}$ de la manière suivante :

$$\forall p \geq 0,\ \forall i \in \{1, \ldots, \beta\},\ x_i^{p+1} = \begin{cases} x_i^p & \text{si } \ i \notin s(p) \\ F_i(\tilde{X}^p) & \text{si } \ i \in s(p) \end{cases} \tag{1.10}$$

où

$$\tilde{X}^p = (x_1^{\rho_1(p)}, \ldots, x_i^{\rho_i(p)}, \ldots, x_\beta^{\rho_\beta(p)}). \tag{1.11}$$

Interprétation de la stratégie et des retards La stratégie rend compte de l'ordre des calculs effectués en parallèle. Les indices appartenant à $s(p)$ désignent les composantes relaxées en parallèle à la $p^{\text{ème}}$ itération.
La suite de retards rend compte de la disponibilité des données, ce qui permet de modéliser l'absence de synchronisation entre les processeurs ainsi que les retards dus aux temps de latence des communications.
L'hypothèse (1.6) sur la stratégie exprime le fait qu'aucune composante ne cesse d'être réactualisée. Quant à l'hypothèse (1.9), elle garantit la mise à l'écart des composantes trop anciennes. Ainsi, les réactualisations des composantes du vecteur itéré font toujours appel à des données suffisamment récentes.

Remarque 1.1 (Algorithme parallèle synchrone) La définition 1.3 des itérations parallèles asynchrones est suffisamment générale pour modéliser les itérations synchrones. Il suffit de considérer une suite de retards nulle [82]. Dans ce contexte, les méthodes de relaxation séquentielles classiques sont modélisées par des choix particuliers de $\mathcal{S}$:

- si $\forall p \in \mathbb{N}$, $s(p) = \{1, \ldots, \beta\}$, nous retrouvons la méthode de Jacobi,
- si $\forall p \in \mathbb{N}$, $s(p) = 1 + (p \bmod \beta)$, nous retrouvons la méthode de Gauss-Seidel.

1.2.2 Théorèmes de convergence

Définition 1.4 (Contraction pour la norme vectorielle)
Soit F une application de $\mathcal{D}(F) \subset E$ à valeurs dans $\mathcal{D}(F)$, telle que $\mathcal{D}(F) \neq \emptyset$. F est contractante en $X^* \in \mathcal{D}(F)$ pour la norme vectorielle $\mathrm{q}(\,.\,)$ *s'il existe une matrice J, de taille $\beta \times \beta$ et de rayon spectral $\rho(J)$, vérifiant :*

$$J \geq 0 \tag{1.12}$$
$$\rho(J) < 1 \tag{1.13}$$

et telle que :

$$\forall W \in \mathcal{D}(F),\ \mathrm{q}(F(W) - F(X^*)) \leq J.\mathrm{q}(W - X^*). \tag{1.14}$$

On dit que J est une matrice de contraction de F en X^*.

Remarque 1.2 Une application contractante pour la norme vectorielle l'est aussi pour la norme uniforme avec poids définie par :

$$\forall X \in E,\ \|X\|_\Gamma = \max_{1 \leq i \leq \beta} \frac{|x_i|_i}{\gamma_i}$$

où J est la matrice de contraction de F en X^*. Quant au vecteur Γ de composantes $(\gamma_i)_{1 \leq i \leq \beta}$, il vérifie $\Gamma > 0$ et $J.\Gamma \leq \nu\,\Gamma$ où ν est un réel tel que $\rho(J) \leq \nu < 1$. L'existence et la non-négativité du vecteur Γ sont établies à l'aide du théorème de Perron-Frobenius [101]. Nous renvoyons à [67] pour la démonstration.

Théorème 1.1 (Convergence en norme vectorielle)
Soit F une application de $\mathcal{D}(F) \subset E$ à valeurs dans $\mathcal{D}(F)$, telle que $\mathcal{D}(F) \neq \emptyset$. Si F admet un point fixe $X^* \in \mathcal{D}(F)$ et si F est contractante en X^* pour la norme vectorielle q(.), *alors la suite $(X^p)_{p\in\mathbb{N}}$ construite à l'aide de l'algorithme parallèle asynchrone classique (1.10) converge vers le point fixe X^*.*

Ce théorème de convergence est démontré par J.C. Miellou dans [67] (voir aussi G. Baudet [10] dans le cas de retards non bornés).

Notons qu'il est possible d'introduire un paramètre de relaxation dans les itérations parallèles asynchrones classiques. On considère alors une application F_ω, de domaine de définition $\mathcal{D}(F)$, telle que :

$$\forall X \in \mathcal{D}(F),\ F_\omega(X) = (1-\omega)X + \omega F(X). \tag{1.15}$$

D'après [67], F_ω possède le même point fixe que F et elle est contractante en norme vectorielle si :

$$\omega \in \left]0\,,\ \frac{2}{1+\rho(J)}\right[. \tag{1.16}$$

Voici un résultat de convergence en norme uniforme avec poids établi par M.N. El Tarazi dans [36] ; cette dernière est définie par :

$$\forall X \in E,\ \|X\|_\Gamma = \max_{1\leq i\leq \beta} \frac{|x_i|_i}{\gamma_i} \tag{1.17}$$

où le vecteur Γ, de dimension β, a des composantes γ_i, strictement positives.

Théorème 1.2 (Convergence en norme scalaire)
Soit F une application de $\mathcal{D}(F) \subset E$ à valeurs dans $\mathcal{D}(F)$, telle que $\mathcal{D}(F) \neq \emptyset$. Si F admet un point fixe $X^* \in \mathcal{D}(F)$ et si F est contractante en X^* pour une norme uniforme avec poids $\|\,.\,\|_\Gamma$, autrement dit :

$$\exists \theta \in]0\,,\,1[,\ \forall W \in \mathcal{D}(F),\ \|F(W) - F(X^*)\|_\Gamma \leq \theta \|W - X^*\|_\Gamma \tag{1.18}$$

alors la suite $(X^p)_{p\in\mathbb{N}}$ construite à l'aide de l'algorithme parallèle asynchrone classique (1.10) converge vers le point fixe X^*, pour la norme $\|\,.\,\|_\Gamma$.

1.2.3 Rappels sur la notion d'accrétivité

Nous présentons ici les éléments permettant d'obtenir des conditions suffisantes de convergence des algorithmes parallèles asynchrones classiques.

La notion d'opérateur accrétif correspond à une généralisation de la notion d'opérateur monotone. Elle permet

- d'analyser l'existence de solution à des problèmes aux limites d'évolution dans des espaces fonctionnels particuliers [11, 9],
- d'analyser des classes de problèmes pour lesquels la résolution numérique par les algorithmes parallèles asynchrones classiques est possible via le résultat énoncé au théorème 1.1 [88, 73, 49].

Opérateurs accrétifs

On indique ci-dessous les définitions générales portant sur l'accrétivité dans les espaces de Banach. Pour une présentation approfondie, nous renvoyons à [11, 9].

Ici E désigne un espace de Banach et E^* son dual topologique, de normes respectives $\|\,.\,\|$ et $\|\,.\,\|^*$. On note $\langle\,.\,,\,.\,\rangle$ le produit de dualité entre E et E^*. Soit Λ un opérateur de $\mathcal{D}(\Lambda) \subset E$ à valeurs dans E. Afin de simplifier les énoncés, on considère Λ univoque.

Définition 1.5 (Opérateur de dualité)
On appelle opérateur de dualité $\mathcal{G}(\,.\,)$ associé à E, l'opérateur de E vers E^* défini par :

$$\forall X \in E,\ \mathcal{G}(X) = \{g \in E^* \mid \|g\|^* = \|X\| \text{ et } \langle X\,,\,g\rangle = \|X\|^2\}. \tag{1.19}$$

On montre que cette multi-application est non-vide, fermée et que c'est le sous-différentiel de la fonction $X \mapsto \frac{1}{2}\|X\|^2$.

Remarque 1.3 Dans le cas où E est un espace de Hilbert, E^* est identifié à E, $\mathcal{G}(X)$ se réduit à X et le produit scalaire est substitué au produit de dualité.

Définition 1.6 (Accrétivité)
Λ est un opérateur accrétif si

$$\forall (X, X') \in \mathcal{D}(\Lambda)^2,\ \exists g \in \mathcal{G}(X - X') \text{ tel que } \langle \Lambda(X) - \Lambda(X')\,,\,g\rangle \geq 0. \tag{1.20}$$

Λ est strictement accrétif si l'inégalité (1.20) est stricte.

Définition 1.7 (Accrétivité forte)
Λ est un opérateur fortement accrétif s'il existe un réel c positif tel que :

$$\begin{aligned}\forall (X, X') \in \mathcal{D}(\Lambda)^2,\ &\exists g \in \mathcal{G}(X - X') \text{ tel que}\\ &\langle \Lambda(X) - \Lambda(X')\,,\,g\rangle \geq c\,\|X - X'\|^2.\end{aligned} \tag{1.21}$$

Accrétivité en dimension finie

Nous nous intéressons à présent à la caractérisation des applications linéaires accrétives de $\mathbb{R}^n$. Pour une étude détaillée, nous renvoyons à [88], notamment pour les démonstrations.

On note $\langle\,.\,,\,.\,\rangle$ le produit scalaire canonique de $\mathbb{R}^n$ et $\|\,.\,\|_2$ la norme euclidienne. Soit A une matrice carrée de taille $n \times n$ et de coefficients $(a_{ij})_{1\leq i,j\leq n}$.

Proposition 1.1 (Accrétivité forte dans $(\mathbb{R}^n, \|\,.\,\|_2)$)
Une matrice A est fortement accrétive dans $\mathbb{R}^n$ muni de la norme euclidienne $\|\,.\,\|_2$ si et seulement si A est fortement définie positive, c'est à dire :

$$\exists c \geq 0,\ \forall X \in \mathbb{R}^n (X \neq 0),\ \langle AX\,,\,X\rangle \geq c\,\|X\|_2^2. \tag{1.22}$$

Corollaire 1.1 (Accrétivité dans $(\mathbb{R}^n, \|\,.\,\|_2)$) *Une matrice A est accrétive dans $\mathbb{R}^n$ muni de $\|\,.\,\|_2$ si et seulement si A est une matrice semi-définie positive.*

Il est également intéressant de caractériser les matrices accrétives de $\mathbb{R}^n$ normé par :

- $\|X\|_1 = \sum_{i=1}^n |x_i|$,
- $\|X\|_\infty = \max_{1\leq i\leq n} |x_i|$.

Comme nous allons le constater, les hypothèses correspondantes sont techniquement beaucoup plus simples à vérifier que l'inégalité (1.22).

Proposition 1.2 (Accrétivité forte dans $(\mathbb{R}^n, \|\,.\,\|_1)$)
Une matrice A est fortement accrétive dans $\mathbb{R}^n$ muni de la norme $\|\,.\,\|_1$ si et seulement si il existe un réel positif c tel que pour tout $i \in \{1, \ldots, n\}$:

$$a_{ii} \geq c \tag{1.23}$$

$$a_{ii} - \sum_{j\neq i} |a_{ji}| \geq c. \tag{1.24}$$

Corollaire 1.2 (Accrétivité dans $(\mathbb{R}^n, \|\,.\,\|_1)$) *Une matrice A est accrétive dans $\mathbb{R}^n$ muni de la norme $\|\,.\,\|_1$ si et seulement si :*

1. *les coefficients diagonaux de A sont positifs ou nuls,*
2. *A est diagonale dominante en colonne.*

Proposition 1.3 (Accrétivité forte dans $(\mathbb{R}^n, \|\,.\,\|_\infty)$)
Une matrice A est fortement accrétive dans $\mathbb{R}^n$ muni de la norme $\|\,.\,\|_\infty$ si et seulement si il existe un réel positif c tel que pour tout $i \in \{1, \ldots, n\}$:

$$a_{ii} \geq c \tag{1.25}$$

$$a_{ii} - \sum_{j\neq i} |a_{ij}| \geq c. \tag{1.26}$$

Corollaire 1.3 (Accrétivité dans $(\mathbb{R}^n, \|\,.\,\|_\infty)$) *Une matrice A est accrétive dans $\mathbb{R}^n$ muni de la norme $\|\,.\,\|_\infty$ si et seulement si :*

1. *les coefficients diagonaux de A sont positifs ou nuls,*
2. *A est diagonale dominante en ligne.*

Remarque 1.4 Nous renvoyons à [90] pour une caractérisation des M-matrices et des H-matrices par des matrices fortement accrétives dans les espaces ℓ_p.

1.2.4 Application à la résolution de systèmes non linéaires

Nous nous intéressons à présent aux systèmes non linéaires de la forme :

$$A.X + \phi(X) = B,\ X \in \mathbb{R}^n \tag{1.27}$$

où A est une matrice de taille $n \times n$, B est un vecteur de $\mathbb{R}^n$ et ϕ une application diagonale a priori non linéaire.

Remarque 1.5 Le système (1.27) est en général issu de la discrétisation de problèmes aux limites de la forme :

$$\begin{cases} \Lambda(u) + \Lambda^d(u) \ni g,\ u \in E \\ \text{et conditions aux limites} \end{cases} \tag{1.28}$$

où E est un espace de Banach réflexif, Λ est une application univoque, et Λ^d une multi-application diagonale. Notons que de telles applications multivoques interviennent, par exemple, dans le cas où la solution u du problème (1.28) est soumise à des contraintes. L'étude de la résolution du problème général (1.28) par des algorithmes parallèles asynchrones classiques a été effectuée par J.C Miellou et P. Spitéri dans [88, 73]. Les notions de H-accrétivité [66, 26] et de m-accrétivité [11] y interviennent notamment.

Afin d'envisager la résolution du système (1.27) à l'aide des algorithmes parallèles asynchrones classiques (1.10), nous allons considérer sa décomposition par blocs dans $\mathbb{R}^n = \prod_{i=1}^{\beta} \mathbb{R}^{\nu(i)}$:

$$\forall i \in \{1, \ldots, \beta\},\ A_{ii}.x_i + \phi_i(x_i) + \sum_{j \neq i} A_{ij}.x_j = b_i. \tag{1.29}$$

Soient $\mathcal{S}$ une stratégie et $\mathcal{R}$ une suite de retards. L'algorithme de point fixe par blocs dédié à la résolution du système (1.27) est construit implicitement comme suit :

$$\forall i \in s(p),\ x_i^{p+1} = F_i(\tilde{X}^p) \tag{1.30}$$

où x_i^{p+1} vérifie :

$$A_{ii}.x_i^{p+1} + \phi_i(x_i^{p+1}) + \sum_{j \neq i} A_{ij}.\tilde{x}_j^p = b_i. \tag{1.31}$$

Le vecteur $\tilde{X}^p$ représente les valeurs disponibles du vecteur itéré lors de la $p^{\text{ème}}$ itération, conformément à la notation (1.11).

Proposition 1.4
Sous les hypothèses et notation suivantes :

1. $\forall i \in \{1, \ldots, \beta\}$, *la sous-matrice bloc diagonale A_{ii} de la matrice A est fortement accrétive, la constante d'accrétivité étant m_{ii} ($m_{ii} > 0$),*

2. $\forall (i,j) \in \{1,\ldots,\beta\}^2,\ i \neq j$, *soit* m_{ij} *la norme matricielle du bloc* A_{ij},

3. $\forall i \in \{1,\ldots,\beta\}$, ϕ_i *est une application croissante,*

4. *la matrice* J *de coefficients diagonaux nuls et hors diagonaux égaux à* $\frac{m_{ij}}{m_{ii}}$ *est une matrice de contraction,*

les algorithmes parallèles asynchrones classiques (1.10) associés à la décomposition par blocs du système (1.27) convergent vers X^**, solution de ce système.*

Ce résultat de convergence est démontré dans [88, 73, 49]. La preuve consiste à vérifier les hypothèses du théorème 1.1.

Remarque 1.6 Notons que si une multi-application est croissante, alors elle est accrétive (*cf.* [88, 49] pour la démonstration).

Remarque 1.7 En pratique, il suffit de vérifier que la matrice J a un rayon spectral strictement inférieur à 1, puisque J est non négative. Cette condition sera vérifiée si par exemple, la matrice $\bar{M}$ de coefficients diagonaux m_{ii} et hors diagonaux $-m_{ij}$ est une M-matrice [88, 73].

Autres critères de convergence

Lorsque le nombre de blocs β est supérieur au nombre de processeurs α, nous nous ramenons à une décomposition plus grossière du problème en α grands blocs ($\alpha < \beta$). Ces derniers sont composés de blocs adjacents de la décomposition initiale. Ce type d'algorithme est appelé algorithme parallèle synchrone ou asynchrone *associé à la décomposition en sous-domaines* du problème (1.27). Sa formulation et l'analyse de sa convergence est traitée dans [88, 73]. Dans ces références, il est en particulier démontré que si les hypothèses de la proposition 1.4 sont vérifiées pour une décomposition en β blocs, alors la convergence est assurée pour une décomposition plus grossière. En complément à la proposition 1.4, nous pouvons énoncer le corollaire suivant :

Corollaire 1.4 *Sous les hypothèses de la proposition 1.4, les algorithmes parallèles asynchrones classiques (1.10) associés à la décomposition en sous-domaines du système (1.27) convergent vers* X^**, la solution de ce système.*

La conséquence de ce corollaire est d'une importance majeure. Si les hypothèses de la proposition 1.4 sont vérifiées pour la décomposition par points, qui correspond à la décomposition la plus fine, alors la convergence est assurée pour n'importe quelle décomposition en sous-domaines. Autrement dit, l'étude de la décomposition par points suffit pour établir la convergence pour toute autre décomposition. Le critère de convergence suivant [88, 73] découle de l'application du corollaire 1.4 :

Proposition 1.5
Si ϕ est un opérateur diagonal croissant et si A est une M-matrice, alors les algorithmes parallèles asynchrones classiques (1.10) associés à la décomposition par blocs du système (1.27) convergent vers X^*, solution du système.

Exemple d'application

Exemple 1.1 Considérons une équation de Poisson perturbée par un opérateur diagonal non linéaire, sur le domaine $\Omega = [0,1]^3$.

$$\begin{cases} -\Delta u + \exp(u) = f \text{ dans } \Omega \\ u_{/\partial\Omega} = 0 \end{cases} . \tag{1.32}$$

La discrétisation de (1.32) par une méthode de différences finies classique conduit à la résolution d'un système algébrique de la forme :

$$A.X + \phi(X) = B,\ X \in \mathbb{R}^n \tag{1.33}$$

où A est la matrice de discrétisation du Laplacien. En choisissant un pas de discrétisation uniforme, A est de la forme :

$$A = \frac{1}{h^2} \begin{pmatrix} C & -I & & & \\ -I & C & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & C & -I \\ & & & -I & C \end{pmatrix} \tag{1.34}$$

où I est l'identité et C est une sous-matrice de la forme :

$$C = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix} \tag{1.35}$$

et T est une sous-matrice de la forme :

$$T = \begin{pmatrix} 6 & -1 & & & \\ -1 & 6 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 6 & -1 \\ & & & -1 & 6 \end{pmatrix} . \tag{1.36}$$

Les matrices T, C et A sont respectivement de de rang ν, ν^2 et $n = \nu^3$. Nous considérons une décomposition en blocs de taille ν :

$$\frac{1}{h^2} T.x_i + \exp(x_i) = b_i - \frac{1}{h^2}(-x_{i-\nu} - x_{i-1} - x_{i+1} - x_{i+\nu}). \tag{1.37}$$

Munissons $\mathbb{R}^n$ de la norme $\|\,.\,\|_\infty$, et vérifions les hypothèses de la proposition 1.4.

1. Les sous-matrices blocs diagonales de A sont fortement accrétives car T étant diagonale dominante stricte, elle est, d'après la proposition 1.3, fortement accrétive. En l'occurrence, les constantes d'accrétivité valent $m_{ii} = \frac{4}{h^2}$.
2. Les normes matricielles des blocs hors-diagonaux valent 0 ou $\frac{1}{h^2}$ car ces blocs sont soit nuls, soit égaux à $-\frac{1}{h^2}I$. Notamment : $m_{i,i-\nu} = m_{i,i-1} = m_{i,i+1} = m_{i,i+\nu} = \frac{1}{h^2}$.
3. La fonction exponentielle est croissante.
4. La matrice $\bar{M}$ de coefficients diagonaux m_{ii} et hors-diagonaux $-m_{ij}$, $j \neq i$, est une M-matrice. D'après la remarque 1.7, on a bien $\rho(J) < 1$.

De plus, sachant que A est elle-même une M-matrice, l'application de la proposition 1.5 est également possible si une décomposition par points est considérée, ce qui entraîne la convergence des itérations (1.10) quelque soit la décomposition considérée du problème.

1.3 Communication flexible et ordre partiel

1.3.1 Position du problème

Commençons par présenter le contexte mathématique dans lequel J.C Miellou, D. El Baz et P. Spitéri ont étudié les algorithmes parallèles asynchrones avec communication flexible par des techniques d'ordre partiel [71]. Dans la suite, nous nous positionnerons dans l'espace vectoriel $\mathbb{R}^n$, considéré comme un produit de β espaces :

$$E = \mathbb{R}^n = \prod_{i=1}^{\beta} \mathbb{R}^{\nu(i)}. \tag{1.38}$$

Les vecteurs et les applications seront décomposés selon (1.2) et (1.5).

Soit Λ une application de $\mathbb{R}^n$ à valeurs dans $\mathbb{R}^n$. Nous nous intéressons alors à la résolution du système

$$\Lambda(X) = 0. \tag{1.39}$$

Soit $W \in \mathbb{R}^n$. Le $i^{\text{ème}}$ sous-problème associé au système (1.39) décomposé en β blocs s'écrit comme suit :

$$\Lambda_i(w_1, \ldots, w_{i-1}, x_i, w_{i+1}, \ldots, w_\beta) = 0 \tag{1.40}$$

où $x_i \in \mathbb{R}^{\nu(i)}$ est l'inconnue. Par souci de commodité, adoptons la notation suivante :

$$\Lambda_i(x_i; W) = \Lambda_i(w_1, \ldots, w_{i-1}, x_i, w_{i+1}, \ldots, w_\beta).$$

Remarque 1.8 L'interprétation de l'équation (1.40) est analogue à celle de l'équation (1.31). Toutefois, les algorithmes considérés dans chaque cas ne sont pas identiques. Le paramètre W de l'équation $\Lambda_i(x_i; W) = 0$ représente les valeurs disponibles des autres composantes du vecteur itéré.

À présent, nous nous intéressons à :
- l'existence d'une solution unique au système (1.39),
- une formulation équivalente du système (1.39) par une équation de point fixe $X = F(X)$, où F est une application monotone.

Nous rappellerons pour cela la notion de *M-fonction* selon W.C. Rheinboldt [76, 79].

Définition 1.8 (M-fonction)
Soit Λ une application de $\mathbb{R}^n$ à valeurs dans $\mathbb{R}^n$ et soient $(\Lambda_i)_{1\leq i\leq n}$ ses composantes ($\Lambda_i : \mathbb{R}^n \to \mathbb{R}$). On note $(e_j)_{1\leq j\leq n}$ les vecteurs de la base canonique de $\mathbb{R}^n$. Λ est une M-fonction si :
- *pour tout $X \in \mathbb{R}^n$ et pour tout $(i, j) \in \{1, \ldots, n\}^2$ tel que $i \neq j$, les applications définies comme suit :*

$$\begin{aligned} \Lambda_{ij} : \mathbb{R} &\to \mathbb{R} \\ t &\mapsto \Lambda_{ij}(t) = \Lambda_i(X + t\, e_j) \end{aligned} \tag{1.41}$$

  sont décroissantes,
- *si Λ est inverse monotone ; c'est à dire :*

$$\forall (X, Y) \in \mathbb{R}^n \times \mathbb{R}^n,\ \Lambda(X) \leq \Lambda(Y) \Longrightarrow X \leq Y. \tag{1.42}$$

L'étude des algorithmes parallèles asynchrones avec communication flexible par une technique d'ordre partiel sera effectuée dans le cadre d'applications Λ vérifiant l'hypothèse suivante :

Hypothèse 1.1 *$\Lambda : \mathbb{R}^n \to \mathbb{R}^n$ est une M-fonction continue et surjective*

Il est possible de caractériser la surjectivité des M-fonctions continues à l'aide de la notion de coercivité pour l'ordre.

Définition 1.9 (Coercivité pour l'ordre)
Une application $\Lambda : \mathbb{R}^n \to \mathbb{R}^n$ est dite coercive pour l'ordre si pour toute suite $(X^p)_{p\in\mathbb{N}}$:
- *$\forall p \in \mathbb{N}$, $X^p \leq X^{p+1}$ et $\lim_{p\to+\infty} X^p = +\infty$ implique $\lim_{p\to+\infty} \Lambda(X^p) = +\infty$;*
- *$\forall p \in \mathbb{N}$, $X^p \geq X^{p+1}$ et $\lim_{p\to+\infty} X^p = -\infty$ implique $\lim_{p\to+\infty} \Lambda(X^p) = -\infty$.*

La notation $\lim_{p\to+\infty} X^p = +\infty$ (resp. $-\infty$) signifie qu'il existe un indice i ($1 \leq i \leq n$) pour lequel la suite réelle $(x_i^p)_{p\in\mathbb{N}}$ tend vers $+\infty$ (resp. $-\infty$).

D'après [79], une M-fonction continue est surjective si et seulement si elle est coercive pour l'ordre.

Sous l'hypothèse 1.1, le système (1.39) et les sous-problèmes (1.40) admettent une solution unique [70]. Soient alors $(z_i)_{1\leq i\leq\beta}$ les solutions respectives des équations $\Lambda_i(x_i; W) = 0$. L'application de point fixe est alors définie comme suit :

$$\forall i \in \{1, \ldots, \beta\},\ F_i(W) = z_i \text{ avec } \Lambda_i(z_i; W) = 0 \tag{1.43}$$

et elle vérifie $\Lambda(X^*) = 0 \iff X^* = F(X^*)$. On montre aussi que de F est monotone d'après l'hypothèse 1.1 [71].

1.3.2 Modèle mathématique

Dans les définitions qui vont suivre, Λ désignera une M-fonction continue surjective. Soit F l'application de point fixe associée à $\Lambda(X) = 0$. On notera $\mathbb{R}^n_+$ le cône des vecteurs positifs de $\mathbb{R}^n$.

Définition 1.10 (Segment d'ordre)
Soient X et Y, deux vecteurs de $\mathbb{R}^n$ tels que $X \leq Y$. Le segment d'ordre $[X\,,\,Y]$ est l'ensemble :

$$[X\,,\,Y] = \{Z \in \mathbb{R}^n \mid X \leq Z \leq Y\}. \tag{1.44}$$

Le segment d'ordre $[x_i\,,\,y_i]_i$ dans $\mathbb{R}^{\nu(i)}$ est défini de manière analogue.

Définition 1.11 (Λ-sur-solution)
Soit $X \in \mathbb{R}^n_+$. X est une Λ-sur-solution si

$$\Lambda(X) \geq 0. \tag{1.45}$$

Remarque 1.9 D'après [71], si X est une Λ-sur-solution, alors

$$\forall i \in \{1, \ldots, \beta\},\ F_i(X) \leq x_i. \tag{1.46}$$

Ceci permet de définir les segments d'ordre $[F_i(X)\,,\,x_i]_i$, lorsque X est une Λ-sur-solution.

Définition 1.12 (Λ-sur-application)
Soit F^Λ une application à valeurs dans $\mathbb{R}^n$ dont les composantes sont $(F_i^\Lambda)_{1\leq i\leq\beta}$, de domaines de définition respectifs :

$$\mathcal{D}(F_i^\Lambda) = \{X \in \mathbb{R}^n \mid \Lambda_i(X) \geq 0\}.$$

F^Λ est une Λ-sur-application associée à F si $\forall i \in \{1, \ldots, \beta\}$, $\forall X \in \mathcal{D}(F_i^\Lambda)$:

$$F_i^\Lambda(X) \leq x_i \tag{1.47}$$

$$\Lambda_i(F_i^\Lambda(X); X) \geq 0 \tag{1.48}$$

$$F_i(X) \neq x_i \implies F_i^\Lambda(X) \neq x_i. \tag{1.49}$$

Interprétation des Λ-sur-applications La Λ-sur-application F^Λ permet d'associer au problème (1.39) à résoudre une application de point fixe approchée de l'application F. En effet, on montre que pour une Λ-sur-application donnée, les conditions (1.47) et (1.48) impliquent[1] $F_i^\Lambda(X) \in [F_i(X)\,,\, x_i]_i$ [71]. De plus, d'après la relation (1.49), si $\bar{X}$ est tel que pour tout i, $F_i^\Lambda(\bar{X}) = \bar{x}_i$, alors $\bar{X}$ est point fixe de F.

Nous reprenons à présent les notions de stratégie et de suite de retards (définitions 1.1 et 1.2), avec quelques changements dus au modèle avec ordre partiel :

$$\forall p \in \mathbb{N},\ s(p) \in \{1, \ldots, \beta\} \tag{1.50}$$

$$\forall p \in \mathbb{N},\ \rho_{s(p)}(p) = p. \tag{1.51}$$

Remarque 1.10 Ici, $s(p)$ devient un singleton. La condition (1.51) exprime le fait que chaque processeur accède à ses propres données sans être pénalisé par des retards.

Ces changements ne restreignent pas la portée du modèle. Notons que les conditions (1.6), (1.8) et (1.9) sont toujours vérifiées.

Pour tout $i \in \{1, \ldots, \beta\}$ et $p \in \mathbb{N}$, nous noterons

$$K_i^p = \{k \in \mathbb{N} \mid s(k) = i, 0 \leq k < p\} \tag{1.52}$$

l'ensemble de tous les numéros d'itération inférieurs à p, pour lesquels la $i^{\text{ème}}$ composante du vecteur itéré a été réactualisée. Compte tenu de ces définitions, les algorithmes parallèles asynchrones avec communication flexible peuvent se formuler comme suit [71] :

Définition 1.13 (Modèle asynchrone avec communication flexible)
Soit Λ une application vérifiant l'hypothèse 1.1. Soient X^0 une Λ-sur-solution, F^Λ une Λ-sur-application, $\mathcal{S}$ une stratégie et $\mathcal{R}$ une suite de retards. Un algorithme parallèle asynchrone avec communication flexible construit récursivement une suite d'itérés $(X^p)_{p\in\mathbb{N}}$ de la manière suivante :

$$\forall p \geq 0,\ \forall i \in \{1, \ldots, \beta\},\ x_i^{p+1} = \begin{cases} x_i^p & si \quad i \neq s(p) \\ F_i^\Lambda(\tilde{X}^p) & si \quad i = s(p) \end{cases} \tag{1.53}$$

où $(\tilde{X}^p)_{p\in\mathbb{N}}$ est une suite récursive telle que

$$\tilde{X}^0 = X^0 \tag{1.54}$$

$$\forall p \geq 1,\ \tilde{X}^p \in [X^p\,,\, \min(X^{\rho(p)}, \tilde{X}^q)] \tag{1.55}$$

avec $q = \max K_{s(p)}^p$. Et lorsque $K_{s(p)}^p = \emptyset$, on a :

$$\tilde{X}^p \in [X^p\,,\, X^{\rho(p)}]. \tag{1.56}$$

[1] La réciproque est fausse.

Interprétation de $(\tilde{X}^p)_{p\in\mathbb{N}}$ Considérons le calcul de la composante $x_{s(p)}^{p+1}$. Le vecteur $\tilde{X}^p$ représente les valeurs disponibles du vecteur itéré. Chaque composante $\tilde{x}_i^p$ est choisie dans le segment d'ordre $[x_i^p\,,\,\min(x_i^{\rho_i(p)},\tilde{x}_i^q)]_i$ où $\tilde{x}_i^q$ est la valeur utilisée lors de la précédente réactualisation de la composante $x_{s(p)}$. Quant à $x_i^{\rho_i(p)}$, il introduit le non déterminisme dans le schéma itératif. Plus précisément, ce segment modélise :

- l'exploitation des données provenant de calculs en cours de réalisation,

  Autrement dit, les processeurs peuvent communiquer à tout moment la valeur courante de leur composante en cours de relaxation.

- le fait qu'une composante échangée n'est pas nécessairement associée à un numéro d'itération donné, contrairement au modèle classique (définition 1.3).

  Ainsi, tout processeur peut itérer avec les valeurs disponibles les plus récentes, même si au sein des composantes utilisées, les diverses valeurs proviennent d'itérations différentes. C'est pour cette raison que $\rho_i(p)$ est appelé le *retard maximal* associé à la $i^{\text{ème}}$ composante [71].

Ce modèle autorise plus de souplesse dans l'implémentation des communications entre les processeurs. En effet, les échanges de composantes du vecteur itéré des algorithmes parallèles asynchrones avec communication flexible sont indépendants du déroulement des calculs. Les échanges peuvent survenir à tout moment. La seule contrainte qui subsiste est la mise à l'écart des composantes les moins récentes.

Remarque 1.11 D'après [71], la suite $(X^p)_{p\in\mathbb{N}}$ construite à l'aide de l'algorithme asynchrone avec communication flexible (1.53) est décroissante. De plus, elle converge vers une Λ-sur-solution.

Remarque 1.12 Toutes les notions vues jusqu'ici peuvent être définies à l'aide de Λ-sous-solutions et de Λ-sous-applications (*cf.* [32] pour les définitions).

1.3.3 Convergence monotone

Afin d'obtenir la convergence de l'algorithme asynchrone avec communication flexible vers la solution du problème, il est nécessaire de considérer des Λ-sur-applications particulières. Elles ont été définies par J.C. Miellou, D. El Baz et P. Spitéri [71].

Définition 1.14 (Continuité pour l'ordre)
Soit $\hat{F}$ une Λ-sur-application. $\hat{F}$ est continue pour l'ordre si pour toute suite $(X^p)_{p\in\mathbb{N}}$, elle vérifie :

$$X^p \underset{p\to\infty}{\downarrow} X^* \Longrightarrow \forall i \in \{1,\ldots,\beta\},\ \hat{F}(X^p) \underset{p\to\infty}{\downarrow} \hat{F}(X^*) \tag{1.57}$$

où la notation $X^p \underset{p\to\infty}{\downarrow} X^*$ signifie

$$X^* \leq \ldots \leq X^{p+1} \leq X^p \leq \ldots \leq X^0 \text{ et } \lim_{p\to\infty} X^p = X^*.$$

Définition 1.15 (Λ-sur-application du premier type)
Soit F^Λ une Λ-sur-application. F^Λ est dite du premier type s'il existe $\hat{F}^\Lambda$, une Λ-sur-application continue pour l'ordre, vérifiant pour tout $i \in \{1, \ldots, \beta\}$:

$$\forall X \in \{X \in \mathbb{R}^n \mid \Lambda_i(X) \geq 0\},\ \hat{F}_i^\Lambda(X) \in [F_i^\Lambda(X)\,,\, x_i]_i. \tag{1.58}$$

De manière plus intuitive, une Λ-sur-application du premier type peut être considérée comme une Λ-sur-application majorée par une Λ-sur-application continue pour l'ordre.

Définition 1.16 (Λ-sur-application du second type)
Soit F^Λ une Λ-sur-application. F^Λ est dite du second type s'il existe $\delta > 0$ tel que pour tout $i \in \{1, \ldots, \beta\}$:

$$\forall X \in \{X \in \mathbb{R}^n \mid \Lambda_i(X) \geq 0\},\ |x_i - F_i^\Lambda(X)|_i \geq \delta\,|x_i - F_i(X)|_i. \tag{1.59}$$

Voici le théorème de convergence des itérations parallèles asynchrones avec communication flexible (1.53) dans un espace partiellement ordonné [71] :

Théorème 1.3 (Convergence monotone)
Soient Λ une M-fonction continue surjective, F l'application de point fixe associée définie par (1.43) et F^Λ une Λ-sur-application associée à F. Si F^Λ est du premier ou du second type alors la suite $(X^p)_{p\in\mathbb{N}}$ construite par l'algorithme parallèle asynchrone avec communication flexible (1.53)-(1.56), vérifie $X^p \underset{p\to\infty}{\downarrow} X^*$ où X^* est l'unique solution du système $\Lambda(X) = 0$.

1.3.4 Application à la résolution de systèmes non linéaires

Ramenons nous à présent à la classe d'application abordée dans la section 1.2.4. Nous avons affaire à des systèmes non-linéaires de la forme :

$$\Lambda(X) = A.X + \phi(X) - B = 0.$$

J.C. Miellou, D. El Baz et P. Spitéri ont montré dans [71, 96] que les algorithmes itératifs asynchrones avec communication flexible appliqués à ce type de problème convergent, si A est une M-matrice et si ϕ est un opérateur diagonal croissant. La Λ-sur-application correspond notamment à k itérations de l'algorithme de Newton.

Considérons l'exemple 1.1, dans lequel l'équation de diffusion non linéaire (1.32) est discrétisée. Le système algébrique obtenu est de la forme

$$A.X + \phi(X) = B.$$

On vérifie aisément que l'application $\Lambda : X \mapsto A.X + \exp(X) - B$ est une M-fonction continue car A est une M-matrice et l'exponentielle est croissante. La surjectivité découle de la coercivité pour l'ordre (définition 1.9). Nous sommes dans le

cadre d'application du théorème 1.3 avec l'algorithme de Newton en tant que Λ-sur-application du premier type [96]. Si le vecteur initial X^0 est tel que $A.X^0+\exp(X^0) \geq B$, alors la convergence monotone des algorithmes asynchrones avec communication flexible est assurée.

1.4 Communication flexible et contraction

1.4.1 Modèle mathématique

L'étude des algorithmes parallèles asynchrones par une méthode de contraction est une approche complémentaire à celle basée sur l'ordre partiel. Elle permet de s'affranchir de la contrainte consistant à choisir une Λ-sur-solution pour initialiser l'algorithme itératif. Nous présentons ici le modèle de D. El Baz, A. Frommer et P. Spitéri [33].

Dans un contexte analogue à celui de la section 1.2, pour une stratégie $\mathcal{S}$ et une suite de retards $\mathcal{R}$ données, nous recherchons le point fixe d'une contraction $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ par l'algorithme itératif asynchrone suivant :

$$\forall i \in \{1, \ldots, \beta\},\ x_i^{p+1} = \begin{cases} x_i^p & \text{si} \quad i \notin s(p) \\ F_i(\tilde{X}^p) & \text{si} \quad i \in s(p) \end{cases}$$

où $\tilde{X}^p = (x_1^{\rho_1(p)}, \ldots, x_i^{\rho_i(p)}, \ldots, x_\beta^{\rho_\beta(p)})$.

Afin de modéliser le calcul approché de $F_i(\tilde{X}^p)$ (*cf.* section 1.3), nous introduisons une suite d'applications $(G^p)_{p\in\mathbb{N}}$, dont les composantes sont G_i^p, $i = 1, \ldots, \beta$. À chaque itération, la réactualisation d'une composante du vecteur itéré s'écrit :

$$\forall p \in \mathbb{N},\ \forall i \in s(p),\ x_i^{p+1} = G_i^p(\tilde{X}^p). \tag{1.60}$$

Ce modèle d'algorithme parallèle asynchrone a été étudié dans [43] pour la résolution de problèmes linéaires.

La formulation de l'algorithme avec communication flexible doit tenir compte de la disponibilité à tout moment des valeurs courantes de toutes les composantes du vecteur itéré. Ceci a été modélisé, dans la section 1.3, à l'aide de l'appartenance des composantes à des segments d'ordre. Dans la présente étude, c'est l'appartenance à des boules emboîtées centrées en X^* qui devient la base du modèle avec communication flexible.

Hypothèse 1.2 *Il existe une norme uniforme avec poids* $\|\,.\,\|_\Gamma$ *(*cf. *équation (1.17)), un vecteur* $X^* \in \mathbb{R}^n$ *et un réel* $\delta \in [0, 1[$ *tels que :*

$$\forall p \in \mathbb{N},\ \forall X \in \mathbb{R}^n,\ \|G^p(X) - X^*\|_\Gamma \leq \delta \|X - X^*\|_\Gamma. \tag{1.61}$$

X^* *est alors le point fixe commun des* $(G^p)_{p\in\mathbb{N}}$.

Définition 1.17 (Modèle asynchrone avec communication flexible)
Soient $X^0 \in \mathbb{R}^n$ un vecteur quelconque et $(G^p)_{p\in\mathbb{N}}$ une suite d'applications vérifiant l'hypothèse 1.2. On note alors X^* le point fixe commun des G^p, $p = 0, 1, \ldots$ Soient $\mathcal{S}$ une stratégie et $\mathcal{R}$ une suite de retards. Un algorithme parallèle asynchrone avec communication flexible construit récursivement une suite d'itérés $(X^p)_{p\in\mathbb{N}}$ de la manière suivante :

$$\forall p \geq 0,\ \forall i \in \{1, \ldots, \beta\},\ x_i^{p+1} = \begin{cases} x_i^p & si \quad i \notin s(p) \\ G_i^p(\tilde{X}^p) & si \quad i \in s(p) \end{cases} . \tag{1.62}$$

Pour tout p, $\tilde{X}^p$ vérifie la contrainte :

$$\|\tilde{X}^p - X^*\|_\Gamma \leq \|X^{\rho(p)} - X^*\|_\Gamma \tag{1.63}$$

où $\|\,.\,\|_\Gamma$ est la norme uniforme avec poids intervenant dans l'hypothèse 1.2.

Interprétation du modèle Le vecteur $\tilde{X}^p$ représente les valeurs disponibles du vecteur itéré. La contrainte (1.63) combinée à l'hypothèse 1.2 exprime le fait que les réactualisations du vecteur itéré peuvent s'effectuer à l'aide de composantes en cours de réalisation et qui ne sont pas nécessairement associées à un numéro d'itération. En effet, la propriété de contraction vérifiée par les applications $(G^p)_{p\in\mathbb{N}}$ permet de générer des itérés appartenant à des boules emboîtées centrées en X^*. Seuls les itérés les plus récents sont pris en compte lors des réactualisations grâce à la contrainte (1.63), où apparaît le retard maximal $\rho(p)$.

1.4.2 Résultats de convergence

Théorème 1.4 (Convergence par contraction)
Si l'hypothèse 1.2 est vérifiée, la suite $(X^p)_{p\in\mathbb{N}}$ construite par l'algorithme parallèle asynchrone avec communication flexible (1.62) converge dans $\mathbb{R}^n$ muni de la norme $\|\,.\,\|_\Gamma$ vers X^*, point fixe commun des applications $(G^p)_{p\in\mathbb{N}}$

La preuve de ce théorème est établie dans [33].

La mise en évidence d'applications $(G^p)_{p\in\mathbb{N}}$ vérifiant l'hypothèse 1.2 est un des points essentiels abordé dans [33]. Intéressons nous au cas des algorithmes à deux niveaux d'itération, rencontrés par exemple lorsqu'on résout un grand système par la méthode de Newton ; le procédé de Newton correspond à l'itération externe et la méthode itérative pour résoudre le système linéaire correspond aux itérations internes [42]. Ce cas de figure peut se formaliser de la manière suivante :
Soit $T_{i,\tilde{X}^p} : \mathbb{R}^{\nu(i)} \to \mathbb{R}^{\nu(i)}$ la fonction d'itération interne associée à la $i^{\text{ème}}$ composante et au vecteur initial $\tilde{X}^p$. L'algorithme itératif s'écrit :

$$\begin{cases} y_i^{p,0} &= \tilde{x}_i^p \\ y_i^{p,q+1} &= T_{i,\tilde{X}^p}(y_i^{p,q}),\ q \geq 0 \end{cases} . \tag{1.64}$$

Lorsque $G_i^p(\tilde{X}^p)$ est calculé à l'aide de $q_i(p)$ itérations de l'algorithme (1.64), son expression est

$$G_i^p(\tilde{X}^p) = y_i^{p,q_i(p)}. \tag{1.65}$$

Compte tenu de ces notations, nous pouvons énoncer le lemme suivant (*cf.* [33] pour la démonstration) :

Lemme 1.1 *Si les hypothèses suivantes sont vérifiées,*

1. *Il existe $X^* \in \mathbb{R}^n$ et $\zeta \in [0,1[$ tels que pour tout $i \in \{1,\ldots,\beta\}$, F vérifie*

$$\forall X \in \mathbb{R}^n,\ |F_i(X) - X^*|_i \leq \zeta\, |x_i - x_i^*|_i. \tag{1.66}$$

2. *Pour tout $i \in \{1,\ldots,\beta\}$ et pour tout $X \in \mathbb{R}^n$, il existe $\zeta_{i,X} \in [0,1[$ tel que $T_{i,X}$ vérifie*

$$\forall y_i \in \mathbb{R}^{\nu(i)},\ |T_{i,X}(y_i) - F_i(X)|_i \leq \zeta_{i,X}\, |y_i - F_i(X)|_i. \tag{1.67}$$

alors pour tout $p \in \mathbb{N}$ et pour tout $i \in \{1,\ldots,\beta\}$:

$$\forall X \in \mathbb{R}^n,\ \frac{|G_i^p(X) - X^*|_i}{\gamma_i} \leq ((\zeta_{i,X})^{q_i(p)}(\zeta+1) + \zeta)\, \|X - X^*\|_\Gamma. \tag{1.68}$$

Si la condition suivante

$$(\zeta_{i,X})^{q_i(p)}(\zeta+1) + \zeta \leq \delta < 1$$

est vérifiée, alors l'hypothèse 1.2 sera satisfaite [33]. Ce lemme met en évidence des contraintes sur l'algorithme itératif interne permettant de garantir la convergence de l'algorithme asynchrone externe. D'une part, la convergence vers $F_i(X)$ doit être vérifiée et d'autre part, le nombre d'itérations doit être suffisant.

1.4.3 Application à la résolution de systèmes linéaires

Afin de compléter l'étude des algorithmes parallèles asynchrones avec communication flexible par une technique de contraction, nous allons présenter une condition nécessaire de convergence dans le cas de la résolution de systèmes linéaires dans $\mathbb{R}^n$ de la forme :

$$A.X = B \tag{1.69}$$

où A est une M-matrice [33]. La décomposition de ce système en β blocs s'écrit :

$$\forall i \in \{1,\ldots,\beta\},\ A_{ii}.x_i = b_i - \sum_{j \neq i} A_{ij}.x_j. \tag{1.70}$$

Chaque composante de la fonction F est alors définie comme étant la solution du $i^{\text{ème}}$ sous-problème :

$$\forall i \in \{1,\ldots,\beta\},\ F_i(\tilde{X}^p) = A_{ii}^{-1}.(b_i - \sum_{j \neq i} A_{ij}.\tilde{x}_j^p). \tag{1.71}$$

Le calcul de $F_i(\tilde{X}^p)$ est effectué par l'intermédiaire d'une méthode de relaxation. Nous considérons alors les décompositions des sous-matrices diagonales de la forme :

$$\forall i \in \{1, \ldots, \beta\},\ A_{ii} = M_i - N_i. \tag{1.72}$$

Dans la suite, seules les décompositions telles que

$$\forall i \in \{1, \ldots, \beta\},\ M_i^{-1} \geq 0 \text{ et } M_i^{-1}.N_i \geq 0 \tag{1.73}$$

seront considérées. Ces décompositions sont dites faiblement régulières [101, 76, 3]. Les algorithmes itératifs classiques tels que Jacobi et Gauss-Seidel possèdent cette particularité. En conservant les notations de (1.64), les itérations internes sont alors définies à l'aide de la fonction $T_{i,\tilde{X}^p}$ suivante :

$$T_{i,\tilde{X}^p}(y_i^{p,q}) = M_i^{-1}.(N_i.y_i^{p,q} + (b_i - \sum_{j \neq i} A_{ij}.\tilde{x}_j^p)). \tag{1.74}$$

De manière analogue à (1.65), l'approximation de $F_i(\tilde{X}^p)$ est notée :

$$\mathcal{G}_i^{q_i(p)}(\tilde{X}^p) = y_i^{p,q_i(p)}. \tag{1.75}$$

En l'occurrence, $\mathcal{G}_i^q$ représente q itérations de la méthode itérative associée à (1.74).

Proposition 1.6
Si A est une M-matrice dont les décompositions des sous-matrices diagonales (1.72) sont faiblement régulières, alors il existe $\delta \in [0, 1[$ tel que pour tout $q \geq 1$, les applications $\mathcal{G}^q$ définies en (1.75) vérifient

$$\forall X \in \mathbb{R}^n,\ \|\mathcal{G}^q(X) - X^*\|_e \leq \delta \|X - X^*\|_e \tag{1.76}$$

où $e = (1, \ldots, 1)$ et X^*, point fixe commun des $\mathcal{G}^q$, est la solution du système (1.69).

Cette proposition est démontrée dans [33]. Elle met en évidence une classe d'itération interne pour laquelle l'hypothèse 1.2 est vérifiée si au moins une itération est effectuée. La convergence des itérations parallèles asynchrones avec communication flexible est alors assurée dans $\mathbb{R}^n$ munie de la norme uniforme classique.

Exemple d'application

Exemple 1.2 De manière analogue à l'exemple 1.1, considérons une équation de Poisson linéaire sur le domaine $\Omega = [0, 1]^3$.

$$\begin{cases} -\Delta u = f \text{ dans } \Omega \\ u_{/\partial\Omega} = 0 \end{cases}. \tag{1.77}$$

En discrétisant (1.77) de la même manière que dans le cas non linéaire (voir exemple 1.1), un système linéaire $A.X = B$ est obtenu. L'expression de la matrice A est

donnée par (1.34), (1.35) et (1.36). Nous considérons une décomposition par blocs du système, similaire à la décomposition (1.37) :

$$\frac{1}{h^2}T.x_i = b_i - \frac{1}{h^2}(-x_{i-\nu} - x_{i-1} - x_{i+1} - x_{i+\nu}). \tag{1.78}$$

Afin de produire un exemple simple d'application de la proposition 1.6, nous allons résoudre (1.78) par l'algorithme de Jacobi[2]. En l'occurrence, la sous-matrice T se décompose sous la forme $T = D - N$ où D est la diagonale de T. Cette décomposition est, de manière évidente, faiblement régulière. Puisque A est une M-matrice, les hypothèses de la proposition 1.6 sont vérifiées. Par conséquent le théorème 1.4 est applicable. La résolution de (1.78) par l'algorithme de Gauss-Seidel conduit à un résultat analogue.

1.5 Lien avec la méthode de Schwarz

1.5.1 Rappels sur la méthode de Schwarz

Soit Ω un domaine tridimensionnel de $\mathbb{R}^3$. Considérons sa décomposition en β sous-domaines non vides, notés $(\Omega_i)_{1\leq i\leq\beta}$, tels que :

$$\forall i \in \{1,\ldots,\beta\},\ \Omega_i \subset \Omega \tag{1.79}$$

$$\bigcup_{i=1}^{\beta} \Omega_i = \Omega. \tag{1.80}$$

Notons adj(i) l'ensemble des sous-domaines adjacents à Ω_i, défini comme suit :

$$\text{adj}(i) = \{j \mid j \neq i \text{ et } \Omega_i \cap \Omega_j \neq \emptyset\}. \tag{1.81}$$

Soient $(\Gamma_i)_{1\leq i\leq\beta}$ les restrictions des frontières de Ω à celles des sous-domaines Ω_i

$$\forall i \in \{1,\ldots,\beta\},\ \Gamma_i = \partial\Omega_i \cap \partial\Omega. \tag{1.82}$$

Quant aux frontières entre Ω_i et ses sous-domaines adjacents, elles sont notées :

$$\gamma_i^j = \partial\Omega_i \cap \Omega_j,\ j \in \text{adj}(i). \tag{1.83}$$

Compte tenu de ces notations, nous allons présenter la méthode de Schwarz à l'aide d'un exemple.

Nous considérons une équation de Poisson sur un domaine tridimensionnel ($\Omega \subset \mathbb{R}^3$) avec des conditions aux limites de Dirichlet :

$$\begin{cases} -\Delta u = f \text{ dans } \Omega \\ u_{/\partial\Omega} = 0 \end{cases}. \tag{1.84}$$

[2]En pratique, la variante du pivot de Gauss pour les systèmes linéaires tri-diagonaux convient mieux.

La méthode de Schwarz est un algorithme itératif. Chaque itération consiste à résoudre β sous-problèmes, de la même forme que le problème initial (1.84), définis sur les sous-domaines $(\Omega_i)_{1\leq i\leq\beta}$. Les conditions aux limites des sous-problèmes découlent de la décomposition en sous-domaines. Soient

$$(f_1,\ldots,f_i,\ldots,f_\beta)$$

les restrictions du second membre sur chacun des sous-domaines $(f_{/\Omega_i})$. La suite de vecteurs itérés engendrée par l'algorithme de Schwarz est notée

$$(u_1^p,\ldots,u_i^p,\ldots,u_\beta^p)_{p\in\mathbb{N}}$$

où u_i^p est une fonction définie sur Ω_i. Chaque itération est définie comme suit :

$$\forall p\in\mathbb{N},\ \forall i\in\{1,\ldots,\beta\},\ \left\{\begin{array}{l} -\Delta u_i^{p+1}=f_i \text{ dans } \Omega_i \\ u_{i/\Gamma_i}^{p+1}=0 \\ u_{i/\gamma_i^j}^{p+1}=u_{j/\gamma_i^j}^{p},\ \text{ pour tout } j\in \text{adj}(i) \end{array}\right. . \tag{1.85}$$

La méthode de Schwarz est bien adaptée au parallélisme [56]. Chaque sous-problème associé à une itération de l'algorithme est résolu indépendamment des autres. L'introduction du parallélisme asynchrone dans cet algorithme est techniquement très simple. Ré-écrivons (1.85) en y incorporant une stratégie $\mathcal{S}$ et d'une suite de retards $\mathcal{R}$:

$$\forall p\in\mathbb{N},\ \left\{\begin{array}{l} i\in s(p)\Longrightarrow \left\{\begin{array}{l} -\Delta u_i^{p+1}=f_i \text{ dans } \Omega_i \\ u_{i/\Gamma_i}^{p+1}=0 \\ u_{i/\gamma_i^j}^{p+1}=\tilde{u}^p_{j/\gamma_i^j},\ \text{ pour tout } j\in \text{adj}(i) \end{array}\right. \\ i\notin s(p)\Longrightarrow u_i^{p+1}=u_i^p \end{array}\right. \tag{1.86}$$

où $\forall j\in\text{adj}(i)$, $\tilde{u}_j^p$ est une application définie sur Ω_j dont l'expression dépend de $\rho_j(p)$ et du type d'algorithme asynchrone choisi :

1. $\tilde{u}_j^p=u_j^{\rho_j(p)}$ dans le cas des algorithmes asynchrones classiques (définition 1.3),
2. $\tilde{u}_j^p\in[u_j^p,\min(u_j^{\rho_j(p)},\tilde{u}_j^q)]$ où $q=\max K_i^p$ dans le cas des algorithmes asynchrones avec communication flexible et convergence monotone (définition 1.13),
3. $\tilde{u}_j^p$ est tel que $\frac{|\tilde{u}_j^p-u_j^*|_j}{\lambda_j}\leq\max\limits_{1\leq i\leq\beta}\frac{|u_i^{\rho_i(p)}-u_i^*|_i}{\lambda_i}$ dans le cas des algorithmes asynchrones avec communication flexible et convergence par contraction (définition 1.17).

Enfin, la notion d'opérateur approché propre aux algorithmes avec communication flexible, intervient lorsque les sous-problèmes $-\Delta u_i^{p+1}=f_i$ sont résolus par des méthodes itératives.

1.5.2 Application des algorithmes parallèles asynchrones

Il faut à présent établir un lien entre la méthode de Schwarz asynchrone (1.86) et les modèles d'algorithmes parallèles asynchrones. La discrétisation des β sous-problèmes de l'algorithme de Schwarz classique (1.85) conduit à la résolution de systèmes algébriques couplés. Dans le but d'appliquer les critères de convergence, il est nécessaire de transformer les matrices de manière à former un système augmenté

$$\hat{A}.\hat{X} = \hat{B} \tag{1.87}$$

L'écriture explicite du système augmenté est techniquement difficile dans le cas tridimensionnel. Nous nous contentons ici d'en donner les principes.

1. Pour les points intérieurs de Ω_i, l'équation $-\Delta u_i^{p+1} = f_i$ discrétisée conduit à $A_i^0.\hat{x}_i^{p+1} = b_i^0$. À ce stade, les paramètres concernant les conditions aux limites et le couplage avec les sous-domaines adjacents sont absents du système.
2. Les conditions aux limites sur Γ_i sont exprimées en complétant le second membre b_i^0. Les coefficients correspondants aux points de discrétisation appartenant à Γ_i prennent les valeurs définies par la condition de Dirichlet. Les lignes de la matrice A_i^0 associées à ces coefficients valent 1 sur la diagonale, 0 ailleurs. A_i^1 et b_i^1 sont ainsi obtenus.
3. Le couplage avec les sous-domaines adjacents est effectué en 2 étapes. Considérons les points de discrétisation situés sur $\bigcup_{j\in \mathrm{adj}(i)} \gamma_i^j$.
   (a) A_i^1 et b_i^1 subissent une première transformation, analogue à la précédente. Au niveau des points considérés, b_i^1 reçoit la valeur 0. Le traitement de la matrice reste le même. A ce stade, nous obtenons les sous-matrices diagonales $\hat{A}_{ii}$ et les blocs $\hat{b}_i$ du second membre du système augmenté.
   (b) Il reste à définir les blocs hors-diagonaux $\hat{A}_{ij}$. Ces blocs permettent de coupler les composantes i et j du vecteur itéré. Soit $j \neq i$.
      i. Si $j \notin \mathrm{adj}(i)$, alors le bloc $\hat{A}_{ij}$ est nul.
      ii. Dans le cas où $j \in \mathrm{adj}(i)$, considérons les points de discrétisation associés à γ_i^j. La mise en relation des points de $\gamma_i^j \subset \Omega_i$ avec leurs analogues dans Ω_j consiste alors à positionner les coefficients adéquats de $\hat{A}_{ij}$ à -1 ; les autres coefficients étant nuls.

Ce procédé permet de reformuler la méthode de Schwarz (1.85) comme la résolution d'un système algébrique à l'aide d'un algorithme itératif par blocs :

$$\forall p \in \mathbb{N},\ \forall i \in \{1, \ldots, \beta\},\ \hat{A}_{ii}.\hat{x}_i^{p+1} = \hat{b}_i - \sum_{j\neq i} \hat{A}_{ij}.\hat{x}_j^p. \tag{1.88}$$

L'analyse de la convergence de l'algorithme de Schwarz parallèle asynchrone, avec ou sans communication flexible, peut alors se ramener à l'étude de la résolution par blocs du système augmenté (1.88) par les itérations parallèles asynchrones.

D'après [39], si la matrice de discrétisation du problème initial est une M-matrice, alors la matrice $\hat{A}$ résultant du procédé d'augmentation de la méthode de Schwarz reste une M-matrice.

De plus, en présence d'une perturbation par un opérateur diagonal croissant (équation (1.32) par exemple), la raisonnement précédent permet d'aboutir à un système algébrique augmenté où intervient une M-fonction, somme d'une M-matrice et d'une application diagonale croissante. Dans ces conditions, les résultats de convergence en ordre partiel (voir section 1.3.3) s'appliquent.

Chapitre 2

Mise en œuvre des algorithmes parallèles asynchrones

2.1 Introduction

Les premières implémentations des algorithmes parallèles asynchrones ont été réalisées dès 1967 par J.L. Rosenfeld [83] qui s'est intéressé à la simulation d'exécution de code en parallèle. Notons que ces simulations ont été réalisées avant la première analyse de convergence établie en 1969 [21]. Suite aux analyses de convergence dans le cas non linéaire [28, 67, 82], G. Baudet a effectué la première étude de performance sur calculateur parallèle [10]. Notamment, une comparaison entre les performances des méthodes itératives synchrones et asynchrones a mis en évidence les avantages de l'asynchronisme. À partir de 1980, des simulations d'exécution de code en parallèle ont été réalisées par J. Julliand, G.R. Perrin et P. Spitéri [61, 87] pour simuler le comportement des itérations parallèles synchrones et asynchrones sur divers types d'architecture.

Par la suite, de nombreuses études ont permis de confirmer les gains de performance obtenus sur calculateur parallèle grâce à l'asynchronisme, pour la résolution de problèmes mathématiques très variés :

- les chaînes de Markov [13, 30],
- le problème de l'obstacle [12, 95],
- la programmation dynamique [100],
- les problèmes d'optimisation et de flot [18, 29, 34],
- les équations aux dérivées partielles [48, 50, 51, 53],
- les problèmes de commande optimale [59].

Malgré l'évolution des architectures parallèles et des méthodes d'implémentation des algorithmes parallèles, l'intérêt des algorithmes asynchrones a toujours été validé par des résultats expérimentaux.

Le travail d'implémentation réalisé dans le cadre de cette étude s'inscrit dans la continuité de la thèse de R. Guivarch [51], dans la mesure où des domaines

tridimensionnels sont considérés. Au cours de ce chapitre, nous nous intéresserons à la mise en œuvre séquentielle, synchrone et asynchrone de l'algorithme de Schwarz décrit dans la section 1.5. En particulier, nous mettrons l'accent sur l'implémentation du concept de communication flexible [71, 33].

Les algorithmes ont été implémentés dans l'optique de résoudre des équations aux dérivées partielles dans des domaines tridimensionnels. Les architectures cibles sont des calculateurs parallèles à mémoire distribuée. Les algorithmes parallèles ont été implémentés à l'aide d'une bibliothèque standard d'échange de message : MPI [63, 64]. L'étude de performance dans le cas tridimensionnel a pour objectif de valider le gain de performance obtenu grâce à l'asynchronisme, dans un contexte où le surcoût engendré par les communications est plus important que dans les cas unidimensionnels et bidimensionnels. Nous serons confrontés à certaines difficultés, en particulier :

– l'augmentation de la taille des messages,
– les complications de nature géométrique qui affectent la structure des messages.

Ce chapitre s'organise de la manière suivante. Nous présenterons dans un premier temps les concepts élémentaires sur les calculateurs et les programmes parallèles. Puis nous présenterons l'implémentation des algorithmes de Schwarz parallèles asynchrones classiques (*cf.* section 1.2) avant d'étudier celle des méthodes de Schwarz avec communication flexible. Quelques difficultés algorithmiques liées à l'implémentation de ces méthodes seront soulevées. Pour finir, des résultats expérimentaux seront donnés afin de comparer et d'analyser les performances des algorithmes synchrones et asynchrones dans le cadre de problèmes classiques.

2.2 Programmation sur calculateur parallèle

2.2.1 Notions sur les architectures parallèles

Les architectures les plus simples de calculateurs parallèles peuvent être classées dans les catégories suivantes.

1. Les processeurs vectoriels, qui sont conçus pour exécuter simultanément la même instruction sur plusieurs données.
2. Les architectures SMP (*Symmetrical MultiProcessing*), qui sont formés de plusieurs processeurs identiques fonctionnant en parallèle et travaillant avec le même espace mémoire.
3. Les architectures AMP (*Asymmetrical MultiProcessing*), qui sont formés de processeurs de natures différentes fonctionnant en parallèle et travaillant avec le même espace mémoire. Chaque processeur est dédié à la réalisation d'une tâche spécifique.
4. Les *clusters*, qui sont formés de calculateurs indépendants interconnectés par un réseau de communication. Ils ne partagent donc pas de mémoire commune.

Chaque calculateur faisant partie du cluster est appelé un nœud. Les nœuds d'un cluster sont indépendants au sens où chaque nœud est un calculateur à part entière, capable de fonctionner de façon autonome.

Notons que sur les architectures SMP et AMP, le noyau du système d'exploitation supervise l'ensemble des processeurs. Tandis que sur un cluster, la supervision de l'ensemble des nœud est prise en charge par des applications réparties.

Exemples d'architectures parallèles

La différence principale entre les processeurs vectoriels et les processeurs classiques (dits *scalaires*) réside dans la conception des pipelines. Plusieurs pipelines sont dédiés à chaque instruction vectorielle, de sorte qu'une opération sur un tableau puisse être découpée en plusieurs opérations simultanées sur des sous-tableaux. Chaque sous-tableau est traité par un pipeline. Par conséquent, la seconde différence par rapport aux processeurs scalaires réside dans la bande passante entre la mémoire et le processeur, qui doit être adaptée à la puissance crête, de manière à pouvoir fournir suffisamment de données au processeur. Par exemple, la bande passante entre processeur et mémoire des calculateurs *NEC SX5* est de 64 Go/s, compte tenu du fait que chaque pipeline associé à une instruction vectorielle existe en 16 exemplaires. Les processeurs scalaires actuels atteignent une dizaine de Go/s. Notons que le calculateur cité ci-dessus est une machine de type SMP formé de processeurs vectoriels.

Un processeur *multicore*, qui résulte de l'association de plusieurs processeurs possédant leur propre mémoire cache sur un seul circuit intégré, fait partie des architectures SMP. Les processeurs qui sont associés partagent la mémoire centrale et éventuellement une partie de la mémoire cache. Actuellement, peu de fabriquants produisent des processeurs multicore possédant plus de 2 processeurs (appelés *dual-core*) et aucun processeur multicore possédant plus de 4 processeurs existe sur le marché.

Les calculateurs possédant plusieurs processeurs connectés à une unique carte mère sont des machines de type SMP. Les processeurs accèdent tous à la même mémoire. Ce type d'architecture permet d'agréger plus facilement un nombre élevé de processeurs par rapport à une architecture multicore. Toutefois, ce nombre est limité par des contraintes matérielles. En effet, en ce qui concerne les accès mémoire, la bande passante est partagée par tous les processeurs. Plus ces derniers seront nombreux, plus les transferts de donnée entre la mémoire centrale et les mémoires caches des processeurs seront lents. À titre indicatif, un *SGI Altix 3700 Bx* peut contenir jusqu'à 256 processeurs.

Le *Cray T3E*[1] possède une architecture mieux adaptée au parallélisme de masse (plus de 1000 processeurs) que la précédente. C'est un assemblage de processeurs connectés par un réseau. Ce n'est toutefois pas un cluster car chaque nœud ne contient

[1]Ce modèle est en train de disparaître.

qu'un processeur, une mémoire et une interface réseau. De plus, seuls quelques processeurs sont dédiés à l'exécution du système d'exploitation. Les nœuds ne sont donc pas indépendants.

Un exemple simple de calculateur AMP : le PC multimédia qui possède en plus du processeur central, une carte graphique et une carte son, ayant chacune un processeur spécialisé.

Un ensemble de stations de travail connectées en réseau local et possédant les logiciels et les bibliothèques permettant d'exécuter des codes parallèles est un cluster.

Remarque 2.1 Les expérimentations qui ont été menées dans le cadre de cette étude sont effectuées sur un IBM Power 4 p690+. Cette architecture est relativement complexe car elle résulte d'un couplage entre les différentes techniques évoquées jusqu'ici. Nous renvoyons à la page 75 pour sa description.

2.2.2 Bases de la programmation d'algorithmes parallèles

La parallélisation consiste à décomposer un problème en sous-problèmes qui seront résolus en parallèle. Chaque sous-problème est affecté à un ou plusieurs processeurs. La *vectorisation* d'un programme (l'adaptation d'un programme séquentiel pour exploiter un processeur vectoriel) ne rentre pas dans le cadre de cet exposé.

On distingue de manière générale 2 méthodes de décomposition :

1. la décomposition de domaine qui consiste à décomposer les données en plusieurs parties ;
2. la décomposition des tâches qui consiste à décomposer le problème global en plusieurs tâches.

Les 2 méthodes peuvent être combinées dans le but d'extraire le maximum de parallélisme. On peut par exemple décomposer un programme séquentiel en tâches dont les données sont également décomposées en plusieurs parties. Chaque tâche est donc destinée à être exécutée sur un calculateur parallèle et les tâches indépendantes seront également exécutées en parallèle.

La programmation des algorithmes parallèles est abordée à l'aide de modèles de programmation. Les 3 modèles classiques de programmation parallèle sont fondés sur les modalités d'accès aux données des autres processeurs.

1. Le modèle à mémoire partagée : les processeurs n'ont pas besoin de communiquer explicitement pour s'échanger des données puisqu'ils ont tous accès directement à une mémoire commune.
2. Le modèle par échange de message : chaque processeur travaille avec une mémoire locale privée. Par conséquent, des échanges de message sont nécessaires pour accéder à des données distantes. Lors d'un échange de message entre 2 processeurs, ces derniers doivent respectivement faire appel à des fonctions d'envoi et de réception de donnée. Autrement dit, les échanges de donnée sont bilatéraux.

3. Le modèle à mémoire virtuellement partagée : la mémoire, bien que physiquement distribuée, est considérée comme partagée. La lecture et l'écriture de donnée dans une mémoire distante se fait par l'intermédiaire de sous-programmes dédiés qui masquent les échanges de message. Contrairement au modèle précédent, les échanges sont unilatéraux.

Les modèles à mémoire partagée et virtuellement partagée exigent de la part du programmeur la mise en œuvre de synchronisations entre les processeurs, afin de garantir la validité des données lues ou écrites. Dans le modèle par échange de message, les processeurs se synchronisent lorsqu'ils attendent la réception d'un message.

2.2.3 Concepts de base associés à MPI

MPI [63] (*Message-Passing Interface*) est une bibliothèque standardisée, dédiée à la mise en œuvre d'applications parallèles reposant sur le modèle par échange de message. Le standard définit uniquement la syntaxe et la sémantique, dans les langages Fortran et C, de fonctions permettant d'exploiter divers types d'architectures parallèles. Les implémentations du standard sont réalisées par les constructeurs de supercalculateur ou fournis par des contributeurs indépendants dans le cas des clusters de stations de travail (MPICH, LAM). L'avantage majeur du standard MPI est la portabilité car les spécifications des fonctions sont indépendantes des architectures. Il est devenu incontournable car il est implémenté sur tous les supercalculateurs actuels. Cependant, MPI ne prend pas en compte la gestion dynamique des tâches. Le nombre de processeurs utilisés, défini au lancement du programme, ne peut être modifié pendant l'exécution.

Le standard MPI a été étendu (MPI-2 [64]) de manière à prendre en compte le modèle à mémoire virtuellement partagée et la création dynamique de tâches. De plus, les fonctions de la bibliothèque sont disponibles en C++. Cependant, cette extension n'a pas été largement implémentée par les constructeurs.

Voici les notions nécessaires à l'implémentation des itérations parallèles synchrones et asynchrones.

Processeurs et mémoires

Un processus est un ensemble de ressources informatiques comprenant un programme à exécuter, une unité de calcul et une mémoire pour les données de travail. Cette notion permet de représenter un processeur possédant une mémoire privée. Dans la suite, nous emploierons le terme “processus”, plus adéquat que le terme processeur.

Remarque 2.2 De manière générale, l'unité de calcul d'un processus n'est pas nécessairement un processeur unique. Un programme décomposé en tâches qui sont exécutées en parallèle par plusieurs processeurs partageant une mémoire commune

est aussi un processus. Le parallélisme au sein du processus est associé à la notion de processus léger (*thread*). Ces derniers accèdent directement à la mémoire associée au processus dont ils font partie. La notion de processus léger n'existe pas dans MPI.

L'exécution d'un algorithme parallèle implémenté avec MPI correspond à l'exécution d'un ensemble de processus ayant en commun un programme séquentiel unique et communiquant à l'aide des fonctions de la bibliothèque MPI.

Le standard MPI offre la possibilité aux processus d'une application d'échanger des messages dans plusieurs contextes de communication indépendants, appelés *communicateurs*. Chaque communicateur est associé à un sous-ensemble de processus. Les processus sont toujours identifiés par des entiers naturels allant de 0 à $n-1$, n étant le nombre de processus appartenant au communicateur. Toute application MPI possède un communicateur par défaut associé à l'ensemble de tous les processus. Ces concepts ont pour but de fournir des structures de donnée standards facilitant la réutilisation de codes parallèles.

Communications

Le standard MPI définit deux types de communications entre processus : les communications point à point (entre deux processus) et les communications collectives (entre tous les processus d'un communicateur).

Les deux opérations fondamentales des communications point à point sont l'envoi et la réception d'un message. Le standard MPI définit de nombreuses méthodes pour réaliser ces opérations. Nous nous contenterons de développer les concepts qui seront mis en œuvre par la suite. Les échanges de message sont effectués à travers des canaux de communication unidirectionnels. Les paramètres tels que la source, la destination et la taille du message sont fixés lors de l'initialisation :

- le processus émetteur fixe la destination, l'emplacement en mémoire où le message à envoyer est rangé et la taille du message,
- le processus récepteur fixe la source, l'emplacement en mémoire où le message reçu sera rangé et la taille de l'emplacement.

Un processus peut alors soumettre des requêtes d'émission ou de réception de message à ses canaux de communication.

Une fois le canal de communication établi, un échange de message est effectué selon le protocole suivant :

point de vue de émetteur

1. le processus soumet sa requête d'émission qui se termine lorsque le processus récepteur a soumis la requête de réception associée au canal,
2. le processus attend la fin de sa requête et lorsque la requête d'émission est terminée, l'échange de message ne l'est pas a priori ;

point de vue du récepteur

1. le processus soumet sa requête de réception qui se termine lorsque le message envoyé par le processus émetteur est reçu dans sa totalité,
2. le processus attend la fin de sa requête et la terminaison de la requête de réception coïncide avec celle de l'échange de message.

L'attente de la terminaison d'une requête est une opération pendant laquelle le processus est inactif. On parle alors d'envois et de réceptions bloquants. À la fin de la requête, ce canal est réutilisé pour transmettre à nouveau des messages.

D'autre part, le standard MPI a défini des envois et des réceptions non-bloquantes où un test de la terminaison des requêtes se substitue à l'attente. La terminaison de la requête est testée à l'aide de fonctions de la bibliothèque.

Enfin, lorsque le canal n'est plus utilisé, le programmeur doit libérer les ressources mémoire et système acquises par les requêtes de communication.

Quant aux communications collectives, elles permettent d'implémenter des opérations telles que :

- la diffusion d'un message vers tous les processus,
- la collecte des messages de la part de tous les processus,
- les opérations arithmétiques et logiques où chaque processus possède une partie des arguments.

Les processus impliqués font partie du même contexte de communication. Les communications collectives sont toujours bloquantes.

2.3 L'algorithme de Schwarz asynchrone

La parallélisation de l'algorithme de Schwarz correspond à une méthode de décomposition de domaine avec recouvrement. Les définitions des itérations parallèles (*cf.* chapitre 1) font appel à la notion d'espace produit (*cf.* les équations (1.1) et (1.38)), qui modélise la répartition des données du problème à résoudre. Dans le cadre de la résolution d'un système linéaire, la décomposition des données concerne la matrice, le second membre et le vecteur solution.

L'utilisation de MPI est un bon compromis pour implémenter l'algorithme de Schwarz car le parallélisme ne concerne que les données. Bien que l'asynchronisme s'exprime plus naturellement par le modèle à mémoire virtuellement partagée, nous avons préféré implémenter les algorithme à l'aide des échanges de message en raison de l'absence d'un standard largement répandu garantissant la scalabilité et la portabilité. Nous renvoyons à [59] pour des implémentations des itérations asynchrones exploitant la mémoire partagée ou virtuellement partagée utilisant les threads POSIX et SHMEM.

Nous considérons le système linéaire suivant :

$$A.X = B \tag{2.1}$$

issu de la discrétisation d'un problème aux limites linéaire défini sur un domaine tridimensionnel.

Remarque 2.3 L'exposé qui suit s'applique aussi aux problèmes linéaires perturbés par une application diagonale croissante. En effet, chaque itération de l'algorithme de Newton consiste à résoudre un système linéaire [49, 50, 96].

L'utilisation de la méthode de Schwarz synchrone et asynchrone consiste à appliquer le procédé d'augmentation décrit dans la section 1.5, puis à effectuer des itérations. Nous adopterons les notations du chapitre 1, en particulier celles des sections 1.2 et 1.5. Ce procédé correspond à la procédure suivante :

1. répartir la matrice de discrétisation A et le second membre B de l'équation (2.1) sur β processus en prenant en compte le recouvrement entre les sous-domaines,
2. compléter $\hat{A}_{ii}$ et $\hat{b}_i$ au niveau des frontières introduites par le découpage.

La matrice $\hat{A}$ et le second membre $\hat{B}$ issus du procédé d'augmentation modélisent la répartition des données.

$$\begin{array}{ccccc} & & A;B & & \\ & \swarrow & \downarrow & \searrow & \\ \hat{A}_{11};\hat{b}_1 & & \cdots & & \hat{A}_{\beta\beta};\hat{b}_\beta \end{array}$$

Chaque bloc hors-diagonal $\hat{A}_{ij}$ $(i \neq j)$ de la matrice augmentée modélise un échange de message entre le $i^{\text{ème}}$ et le $j^{\text{ème}}$ processus ; $\hat{A}_{ij}$ détermine les composantes du vecteur $\hat{x}_j$ intervenant dans le calcul des itérés du $i^{\text{ème}}$ processus.

2.3.1 Les itérations asynchrones classiques

Nous nous intéresserons dans un premier temps à l'implémentation des itérations parallèles synchrones et asynchrones classiques. L'intérêt de l'algorithme classique réside dans la simplicité de sa formulation. Nous pouvons ainsi nous concentrer sur l'implémentation d'un algorithme relativement simple avant d'y greffer le concept de communication flexible.

Considérons le découpage d'un domaine Ω inclus dans $\mathbb{R}^3$ en β sous-domaines non vides qui respectent les contraintes (1.79) et (1.80) de la méthode de Schwarz. Les itérations asynchrones classiques s'écrivent sous la forme :

$$\forall p \in \mathbb{N},\ \forall i \in s(p),\ \hat{x}_i^{p+1} = \hat{A}_{ii}^{-1}.(\hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j) \tag{2.2}$$

où $\hat{A}.\hat{X} = \hat{B}$ est le système algébrique augmenté et $\tilde{\hat{x}}_j$ représente la composante envoyée par le processus j au processus i.

Le membre de droite de l'équation (2.2), correspondant au calcul d'un itéré par le $i^{\text{ème}}$ processus, s'interprète de la manière suivante :

$$\underbrace{\hat{A}_{ii}^{-1}.}_{\text{calculs}} (\hat{b}_i + \underbrace{-\sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j}_{\text{échanges de message}}). \tag{2.3}$$

D'après le procédé d'augmentation donné à la section 1.5, chaque vecteur $-\hat{A}_{ij}.\tilde{\hat{x}}_j$ correspond aux valeurs du vecteur itéré échangées entre les processus i et j. Sachant que $\hat{A}_{ij}$ est une matrice creuse dont les coefficients non nuls valent -1, le processus j n'a pas besoin d'envoyer la totalité de $\tilde{\hat{x}}_j$. Autrement dit, seuls les points de discrétisation appartenant à $\partial\Omega_i \cap \Omega_j$ sont envoyés au processus i de manière à réduire la quantité de données à échanger. Dans la suite, nous emploierons le terme "frontière de recouvrement de Ω_i sur Ω_j" pour désigner l'intersection de la frontière de Ω_i et du sous-domaine adjacent Ω_j ($\partial\Omega_i \cap \Omega_j$).

La figure 2.1 décrit l'algorithme parallèle de Schwarz synchrone. Notons que les échanges de message et les calculs se recouvrent dans le but de réduire les temps d'attente. La réception des messages qui seront exploités pour l'itération suivante commence le plus tôt possible. L'attente de la fin de la réception des messages rend le processus inactif. Cette opération doit alors commencer le plus tard possible. En ce qui concerne les envois, le raisonnement est le même. Chaque canal de communication est dédié à un message.

Quant à la figure 2.2, elle décrit l'algorithme parallèle de Schwarz asynchrone classique ($\tilde{\hat{x}}_j = \hat{x}_j^{\rho_j(p)}$). La différence principale entre la version synchrone et asynchrone réside dans les échanges de message. La suppression des temps d'inactivité des processus consiste à utiliser des tests de terminaison non bloquants pour les requêtes de communication. Les messages ne sont pas tous nécessairement reçus lorsque les calculs sont sur le point de commencer. De même, si l'envoi du message précédent n'est pas terminé, une autre tentative d'envoi sera faite lors de l'itération suivante. Ces opérations de communication asynchrones sont donc à l'origine des retards sur les composantes modélisés par la théorie (*cf.* équations (1.7) à (1.9)).

La seconde différence réside dans le test de convergence global que nous préférons traiter ultérieurement dans la section 2.5 afin de simplifier l'exposé. En effet, la mise en œuvre de la procédure d'arrêt implique des changements majeurs dans les structures de contrôle de l'algorithme asynchrone présenté à la figure 2.2.

• Algorithme exécuté par le $i^{\text{ème}}$ processus :
Paramètres principaux
    $\hat{A}_{ii}$: $i^{\text{ème}}$ bloc diagonal de la matrice augmentée
    $\hat{b}_i$: $i^{\text{ème}}$ composante du second membre augmenté
    $\hat{x}_i^0$: $i^{\text{ème}}$ composante du vecteur initial

Établir des canaux de communication avec les processus
  qui prennent en charge les sous-domaines adjacents à Ω_i
Soumettre les requêtes de réception
tant que convergence globale non détectée
    Attendre la réception des messages associés à $(\tilde{\hat{x}}_j)_{j\in \text{adj}(i)}$
      contenant les valeurs de $\hat{X}$ correspondant
      aux frontières de recouvrement de Ω_i
      sur les sous-domaines adjacents
    Soumettre les requêtes de réception pour l'itération suivante
    Résoudre $\hat{A}_{ii}.\hat{x}_i^{p+1} = \hat{b}_i - \sum_{j\neq i} \hat{A}_{ij}.\hat{x}_j^p$
    Attendre la fin des envois précédents
    Soumettre les requêtes d'envoi des messages contenant
      les valeurs de $\hat{x}_i$ correspondant
      aux frontières de recouvrement
      des sous-domaines adjacents à Ω_i
fin
Libérer les ressources acquises

FIG. 2.1 – Algorithme de Schwarz synchrone

• Algorithme exécuté par le $i^{\text{ème}}$ processus :
Paramètres principaux
$\hat{A}_{ii}$: $i^{\text{ème}}$ bloc diagonal de la matrice augmentée
$\hat{b}_i$: $i^{\text{ème}}$ composante du second membre augmenté
$\hat{x}_i^0$: $i^{\text{ème}}$ composante du vecteur initial

Établir des canaux de communication avec les processus
qui prennent en charge les sous-domaines adjacents à Ω_i
Soumettre les requêtes de réception
tant que convergence globale non détectée
Tester la réception des messages associés à $(\tilde{\hat{x}}_j)_{j\in \text{adj}(i)}$
contenant les valeurs de $\hat{X}$ correspondant
aux frontières de recouvrement de Ω_i
sur les sous-domaines adjacents
pour toutes les requêtes de réception terminées
Soumettre les requêtes de réception pour l'itération suivante
fin
Résoudre $\hat{A}_{ii}.\hat{x}_i^{p+1} = \hat{b}_i - \sum_{j\neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j$
Tester la fin des envois précédents
pour toutes les requêtes d'envoi terminées
Soumettre les requêtes d'envoi des messages contenant
les valeurs de $\hat{x}_i$ correspondant
aux frontières de recouvrement
des sous-domaines adjacents à Ω_i
fin
fin
Libérer les ressources acquises

FIG. 2.2 – Algorithme de Schwarz asynchrone classique

2.3.2 Les échanges de message

Contenu des messages

Le premier problème algorithmique à résoudre est la construction des messages contenant les valeurs du vecteur itéré correspondant aux frontières de recouvrement entre les sous-domaines voisins. Il est nécessaire de mettre en place des structures de donnée qui décrivent les sous-domaines. Chaque message est un vecteur creux dont la structure dépend de paramètres tels que :

- la géométrie des sous-domaines,
- la matrice de discrétisation.

Par exemple, si un sous-domaine Ω_i est représenté par un ensemble de numéros de lignes L_i, la structure des messages reçus par le $i^{\text{ème}}$ processus est déduite en analysant chaque ligne de la matrice de discrétisation dont le numéro appartient à L_i.

Dans le cas où le domaine Ω ainsi que les sous-domaines $(\Omega_i)_{1 \leq i \leq \beta}$ sont parallélépipédiques, et lorsque la matrice de discrétisation du problème tridimensionnel est heptadiagonale (c'est le cas avec la méthode des différences finies à 7 points ou avec les volumes finis), il est possible de définir la structure des messages à partir de considérations géométriques. Ce cas est illustré dans la figure 2.3. Le principe de l'implémentation repose sur une représentation des ensembles de numéros de lignes sous la forme suivante :

$$\{(i,j,k) \in \mathbb{N}^3 \mid i_1 \leq i \leq i_2, j_1 \leq j \leq j_2, k_1 \leq k \leq k_2\}$$

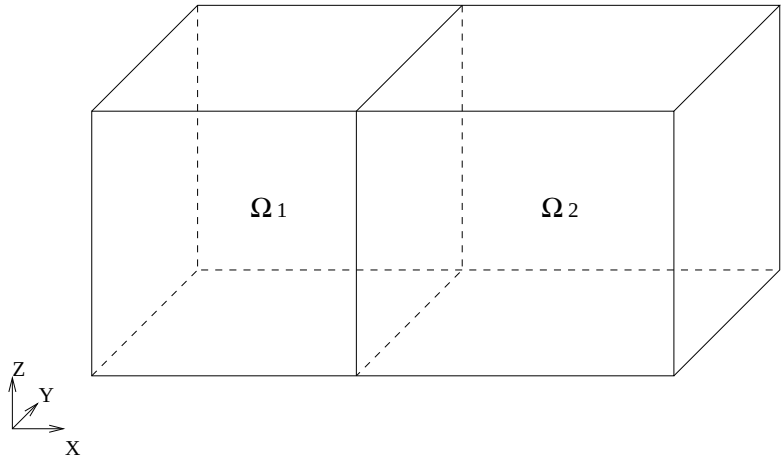
avec $i_1 \leq i_2$, $j_1 \leq j_2$ et $k_1 \leq k_2$. Les numéros de lignes sont remplacés par des coordonnées cartésiennes entières qui sont en adéquation avec le maillage. Les nombres $i_1, i_2, j_1, j_2, k_1, k_2$ caractérisent les frontières du domaine discret. La manipulation des sous-domaines et des frontières est facilitée, car il suffit de stocker ces nombres en mémoire. L'erreur d'implémentation à ne pas commettre est de stocker les vecteurs dans des tableaux tridimensionnels car les vecteurs ne seraient plus nécessairement stockés en mémoire de façon contiguë.

Mise en œuvre des échanges

La mise en œuvre des canaux de communication avec MPI est réalisée en manipulant les *requêtes persistantes* définies par le standard MPI. La création d'un canal de communication consiste à initialiser des requêtes persistantes d'envoi et de réception :

- d'une part, le processus émetteur fait appel à la fonction `MPI_SSEND_INIT()`, qui correspond au mode de communication *synchrone* décrit dans la section 2.2.3,
- et d'autre part, le processus récepteur fait appel à la fonction `MPI_RECV_INIT()`.

Découpage d'un domaine tridimensionnel en 2 sous-domaines :



Une coupe du découpage dans le plan (X, Y) avec le maillage :

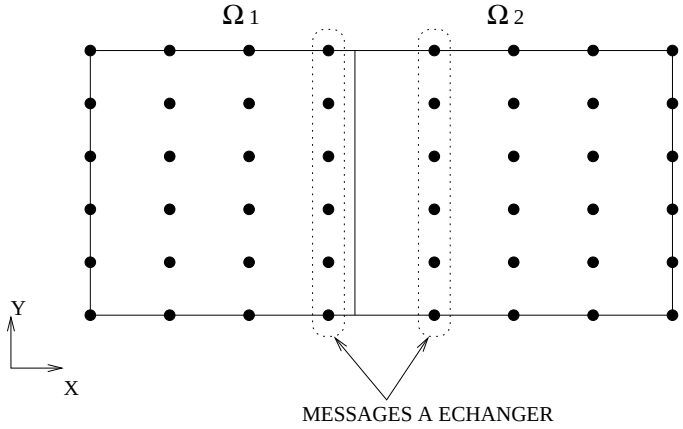


FIG. 2.3 – Exemple de découpage d'un domaine tridimensionnel

Sur chaque canal de communication, les messages qui sont transmis correspondent aux valeur successives du vecteur itéré sur une frontière de recouvrement. Les requêtes d'envoi et de réception sont alors soumises à l'aide de la fonction `MPI_START()`. En ce qui concerne l'attente et le test de la fin des requêtes d'envoi ou de réception, les fonctions `MPI_WAIT()` et `MPI_TEST()`, ainsi que leurs variantes (`MPI_WAITALL()` et `MPI_TESTSOME()`) sont respectivement utilisées.

L'utilisation du mode d'envoi synchrone n'est pas incompatible avec l'asynchronisme. En effet, le mécanisme de rendez-vous qui est sous-jacent à ce mode de communication n'interfère pas avec le concept d'itération parallèle asynchrone, dans la mesure où ce mode de communication n'empêche en aucun cas la suppression des temps d'attente.

Remarque 2.4 Avec le mode d'envoi "*ready*" de MPI, le processus émetteur n'effectue aucune synchronisation avec le récepteur. L'émetteur suppose qu'une requête de réception adéquate a été soumise au moment même où la requête d'envoi est soumise. L'utilisation de ce mode de communication nuit à la robustesse des programmes car le standard ne spécifie aucun comportement particulier dans le cas où le message parvient au récepteur alors que celui-ci n'était pas prêt à le recevoir.

Remarque 2.5 En ce qui concerne le mode d'envoi "*buffered*" de MPI, le processus émetteur recopie le message dans une mémoire tampon et envoie en mode synchrone la copie du message. Ainsi, une requête d'envoi se termine dès que le message est recopié. De plus, la mémoire tampon peut contenir plusieurs messages successifs. Ce mode offre donc une réduction des temps d'attente engendré par les envois de message, sans pour autant sacrifier la robustesse. Étant donné que ce gain de performance se fait au détriment de la quantité de mémoire requise à l'exécution du programme, ce mode de communication n'a pas été retenu. En effet, plus la fréquence de communication est élevée, plus la taille du tampon doit être importante.

À titre d'exemple, la figure 2.4 donne un aperçu de l'implémentation en Fortran 90 de l'envoi et de la réception asynchrone de message. Comme chaque processus doit gérer plusieurs requêtes d'envoi et de réception, la procédure `MPI_TESTSOME()` a été utilisée pour faciliter l'implémentation du test de plusieurs requêtes persistantes. L'appel à la fonction `MPI_START()` sert à réactiver les requêtes terminées. Le "paquetage" et le "dépaquetage" des messages, nécessaires lorsque les données à échanger ne sont pas contigus en mémoire, fait partie du surcoût de la parallélisation aussi bien dans les itérations synchrones qu'asynchrones.

La méthode de communication asynchrone que nous avons présentée ici, consiste à tester la fin des requêtes, puis à réactiver celles qui sont terminées. Même si l'asynchronisme permet de supprimer les temps d'attente, il est tout de même nécessaire de veiller à recevoir un maximum de messages avant de commencer un calcul. La méthode employée revient donc à faire recouvrir les calculs et les communications. Elle restera valable pour les itérations asynchrones avec communication flexible.

```
! nout : nombre de requetes terminees
! outarray : tableau contenant les indices
             des requetes terminees
! starray : tableau des informations associees
            aux requetes terminees
! ierr : code d'erreur

! ENVOI ASYNCHRONE
! NSEND : nombre de requetes d'envoi
! sndt : tableau de requetes persistantes d'envoi

call MPI_TESTSOME(NSEND,sndt,nout,outarray,starray,ierr)
do i=1, nout
  mpos = outarray(i)
  call MSGPK(... PAQUETAGE DU MESSAGE ...)
  call MPI_START(sndt(mpos),ierr)
end do

! RECEPTION ASYNCHRONE
! NRECV : nombre de requetes de reception
! rcvt : tableau de requetes persistantes de reception

call MPI_TESTSOME(NRECV,rcvt,nout,outarray,starray,ierr)
do i=1, nout
  mpos = outarray(i)
  call MSGUPK(... DEPAQUETAGE DU MESSAGE ...)
  call MPI_START(rcvt(mpos),ierr)
end do
```

FIG. 2.4 – Implémentation en Fortran 90 des communications asynchrones à l'aide des requêtes persistantes MPI

2.3.3 Plusieurs sous-domaines par processus

La prise en charge de plusieurs sous-domaines par processus permet de mettre en œuvre une stratégie de relaxation multiplicative (de type Gauss-Seidel) dans le but d'accélérer la convergence des algorithmes parallèles[2]. L'implémentation de cette amélioration peut être envisagée selon 2 méthodes :

1. Les processus prennent en charge plusieurs composantes du vecteur itéré et du second membre, ainsi que plusieurs blocs diagonaux de la matrice. Les composantes $\hat{x}_i$ et $\hat{b}_i$ ainsi que les blocs diagonaux $\hat{A}_{ii}$ qui sont pris en charge par un processus sont donc stockés dans des tableaux distincts, si bien que malgré le recouvrement entre les sous-domaines, les données en mémoire ne se recouvrent pas. Cette méthode correspond donc à une adaptation du programme parallèle pour l'exécution séquentielle.
2. Chaque processus résout son système $\hat{A}_{ii}.\hat{x}_i^{p+1} = \hat{b}_i - \sum_{j\neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j$ avec une méthode de sous-domaine dont l'implémentation diffère de celle des algorithmes parallèles. Il faut considérer la décomposition de chaque composante $\hat{x}_i$ en α_i "*sous-composantes*" qui découlent d'une décomposition de chaque sous-domaine Ω_i en α_i sous-domaines. Chaque processus doit alors prendre en charge la résolution de α_i systèmes linéaires que l'on peut écrire avec les notations suivantes :
$$[\hat{A}_{ii}]_{kk}.[\hat{x}_i^{p+1}]_k = [b'_i]_k - \sum_{\ell\neq k}[\hat{A}_{ii}]_{k\ell}.[\hat{x}_i^p]_\ell, k \in \{1,\ldots,\alpha_i\} \tag{2.4}$$
où
$$b'_i = \hat{b}_i - \sum_{j\neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j. \tag{2.5}$$

   Les notations $[\,.\,]_k$ et $[\,.\,]_{k\ell}$ désignent respectivement la $k^{\text{ème}}$ sous-composante d'un vecteur et le $\ell^{\text{ème}}$ bloc de la matrice associée à la $k^{\text{ème}}$ sous-composante.

   Cette seconde décomposition ne fait pas appel au procédé d'augmentation. Autrement dit, les données des sous-composantes de $\hat{x}_i$ et de $\hat{b}_i$ ne sont pas dispersées dans des tableaux distincts. Il en est de même pour les "*sous-blocs*" de $\hat{A}_{ii}$. Chaque système algébrique (2.4) fait donc intervenir des vecteurs qui ne sont pas nécessairement stockés en mémoire de manière contiguë. L'algorithme parallèle qui en résulte est donc issu du couplage entre deux implémentations différentes de l'algorithme de Schwarz : l'une adaptée au parallélisme avec échange de message, l'autre optimisée pour l'exécution séquentielle.

Nous avons choisi d'implémenter et d'expérimenter la seconde méthode car celle-ci découle d'un algorithme de Schwarz séquentiel plus efficace. Notamment, le procédé d'augmentation n'est pas utilisé. Cet algorithme est donc équivalent à des relaxations

[2]Si les processus ne prennent en charge qu'un seul sous-domaine, on dit que la stratégie de relaxation est additive (de type Jacobi).

par blocs ; les blocs peuvent se recouvrir et leur stockage en mémoire peut être non-contigu.

Un algorithme de Schwarz séquentiel

L'algorithme de Schwarz séquentiel est décrit dans la figure 2.5. Il ne subit aucune

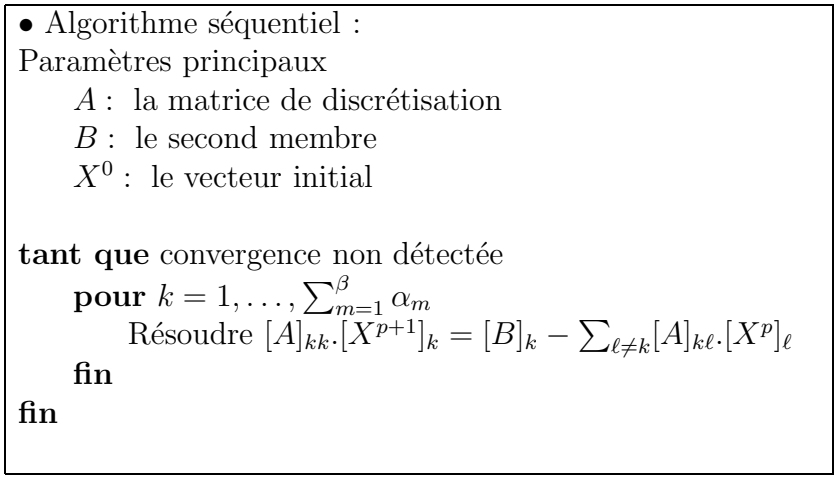


FIG. 2.5 – Algorithme de Schwarz séquentiel

des conséquences de la parallélisation, autant sur le plan de l'algorithmique que des structures de donnée car le procédé d'augmentation n'est pas appliqué. Chaque bloc du vecteur itéré, noté $[X^{p+1}]_k$ avec $k \in \{1, \ldots, \sum_{m=1}^{\beta} \alpha_m\}$, correspond à une sous-composante issue de la décomposition en sous-domaines. Une notation similaire est adoptée pour les blocs de la matrice A. Ces notations sont directement inspirées de celles qui sont employées dans l'équation (2.4).

Les algorithmes de Schwarz parallèles

Quant aux algorithmes de Schwarz parallèles, la répartition des données est effectuée via le procédé d'augmentation de Schwarz appliqué à β sous-domaines. Le découpage final est obtenu en découpant Ω_i en α_i sous-domaines. Ensuite, chaque processus fait appel à l'algorithme séquentiel de la figure 2.5 pour résoudre son système linéaire $\hat{A}_{ii}.\hat{x}_i^{p+1} = \hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_i$ découpé en α_i sous-problèmes. L'algorithme est décrit dans la figure 2.6. La distinction entre les versions synchrones et asynchrones n'apparaît pas explicitement car les méthodes utilisées pour communiquer sont semblables à celles qui sont utilisées dans les algorithmes précédents (*cf.* figures 2.1 et 2.2). Ces derniers ont lieu lors de la mise à jour de chaque sous-composante de manière à effectuer les itérations dans un ordre similaire à celui de l'algorithme séquentiel.

• Algorithme exécuté par le $i^{\text{ème}}$ processus :
Paramètres principaux
$\hat{A}_{ii}$: $i^{\text{ème}}$ bloc diagonal de la matrice augmentée
$\hat{b}_i$: $i^{\text{ème}}$ composante du second membre augmenté
$\hat{x}_i^0$: $i^{\text{ème}}$ composante du vecteur initial
α_i : nombre de sous-domaines pris en charge par le processus

Initialiser les échanges de message
tant que convergence globale non détectée
pour $k = 1, \ldots, \alpha_i$
Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j \in \text{adj}(i)}$
$b'_i = \hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j$
Résoudre $[\hat{A}_{ii}]_{kk}.[\hat{x}_i^{p+1}]_k = [b'_i]_k - \sum_{\ell \neq k}[\hat{A}_{ii}]_{k\ell}.[\hat{x}_i^p]_\ell$
Envoyer les messages associés à $\hat{x}_i$
fin
fin
Libérer les ressources acquises

FIG. 2.6 – Algorithme de Schwarz parallèle avec plusieurs sous-domaines par processus

Stratégies de relaxation

Dans les algorithmes séquentiels et parallèles des figures 2.5 et 2.6, la stratégie de relaxation est additive. Puisque les différents sous-domaines qui sont pris en charge par un processus sont traités de façon séquentielle, il est possible d'introduire une stratégie de relaxation multiplicative au niveau de chaque processus. En effet, le $i^{\text{ème}}$ processus peut exploiter les valeurs les plus récentes issues des α_i sous-composantes, sans échanger de message. C'est une raison pour laquelle la prise en charge de plusieurs sous-domaines par processus est plus avantageuse.

Notons qu'en ce qui concerne l'algorithme de Schwarz séquentiel, l'introduction d'une stratégie de relaxation multiplicative revient à résoudre $A.X = B$ par un algorithme par blocs de type Gauss-Seidel. En revanche, dans le cas des algorithmes parallèles, l'introduction d'une stratégie de relaxation multiplicative au niveau du traitement séquentiel de chaque processus revient seulement à résoudre chaque système $\hat{A}_{ii}.\hat{x}_i = \hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_i$ par un algorithme par blocs de type Gauss-Seidel. Dans ce cas, la stratégie de relaxation associée aux itérations parallèles n'est pas complètement multiplicative. Nous verrons dans la suite comment la rendre multiplicative.

2.3.4 Les difficultés algorithmiques

Les problèmes algorithmiques majeurs liés à cette implémentation de la méthode de Schwarz concernent :

- les impacts du découpage en sous-composantes sur le surcoût de la parallélisation,
- les risques d'interblocage dans les échanges synchrones de message.

La confrontation avec ces difficultés est inévitable car nous avons constaté expérimentalement que les algorithmes de Schwarz parallèles sont plus efficaces lorsque plusieurs sous-domaines sont pris en charge par un processus.

Les problèmes liés aux échanges de message

Les échanges de message sont à l'origine des difficultés rencontrées. Nous avons vu dans la section 2.3.1 que les messages s'exprimaient à l'aide des blocs hors-diagonaux de la matrice augmentée. Leur expression est $-\hat{A}_{ij}.\tilde{\hat{x}}_j$, avec $i \neq j$ et $1 \leq i, j \leq \beta$, sachant que β est le nombre de processus. Dans le cas où plusieurs sous-domaines sont pris en charge par un processus, cette expression ne tient pas compte du découpage en sous-composantes car le second niveau de découpage est ignoré par le procédé d'augmentation. Les échanges de message doivent-ils être implémentés en fonction du premier niveau de découpage, qui est pris en compte par le procédé d'augmentation, ou bien du second niveau de découpage qui correspond au découpage en sous-composantes ?

Remarque 2.6 Le premier niveau de découpage, qui est pris en compte par le procédé d'augmentation du système, est plus grossier que le découpage en sous-composantes, qui correspond au découpage réel du domaine. Autrement dit, le premier niveau de découpage correspond à la répartition des données du problème sur les processus.

Après la mise à jour d'une sous-composante, il est nécessaire de la rendre disponible le plus tôt possible pour les processus qui en ont besoin afin d'obtenir une stratégie de relaxation multiplicative. Le fait d'implémenter les échanges de messages en fonction du niveau de découpage le plus grossier va avoir un impact négatif sur l'efficacité de la parallélisation. En effet, en envoyant ces messages (les vecteurs $-\hat{A}_{ij}.\tilde{\hat{x}}_j$ contenant les valeurs du vecteur itéré qui relient deux processus) après la mise à jour de chaque sous-composante, il y a de fortes chances que des données inutiles, n'ayant pas été affectées par une relaxation, soient envoyées.

Afin de résoudre ce problème, l'implémentation des échanges de message doit prendre en compte le second niveau de découpage, c'est à dire le plus fin. Ce choix n'est pas sans conséquence sur l'implémentation et le déroulement des échanges de message. Deux processus donnés peuvent être amenés à communiquer par l'intermédiaire de plus de 2 canaux, chaque canal étant dédié à un message. Ainsi, les

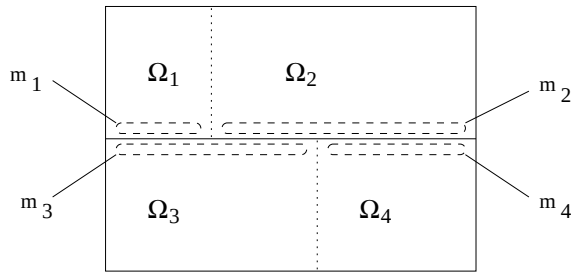
différents messages peuvent être reçus dans un ordre quelconque. Cependant, cette solution n'est pas sans faille pour les raisons suivantes :

- L'augmentation du nombre de requêtes de communication réduit nécessairement la taille des données à échanger. Or c'est en échangeant de grandes quantités de données tout en cherchant à minimiser le nombre de requêtes qu'on parvient à amortir le poids des communications.
- Cette solution n'est pas robuste vis à vis du découpage du domaine car le nombre de canaux de communication utilisés en dépend directement. En effet, le traitement d'une sous-composante peut nécessiter la réception de nombreux messages, notamment si le sous-domaine associé à la sous-composante est adjacent à de nombreux sous-domaines pris en charge par d'autres processus. Un découpage qui nécessite l'utilisation de nombreux canaux sera d'autant plus pénalisé.

Les problèmes liés à la synchronisation

Cette solution a également des conséquences sur l'algorithme synchrone. En effet, l'interblocage (ou *deadlock*) survient lorsqu'un processus attend la réception d'un message qui n'a pas été envoyé. Par conséquent, ce processus n'effectue pas l'itération courante et n'envoie pas les parties des messages qu'il aurait mis à jour, ce qui conduit au blocage de tous les autres processus.

Exemple 2.1 (Interblocage) Prenons un exemple 2D pour illustrer les risques d'interblocage. Considérons un domaine rectangulaire découpé en 4 sous-domaines qui sont pris en charge par 2 processus ($P_1 \leftarrow \Omega_1, \Omega_2$ et $P_2 \leftarrow \Omega_3, \Omega_4$).



Autrement dit, chaque composante du vecteur itéré est découpé en 2 sous-composantes. Chaque processus doit gérer 2 messages notés respectivement m_1, m_2, m_3 et m_4. Lorsque le processus P_2 commence à traiter le sous-domaine Ω_3, il doit attendre l'arrivée de m_1 et de m_2 ; donc P_2 doit attendre la fin du traitement de Ω_1, puis de Ω_2. Or le traitement de Ω_2 ne peut commencer que si P_2 envoie m_3. En résumé, P_2 attend la fin du traitement de Ω_2 par P_1, qui lui-même attend la fin du traitement de Ω_3 par P_2 : c'est l'interblocage.
Pour éviter cela, il faut traiter les sous-domaines dans un ordre particulier. Cependant, dans un contexte tridimensionnel où chaque sous-domaine peut potentiellement

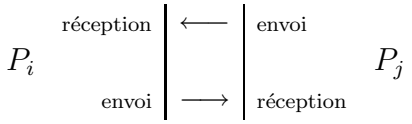
être adjacent à une vingtaine de sous-domaines, trouver l'ordre dans lequel il faut traiter les sous-domaines de manière à éviter l'interblocage est un exercice difficile pour lequel il n'existe pas à notre connaissance de solution générale qui soit valable quelque soit la géométrie et le découpage du domaine.

Il est important de noter que les algorithmes asynchrones sont immunisés contre l'interblocage. Par conséquent, l'implémentation de la solution consistant à prendre en compte le découpage en sous-composantes n'a pas le même impact selon la version synchrone ou asynchrone de l'algorithme parallèle de Schwarz.

La solution algorithmique retenue

Les problèmes présentés ci-dessus ont conduit au compromis suivant.

1. Les échanges de message sont implémentés en fonction du premier niveau de découpage. Cette façon de procéder permet de limiter le nombre de canaux de communication. Entre 2 processus P_i et P_j (tels que $1 \leq i, j \leq \beta$), seuls 2 canaux de communication peuvent exister.



   Le nombre de canaux est donc indépendant du découpage en sous-composantes et la décomposition en sous-composantes n'interfère plus avec la structure des messages. Par conséquent, la totalité d'un message est envoyée même si seule une partie de ce dernier contient des données récemment calculées.
2. Après la mise à jour d'une sous-composante, tous les messages associés à toutes les frontières de recouvrement sont envoyés. Notons que si nous avions voulu n'envoyer que les messages contenant des valeurs issues d'une mise à jour, nous aurions dû faire face à des problèmes d'interblocage dans l'algorithme synchrone.

L'envoi inconditionnel de la totalité des messages après chaque relaxation est donc un gaspillage qui augmente le surcoût de la parallélisation. Toutefois, cela permet de rendre les communications indépendantes du découpage en sous-composantes car cette indépendance est nécessaire pour envisager le développement de méthodes générales. Une solution résolvant partiellement ce problème de gaspillage sera donnée durant la présentation des algorithmes asynchrones avec communication flexible.

2.4 Les communications flexibles

Le but des communications flexibles est de permettre la réception et l'envoi de donnée à tout moment, y compris pendant les calcul des itérés. Cela revient à communiquer pendant la résolution des systèmes algébriques. Le concept de communication

flexible décrit un ensemble d'algorithmes itératifs plus vaste que les algorithmes parallèles asynchrones classiques. Les notions de sur-application et d'itération interne, utilisées respectivement dans les études par les techniques d'ordre partiel (*cf.* section 1.3) et de contraction (*cf.* section 1.4), modélisent l'utilisation d'un algorithme itératif par chaque processus pour résoudre ses systèmes algébriques. Notre exposé s'appuiera sur le modèle par contraction. Notons que le modèle par ordre partiel diffère peu de ce modèle, mis à part l'initialisation.

L'implémentation des algorithmes de Schwarz avec communication flexible est basé sur le principe suivant : les échanges de message ont lieu entre les itérations internes de l'algorithme itératif qui résout les systèmes algébriques.

– Les envois des messages peuvent avoir lieu après n'importe quelle itération interne.
– Les réceptions des messages peuvent s'effectuer avant chaque itération interne.

Les processus peuvent ainsi disposer des valeurs les plus récemment calculées du vecteur itéré. Ces principes sont mis en œuvre dans l'algorithme parallèle à *haute fréquence de communication* [51] (*cf.* figure 2.7). À ce stade, nous pouvons préciser

• Algorithme exécuté par le $i^{\text{ème}}$ processus :
Paramètres principaux
    $\hat{A}_{ii}$: $i^{\text{ème}}$ bloc diagonal de la matrice augmentée
    $\hat{b}_i$: $i^{\text{ème}}$ composante du second membre augmenté
    $\hat{x}_i^0$: $i^{\text{ème}}$ composante du vecteur initial
    α_i : nombre de sous-domaines pris en charge par le processus

Initialiser les échanges de message
tant que convergence globale non détectée
    pour $k = 1, \ldots, \alpha_i$
        tant que convergence non détectée
            Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j \in \text{adj}(i)}$
            $b'_i = \hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j$
            Itérer pour résoudre $[\hat{A}_{ii}]_{kk}.[\hat{x}_i^{p+1}]_k = [b'_i]_k - \sum_{\ell \neq k}[\hat{A}_{ii}]_{k\ell}.[\hat{x}_i^p]_\ell$
            Envoyer les messages associés à $\hat{x}_i$
        fin
    fin
fin
Libérer les ressources acquises

FIG. 2.7 – Algorithme de Schwarz parallèle asynchrone à haute fréquence de communication

que les itérations internes sont des relaxations par blocs ou par points.

Le défaut des algorithmes à haute fréquence de communication réside dans le surcoût dû aux communications [51]. Le travail de mise au point qui sera effectué sur cet algorithme est basé sur le réglage de la fréquence de communication. Nous donnerons également une solution au problème d'efficacité des échanges de message soulevé dans la section 2.3.4.

Le surcoût de la parallélisation dépend principalement de la quantité de données envoyées. La première idée qui a été exploitée est de réduire le nombre d'envois de message en les effectuant toutes les τ itérations. Cependant, cette méthode s'est rapidement révélée insuffisante. En effet, rien ne garantit qu'un message envoyé est exploité dès sa réception. Ce constat est à l'origine de la solution qui est mise en œuvre pour améliorer l'efficacité des algorithmes asynchrones avec communication flexible. Elle consiste essentiellement à :

1. réorganiser les itérations internes de manière à exploiter au plus tôt les messages reçus,
2. mettre en œuvre une stratégie de relaxation multiplicative de type rouge-noir au niveau des sous-domaines, afin de compenser les faiblesses de la politique de communication choisie dans la section 2.3.4.

Ces deux aspects seront présentés dans les sections 2.4.1 et 2.4.2.

2.4.1 Réorganisation des itérations internes

Afin d'exploiter les messages au plus tôt après leur réception, il suffit de changer de sous-composante dès que quelques itérations internes sont effectuées. En effet, un message reçu n'est exploité que lorsque les itérations internes qui font intervenir celui-ci sont effectuées. Changer de sous-domaine revient dont à exploiter un autre message. On simule en quelque sorte une exécution concurrente des itérations internes sur l'ensemble des sous-composantes.

Le fait de changer de sous-composante alors que les itérations internes n'ont pas convergé n'est pas en contradiction avec la théorie. L'ordre des calculs n'a pas d'importance, pourvu qu'aucune sous-composante ne cesse d'être mise à jour [71, 33].

L'implémentation de l'algorithme parallèle asynchrone avec communication flexible est décrit dans la figure 2.8. Conformément à la formulation des algorithmes avec communication flexible, les sous-composantes du vecteur itéré sont mises à jour avec les résultats de calculs intermédiaires : lorsque τ itérations internes sont effectuées, $[\hat{x}_i^{p+1}]_k$ prend la valeur du dernier itéré calculé. Autrement dit, on n'attend pas la convergence du "solveur interne" pour changer de sous-domaine. La fréquence des envois est alors réglée via le paramètre τ.

Notons que l'algorithme asynchrone a la possibilité de recevoir des messages à tout moment du calcul. C'est le principe des algorithmes asynchrones à *moyenne fréquence de communication* [51] qui est mis en œuvre. La réception des messages est donc testée avant chaque itération interne. Notons enfin que les remarques concernant la stratégie de relaxation (*cf.* page 62) sont encore valables.

• Algorithme exécuté par le $i^{\text{ème}}$ processus :
Paramètres principaux
$\hat{A}_{ii}$: $i^{\text{ème}}$ bloc diagonal de la matrice augmentée
$\hat{b}_i$: $i^{\text{ème}}$ composante du second membre augmenté
$\hat{x}_i^0$: $i^{\text{ème}}$ composante du vecteur initial
α_i : nombre de sous-domaines pris en charge par le processus
τ : le nombre d'itérations internes à effectuer

Initialiser les échanges de message
tant que convergence globale non détectée
pour $k = 1, \ldots, \alpha_i$
Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j \in \text{adj}(i)}$
$b'_i = \hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j$
pour $q = 1, \ldots, \tau$
Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j \in \text{adj}(i)}$
$b'_i = \hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\tilde{\hat{x}}_j$
Itérer pour résoudre $[\hat{A}_{ii}]_{kk}.[\hat{x}_i^{p+1}]_k = [b'_i]_k - \sum_{\ell \neq k}[\hat{A}_{ii}]_{k\ell}.[\hat{x}_i^p]_\ell$
fin
Envoyer les messages associés à $\hat{x}_i$
fin
fin
Libérer les ressources acquises

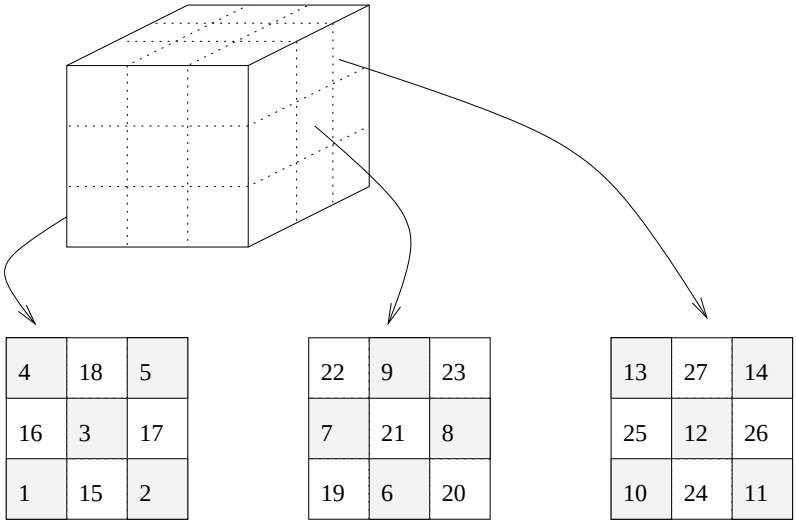
FIG. 2.8 – Réorganisation des itérations internes dans l'algorithme de Schwarz parallèle asynchrone avec communication flexible

2.4.2 Numérotation rouge-noir des sous-domaines

Il est indispensable de prendre en compte le fait que la structure des messages est indépendante du découpage en sous-composantes. En effet, chaque message contient la totalité des valeurs situées sur une frontière de recouvrement. En décidant d'envoyer systématiquement un message après avoir effectué τ itérations internes sur une sous-composante, ce qui revient à offrir aux autres processus les valeurs les plus récentes, seule une partie du message est a priori significative lors de chaque envoi. La vitesse de convergence est donc améliorée au détriment de l'efficacité de la parallélisation.

Une solution simple permettant de résoudre partiellement le problème concernant la partie utile des messages, soulevé dans la section 2.3.4, consiste à envoyer les messages uniquement après la mise à jour de toutes les sous-composantes. Cette solution a l'inconvénient majeur de provoquer une chute de la vitesse de convergence car les processus ne disposent plus des valeurs les plus récemment calculées. De plus, l'algorithme qui en résulte s'éloigne de la flexibilité. Le but est donc de réduire la fréquence de communication tout en cherchant à éviter de dégrader la vitesse de convergence.

La numérotation rouge-noir sur les sous-domaines pris en charge par chaque processus est un compromis entre la solution évoquée ci-dessus et l'algorithme de la figure 2.8. La figure 2.9 donne un exemple de numérotation. Les envois de message sont



Les sous-domaines rouges sont numérotés de 1 à 14, les noirs de 15 à 27.

FIG. 2.9 – Numérotation rouge-noir d'un découpage en sous-domaines

effectués après le traitement de tous les sous-domaines rouges et après celui de tous les noirs. Les messages contiennent donc plus de valeurs mises à jour. La fréquence de communication est réduite sans pour autant dégrader la vitesse de convergence grâce

à la stratégie de relaxation qui découle de la numérotation rouge-noir. L'algorithme est donné dans la figure 2.10.

• Algorithme exécuté par le $i^{\text{ème}}$ processus :
Paramètres principaux
$\hat{A}_{ii}$: $i^{\text{ème}}$ bloc diagonal de la matrice augmentée
$\hat{b}_i$: $i^{\text{ème}}$ composante du second membre augmenté
$\hat{x}_i^0$: $i^{\text{ème}}$ composante du vecteur initial
α_i : nombre de sous-domaines pris en charge par le processus
τ : le nombre d'itérations internes à effectuer

Initialiser les échanges de message
tant que convergence globale non détectée
Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j\in \text{adj}(i)}$
pour $k = 1, \ldots, \frac{\alpha_i}{2}$
pour $q = 1, \ldots, \tau$
Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j\in \text{adj}(i)}$
Itération interne sur la $k^{\text{ème}}$ sous-composante
fin
fin
Envoyer les messages associés à $\hat{x}_i$

Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j\in \text{adj}(i)}$
pour $k = \frac{\alpha_i}{2} + 1, \ldots, \alpha_i$
pour $q = 1, \ldots, \tau$
Recevoir les messages associés à $(\tilde{\hat{x}}_j)_{j\in \text{adj}(i)}$
Itération interne sur la $k^{\text{ème}}$ sous-composante
fin
fin
Envoyer les messages associés à $\hat{x}_i$
fin
Libérer les ressources acquises

FIG. 2.10 – Numérotation rouge-noir appliquée à l'algorithme de Schwarz parallèle asynchrone avec communication flexible

Notons que ce procédé de numérotation revient aussi à appliquer des permutations sur la matrice augmentée. La matrice qui en résulte reste une M-matrice car les signes des coefficients diagonaux et hors-diagonaux ne sont pas modifiés par la transformation.

2.5 Terminaison des algorithmes

2.5.1 Convergence globale

La détection de la convergence des itérations parallèles synchrones et asynchrones est basée sur le principe suivant [15] : un critère de convergence locale est évalué au niveau de chaque processus, de telle sorte que la convergence globale soit atteinte lorsque la convergence locale est vérifiée par tous les processus. Voici quelques exemples de critères de convergence locaux :

- Tester si la norme du résidu, $\|\hat{b}_i - \hat{A}_{ii}.\hat{x}_i^{p+1}\|$, calculée après tous les tests de réception des messages, est inférieure à un seuil ϵ.
- Tester si la norme de la différence entre 2 itérés successifs, $\|\hat{x}_i^{p+1} - \hat{x}_i^p\|$, est inférieure à un seuil ϵ.

L'exposé qui suit a pour but de présenter les techniques algorithmiques mises en œuvre pour détecter l'instant où la convergence locale est atteinte par tous les processus.

La détection de la convergence globale des algorithmes synchrones ne pose aucun problème. Il suffit d'utiliser l'opération collective `MPI_ALLREDUCE()`. En revanche, les difficultés surviennent dans le cas asynchrone.

La détection de la terminaison dans le contexte asynchrone est techniquement difficile notamment à cause de l'absence de synchronisation qui engendre les retards des messages. En effet, un processus ayant atteint la convergence locale ne détient pas nécessairement une composante suffisamment proche de la solution si les composantes du vecteur itéré les plus récentes n'ont pas été reçues (la réception de ces dernières pourraient invalider la convergence locale).

Une seconde difficulté réside dans l'implémentation du test d'arrêt, où toute synchronisation doit être évitée afin de ne pas perdre les bénéfices de l'asynchronisme des échanges de message.

Parmi les méthodes de terminaison des itérations parallèles asynchrones disponibles dans la littérature [15, 31], nous avons choisi d'implémenter un algorithme de terminaison inspiré du *snapshot* de K.M. Chandy et L. Lamport [19] car notre algorithme asynchrone ne rentre pas dans le cadre des calculs diffusants. Le principe de cet algorithme est donné dans [15].

La détection de la terminaison se ramène au problème de l'évaluation de la condition booléenne suivante :

$$\begin{array}{c} \text{“la convergence locale est vérifiée sur tous les processus} \\ \textbf{et} \\ \text{toutes les composantes du vecteur itéré envoyées ont été reçues”.} \end{array} \tag{2.6}$$

La proposition (2.6) est a priori toujours fausse si les envois de message ne cessent pas lorsque le vecteur itéré est suffisamment proche de la solution. D. Bertsekas

et J.N. Tsitsiklis ont introduit la contrainte suivante :

> "tout processus vérifiant la convergence locale
> n'envoie plus de composante du vecteur itéré" (2.7)

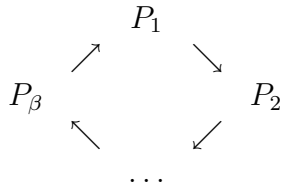
afin de rendre la détection de la convergence globale possible.

Cependant, la contrainte (2.7) est incompatible avec l'implémentation de nos envois de message. En effet, lorsqu'un processus souhaite envoyer un message, si le canal de communication est *obstrué* par la requête précédente qui n'est pas terminée, alors l'envoi est reporté à la prochaine itération. Il faut donc compléter la contrainte (2.7) avec la contrainte suivante :

> "tout processus vérifiant la convergence locale
> doit s'assurer que les valeurs les plus récentes
> de sa composante du vecteur itéré soient envoyées ;
> ensuite, il doit cesser d'envoyer ce type de message" (2.8)

pour éviter les fausses détections.

Le test d'arrêt consiste à faire circuler un message qui contient les données permettant d'évaluer la condition (2.6) parmi les β processus.



Nous appellerons ce message "*jeton*" afin d'éviter toute confusion avec les messages contenant les valeurs du vecteur itéré sur les frontières de recouvrement. Ce jeton est initialement émis par un processus dit *initiateur*, choisi arbitrairement. Ensuite, chaque processus doit le retransmettre en utilisant des réceptions non-bloquantes. L'algorithme réparti qui fait circuler le jeton s'exprime de la manière suivante :

```
•Algorithme exécuté par le i^ème processus :
Tester la réception du jeton de la part du processus (i − 1) mod β
si le jeton est reçu alors
    Mise à jour du jeton
    Soumettre la réception du jeton pour le prochain tour
    Attendre la fin de l'envoi précédent du jeton
    Soumettre l'envoi jeton au processus (i + 1) mod β
fin
```

Le jeton contient des données permettant de détecter la terminaison de l'algorithme de Schwarz asynchrone :

– un booléen indiquant si tous les processus ont localement convergé ;

– un compteur contenant la différence entre le nombre de messages envoyés et reçus par tous les processus : au moment de la réception du jeton, le processus doit compter les nombres de requêtes d'envoi et de réception qui se sont terminées depuis le dernier passage du jeton pour mettre à jour le compteur.

La convergence globale est détectée si le jeton indique que tous les processus ont atteint la convergence locale et si la réception de tous les messages envoyés est confirmée par un compteur qui vaut zéro. Dans ce cas, le processus qui a émis le jeton a la charge d'envoyer un message d'arrêt aux autres processus.

Remarque 2.7 Les envois et les réceptions du jeton et du message d'arrêt sont effectués dans le même contexte de communication que les messages. Le paramètre "*tag*" des procédures de communication permet alors de distinguer différents types de message.

L'utilisation d'un envoi bloquant permet de simplifier l'algorithme. Cela revient à empêcher les processus de garder le jeton pendant les itérations internes. Cette façon de procéder n'engendre aucun temps d'attente. En effet, supposons que le $i^{\text{ème}}$ processus reçoit le jeton pour la $t^{\text{ème}}$ fois. Avant de pouvoir transmettre le jeton au processus suivant, le $i^{\text{ème}}$ processus doit attendre la fin de la requête d'envoi du jeton associée à la $(t-1)^{\text{ème}}$ réception de celui-ci. Dans la mesure où le jeton a effectué un tour complet, cette requête d'envoi est nécessairement terminée.

L'algorithme de Schwarz parallèle asynchrone avec communication flexible avec son test d'arrêt est donné dans la figure 2.11. Les envois de message conditionnés par le critère de convergence locale reflètent la contrainte (2.7). Quant à la contrainte (2.8), elle intervient uniquement lorsque le critère de convergence locale est vérifié après la mise à jour des sous-composantes rouges et noires. Lorsque la variable "etat" vaut 1, le processus doit itérer ; lorsqu'elle vaut 0, il doit surveiller l'arrivée éventuelle d'un message. L'algorithme de la figure 2.11 ne donne pas les détails concernant la manière dont les itérations internes sont effectuées. Nous avons préféré omettre cet aspect afin de nous focaliser sur le test d'arrêt. Notons que la description des itérations internes est donnée dans la figure 2.10.

Remarque 2.8 Le nombre de tours de jeton permet de définir une condition d'arrêt pour les algorithmes de Schwarz asynchrones qui est indépendante des critères de convergence numériques liés à la solution. On peut donc arrêter les itérations parallèles asynchrones si la convergence globale n'est pas vérifiée après un certain nombre de tours complets effectués par le jeton.

2.5.2 Libération des ressources

La libération des ressources consiste essentiellement à terminer toutes les requêtes persistantes MPI utilisées, puis à détruire chaque requête avec la fonction `MPI_REQUEST_FREE()`.

• Algorithme exécuté par le $i^{\text{ème}}$ processus :
Paramètres principaux
$\hat{A}_{ii}$: $i^{\text{ème}}$ bloc diagonal de la matrice augmentée
$\hat{b}_i$: $i^{\text{ème}}$ composante du second membre augmenté
$\hat{x}_i^0$: $i^{\text{ème}}$ composante du vecteur initial

Initialiser les échanges de message et le test d'arrêt
etat $\leftarrow 1$
tant que message d'arrêt non reçu
si etat $= 1$ **alors**
Effectuer les itérations internes en recevant les messages associés à $(\tilde{\hat{x}})_{j\in \text{adj}(i)}$ sur les sous-composantes rouges
Évaluer le critère de convergence locale
si convergence locale non vérifiée **alors**
Envoyer les messages associés à $\hat{x}_i$
fin

Effectuer les itérations internes en recevant les messages associés à $(\tilde{\hat{x}})_{j\in \text{adj}(i)}$ sur les sous-composantes noires
Évaluer le critère de convergence locale
si convergence locale non vérifiée **alors**
Envoyer les messages associés à $\hat{x}_i$
sinon
Forcer l'envoi des messages récents associés à $\hat{x}_i$ qui n'ont pas été envoyés
etat $\leftarrow 0$
fin
sinon
si réception de messages associés à $(\tilde{\hat{x}}_j)_{j\in \text{adj}(i)}$ **alors**
etat $\leftarrow 1$
fin
fin
Tentative de mise à jour et de transmission du jeton
fin
Libérer les ressources acquises

FIG. 2.11 – Intégration du test d'arrêt dans l'algorithme de Schwarz parallèle asynchrone avec communication flexible

Suite à la détection de la convergence, les requêtes d'envoi et de réception des messages et du jeton ne sont pas a priori terminées. Il est nécessaire d'annuler les requêtes persistantes avec la fonction `MPI_CANCEL()` puis de les terminer avec `MPI_WAIT()`.

L'annulation et la terminaison des requêtes d'envoi doit précéder celles des requêtes de réception. Au cas où une réception est annulée avant l'envoi qui lui est associé, cela peut conduire à l'échec du mécanisme de rendez-vous qui est sous-jacent aux envois de message. Cet échec entraîne celui de l'annulation des envois sur certaines architectures. Par conséquent, une synchronisation entre tous les processus doit précéder l'annulation et la terminaison des réceptions. Elle est réalisée à l'aide de la fonction `MPI_BARRIER()`.

2.6 Étude expérimentale

2.6.1 Essais sur IBM Power 4 p690+

Cette étude de performance sur les algorithmes parallèles avec communication flexible a été effectuée sur l'IBM Power 4 p690+ de l'IDRIS (Institut du Développement et des Ressources en Informatique Scientifique).

Description matérielle

Le processeur Power 4 est un dualcore cadencé à 1.3 GHz. Un nœud p690+ comporte 32 processeurs, qui sont en réalité 16 dualcores, accédant à une mémoire partagée.

1. Les processeurs sont naturellement regroupés en couples grâce à l'architecture dualcore. Chaque couple possède une mémoire cache de niveau 2 commune.
2. Les couples de processeurs sont regroupés par 4 dans un "*multi-chip module*" qui leur permet de partager un accès direct aux mémoires caches de niveau 3.
3. Chaque nœud contient donc 4 modules interconnectés qui accèdent à une mémoire commune.
4. Les nœuds sont interconnectés via un réseau de type "*Federation*" dont la bande passante est 1.6 Gbit/s.

Le problème test

L'étude de performance consiste à résoudre une équation de convection-diffusion linéaire tridimensionnelle et stationnaire

$$\begin{cases} -\nu\,\Delta u + 0.5\,\frac{\partial u}{\partial x} + 1.5\,\frac{\partial u}{\partial y} - 0.5\,\frac{\partial u}{\partial z} + 10\,u = f \text{ sur } [0\,,\,1]^3 \\ u_{/\partial[0\,,\,1]^3} = 0 \end{cases} \tag{2.9}$$

afin d'analyser le comportement des itérations asynchrones avec communication flexible en fonction du coefficient de diffusion ν et du nombre de processeurs.

L'équation est discrétisée par différence finies avec $3,750,000$ points. Le schéma de discrétisation ainsi que l'étude de convergence sont donnés dans [51]. Le domaine est décomposé en 256 sous-domaines.

Chaque test consiste à résoudre l'équation (2.9), avec une valeur de ν donnée, à l'aide de l'algorithme synchrone puis de l'algorithme asynchrone. Le découpage du domaine reste identique quelques soient le nombre de processeurs et la valeur de ν.

Dans un premier temps nous avons évalué l'influence de la fréquence des échanges de message, réglée via le nombre d'itérations internes τ, sur le temps de restitution et la vitesse de convergence des algorithmes parallèles. Le nombre de processeurs est fixé à 16. Le programme s'exécute donc sur un seul nœud. Chaque processeur prend en charge 16 sous-domaines et on fait varier le paramètre τ. Le tableau 2.1 montre

Algorithme asynchrone, $\nu = 0.01$		
τ	temps (sec.)	nombre total d'itérations internes
1	39.4	43 343
2	30.8	49 854
4	35.6	67 040
8	52.1	108 072
16	90.3	199 136

Algorithme synchrone, $\nu = 0.01$		
τ	temps (sec.)	nombre total d'itérations internes
1	46.7	42 437
2	35.5	46 646
4	36.5	59 016
8	51.0	92 584
16	85.4	169 632

TAB. 2.1 – Réglage de la fréquence de communication

qu'un bon compromis entre vitesse de convergence et temps de restitution est obtenu pour $\tau = 2$. Cette valeur sera choisie pour les essais qui vont suivre.

Remarque 2.9 Nous pouvons constater dans le tableau 2.1 que pour $\tau \geq 8$, l'algorithme synchrone a de meilleures performances. Par conséquent l'asynchronisme n'est profitable que si la fréquence de communication est suffisamment élevée.

Le paramètre ν influe sur la vitesse de convergence des algorithmes étudiés. Lorsque $\nu = 0.01$, peu d'itérations internes sont nécessaires pour obtenir la convergence (environ $46,000$). En revanche, quand $\nu = 1$, la convergence est assez lente (environ $1,600,000$ itérations internes). Enfin, lorsque $\nu = 0.1$ nous avons affaire à un

cas intermédiaire (environ $400,000$ itérations internes). Cela nous permet d'étudier la scalabilité dans 3 contextes différents.

À titre indicatif, la méthode itérative utilisée est une méthode de Gauss-Seidel par blocs. Chaque itération interne correspond à 2 relaxations par blocs successives. Le sens de balayage change à chaque relaxation de manière à converger plus vite. Notons qu'il est possible de recevoir des messages au début de chaque relaxation.

Résultats expérimentaux

Le tableau 2.2 donne tous les temps de restitution obtenus en faisant varier le

nombre de proc.	$\nu = 0.01$		$\nu = 0.1$		$\nu = 1$	
	sync.	async.	sync.	async.	sync.	async.
1	247 (sec.)		2245 (sec.)		8681 (sec.)	
2 (1)	200	189	1618	1513	6232	5781
4 (1)	117	108	852	788	3132	2846
8 (1)	64	55	510	404	2061	1470
16 (1)	34	30	265	218	1004	796
32 (1)	22	18	155	127	577	466
64 (2)	11	9.3	83	65	317	241
128 (4)	6.8	5.1	49	35	185	133

TAB. 2.2 – Récapitulatif des temps de restitution sur le p690+ des algorithmes de Schwarz parallèles testés sur les 3 problèmes de convection-diffusion

nombre de processeurs et le coefficient ν, tout en conservant le paramètre τ constant ($\tau = 2$). Le nombre de nœuds p690+ utilisés est écrit entre parenthèses dans la colonne indiquant le nombre de processeurs. Dans tous les cas, l'algorithme asynchrone est meilleur que l'algorithme synchrone. La scalabilité des algorithmes est évaluée à l'aide des courbes d'accélération et d'efficacité des figures 2.12, 2.13 et 2.14. L'accélération est le rapport entre le temps de restitution séquentiel et parallèle. L'efficacité est l'accélération divisée par le nombre de processeurs utilisés. Globalement, l'accélération et l'efficacité de l'algorithme asynchrone sont supérieures à celles de l'algorithme synchrone. De plus, plus ν est grand, plus la parallélisation est efficace ; ce fait confirme les résultats obtenus par R. GUIVARCH dans [51] avec des équations de convection-diffusion bidimensionnelles. Cette dépendance des efficacités vis à vis du paramètre ν est due à la répartition des données, à la collecte des résultats, ainsi qu'à l'initialisation et à la libération des canaux de communication, qui font partie du surcoût de la parallélisation. Ces opérations ayant des temps d'exécution incompressibles, ces temps deviennent d'autant plus significatifs lorsque la quantité de calcul diminue. Puisque le nombre d'itérations croît avec ν, il est normal que l'efficacité augmente en conséquence.

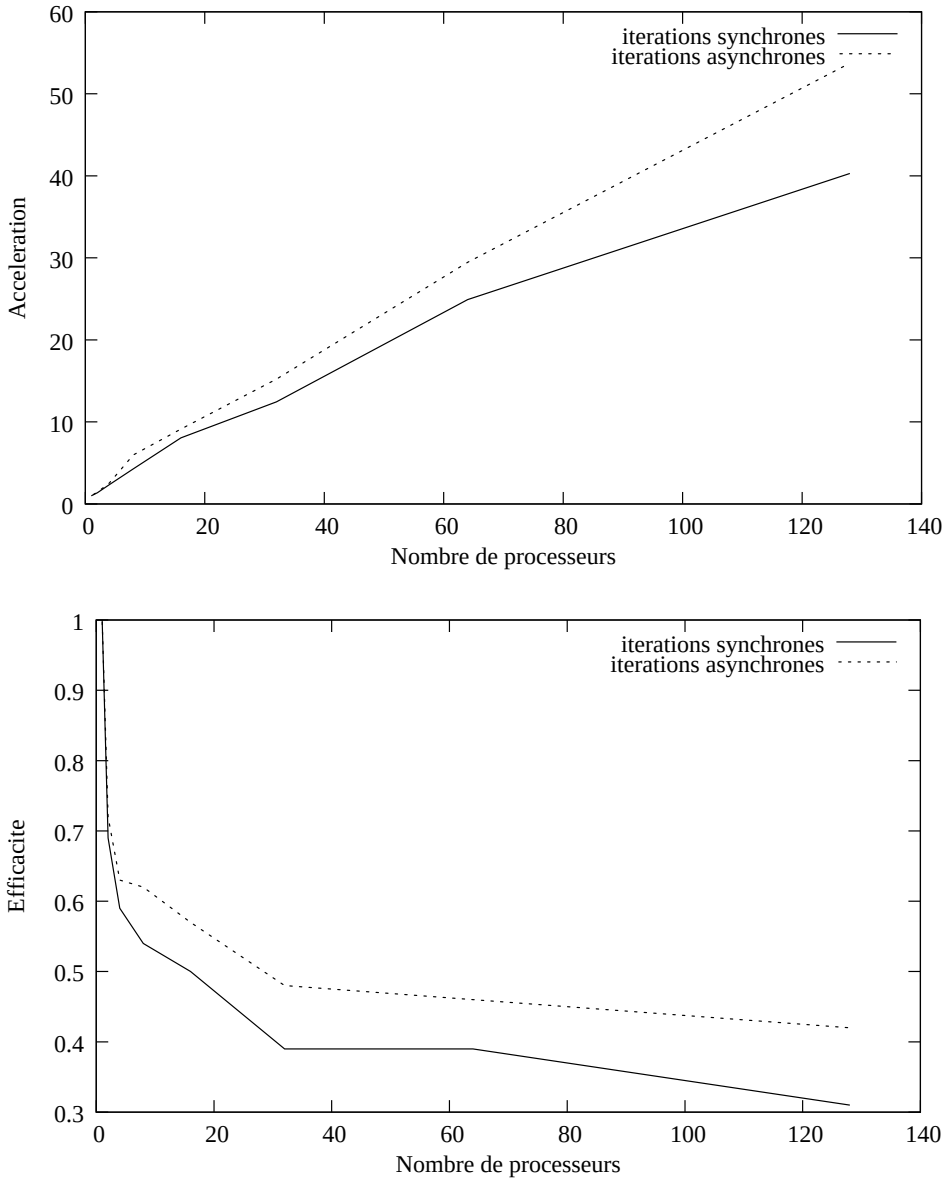


FIG. 2.12 – Accélération et efficacité des algorithmes de Schwarz parallèles pour la résolution du problème de convection-diffusion avec convection dominante ($\nu = 0.01$) sur le p690+

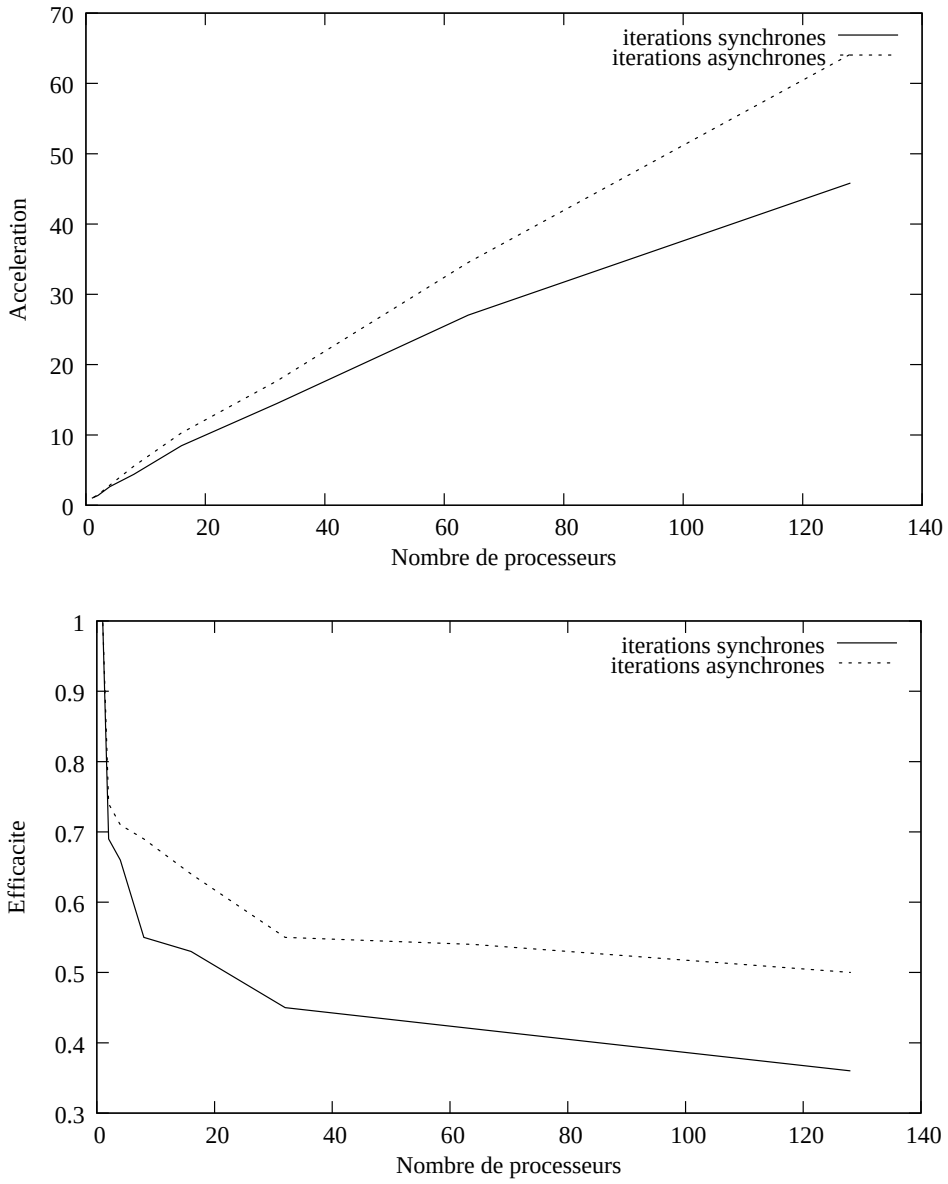


FIG. 2.13 – Accélération et efficacité des algorithmes de Schwarz parallèles pour la résolution du problème de convection-diffusion intermédiaire ($\nu = 0.1$) sur le p690+

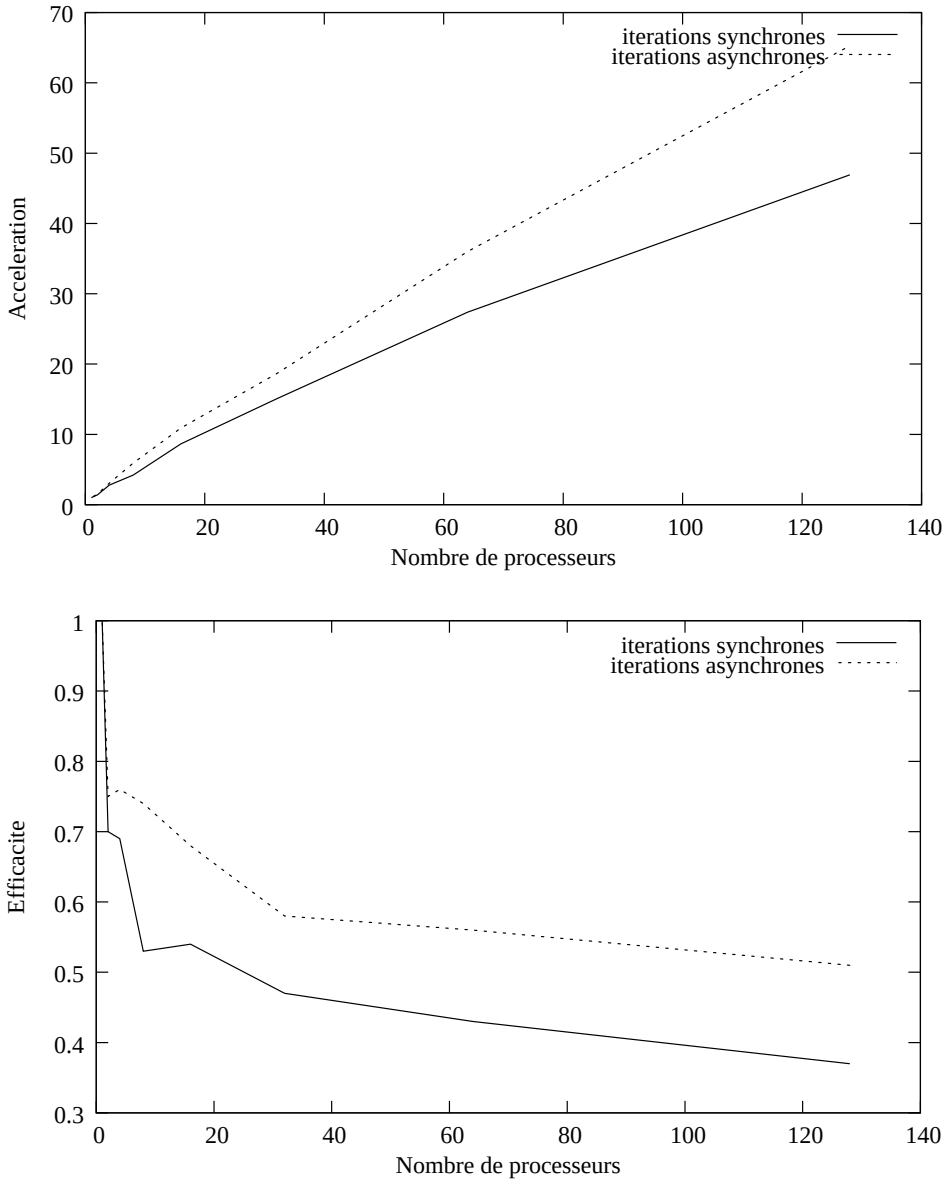


FIG. 2.14 – Accélération et efficacité des algorithmes de Schwarz parallèles pour la résolution du problème de convection-diffusion avec diffusion dominante ($\nu = 1$) sur le p690+

Pour chaque version de l'algorithme, nous pouvons noter les faits suivants.

algorithme asynchrone Pour les 3 valeurs de ν, la relation entre l'accélération et le nombre de processeurs est quasiment linéaire. Les efficacités quant à elles subissent une chute assez rapide entre 2 et 32 processeurs. Toutefois, elles décroissent très peu entre 32 et 128 processeurs.

algorithme synchrone Pour les 3 valeurs de ν, la relation entre l'accélération et le nombre de processeurs est moins linéaire que dans le cas asynchrone. À partir de 64 processeurs, l'accélération croît moins rapidement. Les courbe d'efficacité sont moins régulières que celles de l'algorithme asynchrone. En effet, lorsque $\nu = 1$, l'efficacité pour 16 processeurs est légèrement supérieure à l'efficacité pour 8 processeurs, ce qui est inhabituel.

Nous pouvons conclure que pour les systèmes algébriques de grandes dimensions issus de nos problèmes tests, les performances et la scalabilité de l'algorithme asynchrone sont supérieures à celles de l'algorithme synchrone.

En ce qui concerne l'irrégularité de l'efficacité de l'algorithme synchrone pour $\nu = 1$, nous ne disposons d'aucun élément permettant d'en trouver la cause exacte. Toutefois, nous pouvons soupçonner l'influence du découpage du problème. Sachant que la répartition des données a une influence sur les messages à échanger, il se peut que le découpage pour 16 processeurs soit plus avantageux que le découpage pour 8 processeurs. Cette hypothèse est plausible car pour les cas $\nu = 0.01$ et $\nu = 0.1$, nous pouvons noter qu'entre 8 et 16 processeurs, l'efficacité décroît moins rapidement.

Les courbes donnant le nombre d'itérations internes en fonction du nombre de processeurs sont données dans les figures 2.15, 2.16 et 2.17. Le nombre d'itérations internes effectuées par l'algorithme synchrone croît avec le nombre de processeurs en raison de changements dans l'ordre de traitement des sous-domaines. En revanche, la croissance relativement forte du nombre d'itérations internes effectuées par l'algorithme asynchrone est due au comportement chaotique. Les retards dans les échanges de message ralentissent nécessairement la convergence en terme de nombre d'itérations.

Ce surcoût en itération interne n'est pas pénalisant dans la mesure où il reste inférieur à celui qui est induit par la synchronisation. La comparaison entre les nombres d'itérations internes effectuées par les versions synchrones et asynchrones des algorithmes parallèles montre que plus le nombre de processeurs est élevé, plus l'algorithme a tendance à perdre les gains de la stratégie de relaxation rouge-noir.

Remarque 2.10 Notons que l'augmentation du nombre d'itérations internes a une incidence positive vis à vis du gain en précision [88].

2.6.2 Synthèse

En rendant l'implémentation des communications indépendante du découpage en sous-composantes, le code a gagné en simplicité au prix d'une perte en efficacité que

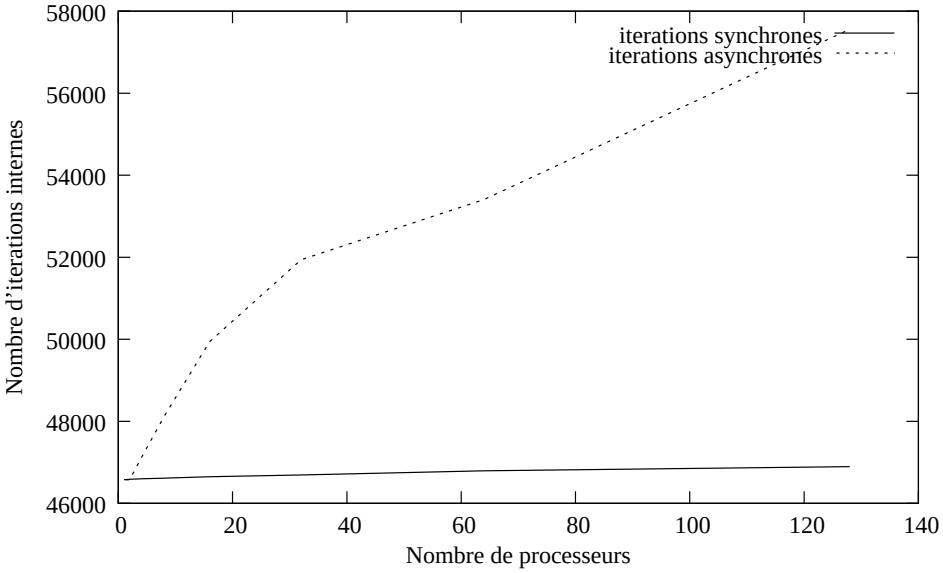


FIG. 2.15 – Comparaison entre les nombres d'itérations internes effectuées par les algorithmes de Schwarz parallèles pour la résolution du problème de convection-diffusion avec convection dominante ($\nu = 0.01$) sur le p690+

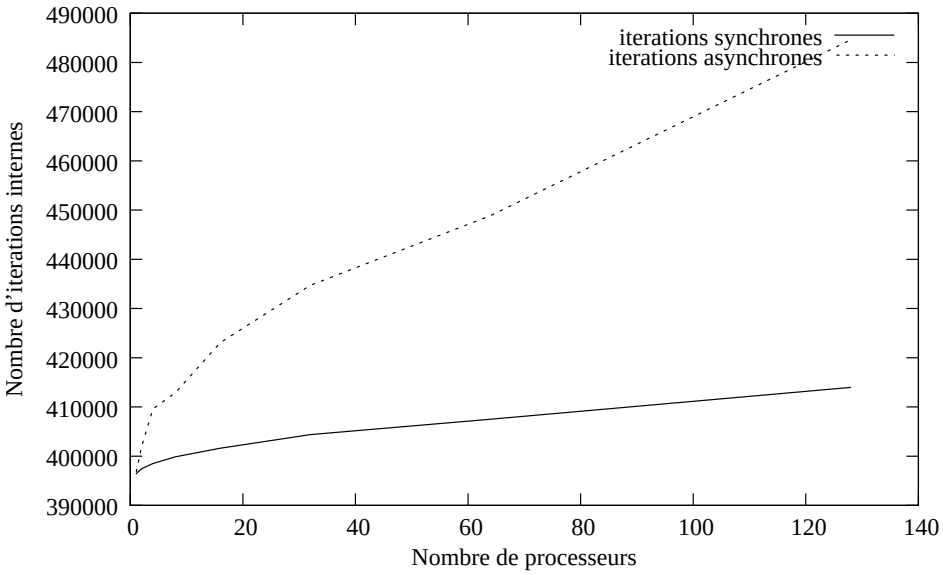


FIG. 2.16 – Comparaison entre les nombres d'itérations internes effectuées par les algorithmes de Schwarz parallèles pour la résolution du problème de convection-diffusion intermédiaire ($\nu = 0.1$) sur le p690+

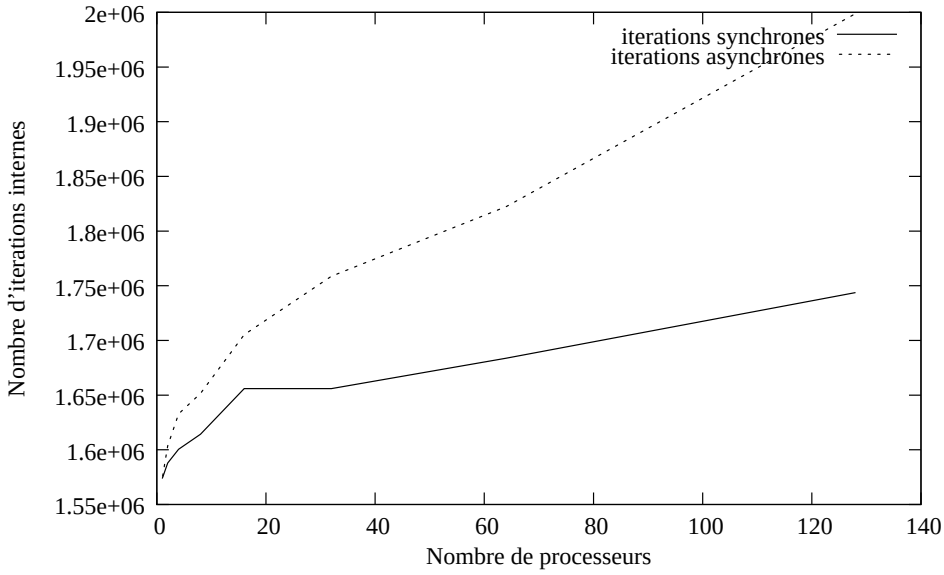


FIG. 2.17 – Comparaison entre les nombres d'itérations internes effectuées par les algorithmes de Schwarz parallèles pour la résolution du problème de convection-diffusion avec diffusion dominante ($\nu = 1$) sur le p690+

nous pouvons déceler en comparant les courbes des figures A.2 (donnée dans l'annexe A), 2.12, 2.13 et 2.14. Le fait d'optimiser les communications en prenant en compte le découpage en sous-composantes reste un principe bon d'implémentation. Cependant, cette approche a des limites :

1. Lorsque le nombre de processeurs est grand, le gain de performance obtenu grâce à l'optimisation des communications n'est pas évident. Les courbes de la figure A.2 montrent que l'écart entre les efficacités des deux algorithmes parallèles a tendance à diminuer lorsque le nombre de processeurs croît.
2. Optimiser les communications en fonction du découpage en sous-domaines ne fait que rendre un solveur parallèle tributaire d'un type de découpage.

Notons que les moyens algorithmiques qui sont mis en œuvre pour optimiser les algorithmes asynchrones ne sont pas nécessairement applicables pour les algorithmes synchrones. Les problèmes d'interblocage en sont une bonne illustration.

En fin de compte, la prise en compte de plusieurs sous-domaines par processus a pour but d'accélérer la convergence grâce à une meilleure stratégie de relaxation. Cependant, étant donné la nature chaotique des algorithmes asynchrones, le surcoût en relaxations induit par l'asynchronisme tend à limiter les effets de la stratégie de relaxation multiplicative. Donc, la tentative d'optimisation de la stratégie de relaxation n'est pas scalable. Les courbes des figures 2.15, 2.16 et 2.17 peuvent suggérer qu'à partir d'un certain nombre de processeurs, le nombre de relaxation effectuées par l'algorithme asynchrone dépassera le double du nombre de relaxations effectuées par

l'algorithme synchrone. Cela signifiera que tous les bénéfices de la stratégie rouge-noir seront perdus à partir de ce seuil.

Une voie d'investigation consisterait à accélérer la convergence de l'algorithme dans le cas où les processus ne prennent en charge qu'un seul sous-domaine. Les approches proposées dans [69, 46, 47, 98] sont envisageables pour permettre une telle accélération. Cela consiste à utiliser des algorithmes qui sont, d'un point de vue mathématique, plus performants que l'algorithme de Schwarz que nous avons implémenté. Ainsi, la mise en œuvre d'une stratégie de relaxation multiplicative, qui est à l'origine des difficultés algorithmiques, n'est pas nécessaire. Notons que la mise en œuvre sur calculateur parallèle de la variante de la méthode alternée de Schwarz proposée par J.C. Miellou [69] a été réalisée dans 2 situations distinctes [48, 51].

En ce qui concerne la réorganisation des itérations internes (*cf.* section 2.4.1), la technique employée a l'inconvénient de limiter la flexibilité des communications. En effet, une composante du vecteur itéré est inaccessible durant les τ itérations internes effectuées sans envois de message. Cette limitation n'est pas impertinente car les architectures parallèles actuellement disponibles ne tolèrent pas toujours des fréquences de communication trop élevées. Le concept d'asynchronisme demeure très intéressant même s'il n'est pas exploité dans sa totalité, dans la mesure où il apporte une grande souplesse dans la programmation des solveurs parallèles grâce à l'absence d'interblocage.

Chapitre 3

Électrophorèse de zone à écoulement continu

3.1 Introduction

Le terme électrophorèse désigne la migration de particules ionisées dans une solution électrolytique, sous l'action d'un champ électrique. La vitesse de migration $\vec{V}$ est proportionnelle au champ électrique $\vec{E}$:

$$\vec{V} = \mu \vec{E}. \tag{3.1}$$

Le coefficient μ est la mobilité électrophorétique. Elle est fonction de la température, des paramètres liés au solvant et des caractéristiques de l'espèce ionique considérée.

Le phénomène d'électrophorèse fut découvert au 19$^{\text{ème}}$ siècle, à l'occasion des études sur la théorie de ions. Il a été mis en évidence en 1897 par Arrhenius, qui a réalisé la séparation de colorants à l'aide d'un champ électrique. Les premières applications sont apparues quarante ans plus tard. Elles exploitent les différences de mobilité entre plusieurs espèces ioniques afin de réaliser leur séparation.

La séparation par électrophorèse est par la suite devenue une technique d'analyse incontournable dans des domaines tels que la biologie moléculaire et la génétique. Grâce au procédé d'électrophorèse de zone (figure 3.1), il est possible de distinguer les différentes protéines que renferment un fluide biologique.

L'électrophorèse de zone est uniquement destinée à l'analyse de mélanges d'espèces ioniques. En effet, les quantités de produits purs récupérés sont de l'ordre du microgramme. Cela est dû à l'immobilité du milieu dans lequel la séparation a lieu (gel ou milieu poreux).

L'électrophorèse de zone à écoulement continu est un procédé électrophorétique qui a été développé dans le but de séparer les constituants de mélanges en plus grande quantité. La migration électrophorétique se déroule dans un milieu liquide en écoulement continu (figure 3.2).

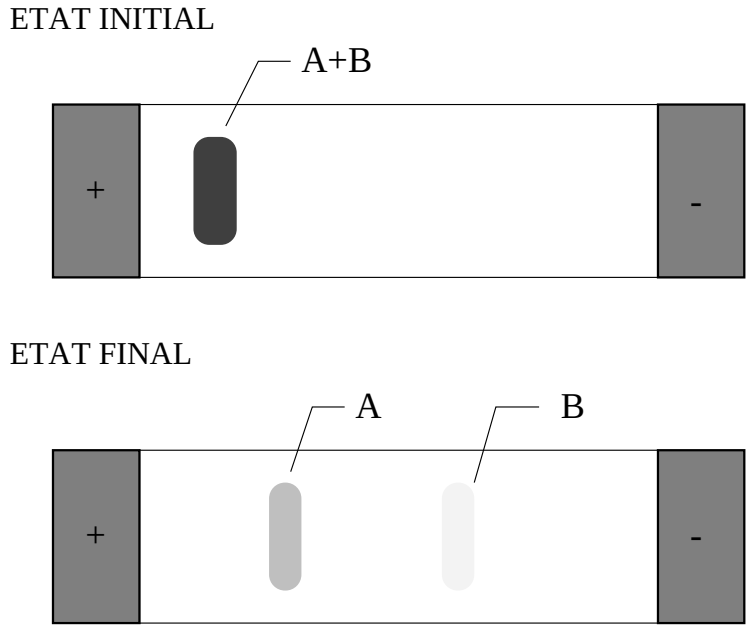


FIG. 3.1 – L'électrophorèse de zone

Le mélange d'espèces ioniques est injecté ponctuellement dans une nappe de liquide appelée *tampon vecteur*. Celui-ci s'écoule verticalement dans une chambre parallélépipédique appelée *cellule d'électrophorèse*, soumise à un champ électrique perpendiculaire au sens de l'écoulement. L'échantillon injecté dans la cellule forme un filet. Les constituants du mélange, transportés par l'écoulement imposé au tampon vecteur, subissent la migration électrophorétique pendant la traversée de la cellule. Leur séparation a lieu grâce aux différences de mobilité. À la sortie, les produits purs sont recueillis à l'aide de collecteurs disposés sur la largeur de la cellule.

L'épaisseur de la cellule est très faible devant sa largeur et sa longueur afin d'assurer un écoulement le plus stable possible. La présence de remous dans l'écoulement remélangerait les constituants séparés.

Le champ électrique est créé à l'aide d'électrodes disposées dans des compartiments situés aux extrémités latérales de la cellule. Elles sont séparées du tampon vecteur par des membranes semi-perméables, ne laissant passer que les ions, porteurs de courant électrique.

Ce procédé, simple au premier abord, est soumis à des phénomènes perturbateurs pouvant être classés en deux catégories : d'une part, les perturbations affectant le filet et d'autre part, celles qui affectent l'écoulement du tampon vecteur. Parmi les phénomènes qui rendent le filet instable, on distingue :

L'électrohydrodynamique La conductivité électrique du fluide qui s'écoule dans la cellule est non uniforme en raison de la présence d'espèces ioniques de natures

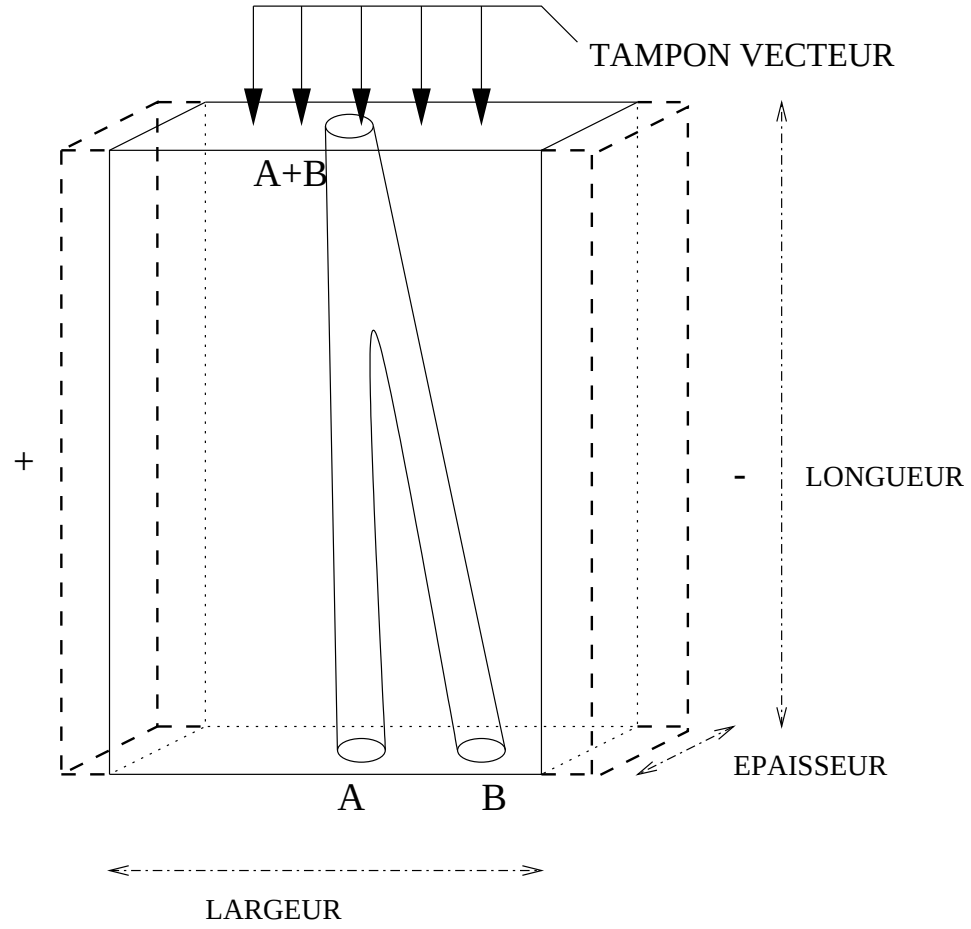


FIG. 3.2 – L'électrophorèse de zone à écoulement continu

différentes. L'écoulement du tampon vecteur est alors soumis à des perturbations au voisinage du filet.

La diffusion Les espèces ioniques se diffusent dans le tampon vecteur. Le diamètre du filet augmente en conséquence.

Le temps de séjour La vitesse d'écoulement étant plus faible au voisinage des parois de la cellule, le temps de séjour des particules ionisées proches des parois est donc plus élevé. Ainsi, les particules situées au centre de la cellule migreront moins que les particules proches des parois.

La gravité En présence d'un champ de gravité, lorsque la masse volumique d'une espèce injectée est différente de celle du tampon vecteur, le filet peut devenir instable. Des risques de fluctuation, de rupture et de sédimentation peuvent apparaître.

L'électro-osmose Ce phénomène peut être assimilé à une vitesse de glissement au niveau des parois latérales de la cellule, perpendiculairement au sens de

l'écoulement. Il est provoqué par la présence de charges à la surface du matériau formant la cellule. Ce mouvement contribue à la déformation du filet à l'instar de la différence de temps de séjour.

Les phénomènes à l'origine de l'instabilité de l'écoulement du tampon vecteur sont :

L'effet Joule Le passage du courant électrique dans la solution provoque un échauffement de celle-ci. En principe, les zones de faible concentration ionique ont tendance à être les plus chaudes.

La polarisation de concentration Au voisinage des membranes semi-imperméables qui séparent les électrodes de la solution, les ions peuvent s'y accumuler ou se raréfier.

La convection naturelle Étant donné que la concentration ionique varie au sein de la cellule à cause de la polarisation de concentration, l'échauffement par effet Joule est par conséquent non uniforme. L'écoulement forcé du tampon vecteur est alors perturbé par des courants de convection dus aux transferts thermiques.

Des études détaillées de ces phénomènes sont disponibles dans [60, 78, 1].

Ce chapitre s'articule de la manière suivante. Le modèle physique [23, 22] est présenté dans la section 3.2. Il repose sur le couplage entre :

- les équations de Navier-Stokes incompressibles qui régissent l'écoulement du tampon vecteur,
- une équation de convection-diffusion linéaire qui régit le transport des particules ionisées dans la cellule,
- et une équation de Laplace généralisée qui régit le champ de potentiel dans la cellule.

Les schémas de discrétisation de ces équations sont établis dans les sections qui suivent. La section 3.3 présente la résolution numérique des équations de Navier-Stokes à l'aide d'une méthode de type prédicteur-correcteur (PISO [57]), dans le cadre d'une discrétisation par volumes finis. La résolution numérique de l'équation de concentration discrétisée par différences finies est traitée dans la section 3.4. Deux méthodes distinctes seront envisagées : l'une implicite, l'autre explicite (MPDATA [84, 85, 86]). La section 3.5 présente le schéma de discrétisation par différences finies qui est appliqué à l'équation de potentiel. Enfin, la mise en œuvre de l'algorithme de Schwarz asynchrone avec communication flexible est décrite dans la section 3.6. Nous vérifierons d'abord que les matrices de discrétisation des équations du modèle sont des M-matrices, ce qui nous permettra d'appliquer les théorèmes de convergence [73, 49, 71, 33] des algorithmes parallèles asynchrones. Puis nous présenterons les résultats des simulations numériques ainsi que les mesures de performance obtenues sur le serveur de calcul de l'IDRIS.

3.2 Modèle physique

Le modèle tridimensionnel présenté ici est inspiré du modèle de M. J. Clifton, H. Roux de Balmann et V. Sanchez [23, 22]. La température du tampon vecteur est uniforme, les effets de la gravité sont négligés.

c	Concentration
D	Coefficient de diffusion
$\vec{E}$	Champ électrique
E_i, $i = 1, 2, 3$	Composantes du champ électrique
K	Conductivité électrique
L_i, $i = 1, 2, 3$	Dimensions (largeur, longueur, épaisseur) de la cellule
p	Pression
Q	Contribution diffusive à la densité de courant
R	Constante des gaz parfaits
T	Température
t	Temps
$\vec{U}$	Champ de vitesse
u_i, $i = 1, 2, 3$	Composantes du champ de vitesse
v_m	Vitesse moyenne
x_i, $i = 1, 2, 3$	Coordonnées spatiales dans un repère cartésien
ϵ	Permittivité diélectrique
λ	Conductivité ionique moyenne
μ	Mobilité électrophorétique
ν	Viscosité cinématique
ρ	Masse volumique
Φ	Potentiel électrique

Tab. 3.1 – Variables et paramètres du modèle physique de l'électrophorèse de zone à écoulement continu

Dans la suite, les notations adoptées pour les variables et les coefficients sont indiquées dans le tableau 3.1. Un repère cartésien (O, e_1, e_2, e_3) est associé à la cellule d'électrophorèse. Celle-ci est représentée dans la figure 3.3. Le domaine sera noté Ω.

$$\Omega = \prod_{i=1}^{3} [0, L_i]. \tag{3.2}$$

Soit f une fonction de variable $(x, y, z, t) \in \Omega \times [0, +\infty[$. Nous adopterons les notations suivantes pour les opérateurs classiques :

- le gradient d'un champ scalaire f est noté $\vec{\text{grad}}(f) = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3})$,
- la divergence d'un champ de vecteur $\vec{F}$ est notée $\text{div}(\vec{F}) = \frac{\partial f_1}{\partial x_1} + \frac{\partial f_2}{\partial x_2} + \frac{\partial f_3}{\partial x_3}$,

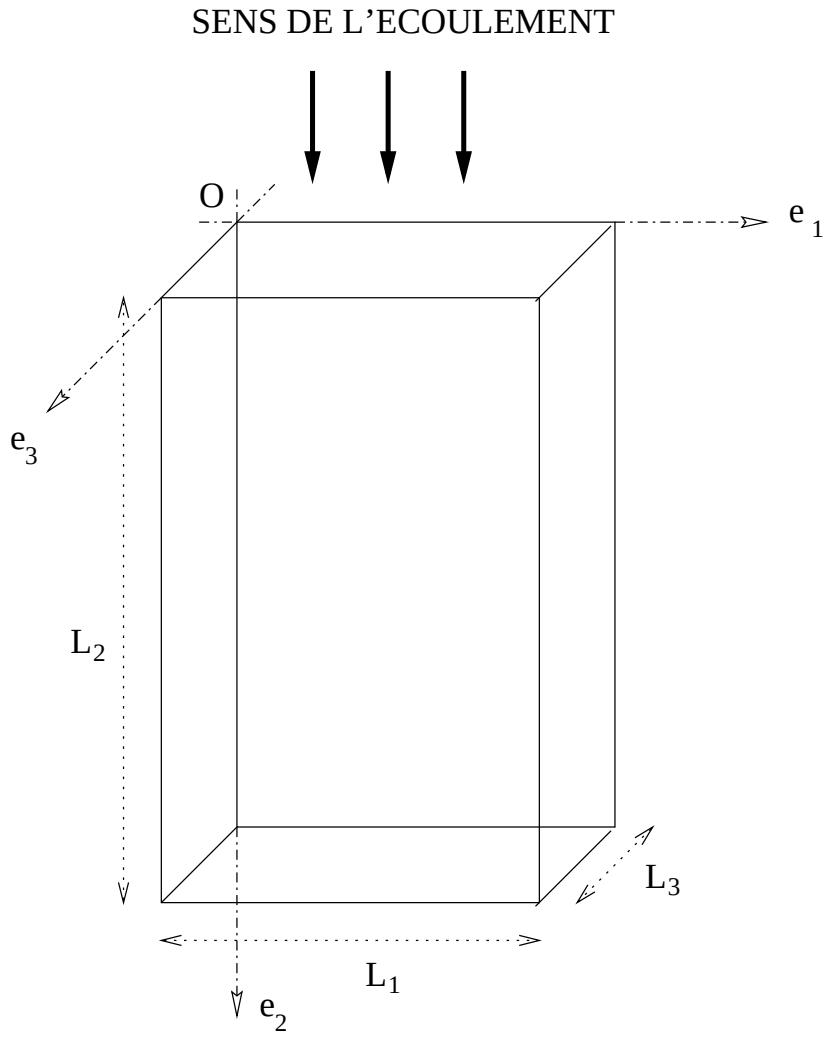


FIG. 3.3 – Géométrie de la cellule d'électrophorèse

– le Laplacien de f est noté Δf.

Les restrictions de f aux frontières du domaine seront notées :

$$f_{/x_i=0} \text{ ou } f_{/x_i=L_i},\ i = 1, 2, 3.$$

Une condition de Dirichlet sera notée :

$$f_{/x_i=0} = g \text{ ou } f_{/x_i=L_i} = g,\ i = 1, 2, 3.$$

Une condition de Neumann sera notée :

$$\frac{\partial f}{\partial n}_{/x_i=0} = g \text{ ou } \frac{\partial f}{\partial n}_{/x_i=L_i} = g,\ i = 1, 2, 3.$$

Une condition initiale sera notée :

$$f_{/t=0} = g.$$

3.2.1 Modélisation

Nous reprenons la mise en équation du procédé d'électrophorèse à écoulement continu [22] en adoptant quelques simplifications :

- l'électro-osmose sera négligée,
- la migration d'une seule espèce ionique sera considérée.

L'écoulement incompressible

L'écoulement du tampon vecteur est modélisé par une équation de Navier-Stokes incompressible tridimensionnelle. Le champ de vitesse $\vec{U}$ du tampon vecteur ainsi que la pression p sont régis par les équations quantité de mouvement :

$$\frac{\partial u_i}{\partial t} - \nu\,\Delta u_i + \sum_{j=1}^{3} u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \mathcal{F}_i,\ i = 1, 2, 3. \tag{3.3}$$

Le terme $\mathcal{F}_i$ dépend des forces extérieures. Le champ de vitesse doit aussi respecter l'équation de conservation de la masse :

$$\mathrm{div}(\vec{U}) = 0. \tag{3.4}$$

La migration électrophorétique

La concentration c d'une espèce donnée est régie par une équation de transport de la forme :

$$\frac{\partial c}{\partial t} - D\,\Delta c + \sum_{i=1}^{3} w_i\,\frac{\partial c}{\partial x_i} = 0. \tag{3.5}$$

où w_i sont les composantes de la vitesse de convection de l'espèce chimique.

L'espèce ionique est d'une part soumise à la migration électrophorétique (3.1), et d'autre part entraînée par le tampon vecteur à une vitesse d'écoulement $\vec{U}$, régie par (3.3) et (3.4). Le vecteur vitesse

$$\vec{W} = \vec{U} + \mu\,\vec{E}$$

est donc la vitesse de convection intervenant dans l'équation de transport (3.5).

Le potentiel électrique

Le potentiel électrique Φ est régi par une équation de Poisson généralisée de la forme :

$$-\mathrm{div}(K\,\vec{\mathrm{grad}}(\Phi)) = \Delta Q. \tag{3.6}$$

La fonction K est la conductivité du milieu et Q est la contribution diffusive à la densité de courant. Nous adopterons dans la suite une hypothèse simplificatrice employée dans [23, 24] :

$$\Delta Q = 0. \tag{3.7}$$

L'électrohydrodynamique

La conductivité dépend de la concentration de l'espèce ionique migrant dans le tampon vecteur :

$$K = K_0 + \lambda c.$$

La prise en compte de l'effet électrohydrodynamique consiste à coupler l'équation de potentiel (3.6) à l'équation de quantité de mouvement du tampon vecteur (3.3). Cet effet est considéré comme une force extérieure agissant sur le tampon vecteur, dont les composantes selon les axes valent :

$$\epsilon\,\mathrm{div}(E_i\,\vec{E}),\ i = 1, 2, 3.$$

Le champ électrique est déterminé par la relation $\vec{E} = -\vec{\mathrm{grad}}(\Phi)$.

Récapitulatif

L'électrophorèse de zone à écoulement continu est régi par un ensemble d'équations aux dérivées partielles résultant du couplage entre les équations associées à chacun des phénomènes physiques en jeu. Voici un récapitulatif du modèle issu de [23, 22] régissant la migration d'une espèce ionique dans la cellule d'électrophorèse :

$$\frac{\partial u_i}{\partial t} - \nu\,\Delta u_i + \sum_{j=1}^{3} u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \epsilon\,\mathrm{div}(E_i\,\vec{E}),\ i = 1, 2, 3 \tag{3.8}$$

$$\mathrm{div}(\vec{U}) = 0 \tag{3.9}$$

$$\frac{\partial c}{\partial t} - D\,\Delta c + \sum_{i=1}^{3}(u_i + \mu\,E_i)\frac{\partial c}{\partial x_i} = 0 \tag{3.10}$$

$$K = K_0 + \lambda c \tag{3.11}$$

$$-\mathrm{div}(K\,\vec{\mathrm{grad}}(\Phi)) = 0 \tag{3.12}$$

$$\vec{E} = -\vec{\mathrm{grad}}(\Phi). \tag{3.13}$$

3.2.2 Conditions aux limites

Le champ de vitesse

À l'entrée de la cellule, le vecteur vitesse est fixé par des conditions aux limites de Dirichlet :

$$\begin{aligned} u_{1/x_2=0} &= 0 \\ u_{2/x_2=0} &= v \\ u_{3/x_2=0} &= 0. \end{aligned}$$

où v est la vitesse d'écoulement du tampon vecteur injecté dans la cellule. En raison de la faible épaisseur de la cellule, nous supposerons que cette vitesse possède un profil parabolique selon l'épaisseur, de la forme

$$v = r\, x_3(L_3 - x_3).$$

Les conditions aux limites au niveau des parois latérales, associées aux composantes perpendiculaires au sens de l'écoulement sont :

$$\begin{aligned} u_{1/x_1=0} = u_{1/x_1=L_1} &= 0 \\ u_{3/x_1=0} = u_{3/x_1=L_1} &= 0 \\ u_{1/x_3=0} = u_{1/x_3=L_3} &= 0 \\ u_{3/x_3=0} = u_{3/x_3=L_3} &= 0. \end{aligned}$$

Quant à la composante u_2, les conditions aux limites sur les parois latérales sont :

$$\begin{aligned} \frac{\partial u_2}{\partial n}_{/x_1=0} = \frac{\partial u_2}{\partial n}_{/x_1=L_1} &= 0 \\ u_{2/x_3=0} = u_{2/x_3=L_3} &= 0. \end{aligned}$$

Enfin, à la sortie de la cellule, des conditions aux limites de Neumann s'appliquent à toutes les composantes :

$$\begin{aligned} \frac{\partial u_1}{\partial n}_{/x_2=L_2} &= 0 \\ \frac{\partial u_2}{\partial n}_{/x_2=L_2} &= 0 \\ \frac{\partial u_3}{\partial n}_{/x_2=L_2} &= 0. \end{aligned}$$

La concentration

Les conditions de Neumann sont prépondérantes pour l'équation de transport. La concentration est définie par une condition de Dirichlet à l'entrée de la cellule uniquement :

$$c_{/x_2=0} = c^0 \tag{3.14}$$

où c^0 est la concentration de l'espèce imposée à l'entrée de la cellule.

Au niveau des parois latérales et de la sortie, nous avons :

$$\begin{aligned} \frac{\partial c}{\partial n}_{/x_1=0} = \frac{\partial c}{\partial n}_{/x_1=L_1} &= 0 \\ \frac{\partial c}{\partial n}_{/x_3=0} = \frac{\partial c}{\partial n}_{/x_3=L_3} &= 0 \\ \frac{\partial c}{\partial n}_{/x_2=L_2} &= 0. \end{aligned}$$

Le potentiel électrique

À l'entrée de la cellule, le potentiel dépend de la concentration initiale c^0. Une condition aux limites de Dirichlet non homogène est imposée. Le potentiel à l'entrée de la cellule est la solution de l'équation de potentiel bidimensionnelle.

$$\Phi_{/x_2=0} = \Phi_0 \tag{3.15}$$

$$-\mathrm{div}(K\,\vec{\mathrm{grad}}(\Phi_0)) = 0 \text{ dans } \{x \in \Omega \mid x_2 = 0\}. \tag{3.16}$$

La fonction Φ_0 ne varie pas au cours du temps. Elle est calculée au début de la simulation.

Au niveau des électrodes, le potentiel constant se traduit par des conditions aux limites de Dirichlet :

$$\Phi_{/x_1=0} = \Phi_c \tag{3.17}$$

$$\Phi_{/x_1=L_1} = \Phi_m. \tag{3.18}$$

Le potentiel électrique étant défini à une constante près, nous imposons :

$$\Phi_c = 0. \tag{3.19}$$

Quant aux 2 autres parois latérales, le potentiel est obtenu par interpolation linéaire :

$$\Phi_{/x_3=0} = \Phi_{/x_3=L_3} = \frac{x_1}{L_1}\,\Phi_m.$$

La condition aux limites à la sortie de la cellule est définie de manière analogue à celle de l'entrée (*cf.* (3.15) et (3.16)) :

$$\Phi_{/x_2=L_2} = \Phi_t \tag{3.20}$$

$$-\mathrm{div}(K\,\vec{\mathrm{grad}}(\Phi_t)) = 0 \text{ dans } \{x \in \Omega \mid x_2 = L_2\}. \tag{3.21}$$

Contrairement à Φ_0, Φ_t est une fonction dépendante du temps. Par conséquent, à chaque pas de temps, il faut recalculer Φ_t avant de résoudre l'équation de potentiel sur la cellule.

3.2.3 Conditions initiales

Le champ de vitesse et la pression à l'instant initial doivent respecter les équations (3.3) et (3.4) avec $\mathcal{F}_i = 0$, $i = 1, 2, 3$. Compte tenu du profil parabolique de la vitesse à l'entrée de la cellule, les composantes du champ de vitesse initial valent :

$$\begin{aligned} u_{1/t=0} = u_{3/t=0} &= 0 \\ u_{2/t=0} &= \frac{6\, v_m}{(L_3)^2}\, x_3(L_3 - x_3) \end{aligned} \tag{3.22}$$

où v_m est la vitesse moyenne de l'écoulement. La pression est obtenue en intégrant la seconde équation de quantité de mouvement (3.3) :

$$p_{/t=0} = -\frac{12\, \rho\, \nu\, v_m}{(L_3)^2}\, x_2 + r \tag{3.23}$$

où r est une constante arbitraire.

Initialement, la concentration est nulle à l'intérieur de la cellule d'électrophorèse. Au niveau de l'entrée de la cellule, elle vaut c^0. Quant au potentiel électrique, il vaut :

$$\Phi_{/t=0} = \frac{x_1}{L_1}\, \Phi_m. \tag{3.24}$$

3.2.4 Étude adimensionnelle

Le passage aux variables adimensionnelles est un procédé permettant de réduire le nombre de paramètres physiques intervenant dans les équations.

Principe

Les coordonnées spatiales et temporelles doivent d'abord subir un changement d'échelle. La longueur de référence choisie est la distance L_1 séparant les 2 électrodes. Le temps nécessaire pour parcourir la distance L_1 à la vitesse v_m devient alors le temps de référence. Les fonctions de l'espace et du temps subissent par conséquent le changement de variable suivant :

$$\left\{ \begin{array}{rcll} x_1 &=& L_1\, \chi_1, & \chi_1 \in [0, 1] \\ x_2 &=& L_1\, \chi_2, & \chi_2 \in [0, \frac{L_2}{L_1}] \\ x_3 &=& L_1\, \chi_3, & \chi_3 \in [0, \frac{L_3}{L_1}] \\ t &=& \frac{L_1}{v_m}\, \tau, & \tau \in [0, +\infty[\end{array} \right. . \tag{3.25}$$

Les champs adimensionnels sont obtenus en effectuant un changement d'échelle sur les variables et les valeurs des fonctions de l'espace et du temps. Chaque fonction subit le changement de variable (3.25) ainsi qu'une division par une valeur de

référence. Par exemple, pour un champ f donné, le champ adimensionnel f^* associé s'écrit

$$f^*(\chi_1, \chi_2, \chi_3, \tau) = \frac{1}{f_0} f(L_1\,\chi_1, L_1\,\chi_2, L_1\,\chi_3, \frac{L_1}{v_m}\,\tau).$$

Les expressions des dérivées partielles de f deviennent alors :

$$\begin{aligned}
\frac{\partial f}{\partial t} &= f_0 \frac{v_m}{L_1} \frac{\partial f^*}{\partial \tau} \\
\frac{\partial f}{\partial x_i} &= \frac{f_0}{L_1} \frac{\partial f^*}{\partial \chi_i} \\
\frac{\partial^2 f}{\partial x_i^2} &= \frac{f_0}{(L_1)^2} \frac{\partial^2 f^*}{\partial \chi_i^2}.
\end{aligned}$$

Les notations associées aux opérateurs de gradient, de divergence et du Laplacien resteront inchangées.

Variables adimensionnelles

Commençons par définir les vitesses, la pression, les concentrations et le potentiel électrique adimensionnels. Posons :

$$u_i^* = \frac{u_i}{v_m},\ i = 1, 2, 3 \tag{3.26}$$

$$p^* = \frac{p}{\rho\, v_m^2} \tag{3.27}$$

$$c_\ell^* = \frac{c_\ell}{c_\ell^0} \tag{3.28}$$

$$\Phi^* = \frac{\Phi}{\Phi_m}. \tag{3.29}$$

L'expression du champ électrique adimensionnel est déduit de l'équation (3.13) :

$$E_i^* = \frac{L_1}{\Phi_m}\, E_i,\ i = 1, 2, 3. \tag{3.30}$$

La conductivité adimensionnelle s'écrit :

$$K^* = \frac{K}{\lambda\, c^0}.$$

Ce qui donne après simplification :

$$K^* = K_0^* + c^* \tag{3.31}$$

où

$$K_0^* = \frac{K_0}{\lambda\, c^0}. \tag{3.32}$$

Équations adimensionnelles

Les équations régissant la migration des espèces ioniques peuvent à présent être réécrites de manière simplifiée. Ce seront celles-là qui seront discrétisées. L'écoulement du tampon vecteur est alors régi par les équations adimensionnelles suivantes, déduites de (3.8), (3.26), (3.27) et (3.30) :

$$\frac{\partial u_i^*}{\partial \tau} - \frac{1}{\mathrm{Re}}\Delta u_i^* + \sum_{j=1}^{3} u_j^* \frac{\partial u_i^*}{\partial \chi_j} = -\frac{\partial p^*}{\partial \chi_i} + \epsilon^* \mathrm{div}(E_i^* \, \vec{E}^*),\ i = 1,2,3 \tag{3.33}$$

$$\mathrm{div}(\vec{U}^*) = 0. \tag{3.34}$$

Les expressions des constantes associées sont :

$$\mathrm{Re} = \frac{v_m \, L_1}{\nu} \text{ (nombre de Reynolds)} \tag{3.35}$$

$$\epsilon^* = \frac{\epsilon \, \Phi_m^2}{\rho \, (v_m \, L_1)^2}. \tag{3.36}$$

L'équation de transport (3.10) devient à l'aide de (3.28) et (3.30) :

$$\frac{\partial c^*}{\partial \tau} - \frac{1}{\mathrm{Pe}}\Delta c^* + \sum_{i=1}^{3} (u_i^* + m^* \, E_i^*) \frac{\partial c^*}{\partial \chi_i} = 0 \tag{3.37}$$

avec les constantes suivantes :

$$\mathrm{Pe} = \frac{v_m \, L_1}{D} \text{ (nombre de Peclet)} \tag{3.38}$$

$$m^* = \frac{\mu \, \Phi_m}{v_m \, L_1}. \tag{3.39}$$

L'équation de potentiel adimensionnelle est déduite de (3.12), (3.29) et (3.31). Elle s'écrit :

$$-\mathrm{div}(K^* \vec{\mathrm{grad}}(\Phi^*)) = 0. \tag{3.40}$$

Enfin, conformément à (3.13), (3.29) et (3.30), l'expression du champ électrique adimensionnel devient :

$$\vec{E}^* = -\vec{\mathrm{grad}}(\Phi^*). \tag{3.41}$$

Dans la suite, toutes les variables physiques seront considérées comme étant adimensionnelles afin d'alléger les notations.

3.3 Discrétisation de l'équation d'écoulement

Les équations (3.33) et (3.34) sont discrétisées par la méthode des volumes finis. Leur résolution numérique est réalisée à l'aide de l'algorithme PISO (Pressure Implicit with Splitting of Operators), mis au point par R.I. Issa [57]. Elle permet la

résolution des équations de Navier-Stokes incompressibles et compressibles. Le choix de cet algorithme est motivé par les raisons suivantes :

- le schéma de discrétisation temporelle associé à l'algorithme PISO est inconditionnellement stable [57] ;
- la mise en œuvre de l'algorithme PISO conduit à la résolution de systèmes linéaires ayant des propriétés intéressantes dans l'optique de l'utilisation des itérations parallèles asynchrones.

3.3.1 Maillages décalés

L'utilisation de maillages décalés est motivée par la résolution numérique de l'équation de Navier-Stokes. Le fait de calculer toutes les variables physiques sur un maillage commun conduit classiquement à des solution numériques physiquement irréalistes (*cf.* les exemples de [77]).

Cette technique consiste à définir 4 maillages distincts. Le champ de pression est calculé sur un maillage dit *principal* et chaque composante du champ de vitesse sur un maillage dit *décalé*, de sorte qu'en aucun point du domaine, on ne peut connaître directement les valeurs de toutes les grandeurs physiques. Les valeurs manquantes sont obtenues par interpolation.

Les points des maillages décalés sont placés au milieu des points adjacents du maillage principal (figure 3.4). La figure 3.5 montre une vue en coupe du $i^{\text{ème}}$ maillage décalé par rapport au maillage principal, dans un plan (O, e_i, e_j), avec $j \neq i$ (e_1, e_2 et e_3 étant les vecteurs de la base canonique de $\mathbb{R}^3$). Seule la $i^{\text{ème}}$ composante du champ de vitesse sera directement connue sur le $i^{\text{ème}}$ maillage décalé.

De manière générale, les champs scalaires tels que la pression, les concentrations ou le potentiel électrique sont calculés sur le maillage principal. Quant aux champs vectoriels tels que la vitesse d'écoulement ou le champ électrique, ils sont calculés sur les maillages décalés ; la $i^{\text{ème}}$ composante étant associée au $i^{\text{ème}}$ maillage décalé.

3.3.2 Volumes de contrôle

Nous effectuons à présent la mise en place des volumes de contrôle, étape préliminaire à la discrétisation par volumes finis.

Un volume de contrôle contenant un point du $i^{\text{ème}}$ maillage décalé est représenté à la figure 3.6. C'est un parallélépipède dont la longueur des arêtes dépend de la position des points adjacents au point considéré. La vue en coupe de la figure 3.7 permet de définir les dimensions du volume de contrôle sans ambiguïté.

Un volume de contrôle du maillage principal est représenté à la figure 3.8. Quant aux dimensions du volume de contrôle, elles sont définies dans la vue en coupe, figure 3.9.

Conformément aux figures 3.7 et 3.9, les notation suivantes seront adoptées :

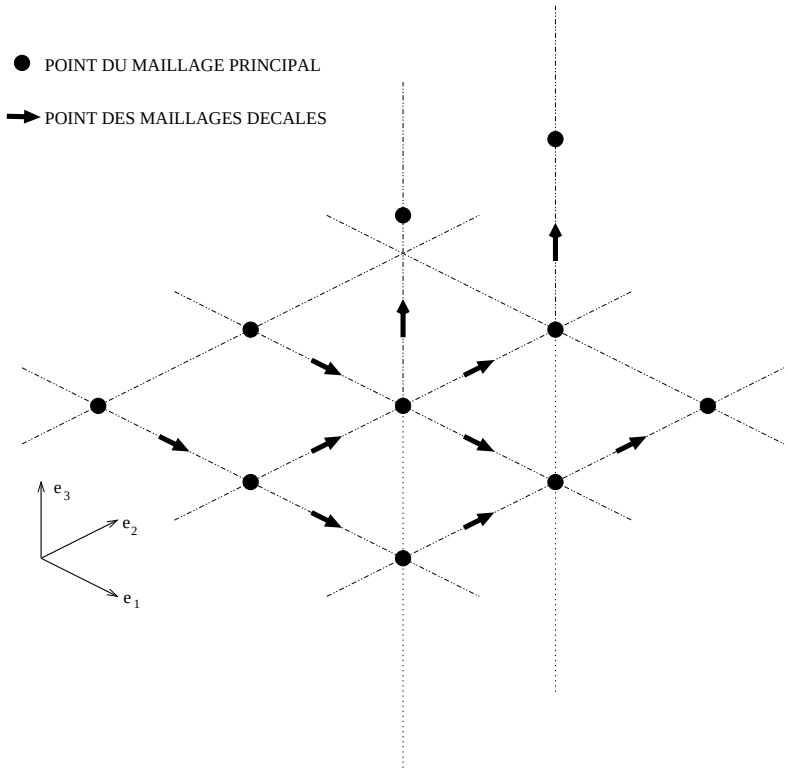


FIG. 3.4 – Vue 3D des maillages décalés

- δx_1^i, δx_2^i et δx_3^i désigneront les longueurs des côtés des volumes de contrôle. L'exposant i fait référence au maillage. Pour le maillage principal, i vaut 0. Quant aux maillages décalés, i peut prendre les valeurs 1, 2 ou 3. Enfin, l'indice inférieur fait référence à un des axes du repère cartésien. Par exemple, δx_2^3 représente la longueur du côte parallèle à l'axe (O, e_2) d'un volume de contrôle du $3^{\text{ème}}$ maillage décalé.
- hx_{k-}^i et hx_{k+}^i, $k = 1, 2, 3$, désigneront les distances entre le point situé au centre d'un volume de contrôle et les points situés dans le voisinage du volume considéré. L'exposant i fait encore référence au maillage. Par exemple, hx_{1-}^2 désigne la distance entre le point M, situé au centre du volume de contrôle, et le point de la droite (M, e_1) qui précède M. De plus, les points considérés font partie du second maillage décalé.

Nous adopterons également une convention pour désigner les points importants qui se trouvent au voisinage d'un volume de contrôle. Les notations adoptées sont définies dans la figure 3.10. Les noms des points sont choisis en fonction de leur position relative par rapport au volume de contrôle : ***w**est, **e**ast, **s**outh, **n**orth, **b**ottom, **t**op, **m**iddle*. Par exemple, si on note P le champ de pression, P_E est la valeur de cette dernière au point de discrétisation E situé à l'*est* d'un volume de contrôle et

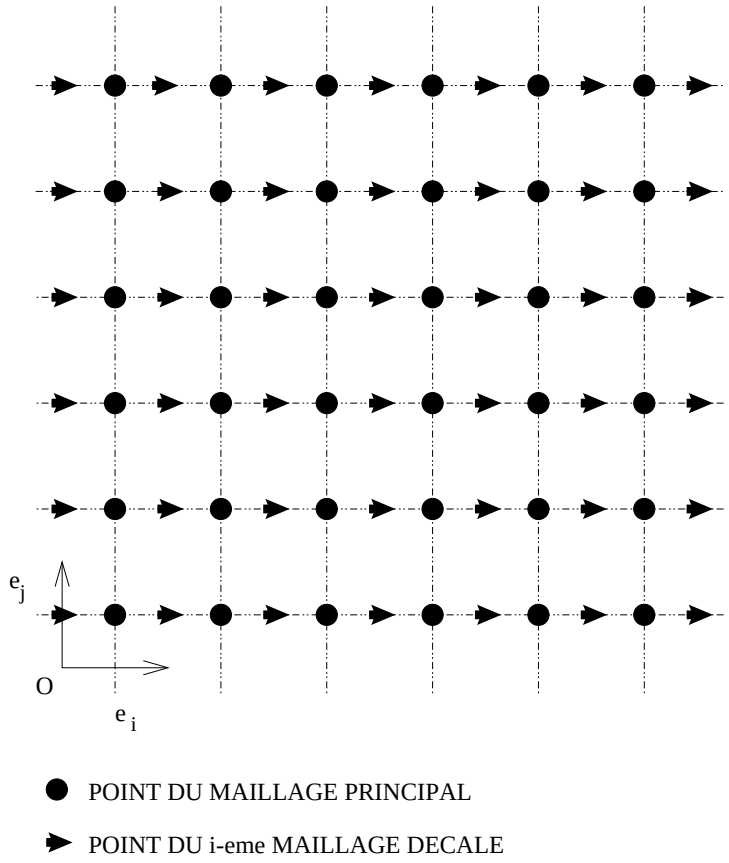


FIG. 3.5 – Vue en coupe du $i^{\text{ème}}$ maillage décalé par rapport au maillage principal

P_e est la valeur la pression sur la face *est* d'un volume de contrôle.

3.3.3 Rappel de l'algorithme PISO

Le schéma implicite associé au problème d'écoulement se présente sous la forme suivante :

$$A_i.U_i^{n+1} = -\Delta_i^i.P^{n+1} + S_i(\vec{E}, U_i^n),\ i = 1, 2, 3 \tag{3.42}$$

$$\sum_{i=1}^{3} \Delta_i^0.U_i^{n+1} = 0. \tag{3.43}$$

La $i^{\text{ème}}$ équation de quantité de mouvement, associée à la $i^{\text{ème}}$ composante du champ de vitesse, est discrétisée sur le $i^{\text{ème}}$ maillage décalé, tandis que l'équation de conservation de la masse est discrétisée sur le maillage principal. En ce qui concerne les notations employées :

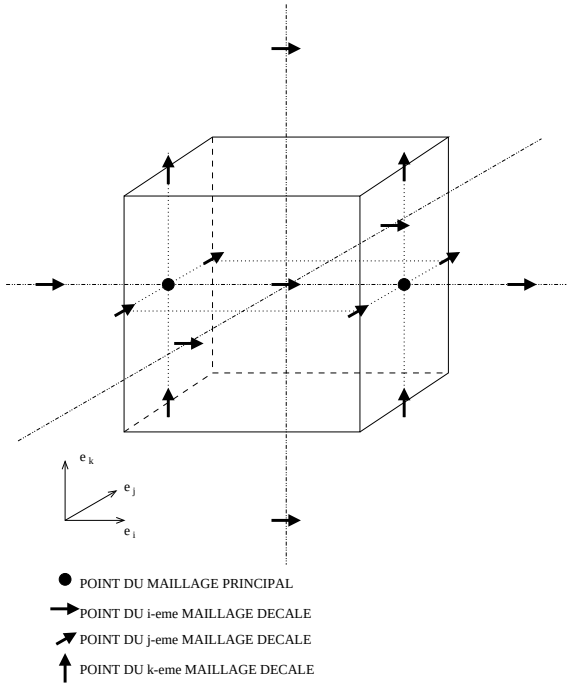


FIG. 3.6 – Vue 3D d'un volume de contrôle du $i^{\text{ème}}$ maillage décalé

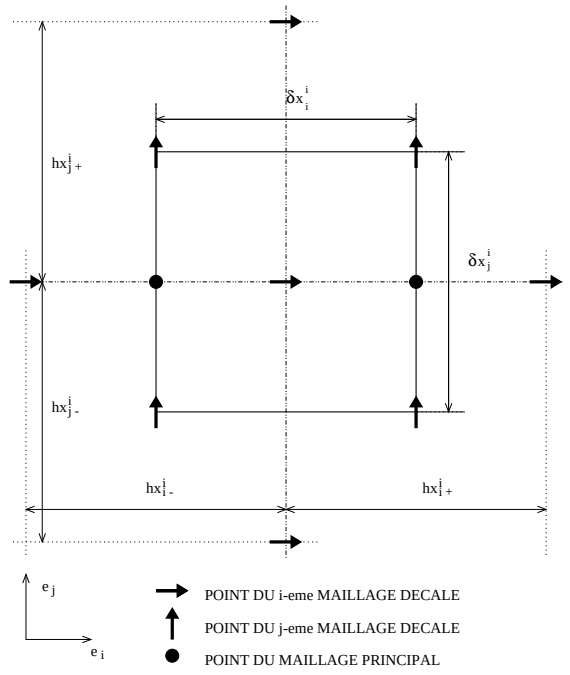


FIG. 3.7 – Vue en coupe d'un volume de contrôle du $i^{\text{ème}}$ maillage décalé

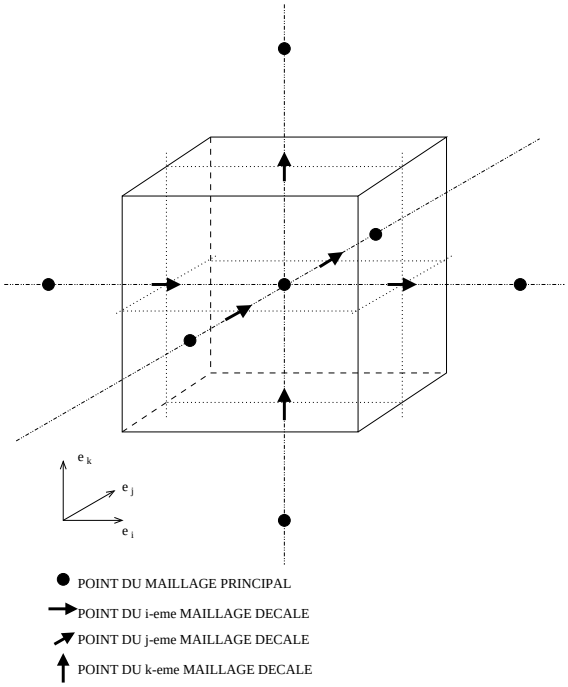


FIG. 3.8 – Vue 3D d'un volume de contrôle du maillage principal

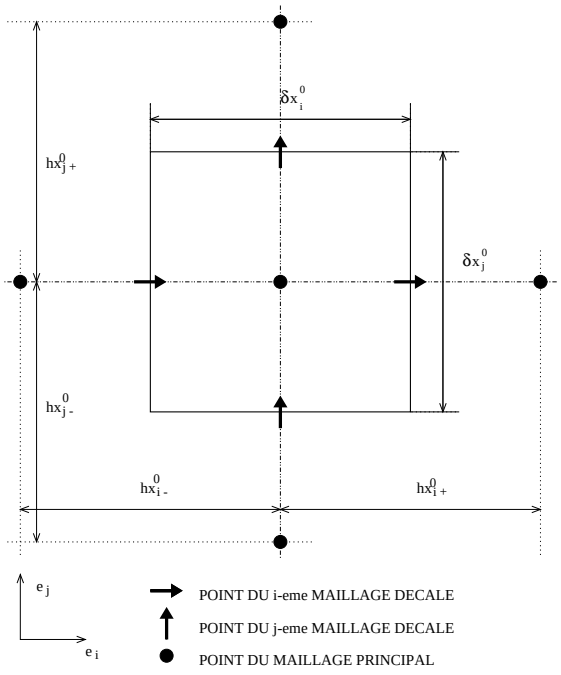


FIG. 3.9 – Vue en coupe d'un volume de contrôle du maillage principal

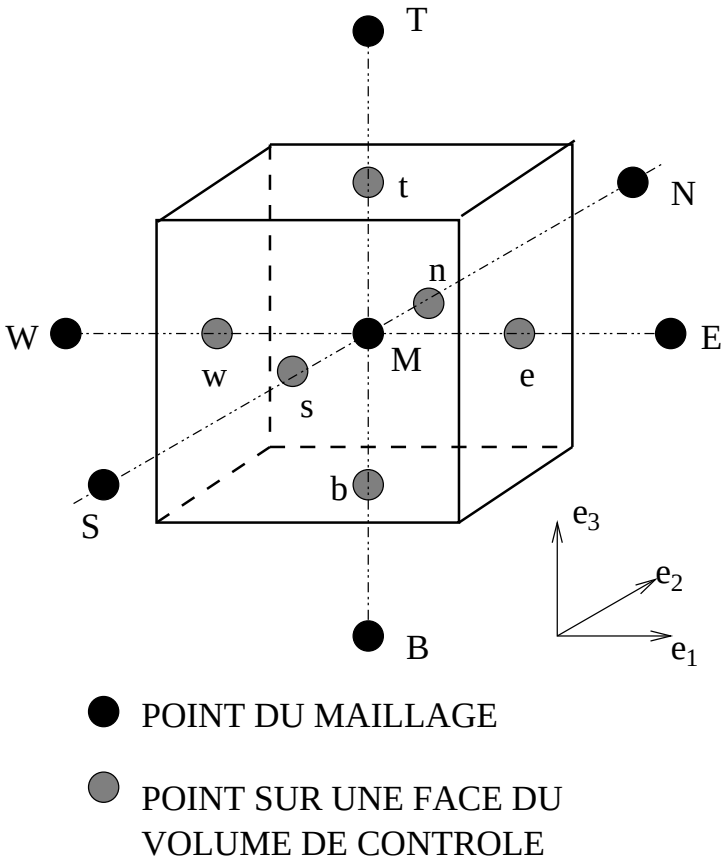


FIG. 3.10 – Vue 3D des points au voisinage d'un volume de contrôle

- $(A_i)_{i=1,2,3}$ sont les matrices associées à la discrétisation par volumes finis des équations de quantité de mouvement (3.33).
- Les champs de vitesse et de pression discrétisés associés au $n^{\text{ème}}$ pas de temps sont respectivement notés $(U_i^n)_{i=1,2,3}$ et P^n.
- Les opérateurs Δ_j^i ($i = 0, 1, 2, 3$ et $j = 1, 2, 3$) sont les opérateurs discrets de dérivation partielle $\frac{\partial}{\partial x_j}$, obtenus via la méthode des volumes finis. L'exposant i fait référence au $i^{\text{ème}}$ maillage du domaine.
- Quant aux vecteurs $(S_i)_{i=1,2,3}$, ils contiennent les contributions respectives des forces extérieures et de la discrétisation des dérivées partielles par rapport au temps. Les forces extérieures dépendent du champ électrique $\vec{E}$ et le schéma d'Euler implicite fait intervenir la vitesse $\vec{U}$ du $n^{\text{ème}}$ pas de temps.

La principale difficulté dans la résolution numérique de l'équation d'écoulement incompressible par une méthode implicite réside dans le fait qu'il n'y a pas de relation physique permettant de déterminer directement la pression.

Le principe de l'algorithme PISO consiste à traiter le couplage entre le champ de vitesse et la pression en divisant chaque pas de temps en 3 sous-pas.

1. Le pas prédicteur consiste à résoudre les équations de quantité de mouvement dans lesquelles seules les vitesses évoluent. Il en résulte un champ de vitesse approché $(U_i^{n+\frac{1}{3}})_{i=1,2,3}$ qui ne respecte pas a priori le principe de conservation de la masse.
2. Un premier pas correcteur permet de corriger les champs de vitesse et de pression de manière à obtenir une conservation approchée de la masse. Il en résulte un champ de pression approché $P^{n+\frac{1}{2}}$, puis un champ de vitesse $(U_i^{n+\frac{2}{3}})_{i=1,2,3}$.
3. Un second pas correcteur, dont le principe est identique à celui du pas correcteur précédent, permet d'améliorer le degré d'approximation. R.I. Issa a montré que deux pas correcteurs suffisent pour assurer la convergence de l'algorithme [57]. Les champs P^{n+1} et $(U_i^{n+1})_{i=1,2,3}$ sont ainsi obtenus.

Voici un rappel succinct de la méthode PISO. Nous renvoyons à [57] pour une présentation détaillée. Le pas prédicteur donne lieu à la résolution numérique des systèmes linéaires suivants :

$$A_i.U_i^{n+\frac{1}{3}} = -\Delta_i^i.P^n + S_i(\vec{E}, U_i^n),\ i = 1, 2, 3. \tag{3.44}$$

Les pas correcteurs sont basés sur une approximation permettant d'obtenir des expressions explicites des vitesses en fonction du gradient de pression. En considérant les décomposition de la matrice A_i sous la forme

$$A_i = D_i - H_i \tag{3.45}$$

où D_i représente la diagonale de A_i, l'approximation en question peut se formuler comme suit :

$$A_i.U_i^{n+\frac{2}{3}} \simeq D_i.U_i^{n+\frac{2}{3}} - H_i.U_i^{n+\frac{1}{3}} \tag{3.46}$$

$$A_i.U_i^{n+1} \simeq D_i.U_i^{n+1} - H_i.U_i^{n+\frac{2}{3}}. \tag{3.47}$$

Chacun des pas correcteurs est alors régi par les équations suivantes, déduites des approximations (3.46) et (3.47) :

1. Premier pas correcteur :

$$D_i.U_i^{n+\frac{2}{3}} = H_i.U_i^{n+\frac{1}{3}} - \Delta_i^i.P^{n+\frac{1}{2}} + S_i(\vec{E}, U_i^n),\ i = 1, 2, 3 \tag{3.48}$$

$$\sum_{i=1}^{3} \Delta_i^0.U_i^{n+\frac{2}{3}} = 0. \tag{3.49}$$

2. Second pas correcteur :

$$D_i.U_i^{n+1} = H_i.U_i^{n+\frac{2}{3}} - \Delta_i^i.P^{n+1} + S_i(\vec{E}, U_i^n),\ i = 1,2,3 \quad (3.50)$$

$$\sum_{i=1}^{3} \Delta_i^0.U_i^{n+1} = 0. \quad (3.51)$$

Les termes $S_i(\vec{E}, U_i^n)$ sont éliminés des équations (3.48) et (3.50) en retranchant (3.44) à (3.48) et (3.48) à (3.50), de manière à obtenir des équations indépendantes des forces extérieures :

$$\begin{aligned} U_i^{n+\frac{2}{3}} &= U_i^{n+\frac{1}{3}} - D_i^{-1}.\Delta_i^i.(P^{n+\frac{1}{2}} - P^n) \\ U_i^{n+1} &= U_i^{n+\frac{2}{3}} + D_i^{-1}.(H_i.(U_i^{n+\frac{2}{3}} - U_i^{n+\frac{1}{3}}) - \Delta_i^i.(P^{n+1} - P^{n+\frac{1}{2}})). \end{aligned}$$

Ces expressions explicites de $U_i^{n+\frac{2}{3}}$ et de U_i^{n+1} se substituent alors aux variables vitesses dans les équations de conservation de la masse (3.49) et (3.51). Les seules inconnues qui subsistent dans ces dernières sont $P^{n+\frac{1}{2}}$ et P^{n+1}. L'algorithme PISO se présente alors sous la forme suivante :

1. Pas prédicteur : calcul de $(U_i^{n+\frac{1}{3}})_{i=1,2,3}$ vérifiant

$$A_i.U_i^{n+\frac{1}{3}} = -\Delta_i^i.P^n + S_i(\vec{E}, U_i^n). \quad (3.52)$$

2. Premier pas correcteur : calcul de $P^c = P^{n+\frac{1}{2}} - P^n$ et de $(U_i^{n+\frac{2}{3}})_{i=1,2,3}$ vérifiant

$$-\sum_{i=1}^{3} \Delta_i^0.D_i^{-1}.\Delta_i^i.P^c = -\sum_{i=1}^{3} \Delta_i^0 U_i^{n+\frac{1}{3}} \quad (3.53)$$

$$U_i^{n+\frac{2}{3}} = U_i^{n+\frac{1}{3}} - D_i^{-1}.\Delta_i^i.P^c. \quad (3.54)$$

3. Second pas correcteur : calcul de $P^{cc} = P^{n+1} - P^{n+\frac{1}{2}}$ et de $(U_i^{n+1})_{i=1,2,3}$ vérifiant

$$-\sum_{i=1}^{3} \Delta_i^0.D_i^{-1}.\Delta_i^i.P^{cc} = -\sum_{i=1}^{3} \Delta_i^0.D_i^{-1}.H_i.(U_i^{n+\frac{2}{3}} - U_i^{n+\frac{1}{3}}) \quad (3.55)$$

$$U_i^{n+1} = U_i^{n+\frac{2}{3}} + D_i^{-1}.(H_i.(U_i^{n+\frac{2}{3}} - U_i^{n+\frac{1}{3}}) - \Delta_i^i.P^{cc}). \quad (3.56)$$

Notons que l'équation de pression (3.55) tient compte du fait que $\sum_i U_i^{n+\frac{2}{3}} = 0$.

Les variables intermédiaires P^c et P^{cc} sont appelées *incréments de pression*. Les équations (3.53) et (3.55) sont usuellement appelées *équations de correction de pression*.

3.3.4 Conditions aux limites

Le traitement des conditions aux limites de l'équation d'écoulement est l'un des aspects les plus délicats dans la mise en œuvre de l'algorithme PISO. Les équations de correction de pression (3.53) et (3.55) sont à l'origine de ces difficultés.

Pas prédicteur

Commençons par examiner le traitement des conditions aux limites au niveau du pas prédicteur. Passer en revue toutes les conditions aux limites pour les 3 composantes du champ de vitesse, sur les 6 faces de la cellule d'électrophorèse serait un travail répétitif. Nous nous contentons d'isoler deux configurations importantes :

1. la composante du vecteur vitesse est tangente à la paroi (figure 3.11)

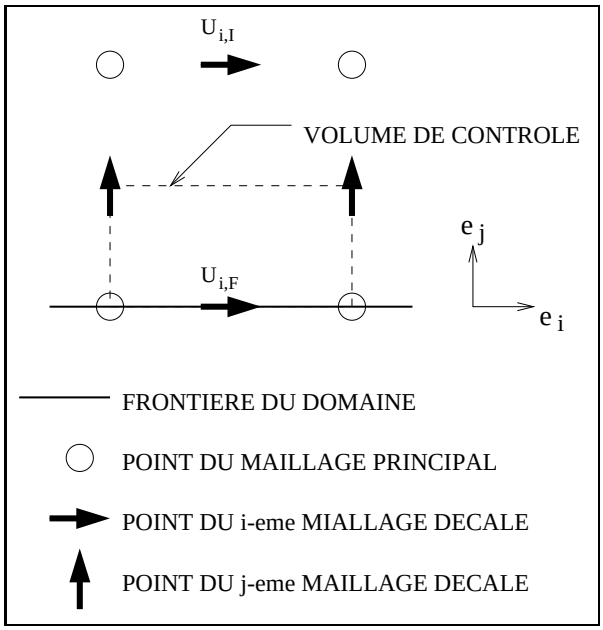


FIG. 3.11 – Volume de contrôle adjacent à la frontière associé à une vitesse tangente à la paroi

2. la composante du vecteur vitesse est normale à la paroi (figure 3.12)

Pour des raisons de lisibilité, les volumes de contrôle sont représentés sur les figures en vues de coupe. $U_{i,F}$ et $U_{i,I}$ désignent respectivement les valeurs de U_i sur la frontière et à l'intérieur du domaine.

Au niveau du pas prédicteur, la condition de Neumann nulle $\frac{\partial u_i}{\partial n} = 0$ se discrétise dans les deux configurations de manière identique :

$$U_{i,F}^{n+\frac{1}{3}} - U_{i,I}^{n+\frac{1}{3}} = 0. \tag{3.57}$$

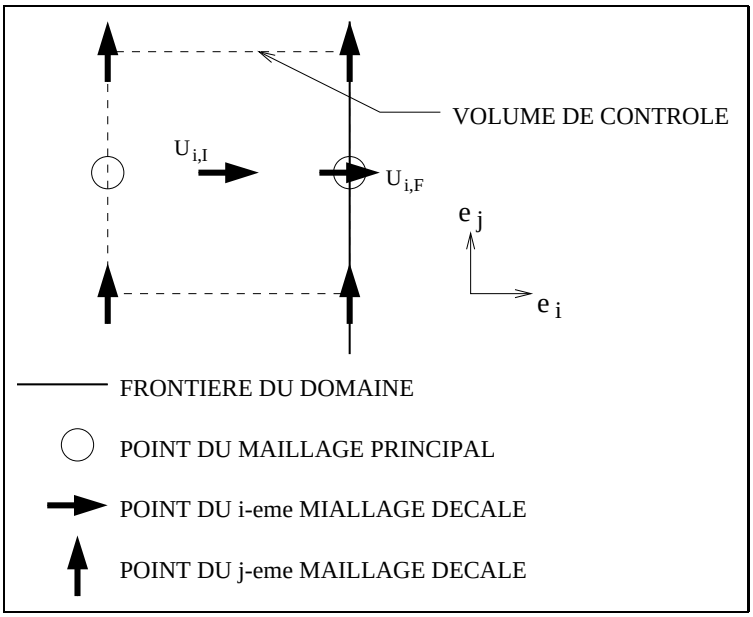


FIG. 3.12 – Volume de contrôle adjacent à la frontière associé à une vitesse normale à la paroi

Dans la configuration avec vitesse tangente à la paroi, la condition de Dirichlet $u_i = v$ se discrétise comme suit :

$$U_{i,F}^{n+\frac{1}{3}} = v. \tag{3.58}$$

Quant à la configuration avec vitesse normale à la paroi, la condition de Dirichlet peut être discrétisée de deux manières différentes :

1. fixer la valeur au niveau de la frontière

$$U_{i,F}^{n+\frac{1}{3}} = v, \tag{3.59}$$

2. ou fixer la valeur au milieu du volume de contrôle adjacent à la frontière en plus de la valeur à la frontière

$$U_{i,I}^{n+\frac{1}{3}} = U_{i,F}^{n+\frac{1}{3}} = v. \tag{3.60}$$

La solution retenue pour discrétiser les conditions de Dirichlet dans la configuration ci-dessus sera la méthode (3.60). En effet, nous verrons dans la suite que le choix de la méthode (3.59) est incompatible avec la formulation des pas correcteurs de l'algorithme PISO.

Pas correcteurs

Les conditions aux limites sur les vitesses n'évoluent pas au cours des étapes successives de l'algorithme PISO. L'exposé qui suit est valable pour les deux pas correcteurs.

Nous pouvons constater que les équations des pas correcteurs sont de la même forme que des équations de Poisson généralisées, dont les inconnues sont les incréments de pression. Cependant, la pertinence de cette analogie devient discutable lorsqu'on se focalise sur le sens physique des schémas de discrétisation, en laissant les propriétés purement mathématiques au second plan. En effet, la discrétisation par volumes finis selon [77] de l'équation $\mathrm{div}(k\,\vec{\mathrm{grad}}(u)) = f$, où k est une constante, conduit à une matrice dont les coefficients hors-diagonaux sont de la forme :

$$\frac{k\prod_{j\neq i}\delta x_j^0}{hx_{i\pm}^0}.$$

Or, les coefficients hors-diagonaux de la matrice associée aux équations de correction de pression sont de la forme :

$$k\,(\prod_{j\neq i}\delta x_j^0)\times(\prod_{j\neq i}\delta x_j^i).$$

Ce schéma de discrétisation n'est pas assimilable à un problème de Poisson. Il est donc dangereux de se fier à de telles analogies.

Les équations de correction de pression sont en réalité des équations algébriques déduites des équations de conservation de la masse (3.49) et (3.51), en substituant les expressions (3.54) et (3.56) aux vitesses. Par conséquent, les conditions aux limites des équations de correction de pression doivent découler des conditions aux limites sur les vitesses. Il faut donc se baser sur les relations entre vitesse et pression dans les volumes de contrôle situés au niveau des frontières du domaine.

Les conditions aux limites sur les incréments de pression sont obtenues en prolongeant les corrections de vitesses (3.54) et (3.56), valables a priori sur l'intérieur du domaine, vers les frontières du domaine. Le fait d'imposer des valeurs particulières au gradient de l'incrément de pression permet d'obtenir des corrections de vitesses en adéquation avec les conditions aux limites du problème d'écoulement. Notamment, imposer un gradient d'incrément de pression nul laissera la vitesse invariante. Cependant, nous constatons dans la figure 3.13 que P_F^c et P_I^c servent essentiellement à corriger la vitesse $U_{i,I}$. Par conséquent, imposer une contrainte sur le gradient d'incrément de pression dans le but de satisfaire une condition aux limites sur $U_{i,F}$ contribue à fausser la correction de vitesse. C'est la raison qui nous a poussé à fixer à la fois $U_{i,I}$ et $U_{i,F}$ lorsque la condition aux limites est de type Dirichlet, dans la configuration où U_i est normale à la frontière.

Le traitement de la condition de Dirichlet consiste à introduire une frontière artificielle (*cf.* figure 3.13) en déplaçant légèrement la frontière réelle vers l'intérieur

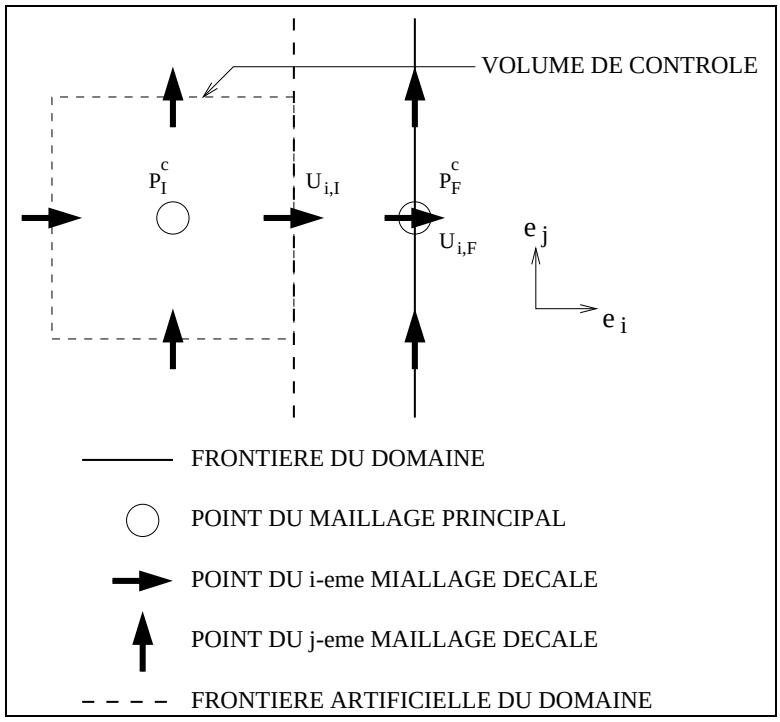


FIG. 3.13 – Volume de contrôle adjacent à la frontière associé aux équations de correction de pression

du domaine. Puis on impose la condition suivante au niveau de la frontière artificielle (*cf.* figure 3.13) :

$$U_{i,I}^{n} = U_{i,I}^{n+\frac{1}{3}} = U_{i,I}^{n+\frac{2}{3}} = U_{i,I}^{n+1}. \tag{3.61}$$

À l'aide des relations (3.54) et (3.56) liant les vitesses aux incréments de pression, nous en déduisons :

$$\Delta_i^i P^c = \Delta_i^i P^{cc} = 0. \tag{3.62}$$

Ainsi, si U_i est une composante du champ de vitesse normale à la frontière, la condition de Dirichlet devient une condition portant sur la $i^{\text{ème}}$ composante du gradient d'incrément de pression. En adoptant les notations de la figure 3.13, cette condition (3.62) devient tout simplement :

$$P_F^c - P_I^c = P_F^{cc} - P_I^{cc} = 0. \tag{3.63}$$

Quant aux vitesses sur les frontières réelles, elles ont les mêmes valeurs que sur les frontières artificielles :

$$U_{i,F}^{n} = U_{i,F}^{n+\frac{1}{3}} = U_{i,F}^{n+\frac{2}{3}} = U_{i,F}^{n+1}. \tag{3.64}$$

En ce qui concerne les conditions de Neumann, à l'instar du cas de figure précédent, il est inutile de chercher une relation entre $U_{i,F}$ et l'incrément de pression. La

condition (3.57) ne peut pas être satisfaite à l'aide d'une contrainte sur l'incrément de pression. Par conséquent, il faut ajouter au niveau de chaque pas correcteur une correction de vitesse *ad-hoc* pour $U_{i,F}$:

$$U_{i,F}^{n+\frac{2}{3}} = U_{i,I}^{n+\frac{2}{3}} \tag{3.65}$$

$$U_{i,F}^{n+1} = U_{i,I}^{n+1}. \tag{3.66}$$

De plus, il est nécessaire de fixer arbitrairement l'incrément de pression au niveau de la frontière afin d'avoir un système non singulier. Nous choisissons d'imposer au niveau de la sortie de la cellule d'électrophorèse :

$$P_F^c = P_F^{cc} = 0. \tag{3.67}$$

Cela équivaut à fixer arbitrairement la pression sur une face de la cellule.

Notons que jusqu'ici, seules les vitesses normales aux frontières ont été prises en compte pour déterminer les conditions sur la pression. Les conditions aux limites portant sur les vitesses tangentes aux frontières n'ont pas d'impact sur les équations de correction de pression. Nous pouvons en effet le constater dans la figure 3.13 : les vitesses tangentes à la frontière du domaine n'interviennent pas dans le bilan de quantité de mouvement du volume de contrôle.

3.3.5 Construction des systèmes linéaires

Nous adopterons la convention suivante pour décrire une ligne du système algébrique $A.X = B$, conformément aux notations employées dans la figure 3.10 :

$$a_M X_M + a_B X_B + a_S X_S + a_W X_W + a_E X_E + a_N X_N + a_T X_T = B_M.$$

Les indices M, B, S, W, *etc...*dépendent de la position du point considéré par rapport au volume de contrôle. Ainsi, a_M représente le coefficient diagonal d'une ligne de A et $(a_J)_{J\neq M}$ sont les coefficients hors-diagonaux. La somme de ces derniers sera notée $\sum_{J\neq M} a_J$. Dans le cas des pas prédicteurs, où les matrices et les vecteurs sont indexés ($A_i.U_i = B_i$), nous emploierons la notation suivante :

$$a_{i,M} U_{i,M} + a_{i,B} U_{i,B} + a_{i,S} U_{i,S} + a_{i,W} U_{i,W} + a_{i,E} U_{i,E} + a_{i,N} U_{i,N} + a_{i,T} U_{i,T} = B_{i,M}.$$

Pas prédicteur

Les matrices $(A_i)_{i=1,2,3}$ et les second membres $(B_i)_{i=1,2,3}$ associés au pas prédicteur de l'algorithme PISO sont obtenus en discrétisant les équations de convection-diffusion suivantes par la méthode des volumes finis :

$$\frac{1}{\delta t} u_i - \frac{1}{\mathrm{Re}} \Delta u_i + \sum_{j=1}^{3} u_j^n \frac{\partial u_i}{\partial x_j} = \frac{1}{\delta t} u_i^n - \frac{\partial p^n}{\partial x_i} + \mathcal{F}_i,\ i = 1, 2, 3$$

où $(u_j^n)_{j=1,2,3}$ et p^n représentent respectivement le champ de vitesse et la pression à l'instant $\frac{n}{\delta t}$ et $\mathcal{F}_i$ la contribution de $\epsilon^*\text{div}(E_i.\vec{E})$.

Chacune des équations associée à u_i est discrétisée sur le $i^{\text{ème}}$ maillage décalé (*cf.* figures 3.6 et 3.7) à l'aide des schémas de discrétisation donnés par [77]. Les lignes des matrices $(A_i)_{i=1,2,3}$ obtenues s'écrivent :

$$\left\{\begin{array}{rcl} a_{i,B} &=& -\delta x_1^i \delta x_2^i (\mathcal{A}(|\frac{f_{i,b}}{\nu_{i,b}}|)\nu_{i,b} + \max(f_{i,b},0)) \\ a_{i,S} &=& -\delta x_1^i \delta x_3^i (\mathcal{A}(|\frac{f_{i,s}}{\nu_{i,s}}|)\nu_{i,s} + \max(f_{i,s},0)) \\ a_{i,W} &=& -\delta x_2^i \delta x_3^i (\mathcal{A}(|\frac{f_{i,w}}{\nu_{i,w}}|)\nu_{i,w} + \max(f_{i,w},0)) \\ a_{i,E} &=& -\delta x_2^i \delta x_3^i (\mathcal{A}(|\frac{f_{i,e}}{\nu_{i,e}}|)\nu_{i,e} + \max(-f_{i,e},0)) \\ a_{i,N} &=& -\delta x_1^i \delta x_3^i (\mathcal{A}(|\frac{f_{i,n}}{\nu_{i,n}}|)\nu_{i,n} + \max(-f_{i,n},0)) \\ a_{i,T} &=& -\delta x_1^i \delta x_2^i (\mathcal{A}(|\frac{f_{i,t}}{\nu_{i,t}}|)\nu_{i,t} + \max(-f_{i,t},0)) \\ a_{i,M} &=& \frac{\delta x_1^i \delta x_2^i \delta x_3^i}{\delta t} + \sum_{J\neq M} -a_{i,J} \end{array}\right. \tag{3.68}$$

avec

$$\left\{\begin{array}{rcl@{\quad}rcl} f_{i,b} &=& U_{3,b}^n & \nu_{i,b} &=& \frac{1}{\text{Re}\times hx_{3-}^i} \\ f_{i,s} &=& U_{2,s}^n & \nu_{i,s} &=& \frac{1}{\text{Re}\times hx_{2-}^i} \\ f_{i,w} &=& U_{1,w}^n & \nu_{i,w} &=& \frac{1}{\text{Re}\times hx_{1-}^i} \\ f_{i,e} &=& U_{1,e}^n & \nu_{i,e} &=& \frac{1}{\text{Re}\times hx_{1+}^i} \\ f_{i,n} &=& U_{2,n}^n & \nu_{i,n} &=& \frac{1}{\text{Re}\times hx_{2+}^i} \\ f_{i,t} &=& U_{3,t}^n & \nu_{i,t} &=& \frac{1}{\text{Re}\times hx_{3+}^i} \end{array}\right. . \tag{3.69}$$

La fonction $\mathcal{A}$, donnée par le tableau 3.2, permet de se ramener à divers schémas de discrétisation des dérivées partielles.

Schéma	Expression de $\mathcal{A}(\lvert\mathcal{P}\rvert)$
différences centrées	$1 - 0.5\lvert\mathcal{P}\rvert$
amont	1
hybride	$\max(0, 1 - 0.5\lvert\mathcal{P}\rvert)$
power law	$\max(0, (1 - 0.5\lvert\mathcal{P}\rvert)^5)$
exponentiel	$\lvert\mathcal{P}\rvert/(\exp(\lvert\mathcal{P}\rvert) - 1)$

TAB. 3.2 – $\mathcal{A}(|\mathcal{P}|)$ en fonction des schémas de discrétisation

Les seconds membres des systèmes linéaires du pas prédicteur s'écrivent :

$$B_{1,M} = \delta x_1^1 \delta x_2^1 \delta x_3^1 (\frac{1}{\delta t} U_{i,M}^n + \mathcal{F}_{1,M}) - \delta x_2^1 \delta x_3^1 (P_e^n - P_w^n) \tag{3.70}$$

$$B_{2,M} = \delta x_1^2 \delta x_2^2 \delta x_3^2 (\frac{1}{\delta t} U_{i,M}^n + \mathcal{F}_{2,M}) - \delta x_1^2 \delta x_3^2 (P_n^n - P_s^n) \tag{3.71}$$

$$B_{3,M} = \delta x_1^3 \delta x_2^3 \delta x_3^3 (\frac{1}{\delta t} U_{i,M}^n + \mathcal{F}_{3,M}) - \delta x_1^3 \delta x_2^3 (P_t^n - P_b^n) \tag{3.72}$$

Pas correcteurs

Soit A la matrice associée à l'opérateur $-\sum_{i=1}^{3} \Delta_i^0 . D_i^{-1} . \Delta_i^i$, commune aux deux pas correcteurs et soient B^c et B^{cc} les seconds membres respectifs. Les coefficients des matrices diagonales D_i seront écrits de la même manière que les valeurs des champs de vitesse, puisqu'elles sont assimilables à des vecteurs. Par exemple, $D_{2,n}$ est la valeur du coefficient diagonal de D_2 correspondant à la face nord du volume de contrôle.

Les systèmes linéaires sont obtenus par l'intermédiaire de la discrétisation par volumes finis de $\mathrm{div}(\vec{U}) = 0$ sur le maillage principal (*cf.* figures 3.8 et 3.9), en substituant les expressions (3.54) puis (3.56) à $(U_i)_{i=1,2,3}$. La façon de procéder est similaire à l'approche employée par PATANKAR [77] dans les méthodes SIMPLE et SIMPLER. Les lignes de la matrice A s'écrivent :

$$\left\{ \begin{array}{rcl} a_B &=& -\delta x_1^0 \delta x_2^0 \frac{1}{D_{3,b}} \delta x_1^3 \delta x_2^3 \\ a_S &=& -\delta x_1^0 \delta x_3^0 \frac{1}{D_{2,s}} \delta x_1^2 \delta x_3^2 \\ a_W &=& -\delta x_2^0 \delta x_3^0 \frac{1}{D_{1,w}} \delta x_2^1 \delta x_3^1 \\ a_E &=& -\delta x_2^0 \delta x_3^0 \frac{1}{D_{1,e}} \delta x_2^1 \delta x_3^1 \\ a_N &=& -\delta x_1^0 \delta x_3^0 \frac{1}{D_{2,n}} \delta x_1^2 \delta x_3^2 \\ a_T &=& -\delta x_1^0 \delta x_2^0 \frac{1}{D_{3,t}} \delta x_1^3 \delta x_2^3 \\ a_M &=& \sum_{J \neq M} -a_J \end{array} \right. . \tag{3.73}$$

Les seconds membres respectifs du premier et second pas correcteur s'écrivent :

$$\begin{aligned} B_M^c &= x_2^0 \delta x_3^0 (U_w^{n+\frac{1}{3}} - U_e^{n+\frac{1}{3}}) + \delta x_1^0 \delta x_3^0 (U_s^{n+\frac{1}{3}} - U_e^{n+\frac{1}{3}}) \\ &\quad + \delta x_1^0 \delta x_2^0 (U_b^{n+\frac{1}{3}} - U_t^{n+\frac{1}{3}}) \qquad (3.74) \\ B_M^{cc} &= x_2^0 \delta x_3^0 (\hat{U}_w - \hat{U}_e) + \delta x_1^0 \delta x_3^0 (\hat{U}_s - \hat{U}_e) + \delta x_1^0 \delta x_2^0 (\hat{U}_b - \hat{U}_t) \qquad (3.75) \end{aligned}$$

avec

$$\hat{U}_i = D_i^{-1} . H_i . (U_i^{n+\frac{2}{3}} - U_i^{n+\frac{1}{3}}). \tag{3.76}$$

3.4 Discrétisation de l'équation de transport

L'équation de transport adimensionnelle (3.37) est discrétisée sur le maillage principal (*cf.* figures 3.8 et 3.9). Étant donné que la vitesse de convection $\vec{W}$ vaut $\vec{U} + m^* \vec{E}$, celle-ci ne vérifie pas a priori l'équation de conservation de la masse ($\mathrm{div}(\vec{W}) \neq 0$). Par conséquent, la mise en œuvre de la méthode des volumes finis devient plus complexe (il faut prendre en compte le terme $m^* \mathrm{div}(\vec{E})$). Nous avons donc choisi la méthode des différences finies pour discrétiser l'équation de transport

$$\frac{\partial c}{\partial t} - \frac{1}{\mathrm{Pe}} \Delta c + \sum_{i=1}^{3} (u_i + m^* E_i) \frac{\partial c}{\partial x_i} = 0.$$

Les dérivées partielles premières sont discrétisées selon un schéma décentré en amont. Le schéma d'évolution en temps a été traité par deux méthodes différentes :

1. le schéma implicite classique ;
2. le schéma explicite MPDATA (Fully Multidimensional Positive Definite Advection Transport Algorithm) de P. K. Smolarkiewicz [84, 85, 86] permettant de réduire la diffusion numérique ; cette méthode a été employée dans [22].

3.4.1 Schéma implicite

Nous adopterons les conventions de notation de la section 3.3.5 pour l'écriture des matrices et des vecteurs. Soient A et B la matrice de discrétisation et le second membre. Nous noterons C^n le champ de concentration discret au $n^{\text{ème}}$ pas de temps. Les lignes de la matrice A s'écrivent :

$$\left\{\begin{array}{rcl} a_B &=& -\left(\frac{2}{\mathrm{Pe}\times hx^0_{3-}(hx^0_{3-}+hx^0_{3+})}+\frac{\max(w_3,0)}{hx^0_{3-}}\right)\\ a_S &=& -\left(\frac{2}{\mathrm{Pe}\times hx^0_{2-}(hx^0_{2-}+hx^0_{2+})}+\frac{\max(w_2,0)}{hx^0_{2-}}\right)\\ a_W &=& -\left(\frac{2}{\mathrm{Pe}\times hx^0_{1-}(hx^0_{1-}+hx^0_{1+})}+\frac{\max(w_1,0)}{hx^0_{1-}}\right)\\ a_E &=& -\left(\frac{2}{\mathrm{Pe}\times hx^0_{1+}(hx^0_{1-}+hx^0_{1+})}+\frac{\max(-w_1,0)}{hx^0_{1+}}\right)\\ a_N &=& -\left(\frac{2}{\mathrm{Pe}\times hx^0_{2+}(hx^0_{2-}+hx^0_{2+})}+\frac{\max(-w_2,0)}{hx^0_{2+}}\right)\\ a_T &=& -\left(\frac{2}{\mathrm{Pe}\times hx^0_{3+}(hx^0_{3-}+hx^0_{3+})}+\frac{\max(-w_3,0)}{hx^0_{3+}}\right)\\ a_M &=& \frac{1}{\delta t}+\sum_{J\neq M}-a_J \end{array}\right. \tag{3.77}$$

avec

$$w_i = U_{i,M} + m^* E_{i,M},\ i = 1, 2, 3. \tag{3.78}$$

L'expression du second membre est :

$$B_M = \frac{1}{\delta t} C^n_M. \tag{3.79}$$

Ici, le traitement des conditions aux limites ne comporte pas de difficulté. En adoptant les notations de la section 3.3.4, nous obtenons :

– pour les conditions de Dirichlet $c = c^0$,

$$C^{n+1}_F = c^0 \tag{3.80}$$

– pour les conditions de Neumann $\frac{\partial c}{\partial n} = 0$,

$$C^{n+1}_F - C^{n+1}_I = 0 \tag{3.81}$$

Notons que les conditions de Neumann sont prépondérantes (*cf.* page 93). Le conditionnement de la matrice de discrétisation est donc mauvais.

3.4.2 Algorithme MPDATA

La mise en œuvre de l'algorithme MPDATA [84, 85, 86] a été motivé par la présence d'une forte diffusion numérique. De plus, cet algorithme possède une propriété importante : elle garantit la positivité du champ de concentration. Voici un bref résumé de l'algorithme de base exposé dans [84, 85].

Soit $\vec{W}$ le vecteur vitesse de composantes $(w_i)_{i=1,2,3}$ défini par l'équation (3.78). L'équation de transport (3.37) se ré-écrit sous la forme d'une équation d'advection :

$$\frac{\partial c}{\partial t} + \sum_{i=1}^{3} w_i \frac{\partial c}{\partial x_i} = 0. \tag{3.82}$$

Le terme $-\frac{1}{\mathrm{Pe}}\Delta c$ lié à la diffusion est négligé ($\mathrm{Pe} \geq 10^5$). L'algorithme MPDATA est une variante d'un schéma explicite décentré [27], appelé aussi "*donor-cell*". Il consiste à corriger la diffusion numérique engendrée par le schéma explicite décentré. Le principe de l'algorithme MPDATA est le suivant :

1. une vitesse dite "*antidiffusive*" est calculée à partir d'une estimation de la diffusion numérique présente dans le champ de concentration,
2. l'équation d'advection (3.82) est discrétisée avec la vitesse antidiffusive et le schéma explicite décentré est appliqué à nouveau.

Cette correction est ensuite appliqué de manière itérative. En pratique, elles n'ont plus d'effet significatif au delà de la $3^{\text{ème}}$ application [84]. Cet algorithme n'est pas inconditionnellement stable. Une étude sur les critères de stabilité est effectuée dans [84]. Il y est montré que la condition de Courant, Friedrichs et Lewy (CFL) [27]

$$\forall M, \sum_{i=1}^{3} |W_{i,M}| \frac{\delta t}{\delta x_i^0} < 1$$

doit être satisfaite à chaque itération.

Dans la suite, les vecteurs $(C^{(*)^k})_{k\geq 1}$ désigneront les concentrations calculées à chaque itération de l'algorithme MPDATA. Par convention, on posera

$$C^{(*)^0} = C^n \tag{3.83}$$

C^n étant la concentration au $n-$ème pas de temps. Les vecteur $(\vec{W}^{(*)^k})_{k\geq 1}$ désigneront les vecteurs vitesses intervenant à chaque itération. Le vecteur vitesse associé à la première itération vaut donc :

$$\vec{W}^{(*)^1} = \vec{U} + m^* \vec{E}. \tag{3.84}$$

On notera $(A_k)_{k\geq 1}$ les matrices de discrétisation associées à chaque itération. Chaque matrice A_k est calculée en fonction de $\vec{W}^{(*)^k}$ à l'aide du schéma explicite décentré.

La première itération de MPDATA consiste à calculer la concentration en appliquant le schéma explicite décentré à l'équation d'advection (3.82) :

$$C^{(*)^1} = C^{(*)^0} - A_1.C^{(*)^0}. \tag{3.85}$$

La concentration C^{n+1} est calculée en appliquant éventuellement plusieurs corrections successives à $C^{(*)^1}$.

Les vitesses antidiffusives $(\vec{W}^{(*)^k})_{k\geq 2}$ sont calculées récursivement de la manière suivante :

$$\vec{W}^{(*)^k} = \tilde{u}(\vec{W}^{(*)^{k-1}}, C^{(*)^{k-1}}),\ k \geq 2. \tag{3.86}$$

Pour simplifier l'exposé, nous renvoyons à [84, 85] pour l'expression de la fonction $\tilde{u}$ calculant la vitesse antidiffusive à chaque itération. À titre indicatif, ce calcul est basé sur le développement de Taylor du $2^{\text{ème}}$ ordre de l'équation (3.82).

Les vitesses antidiffusives exprimées par la relation (3.86) sont ensuite utilisées pour inverser l'effet de la diffusion numérique :

$$C^{(*)^k} = C^{(*)^{k-1}} - A_k.C^{(*)^{k-1}},\ k \geq 2. \tag{3.87}$$

La matrice A_k $(k \geq 2)$ est obtenue en discrétisant l'équation d'advection

$$\frac{\partial c}{\partial t} + \sum_{i=1}^{3} w_i^{(*)^k} \frac{\partial c}{\partial x_i} = 0,\ k \geq 2. \tag{3.88}$$

à l'aide du schéma explicite décentré. Les conditions aux limites sur la concentration ne changent pas. Un pas correcteur consiste donc à faire évoluer la concentration $C^{(*)^{k-1}}$ d'un pas de temps en utilisant la vitesse antidiffusive.

En notant k_0 le nombre d'itérations souhaitées $(k_0 \geq 1)$, la concentration

$$C^{n+1} = C^{(*)^{k_0}}$$

est la concentration au $(n+1)^{\text{ème}}$ pas de temps. On dit que C^{n+1} est calculé via l'algorithme MPDATA d'ordre k_0. Notons que l'algorithme MPDATA d'ordre 1 correspond au schéma explicite décentré.

Enfin, en ce qui concerne la discrétisation des équations d'advection (3.82) et (3.88) par le schéma explicite décentré dans des maillages décalés, nous renvoyons à [27].

3.5 Discrétisation de l'équation de potentiel

Dans cette section, les conventions de notation de la section 3.3.5 sont conservées. Pour résoudre numériquement l'équation adimensionnelle de potentiel (3.40), nous avons discrétisé par différences finies sur le maillage principal (*cf.* figures 3.8 et 3.9). Le schéma de discrétisation est obtenu via deux schémas intermédiaires. Le schéma associé à $-\frac{\partial}{\partial x_1}(K\frac{\partial \Phi}{\partial x_1})$ par exemple, est obtenu en trois étapes.

1. Discrétisation par le schéma "*avant-arrière*" :

$$-\frac{\partial}{\partial x_1}\left(K\frac{\partial \Phi}{\partial x_1}\right) = \frac{1}{hx^0_{1+}}\left(K_E\frac{\Phi_E-\Phi_M}{hx^0_{1+}} - K_M\frac{\Phi_M-\Phi_W}{hx^0_{1-}}\right)$$

2. Discrétisation par le schéma "*arrière-avant*" :

$$-\frac{\partial}{\partial x_1}\left(K\frac{\partial \Phi}{\partial x_1}\right) = \frac{1}{hx^0_{1-}}\left(K_M\frac{\Phi_E-\Phi_M}{hx^0_{1+}} - K_W\frac{\Phi_M-\Phi_W}{hx^0_{1-}}\right)$$

3. Le schéma de discrétisation final est la moyenne des schémas précédents.

La discrétisation de l'équation

$$-\sum_{i=1}^{3}\frac{\partial}{\partial x_i}\left(K\frac{\partial \Phi}{\partial x_i}\right) = 0$$

par la méthode ci-dessus conduit à une matrice A et un second membre B dont les coefficients sont donnés dans les équations suivantes :

$$\left\{\begin{array}{rcl} a_B &=& -\frac{1}{2}(K_B\frac{1}{(hx^0_{3-})^2} + K_M\frac{1}{hx^0_{3-}\,hx^0_{3+}}) \\ a_S &=& -\frac{1}{2}(K_S\frac{1}{(hx^0_{2-})^2} + K_M\frac{1}{hx^0_{2-}\,hx^0_{2+}}) \\ a_W &=& -\frac{1}{2}(K_W\frac{1}{(hx^0_{1-})^2} + K_M\frac{1}{hx^0_{1-}\,hx^0_{1+}}) \\ a_E &=& -\frac{1}{2}(K_E\frac{1}{(hx^0_{1+})^2} + K_M\frac{1}{hx^0_{1-}\,hx^0_{1+}}) \\ a_N &=& -\frac{1}{2}(K_N\frac{1}{(hx^0_{2+})^2} + K_M\frac{1}{hx^0_{2-}\,hx^0_{2+}}) \\ a_T &=& -\frac{1}{2}(K_T\frac{1}{(hx^0_{3+})^2} + K_M\frac{1}{hx^0_{3-}\,hx^0_{3+}}) \\ a_M &=& \sum_{J\neq M} -a_J \end{array}\right. \tag{3.89}$$

$$B_M = 0. \tag{3.90}$$

Le calcul du potentiel à l'entrée et à la sortie de la cellule (conditions aux limites (3.16) et (3.21)) est basé sur la résolution de problèmes bidimensionnels. Les matrices de discrétisation associées sont obtenues par la méthode de discrétisation utilisée pour calculer la matrice (3.89) ; les coefficients a_S et a_N n'interviennent pas.

Enfin, la discrétisation des conditions aux limites $\Phi = \phi$, de type Dirichlet, ne posent aucun problème. Nous obtenons :

$$\Phi_F^{n+1} = \phi. \tag{3.91}$$

3.6 Mise en œuvre et expérimentation

L'algorithme de simulation résulte du couplage des algorithmes traités dans les sections 3.3, 3.4 et 3.5. L'algorithme de la figure 3.14 résume les étapes essentielles

Initialiser $\vec{U}^0, P^0, C^0, \Phi^0$
pour $n = 0, 1, \ldots, n_t$
$\vec{E} = -\vec{\text{grad}}(\Phi^n)$
pour $i = 1, 2, 3 : \mathcal{F}_i = \epsilon^* \text{div}(E_i.\vec{E})$ **fin**
`Pas prédicteur de PISO` :
pour $i = 1, 2, 3$:
calculer la matrice A_i et le second membre B_i
Résoudre : $A_i.U_i^{n+\frac{1}{3}} = B_i$
fin
`Pas correcteurs de PISO` :
calculer la matrice A
calculer le second membre B
Résoudre : $A.P^c = B$
calculer $P^{n+\frac{1}{2}}$ et $\vec{U}^{n+\frac{2}{3}}$ d'après (3.54)
calculer le second membre B
Résoudre : $A.P^{cc} = B$
calculer P^{n+1} et $\vec{U}^{n+1}$ d'après (3.56)

$\vec{W} = \vec{U}^{n+1} + m^* \vec{E}$
si schéma implicite **alors**
calculer la matrice A et le second membre B
Résoudre : $A.C^{n+1} = B$
sinon
Algorithme MPDATA à l'ordre 2 :
$C^{n+1} = C^{(*)^2}$
fin

$K = K_0^* + C^{n+1}$
calculer la matrice A et le second membre B
Résoudre : $A.\Phi^{n+1} = B$
fin

FIG. 3.14 – Algorithme de simulation du procédé d'électrophorèse

de la simulation. L'algorithme de Schwarz asynchrone avec communication flexible a été mis en œuvre pour résoudre en parallèle les 7 systèmes linéaires mentionnés dans la figure 3.14. Le solveur parallèle décrit au chapitre 2 a été utilisé. Quant à la parallélisation des autres parties du code, elle n'a pas été envisagée pour le moment car il semble qu'une telle parallélisation influe peu sur les performances.

3.6.1 Résolution numérique des systèmes linéaires

Nous établissons à présent la convergence de l'algorithme de Schwarz synchrone et asynchrone pour la résolution numérique des systèmes linéaires intervenant dans chaque pas de simulation du procédé d'électrophorèse de zone à écoulement continu. Pour cela, nous nous appuyons principalement sur les résultats énoncés dans la section 1.5 du chapitre 1. Notons que la mise en œuvre des méthodes asynchrones dans l'algorithme PISO a été envisagée par H.C. Boisson et P. Spitéri en 1993 [91].

Propriétés des matrices

Vérifions que les matrices de discrétisation issues des équations d'écoulement, de transport et de potentiel construites dans les sections 3.3 à 3.5 sont des M-matrices.

Proposition 3.1
Les matrices de discrétisation du pas prédicteur de l'algorithme PISO (3.68) sont des M-matrices pour tous les schémas de discrétisation du tableau 3.2 excepté pour le schéma par différences centrées. Pour ce cas particulier, les matrices du pas prédicteur sont des M-matrices si $|\mathcal{P}| \leq 2$.

Démonstration Pour des schémas de discrétisation autres que les différences centrées, les matrices $(A_i)_{i=1,2,3}$ du pas prédicteur, données par l'équation (3.68) possèdent des coefficients diagonaux strictement positifs et des coefficients hors-diagonaux négatifs ou nuls. De plus, elles sont diagonales dominantes irréductibles. Ce sont donc des M-matrices. En ce qui concerne le schéma par différences centrées, les arguments ci-dessus sont valables lorsque $|\mathcal{P}| \leq 2$. □

Proposition 3.2
La matrice de discrétisation des pas correcteurs de l'algorithme PISO (3.73) est une M-matrice.

Démonstration La matrice A des pas correcteurs, donnée par l'équation (3.73) possède des coefficients diagonaux strictement positifs et des coefficients hors-diagonaux négatifs. Elle est de plus diagonale dominante irréductible. Par ailleurs, cette matrice est inversible grâce à la condition de type Dirichlet sur les incréments de pression discrétisée sous la forme (3.67). C'est donc une M-matrice. □

Proposition 3.3
La matrice de discrétisation de l'équation de transport (3.77) associée au schéma implicite est une M-matrice.

Le principe de la démonstration est le même que celui de la proposition 3.1.

Proposition 3.4
La matrice de discrétisation de l'équation de potentiel (3.89) est une M-matrice.

Le principe de la démonstration est le même que celui de la proposition 3.2.

Convergence des itérations asynchrones

Étudions à présent la convergence de l'algorithme de Schwarz asynchrone avec communication flexible dans le cadre de la résolution des systèmes linéaires issus de la discrétisation des équations du procédé d'électrophorèse de zone à écoulement continu. L'analyse est effectuée dans le contexte où le schéma itératif et les communications sont modélisés par une méthode de contraction (*cf.* section 1.4).

Nous exploiterons la notion de système augmenté, qui permet d'établir une connexion entre la méthode de Schwarz et les algorithmes de relaxation. La résolution par blocs du système augmenté modélise la résolution par sous-domaine du problème initial (*cf.* section 1.5).

Proposition 3.5
*L'algorithme de Schwarz asynchrone avec communication flexible, appliqué à la résolution des problèmes discrets d'écoulement, de transport via le schéma implicite et de potentiel (*cf. *les matrices (3.68), (3.73), (3.77) et (3.89)), converge vers les solutions respectives des systèmes linéaires.*

Démonstration Nous considérons pour cela la résolution du système augmenté par des itérations parallèles asynchrones avec communication flexible décrites par (1.62) et (1.63).

D'après les propositions 3.1, 3.2, 3.3 et 3.4, les matrices (3.68), (3.73), (3.77) et (3.89) sont des M-matrices,

Lorsque la matrice de discrétisation est une M-matrice, on peut déduire grâce à un résultat de D.J. Evans et W. Deren [39] que la matrice du système augmenté associé est aussi une M-matrice.

L'algorithme de relaxation employé au niveau des sous-domaines est l'algorithme de Gauss-Seidel par blocs utilisant la variante du pivot de Gauss pour les systèmes linéaires tri-diagonaux. En présence d'une M-matrice, les décompositions des sous-matrices diagonales sont faiblement régulières.

La convergence des itérations asynchrones avec communication flexible (1.62) est alors assurée car la proposition 1.6 s'applique. □

Remarque 3.1 Les algorithmes de Jacobi et de Gauss-Seidel conduisent aussi à des décompositions faiblement régulières pour les sous-matrices diagonales (*cf.* section 1.4.3).

Notons que la convergence de l'algorithme de Schwarz asynchrone avec communication flexible se démontre en utilisant une analyse basée sur une technique d'ordre partiel [51, 71]. De plus, l'itéré initial doit satisfaire $\hat{A}.\hat{X}^0 - \hat{B} \geq 0$ pour garantir la convergence (*cf.* théorème 1.3). Dans le cadre du problème d'électrophorèse, l'initialisation des solveurs n'est pas aisé dans le cas général car les matrices dépendent des composantes du vecteur vitesse qui elles-mêmes font partie des inconnues du problème. L'analyse par une méthode de contraction permet en l'occurrence de s'affranchir de ce problème.

3.6.2 Résultats des simulations

Des simulations séquentielles ont été effectuées dans le but de vérifier la stabilité des méthodes numériques et de vérifier l'adéquation des résultats avec la physique. Les paramètres utilisés sont indiqués dans le tableau 3.3.

Le maillage utilisé est régulier mais non uniforme car la vitesse axiale u_2 varie très peu en fonction de x_2 [22]. Il n'est donc pas nécessaire d'utiliser un nombre important de points de discrétisation pour l'axe e_2.

La stabilité de l'algorithme PISO a été validée expérimentalement pour des cas où le nombre de Reynolds est supérieur à 100 (*cf.* les travaux ainsi que la bibliographie de la thèse de doctorat de J.L. Estivalezes [38]). Lorsque le nombre de Reynolds est plus faible, l'algorithme devient instable. Cela est contraire à la définition de la stabilité des écoulements et des schémas convectifs. De plus, le calcul de la pression est extrêmement long. L'origine de ce problème semble purement numérique.

En choisissant la largeur de la cellule comme longueur de référence pour former les équations adimensionnelles, on obtient un nombre de Reynolds supérieur à 100. Si l'épaisseur de la cellule avait été utilisée comme longueur de référence, le nombre de Reynolds obtenu aurait été inférieur à 10 et la simulation n'aurait pas donné des résultats satisfaisants.

Simulation avec l'algorithme MPDATA

La figure 3.15 montre la déviation du jet et une coupe effectuée au milieu de la cellule, mettant en évidence le profil du filet. Notons que le filet est déformé à cause du temps de séjour non uniforme. La figure 3.16 met en évidence les perturbations du vecteur vitesse au voisinage du filet, dus à l'électrohydrodynamique.

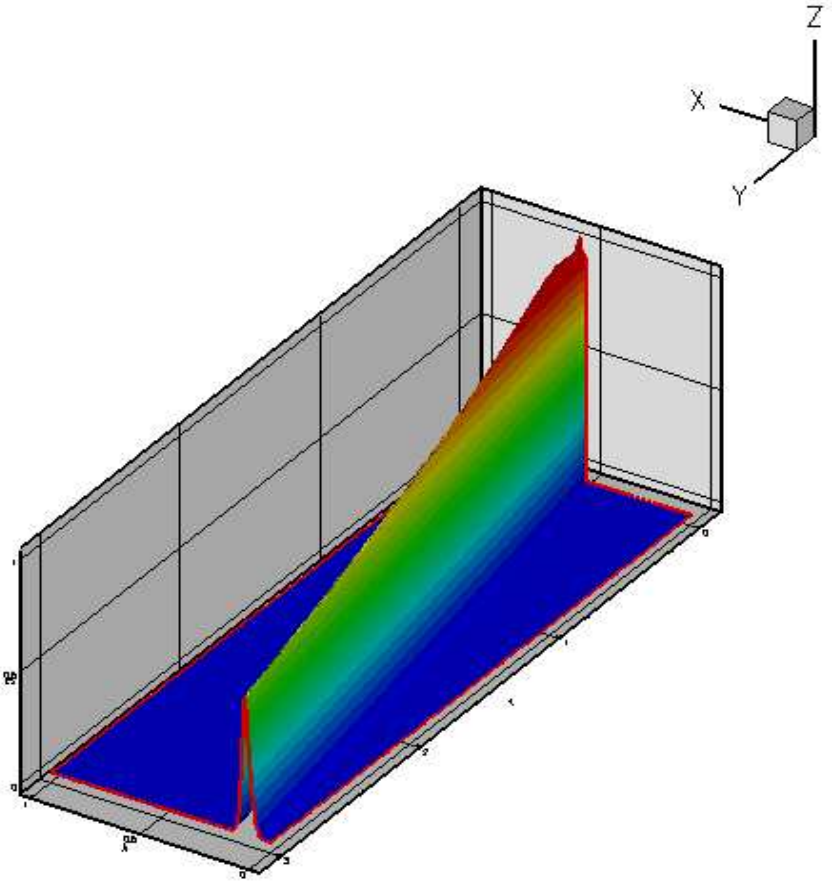
Les courbes de la figure 3.17 montrent que la condition CFL est respectée durant toute la simulation. Elles restent inférieures au seuil qui vaut 1.

Quant aux courbes de la figure 3.18, elles permettent de juger de la stabilité de l'algorithme PISO au cours de la simulation. Les indicateurs utilisés sont les normes

Dimensions de la cellule	
largeur adimensionnelle	1
longueur adimensionnelle	3
épaisseur adimensionnelle	0.05
Taille du maillage	
nombre de points sur l'axe e_1	400
nombre de points sur l'axe e_2	150
nombre de points sur l'axe e_3	20
Discrétisation en temps	
durée adimensionnelle	3
nombre de pas de temps	900
Équation d'écoulement	
Schéma de discrétisation du pas prédicteur	hybride
nombre de Reynolds	250
permittivité diélectrique	0.1
Équation de transport	
coordonnées du jet	$x_1 = 0.5\,;\, x_3 = 0.025$
rayon du jet	0.015
nombre de Peclet	10^5
mobilité électrophorétique	0.2
Équation de potentiel	
conductivité du milieu	10

TAB. 3.3 – Paramètres des simulations numériques du procédé d'électrophorèse de zone à écoulement continu

Courbe $c(x_1, x_2)$:



Coupe dans le plan (e_1, e_3) :

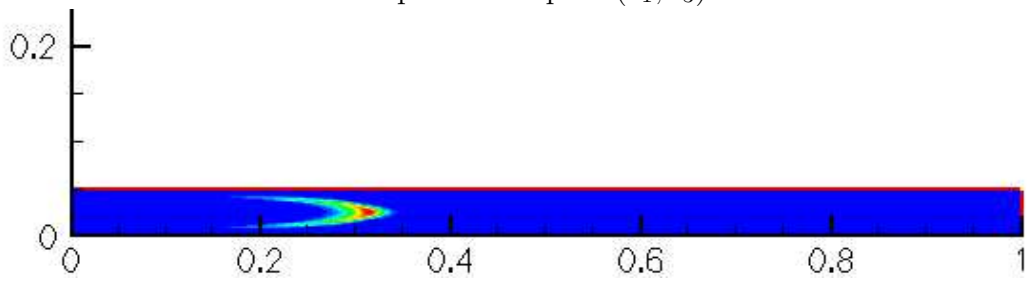


FIG. 3.15 – Champ de concentration calculé via MPDATA

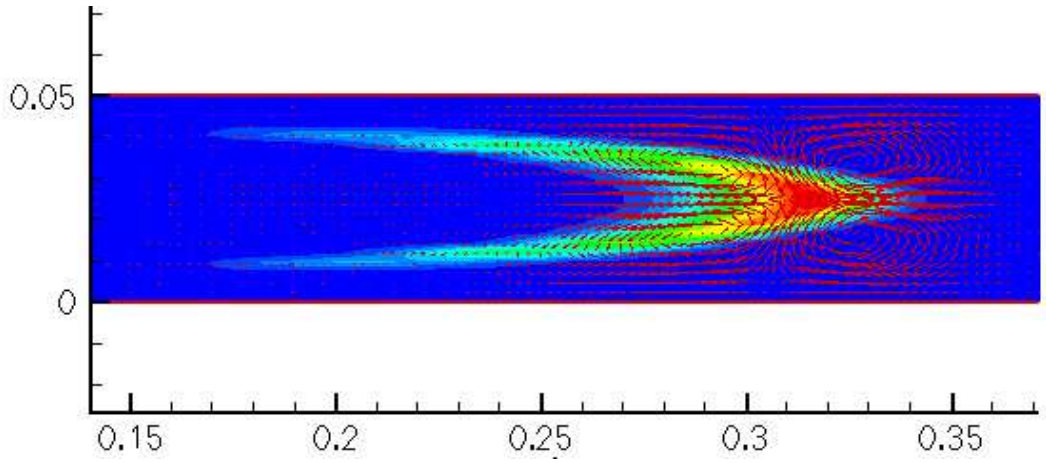


FIG. 3.16 – Champ de vitesse au voisinage du filet (MPDATA) - coupe (e_1, e_3)

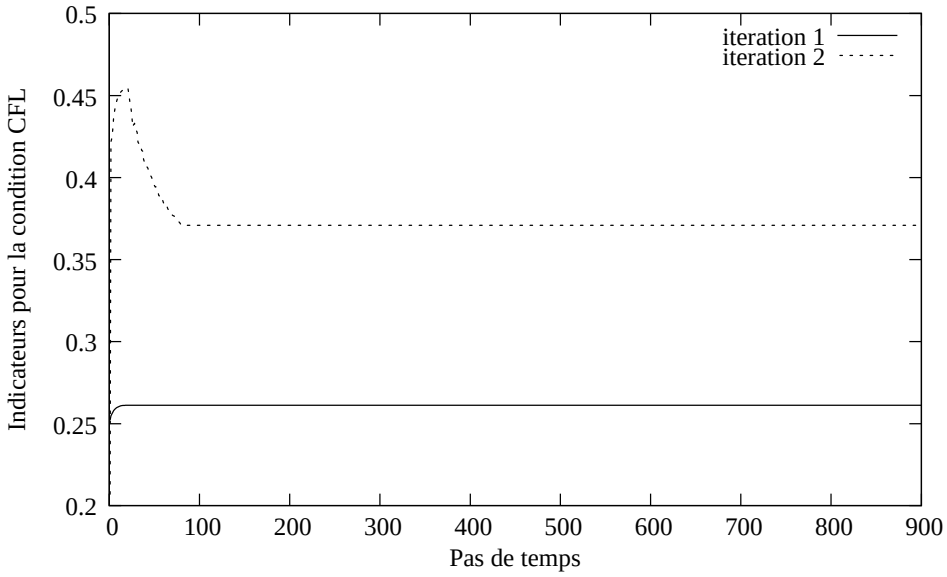


FIG. 3.17 – Indicateurs de stabilité pour l'algorithme MPDATA

de $\mathrm{div}(\vec{V}^{n+\frac{1}{3}})$, $\mathrm{div}(\vec{V}^{n+\frac{2}{3}})$ et $\mathrm{div}(\vec{V}^{n+1})$, calculés à l'issue du pas prédicteur et des pas correcteurs. Malgré un accroissement brutal des indicateurs au voisinage du $600^{\text{ème}}$ pas de temps, l'algorithme demeure stable. En effet, $\|\mathrm{div}(\vec{V})\|$ reste borné.

À titre indicatif, la simulation effectuée avec l'algorithme MPDATA d'ordre 2 dure environ 45 heures sur un PC de bureau (Pentium 4 à 2.8 GHz).

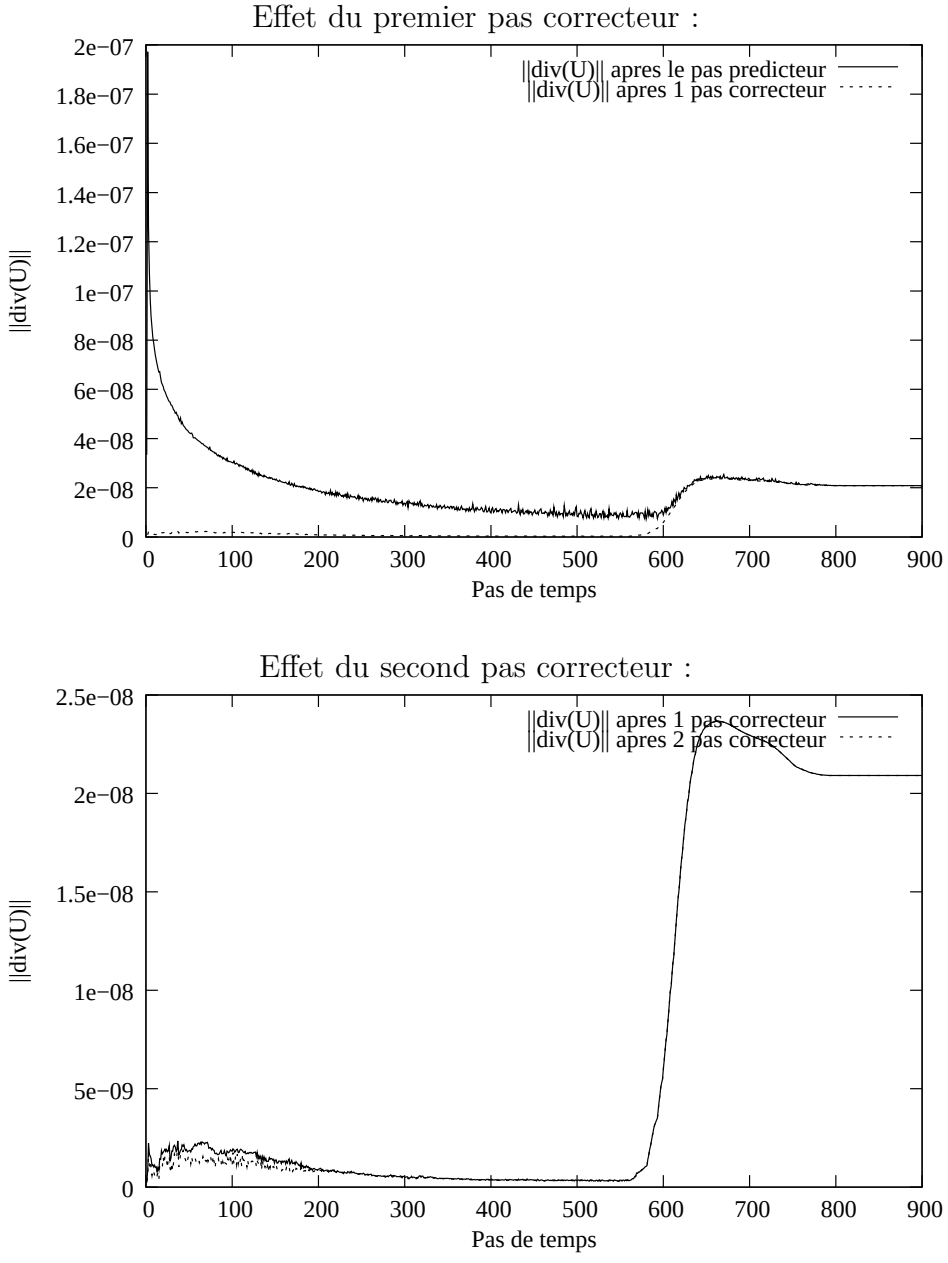


FIG. 3.18 – Effet des pas correcteur de PISO sur $\|\mathrm{div}(\vec{U})\|$ (MPDATA)

Simulation avec le schéma implicite

Bien que le schéma implicite présente l'avantage majeur d'être inconditionnellement stable par rapport à des schémas explicites tels que l'algorithme MPDATA, la présence de diffusion numérique fausse légèrement les résultats. Une comparaison entre les figures 3.15 et 3.19 met clairement en évidence ce phénomène numérique.

Notons que la diffusion numérique est un phénomène qui est propre aux domaines multidimensionnels. Elle survient lorsque les 2 conditions ci-dessus sont réunies :

1. la vitesse de transport est oblique par rapport aux axes ;
2. le gradient de la variable dans la direction normale à la vitesse de transport est non nul.

Un coefficient de diffusion très faible contribue à accentuer les effets de la fausse diffusion. Une illustration de ce phénomène numérique est donnée dans [77].

La coupe de la cellule dans la figure 3.19 met en évidence la déformation du jet due au temps de séjour non uniforme. La figure 3.20 montre les perturbation du vecteur vitesse dus à l'effet électrohydrodynamique.

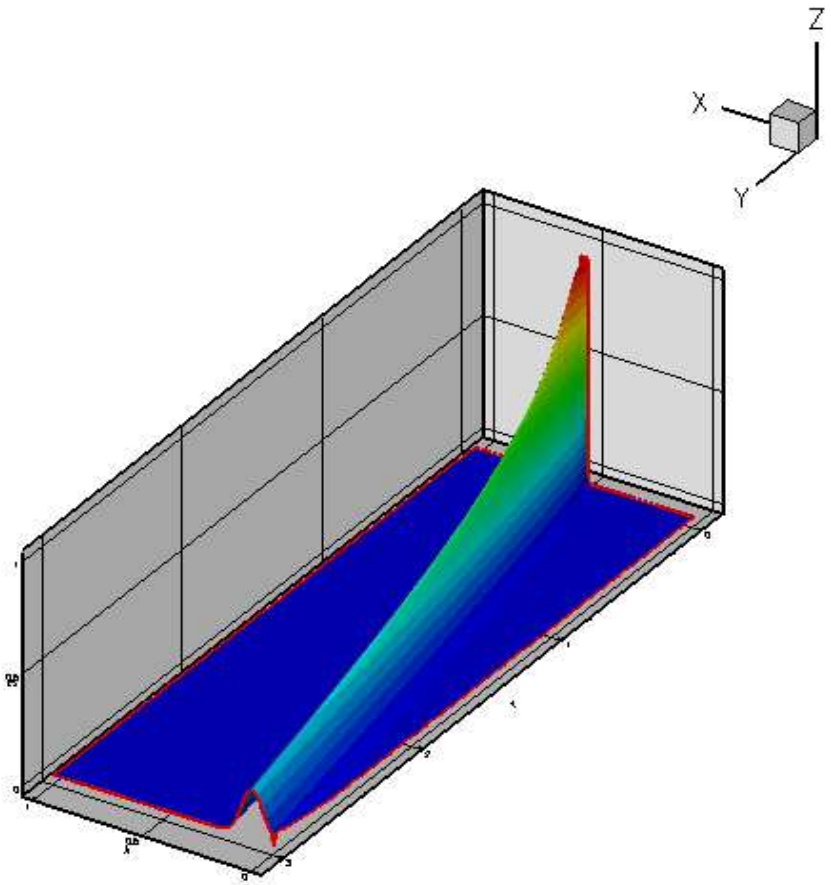
Sur la figure 3.21, une comparaison des concentrations calculées avec le schéma implicite (à gauche) et par l'algorithme MPDATA (à droite) est donnée. En raison de la diffusion numérique, la concentration à la sortie de la cellule est faible. Nous noterons toutefois que les 2 jets sont superposables.

Les courbes de la figure 3.22 montrent que l'écoulement est stable. Les courbes de la figure 3.23 mettent en avant l'influence de l'algorithme de résolution de l'équation de concentration sur un indicateur de stabilité de l'algorithme PISO. Nous pouvons constater un accroissement brutal de $\|\mathrm{div}(\vec{U})\|$ dans les 2 simulations. Cela montre que malgré la diffusion numérique, les simulations effectuées avec le schéma implicite sont pertinents.

À titre indicatif, la simulation effectuée avec le schéma implicite 2 dure environ 46 heures sur un PC de bureau (Pentium 4 à 2.8 GHz).

Remarque 3.2 Les conditions aux limites de type Neumann prédominent dans l'équation de concentration (*cf.* page 93) et dans les équations de pression de l'algorithme PISO (*cf.* page 108). Leur résolution numérique à l'aide de méthodes itératives est a priori très lente compte tenu du mauvais conditionnement qui est induit par la prédominance de ce type de condition aux limites. Toutefois, lors de la résolution numérique de l'équation de concentration avec le schéma implicite, la vitesse de convergence reste rapide car les matrices décrites par l'équation (3.77) bénéficient d'un bon conditionnement grâce au terme additionnel dû à la discrétisation de la dérivée par rapport au temps.

Courbe $c(x_1, x_2)$:



Coupe dans le plan (e_1, e_3) :

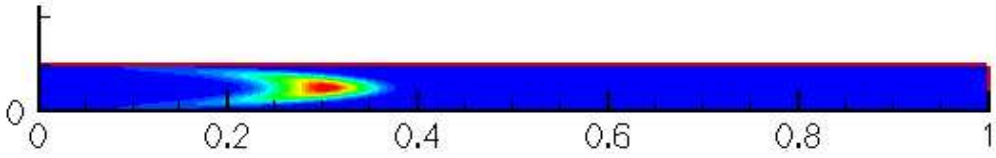


FIG. 3.19 – Champ de concentration calculé via le schéma implicite

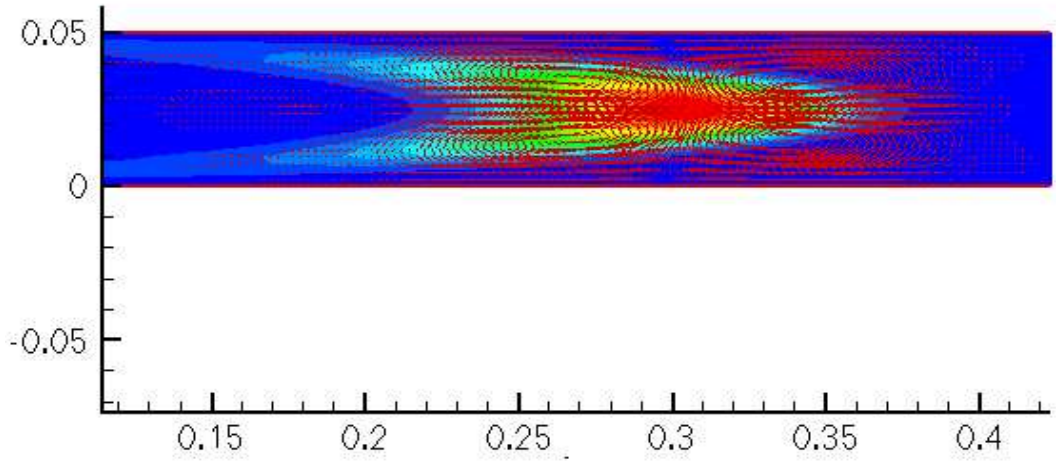


FIG. 3.20 – Champ de vitesse au voisinage du filet (schéma implicite) - coupe (e_1, e_3)

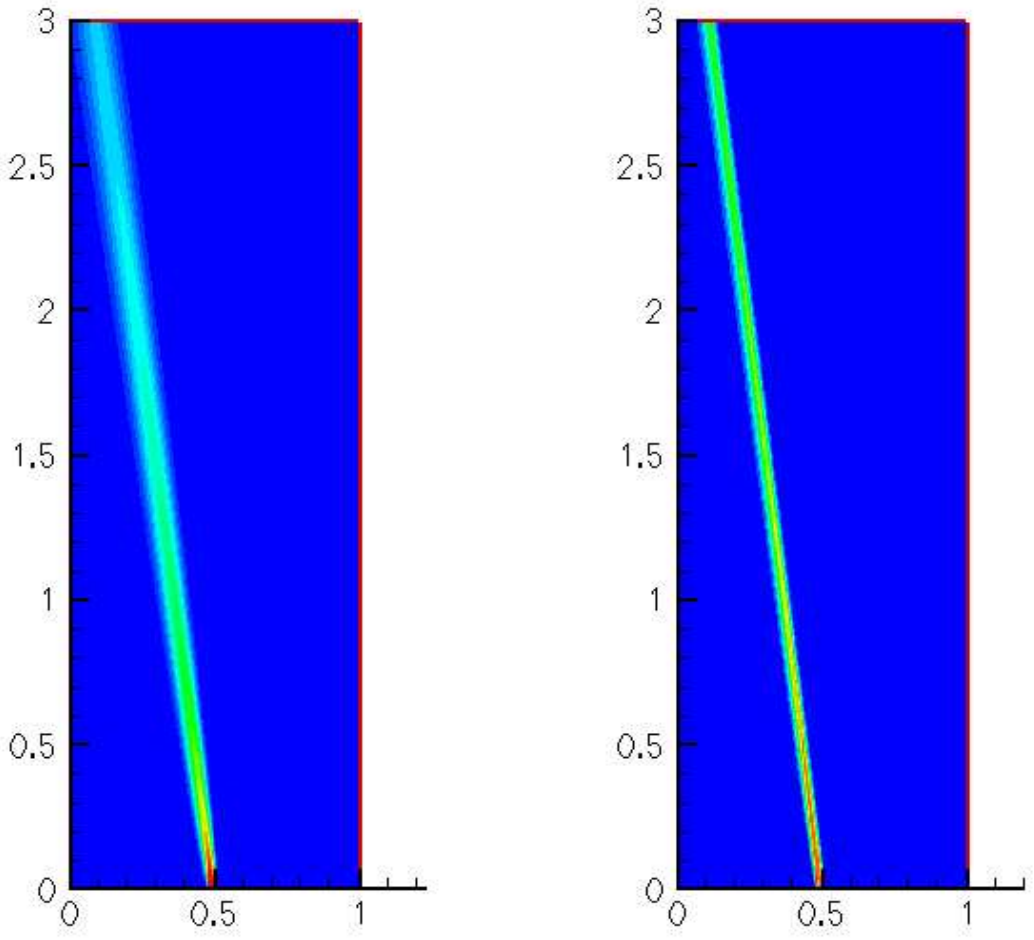


Schéma implicite Algorithme MPDATA

FIG. 3.21 – Comparaison des concentrations calculées par le schéma implicite et par MPDATA - vue 2D -

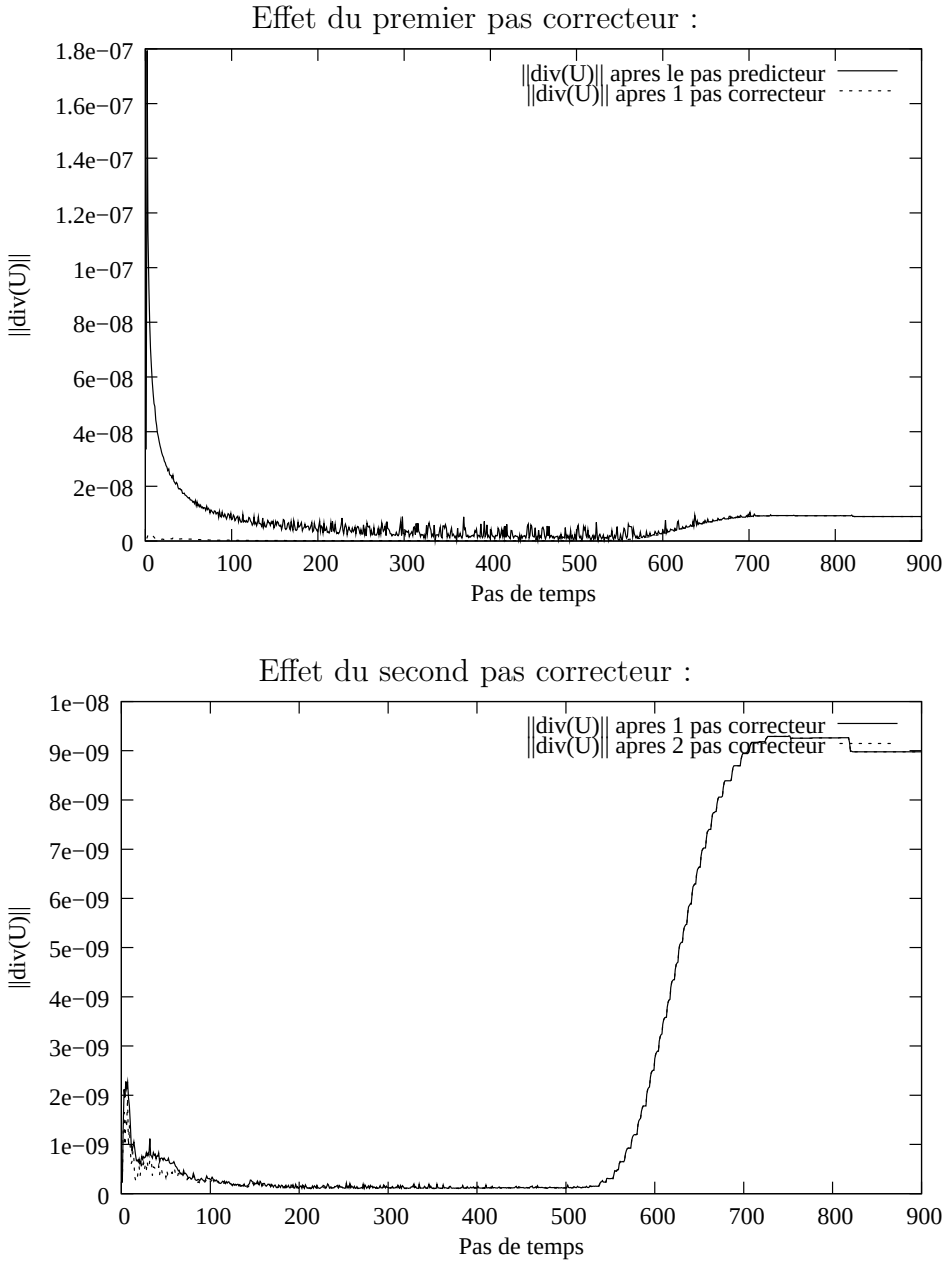


FIG. 3.22 – Effet des pas correcteur de PISO sur $\|\mathrm{div}(\vec{U})\|$ (schéma implicite)

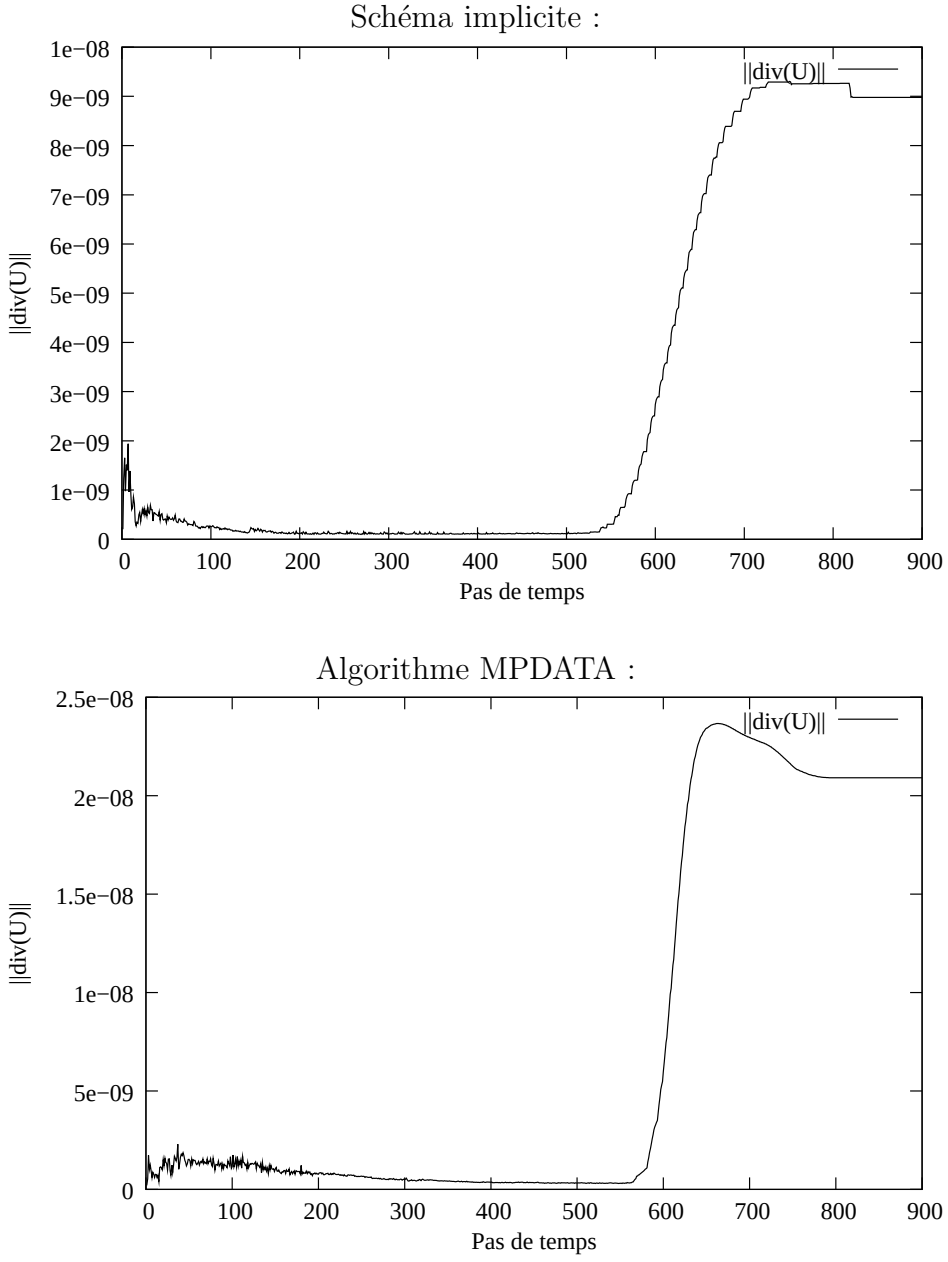


FIG. 3.23 – $\|\mathrm{div}(\vec{U})\|$ en fonction des pas de temps (schéma implicite et MPDATA)

3.6.3 Étude de performance sur IBM Power 4 p690+

L'architecture cible est décrite à la page 75 du chapitre 2. L'étude de performance des simulations sur calculateur parallèle a été réalisée avec une concentration initiale différente des simulations précédentes. Les études de performance ont été effectuées sur 20 pas de temps seulement compte tenu de la quantité de ressources de calcul disponibles.

En effet, nous avons constaté que lors des premiers pas de la simulation, un seul processeur était actif : celui qui prend en charge les sous-domaines situés à l'entrée de la cellule. De manière générale, très peu voire aucune itération n'est effectuée au niveau d'un sous-domaine si un régime stationnaire est atteint dans le sous-domaine. Par conséquent, pour profiter du parallélisme, le découpage en sous-domaines n'est pas trivial car aucun processeur ne doit prendre en charge des sous-domaines dans lesquels la concentration est nulle. Une solution envisageable consisterait à calculer un découpage au début de chaque pas de temps, en fonction des valeurs des grandeurs physiques. Cependant, la mise au point d'une méthode de découpage robuste qui évite le déséquilibrage de charge excessif durant toute la simulation est un problème difficile sur lequel nous ne nous sommes pas penché. Par conséquent, la cellule contient initialement un jet rectiligne, non déformé, traversant la cellule de part en part. Cette façon de procéder nous place dans des conditions d'équilibrage de charge raisonnables.

Comportement du solveur

Dans un premier temps, une étude de performance du solveur similaire à celle du chapitre 2 a été réalisée. Le solveur a été testé sur la résolution du problème de convection-diffusion suivant

$$0.1 \times \Delta u + 0.5 \times \frac{\partial u}{\partial x_1} + 1.5 \times \frac{\partial u}{\partial x_2} - 0.5 \times \frac{\partial u}{\partial x_3} + 300 \times u = f \tag{3.92}$$

avec 1, 2, 4, 8, et 16 processeurs. Le domaine est découpé en 128 sous-domaines. L'équation (3.92) est discrétisée sur le maillage principal du problème d'électrophorèse ($400 \times 150 \times 20$ points intérieurs). La comparaison des performances des itérations synchrones et asynchrones est donnée dans les figures suivants :

- la figure 3.24 donne l'accélération et l'efficacité ;
- la figure 3.25 donne le surcoût engendré par MPI ; c'est le temps passé dans les fonctions MPI, cumulé par tous les processeurs.

Ces courbes montrent que les itérations asynchrones sont plus efficaces que les itérations synchrones sauf lorsque 2 processeurs sont utilisés. En effet, le surcoût de la parallélisation n'est pas suffisamment élevé pour que l'asynchronisme soit rentable. Toutefois, l'asynchronisme n'est pas pénalisant étant donné que les temps de restitutions sont très proches. Notons qu'à partir de 16 processeurs, les performances des deux algorithmes s'effondrent. La charge de calcul devient trop faible par rapport au surcoût de la parallélisation.

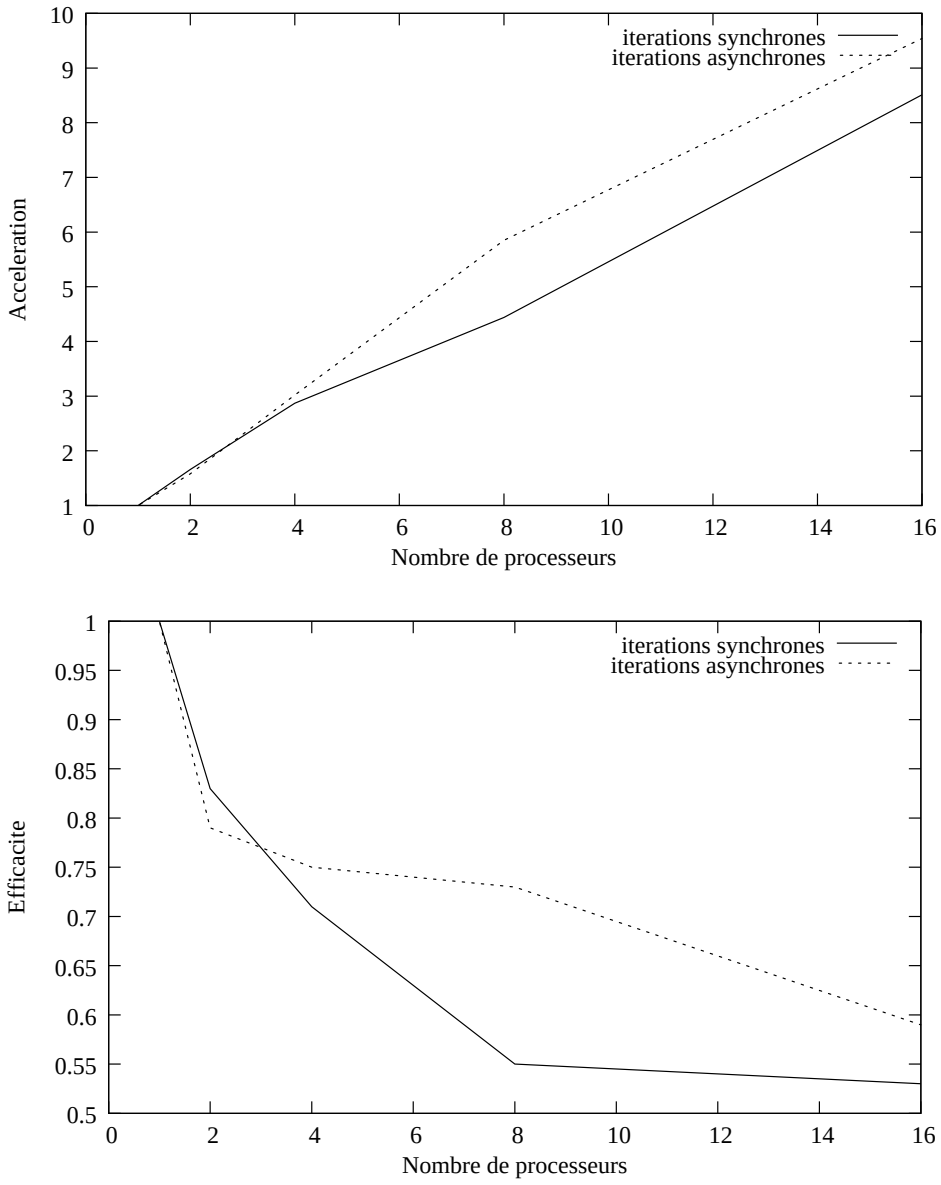


FIG. 3.24 – Accélération et efficacité des itérations synchrones et asynchrones pour un problème de convection-diffusion de dimension $400 \times 150 \times 20$ sur le p690+

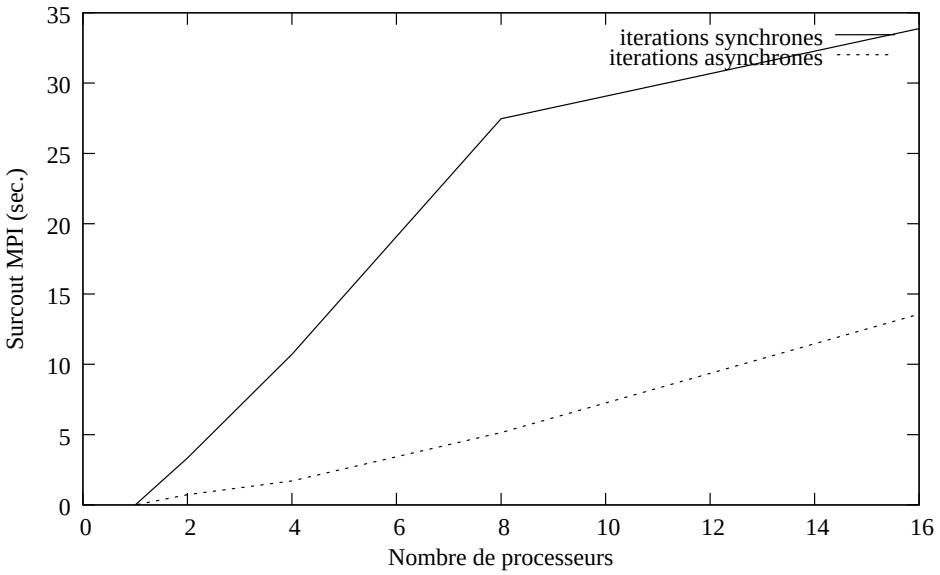


FIG. 3.25 – Comparaison du surcoût engendré par MPI dans les itérations synchrones et asynchrones pour un problème de convection-diffusion de dimension $400 \times 150 \times 20$ sur le p690+

Simulations parallèles

Les résultats des mesures de performance sur les simulations parallèles de l'électrophorèse sont donnés dans les figures 3.26, 3.27 et 3.28. Ces figures permettent de comparer les temps de restitution, les accélérations et les efficacités obtenus avec les 2 versions de l'algorithme de simulation :

1. l'équation de concentration résolue avec l'algorithme MPDATA ;
2. l'équation de concentration résolue avec le schéma implicite.

Les temps de restitution sont également présentés de manière détaillée dans le tableau 3.4. Les temps d'exécution consacrés à la résolution parallèle des systèmes linéaires sont indiqués entre parenthèses. Ces données montrent clairement que le poids du traitement séquentiel est plus élevé lorsque l'algorithme MPDATA est utilisé. Ce dernier s'élève à environ 1000 secondes tandis qu'avec le schéma implicite, environ 700 secondes sont consacrées au traitement séquentiel.

La comparaison des accélérations obtenues (figure 3.27) montre que la parallélisation est plus rentable lorsque le schéma implicite est utilisé. Les accélérations et les efficacités obtenues avec le schéma implicite sont supérieures à celles qui sont obtenues avec l'algorithme MPDATA. Ce comportement confirme le fait que plus le traitement séquentiel est important, moins l'accélération est importante.

En ce qui concerne l'algorithme MPDATA, le calcul des vitesses antidiffusives est difficile à paralléliser car il aurait fallu prendre en compte le décalage entre les

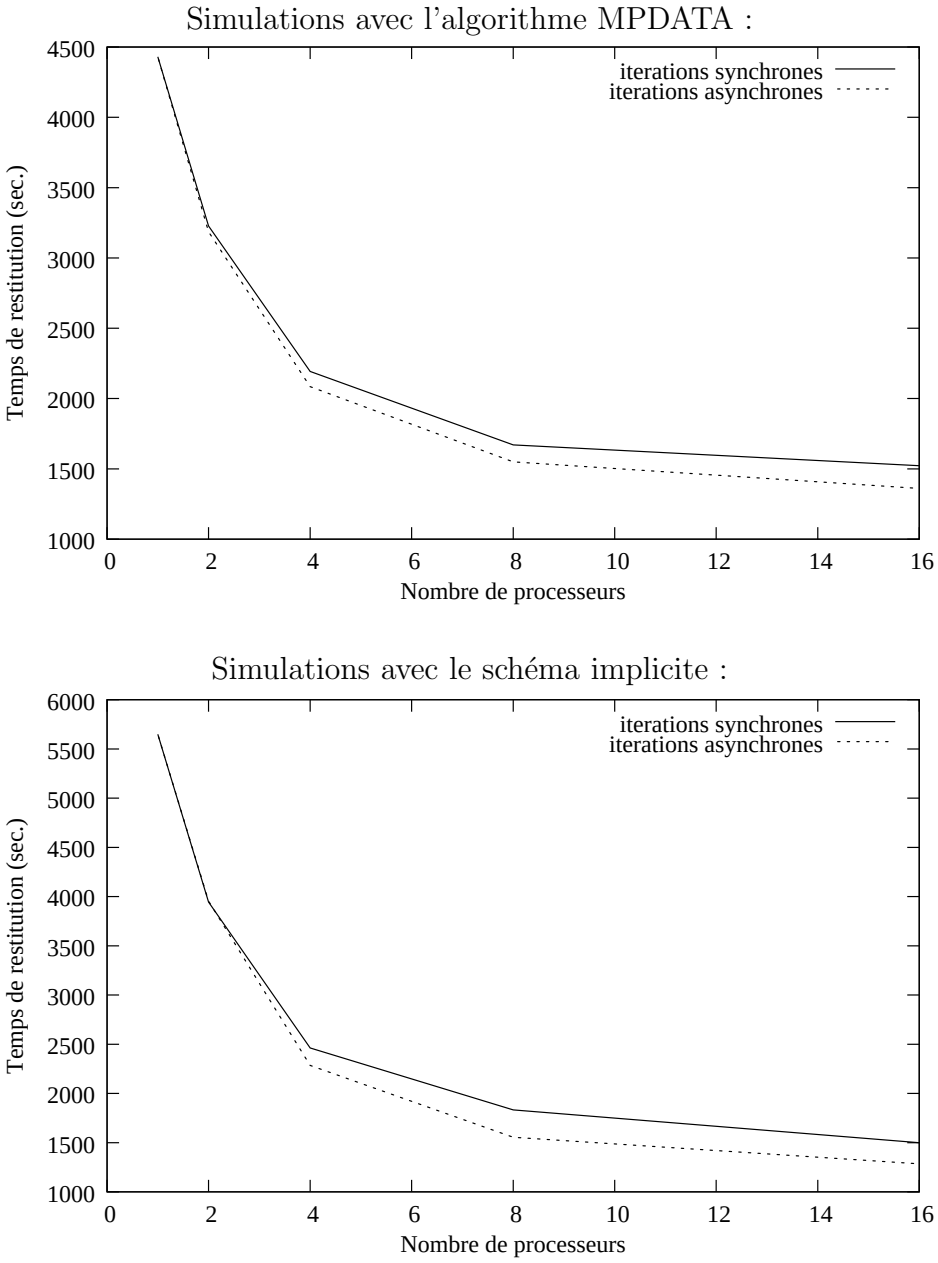


FIG. 3.26 – Temps de restitution des simulations parallèles de l'électrophorèse sur le p690+

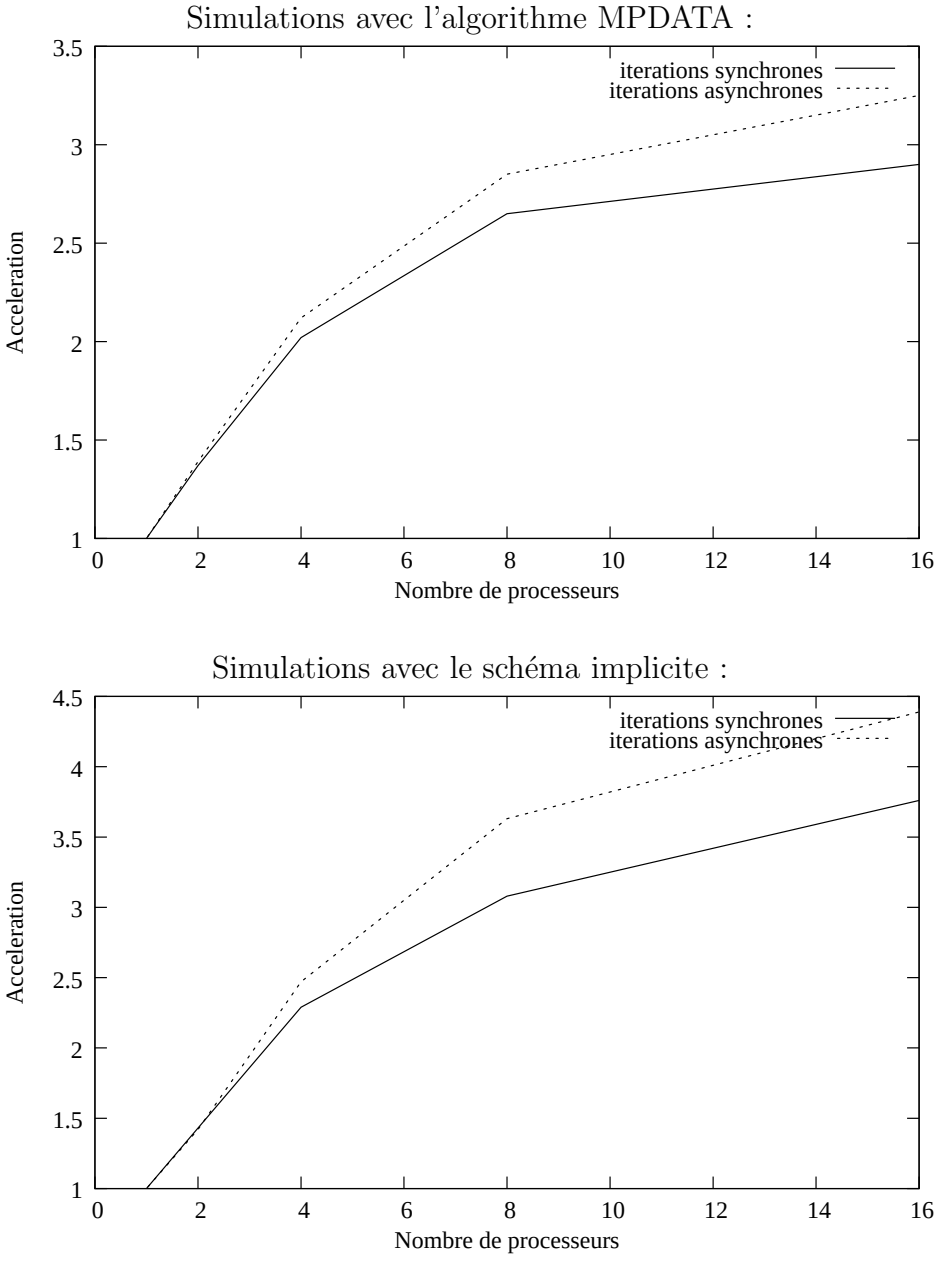


FIG. 3.27 – Accélérations obtenues pour des simulations parallèles de l'électrophorèse sur le p690+

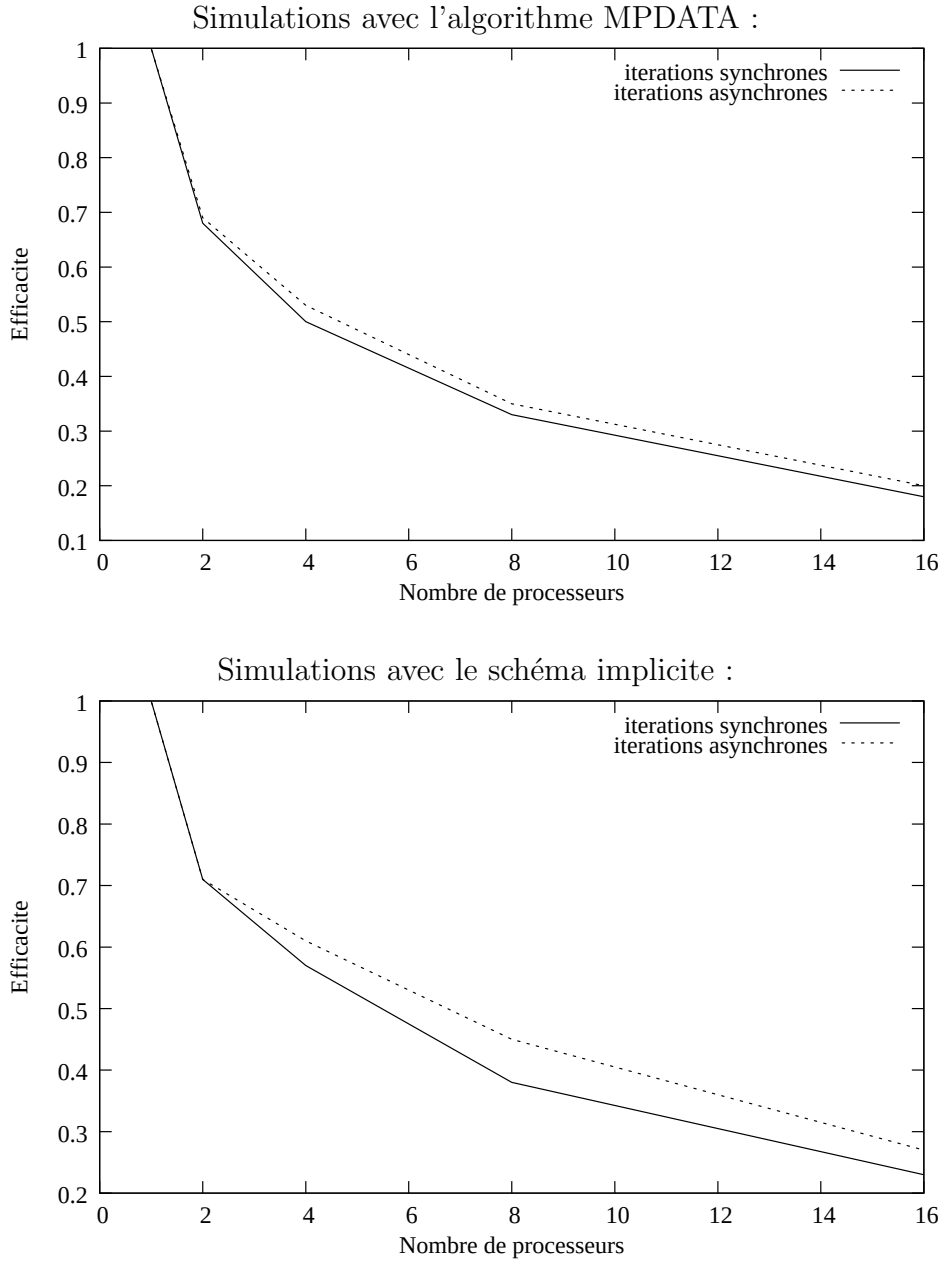


FIG. 3.28 – Efficacités obtenues pour des simulations parallèles de l'électrophorèse sur le p690+

nombre de proc.	Schéma implicite		Algorithme MPDATA	
	sync.	async.	sync.	async.
1	5649 (sec.)		4431 (sec.)	
2	3844 (3135)	3954 (3146)	3227 (2218)	3186 (2177)
4	2461 (1670)	2285 (1497)	2193 (1201)	2085 (1095)
8	1832 (1041)	1555 (769)	1670 (679)	1550 (559)
16	1499 (709)	1285 (499)	1525 (532)	1361 (371)

Les temps indiqués entre parenthèses sont associés à la partie parallélisée des simulations.

TAB. 3.4 – Récapitulatif des temps de restitution des simulations parallèles de l'électrophorèse sur le p690+.

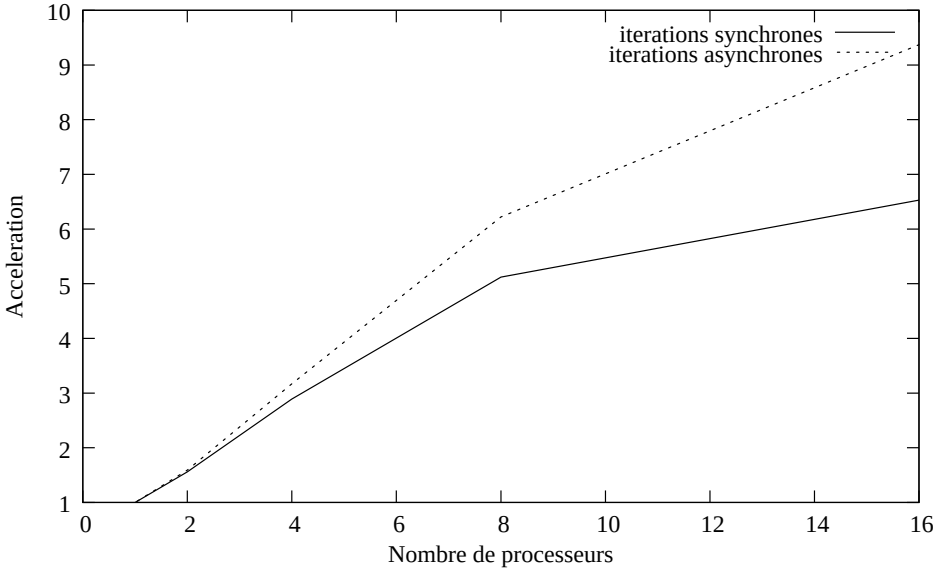
maillages lors de la décomposition du domaine. De façon plus générale, la simulation parallèle du procédé d'électrophorèse est pénalisée, du point de vue des performances, par d'autres traitements séquentiels dont la parallélisation est difficile. En guise d'exemple, nous pouvons citer le calcul de la force électrohydrodynamique, des matrices et des vecteurs faisant intervenir le vecteur vitesse. Le point commun de ces traitements séquentiels est la présence de calculs d'interpolation dans lesquels le décalage des maillages intervient.

Les courbes d'efficacité mettent en évidence l'effondrement des performances lorsque le nombre de processeurs atteint 16. Cet effondrement est cohérent avec le comportement du solveur dans ce contexte (*cf.* figure 3.24). Pour un maillage comportant $400 \times 150 \times 20$ points, 8 processeurs suffisent pour réduire le temps de restitution de manière significative. Ce comportement est confirmé par les courbes d'accélération et d'efficacité des parties parallélisées du simulateur, consacrées à la résolution des systèmes linéaires (*cf.* figures 3.29 et 3.30).

Remarque 3.3 Nous pouvons comparer l'évaluation de performance du solveur, effectuée dans le cadre de la résolution d'un problème de convection-diffusion de dimension $400 \times 150 \times 20$ (figure 3.24), avec les résultats obtenus dans le cadre de la résolution l'ensemble des systèmes linéaires de la simulation de l'électrophorèse (figures 3.29 et 3.30). Le solveur est visiblement plus performant dans le cadre de la simulation. Ce fait remarquable confirme que le parallélisme est d'autant plus efficace que la quantité de calcul est importante. En effet, une part très importante du temps d'exécution est consacrée à la résolution des équations de pression et de potentiel qui, contrairement à la résolution de l'équation (3.92), nécessitent de nombreuses itérations compte tenu du mauvais conditionnement de ces systèmes.

Nous pouvons remarquer que les différences d'efficacité entre les simulations synchrones et asynchrones (*cf.* figures 3.28 et 3.30) sont plus importantes lorsque le schéma implicite est utilisé. Ceci est dû au fait que l'équation de concentration est résolue en parallèle. Il est donc naturel que les simulations effectuées avec le schéma

Résolution des systèmes linéaires dans les simulations avec l'algorithme MPDATA :



Résolution des systèmes linéaires dans les simulations avec le schéma implicite :

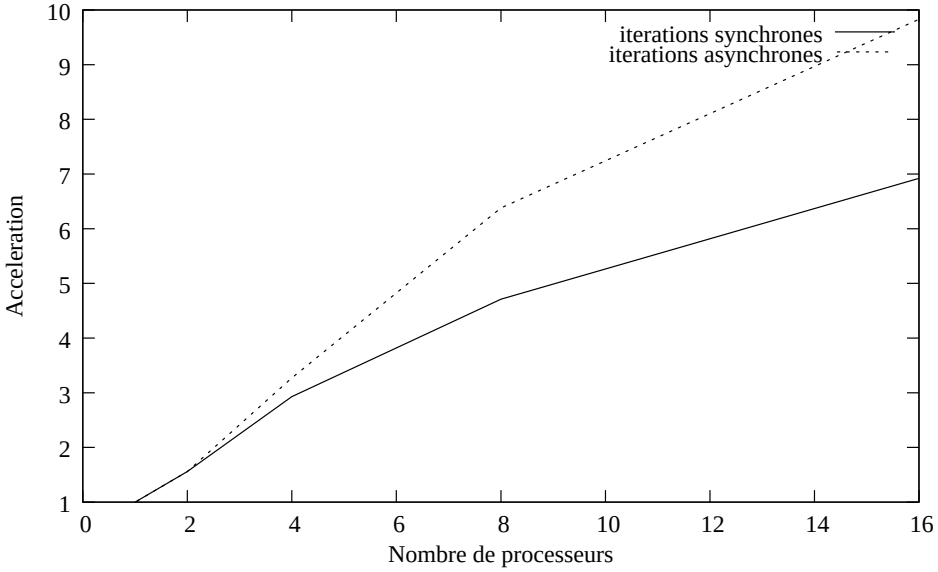
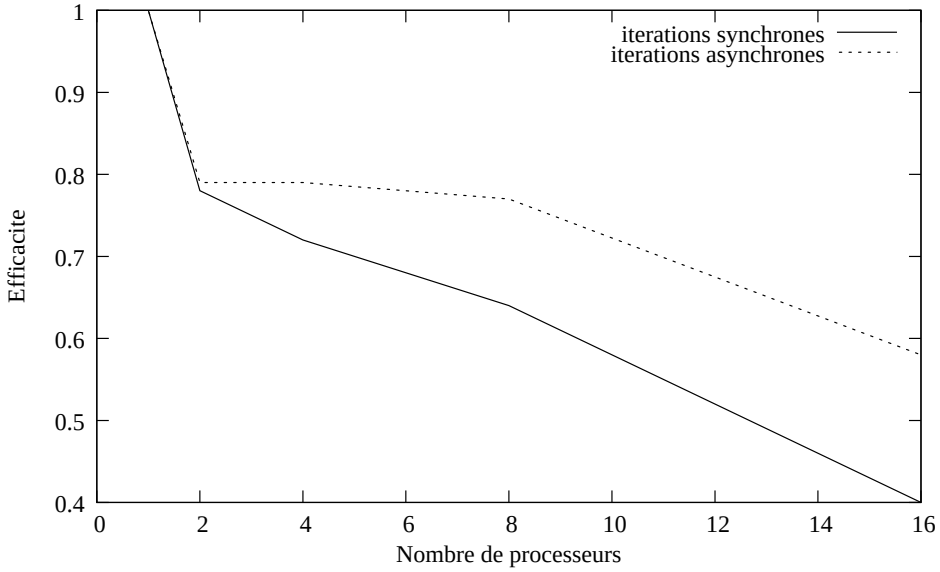


FIG. 3.29 – Accélérations obtenues pour la partie parallélisée des simulations de l'électrophorèse sur le p690+

Résolution des systèmes linéaires dans les simulations avec l'algorithme MPDATA :



Résolution des systèmes linéaires dans les simulations avec le schéma implicite :

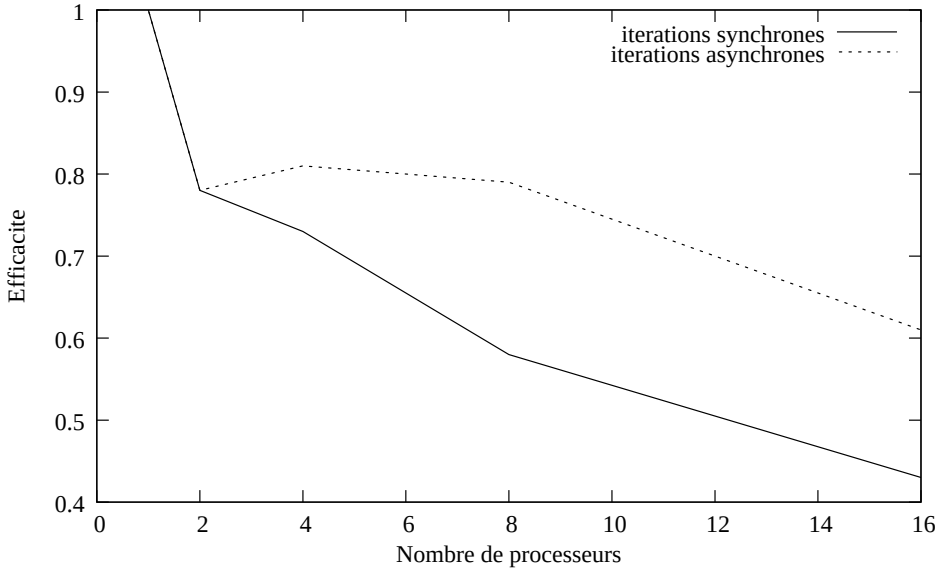


FIG. 3.30 – Efficacités obtenues pour la partie parallélisée des simulations de l'électrophorèse sur le p690+

implicite profitent mieux de l'asynchronisme.

À partir de cette étude, nous pouvons conclure que le gain de temps obtenu grâce aux itérations asynchrones est significatif. En effet, pour 20 pas de temps et avec 8 ou 16 processeurs, les simulations faisant appel au solveur asynchrone ont des temps de restitutions inférieurs, de 7 à 15 pourcent selon les cas, à celles qui font appel au solveur synchrone. De plus, l'efficacité de la parallélisation de la résolution des systèmes linéaires reste correcte malgré un effondrement pour 16 processeurs (*cf.* figure 3.30). Nous ne pouvons qu'être optimistes quant aux gains de performance qui seraient obtenus sur des simulations réelles avec quelques milliers de pas de temps, des maillages plus fins et plus de processeurs. Ce type d'expérimentation n'a pas été testé dans la mesure où nous n'avons pas eu accès de manière privilégiée à des moyens de calcul parallèle performants.

Chapitre 4

Le problème de l'obstacle

4.1 Introduction

Le problème de l'obstacle intervient dans des domaines variés tels que la mécanique ou les mathématiques financières. Dans le cas stationnaire, le problème de l'obstacle peut se formuler de la manière suivante :

$$\begin{cases} \text{Trouver } u^* \text{ tel que} \\ \Lambda(u^*) - f \geq 0,\ u^* \geq \phi \text{ presque partout dans } \Omega \\ (\Lambda(u^*) - f)(\phi - u^*) = 0 \text{ presque partout dans } \Omega \\ \text{Conditions aux limites} \end{cases} \tag{4.1}$$

où Ω est un ouvert de $\mathbb{R}^2$ ou de $\mathbb{R}^3$, Λ un opérateur elliptique, $f \in \mathrm{L}^2(\Omega)$ et ϕ une fonction donnée.

Dans la littérature, il existe de nombreuses formulations équivalentes au problème de l'obstacle et nous renvoyons à [62] pour plus d'informations. En effet, le problème (4.1) est équivalent à un problème complémentaire

$$\begin{cases} \sup(\Lambda(u^*) - f, \phi - u^*) = 0 \\ \text{Conditions aux limites} \end{cases}$$

ou bien à une inéquation variationnelle

$$\begin{cases} \text{Trouver } u^* \in K \text{ tel que} \\ \forall v \in K, \langle \Lambda(u^*)\,,\, v - u^* \rangle \geq \langle f\,,\, v - u^* \rangle \end{cases} \tag{4.2}$$

dans laquelle K est un ensemble convexe et fermé défini par

$$K = \{v \mid v \geq \phi \text{ presque partout dans } \Omega\}.$$

Ici, $\langle\,.\,,\,.\,\rangle$ représente le produit scalaire standard dans $\mathrm{L}^2(\Omega)$ défini par $\langle u\,,\, v\rangle = \int_\Omega u\, v\, dx$. Le problème de l'obstacle peut également être formulé comme un problème d'optimisation avec contrainte dans lequel la fonction coût est

$$J(v) = \frac{1}{2}\langle \Lambda(v)\,,\, v\rangle - \langle f\,,\, v\rangle.$$

On montre dans [62] que u^*, solution de l'inéquation variationnelle, est la solution du problème d'optimisation suivant :

$$\begin{cases} \text{Trouver } u^* \in K \text{ tel que} \\ \forall v \in K,\ J(u^*) \leq J(v) \end{cases} . \tag{4.3}$$

Il est possible de résoudre ce problème d'optimisation par une méthode de relaxation projetée sur le convexe K.

La résolution numérique du problème (4.3) peut conduire à la résolution de systèmes algébriques de grande taille. Dans ce contexte, les algorithmes numériques parallèles peuvent être employés afin de réduire les temps de restitution. Dans la suite, nous allons étudier les variantes parallèles synchrones et asynchrones de la méthode de Richardson projetée. Le but de cette étude est d'analyser, aussi bien sur le plan théorique qu'expérimental, l'algorithme itératif parallèle asynchrone. L'algorithme considéré étant une variante de la méthode de Richardson projetée, elle peut alors être modélisée par une application de point fixe. L'étude de convergence s'effectuera dans le cadre théorique décrit dans le chapitre 1. Nous décrirons également la mise en œuvre de l'algorithme considéré sur un IBM SP3 à l'aide de MPI. Puis nous comparerons les performances des versions synchrones et asynchrones de l'algorithme parallèle étudié, à l'aide de la résolution d'un problème test. Nous mettrons en évidence le fait que l'asynchronisme permet de réduire les temps d'attente dus aux synchronisations.

4.2 L'algorithme de Richardson asynchrone

L'algorithme de Richardson parallèle asynchrone est modélisé à l'aide des concepts associés aux itération parallèles asynchrones classiques [72]. Nous reprendrons donc les notations employées dans la section 1.2. Soient β un entier naturel positif et E un espace de Hilbert décomposé comme suit :

$$E = \prod_{i=1}^{\beta} E_i$$

où pour tout $i \in \{1, \ldots, \beta\}$, E_i est un espace de Hilbert muni d'un produit scalaire noté $\langle\, .\, ,\, .\,\rangle_i$. La norme associée à ce dernier est notée $|\, .\,|_i$. Tout vecteur V de E se décompose sous la forme $V = (v_1, \ldots, v_\beta)$. Ainsi, le produit scalaire sur E est défini comme suit :

$$\forall (U, V) \in E^2,\ \langle U\, ,\, V\rangle = \sum_{i=1}^{\beta} \langle u_i\, ,\, v_i\rangle_i$$

et la norme standard sur E associée sera notée $\|\, .\,\|$. Dans la suite, nous nous proposons d'analyser la convergence de l'algorithme de Richardson sur la base de la formulation variationnelle (4.2) du problème de l'obstacle.

Soit A une application linéaire définie sur E. Sa décomposition dans l'espace produit E est de la forme :

$$\forall V \in E,\ A.V = (A_1.V, \ldots, A_\beta.V).$$

Dans la formulation classique de l'algorithme de Richardson, la convergence est assurée par une condition de coercivité :

$$\forall V \in E,\ \langle A.V\ ,\ V\rangle \geq c\,\|V\|^2. \tag{4.4}$$

où c est une constante réelle strictement positive. Dans la formulation par blocs de l'algorithme de Richardson qui est étudiée ici, la convergence est assurée par une hypothèse plus faible qui prend en compte la décomposition en sous-domaines, nécessaire à la parallélisation :

Hypothèse 4.1

$$\forall i \in \{1, \ldots, \beta\},\ \forall V \in E,\ \langle A_i.V\ ,\ v_i\rangle_i \geq \sum_{j=1}^{\beta} n_{ij}|v_i|_i|v_j|_j \tag{4.5}$$

où

$$N = (n_{ij})_{1\leq i,j\leq\beta} \textit{ est une M-matrice de taille } \beta \times \beta. \tag{4.6}$$

Les coefficients diagonaux n_{ii} de la matrice N correspondent aux plus petites valeurs propres des blocs diagonaux A_{ii} de la matrice A, lorsque ces derniers sont définies positives :

$$\forall v_i \in E_i,\ \langle A_{ii}.v_i\ ,\ v_i\rangle \geq n_{ii}\,|v_i|_i^2. \tag{4.7}$$

Les opposés des coefficients hors-diagonaux de N sont des majorants des normes matricielles des blocs hors-diagonaux $(A_{ij})_{i\neq j}$:

$$]|A_{ij}|[\leq -n_{ij}. \tag{4.8}$$

On montre dans [88] que la condition (4.5) est vérifiée si et seulement si A satisfait les conditions (4.7) et (4.8).

Soit $B = (b_1, \ldots, b_\beta)$ un vecteur de E représentant la discrétisation de f. Soient $(K_i)_{1\leq i\leq\beta}$ une famille de β parties fermées convexes tels que : $\forall i \in \{1, \ldots, \beta\}$, $K_i \subset E_i$. Nous considérons alors l'ensemble fermé et convexe suivant :

$$K = \prod_{i=1}^{\beta} K_i$$

avec lequel nous définissons l'inéquation variationnelle suivante :

$$\left\{\begin{array}{l} \text{Trouver } U^* \in K \text{ tel que} \\ \forall V \in K, \langle A.U^*\ ,\ V - U^*\rangle \geq \langle B\ ,\ V - U^*\rangle \end{array}\right. \tag{4.9}$$

Soit P_K l'opérateur de projection sur l'ensemble fermé et convexe K. Cet opérateur se décompose de manière suivante :

$$\forall V \in E,\ P_K(V) = (P_{K_1}(v_1), \ldots, P_{K_\beta}(v_\beta))$$

où pour tout i, l'opérateur P_{K_i} est la projection de E_i sur K_i.

Soit δ un réel strictement positif. L'algorithme de Richardson est modélisé par l'application de point fixe suivante :

$$\forall V \in E,\ F_\delta(V) = P_K(V - \delta\,(A.V - B)) \tag{4.10}$$

dont la décomposition par blocs s'écrit :

$$\forall V \in E,\ F_\delta(V) = (F_{1,\delta}(V), \ldots, F_{\beta,\delta}(V)) \tag{4.11}$$

avec

$$\forall i \in \{1, \ldots, \beta\},\ \forall V \in E,\ F_{i,\delta}(V) = P_{K_i}(v_i - \delta(A_i.V - b_i)). \tag{4.12}$$

Les communications interviennent au niveau de l'opération $A_i.V - b_i$ qui s'écrit dans le cas où A est considérée comme une matrice :

$$(A_{ii}.v_i - b_i) + \sum_{j \neq i} A_{ij}.\tilde{v}_j.$$

On considère ici une décomposition par blocs de chaque composante A_i, qui prend en compte le découpage du vecteur itéré V. Le vecteur $\tilde{v}_j$ représente la composante envoyée par le $j^{\text{ème}}$ processeur.

Proposition 4.1
Sous les hypothèses (4.5) et (4.6), il existe un réel δ_0 strictement positif tel que $\forall \delta \in]0, \delta_0[$, les itération parallèles asynchrones classiques (définition 1.3), associées à l'application de point fixe F_δ de l'algorithme de Richardson (4.10), convergent vers U^*, la solution unique du problème de l'obstacle (4.9).

Démonstration Il s'agit de trouver un ensemble de réels δ tels que l'application F_δ soit une contraction, sachant que la projection est une contraction. Nous nous baserons sur le théorème de convergence en norme uniforme avec poids de M.N. El Tarazi [36] (*cf.* le théorème 1.2 du chapitre 1). Munissons l'espace E de la norme uniforme avec poids :

$$\|V\|_\gamma = \max_{1 \leq i \leq \beta} \frac{|v_i|_i}{\gamma_i}$$

dans laquelle le vecteur des poids $\gamma = (\gamma_1, \ldots, \gamma_\beta) \in \mathbb{R}^\beta$ est le vecteur propre associé au rayon spectral de la matrice de Jacobi de N, notée J. L'existence et la stricte positivité du vecteur γ sont établies à l'aide du théorème de Perron-Frobenius [101]. En effet, étant donné que N est une M-matrice, J est une matrice non-négative.

Ainsi, J et γ vérifient

$$\begin{aligned} J.\gamma &= \rho(J)\,\gamma & (4.13)\\ \gamma &> 0. & (4.14)\end{aligned}$$

Soit $V \in K$, et posons $W = V - U^*$. D'après les notations employées précédemment, on a $w_i = v_i - u_i^*$. Soit $i \in \{1, \ldots, \beta\}$. Posons

$$\begin{aligned} R_i &= \frac{1}{n_{ii}} \frac{|w_i - \delta A_i.W|_i^2}{\gamma_i^2} \\ &= \frac{1}{n_{ii}} \frac{|w_i|_i^2 - 2\delta\langle A_i.W\,,\, w_i\rangle + \delta^2 |A_i.W|_i^2}{\gamma_i^2} \end{aligned}$$

Puisque $A_i \in \mathcal{L}(V, V_i)$,

$$\exists M_i \geq 0,\ \forall W \in E,\ |A_i.W|_i \leq M_i \|W\|.$$

De plus, d'après l'hypothèse (4.5), l'expression R_i est majorée par :

$$R_i \leq \frac{1}{n_{ii}} \frac{|w_i|_i^2 - 2\,\delta \sum_{j=1}^{\beta} n_{ij} |w_i|_i |w_j|_j + \delta^2 M^2 \|W\|^2}{\gamma_i^2}$$

où $M = \max_{1\leq i\leq\beta} M_i$. Donc,

$$R_i \leq \left(\frac{1}{n_{ii}} - 2\,\delta\right) \frac{|w_i|_i^2}{\gamma_i^2} - 2\,\delta \left(\sum_{j\neq i} \frac{n_{ij}}{n_{ii}} \gamma_j \frac{|w_j|_j}{\gamma_j}\right) \frac{|w_i|_i}{\gamma_i^2} + \frac{\delta^2 M^2}{n_{ii}} \frac{\|W\|^2}{\gamma_i^2}.$$

Et d'après (4.13),

$$R_i \leq \left(\frac{1 - 2\,\delta\, n_{ii}}{n_{ii}}\right) \frac{|w_i|_i^2}{\gamma_i^2} + 2\,\delta\,\rho(J) \left(\max_{1\leq j\leq\beta} \frac{|w_j|_j}{\gamma_j}\right) \frac{|w_i|_i}{\gamma_i} + \frac{\delta^2 M^2}{n_{ii}} \frac{\|W\|^2}{\gamma_i^2}.$$

Donc

$$R_i \leq \frac{1}{n_{ii}} \left(\left(1 - 2\,\delta\, n_{ii}(1 - \rho(J))\right) \|W\|_\gamma^2 + \frac{\delta^2 M^2}{\gamma_i^2} \|W\|^2 \right).$$

Posons

$$\begin{aligned} \underline{\gamma} &= \min_{1\leq i\leq\beta} \gamma_i \\ \underline{n} &= \min_{1\leq i\leq\beta} n_{ii}. \end{aligned}$$

Puisque

$$\|W\|^2 = \sum_{i=1}^{\beta} \gamma_i^2 \frac{|w_i|_i^2}{\gamma_i^2} \leq \|\gamma\|_2^2 \|W\|_\gamma^2$$

où $\|.\|_2$ est la norme euclidienne de $\mathbb{R}^\beta$. Nous obtenons finalement une majoration à l'aide d'un trinôme du second degré :

$$\|W - \delta A.W\|_\gamma^2 \leq \left(1 - 2\,\underline{n}(1-\rho(J))\,\delta + M^2 \frac{\|\gamma\|_2^2}{\underline{\gamma}^2}\delta^2\right) \|W\|_\gamma^2$$

sachant que

$$\|W - \delta A.W\|_\gamma^2 = \max_{1\leq i\leq \beta} n_{ii}\, R_i.$$

Par conséquent, δ étant un réel positif, l'algorithme itératif considéré converge si $\delta \in]0, \delta_0[$, avec

$$\delta_0 = \frac{2\underline{n}(1-\rho(J))\underline{\gamma}^2}{M^2\|\gamma\|_2^2}. \tag{4.15}$$

□

4.3 Étude expérimentale

4.3.1 Le problème test

Le problème test avec lequel nous avons évalué les performances des algorithmes de Richardson parallèles synchrones et asynchrones est le problème de l'obstacle stationnaire suivant :

$$\begin{cases} \text{Trouver } u^* \in K \text{ tel que} \\ -\Delta u^* - f \geq 0,\ u^* \geq 0 \text{ presque partout dans } [0\,,\,1]^3 \\ (-\Delta u^* - f)u^* = 0 \text{ presque partout dans } [0\,,\,1]^3 \\ u^*_{/\partial[0\,,1]^3} = 0 \end{cases} \tag{4.16}$$

où Δ est le Laplacien. Le convexe K est défini comme :

$$K = \{u : [0\,,\,1]^3 \mapsto \mathbb{R} \mid u \geq 0\}.$$

La matrice A est calculée à l'aide d'une discrétisation par différences finies à 7 points, avec un pas de discrétisation uniforme.

La décomposition par blocs du vecteur itéré est en adéquation avec la structure par blocs de la matrice de discrétisation. Le vecteur itéré de dimension n^3 est décomposé en n blocs de n^2 valeurs. Chaque bloc correspond au plan $[0\,,\,1]^2$ et chaque processus prend en charge plusieurs blocs. Les n blocs sont répartis sur β processeurs sachant que $\beta < n$. Avec ce type de décomposition, chaque processus ne communique qu'avec 2 voisins et les messages ont la même taille que les blocs.

Avec ce type de découpage, le Laplacien discret est considéré comme une matrice tridiagonale par blocs. Soit I la matrice identité de dimension $n^2 \times n^2$ et soit $h = \frac{1}{n+1}$

le pas de discrétisation. La décomposition par blocs de la matrice A est de la forme :

$$\begin{aligned} A_{i,i\pm1} &= -\frac{1}{h^2}I \\ A_{i,i} &= \frac{2}{h^2}I + A' \end{aligned}$$

où A' est le Laplacien discrétisé sur $[0\,,\,1]^2$. Sachant que A' est une matrice définie positive [66, 73, 49, 87, 88, 89], les hypothèses (4.5) et (4.6) sont vérifiées avec :

$$N = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$

qui est une M-matrice [101]. La convergence des algorithmes de Richardson synchrones et asynchrones est donc établie à l'aide de la proposition 4.1. La valeur de δ_0 a été déterminée expérimentalement par dichotomie.

Remarque 4.1 Dans le cadre de ce problème test, la convergence peut également être établie à l'aide d'une méthode plus générale. En considérant une décomposition par points du problème (4.16) et d'après [88, 73], les hypothèses (4.5) et (4.6) sont vérifiées. Alors, l'algorithme de Richardson parallèle asynchrone converge pour la décomposition par points. En appliquant les résultats de [73], on obtient la convergence dans le cas de n'importe quelle décomposition plus grossière.

4.3.2 L'implémentation

Nous pouvons à ce stade formuler un algorithme parallèle qui ne prend pas encore en compte l'implémentation synchrone ou asynchrone des communications. Soit $k \in \{1, \ldots, \beta\}$. Le $k^{\text{ème}}$ processus prend alors en charge les blocs

$$u_{\mathrm{p}(k)}, u_{\mathrm{p}(k)+1}, \ldots, u_{\mathrm{d}(k)-1}, u_{\mathrm{d}(k)}$$

où $\mathrm{p}(k)$ et $\mathrm{d}(k)$ sont respectivement les indices du premier et du dernier bloc pris en charge par le $k^{\text{ème}}$ processus. Les relaxations par blocs effectuées par chaque processus sont définies conformément à l'équation (4.12). Étant donnée la structure tridiagonale par blocs de la matrice A, une relaxation s'écrit :

$$F_{i,\delta}(x, y, z) = P_{K_i}(y - \delta(A_{i,i-1}.x + A_{i,i}.y + A_{i,i+1}.z - b_i))$$

où x, y et z sont des blocs de dimension n^2. L'algorithme de principe est donné dans la figure 4.1. Il a été implémenté avec le langage C. Les cas limites $k = 1$ et $k = \beta$ ne sont pas pris en compte dans la figure 4.1. Notons que le premier processus ne communique

• Algorithme exécuté par le $k^{\text{ème}}$ processus :

Envoyer $u^0_{\mathrm{d}(k)}$ au processus $k+1$ [message 1]
tant que convergence globale non détectée
    $i \leftarrow \mathrm{p}(k)$
    Recevoir $\tilde{u}_{i-1}$ de la part du processus $k-1$ [message 1]
    $u_i^{p+1} \leftarrow F_{i,\delta}(\tilde{u}_{i-1}, u_i^p, u_{i+1}^p)$
    Envoyer u_i^{p+1} au processus $k-1$ [message 2]

    pour $i = \mathrm{p}(k)+1, \ldots, \mathrm{d}(k)-1$
        $u_i^{p+1} \leftarrow F_{i,\delta}(u_{i-1}^{p+1}, u_i^p, u_{i+1}^p)$
    fin

    $i \leftarrow \mathrm{d}(k)$
    Recevoir $\tilde{u}_{i+1}$ de la part du processus $k+1$ [message 2]
    $u_i^{p+1} \leftarrow F_{i,\delta}(u_{i-1}^{p+1}, u_i^p, \tilde{u}_{i+1})$
    Envoyer u_i^{p+1} au processus $k+1$ [message 1]
fin

FIG. 4.1 – Algorithme de Richardson parallèle

qu'avec le second processus, de même que le $\beta^{\text{ème}}$ processus ne communique qu'avec le $(\beta-1)^{\text{ème}}$ processus.

Conformément aux notations du chapitre 1, les blocs $\tilde{u}_i$ représentent les messages reçus. Dans le cas synchrone, les messages ne subissent pas de retard. Les échanges de messages ont été implémentés avec les requêtes persistantes de la bibliothèque MPI. Les messages sont échangés selon une politique qui est semblable à celle qui est employée dans l'implémentation du solveur décrit dans le chapitre 2. La différence réside dans la simplicité du découpage qui est imposé ici. Il n'y a que 4 canaux de communication à gérer et lorsqu'on cherche à rendre disponibles les composantes récemment calculées le plus tôt possible, il n'y a aucun risque d'interblocage dans le cas synchrone. Les problèmes d'efficacité soulevés dans la section 2.3.4 ne se sont donc pas posés. Dans les 2 versions de l'algorithme parallèle, la politique suivante a été appliquée :

- soumettre les requêtes d'envoi juste après la mise à jour d'un message,
- soumettre les requêtes de réception juste après avoir exploité les données reçues,
- attendre ou tester la fin d'une requête juste avant la lecture ou la mise à jour d'un message.

La détection de la convergence globale est réalisée à l'aide de l'algorithme décrit dans la section 2.5. Le critère de convergence locale est la norme uniforme de la différence entre 2 itérés successifs (le seuil de tolérance est fixeé à 10^{-11}).

Remarque 4.2 Les vecteurs ont été rangés en mémoire dans des tableaux à 3 entrées de la forme

```
double TAB[NMAX][NMAX][NMAX] ;
```

afin de pouvoir accéder aux valeurs à l'aide des coordonnées cartésiennes tridimensionnelles des différents nœuds du maillage. Par conséquent, lorsque la dimension du système algébrique est inférieure à la dimension du tableau, les données ne sont plus rangées en mémoire de façon contiguë. Le paquetage explicite des données s'impose dans ce cas.
En revanche l'utilisation d'un tableau classique de la forme

```
double TAB[NMAX] ;
```

aurait permis de conserver la contiguïté des données en mémoire de sorte que le paquetage explicite ne soit plus nécessaire. La conversion des coordonnées cartésiennes tridimensionnelles en indice de tableau est dans ce cas à la charge du programmeur.

4.3.3 Étude de performance

Les essais ont été menés sur l'IBM SP3 de l'IDRIS avec 2, 4, 8, 16 et 32 processeurs. À titre indicatif, ce calculateur est un cluster de SMP dans lequel chaque nœud possède 16 processeurs partageant une mémoire commune. Les accélérations et

les efficacités sont calculés en fonction d'un algorithme séquentiel dans lequel aucun paquetage ou dépaquetage n'est effectué.

Les solveurs ont été évalués avec deux problèmes de tailles différentes : $80 \times 80 \times 80$ et $96 \times 96 \times 96$. Le tableau 4.1 donne les temps de restitution obtenus. Le nombre de

nombre de proc.	$80 \times 80 \times 80$		$96 \times 96 \times 96$	
	sync.	async.	sync.	async.
1	332 (sec.)		708 (sec.)	
2 (1)	148	149	358	349
4 (1)	68	63	161	163
8 (1)	36	33	80	76
16 (1)	21	18	46	40
32 (2)	-	-	35	23

TAB. 4.1 – Récapitulatif des temps de restitution sur le SP3 des algorithmes de Richardson parallèles pour un problème de l'obstacle

nœuds du SP3 utilisés est écrit entre parenthèses dans la colonne indiquant le nombre de processeurs. Les 2 algorithmes parallèles donnent des résultats très proches. En dessous de 8 processeurs, l'algorithme synchrone peut avoir de meilleurs temps de restitution que l'algorithme asynchrone. En revanche, à partir de 8 processeurs, la version asynchrone est toujours meilleure que la version synchrone.

Les courbes d'accélération et d'efficacités sont donnés dans les figures 4.2 et 4.3. Pour le problème de dimension $80 \times 80 \times 80$, l'algorithme asynchrone se démarque de l'algorithme synchrone à partir de 4 processeurs. Dans les mesures effectuées avec le problème de dimension $96 \times 96 \times 96$, l'algorithme asynchrone se démarque de l'algorithme synchrone à partir de 8 processeurs. Globalement, les 2 versions de l'algorithme ont des performances équivalentes dans le cas où un seul nœud du SP3 est utilisé. Une différence réellement significative entre les deux algorithmes parallèles apparaît pour 32 processeurs. Dans ce cas particulier, 2 nœuds du SP3 sont utilisés. La perte d'efficacité de l'algorithme synchrone s'explique de la manière suivante :

- Lorsque 32 processeurs sont utilisés, chaque processus ne prend en charge que 3 blocs. Puisque les messages et les blocs ont la même taille, le surcoût engendré par les communications atteint une proportion significative.
- Deux des processus sont impliqués dans des communications faisant intervenir le réseau d'interconnexion entre les nœuds. Ce type de communication étant plus lent, cela entraîne un déséquilibrage de charge qui retarde tous les autres processus.

Il est clair que l'algorithme asynchrone est moins sensible aux problèmes liés aux communications que l'algorithme synchrone. Notons que l'algorithme asynchrone subit aussi une chute de performance avec 32 processeurs. Toutefois, l'efficacité reste bonne.

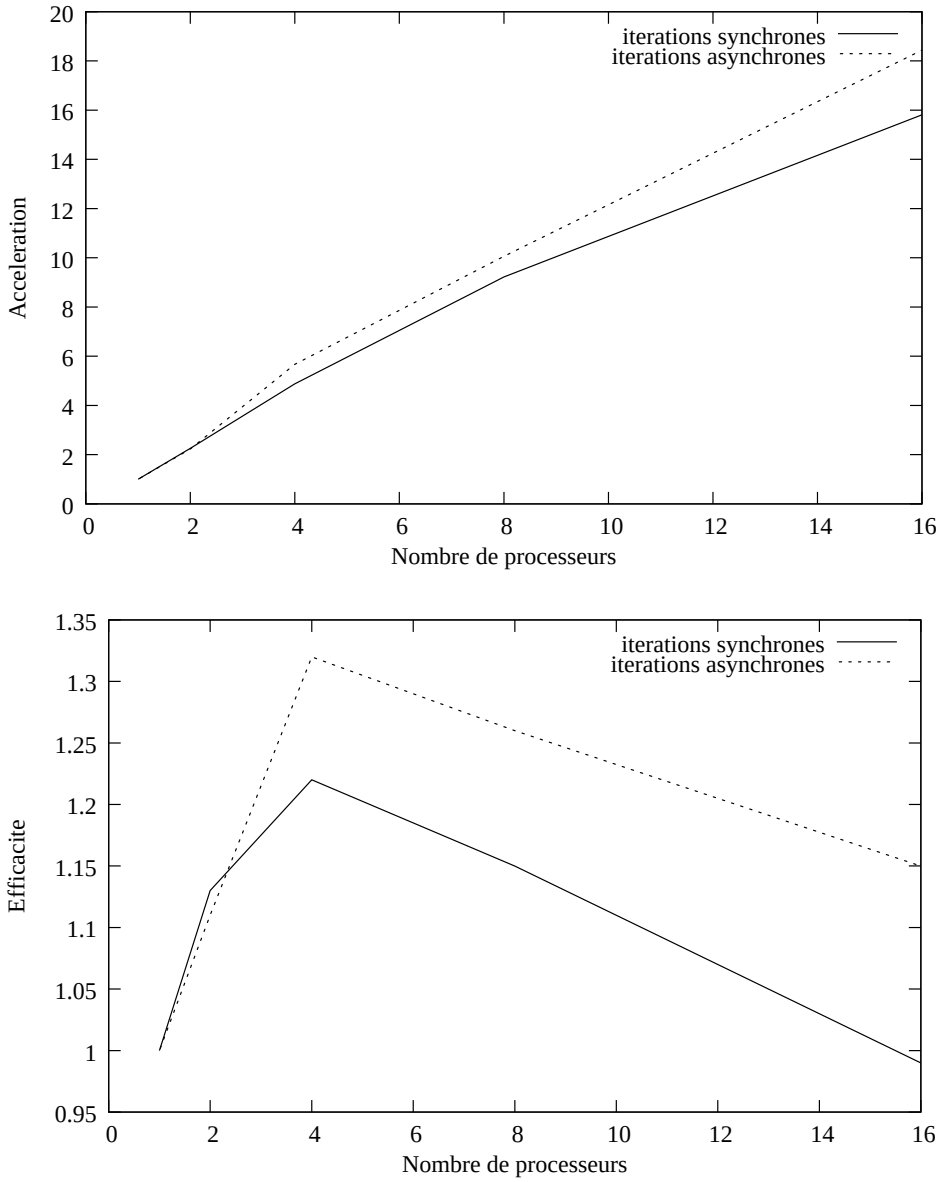


FIG. 4.2 – Accélération et efficacité des algorithmes de Richardson parallèles sur le SP3 testés pour un problème de l'obstacle avec $80 \times 80 \times 80$ points de discrétisation

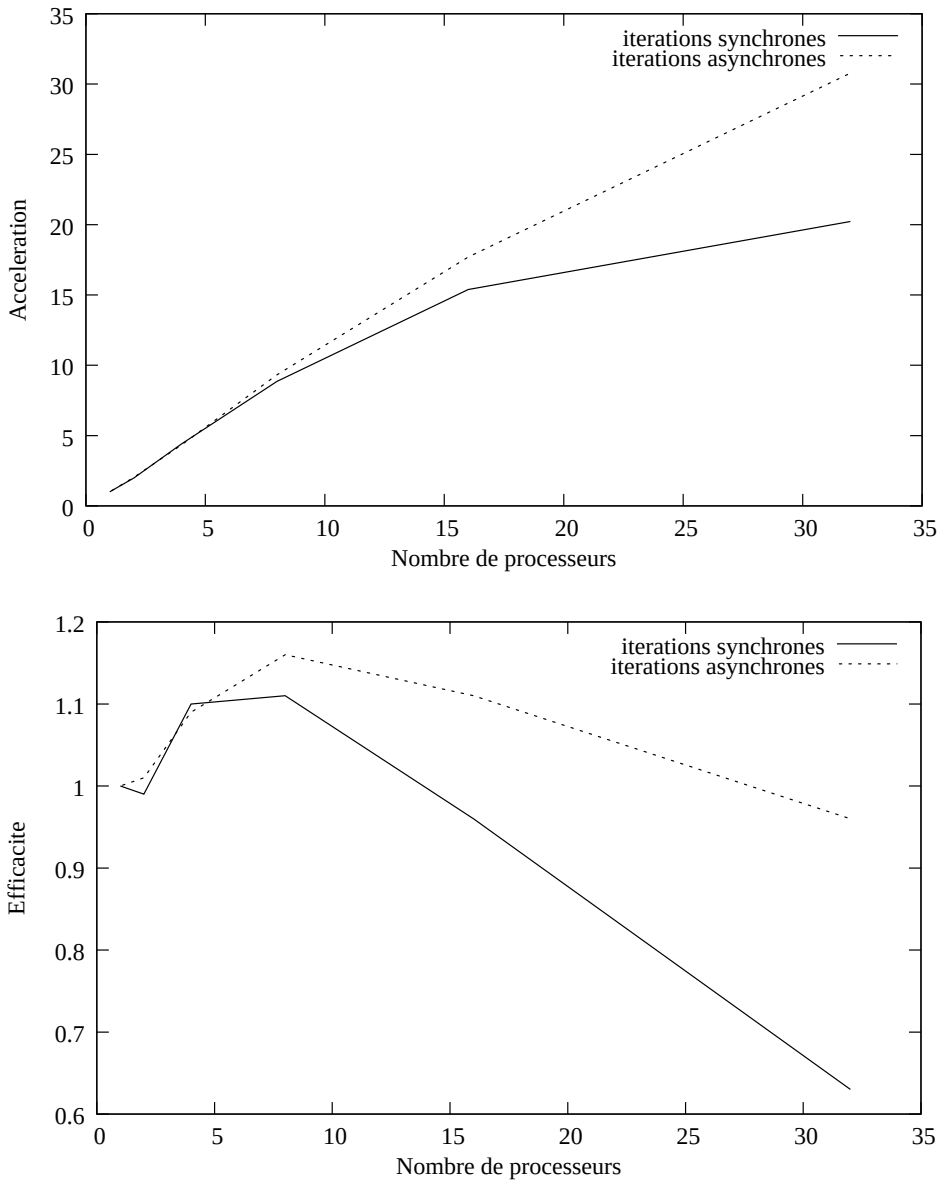


FIG. 4.3 – Accélération et efficacité des algorithmes de Richardson parallèles sur le SP3 testés pour un problème de l'obstacle avec $96 \times 96 \times 96$ points de discrétisation

Bien que l'ordre de traitement des blocs soit perturbé par la parallélisation, le nombre d'itérations effectuées par l'algorithme synchrone est constant pour chaque cas (*cf.* figure 4.4). En ce qui concerne l'étude du surcoût en itération engendré par les retards, le comportement de l'algorithme asynchrone confirme les résultats obtenus avec le problème de convection-diffusion dans la section 2.6. Le nombre d'itérations a tendance à croître quand le nombre de processeurs devient élevé.

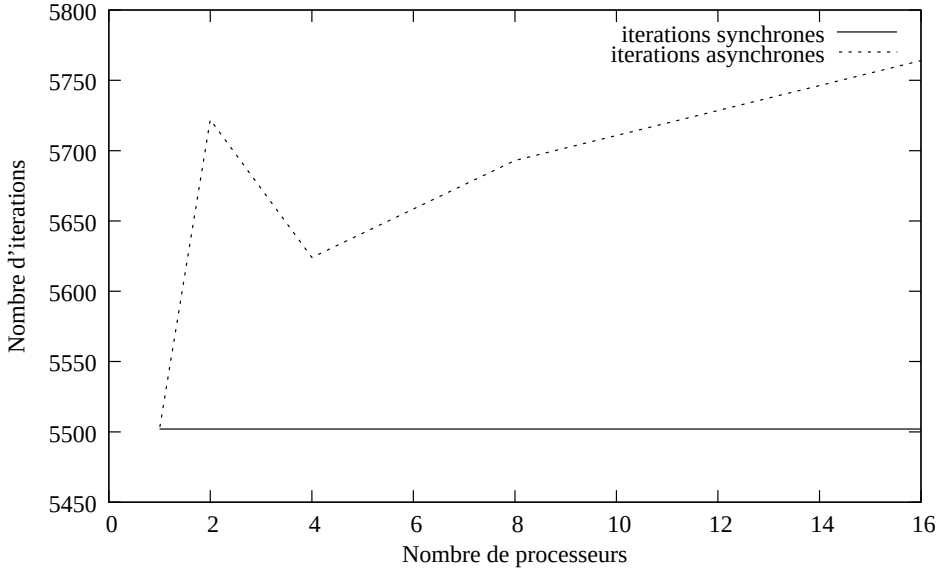
nombre de proc.	$80 \times 80 \times 80$		$96 \times 96 \times 96$	
	sync.	async.	sync.	async.
1	5502 itérations		7662 itérations	
2 (1)	5502	5722	7662	7740
4 (1)	5502	5624	7662	8099
8 (1)	5502	5693	7662	7863
16 (1)	5502	5764	7662	7877
32 (2)	-	-	7662	8012

TAB. 4.2 – Récapitulatif du nombre d'itérations effectuées par les algorithmes de Richardson parallèles pour un problème de l'obstacle

nombre de proc.	$80 \times 80 \times 80$		$96 \times 96 \times 96$	
	sync.	async.	sync.	async.
2 (1)	2.26	2.23	1.98	2.03
4 (1)	4.88	5.67	4.4	4.34
8 (1)	9.22	10.06	8.85	9.32
16 (1)	15.81	18.44	15.39	17.7
32 (2)	-	-	20.23	30.78

TAB. 4.3 – Accélération des algorithmes de Richardson parallèles pour un problème de l'obstacle sur le SP3

Résolution d'un problème de l'obstacle avec $80 \times 80 \times 80$ points de discrétisation



Résolution d'un problème de l'obstacle avec $96 \times 96 \times 96$ points de discrétisation

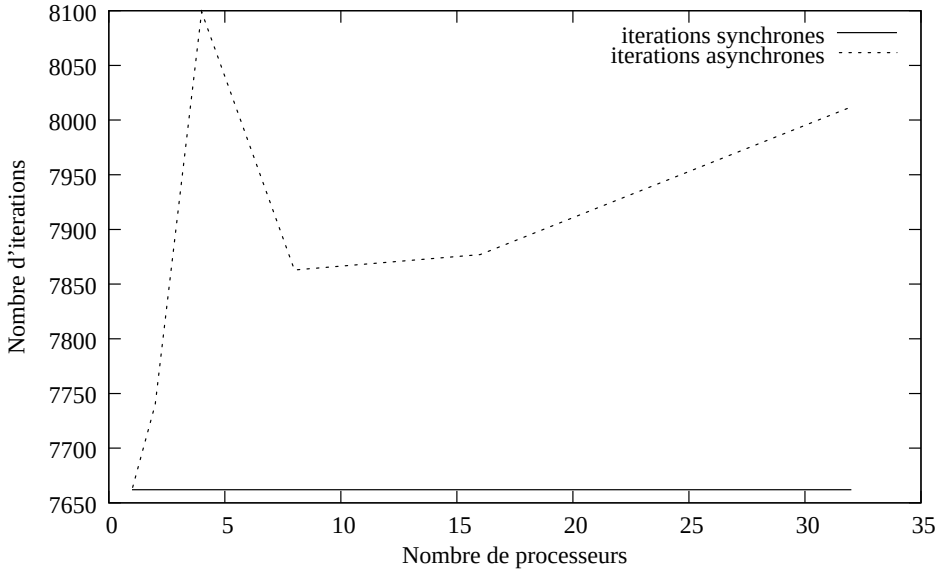


FIG. 4.4 – Nombre d'itérations effectuées par les algorithmes de Richardson parallèles pour la résolution d'un problème de l'obstacle sur le SP3

nombre de proc.	$80 \times 80 \times 80$		$96 \times 96 \times 96$	
	sync.	async.	sync.	async.
2 (1)	1.13	1.11	0.99	1.01
4 (1)	1.22	1.22	1.10	1.09
8 (1)	1.15	1.15	1.11	1.16
16 (1)	0.99	0.99	0.96	1.11
32 (2)	-	-	0.63	0.96

TAB. 4.4 – Efficacité des algorithmes de Richardson parallèles pour un problème de l'obstacle sur le SP3

Annexe A

Comparaison avec les résultats antérieurs

Dans les sections 2.3 et 2.4, seule la description de la version actuelle du solveur datant de 2003 a été faite afin de simplifier l'exposé. Deux versions du solveur ont été implémentées auparavant. Cette partie a pour but de mettre en évidence l'évolution de l'implémentation et des performances obtenues.

Essais sur le problème de diffusion

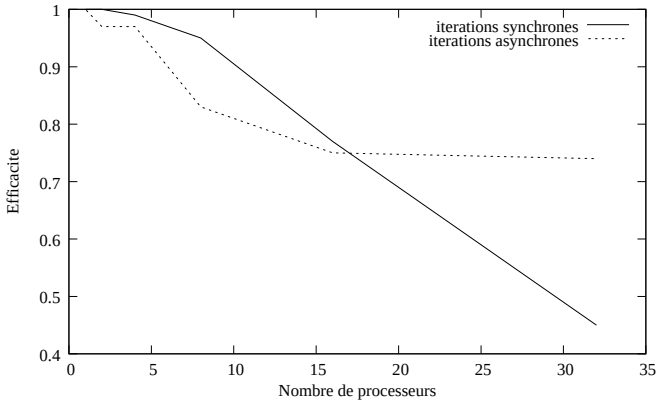
Une première version des solveurs a été implémentée en 2001 [93]. Elle a été testée sur le problème de Poisson tridimensionnel

$$\begin{cases} -\Delta u = f \text{ dans } [0\,,\,1]^3 \\ u_{/\partial[0\,,\,1]^3} = 0 \end{cases}.$$

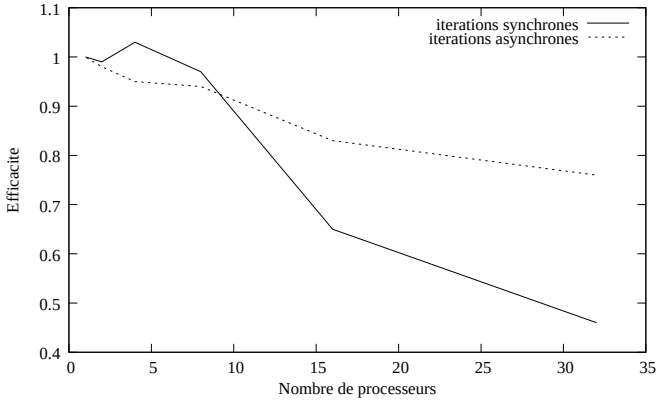
Les essais ont été menés sur l'IBM SP3 de l'IDRIS, qui est un ensemble de SMP comportant 16 processeurs Power 3 cadencés à 375 MHz. Cette génération de supercalculateur est antérieure au Power 4. Dans les solveurs, le découpage en sous-composantes est pris en compte dans l'implémentation des communications. La numérotation rouge-noir est également appliquée. Dans la version synchrone, toutes les requêtes d'envoi sont soumises après le traitement de chaque sous-domaine, contrairement à la version asynchrone où seuls les messages significatifs sont envoyés. Cette différence est due au problème d'interblocage illustré dans l'exemple 2.1. Le fait d'envoyer tous les messages, y compris ceux qui n'ont pas été affectés par une mise à jour, permet en l'occurrence d'éviter l'interblocage.

L'équation de Poisson a été discrétisée avec $216,000$, $512,000$ et $1,000,000$ de points dans le but d'évaluer la scalabilité du solveur dans des contextes différents. Les courbes d'efficacité obtenues (*cf.* figure A.1) montrent que la suppression des synchronisations n'a été profitable que lorsque suffisamment de processeurs sont utilisés, ce qui paraît logique ; cette remarque précise l'intérêt des itérations asynchrones.

Résolution avec un maillage de $216,000$ points



Résolution avec un maillage de $512,000$ points



Résolution avec un maillage de $1,000,000$ points

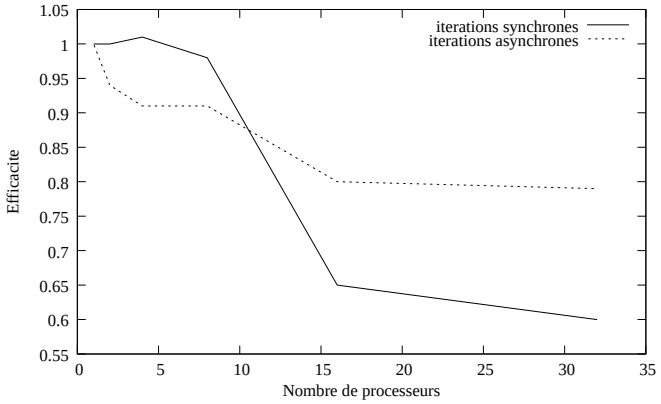


FIG. A.1 – Efficacité des algorithmes de Schwarz parallèles pour la résolution du problème Poisson sur le SP3

Avec peu de processeurs, les pénalités engendrées par la synchronisation sont plus faibles. L'algorithme synchrone offre des performances très satisfaisantes à condition d'utiliser peu de processeurs (≤ 8). À partir de 16 processeurs, nous assistons à un effondrement de l'efficacité de la parallélisation synchrone qui est essentiellement dû à l'implémentation des échanges de message. En effet, la faiblesse de cet algorithme réside dans le fait que les problèmes d'interblocage qui surviennent dans les échanges de message synchrones n'ont pas été résolus par une méthode performante et scalable en raison de leur difficulté.

Essais sur le problème de convection-diffusion

Pour la seconde version des solveurs, les essais ont été effectués sur IBM Power 4 p690 (une variante moins rapide du p690+) en 2002 [94]. Le problème test est l'équation de convection-diffusion (2.9). Les solveurs ont été testés avec 3 valeurs de ν : 0.01, 0.1 et 1. Le maillage comporte $1,728,000$ points et le domaine est découpé en 128 sous-domaines.

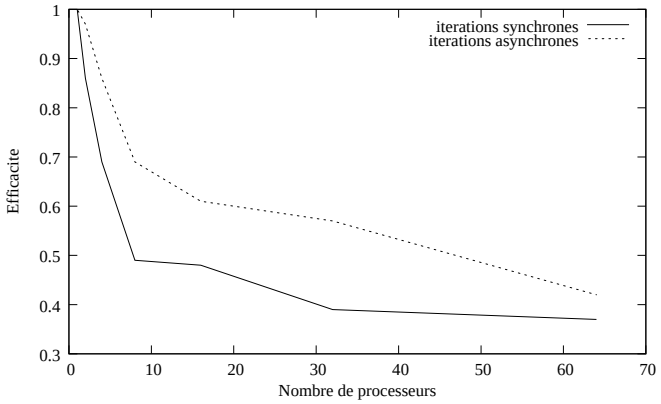
Les échanges de message prennent toujours en compte le découpage en sous-composantes. Les envois de message ont lieu après le traitement de chaque sous-domaine. Dans l'algorithme synchrone, la solution naïve consistant à envoyer tous le messages après le traitement d'un sous-domaine est encore appliquée. Les résultats des mesures de performance sont présentés dans la figure A.2. Ils montrent une amélioration nette des performances par rapport aux solveurs implémentés en 2001 (*cf.* figure A.1). En effet, la version asynchrone est maintenant toujours plus efficace que la version synchrone.

En comparant ces résultats avec ceux qui sont obtenus en 2004 (*cf.* figures 2.12, 2.13 et 2.14), nous constatons que la différence d'efficacité entre les versions synchrones et asynchrones est plus marquée dans l'implémentation datant de 2002. L'asynchronisme permet d'optimiser facilement les communications car on fait totalement abstraction des problèmes d'interblocage. Après le traitement d'un sous-domaine, on se contente en effet d'envoyer uniquement les parties qui ont été mises à jour. La quantité de données échangées par l'algorithme asynchrone est donc inférieure à celle de l'algorithme synchrone. Notons que cet écart a tendance à diminuer lorsque le nombre de processeurs est grand.

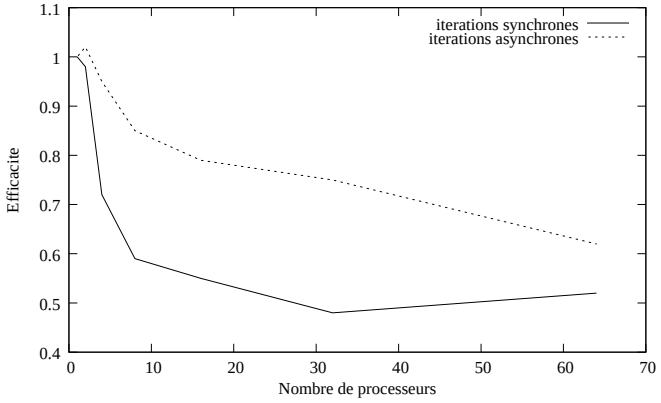
Bien que le solveur asynchrone donne des résultats satisfaisants, le code a évolué pour les raisons suivantes :

– En traitant plusieurs sous-domaines par processus, nous avons introduit deux niveaux de découpage. Le premier correspond à la répartition des données sur l'ensemble des processus. Le second sert à implémenter une stratégie de relaxation multiplicative dans le but d'accélérer la convergence. Nous avons voulu rendre la structure des messages indépendante du second niveau de découpage à l'instar de [52] dans l'optique de simplifier l'implémentation.

Résolution du problème avec convection dominante ($\nu = 0.01$)



Résolution du problème intermédiaire ($\nu = 0.1$)



Résolution du problème avec diffusion dominante ($\nu = 1$)

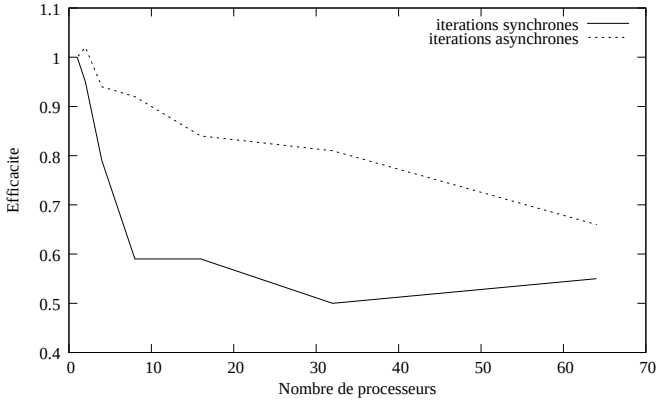


FIG. A.2 – Efficacité des algorithmes de Schwarz parallèles pour la résolution de problèmes de convection-diffusion sur le p690

– Afin d'évaluer objectivement les bénéfices de l'asynchronisme, il est nécessaire de résoudre le problème d'interblocage de l'algorithme synchrone qui se pose lorsque plusieurs sous-domaines sont pris en charge par les processus. En l'absence de solution générale valable quelque soit le découpage, nous avons opté pour un compromis consistant à implémenter les deux algorithme de manière quasi-identique (*cf.* section 2.3.4).

Annexe B

Paramètres des simulations de l'électrophorèse

Conditions appliquées
- Champ électrique : 950 $\mathrm{V.m^{-1}}$
- Vitesse moyenne du tampon vecteur : 0.0025 $\mathrm{m.s^{-1}}$

Chambre
- Rayon d'injection : 0.0015 m
- Longueur de la chambre : 0.3 m
- Épaisseur de la chambre : 0.005 m
- Largeur de la chambre : 0.1 m
- Coordonnées du centre du jet : $(x_1 = 0.05, x_3 = 0.0025)$

Propriétés du tampon vecteur
- Conductivité : 26.6 $\mu\mathrm{S.cm^{-1}}$
- Viscosité : 1.005×10^{-6} $\mathrm{m^2.s^{-1}}$
- Masse volumique : 1000 $\mathrm{kg}.m^{-3}$
- Permittivité : 6.9×10^{-10} $\mathrm{C^2.N^{-1}.m^{-2}}$

Soluté
- Rapport de conductivité par rapport au tampon vecteur : 0.1
- Mobilité électrophorétique : 5.26×10^{-8} $\mathrm{m^2.V^{-1}.s^{-1}}$
- Coefficient de diffusion : 2.5×10^{-9} $\mathrm{m^2.s^{-1}}$

Bibliographie

[1] J.L. Afonso. *Simulation et optimisation de l'electrophorèse à écoulement continu : couplage des phénomènes de transfert.* Thèse de Doctorat, Université Paul Sabatier de Toulouse, Laboratoire de Génie Chimique et Electrochimie (UPS Toulouse), 1998.

[2] J. Arnal, V. Migallón, and J. Penadés. Newton two stage paralle iterative methods for non linear systems. *BIT Numerical Mathematics*, 43 :849–861, 2003.

[3] O Axelsson. *Iterative solution methods.* Cambridge University Press, 1996.

[4] J. Bahi, E. Griepentrog, and J.C. Miellou. Parallel treatment of a class of differential-algebraic systems. *SIAM Journal of Numerical Analysis*, 23(5) :1969–1996, 1996.

[5] J. Bahi and J.C. Miellou. Contractive mappings with maximum norms. Comparison of constants of contraction and application to asynchronous iterations. *Parallel Computing*, 19 :511–523, 1993.

[6] J. Bahi, J.C. Miellou, and K. Rhofir. Asynchronous multisplitting methods for nonlinear fixed point problems. *Numerical Algorithms*, 15 :315–345, 1997.

[7] Z.Z. Bai and D.J. Evans. Matrix multisplitting methods with applications to linear complementary problems : Parallel asynchronous methods. *Int. J. Comput. Math.*, 79 :205–232, 2002.

[8] Z.Z Bai, D.R. Wang, and D.J. Evans. Models of asynchronous parallel nonlinear multisplitting relaxed iterations. *Journal of Computational Mathematics*, 13 :369–386, 1995.

[9] V. Barbu. *Non linear semi-groups and differential equations in Banach spaces.* Noordhoff international publishing, Gröningen, 1976.

[10] G.M. Baudet. Asynchronous iterative methods for multiprocessor. *J. Assoc. Comput. Mach.*, 2 :226–244, 1978.

[11] P. Benilan. *Equations d'évolution dans un espace de Banach quelconque et applications.* Thèse de Doctorat ès Sciences, Orsay, 1972.

[12] S. Benjelloun, P. Spitéri, and G. Authié. Parallel algorithms for solving the obstacle problem. *Computational Mechanics Publ., Springer-Verlag*, 2 :275–281, 1989.

[13] J. Bernussou, F. Le Gall, and G. Authié. About some iterative synchronous and asynchronous methods for Markov chain distribution computation. In *10-th I.F.A.C. World Congress*, 1987.

[14] D.P. Bertsekas and D. El Baz. Distributed asynchronous relaxation methods for convex network flow problems. *SIAM J. Control and Optimization*, 25 :74–85, 1987.

[15] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation, Numerical Methods*. Prentice Hall, Englewood Cliffs N.J., 1989.

[16] R. Bru, V. Migallón, J. Penadés, and D. Szyld. Parallel, synchronous and asynchronous two-stage multisplitting methods. *Electronic Transactions on Numerical Analysis*, 3 :24–38, 1995.

[17] J. Castel, V. Migallón, and J. Penadés. Convegence of non-stationary parallel multisplitting methods for hermitian positive definite matrices. *Mathematics of Computation*, 67(221) :209–220, 1998.

[18] E. Chajakis and S.A. Zenios. Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization. *Parallel Computing*, 17 :873–894, 1991.

[19] K.M. Chandy and L. Lamport. Distributed snapshots : determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1) :63–75, 1985.

[20] M. Charnay. *Itérations chaotiques sur un produit d'espaces métriques*. Thèse de Doctorat, Université Claude Bernard, Lyon, 1975.

[21] D. Chazan and W. Miranker. Chaotic relaxation. *Linear Algebra Appl.*, 2 :199–222, 1969.

[22] M.J. Clifton. Numerical simulation of protein separation by continuous-flow electrophoresis. *Electrophoresis*, 14 :1284–1291, 1993.

[23] M.J. Clifton, H. Roux-de-Balmann, and V. Sanchez. Electrohydrodynamic deformation of the sample stream in continuous-flow electrophoresis with an ac electric field. *The Canadian Journal of Chemical Engineering*, 70 :1055–1062, 1992.

[24] M.J. Clifton, H. Roux-de-Balmann, and V. Sanchez. Protein separation by continuous-flow electrophoresis in microgravity. *AIChE Journal*, 42(7) :2069–2079, 1996.

[25] P. Comte. Itérations chaotiques à retards. Étude de la convergence dans le cas d'un espace produit d'espaces vectoriellement normés. *CRAS série A*, 281 :863–866, 1975.

[26] P. Comte, J.C. Miellou, and P. Spitéri. La notion de H-accrétivité, applications. *CRAS série A*, 283 :655–658, 1976.

[27] R. Dautray and J.L. Lions. *Analyse mathématique et calcul numérique pour les sciences et les techniques*, volume 9. Masson, 1988.

[28] J.D.P. Donnely. Periodic chaotic relaxation. *Linear Algebra appl.*, 4 :117–128, 1971.

[29] D. El Baz. Asynchronous implementation of relaxation and gradient algorithms for convex network flow problems. *Parallel Computing*, 19 :1019–1028, 1993.

[30] D. El Baz. Parallel iterative algorithms for the solution of Markov systems. In *33-rd IEE Conference on Decision and Control*, pages 2524–2527, Orlando, U.S.A., 1994.

[31] D. El Baz. A method of terminating asynchronous iterative algorithms on message passing systems. *Parallel Algorithms and Applications*, 9 :153–158, 1996.

[32] D. El Baz. *Contribution à l'algorithmique parallèle. Le concept d'asynchronisme : étude théorique, mise en œuvre, et application.* Habilitation à Diriger des Recherches, Institut National Polytechnique de Toulouse, Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, 1998.

[33] D. El Baz, A. Frommer, and P. Spitéri. Asynchronous iterations with flexible communication : contracting operators. *Journal of Computational and Applied Mathematics*, 176 :91–103, 2005.

[34] D. El Baz, P. Spitéri, J.C. Miellou, and D. Gazen. Asynchronous iterative algorithms with flexible communication for non linear network flow problems. *Journal of Parallel and Distributed Computing*, 38 :1–15, 1996.

[35] M.N. El Tarazi. *Contraction et ordre partiel pour l'étude d'algorithmes synchrones et asynchrones en analyse numérique.* Thèse de Doctorat ès Sciences, Université de Besançon, 1981.

[36] M.N. El Tarazi. Some convergence results for asynchronous algorithms. *Num. Math.*, 39 :325–340, 1982.

[37] M.N. El Tarazi. Algorithmes mixtes asynchrones. Étude de la convergence monotone. *Num. Math.*, 44 :363–369, 1984.

[38] J.L. Estivalezes. *Calcul d'écoulements instationnaires internes et externes par un algorithme de pression implicite à pas séparés.* Thèse de Doctorat, Institut National Polytechnique de Toulouse, 1989.

[39] D.J. Evans and W. Deren. An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations. *Parallel Computing*, 17 :165–180, 1991.

[40] A .Frommer, H. Schwandt, and D. Szyld. Asynchronous weighted additive Schwarz methods. *Electronic Transactions on Numerical Analysis*, 5 :48–61, 1997.

[41] A. Frommer and P. Spiteri. On linear asynchronous iterations when the spectral radius of the modulus matrix is one. *Computing Suppl.*, 15 :91–104, 2001.

[42] A. Frommer and D. Szyld. On asynchronous two-stage iterative methods. *Numer. Math.*, 69 :141–153, 1994.

[43] A. Frommer and D. Szyld. Asynchronous iterations with flexible communication for linear systems. *Calculateurs Parallèles, Réseaux et Systèmes Répartis*, 10 :421–429, 1998.

[44] A. Frommer and D. Szyld. Weighted max norms, splittings and overlapping additive Schwarz iterations. *Numer. Math.*, 83 :259–278, 1999.

[45] A. Frommer and D. Szyld. On asynchronous iterations. *Journal of Computational and Applied Mathematics*, 123 :201–216, 2000.

[46] M.J. Gander. Optimized Schwarz methods. A paraitre dans SIAM Journal of Numerical Analysis.

[47] M. Garbey and D. Tromeur-Dervout. On some Aitken like acceleration of the Schwarz methods. *Int. J. for Numerical Methods in Fluids*, 40(12) :1493–1513, 2002.

[48] L. Giraud. *Implantations parallèles de méthodes de sous-domaines synchrones et asynchrones pour la résolution de problèmes aux limites.* Thèse de Doctorat, Institut National Polytechnique de Toulouse, Laboratoire d'Informatique et de Mathématiques Appliquées (ENSEEIHT-IRIT), 1991.

[49] L. Giraud and P. Spitéri. Résolution parallèle de problèmes aux limites non linéaires. *M2AN*, 25 :73–100, 1991.

[50] L. Giraud and P. Spitéri. Implementations of parallel solutions for nonlinear boundary value problems. In Evans, Joubert, and Liddel, editors, *Parallel Computing'91 Advances*, Parallel Computing, pages 203–211, Amsterdam, North-Holland, 1992.

[51] R. Guivarch. *Résolution parallèle de problèmes aux limites couplés par des méthodes de sous-domaines synchrones et asynchrones.* Thèse de Doctorat, Institut National Polytechnique de Toulouse, Laboratoire d'Informatique et de Mathématiques Appliquées (ENSEEIHT-IRIT), 1997.

[52] R. Guivarch, G. Padiou, and P. Papaix. Asynchronous schwarz alternating method in observation based distributed environment. *Réseaux et Systèmes Répartis - Calculateurs Parallèles*, 13(1) :35–45, 2001.

[53] R. Guivarch and P. Spitéri. Implantation de méthodes de sous-domaines asynchrones avec PVM et MPI sur IBM-SP2. *Calculateurs Parallèles*, 10(1) :431–438, 1998.

[54] R. Guivarch, P. Spitéri, H.C. Boisson, and J.C. Miellou. Schwarz alternating parallel algorithm applied to incompressible flow computation in vorticity stream function. *Parallel Algorithm and Application*, 11 :205–225, 1997.

[55] R. Hiromoto, B. R. Wienke, and R. G. Brickner. The performance of asynchronous iteration schemes applied to the linearized Boltzmann transport equation. *Parallel Computing*, 18(3) :241–268, 1992.

[56] K.H. Hoffman and J. Zou. Parallel efficiency of domain decomposition methods. *Parallel Computing*, 19 :1375–1391, 1993.

[57] R.I. Issa. Solution of the implicitely discretised fluid flow equations by operator splitting. *Journal of Computational Physics*, 62 :40–65, 1986.

[58] C. Jacquemard. *Contribution à l'étude d'algorithmes de relaxation à convergence monotone*. Thèse de Doctorat, Université de Besançon, 1977.

[59] M. Jarraya. *Mise en œuvre et étude de performance des algorithmes itératifs parallèles sur diverses architectures, application à l'optimisation et la commande*. Thèse de Doctorat, Université Paul Sabatier de Toulouse, Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, 2000.

[60] N. Jouve. *Modélisation et optimisation du procédé d'électrophorèse de zone à écoulement continu. Limites d'application au sol et en microgravité*. Thèse de Doctorat, Université Paul Sabatier de Toulouse, Laboratoire de Génie Chimique et Electrochimie (UPS Toulouse), 1991.

[61] J. Julliand, G.R. Perrin, and P. Spitéri. Simulations d'exécutions parallèles d'algoritmes numériques asynchrones. In *1ère conférence A.M.S.E.*, Lyon, 1981.

[62] J.L. Lions. *Quelques méthodes de résolution des problèmes aux limites non linéaires*. Dunod, 1969.

[63] Message Passing Interface Forum. *MPI : A Message-Passing Interface Standard*, 1994.

[64] Message Passing Interface Forum. *MPI-2 : Extensions to Message-Passing Interface*, 1997.

[65] J. C. Miellou, M. Laaraj, and M.J. Gander. Overlapping multi-subdomain asynchronous fixed point methods for elliptic boundary value problems. In *7th International Colloquium on Numerical Analysis and Computer Science with Applications*, Bulgarie, 1998.

[66] J.C. Miellou. *Opérateurs para-monotones*. Thèse de Doctorat ès Sciences, I.M.A.G., Université de Grenoble, 1970.

[67] J.C. Miellou. Algorithmes de relaxation chaotique à retards. *RAIRO*, 1 :55–82, 1975.

[68] J.C. Miellou. Itérations chaotiques à retards, étude de la convergence dans le cas d'espaces partiellement ordonnés. *CRAS Paris*, 280 :233–236, 1975.

[69] J.C. Miellou. Variantes synchrones et asynchrones de la méthode alternée de Schwarz. Technical Report E.R.A. de mathématiques n° 70654, Université de Besançon, 1982.

[70] J.C. Miellou. Asynchronous iterations in order intervals. In M. Cosnard et al., editor, *Parallel Algorithms*, pages 85–96. North-Holland, 1986.

[71] J.C. Miellou, D. El Baz, and P. Spitéri. A new class of iterative algorithms with order intervals. *Mathematics of Computation*, 67 :237–255, 1998.

[72] J.C. Miellou and P. Spiteri. Two criteria for the convergence of asynchronous iterations. In P. Chenin et al. ed., editor, *Computers and computing*, pages 91–95, Paris, 1985. Wiley-Masson.

[73] J.C. Miellou and P. Spitéri. Un critère de convergence pour des méthodes générales de point fixe. *M2AN*, 19(4) :645–669, 1985.

[74] D. Mitra. Asynchronous relaxations for the numerical solution of differential equations by parallel processors. *SIAM J. Sci. Stat. Comput.*, 8 :43–58, 1987.

[75] D.P. O'Leary and R.E. White. Multi-splittings of matrices and parallel solution of linear systems. *SIAM J. Alg. Disc. Meth.*, 6 :630–640, 1985.

[76] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970.

[77] S.V. Patankar. *Numerical heat transfer and fluid flow*. Mc Graw Hill, 1980.

[78] K. Revelli. *Étude des instabilités gravitaionnelles dans le procédé d'électrophorèse de zone à écoulement continu*. Thèse de Doctorat, Université Paul Sabatier de Toulouse, Laboratoire de Génie Chimique et Electrochimie (UPS Toulouse), 1995.

[79] W.C. Rheinboldt. On M-functions and their application to nonlinear Gauss-Seidel iterations and to network flows. *J. Math. Anal. and Appl.*, 32 :273–307, 1970.

[80] F. Robert. *Étude et utilisation de normes vectorielles en analyse numérique linéaire*. Thèse de Doctorat ès Sciences, Grenoble, 1968.

[81] F. Robert. Contraction en norme vectorielle : convergence d'itérations chaotiques. *Linear algebra and its applications*, 13 :19–35, 1975.

[82] F. Robert, M. Charnay, and F. Musy. Itérations chaotiques série-parallèle pour des équations non linéaires de point fixe. *Aplikate Mathematiky*, 20 :1–38, 1975.

[83] J.L. Rosenfeld. A case study on programming for parallel processors. Technical Report RC-64, I.B.M. Thomas J. Watson Research Center, U.S.A., 1967.

[84] P. K. Smolarkiewicz. A fully multidimensional positive definite advection transport algorithm with small implicit diffusion. *Journal of Computational Physics*, 54 :325–362, 1984.

[85] P. K. Smolarkiewicz and T. L. Clark. The multidimensional positive definite advection transport algorithm : further development and applications. *Journal of Computational Physics*, 67 :396–438, 1986.

[86] P. K. Smolarkiewicz and W. W. Grabowski. The multidimensional positive definite advection transport algorithm : nonoscillatory option. *Journal of Computational Physics*, 86 :355–375, 1990.

[87] P. Spitéri. Simulation d'éxécutions parallèles pour la résolution d'inéquations variationnelles stationnaires. *Revue E.D.F. Informatique et Mathématiques Appliquées, Série C*, 1 :149–158, 1983.

[88] P. Spitéri. *Contribution à l'étude de grands systèmes non linéaires*. Thèse de Doctorat ès Sciences, Faculté des Sciences et des Techniques de l'Université de Franche-Comté, 1984.

[89] P. Spitéri. Parallel asynchronous algorithms for solving boundary value problems. In M. Cosnard et al., editor, *Parallel Algorithms*, pages 73–84. North-Holland, 1986.

[90] P. Spitéri. A new characterization of M-matrices and H-matrices. *BIT Numerical Mathematics*, 43 :1019–1032, 2003.

[91] P. Spitéri and H.C. Boisson. Subdomain predictor-corrector algorithms for solving the incompressible Navier-Stokes equation. In H.G. Kaper and M. Garbey, editors, *Asymptotic and numerical methods for partial differential equations with critical parameters*, volume 384 of *NATO ASI series*, pages 335–347. Kluwer Academic Publishers, 1993.

[92] P. Spitéri and M. Chau. Parallel asynchronous Richardson method for the solution of the obstacle problem. In J.N Almhana and V.C. Bhavsars, editors, *HPCS 2002*, High Performance Computing Systems and Applications, pages 133–138, Moncton, 2002. IEEE.

[93] P. Spitéri, M. Chau, and R. Guivarch. Parallelization of subdomain methods with overlapping for the solution of 3D diffusion problem. International Conference on Numerical Algorithm, Marrakech, 2001.

[94] P. Spitéri, R. Guivarch, D. El Baz, and M. Chau. Parallelization of subdomain methods with overlapping for the linear and nonlinear convection-diffusion problems. In A. Clematis, editor, *PDP 2003*, 11th Euromicro Conference on Parallel and Distributed Network based Processing, pages 341–348, Gênes, 2003. IEEE.

[95] P. Spitéri, J.C. Miellou, and D. El Baz. Asynchronous Schwarz alternating method with flexible communication for the obstacle problem. *Réseaux et Systèmes Répartis*, 13(1) :47–66, 2001.

[96] P. Spitéri, J.C. Miellou, and D. El Baz. Parallel asynchronous Schwarz and multisplitting method for a non linear diffusion problem. *Numerical Algorithm*, 33 :461–474, 2003.

[97] J.C. Strikwerda. A probabilistic analysis of asynchronous iteration. *Linear Algebra and its Application*, 349 :125–154, 2002.

[98] D. Tromeur-Dervout and M. Garbey. Méthodes numériques hautes performances avec forte contrainte sur la localisation des données pour un métacomputing efficace. In *Canum 2002*, pages 173–176, 2002.

[99] A. Uresin and M. Dubois. Sufficient conditions for the convergence of asynchronous iterations. *Parallel Computing*, 10 :83–92, 1989.

[100] A. Uresin and M. Dubois. Parallel asynchronous algorithms for discrete data. *Journal of the association for computing machinery*, 37(3) :558–606, 1990.

[101] R. Varga. *Matrix Iterative Analysis*. Prentice Hall, Englewood Cliffs, 1962.

[102] D. Wang, Z.Z. Bai, and D.J. Evans. Asynchronous multisplitting relaxed iterations for weakly non linear systems. *Int. Jour. Computer Math.*, 54 :57–76, 1994.