N° d'ordre : XXX

HABILITATION
À DIRIGER DES RECHERCHES

présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

par

Ming CHAU

Algorithmes Parallèles Asynchrones et Applications en Calcul Scientifique

Soutenue le XXX XXX XXX devant le jury composé de :

M. XXX Rapporteur
M. XXX Rapporteur
M. XXX Rapporteur
M. XXX Examinateur
M. XXX Examinateur

Résumé

Abstract

Perfect heroes are cool,
but no one can really empathize or identify with them.
Masashi Kishimoto

Remerciements

Table des matières

Liste des tableaux

Table des figures

Introduction générale

L'informatique fournit des outils devenus 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 la voie 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. 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, car les temps d'attente induisent des temps d'inactivité des processeurs. Le problème soulevé devient d'autant plus vrai quand les processeurs sont séprées géographiquement par de longues distances.

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 (cette liste n'est pas exhaustive) ont permis d'établir des théorèmes de convergence dans le cadre non linéaire, avec des modèles de l'asynchronisme de plus en plus généraux, prenant en compte des situations très diverses.

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. Des expérimentation ont été menés sur Grid'5000, la plateforme expérimentale de Grid Computing de la France, dans le but de valider la pertinence de l'approche asynchrone lorsque les processurs sont répartis entre plusieurs villes à travers le pays. De plus, des essais sur cluster de GPU sur réseau local Infiniband ont pu être effectués. Les applications traitées durant ces travaux couvrent des domaines variés, tels que les mathématiques financières, la mécanique, l'imagerie médicale et l'électrophorèse. Par ailleurs, la mise en œuvre d'algorithmes synchrones a été étudiée dans le cadre d'un projet de simulation vasculaire. La contribution scientifique apportée dans le cadre de cette thèse d'habilitation est essentiellment expérimentale.

Structure du document

Le chapitre 1 rappelle quelques notions mathématiques permettant de formaliser la notion d'algorithme itératif asynchrone. Il permet de situer le contexte des algorithmes étudiés et employés au cours de nos travaux de recherche.

Le chapitre 2 effectue une synthèse des applications traitées avec des algorithmes itératifs asynchrones. Les équations et les méthodes de discrétisation sont décrites brièvement. Quelques résultats en cours de publication sont donnés.

Chapitre 1

Cadre théorique

1.1 Introduction

Nous présentons ici les bases d'un cadre mathématique pour les algorithmes itératifs parallèles dédiés à la résolution de systémes d'équations linéaires ou non-linéaires. Dans ce contexte, les algorithmes asynchrones sont des solveurs itératifs dans lesquelles les réactualisations du vecteur itéré sont effectuées sans synchronisation, ni ordre précis (*cf.* figure 1.1). De plus, des retards dans les échanges de données entre processeurs sont autorisés. Le calcul se poursuit avec les dernières valeurs disponibles. Ces algorithmes parallèles asynchrones présentent actuellement un intérêt avec le développement du parallélisme massif et des calculs répartis sur des sites distants (Grid Computing, Cloud Computing). À l'origine, ces méthodes itératives ont été introduites en 1969 par D. Chazan et W. Miranker [1] pour la résolution de systèmes linéaires. Cette étude a été généralisée en 1974 par J.C. Miellou [2] 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 [3]. 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.

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

- les techniques de contraction pour une norme vectorielle adaptée [4], employées par D. Chazan et W. Miranker, J.C. Miellou durant ses premiers travaux et G. Baudet ; ces techniques ont aussi été utilisées dans [5, 6, 7, 8],
- les techniques de contraction sur un espace produit muni de la norme uniforme avec poids, employées par M.N. El Tarazi en 1982 [9, 10],
- les techniques d'ordre partiel (convergence monotone dans un espace partiellement ordonné), employées par J.C. Miellou en 1975 [11] ; ces techniques ont également été utilisées dans [12, 9, 13],
- les techniques d'ensembles emboîtés, employées par D.P. Bertsekas et J.N. Tsitsiklis en 1989 [14],

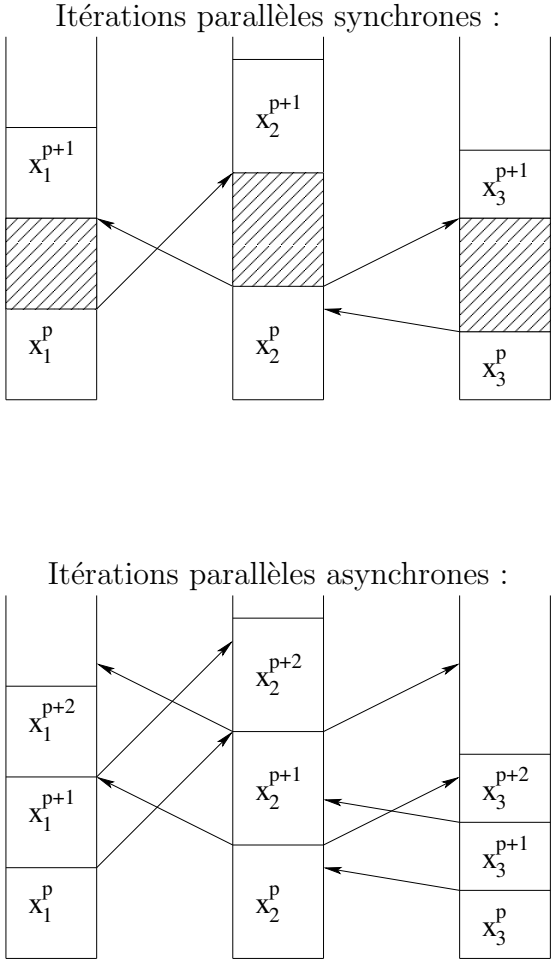


FIGURE 1.1 – Itérations parallèles synchrones et asynchrones

– les méthodes probabilistes employées par J.C. STRIKWERDA [15].

En 1998, J.C. MIELLOU, D. EL BAZ et P. SPITÉRI ont introduit les algorithmes itératifs parallèles asynchrones avec communication flexible [16]. Ce modèle présente la particularité d'intégrer étroitement les aspects communications aux aspects calculs. Les communications concernent les valeurs disponible à n'importe quel moment du calcul, et non à l'issue de chaque itération. 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 [17].

Dans le présent chapitre, nous présenterons dans la section 1.2 le modèle d'algorithmes asynchrones de J.C. MIELLOU [2] et nous rappellerons quelques résultats de convergence.

1.2 Un modèle de calcul asynchrone

1.2.1 Formulation des itérations asynchrones

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 [1, 2, 3] :

Définition 1.3 (Algorithme parallèle asynchrone)
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 construit récursivement une suite d'itérés $(X^p)_{p\in\mathbb{N}}$ de la manière suivante :

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

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 [18]. 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^*.

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

Remarque 1.2 Voir aussi G. Baudet [3] dans le cas de retards non bornés.

Il existe aussi un théorème de convergence utilisant la norme uniforme avec poids définie par :

$$\forall X \in E,\ \|X\|_\Gamma = \max_{1\leq i\leq \beta} \frac{|x_i|_i}{\gamma_i} \tag{1.15}$$

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 :

$$\exists \theta \in]0\,,\,1[,\ \forall W \in \mathcal{D}(F),\ \|F(W) - F(X^*)\|_\Gamma \leq \theta \|W - X^*\|_\Gamma \tag{1.16}$$

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

Nous renvoyons à [2] et [10] pour les démonstrations.

1.3 Application à la résolution de systèmes non linéaires

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

$$A.X + \phi(X) = B,\ X \in \mathbb{R}^n \tag{1.17}$$

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. Nous rappelons à présent quelques critères portant sur les coefficients de la matrice A et les propriétés de ϕ et qui permettent la convergence des algorithmes asynchrones.

Remarque 1.3 Le système (1.17) est en général issu de la discrétisation de problèmes aux limites de la forme :

$$\begin{cases} \Lambda(u) + \Lambda^d(u) \ni g,\ u \in E \\ \text{et conditions aux limites} \end{cases} \tag{1.18}$$

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 (1.18) est soumise à des contraintes. L'étude de la résolution du problème général (1.18) par des algorithmes parallèles asynchrones a été effectuée par J.C Miellou et P. Spitéri dans [19, 20]. Les notions de H-accrétivité [21, 22] et de m-accrétivité [23] y interviennent notamment.

Remarque 1.4 (La notion d'accrétivité) Les matrices diagonale dominantes en ligne et avec coefficients diagonaux positifs sont accrétives. La notion d'opérateur accrétif correspond à une généralisation de la notion d'opérateur monotone [23, 24]. Elle correspond en dimension finie à des critères classiques et simples à vérifier pour être danbs le cadre du théorème 1.1 [20, 25]. Soit A une matrice carrée de taille $n \times n$ et de coefficients $(a_{ij})_{1 \leq i,j \leq n}$. A est fortement accrétive pour la norme uniforme $(\|.\|_\infty)$ s'il existe $m \geq 0$ tel que :

$$a_{ii} \geq m \tag{1.19}$$

$$a_{ii} - \sum_{j \neq i} |a_{ij}| \geq m. \tag{1.20}$$

Nous renvoyons à [26] pour une caractérisation des M-matrices et des H-matrices par des matrices fortement accrétives dans les espaces ℓ_p, et [19] pour une caractérisations dans $\mathbb{R}^n$.

1.3.1 Critères pratiques

Afin d'envisager la résolution du système (1.17) à l'aide des algorithmes parallèles asynchrones (1.10), nous allons considérer sa décomposition par blocs dans $\mathbb{R}^n = \prod_{i=1}^{\beta} \mathbb{R}^{\nu(i)}$:

$$\forall i \in \{1, \ldots, \beta\},\ A_{ii}.x_i + \phi_i(x_i) + \sum_{j \neq i} A_{ij}.x_j = b_i. \tag{1.21}$$

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 (1.17) est construit implicitement comme suit :

$$\forall i \in s(p),\ x_i^{p+1} = F_i(\tilde{X}^p) \tag{1.22}$$

où x_i^{p+1} vérifie :

$$A_{ii}.x_i^{p+1} + \phi_i(x_i^{p+1}) + \sum_{j \neq i} A_{ij}.\tilde{x}_j^p = b_i. \tag{1.23}$$

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 1.1
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 (1.10) associés à la décomposition par blocs du système (1.17) convergent vers X^*, *solution de ce système.*

Ce résultat de convergence est démontré dans [19, 20, 25].

Remarque 1.5 Notons que si une multi-application est croissante, alors elle est accrétive (*cf.* [19, 25] pour la démonstration).

Remarque 1.6 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 [19, 20].

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 (1.17). Sa formulation et l'analyse de sa convergence est traitée dans [19, 20]. Dans ces références, il est en particulier démontré que si les hypothèses de la proposition 1.1 sont vérifiées pour une décomposition en β blocs, alors la convergence est assurée pour une décomposition plus grossière. En complément à la proposition 1.1, nous pouvons énoncer le corollaire suivant :

Corollaire 1.1 *Sous les hypothèses de la proposition 1.1, les algorithmes parallèles asynchrones (1.10) associés à la décomposition en sous-domaines du système (1.17) convergent vers* X^*, *la solution de ce système.*

Autrement dit, l'étude de la décomposition par points suffit pour établir la convergence pour toute autre décomposition. Le critère de convergence suivant [19, 20] découle de l'application du corollaire 1.1 :

Proposition 1.2
Si ϕ est un opérateur diagonal croissant et si A est une M-matrice, alors les algorithmes parallèles asynchrones (1.10) associés à la décomposition par blocs du système (1.17) convergent vers X^*, solution du système.

1.3.2 Lien avec la méthode de Schwarz

Formulation 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 :

$$\begin{gathered}\forall i \in \{1,\ldots,\beta\},\ \Omega_i \subset \Omega \\ \bigcup_{i=1}^{\beta} \Omega_i = \Omega.\end{gathered}$$

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

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

Soient $(\Gamma_i)_{1\leq i\leq\beta}$ les restrictions des frontières de Ω à celles des sous-domaines Ω_i tels que $\Gamma_i = \partial\Omega_i \cap \partial\Omega$. Quant aux frontières entre Ω_i et ses sous-domaines adjacents, elles sont notées : $\gamma_i^j = \partial\Omega_i \cap \Omega_j$, pour tout $j \in \text{adj}(i)$.

Nous considérons une équation aux dérivées partielles linéaire sur un domaine tridimensionnel ($\Omega \subset \mathbb{R}^3$) avec conditions aux limites :

$$\left\{\begin{array}{l} -\Lambda(u) = f \text{ dans } \Omega \\ B(u_{/\partial\Omega}) = 0 \end{array}\right. . \tag{1.24}$$

La méthode de Schwarz est un algorithme itératif. Chaque itération consiste à résoudre β sous-problèmes, de la même forme que le problème initial (1.24), 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 :

$$\forall p \in \mathbb{N},\ \forall i \in \{1,\ldots,\beta\},\ \left\{\begin{array}{l} -\Lambda(u_i^{p+1}) = f_i \text{ dans } \Omega_i \\ B(u_{i/\Gamma_i}^{p+1}) = 0 \\ u_{i/\gamma_i^j}^{p+1} = u_{j/\gamma_i^j}^{p},\ \text{ pour tout } j \in \text{adj}(i) \end{array}\right. . \tag{1.25}$$

La méthode de Schwarz est bien adaptée au parallélisme [27]. Chaque sous-problème associé à une itération de l'algorithme est résolu indépendamment des autres. Nous renvoyons à [28, 29, 30, 31, 32] pour des présentations détaillées de la méthode.

L'introduction du parallélisme asynchrone dans cet algorithme consiste à ré-écrire (1.25) en y incorporant une stratégie $\mathcal{S}$ et d'une suite de retards $\mathcal{R}$:

$$\forall p \in \mathbb{N}, \left\{ \begin{array}{l} i \in s(p) \Longrightarrow \left\{ \begin{array}{l} -\Lambda(u_i^{p+1}) = f_i \text{ dans } \Omega_i \\ B(u_{i/\Gamma_i}^{p+1}) = 0 \\ u_{i/\gamma_i^j}^{p+1} = u_j^{\rho_j(p)}, \text{ pour tout } j \in \text{adj}(i) \end{array} \right. \\ i \notin s(p) \Longrightarrow u_i^{p+1} = u_i^p \end{array} \right. . \tag{1.26}$$

Analyse de convergence en asynchrone

On peut à présent établir un lien entre la méthode de Schwarz asynchrone (1.26) et les modèles d'algorithmes parallèles asynchrones. La discrétisation des β sous-problèmes de l'algorithme de Schwarz classique (1.25) conduit à la résolution de systèmes algébriques couplés. D'après [33], si la matrice de discrétisation du problème initial (1.24) est une M-matrice, alors la matrice $\hat{A}$ résultant d'un procédé de transformation permettrant de prendre en compte les recouvrements entre sous-domaines est aussi une M-matrice. Le système augmenté associé à (1.25) s'écrit :

$$\hat{A}.\hat{X} = \hat{B}. \tag{1.27}$$

De plus, si A est perturbé par un opérateur diagonal croissant, ce procédé permet d'aboutir à un système algébrique augmenté où intervient une M-fonction, somme d'une M-matrice et d'une application diagonale croissante.

Ce procédé permet de reformuler la méthode de Schwarz (1.25) comme un algorithme par blocs :

$$\forall p \in \mathbb{N}, \forall i \in s(p), \ \hat{A}_{ii}.\hat{x}_i^{p+1} = \hat{b}_i - \sum_{j \neq i} \hat{A}_{ij}.\hat{x}_j^p \tag{1.28}$$

où le traitement des β sous-domaines est parallèle asynchrone.

Chapitre 2

Applications en calcul scientifique

2.1 Introduction

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 [19, 34, 35, 33, 36, 25, 37, 38, 39, 40, 41, 42, 43, 44, 45],
- 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 [46, 47, 48, 49],
- la résolution de grands systèmes linéaires creux dans le domaine de la modélisation de grands systèmes à l'aide de chaînes de Markov [50, 51, 52],
- la résolution de grands systèmes linéaires par les méthodes de *multisplitting* (introduites par D.P. O'Leary et R.E. White [53]) [54, 55, 56, 57, 58, 59, 60, 61, 42].

Ces méthodes asynchrones peuvent aussi s'appliquer à la résolution de problèmes algébro-différentiels [62, 63], à la programmation dynamique [64, 65], à la résolution de l'équation de Boltzmann [66], 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 [67, 68].

Les sections suivantes contiennent une synthèse, puis récapitulent les formulations des problèmes numériques traités.

2.2 Synthèse des travaux

Nous présentons dans ce chapitre les applications qui ont été étudiés dans le cadre du travail de recherche effectué en algorithmes parallèles asynchrones. Ces applications concernent :

- le traitement d'images echographiques 2D [69] et PET 3D+t [70],
- les mathématiques financières [71, 72, 73, 74],
- l'électrophorèse à écoulement continu [75, 76, 77, 78, 79],
- la résolution de problèmes de convection-diffusion et de diffusion non-linéaires [80, 81, 45, 82, 83, 84],
- le G-mouvement Brownien et les milieux poreux en collaboration avec l'université de Annaba (ces travaux n'ont pas encore été publiés) ; les expérimentations ont été effectués sur le calculateur HPC@LR à Montpellier.

Les expérimentations sur Grid'5000 [85] ont montré que l'asynchronisme résout facilement le problème des déséquilibrages de charge, au sens architecture et réseaux, sans faire appel à des techniques de migration de charge [86], dans le contexte de machines hétérogènes, géographiquement répartis, et dont les propriétés varient dynamiquement. La structure de ce chapitre est décrite ci-dessous.

Dans l'axe de recherche des applications en traitement d'images, l'étude a concerné deux variantes du filtrage par diffusion anisotrope de P. Perona et J. Malik [87, 88]. Au cours de ces travaux, nous nous sommes essentiellement intéressé à la résolution de problèmes de diffusion anisotrope non stationnaires, 2D et 3D (analyse numérique) et au calcul des coefficients de diffusion (imagerie et statistiques). Peu de résultats concernent les comparaisons de performance entre algorithmes synchrones et asynchrones. Nous renvoyons à [89, 69, 70] pour les aspects techniques concernant les travaux effectués. Des méthodes de résolution variées ont été étudiées :

- discrétisations temporelles explicites et semi-implicites (Euler, Gear),
- conditions de stabilité dans le cas explicite,
- paramètres et conditions de convergence (nombre de conditionnement, rayon spectral de la matrice de Jacobi, paramètre de relaxation optimal, propriété de M-matrice) pour plusieurs solveurs, dans les cas semi-implicites : Gradient Conjugué, S.O.R., Jacobi, Gauss-Seidel, algorithmes parallèles synchrones et asynchrones.

Plusieurs faits intéressants sur le plan numérique ont été démontrés dans le cadre du filtrage par diffusion anisotrope. Les nombres de conditionnement des matrices de discrétisation sont proches de 1, ce qui permet de ne pas préconditionner le Gradient Conjugué. De plus ce sont des M-matrices, assurant la convergence des algorithmes asynchrones. Les rayons spectraux des matrices de Jacobi associées sont significativement inférieurs à 1, ce qui permet de prédire une convergence rapide pour les méthodes de relaxation. Ces bonnes propriétés sont les conséquences d'une convention souvent adoptée en traitement d'images [87, 88] : les pas de discrétisations ne sont pas les dimensions des pixels ou des voxels, mais ce sont les distances relatives

par rapport à la longueur la plus courte.

Dans les axes de recherche des mathématiques financières et des équations de convection-diffusion et de diffusion non-linéaires basées sur la projection des itérés sur un convexe fermé, des essais de solveurs asynchrones ont aussi été effectués dans le cadre de Grid5000. Les études ont essentiellement porté sur le probléme d'obstacle non stationnaire avec ou sans convection [81, 45, 82, 71, 72, 73, 83], et sur des problèmes similaires où la contrainte (l'obstacle) s'exerce sur les conditions aux limites [84]. Les essais sur Grid5000 ont donné des temps de calculs favorables pour les algorithmes asynchrones. Des essais ont été effectués avec jusqu'à 134 millions d'inconnues, sur plusieurs centaines de cores CPU. De meilleures performances avec sites distants et algorithmes asynchrones ont été observées. Par ailleurs, les solveurs parallèles du problème d'obstacle ont été portés sur cluster de GPU (connectés en réseau local Infiniband 20 Gb/s) par Lilia Ziane Khodja [74]. Le parallélisme asynchrone intervient notamment pour la communication entre nœuds GPU. Les bonnes performances obtenues par les algorithmes asynchrones permettent d'envisager plus tard l'exploitation des clusters de GPU géographiquement répartis.

Remarque 2.1 Notons que dans la publication [80], seules des expérimentations pour problèmes de convection-diffusion linéaires sont effectués. Nous renvoyons dès à présent à la publication.

Dans l'axe de recherche de l'électrophorèse, des résultats ont été obtenus d'abord sur calculateur parallèles CPU (desktop et SMP à l'IDRIS) [75, 76]. Ces travaux valident les meilleures performances des algorithmes asynchrones, et un certain degré de justesse des simulations obtenues. Des essais similaires ont été par la suite effectué sur calculateurs CPU répartis sur toute la France (Grid5000). Les travaux [77, 78, 79] ont mis en évidence l'intérêt des algorithmes asynchrones dans l'utilisation de solveurs parallèles sur calculateurs répartis. Les algorithmes asynchrones ont obtenu de meilleurs temps de calcul que leurs équivalents synchrones, et d'autant meilleurs quand la résolution parallèle d'un système est répartie sur des sites géographiquement distants.

Remarque 2.2 Deux libraires de communications ont été utilisées et testées sur Grid5000 : MPI et P2PDC [90]. Notons que l'utilisation de P2PDC concerne uniquement les travaux effectués dans le cadre du projet ANR-CIP [81, 71, 72, 78].

Dans la collaboration avec l'université d'Annaba, les problèmes aux limites traités sont des équations de diffusion non-linéaires qui nécessitent la mise en œuvre de techniques itératives de linéarisation. Ce n'est pas le cas pour les problèmes non-linéaires liés au problème de l'obstacle où les vecteurs itérés du solveur de système d'équations sont projetés sur un convexe fermé. Ces collaborations sont effectuées dans le cadre de la thèse de doctorat de deux étudiants de l'université d'Annaba.

Dans la suite du chapitre, étant donné que les équations continues et discrètes sont séparées en sous-sections distinctes, nous écrirons de la même manière les

fonctions continues et discrétisées telles que u, v, p, etc. Ceci évite d'introduire systématiquement des notations supplémentataires en majuscules.

2.3 Imagerie médicale

Le filtre de diffusion anisotrope a pour objectifs le débruitage des images et la préservation des contours. Le débruitage est effectué via le processus de diffusion de l'équation de la chaleur (équivalent au flou Gaussien). L'anisotropie de la diffusion assure alors la préservation des motifs ayant des contours nets. Ce type de traitement d'image permet notamment de lisser une image avant l'utilisation d'un détecteur de contour tel que Level Set [91]. Les travaux effectués [69, 70] concernent l'application du filtre de diffusion anisotrope en imagerie échographique et en tomographie par émission de positrons (PET) ; voir figure 2.1.

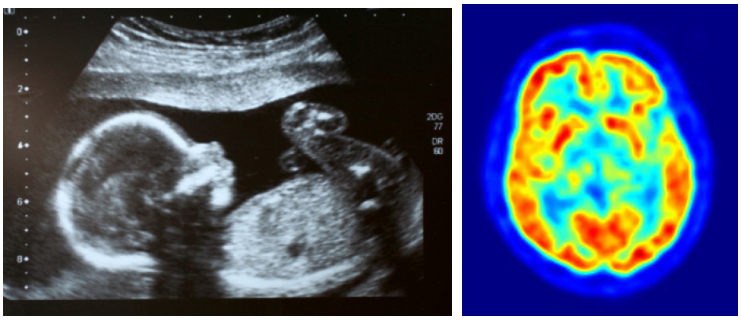


FIGURE 2.1 – Image échographique (gauche) et PET (droite)

L'imagerie échographique est basée sur la propagation et la réflexion des ultra-sons dans le but de visualiser les organes dans un corps vivant, ainsi que sur l'effet Doppler dans le but de mesurer des flux sanguins. La formation des images échographique est relativement simple car elle repose sur la mesure de temps entre émission d'un son et réception de l'écho. Le *speckle* est engendré par la propagation des ultra-sons dans des milieux où les irrégularités sont aux mêmes échelles que les longueurs d'ondes des ultra-sons. C'est le cas des organes du coprs. L'imagerie PET est basée sur la détection de l'activité radioactive dans un corps vivant, suite à l'injection d'une substance radioactive (radiotraceur). Cette technique permet de localiser et mesurer certains processus physiologiques, comme par exemple l'activité cérébrale. La formation des images PET est plus complexe car elle repose sur des algorithmes rétroprojection couplés à des traitements statistiques. Contrairement à l'échographie, le bruit en imagerie TEP est essentiellement d'origine statistique.

2.3.1 Equations

Nous dévrivons ici de manière succinte les modèles de diffusion anisotrope qui interviennent, de manière à établir rapidement le lien avec le calcul numérique. Nous renvoyons aux articles [89, 69, 70] pour les modèles détaillés.

Filtre pour images échographiques Nous présentons ici un modèle 2D. Soient $\Omega \subset \mathbb{R}^2$ le domaine, u^0 l'image initiale, et u un fonction de l'espace et du temps qui représente l'image subissant le filtrage. Le filtre de diffusion anisotrope s'écrit :

$$\begin{cases} \frac{\partial u}{\partial t} - \mathrm{div}(f(u)\vec{\mathrm{grad}}(u)) = 0 \text{ dans } \Omega \\ u(t=0) = u^0 \\ \frac{\partial u}{\partial n} = 0 \text{ dans } \partial\Omega. \end{cases} \tag{2.1}$$

Le coefficient de diffusion $f(u)$ est une fonction positive et bornée sur Ω qui dépend de l'image elle-même en cours de traitement. Il est calculé à partir d'une analyse statistique des différences d'intensité lumineuse entre chaque paire de pixels voisins. Il est donc naturel par la suite de discrétiser le coefficient de diffusion en posant ses points de discrétisation sur les bords des pixels, et non au centre.

Filtre pour images PET L'imagerie PET est de type 3D. De plus, l'activité radioactive est observée à plusieurs instants, ce qui engendre des séries d'images 3D. Le but du modèle de Clovis Tauber [70] est de traiter toutes les images avec le même coeficient de diffusion. Ainsi, les informations apportées par chaque image d'une série profitent à toute la série. Soient $\Omega \subset \mathbb{R}^3$ le domaine, $(u_k^0)_{1\leq k\leq m}$ la série initiale de m images, et $(u_k)_{1\leq k\leq m}$ les fonction de l'espace et du temps qui représente les images subissant le filtrage. Le filtre de diffusion anisotrope s'écrit :

$$\begin{cases} \frac{\partial u_k}{\partial t} - \mathrm{div}(f(u_1,\ldots,u_m)\vec{\mathrm{grad}}(u_k)) = 0 \text{ dans } \Omega \\ u_k(t=0) = u_k^0 \\ \frac{\partial u_k}{\partial n} = 0 \text{ dans } \partial\Omega. \end{cases} \tag{2.2}$$

pour $k = 1, \ldots, m$. Notons qu'il ne faut pas confondre le temps du processus de diffusion (variable t) avec le temps de l'acquisition des images PET (indice k). Dans ce modèle, $f(u_1, \ldots, u_m)$ est calculé à partir de toutes les images. Ainsi, il faut résoudre m équations non stationnaires couplées. Au cours du calcul de chaque pas de temps du processus de diffusion, les m équations deviennent indépendantes quand le coefficient de diffusion commun est calculé. Ce sont donc m algorithmes parallèles qui calculent simultanément (2 niveaux de parallélisme).

2.3.2 Discrétisation

La méthode de discrétisation est commune aux deux modèles. Le coefficient de diffusion est discrétisé sur les facettes des voxels (ou bords des pixels). Les intensités

lumineuses sont discrétisées au centre des voxels (ou pixels). Cela simplifie le schéma de discrétisation des dérivées secondes. Voici un exemple en 1D :

$$\frac{\partial}{\partial x}(f\frac{\partial u}{\partial x})(x_i) = \frac{f(x_{i+\frac{1}{2}})(u(x_{i+1}) - u(x_i)) - f(x_{i-\frac{1}{2}})(u(x_i) - u(x_{i-1}))}{h^2} \tag{2.3}$$

avec $h = 1$ (dimension relative). Ce procédé est similaire à la méthode des volumles finis [92]. Nous noterons $A(u)$ la matrice associée à l'opérateur $-\text{div}(f(u)\vec{\text{grad}}(u))$. Dans les cas 2D et 3D, si l_x, l_y et l_z sont les dimensions des voxels (ou pixels), alors les pas de discrétisation valent : $h_x = \frac{l_x}{l_x}$, $h_y = \frac{l_y}{l_x}$, et $h_z = \frac{l_z}{l_x}$.

Le schéma temporel explicite d'Euler s'écrit :

$$u^{p+1} = (I_d - \delta_t A(u^p)).u^p. \tag{2.4}$$

Les schémas de discrétisation temporelle implicites sont en fait semi-implicite : les valeurs de u utilisées pour calculer $f(u)$ sont celles du pas de temps précédent. Ils sont inconditionnellement stables. Les schémas d'Euler et Gear s'écrivent :

$$(\theta I_d + \delta_t A(u^p)).u^{p+1} = u^p + (1-\theta)u^{p-1}, \tag{2.5}$$

avec $\theta = 1$ pour Euler et $\theta = \frac{3}{2}$ pour Gear. Les méthodes de discrétisation conduisent à des M-matrices dans les deux cas semi-implicites.

2.3.3 Résultats

Voici quelques résultats en cours de publication prsentés sous forme synthétique. Ils concernent en particuler les aspects numériques du filtrage des images PET. Sur la plan expérimental, les calculs parallèles ont été effectués au CINES (serveur `jade.cines.fr`, processeurs Intel Xeon Harpertown/Nehalem, réseau Infiniband 40 Gb/s). Le code est écrit en C et MPI. Les dimensions des voxels des images PET utilisées sont : $l_x = l_y = 2\,mm$, et $l_z = 2.8\,mm$. Les pas de discrétisation spatiaux sont donc $h_x = h_y = 1$ et $h_z \geq 1$.

– Condition de stabilité pour le schéma explicite :

$$\delta_t < \frac{1.0}{2(2 + \frac{1.0}{h_z^2})}. \tag{2.6}$$

Compte-tenu de $h_z \geq 1$, la stabilité est assurrée pour des valeurs de δ_t relativement grandes.

– Rayon spectral de la matrice de Jacobi $\mathcal{J}^p$ associée à $(\theta I_d + \delta_t A(u^p))$:

$$\rho(\mathcal{J}^p) \leq \frac{2\delta_t(2 + \frac{1}{h_z^2})}{\theta + 2\delta_t(2 + \frac{1}{h_z^2})} < 1. \tag{2.7}$$

– Nombre de conditionnement à chaque pas de temps p :

$$\kappa(p) \leq 1 + \frac{4\delta_t}{\theta}(2 + \frac{1}{h_z^2}). \tag{2.8}$$

Des calculs ont été effectués sur une série de 20 images 3D ayant chacune $262 \times 262 \times 64$ voxels (plus de 4 millions de points de discrétisation par équation). Les 20 images sont traiées avec 2 niveaux de parallélisme : les 20 équations sont résolues simultanément sur tous les processeurs, chacune des équation est résolue en parallèle (décomposition de domaine). Les résultats sont présentés dans les tableaux 2.1 et 2.3.

schema - algorithme	nb. core	temps (sec.)	nb. relaxations
explicite	1	185	-
explicite	24	80	-
explicite	96	53	-
Euler	1	2103	8915
Euler - synchrone	24	249	8969
Euler - synchrone	96	93	8962
Euler - asynchrone	24	250	10333
Euler - asynchrone	96	129	17574

TABLE 2.1 – Temps de calcul et nombres de relaxations pour le filtrage des images TEP sur 34 pas de temps ($\delta_t = 0.15$).

schema - algorithme	nb. core	accélŕation	efficacité
explicite	24	2.31	0.09
explicite	96	3.49	0.03
Euler - synchrone	24	8.44	0.35
Euler - synchrone	96	22.61	0.23
Euler - asynchrone	24	8.41	0.35
Euler - asynchrone	96	16.30	0.16

TABLE 2.2 – Accélérations et efficacités pour le filtrage des images TEP sur 34 pas de temps ($\delta_t = 0.15$).

D'apès les mesures de performance (tableaux 2.1, 2.2 et 2.3), le calcul explicite ne profite pas du parallélisme car le calcul du coefficient de diffusion ($\mathcal{O}(nlog(n))$) est purement séquentiel et relativement lourd par rapport à un produit entre matrice creuse 7 bandes et vecteur ($\mathcal{O}(n)$). Toutefois, le calcul explicite reste le plus rapide. Les calculs semi-implicites se parallélisent bien. Une centaine de processeurs

schema - algorithme	nb. core	temps (sec.)	nb. relaxations
explicite	24	2438	-
explicite	96	1235	-
Euler - synchrone	24	5726	54968
Euler - synchrone	96	2063	55705
Euler - asynchrone	24	5846	65800
Euler - asynchrone	96	3396	132886

TABLE 2.3 – Temps de calcul et nombres de relaxations pour le filtrage des images TEP sur 500 pas temps ($\delta_t = 0.0083$).

est nécessaire pour rendre ces méthodes attractives par rapport au schéma explicite. La comparaison entre synchrone et asynchrone est en faveur de l'algorithme synchrone. Pour 24 core, l'avantage de l'algorithme synchrone n'est pas surprenant. Pour 96 core, l'algorithme asynchrone effectue trop de relaxations supplémentaires. L'origine de ce comportement est encore indéterminé, mais d'autres expérimentations tendent à suggérer que ces nombres très élevés de relaxations supplémentaires sont problème-dépendants. Ces expérimentations montrent que l'importance des relaxations supplémentaires de l'algorithme asynchrone ne doit pas être sous-estimé.

2.4 Problèmes non-linéaires avec projection

Durant ces dernières années, les applications du calcul en mathématiques financières ont connu un grand développement. Le pricing des options Européennes et Américaines modélisées par les équations classiques de BLACK et SCHOLES ont reçu un grand intérêt [93]. Rapellons que ce sont des équations de convection-diffusion non stationnaires classiques (options Européennes) ou bien des inéquations variationnelles (options Américaines) définies à partir des équations classiques et de la projection sur un sous-ensemble convexe fermé de l'espace des solutions recherchées. Le domaine de l'espace 3D est non borné. Un artifice que nous ne traitons pas ici consiste à agrandir un domaine borné au fur et à mesure du calcul [94]. La parallélisation de tels calculs est donc une nécessité.

Le problème d'obstacle est l'équation de base (cas stationnaire) de ces calculs en finance, qu'on retrouve aussi dans la mécanique, dans les problèmes à frontère libre. Elle est liée à l'équation de Hamilton-Jacobi-Bellman (voir [95] pour les formulations équivalentes). Il a fait l'objet de nombreuses publications dans les domaines du calcul et des mathématiques appliquées :

- études théoriques sur des algorithmes séquentiels [96, 97],
- étude du taux de convergence [98, 99, 100, 101],
- étude avec des algorithmes par blocs ou la méthode de Schwarz [102, 103],

– étude avec algorithmes parallèles asynchrones [104].
– étude avec algorithmes asynchrones sur cluster de GPU [74] avec une analyse reposant sur la notion de sous-différentiel (voir [24]), permettant de se ramener de façon rapide aux résultats présentés au chapitre 1, en particulier à l'équation (1.17).

Les méthodes itératives asynchrone sont bien adaptées pour ce type de problème mathématique grâce à la possibilité de projeter chaque itéré sur un convexe fermé durant la résolution du problème d'obstacle. Ceci est valable pour des méthodes de relaxation par point (Richardson projeté) ou par blocs (Gauss-Seidel par blocs avec une factorisation de matrice 3 bandes et projection de chaque bloc itéré). Nous renvoyons aux publications [71, 72, 73, 74, 81, 45, 82, 83, 84] pour les expérimentations numériques.

2.4.1 Equations

Les options Américaines Les équations de Black et Scholes permettent de déterminier un prix v pour une action. Elles sont rétrogrades dans le temps : on évalue l'état initial à partir d'un état final donné. Elles peuvent s'écrire :

$$\begin{cases} \frac{\partial v}{\partial t} + (r - \frac{\sigma^2}{2})\nabla v + \frac{\sigma^2}{2}\Delta v - rv \geq 0,\ v \geq \phi,\ \text{dans } [0,T] \times \mathbb{R}^n \\ (\frac{\partial v}{\partial t} + (r - \frac{\sigma^2}{2})\nabla v + \frac{\sigma^2}{2}\Delta v - rv)(v - \phi) = 0,\ \text{dans } [0,T] \times \mathbb{R}^n \\ \text{Conditions limites et finales,} \end{cases} \tag{2.9}$$

où $\phi = \phi(S) = \max(S - K, 0)$ dans le cas du *call*, et $\phi = \max(K - S, 0)$ dans le cas du *put*. Les notations sont définies dans le tableau 2.4.

r	Taux d'intérêt
σ	Volatilité (variabilité)
K	Prix d'exercice
S	Valeur de l'option
T	Temps final

Table 2.4 – Variables et paramètres du modèle des options Américaines

Le problème d'obstacle Le traitement du schéma de discrétisation temporelle conduit au problème d'obstacle suivant :

$$\begin{cases} -(r - \frac{\sigma^2}{2})\nabla v - \frac{\sigma^2}{2}\Delta v + (r + \frac{1}{\delta_t})v - g \geq 0,\ v \geq \phi,\ \text{dans } \mathbb{R}^n \\ (-(r - \frac{\sigma^2}{2})\nabla v - \frac{\sigma^2}{2}\Delta v + (r + \frac{1}{\delta_t})v - g)(v - \phi) = 0,\ \text{dans } \mathbb{R}^n \\ \text{Conditions limites et initiales,} \end{cases} \tag{2.10}$$

où $g = \frac{1}{\delta_t} v^p$ et v^p est la valeur de v au temps précédent, δ_t est le pas de temps. Puis, par un changement de variable classique (voir [73] pour les détails), on obtient un problème d'obstacle dans lequel un opérateur autoadjoint intervient :

$$-\nu\Delta u + (\frac{\|b\|_2^2}{4\nu} + c)u = e^{-a}.g = f, \tag{2.11}$$

où a et b, ν, c, $u = e^{-a}v$ sont déduits de l'équation (2.10). Un opérateur autoadjoint est nécessaire pour utiliser l'algorithme de Richardson projeté. Dans le cas du GPU, pour des raisons liées à cette architecture et aux algorithmes, la méthode par points de Richardson calcule plus rapidement qu'une méthode par blocs (valable pour le cas non autoadjoint) qui converge pourtant avec moins d'itérations [74], d'où l'intérêt du changement de variable.

Le probème d'obstacle frontière Ce problème a été traité dans [84]. Dans cette variante de problème d'obstacle, la projection a lieu au niveau des frontières. Soient $\Omega \subset \mathbb{R}^n$, ψ une fonction sur Ω. La frontière de Ω se décompose en deux parties : $\partial\Omega = \Gamma_0 \cup \Gamma_1$. L'équation du problème d'obstacle frontière s'écrit comme un problème linéaire classique dans lequel une non linéarité est introduite dans les conditions aux limites. On cherche alors à déterminer u tel que :

$$\begin{cases} -\nu\Delta u + cu = f \text{ dans } \Omega \\ u = \psi \text{ dans } \Gamma_0 \\ \text{si } u > \psi \text{ alors } \frac{\partial u}{\partial n} = 0 \text{ dans } \Gamma_1 \\ \text{si } u \leq \psi \text{ alors } u = \phi \text{ et } \frac{\partial u}{\partial n} \geq 0 \text{ dans } \Gamma_1, \end{cases} \tag{2.12}$$

où c, ν sont des constantes positives, et f est un second membre donné. Cela revient à choisir le type de condition aux limites en fonction de la valeur de la solution. L'ensemble des fonctions qui vérifient les conditions imposées sur la frontière Γ_1 est un convexe fermé qui s'écrit :

$$K = \{u | \text{fonction sur } \Omega \text{ tels que } u = \psi \text{ sur } \Gamma_0 \text{ et } u \geq \psi \text{ sur } \Gamma_1\}. \tag{2.13}$$

2.4.2 Discrétisation

Options Américaines et problème d'obstacle Le schéma d'évolution en temps utilisé dans cette étude est un schéma d'Euler qui conduit à écrire le problème d'obstacle discret suivant :

$$\begin{cases} (\lambda I_d + \alpha A).u^{p+1} - u^p \geq 0, u^{p+1} \geq \phi \\ ((\lambda I_d + \alpha A).u^{p+1} - u^p)^t (u^{p+1} - \Phi) = 0, \end{cases} \tag{2.14}$$

où A est une M-matrice et u^{p+1} est obtenu en résolvant un problème d'obstacle discrétisé, λ et α sont des réels positifs ($(\lambda I_d + \alpha A)$) est aussi une M-matrice). Les théorèmes de convergence 1.1 et 1.2 du chapitre 1 s'appliquent. Nous renvoyons à [73] pour une description des discrétisations, ainsi que pour une étude de convergence des algorithmes asynchrones appliqués à la résolution de ce problème.

Problème d'obstacle frontière La discrétisation de ce problème ne pose aucun problème en ce qui concerne l'équation aux dérivées partielles à l'intérieur de Ω ou sur la frontière Γ_0. Le traitement de la frontière Γ_1 consiste à introduire des instructions conditionnelles dans l'expression des coefficients de matrice et du second membre. Soit i_1, l'indice d'une composante u_{i_1} située sur Γ_1, et soit j_1, l'indice de la composante de u nécessaire pour calculer la condition de Neumann à la ligne i_1 du système. Les coefficients de matrice A_{i_1,i_1} et A_{i_1,j_1} sont :

$$A_{i_1,i_1} = 1\,;\; A_{i_1,j_1} = (\textbf{si } u_{i_1} \leq \psi_{i_1} \textbf{ alors } 0 \textbf{ sinon } -1), \tag{2.15}$$

et le coefficient b_{i_1} du second membre est :

$$b_{i_1} = (\textbf{si } u_{i_1} \leq \psi_{i_1} \textbf{ alors } \psi_{i_1} \textbf{ sinon } 0). \tag{2.16}$$

Ceci se représente bien avec la notion de projection ou encore avec la notion de sous-différentiel [24] (contexte des fonctions multivaluées). Des résultats [20, 25, 16] montrent que si la matrice de discrétisation de (2.12) est une M-matrice, alors le traitement de la frontière Γ_1 conduit à une M-fonction multivaluée. On revient au cadre du chapitre 1, équation (1.17).

2.5 Electrophorèse

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 $\vec{E}$. La séparation par électrophorèse est une technique d'analyse incontournable dans des domaines tels que la biologie moléculaire et la génétique (figure 2.2).

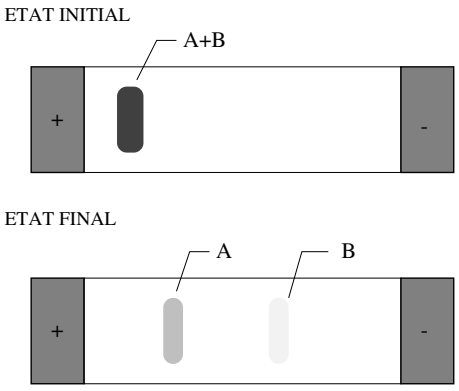


FIGURE 2.2 – L'électrophorèse de zone

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 2.3). Le mélange d'espèces ioniques est injecté ponctuel-

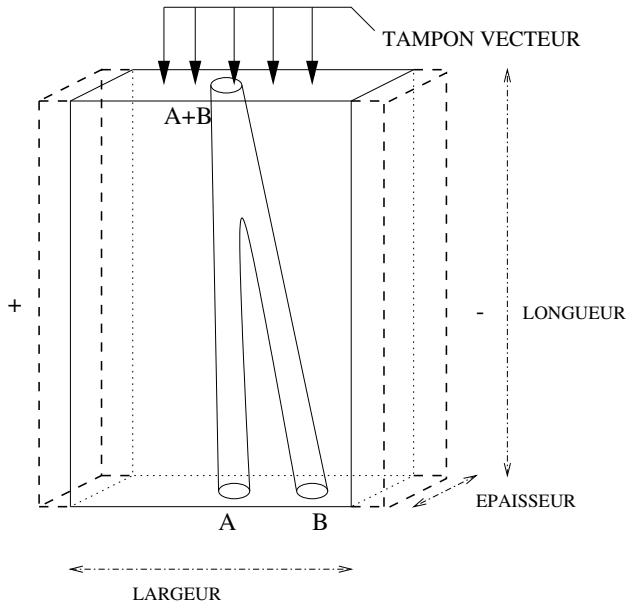


FIGURE 2.3 – L'électrophorèse de zone à écoulement continu

lement 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. 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.

La simulation est nécessaire pour trouver les conditions de faisabilité d'un tel procédé, en présence de phénomènes physiques couplés. Nous renvoyons aux publications [75, 76, 77, 78, 79] pour les expérimentations numériques. Une description très détaillée des équations, des schémas de discrétisations, du traitement des conditions limites (point particulièrement délicat pour l'équation de Navier-Stokes) est donnée dans la thèse de doctorat [105].

2.5.1 Equations

Les notations sont récapitulées dans le tableau 2.5. Le modèle tridimensionnel présenté ici est inspiré du modèle de M. J. Clifton, H. Roux de Balmann et V. Sanchez [106, 107]. La température est uniforme, les effets de la gravité sont négligés. Nous renvoyons à [105] pour une description détaillée des conditions aux

c	Concentration
D	Coefficient de diffusion
$\vec{E}$	Champ électrique
E_i, $i = 1, 2, 3$	Composantes du champ électrique
K	Conductivité électrique
p	Pression
$\vec{U}$	Champ de vitesse
u_i, $i = 1, 2, 3$	Composantes du champ de vitesse
ϵ	Permittivité diélectrique
λ	Conductivité ionique moyenne
μ	Mobilité électrophorétique
ν	Viscosité cinématique
ρ	Masse volumique
Φ	Potentiel électrique

Table 2.5 – Variables et paramètres du modèle physique de l'électrophorèse de zone à écoulement continu

limites et coefficients de matrices pour chaque équation.

L'écoulement incompressible L'écoulement du tampon vecteur est modélisé par une équation de Navier-Stokes incompressible tridimensionnelle :

$$\frac{\partial u_i}{\partial t} - \nu\,\Delta u_i + \sum_{j=1}^{3} u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \mathcal{F}_i,\ i = 1, 2, 3 \tag{2.17}$$

$$\mathrm{div}(\vec{U}) = 0. \tag{2.18}$$

Le terme $\mathcal{F}_i$ dépend des forces extérieures,

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

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

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

$$\frac{\partial c}{\partial t} - D\,\Delta c + \sum_{i=1}^{3} w_i \,\frac{\partial c}{\partial x_i} = 0. \tag{2.19}$$

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

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

où $\vec{U}$ est la vitesse d'écoulement régie par (2.17) et (2.18).

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

$$-\mathrm{div}(K\,\vec{\mathrm{grad}}(\Phi)) = 0, \tag{2.20}$$

avec

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

2.5.2 Discrétisation

La discrétisation des équations précédentes conduit á résoudre 7 systèmes linéaires par pas de temps. Les matrices de ces systémes sont toutes des M-matrices [76, 75].

Navier-Stokes Les équations (2.17) et (2.18) sont discrétisées par la méthode des volumes finis en cartésien avec grilles décalés. 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 [108]. 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 [108] ;
- la mise en œuvre de l'algorithme PISO conduit à la résolution de systèmes linéaires ayant les propriétés adéquates (M-matrices) pour l'utilisation des itérations parallèles asynchrones.

Voici un rappel succinct de la méthode PISO. Nous renvoyons à [108] pour une présentation détaillée.

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

$$A_i.u_i^{n+\frac{1}{3}} = -\frac{\partial}{\partial x_i}.p^n + \mathcal{F}_i. \tag{2.21}$$

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

$$-\sum_{i=1}^{3} \frac{\partial}{\partial x_i}.D_i^{-1}.\frac{\partial}{\partial x_i}.p^c = -\sum_{i=1}^{3} \frac{\partial}{\partial x_i} u_i^{n+\frac{1}{3}} \tag{2.22}$$

$$u_i^{n+\frac{2}{3}} = u_i^{n+\frac{1}{3}} - D_i^{-1}.\frac{\partial}{\partial x_i}.p^c. \tag{2.23}$$

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

$$-\sum_{i=1}^{3} \frac{\partial}{\partial x_i}.D_i^{-1}.\frac{\partial}{\partial x_i}.p^{cc} = -\sum_{i=1}^{3} \frac{\partial}{\partial x_i}.D_i^{-1}.H_i.(u_i^{n+\frac{2}{3}} - u_i^{n+\frac{1}{3}}) \tag{2.24}$$

$$u_i^{n+1} = u_i^{n+\frac{2}{3}} + D_i^{-1}.(H_i.(u_i^{n+\frac{2}{3}} - u_i^{n+\frac{1}{3}}) - \frac{\partial}{\partial x_i}.p^{cc}). \tag{2.25}$$

Les variables intermédiaires p^c et p^{cc} sont appelées *incréments de pression*. Les équations (2.22) et (2.24) sont usuellement appelées *équations de correction de pression*. $(A_i)_{i=1,2,3}$ sont des M-matrices associées à la discrétisation par volumes finis des équations de quantité de mouvement (2.17). 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

$$A_i = D_i - H_i \tag{2.26}$$

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

$$\begin{aligned} A_i.u_i^{n+\frac{2}{3}} &\simeq D_i.u_i^{n+\frac{2}{3}} - H_i.u_i^{n+\frac{1}{3}} \\ A_i.u_i^{n+1} &\simeq D_i.u_i^{n+1} - H_i.u_i^{n+\frac{2}{3}}. \end{aligned}$$

Notons que les M-matrices des systèmes d'équations associées à (2.22) et (2.22) sont mal conditionnées (même comportement que l'équation de Poisson stationnaire, anisotrope, avec conditions aux limites de Neumann prépondérantes).

Convection-diffusion linéaire L'équation de transport (2.19) est discrétisée en espace selon une techniques standard en différences finies (schémas *upwind* pour les termes convectifs). Le schéma d'évolution en temps a été traité par deux méthodes différentes :

1. le schéma implicite classique ; cette méthode conduit à résoudre un système linéaire par pas de temps, de la forme

$$(I_d + \delta_t A).c^{n+1} = c^n, \tag{2.27}$$

où $(I_d + \delta_t A)$ est une M-matrice (δ_t est le pas de temps).

2. le schéma explicite MPDATA (Fully Multidimensional Positive Definite Advection Transport Algorithm) de P. K. Smolarkiewicz [109, 110, 111] permettant de réduire la diffusion numérique ; cette méthode a été employée dans [107]. Chaque pas de temps consiste à appliquer successivement deux schéma temporel d'Euler explicites, l'un avec la vitesse de convection, l'autre avec la vitesse dite *anti-diffusive*. Cela revient à une discrétisation spatiale d'ordre 2 pour les termes convectifs.

Poisson anisotrope La discrétisation de l'équation de potentiel (2.20) est réalisée à l'aide d'une technique d'ordre 2 en espace, qui revient à moyenner deux schémas de discrétisation d'ordre 1. Le potentiel discret s'écrit :

$$A(c^n).\Phi^{n+1} = b^n, \tag{2.28}$$

où $A(c^n)$ est une M-matrice calculée en fonction de la concentration au pas de temps n, b^n un second membre contenant essentiellement les conditions aux limites de Dirichlet non homogènes. C'est l'équation résolue en premier à chaque pas de temps, suivie de PISO (2.21),(2.22),(2.24), puis de la convection diffusion (2.27).

2.6 Algorithmes à deux niveaux d'itérations

Dans cette section, nous décrivons la résolution d'équations de diffusion non-linéaires et non-stationnaires nécéssitant l'implémentaion de méthodes de linéarisation itératives. La discrétisation en temps est de type Euler implicite. Ceci garantit une stabilité inconditionnelle. Pour chaque pas de temps, des algorithmes à deux niveaux d'itérations ont été mis en œuvre. L'approche asynchrone ne concerne que le niveau d'itération interne, à savoir la résolution des systémes linéaires issus de chaque itération de linéarisation. Notons que les résultats d'existence et d'unicité de solution pour ces équations ne sont pas abordés ici. Les discrétisations et les résultats d'expérimentations numériques seront présentés.

2.6.1 Equations

G-mouvement Brownien Soit $\Omega \subset \mathbb{R}^n$ un domaine borné et u une fonction définie sur $[0,T] \times \Omega$. L'équation du G-mouvement Brownien sous sa forme générale (G-équation de la chaleur) s'écrit :

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{1}{2}\displaystyle\sum_{i=1}^{n}(\overline{\sigma_i}^2 \max(\frac{\partial^2 u}{\partial x_i^2}, 0) - \underline{\sigma_i}^2 \max(-\frac{\partial^2 u}{\partial x_i^2}, 0)) = 0 \text{ dans } [0,T] \times \Omega \\ u(t=0) = f \\ \text{conditions limites sur } \partial\Omega, \end{cases} \tag{2.29}$$

où f définit la condition initiale. En adoptant des simplifications ($\overline{\sigma_i} = \overline{\sigma}$, $\underline{\sigma_i} = \underline{\sigma}$, $n = 3$), et suite à quelques transformations sur l'équation (2.29) que nous ne détaillerons pas ici, nous obtenons :

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{1}{2}(\overline{\sigma}^2 \max(\Delta u, 0) + \underline{\sigma}^2 \min(\Delta u, 0)) = 0 \text{ dans } [0,T] \times \Omega \\ u(t=0) = f \\ u = 0 \text{ sur } \partial\Omega, \end{cases} \tag{2.30}$$

où Δ est l'opérateur Laplacien classique de l'équation de la chaleur. Dans la suite nous noterons $G(\Delta u)$ l'opérateur suivant :

$$G(\Delta u) = -\frac{1}{2}(\overline{\sigma}^2 \max(\Delta u, 0) + \underline{\sigma}^2 \min(\Delta u, 0)). \tag{2.31}$$

Milieux poreux Soit $\Omega \subset \mathbb{R}^3$ un domaine borné et u une fonction définie sur $[0,T] \times \Omega$, tel que $u(t,x) \in [0,1]$. L'équation du modèle étudié s'écrit :

$$\begin{cases} \dfrac{\partial u}{\partial t} - \Delta(\dfrac{u}{u+\theta})^m = 0 \text{ dans } [0,T] \times \Omega \\ u(t=0) = f \\ u = 0 \text{ sur } \partial\Omega, \end{cases} \tag{2.32}$$

où m est un entier positif impair, θ un paramètre réel strictement positif, et f une condition initiale. A l'aide du changement de variable suivant :

$$v = (\frac{u}{u+\theta})^m \tag{2.33}$$

où $v(t,x) \in [0, (\frac{1}{1+\theta})^m]$, le problème (2.32) devient :

$$\begin{cases} \dfrac{\partial \psi(v)}{\partial t} - \Delta v = 0 \text{ dans } [0,T] \times \Omega \\ v(t=0) = (\frac{f}{f+\theta})^m \\ v = 0 \text{ sur } \partial\Omega, \end{cases} \tag{2.34}$$

avec

$$\psi(v) = u = \frac{\theta v^{\frac{1}{m}}}{1 - v^{\frac{1}{m}}}. \tag{2.35}$$

2.6.2 Discrétisation

G-mouvement Brownien La discrétisation temporelle implicite de l'équation (2.30) selon un schéma d'Euler s'écrit :

$$u^{p+1} + \delta_t G(\Delta u^{p+1}) = u^p, \tag{2.36}$$

où δ_t est le pas de temps et p est l'indice associé au temps. Pour obtenir une discrétisation spatiale, écrivons $G(\Delta u)$ sous la forme :

$$G(\Delta u) \approx A(u).u$$

où $A(u)$ est une matrice calculée en fonction du vecteur u. D'après (2.31), $A(u)$ est une matrice dont les coefficients sont déterminés en fonction du signe du laplacien de u, selon la méthode suivante, en considérant un schéma classique à 7 points et un pas de discrétisation h uniforme à m points dans chaque direction de l'espace 3D :

$$\left\{ \begin{array}{lcl} (\Delta u)_i = 0 & \Rightarrow & A_{i,j} = 0 \text{ pour tout } j \\ (\Delta u)_i > 0 & \Rightarrow & A_{i,j} = -\overline{\sigma}\frac{1}{2h^2} \text{ pour } j = i \pm 1, i \pm m, i \pm m^2 \\ & & A_{i,i} = \overline{\sigma}\frac{3}{h^2} \\ (\Delta u)_i < 0 & \Rightarrow & A_{i,j} = -\underline{\sigma}\frac{1}{2h^2} \text{ pour } j = i \pm 1, i \pm m, i \pm m^2 \\ & & A_{i,i} = \underline{\sigma}\frac{3}{h^2}. \end{array} \right. \tag{2.37}$$

Le laplacien $(\Delta u)_i$ en chaque point de discrétisation indexé par i est aussi calculé avec un schéma classique. Le schéma implicite incluant la discrétisation s'écrit :

$$u^{p+1} + \delta_t A(u^{p+1}).u^{p+1} = u^p. \tag{2.38}$$

La méthode itérative permettant d'aboutir à (2.38) consiste à réitérer le calcul de la matrice, ainsi que la résolution d'un pas semi-implicite :

$$\left\{ \begin{array}{l} u^{p+1,0} = u^p \\ u^{p+1,q+1} + \delta_t A(u^{p+1,q}).u^{p+1,q+1} = u^p. \end{array} \right. \tag{2.39}$$

Cette méthode de linéarisation remplace la méthode de Newton qui n'est pas applicable ici car l'opérateur $G(\Delta u)$ n'est pas dérivable. L'arrêt de la linéarisation est déterminé par

$$\|u^{p+1,q+1} - u^{p+1,q}\|_\infty < \epsilon_L.$$

Le choix de la norme uniforme permet d'avoir des seuils ϵ_L indépendants du nombre de points de discrétisation. Enfin, on démontre facilement que

$$Id + \delta_t A(u^{p+1,q}) \tag{2.40}$$

est toujours une M-matrice, ce qui garantit la convergence des solveurs asynchrones.

Remarque 2.3 En pratique, la méthode de linéarisation (2.39) converge dans le cadre de la G-équation de la chaleur. L'étude théorique, ainsi que la généralisation à d'autres problèmes semblables du type

$$u^{p+1} + A(u^{p+1}).u^{p+1} = u^p$$

(c'est le cas du filtre de diffusion anisotrope section 2.3) n'a pas été abordée.

Remarque 2.4 Un schéma explicite très simple d'ordre 1 pour la G-équation de la chaleur peut s'écrire :

$$u^{p+1} = u^p - \delta_t A(u^p).u^p.$$

En l'absence de résultat donnant un critère de stabilité, seuls quelques essais à petite échelle ont été effectués afin de vérifier les résultats du schéma implicite, en prenant $\delta_t = \alpha \times h^2$ (α est une constante arbitrairement petite, h est le pas de discrétisation spatial).

Milieux poreux La discrétisation temporelle implicite de l'équation (2.34) selon un schéma d'Euler s'écrit :

$$\psi(v^{p+1}) + \delta_t \Delta v^{p+1} = \psi(v^p). \tag{2.41}$$

Grâce au changement de variable, la discrétisation spatiale s'écrit facilement sous la forme :

$$\psi(v^{p+1}) + \delta_t A.v^{p+1} - \psi(v^p) = 0, \tag{2.42}$$

où A est la matrice de discrétisation de l'opérateur laplacien classique. L'équation (2.42) est résolue à chaque pas de temps avec l'algorithme de Newton, dont les itérations s'écrivent :

$$\begin{cases} v^{p+1,0} = v^p \\ (\delta_t A + \psi'(v^{p+1,q})).\delta X = -(\psi(v^{p+1,q}) + \delta_t A.v^{p+1,q} - \psi(v^p)) \\ v^{p+1,q+1} = v^{p+1,q} + \delta X, \end{cases} \tag{2.43}$$

avec

$$\psi'(z) = \frac{\theta}{m} \frac{z^{\frac{1}{m}-1}}{(1 - z^{\frac{1}{m}})^2}. \tag{2.44}$$

L'arrêt de la linéarisation est déterminé par

$$\|\delta X\|_\infty < \epsilon_L.$$

Le choix de la norme uniforme permet d'avoir des seuils ϵ_L indépendants du nombre de points de discrétisation. La valeur de la fonction ψ' (2.44) prise en $z = v^{p+1,q}$ est un vecteur positif ajouté à la diagonale de la matrice $\delta_t A$. Ce sont donc des M-matrices, ce qui assure la convergence des méthodes asynchrones.

Remarque 2.5 Le développement en série entière de $\psi(v)$ (avec $0 \leq v < 1$) est :

$$\psi(v) = \theta v^{\frac{1}{m}} \sum_{k=0}^{\infty} v^{\frac{k}{m}}.$$

Cela donne accès à une primitive de ψ, permettant d'étudier l'équation stationnaire

$$\psi(v) + \delta_t \Delta v = g$$

sous la forme d'un problème d'optimisation. L'existence et l'unicité de solution aux équations (2.41) peuvent alors être établies.

Remarque 2.6 Un schéma explicite très simple d'ordre 1 pour l'équation de milieu poreux peut s'écrire sans changement de variable :

$$u^{p+1} = u^p - \delta_t A.(\frac{u^p}{u^p + \theta})^m.$$

En l'absence de résultat donnant un critère de stabilité, seuls quelques essais à petite échelle ont été effectués afin de vérifier les résultats du schéma implicite avec changement de variable, en prenant $\delta_t = \alpha \times h^2$ (α est une constante arbitrairement petite, h est le pas de discrétisation spatial).

2.6.3 Résultats

Les expérimentations numériques pour ces deux problèmes ont été effectuées sur le serveur de calcul HPC@LR. Les nœuds de calcul utilisés sont des bi-processeurs de type Intel Xeon Westmere, connectés via réseau Infiniband 40 Gb/s. Chaque nœud contient deux processeurs ayant chacun 6 core, accédant à une mémoire de 24 Go.

Le solveur de systèmes linéaires employé pour ces deux problèmes est un algorithme de Gauss-Seidel par points. Le critère de convergence utilisé est la norme uniforme de la différence entre deux itérations successives. Le domaine Ω est le cube $[0, 1]^3$, découpé suivant les axes y et z. Le sous-domaine assigné à chaque processus MPI est donc un parallélépipède.

G-mouvement Brownien Pour ce problème, les paramètres utilisés lors des expérimentations numériques sont :

- nombre de points de discrétisation en x, y, et z : 300,
- $\underline{\sigma} = 0.353$,
- $\overline{\sigma} = 0.707$,
- $\delta_t = 0.01$,
- nombre de pas de temps : 5,
- seuil de tolérance pour l'arrêt de la linéarisation : 10^{-8},
- seuil de tolérance pour l'arrêt des relaxations : 10^{-10}.

La condition initiale est :

$$u(t = 0) = f(x, y, z) = 10^7 \times P(x) \times P(y) \times P(z)$$

où

$$P(\chi) = \chi(1 - \chi)(0.5 - \chi)(0.333 - \chi)(0.666 - \chi).$$

Cette condition initiale est admissible pour les conditions limites de Dirichlet et sa courbure change de sens, ceci afin de faire varier le signe de Δu.

nb. nœud	nb. core	synchrone			asynchrone		
		temps	accélération	efficacité	temps	accélération	efficacité
1	1	50413	-	-	-	-	-
1	6	14549	3.46	0.57	15695	3.21	0.53
1	12	7433	6.78	0.56	8564	5.88	0.49
2	24	4778	10.55	0.43	4153	12.13	0.50
4	48	1911	26.38	0.54	2073	24.31	0.50
6	72	1291	39.04	0.54	1421	35.47	0.49
8	96	1321	38.16	0.39	1053	47.87	0.49
10	120	1130	44.61	0.37	840	60.01	0.50

TABLE 2.6 – Temps de calcul (sec.), accélérations et efficacités pour le G-mouvement Brownien 3D ($300 \times 300 \times 300$) sur 5 pas de temps.

nb. nœud	nb. core	synchrone		asynchrone	
		relaxations	iter. lin.	relaxations	iter. lin.
1	1	68972	25	-	-
1	6	69253	25	71859	25
1	12	69491	25	76225	26
2	24	69701	25	78343	27
4	48	69941	25	77033	28
6	72	70104	25	78940	28
8	96	70321	25	79416	29
10	120	70466	25	80171	28

TABLE 2.7 – Nombres de relaxations et d'itérations de linéarisation pour le G-mouvement Brownien 3D ($300 \times 300 \times 300$) sur 5 pas de temps.

Milieux poreux Pour ce problème, les paramètres utilisés lors des expérimentations numériques sont :

- nombre de points de discrétisation en x, y, et z : 300,
- $\theta = 0.5$,
- $\delta_t = 0.01$,
- nombre de pas de temps : 5,
- seuil de tolérance pour l'arrêt de la linéarisation : 10^{-8},
- seuil de tolérance pour l'arrêt des relaxations : 10^{-10}.

La condition initiale est :

$$u(t=0) = f(x,y,z) = 0.5 \times \sin(\pi x) \times \sin(\pi y) \times \sin(\pi z).$$

Cette condition initiale est admissible pour les conditions limites de Dirichlet.

nb. nœud	nb. core	efficacité moyenne par relaxation	
		synchrone	asynchrone
1	6	0.57	0.55
1	12	0.56	0.54
2	24	0.44	0.57
4	48	0.55	0.56
6	72	0.55	0.56
8	96	0.40	0.57
10	120	0.37	0.58

TABLE 2.8 – Efficacités moyennes par relaxation pour le G-mouvement Brownien 3D ($300 \times 300 \times 300$) sur 5 pas de temps.

nb. nœud	nb. core	synchrone			asynchrone		
		temps	accélération	efficacité	temps	accélération	efficacité
1	1	54264	-	-	-	-	-
1	6	14328	3.78	0.63	16600	3.26	0.54
1	12	8621	6.29	0.52	10140	5.35	0.44
2	24	4100	13.23	0.55	4652	11.66	0.48
4	48	2307	23.52	0.48	2399	22.61	0.47
6	72	1505	36.05	0.50	1613	33.64	0.46
8	96	1666	32.57	0.33	1255	43.23	0.45
10	120	1358	39.95	0.33	978	55.48	0.46

TABLE 2.9 – Temps de calcul (sec.), accélérations et efficacités pour le modèle milieu poreux 3D ($300 \times 300 \times 300$) sur 5 pas de temps.

Interprération des résultats A première vue les efficacités (tableaux 2.6 et 2.9) sont faibles en raison du poids des communications par rapport au poids des calculs. Le découpage en sous-domaines n'est pas optimal. Les calculateurs actuels ont des vitesses de calcul qui progressent plus rapidement que les vitesses de communication. Avec le même type de découpage, les efficacités obtenues en 2007 publiés dans [80] étaient meilleurs. L'importance de l'optimisation du découpage en sous-domaines est devenu primordial. Ceci dit, pour la comparaison entre algorithmes synchrones et asynchrones, cela ne pose pas de problème. Mais pour exploiter les algorithmes asynchrones dans des codes où la performance est un facteur décisif, l'optimisation du découpage en sous-domaines est impératif, afin de minimiser le poids des communications. L'utilisation d'outils classiques d'équilibrage de charge (découper le maillage équitablement, tout en minimisant les volumes de communication) tels que METIS ou Zoltan, reste la méthode la plus simple pour résoudre ce problème de manque d'efficacité. Cela revient à s'inspirer de la démarche du chapitre 3 de la thèse [112],

nb. nœud	nb. core	synchrone		asynchrone	
		relaxations	iter. lin.	relaxations	iter. lin.
1	1	80489	21	-	-
1	6	81025	21	93088	21
1	12	81073	21	91710	21
2	24	81421	21	89311	22
4	48	81700	21	89284	25
6	72	81880	21	90238	26
8	96	82148	21	94722	27
10	120	82292	21	94210	26

TABLE 2.10 – Nombres de relaxations et d'itérations de linéarisation pour le modèle milieu poreux 3D ($300 \times 300 \times 300$) sur 5 pas de temps.

nb. nœud	nb. core	efficacité moyenne par relaxation	
		synchrone	asynchrone
1	6	0.63	0.63
1	12	0.52	0.50
2	24	0.55	0.53
4	48	0.49	0.52
6	72	0.50	0.52
8	96	0.34	0.52
10	120	0.34	0.54

TABLE 2.11 – Efficacités moyennes par relaxation pour le modèle milieu poreux 3D ($300 \times 300 \times 300$) sur 5 pas de temps.

adoptée dans le cadre de GMRES. Attention, à ne pas confondre avec l'équilibrage de charge au sens architecture et réseaux [86]. En faisant abstraction des problèmes de performance liés au découpage, nous pouvons tirer les conclusions suivantes.

- Les tableaux 2.6 et 2.9 montrent que les algorithmes synchrones sont meilleurs que les algorithmes asynchrones, jusqu'à un certain nombre de nœuds. Les tableaux 2.8 et 2.11 montrent que la chute de performance des algorithmes synchrones n'est pas liée au ralentissement de la convergence de Gauss-Seidel (augmentation du nombre de relaxations liée à la parallélisation, mis en évidence dans les tableaux 2.7 et 2.10).
- Ces tableaux montrent aussi que les performances de l'algorithme asynchrone ont moins tendance à varier. L'algorithme asynchrone reste moins sensible aux perturbations du réseau de communication, qui sont liées en l'occurrence au partage des ressources du serveur de calcul entre plusieurs utilisateurs.
- Enfin, les tableaux 2.7 et 2.10 révèlent une légère augmentation du nombre

d'itération de linéarisation avec le solveur asynchrone. Cela signifie que les solutions fournies par le solveur asychnrone utilisé dans cette étude sont légèrement moins précises que celles du solveur synchrone. Cet aspect est dépendant du test d'arrêt des algorithmes asynchrones, et sera discuté de manière plus détaillée dans la section 3.2.3 du chapitre 3. Notons que l'augmentation du nombre total de relaxations est aussi une conséquence de ces quelques itérations supplémentaires dans la linárisation.

Chapitre 3

Solveur parallèle asynchrone

3.1 Introduction

Les premières implémentations des algorithmes parallèles asynchrones ont été réalisées dès 1967 par J.L. Rosenfeld [113] 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 [1]. Suite aux analyses de convergence dans le cas non linéaire [114, 2, 18], G. Baudet a effectué la première étude de performance sur calculateur parallèle [3]. 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 [115, 116] 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 [50, 51],
- le problème de l'obstacle [35, 40, 41, 45, 105],
- la programmation dynamique [65],
- les problèmes d'optimisation et de flot [47, 48, 49],
- les équations aux dérivées partielles [36, 37, 38, 117, 80, 75, 76, 105],
- les problèmes de commande optimale [52].

Ce chapitre illustre une implémentation en C et MPI de l'algorithme parallèle asynchrone présenté dans la section 1.2. Le développement a été effectué en 2010 pour les projets de recherche du chapitre 2. Dans le contexte du projet ANR-CIP, son adaptation à la bibliothèque de calcul parallèle asynchrone P2PDC [90] a été développée en collaboration avec Thierry Garcia [71, 78, 72], ainsi qu'un étudiant en thèse (The Tung Nguyen [81, 118]). A l'occasion de ces travaux avec P2PDC, la librairie P2PDC a été étendue, notamment avec une routine de synchronisation

`P2P_Wait` qui était absente à l'origine, et qui s'est révélée être indispensable pour les schémas d'évolution et pour les tests d'arrêt. En effet, les échanges de messages dans P2PDC sont soit toujours synchrones, soit toujours asynchrones, or il était impossible de changer de mode de communication en cours de calcul. De plus, des opérations collectives telles que `Scatter` et `Gather` ont pu voir le jour dans l'environnement P2PDC grâce à `P2P_Wait`.

Ce programme est adapté aux problémes 3D avec matrice creuse et vecteur plein en maillage cartésien. Sa parallélisation consiste à découper les axes y et z d'un parallélépipède. Sa conception est plus basique que les codes plus généraux pour maillages non structurés. Ces compromis facilitent l'utilisation, la lecture et les modifications. Notons que ce type de découpage n'est pas optimal pour l'efficacité en parallèle. Dans le cadre de la comparaison entre les approches synchrones et asynchrones, il est tout à fait satisfaisant. Ce code a servi pour :

- des tests de performance (MPI, P2PDC) sur le calculateur national Grid 5000, notamment pour des systèmes d'équations non-linéaires (problème d'obstacle, mathématiques financières) [73, 84, 72, 82, 71, 81], la simulation de la séparation de protéines par électrophorèse [79, 78, 83, 77],
- des calculs au CINES en traitement d'images (3D PET) où plusieurs systèmes indépendants sont résolus en parallèle.

Il a aussi servi à Lilia Ziane Khodja qui a porté et expérimenté des calculs asynchrones sur cluster de GPU [74].

3.2 Implémentation avec MPI

3.2.1 Algorithmes numériques

Le découpage du domaine conduit naturellement à considérer des blocs de composantes selon l'axe des x. On obtient ainsi des blocs contigus pour les vecteurs et des blocs tri-diagonaux pour les matrices. Il est donc facile de mettre en œuvre une méthode directe pour inverser les blocs tri-diagonaux (méthode de Gauss). Chaque processeur prend en charge pluisieurs blocs-composantes adjacents, de sorte que la balyage des blocs soit de type Gauss-Seidel. La base du code asynchrone développé est un solveur Gauss-Seidel par blocs, munie d'une procédure qui répartit les blocs en accord avec un découpage selon les axes y et z du domaine de l'équation aux dérivées partielles.

Quant aux solveurs pour problèmes non linéaires, la projection a lieu après l'inversion de chaque bloc-composante dans le cas de l'algorithme par blocs (projected block-Gauss-Seidel). Pour le cas de l'algorithme de Richardson projeté, qui est un algorithme par points, les blocs sur chaque processeurs sont considérés comme un seul grand bloc.

```
int asynchronousRecv(solverData_t *alldat) {
  int i, recv_flag; MPI_Status st;

  for(i=0; i<N4; i++) {
    if(alldat->vflags[i]) {
      if(alldat->vecRecv[i] != MPI_REQUEST_NULL) {
        MPI_Test(alldat->vecRecv + i,& recv_flag,& st);
      } else {
        /* NOT REACHED */
      }

      if(recv_flag) {
        alldat->nrecv ++;
        MPI_Irecv(
            alldat->recvBuf[i],alldat->nmess[i],MPI_DOUBLE,alldat->topo[i],
            TAG_VEC,alldat->mpiCom,alldat->vecRecv + i);
      }
    }
  }
}
```

FIGURE 3.1 – Réception asynchrones avec MPI .

3.2.2 Communications asynchrones

Les communications asynchrones sont implémentées en MPI à l'aide de routines non bloquantes standard. Les procédures sont présentés dans les figures 3.1 et 3.2.

Notons que les requêtes de communication dites persistantes de la bibliothèque MPI n'ont pas été utilisées. Il a été constaté que l'annulation de telles requêtes a tendance à échouer plus souvent. Notons également que le mode d'envoi *synchrone* est utilisé (le message part sans transiter dans une mémoire tampon quand un récepteur est prêt) car c'est le plus robuste et le moins gourmand en mémoire. Pour cette raison,

```
static int asynchronous_pack_isend(solverData_t *alldat, double *Vec, int i) {
  all_pack(alldat->pack_i[i][0],alldat->pack_i[i][1],
      Vec + alldat->pack_i[i][2],
      alldat->pack_i[i][3],alldat->pack_i[i][4],
      alldat->sendBuf[i]);

  MPI_Issend(
      alldat->sendBuf[i],alldat->nmess[i],MPI_DOUBLE,alldat->topo[i],
      TAG_VEC,alldat->mpiCom,alldat->vecSend + i);
}
```

FIGURE 3.2 – Envois asynchrones avec MPI .

chaque réception est réactivée dès qu'un message est reçu. Enfin, le paquetage des données est systématique, afin de toujours utiliser les routines MPI les plus rapides.

Remarque 3.1 Les communications synchrones ont été implémentées avec les routines MPI les plus efficaces : `MPI_Sendrecv`.

3.2.3 Terminaison asynchrone

Le test d'arrêt de Bertsekas

La détection de la convergence des itérations parallèles synchrones et asynchrones est basée sur le principe suivant [14] : 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. Les critères de convergence locaux sont par exemple, la norme du résidu pour un bloc-composante, la différence entre deux bloc-composantes successives. 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. 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 [14, 119], nous avons choisi d'implémenter un algorithme de terminaison utilisant le *snapshot* de K.M. Chandy et L. Lamport [120]. Le principe de cet algorithme est donné dans [14].

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

$$\begin{array}{c}\text{la convergence locale est vérifiée sur tous les processus}\\ \textbf{et}\\ \text{toutes les composantes du vecteur itéré envoyées ont été reçues.}\end{array} \tag{3.1}$$

La proposition (3.1) 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 :

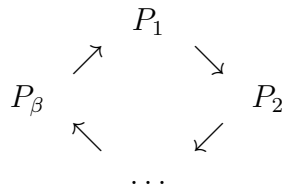
$$\begin{array}{c}\text{tout processus vérifiant la convergence locale}\\ \text{n'envoie plus de composante du vecteur itéré}\end{array} \tag{3.2}$$

afin de rendre la détection de la convergence globale possible. La justification de ce procédé est donnée dans [121], section 9.

Cependant, la contrainte (3.2) 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 (3.2) avec la contrainte suivante :

$$\begin{array}{c}\text{tout processus vérifiant la convergence locale}\\ \text{doit s'assurer que les valeurs les plus récentes}\\ \text{de sa composante du vecteur itéré soient envoyées.}\end{array} \tag{3.3}$$

Le test d'arrêt consiste à faire circuler un *jeton* (petit message) qui contient les données permettant d'évaluer la condition (3.1) parmi les processus.



Ce jeton est initialement émis par un des processus choisi arbitrairement. Ensuite, chaque processus doit le retransmettre en utilisant des opérations non-bloquantes. Le jeton contient les données suivantes :

- 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 ;

chaque processus doit compter les nombres de requêtes d'envoi et de réception qui se sont terminées depuis le dernier passage du jeton. 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.

Remarque 3.2 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.

Remarque 3.3 Le nombre maximum de tours de jeton permet de définir une condition d'arrêt pour les algorithmes asynchrones.

Remarque 3.4 Actuellement, la routine MPI qui permet d'envoyer les informations pour l'implémentation du test d'arrêt (jeton et message de décision d'arrêt) est `MPI_Bsend`. Cette routine convient parfaitement à des messages de petite taille

(quelques entier de type `MPI_INT`). Sur certains centres de calculs, nous avons remarqué que le buffer interne de MPI doit être alloué par l'utilisateur, alors que dans la majorité des cas, cette allocation est automatiquement effectuée par la librairie MPI. L'utilisation de cette routine est donc fortement déconseillée afin d'assurer la portabilité.

Limitations de la méthode de Bertsekas

La procédure décrite ci-dessus a révélé des insuffisances sur deux points, lors d'une tentative de passage à plus grande échelle (plusieurs milliers de core) :

1. la topologie en anneau induite par la technique du jeton circulant n'est pas assez efficace,
2. la contrainte qui consiste à cesser les envois de valeurs aux frontières de recouvrement conduit à des arrêts prématurés qui sont d'autant plus nuisibles que le nombre de processeurs est élevé (on peut constater les prémices de ce phénomène dans les tableaux 2.7 et 2.10 de la section 2.6.3).

Cette tentative de passage à plus grande échelle, réalisée en 2012 avec la collaboration avec l'Université de Franche-Comté, n'a malheureusement pas donné lieu à des publications, ni été présentée en congrès.

Durant le projet ANR-CIP, des comparaisons entre deux tests d'arrêts ont été effectués :

1. un test d'arrêt *décentralisé* similaire à la méthode de Bertsekas,
2. un test d'arrêt *centralisé* [122] qui consiste à effectuer l'opération `Gather` de façon asynchrone ; cette méthode a été implémentée avec les communications asynchrones de la librairie P2PCD.

L'étude publiée dans [72] révèle que la méthode centralisée est plus robuste en raison d'un nombre plus élevé de relaxations effectuées avant l'arrêt. Dans cette étude l'implémentation du test d'arrêt centralisé est basée sur l'utilisation d'un nœud maître exclusivement dédié à la réception des informations de convergence. C'est donc une topologie en étoile qui est sous-jacente, ce qui risque de poser des problèmes d'efficacité avec un très grand nombre de nœuds. L'arrêt du calcul est décidé lorsque l'information de convergence locale sur tous les processeurs est confirmée à plusieurs reprises.

Les données collectées durant ces deux études permettent de formuler un bon compromis pour une procédure d'arrêt rapide et robuste :

- optimiser la collecte des informations de convergence avec une topologie en arbre ou en hypercube ; ceci, réduit considérablement ($\mathcal{O}(\log(n))$) la durée de la collecte et minimise le nombre de messages de convergence à traiter sur tous les nœuds,
- même en cas de convergence locale, les processus doivent continuer à itérer et mettre à jour les valeurs aux frontières pour les voisins ; ceci aura une

répercussion sur la manière de libérer les ressources (acquitement des derniers messages en cours, voir section 3.2.4),
– adopter la méthode de confirmation à plusieurs reprises pour l'évaluation asynchrone de l'état global des itérations parallèles ; en pratique, une dizaine de confirmations s'est avéré suffisant [72], mais un étude plus approfondie est nécessaire pour vérifier dans quelle mesure le nombre de confirmations est problème dépendant.

3.2.4 Libération des ressources

La libération des ressources à la fin d'une résolution de système d'équations est importante pour assurrer la ré-entrance des programmes. La proprété de ré-entrance est incontournable dès que la résolution d'un système d'équations est lui-même inclus dans un processus itératif (schémas d'évolution, linárisations). Cette phase est délicate quand il s'agit des requêtes d'envois MPI : le standard spécifie clairement que leur annulation peut échouer. La procédure présentée dans la figure 3.3 décrit une manière d'éviter le plus souvent possible les cas d'échecs : elle consiste à redémarrer les requêtes de réceptions car l'annulation des envois nécessite des récepteurs prêts. Les barrières de synchronisation entre les différentes phases de la procédure sont indispensables.

```
int asyncFinalize(solverData_t *alldat) {
  int i; MPI_Status st;

  MPI_Barrier(alldat->mpiCom);
  /* Annuler et redemarrer les receptions */
  for(i=0; i<N4; i++) {
    if(alldat->vflags[i]) {
      if(alldat->vecRecv[i] != MPI_REQUEST_NULL) {
        MPI_Cancel(alldat->vecRecv + i);
        MPI_Wait(alldat->vecRecv + i,& st);
      }
      MPI_Irecv(
          alldat->recvBuf[i],alldat->nmess[i],MPI_DOUBLE,alldat->topo[i],
          TAG_VEC,alldat->mpiCom,alldat->vecRecv + i);
    }
  }
  MPI_Barrier(alldat->mpiCom);
  /* Annulation des envois */
  for(i=0; i<N4; i++) {
    int flag;

    if(alldat->vflags[i]) {
      if(alldat->vecSend[i] != MPI_REQUEST_NULL) {
        MPI_Test(alldat->vecSend + i,& flag,& st);
        if(! flag) {
          MPI_Cancel(alldat->vecSend + i);
          MPI_Wait(alldat->vecSend + i,& st);
        }
      }
    }
  }
  MPI_Barrier(alldat->mpiCom);
  /* Annulation des receptions */
  for(i=0; i<N4; i++) {
    if(alldat->vflags[i]) {
      if(alldat->vecRecv[i] != MPI_REQUEST_NULL) {
        MPI_Cancel(alldat->vecRecv + i);
        MPI_Wait(alldat->vecRecv + i,& st);
      }
    }
  }
  MPI_Barrier(alldat->mpiCom);
  /* Annulation des echanges de jeton */
  if(alldat->tokRecv != MPI_REQUEST_NULL) {
    MPI_Cancel(& alldat->tokRecv);
    MPI_Wait(& alldat->tokRecv,& st);
  }
}
```

FIGURE 3.3 – Procédure de libération des ressources MPI .

Chapitre 4

Simulations CFD vasculaires

4.1 Introduction

Ce chapitre contient une synthèse sur les développements réalisés au sein du projet ANR-OCFIA (CIS-2007), pour la partie concernant l'optimisation des procédés de simulation et de traitement d'image. Il contient une description des amélioration apportées au code de calcul des déformations non-linéaires (maillages mobiles), au procedé de simulation dans sa globalité. Il mentionne également une contribution à la réalisation d'un code CFD qui est proche des images médicales (maillage cartésien structuré) et qui répond à l'aspect patient-spécifique de la modélisation des organes vivants. Ce chapitre n'a pas de lien direct avec les algorithmes asynchrones. C'est un compte-rendu des travaux effectués en tant qu'ingénieur de recherche, au sein de la société privée ASA (Advanced Solutions Accelerator). L'application traitée ici est un exemple très complet et très complexe en calcul scientifique.

Pour situer le contexte, le projet OCFIA (ANR-CIS 2007) (Optimised Computational Functional Imaging for Arteries) est un projet de recherche collaborative entre 3 entités, I2MC (Institut des Maladies Métaboliques et Cardiovasculaires, INSERM Toulouse), I3M (Institut de Mathématiques et de Modélisation de Montpellier, UM2), et ASA (Advanced Solutions Accelerator, PME basée à Montpellier). Ce projet qui a débuté en 2008, a pour objectif d'optimiser un procédé de traitement de l'information dont la fonction est d'effectuer des simulations CFD à partir d'images médicales issues d'une séquence IRM adaptée (voir figure 4.1). Le travail d'optimisation du proédé a eu pour objectifs la réduction du temps de calcul (codes de calcul parallèles, technologie HPC) d'une part et la minimisation des opérations nécessitant intervention humaine chronophages d'autre part. Le point de départ des travaux est la thèse de doctorat de Ramiro Moreno [123]. Ce projet de recherche a donné lieu à des publications [124, 125, 126], et à des collaborations avec 3 étudiants en thèse de doctorat : Shirod Jeetoo [127], Adil Baali [128] et Bruno Tayllamin [129].

La section 4.2 présente les améliorations apportées à la méthode de calcul des maillages mobiles. La section 4.3 présente les améliorations apportées à la chaîne

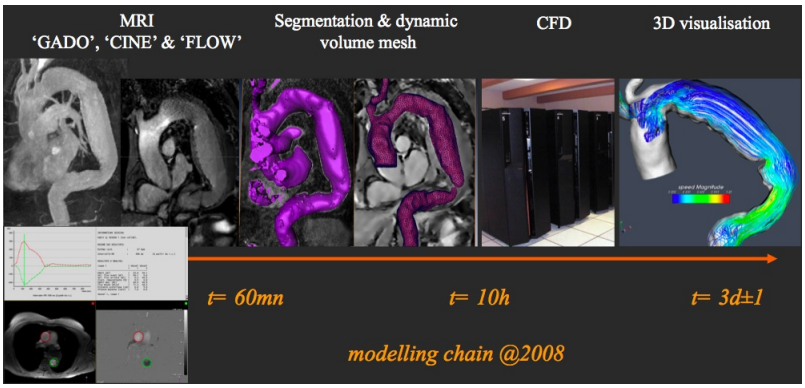


FIGURE 4.1 – Description du procédé OCFIA et de ses performances en début 2008.

de traitement OCFIA dans sa globalité. Quant à la section 4.4, elle présente une participation à la réalisation d'un solveur CFD principalement développé à l'UM2.

4.2 Le calcul des déformations non-linéaires

4.2.1 Introduction

La méthode de calcul des déformations non-linéaires est initialement implémentée sous environnement Matlab, dans la bibliothèque SPM2 (Statistical Parametric Mapping, développé au Institute of Neurology de University College London [130, 131]). Le calcul effectif du noyau de déformation est un algorithme de descente dépourvu de test d'arrêt qui minimise une fonctionnelle. Cet algorithme a été originellement conçu pour effectuer du recalage d'image (déformer des images cérébrales pour les faire correspondre à un atlas standard du cerveau).

L'action qui a été menée au niveau du calcul de champ de déformation a consisté à :

1. porter le noyau de calcul de déformation (code CMex) sur un environnement de compilation de type Unix,
2. implémenter un test d'arrêt rudimentaire consistant à utiliser la variance entre le modèle et l'image source déformée à l'itération courante ; la mise en œuvre de critères du premier ou deuxième ordre prenant en compte le gradient ou le Hessien de la fonctionnelle de coût a été jugé trop difficile, notamment à cause de la difficulté à effectuer un reverse engineering exhaustif du noyau de calcul des déformations non-linéaires,
3. remettre à plat, corriger, optimiser et porter sous Unix les scripts Matlab permettant d'appeler le noyau de calcul,

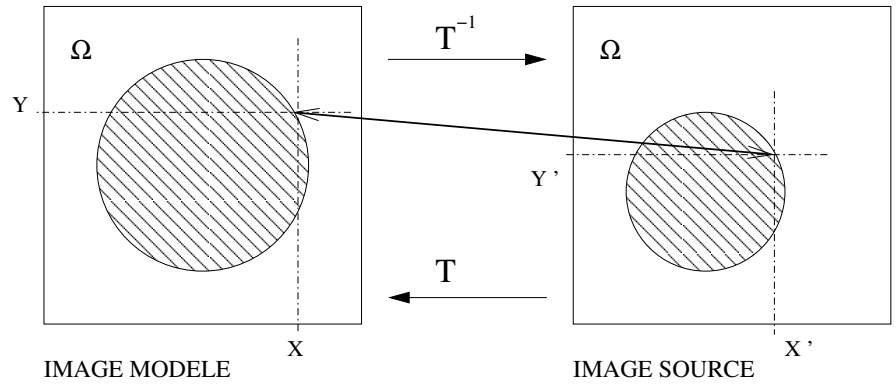


FIGURE 4.2 – Vue schématique en coupe de la déformation entre 2 cercles de centre et de rayon distincts. Par convention, l'image modèle est celle qui sert de référence (atlas cérébral ou image segmentée ayant servi à mailler l'artère) et l'image source est celle qu'on cherche à recaler. T^{-1} est la transformation qui associe $(X, Y) \mapsto (X', Y')$, autrement dit qui recale la source sur le modèle, et en pratique qui sert à obtenir le maillage de l'artère sur l'image source. T est celle qui recale le modèle sur la source.

4. porter en C les opérations de traitement d'image nécessaires au calcul de déformation (quelques éléments de la bibliothèque SPM2).

4.2.2 De la théorie à la pratique

Le principe du recalage (voir figure 4.2) consiste à estimer une bijection [1] permettant de faire correspondre les pixels de 2 images distinctes. En formalisant en 3D à l'aide des notations de la figure 4.2 (représentation en 2D), on cherche à déterminer une bijection T du domaine de l'image (le domaine $\Omega \subset \mathbb{R}^3$), telle que :

$$\begin{aligned} S(T^{-1}(X, Y, Z)) &= M(X, Y, Z) \\ M(T(X', Y', Z')) &= S(X', Y', Z') \end{aligned}$$

où S et M sont respectivement les champs d'intensité des images source et modèle (fonctions de $\Omega \mapsto \mathbb{R}$). On peut le voir comme la recherche d'un "bon" changement de variable où seule la fonction inverse T^{-1} est utile en pratique.

Afin de clarifier le fait que ce soit la transformation inverse T^{-1} qui soit utile en pratique, il y a deux confusions à éviter à tout prix.

1. Un champ de déformation (T ou T^{-1}) transforme l'espace, pas les images (du moins de façon directe). Les images servent dans la recherche d'*une* bonne transformation de l'espace, à savoir une qui fait correspondre 2 images.

1. On peut parler d'homéomorphisme car une bijection non bi-continue ne permet pas un mouvement de maillage correct.

2. L'acte de transformer une image par l'intermédiaire d'un champ de déformation consiste à ré-échantillonner l'intensité des pixels à l'aide d'un champ de déformation. Donc pour aller de l'image source vers l'image modèle (sens conventionnel en neuro-imagerie), on ré-échantillonne à l'aide du T^{-1} (fonction inverse) calculée par l'algorithme de SPM2. En construisant le maillage de l'artère sur l'image modèle (et non sur l'image source), l'algorithme calcule directement la transformation qui permet le déplacement des coordonnées de maillage.

La méthode de calcul des déformations dans SPM2 repose sur la résolution d'un problème d'optimisation de type moindre carré (4.1) + un terme régularisateur (4.2) :

$$f_1(T) = \int_\Omega (M(x) - S(T^{-1}(x)))^2 dx \tag{4.1}$$

$$f_2(T) = \int_\Omega (\det(J_T(x)) - 1)dx + \int_\Omega (\det(J_{T^{-1}}(x)) - 1)dx \tag{4.2}$$

$$f(T) = f_1(T) + \lambda f_2(T) \tag{4.3}$$

où $x \in \Omega \subset \mathbb{R}^3$ et λ est un nombre réel strictement positif appelé coefficient de régularisation. La notation J_T signifie la matrice Jacobienne d'un champ de déformation en tout point du domaine [2]. Nous voyons d'emblée que l'évaluation du terme régularisateur (4.2), ainsi que sa dérivée représentent des difficultés calculatoires majeures. Les auteurs de SPM2 simplifient ces calculs en effectuant des approximations par développements limités.

Afin d'améliorer la qualité du calcul des gradients de la fonctionnelle f_1, un pré-lissage des images M et S est mis en place. En écrivant $\mathcal{G}^\sigma$ le noyau du filtre Gaussien de largeur σ et $\mathcal{G}^\sigma * I$ l'application du filtre à une image I par convolution, la fonctionnelle f_1 devient

$$f_1^\sigma(T) = \int_\Omega ([\mathcal{G}^\sigma * M](x) - [\mathcal{G}^\sigma * S](T^{-1}(x)))^2 dx. \tag{4.4}$$

Attention, ne pas confondre la convolée d'une image ré-échantillonnée

$$\mathcal{G}^\sigma * [I \circ T]$$

avec le ré-échantillonnage d'une image convolée

$$[\mathcal{G}^\sigma * I] \circ T.$$

On parle bien ici de ré-échantillonnage et a fortiori de recalage d'images convolées (*pré-lissées*). L'idée intuitive qui est sous-jacente est : avec des images plus lisses, on

2. Chercher une déformation bi-continue ne suffit donc pas.

obtient une fonctionnelle plus lisse, de même pour son gradient. Une valeur élevée de σ permet à l'algorithme de se focaliser sur les mouvements d'ensemble (recalage entre images floues). Une valeur faible de σ permet à l'algorithme de prendre en compte les détails.

Il est même possible de recaler deux images de manière itérative en jouant sur le paramètre σ. Soit $\mathcal{T}^\sigma$ le champ de déformation qui est *une* solution du problème d'optimisation (4.3) utilisant la fonctionnelle lissée (4.4). Avec une suite décroissante de tailles de noyau Gaussien ($\sigma_1 > \sigma_2 > \ldots > \sigma_n$) et en utilisant $\mathcal{T}^{\sigma_{i-1}}$ pour initialiser le calcul de $\mathcal{T}^{\sigma_i}$, il est possible d'effectuer un recalage progressif qui prend en compte les détails des images au fur et à mesure [3].

Enfin, il s'est avéré que le choix du terme régularisateur λ n'apportait pas d'amélioration significative à la résolution du problème de recalage (4.3). L'utilisation des solutions issues de SPM2 pour déformer le maillage du l'artère aorte engendrait trop souvent des tétrahèdres non-conformes (vérification automatique à l'aide de l'outil HIP). Une solution permettant de palier à ce problème consistait à lisser à l'aide d'un flou Gaussien les trois composantes du champ de déformation calculé par SPM2. De cette manière, lors du déplacement des points de maillage, la consultation du champ de déformation par interpolation tri-linéaire produit un résultat plus lisse. Afin de limiter les effets de bords du filtre flou (effets lié à la gestion du noyau de convolution qui déborde de l'image), le lissage est appliqué selon la formule (4.5).

$$T' = (\mathcal{F}^\rho.T_X, \mathcal{F}^\rho.T_Y, \mathcal{F}^\rho.T_Z) \tag{4.5}$$

où T_X, T_Y et T_Z sont des fonctions scalaires de l'espace (a fortiori compatibles avec les opérateurs de traitement d'image) et $\mathcal{F}$ est un filtre flou spécifique à l'objet traité à savoir un champ de déformation (et non une image). Pour définir $\mathcal{F}$, il faut utiliser la déformation neutre

$$\forall(X,Y,Z), I(X,Y,Z) = (X,Y,Z) = (I_X(X,Y,Z), I_Y(X,Y,Z), I_Z(X,Y,Z))$$

et appliquer le flou aux composantes du vecteur déplacement

$$\mathcal{F}^\rho.T_i = I_i + \mathcal{G}^\rho * (T_i - I_i) \tag{4.6}$$

où i est X , Y, ou Z. Cette manière de procéder permet de se conformer aux deux méthodes classiques de traitement des bords lors des convolutions : traiter l'image de façon périodique (ce que nous avons implémenté) ou introduire des pixels supplémentaires d'intensité nulle. Nous obtenons des déformations régulières qui n'engendrent pas de tétrahèdres non-conformes et qui ne comportent pas de déplacements artificiels sur les bords du domaine de l'image. De plus, l'équation (4.6) définit un procédé de filtrage $\mathcal{F}^\rho$ qui laisse invariante la déformation neutre. Le paramètre λ a été fixé à 1.

3. Au cas où on peut démontrer que la solution $\mathcal{T}^\sigma$ varie continuement en fonction du paramètre σ, on peut parler d'une homotopie discrète sur σ.

4.2.3 Performances et résultats

	avant	maintenant
logiciel	matlab + C-Mex	C + shell + perl
architecture	PC et Mac (32 bit)	jade.cines.fr (64 bit)
parallélisme	1 CPU	19 CPU (MPI)
vectorisation	non	oui (compilateur)
temps	6 heures	15 minutes
interface	GUI matlab	terminal
utilisation	1 centaine de clics	1 ligne de commande

FIGURE 4.3 – Gain de productivité obtenu sur le calcul de déformation. Le cas présenté ici correspond aux 19 calculs de déformation nécessaires à l'animation d'un maillage sur un cycle cardiaque.

D'après la figure 4.3, les performances sont satisfaisantes au niveau du portage et de la parallélisation du code de déformation, ($6 \times 60\text{min}/19 \simeq 19\text{min}$). L'interface d'utilisation du processus de déformation a été refaite, de manière à éliminer les opérations manuelles chronophages. Le portage en C des routines de traitement d'images a eu essentiellement deux effets bénéfiques :

1. une accélération du calcul grâce au code binaire et à la vectorisation automatique du compilateur Intel,
2. une utilisation plus économe de la mémoire qui permet aujourd'hui le traitement d'images à haute résolution.

Les calculs optimisés de déformation ont été exploités dans les cas suivants :
- images IRM d'un modèle aortique comportant un anévrisme compliant,
- images IRM d'artères aortes sur des volontaires du CHU Rangueil,
- images scanner (rayons X) dans le cadre de l'étude du ventricule gauche et des coronaires [128].

La méthodologie donne des résultats satisfaisants (mouvements du maillage en accord avec les images médicales) pour les images d'artères aortes et de ventricules gauche, mais pas pour les images de coronaires, dont les mouvements ont trop d'amplitude.

4.3 La chaîne de traitement optimisée

4.3.1 La nouvelle chaîne OCFIA

La figure 4.4 montre le workflow complet de l'information issue de l'imagerie médicale. Toute la partie informatique de la chaîne de traitement OCFIA (en aval de l'acquisition d'images IRM) a bénéficié d'un re-engineering similaire à ce qui a été réalisé dans le cadre du calcul de déformation.

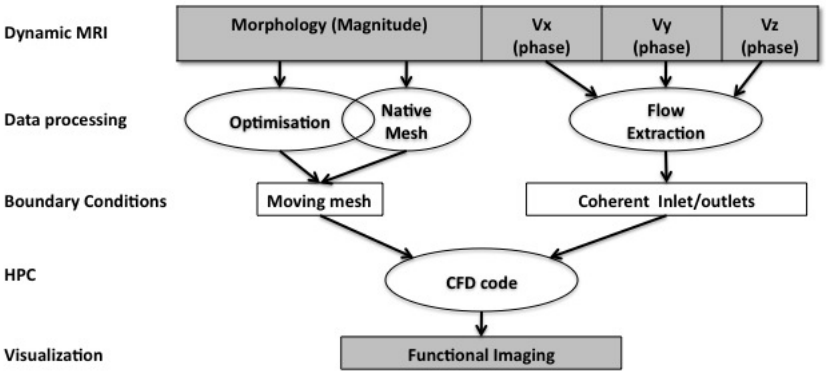


FIGURE 4.4 – Nouveau workflow de l'information issue de l'imagerie médicale pour la création d'images fonctionnelles patients spécifiques.

1. Remise à plat et portage en langage C des opérations de normalisation (uniformiser les résolutions spatiales et les formats d'encodage des pixels) et de re-dimensionnement (recadrer dans un domaine plus petit, re-centré par rapport au centre d'inertie du maillage).
2. Ecriture en langage C des opérations qui modifient les images morphologiques de manière à immobiliser les zones d'entrée-sorties (superposer sur les images des boules sombres, dont les rayons et centres sont déterminées automatiquement grâce à la donnée des points et des différentes surfaces d'entrées et sorties disponible dans le fichier de maillage).
3. Automatisation de la partie chronophage du processus de simulation CFD, qui a consisté à écrire sous forme de script shell l'enchaînement des P phases sur N cycles cardiaques et l'initialisation d'un calcul AVBP (logiciel CFD avec maillage non structuré, CERFACS, Toulouse).
4. Extraction automatique des vitesses dans les régions des entrées et sortie à partir des images d'IRM vélocimétrique, en exploitatant les informations de localisation disponibles dans le fichier de maillage.
5. Automatisation de l'acquisition des conditions limites d'entrée par le logiciel AVBP. Désormais, il n'est plus nécessaire de recompiler le logiciel AVBP pour chaque patient.

6. Amélioration du procédé de génération des images fonctionnelles (passage de la simulation à la visualisation). Les résultats de la CFD ainsi que les images IRM (morphologie et vélocimétrie) sont convertis au format VTK, qui est lisible à l'aide de Paraview (un logiciel de visualisation gratuit Open Source). Ce logiciel permet le calcul de la vorticité et des lignes de courant.
7. Enchaînement automatique des traitements préalables et consécutifs à la CFD et au calcul de déformation. Des langages de script ont servi à relier les différents programmes compilés. Le langage Bash a été choisi en raison de sa disponibilité dans divers environnements UNIX. Le langage Perl, largement répandu sous UNIX et complémentaire au Shell, a servi pour le traitement des données ASCII chaque fois que les fonctionnalités du Shell se sont révélées insuffisantes.

Actuellement, la seule étape qui nécessite l'intervention humaine et l'interaction homme-machine est le calcul des fonctions périodiques d'entrée-sortie. Ces calculs font appel au module CFTools (Curve Fitting) de Matlab. Ils consistent à estimer la meilleure représentation (au sens des moindres carrés) des flux aux entrées et sorties de la zone d'écoulement, sous forme d'une combinaison linéaire de fonctions trigonométriques. C'est l'algorithme de Newton-Gauss qui effectue le calcul. Cette étape nécessite un contrôle visuel pour valider les courbes de flux d'entrée-sorties imposées à la simulation. Les scripts Matlab effectuant ces étapes ont été optimisés de manière à rendre cette étape manuelle la moins chronophage possible.

L'ensemble des développements lociciels est packagé à l'aide d'une procédure d'installation sous UNIX en une ligne de commande. Ces outils fonctionnent sur les machines suivantes : Jade (SGI et Intel Xeon) au CINES, Zeus (IBM + Power4/Power5) au CINES, PC Linux (Ubuntu 32 et 64 bit). Ils complètent le logiciel AVBP au sens où ils relient les données médicales patients spécifiques au calcul CFD.

4.3.2 Gains de performance

	avant	maintenant
logiciel	matlab + C-Mex + AVBP	AVBP + C + shell + perl
architecture	PC, Mac (32 bit), Power4	jade.cines.fr (64 bit)
parallélisme	1+ 24 CPU	160 CPU (MPI)
vectorisation	non	oui (compilateur)
temps	48H-72H	3 H
interface	GUI matlab	terminal
utilisation	plusieurs centaines de clics	quelques ligne de commande

FIGURE 4.5 – Gain de productivité obtenu dans la partie simulation numérique et traitement d'images médicales de la chaîne OCFIA.

La figure 4.5 donne une vue globale sur les gains de performance obtenus grâce

à l'optimisation du workflow. Sachant que la partie acquisition IRM ne dure pas plus d'une heure (temps de préparation du volontaire compris), l'objectif consistant à rendre l'image fonctionnelle cohérente de l'artère aorte accessible en moins de 5 heures, est clairement atteint (la cohérence étant obtenue par la CFD [4]). La totalité des calculs intensifs sont effectués au CINES. Le transfert des données s'effectue via le réseau RENATER.

Il faut garder à l'esprit que temps de restitution d'un calcul CFD tel que AVBP dépend fortement du maillage. Plus le maillage est fin, plus le calcul est long. Or le maillage est une étape manuelle qui reste indispensable à l'heure actuelle. Ces temps de calculs demeurent par conséquent opérateurs dépendants. L'intérêt de développer un code CFD sur maillage cartésien structuré devient clair : passer directement de l'imagerie médicale au calcul CFD sans avoir besoin de créer un maillage.

4.4 Le solveur Poisson de IAS

4.4.1 Introduction

IAS (Immersed Aorta Simulator) est le code CFD sur grille cartésienne invariante, dédié à l'imagerie fonctionnelle patient spécifique. La discrétisation spatiale est de type différences finies, dans le but d'obtenir la plus grande vitesse de calcul avec les schémas les plus simples (le moins possible d'opérations arithmétiques). La gestion des géométries complexes sans maillage, autrement dit sans approche body-fitted, est effectuée par la méthode des frontières immergées [132, 133]. Nous renvoyons à [129] pour une description détaillée du code. Son schéma de discrétisation spatio-temporel conduit à la résolution de 3 problèmes de Poisson 3D stationnaires (une matrice creuse à stocker) par pas de temps, à savoir :

$$\begin{cases} -\Delta p_q = b_q \text{ sur } \Omega_t \text{ et } q = 1, 2, 3 \\ \frac{\partial p_q}{\partial n} = 0 \text{ sur } \partial\Omega_t \end{cases} \tag{4.7}$$

où Ω_t est le domaine de résolution de l'EDP qui est fonction du temps t (domaine borné de $\mathbb{R}^3$), p_q et b_q sont respectivement les champs de pression à calculer et les seconds membres au sein de chaque pas de temps (Runge-Kutta d'ordre 3).

Remarque 4.1 La discrétisation de (4.7) conduit à une matrice singulière à cause des conditions de Neumann homogènes. En pratique le solveur GMRES converge vers la solution malgré la singularité. De plus, les relaxations de Jacobi dans multigrille

4. La cohérence, c'est ce que les mesures de vélocimétrie IRM n'ont pas. Les images vélocimétriques sont parasités par des bruits, des distorsions spatiales, des discontinuités temporelles. Le champ de vitesse que fournit la mesure physique n'est pas le fruit d'un processus ordonné, déterministe, régi par des lois mathématiques. Le calcul CFD fournit une dynamique détaillée au milimètre près du fluide sanguin obtenu par un procédé calculatoire qui repose sur des principes de physique et de mathématiques garantissant la cohérence et dans une certaine mesure, le réalisme.

ne divergent pas lors du préconditionnement. En fait, la matrice de discrétisation n'est pas diagonale dominante irréductible à cause d'un seul détail : les lignes sont diagonales dominantes, mais aucune ligne ne possède la digonale dominance stricte. Nous avons probablement affaire à une matrice de Jacobi associée ayant un rayon spectral égal à 1 [68].

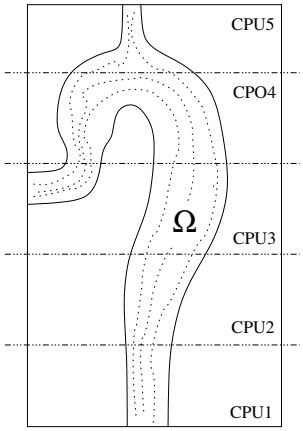


FIGURE 4.6 – Vue en coupe du pavé contenant la zone d'écoulement Ω et son découpage en sous-domaines.

La grille cartésienne invariante est définie à partir du pavé qui contient la région de l'artère aorte. Ce cadre cartésien conduit à une implémentation très simple des schémas de discrétisation en des dérivées spatiales en différences finies. La parallélisation de IAS repose sur un découpage du pavé en tranches (voir figure 4.6). C'est donc une méthode de parallélisation par sous-domaines où chaque CPU ne communique qu'avec au plus 2 voisins. Les calculs flottants ont lieu uniquement au niveau de la zone d'écoulement (Ω_t). Toutefois, ce découpage, qui est le fruit d'un compromis entre efficacité et simplicité, n'est pas optimal sur le plan numérique.

Face à la complexité intrinsèque de l'élaboration d'un code CFD "from scratch", nous avons pris le parti de privilégier l'obtention du linear scaling entre le nombre de points de discrétisation et le temps de calcul. Compte tenu des faits suivants :

- cette matrice creuse varie en fonction du temps, notamment à cause du mouvement des parois qui est gérée sur grille invariante par la méthode des frontières immergées (Immersed Boundary),
- la stationnarité du problème de Poisson 3D engendre des matrices mal conditionnées,
- les conditions de limites de type Neumann homogène sont imposées pour le moment, ce qui rend la matrice semi-définie positive,

nous avons choisi d'utiliser une méthode itérative (GMRES - implémentation du CERFACS, Toulouse) couplée à un préconditionneur multigrille. Ainsi, il n'est pas

question de refactoriser la matrice au cours du temps (à chaque changement de domaine).

De plus, seules les composantes des vecteurs qui correspondent à des points à l'intérieur de Ω sont stockées en mémoire. Cette technique à base de matrice et vecteur creux permet d'accélérer significativement les opérations d'algèbre linéaires associées à la résolution de l'équation de Poisson. Par conséquent, c'est l'approche algébrique de la méthode multigrille (AMG) qui a été utilisée. Il est donc nécessaire de recalculer les matrices sur grilles grossières au cours du temps, ce qui reste plus rapide que la refactorisation (même incomplète) de la matrice de Poisson.

D'autre part, il existe un implémentation open source du multigrille algébrique qui a fait ses preuves : boomerAMG - Lawrence Livermore National Laboratory. Cette implémentation fait partie de la suite de préconditionneurs parallèles HYPRE. La version stable 2.0.0 a été utilisée dans IAS.

Des tests de performance ont été réalisés pour une géométrie réaliste (issue du modèle d'artère aorte en silicone photographiée par IRM - ce sont donc des tests qui correspondent à des réalités pratiques du processus de création des images fonctionnelles patient spécifiques de l'artère aorte) avec parois statiques. Le linear scaling a été effectivement obtenu sur une plage d'utilisation pré-établie (1 à 10 millions de points dans Ω, 64 CPU au maximum).

4.4.2 Protocole de test

Le run de référence est une géométrie d'artère aorte (issue du modèle en silicone), avec un état initial aléatoire (voir figure 4.6).

Ensuite, à l'aide d'un interpolateur, on engendre des runs de plus en plus raffinés en augmentant les nombres de points en x, y, z d'un facteur 1.2. Les données interpolées sont la géométrie et l'état initial. Ce procédé est appliqué récursivement. Cela donne des systmes linéaires de tailles : 726827, 1232355, 2092196, 3579396, 6116733, 10521175, soit 0.7 à 10 millions.

Ces systèmes sont résolus avec 16, 24, 32, 40, 48, 56, 64 processeurs sur jade.cines.fr, ce qui donne $8 \times 6 = 48$ configurations pour chaque variante de AMG. Dans chaque run, il y a 3 résolutions de systèmes linéaires. On ne comptabilise que le temps de restitution du premier système pour ces mesures. Une régression linéaire du temps de restitution en fonction de la taille du vecteur, sur échelles logarithmiques, permet de calculer les facteurs de scaling. Le scaling linéaire correspond à un facteur qui vaut 1.

4.4.3 Paramétrage de boomerAMG

La résolution avec le préconditionneur boomerAMG n'a pas réussi du premier coup. il a fallu prendre en compte la singularité de la matrice, directement liée aux conditions de Neumann homogènes :

remplacer l'élimination gaussienne du dernier niveau de grille (la plus grossière) par une itération de la méthode de Jacobi.

Ce réglage est fondamental, car l'élimination gaussienne est incompatible avec la singularité (division par zéro). Avec ceci, tous les schémas de restrictions (CLJP, RS, Falgout, PMIS, HMIS) sont utilisables.

L'implémentation HYPRE du multigrille algébrique possède une quarantaine de paramètres. Il est hors de question de laisser autant de degrés de libertés, compte-tenu des risques d'instabilités. Cinq paramètres pertinents sont réglables sans prise de risque au niveau stabilité, par un utilisateur qui n'est pas nécessairement numéricien, et qui cherche à accélérer le calcul :

1. le nombre de d'itérations de boomerAMG,
2. le nombre maximum de niveaux de grilles,
3. le seuil α qui contrôle la force de la restriction [134],
4. le nombre de niveaux de restriction agressive,
5. le schéma de restriction.

Accessoirement, il est possible d'activer l'affichage d'informations pour le déboguage.

4.4.4 Résultats et interprétations

Tests sur géométrie complexe

Les valeurs des paramètres entiers et réels de boomerAMG ont été fixés afin de simplifier : nombre d'itération = 1, niveaux maximum = 25 (le problème le plus difficile de la série de tests en utilise moins), seuil $\alpha = 0.15$ (déterminé empiriquement sur des petits problèmes), niveaux de restriction agressive = 0 Le but étant de tester d'abord les schémas de restriction :

1. CLJP (Cleary - Luby - Jones - Plassman), tableau 4.1,
2. RS (Ruge - Stueben) 3 passes, tableau 4.2,
3. Falgout, tableau 4.3,
4. PMIS (parallel modified independent set), tableau 4.4,
5. HMIS (hybrid modified independent set), tableau 4.5.

Quelques faits notables :
- Le schéma de restriction PMIS conduit sur certains cas à des matrices dont certaines diagonales sont nulles. Cela n'empêche pas la convergence du solveur.
- Le phénomène précédent est également présent dans le cas de la restriction HMIS.

À l'aune des temps de calculs et des scaling factors, les deux meilleurs schémas de restriction sont Falgout et RS 3 passes. Le schéma RS 3 passes a l'avantage de la scalabilité, tandis que le schéma Falgout a celui du temps de calcul. Les méthodes

	ncpu						
nvec	16	24	32	40	48	56	64
726827	3.4	3.5	2.3	1.7	1.5	1.2	1.1
1232355	6.4	7.1	5.1	4.7	3.3	3.0	2.6
2092196	13	13	10	9.1	6.5	6.2	5.2
3579396	29	26	18	16	12	11	10
6116733	63	44	39	33	25	22	19
10521175	100	102	73	59	43	43	36
scal.	1.30	1.22	1.27	1.28	1.25	1.30	1.28

TABLE 4.1 – Temps de calcul (sec.) et facteurs de scaling de l'approche GMRES préconditionné par boomerAMG, restriction CLJP

	ncpu						
nvec	16	24	32	40	48	56	64
726827	1.6	1.7	1.1	1.2	1.2	1.2	1.4
1232355	3.1	2.9	2.5	2.3	2.1	1.8	1.8
2092196	5.7	5.3	4.9	4.2	3.4	3.6	3.9
3579396	8.9	8.7	8.8	8.3	6.8	6.4	6.6
6116733	17	17	15	12	11	9.9	11
10521175	29	28	26	25	17	20	19
scal.	1.07	1.07	1.15	1.10	1.02	1.04	1.02

TABLE 4.2 – Temps de calcul (sec.) et facteurs de scaling de l'approche GMRES préconditionné par boomerAMG, restriction RS 3 passes

PMIS et CLJP sont celles qui passent le moins bien à l'échelle et aussi les plus lentes. La méthode HMIS n'améliore pas les deux dernières.

Les efficacités parallèles de ces calculs AMG sont très faibles, compte-tenu de la méthode de découpage de Ω. De plus, il faut garder en vue le fait que ce solveur Poisson n'est qu'une partie d'un code CFD. Pour mesurer l'efficacité globale, nous effectuons un test sur 10 pas de temps pour les problème à 1232355 (tableau 4.6) et 2092196 (tableau 4.7) millions d'inconnues, avec le schéma Falgout. Nous constatons que la partie hors-solveur du code CFD manque également d'efficacité parallèle. La conclusion de cette expérience de calcul parallèle est la suivante : même en différences finies sur maillage structuré, l'utilisation des structures de données propres aux méthodes sur maillage non structurés de type éléments finis est indispensable pour obtenir de la performance. L'usage de partitionneurs de graphes et maillages tels que METIS s'avère important.

nvec	ncpu						
	16	24	32	40	48	56	64
726827	1.8	1.2	1.1	1.1	0.8	0.7	0.9
1232355	4.2	2.5	2.1	2.2	1.8	1.8	1.5
2092196	4.7	4.6	4.2	3.4	2.7	2.8	2.9
3579396	11	9.3	7.8	6.6	6.9	6.8	5.8
6116733	17	21	15	12	11	9.3	9.9
10521175	34	33	25	25	19	22	18
scal.	1.05	1.25	1.17	1.14	1.16	1.21	1.14

TABLE 4.3 – Temps de calcul (sec.) et facteurs de scaling de l'approche GMRES préconditionné par boomerAMG, restriction Falgout

nvec	ncpu						
	16	24	32	40	48	56	64
726827	4.9	5.2	2.7	2.3	1.6	1.3	1.1
1232355	14	10	6.1	5.7	4.3	4.5	3.0
2092196	16	21	14	13	10	7.8	7.3
3579396	36	35	26	21	18	16	11
6116733	68	67	50	41	35	30	29
10521175	142	137	102	81	70	64	53
scal.	1.19	1.20	1.33	1.28	1.36	1.38	1.41

TABLE 4.4 – Temps de calcul (sec.) et facteurs de scaling de l'approche GMRES préconditionné par boomerAMG, restriction PMIS

Tests supplémentaires sur canal droit

Pour confirmer la scalabilité, un test du "canal droit qui s'allonge" est effectué pour le schéma RS 3 passes (figure 4.7), sur un seul pas de temps. Cette fois-ci, les résultats des 3 systèmes linéaires sont reportés sur le graphique. Ce test consiste à augmenter le nombre d'inconnues et de CPU simultanément, de sorte que chaque processeur traite le même nombre d'inconnues (weak scaling). Le choix de la géométrie cylindrique permet de statisfaire aisément les contraintes du protocole. Pour ce test, le nombre d'inconnues initial est 26600.

Le weak scaling de l'approche AMG n'est pas idéal (la courbe de temps verte dans la figure 4.7 n'est pas horizontale), mais il est largement supérieur au weak scaling du solveur témoin (FGMRES préconditionné à droite par GMRES, lui même préconditionné par block-ILU). Ce solveur témoin fut le solveur originel qui a servi de base pour démarrer IAS. Le préconditionnement à droite est réalisé via l'imbrication du procédé d'Arnoldi. Ce procédé interne peut lui-même bénéficier d'un

nvec	ncpu						
	16	24	32	40	48	56	64
726827	3.4	2.9	2.3	2.1	1.4	1.3	1.1
1232355	6.7	6.8	5.1	3.9	3.6	3.7	2.7
2092196	11	12	9.9	8.4	6.7	7.1	4.7
3579396	22	21	16	15	10	10	9.7
6116733	45	38	30	27	24	21	18
10521175	69	78	62	58	43	50	32
scal.	1.14	1.18	1.19	1.23	1.23	1.28	1.23

TABLE 4.5 – Temps de calcul (sec.) et facteurs de scaling de l'approche GMRES préconditionné par boomerAMG, restriction HMIS

ncpu	16	24	32	40	48	56	64
solveur	81	75	54	55	48	46	38
hors-solveur	72	63	56	57	62	56	59
total	153	138	111	112	110	103	98
ratio solveur/total	0.53	0.54	0.48	0.49	0.43	0.45	0.39

TABLE 4.6 – Temps de calcul (sec.) sur 10 pas de temps pour environ 1 Million d'inconnues (pression), restriction Falgout

préconditionneur constant de type block-ILU. Le procédé d'Arnoldi le plus externe (FGMRES) doit gérer le fait que le préconditionneur interne (GMRES + block-ILU) n'est pas constant car le changement de second membre change l'espace de Krylov, d'où la nécessité d'utiliser la version flexible de GMRES au niveau externe.

Nous pouvons constater que pour un nombre de CPU supérieur à 32, l'augmentation du temps de calcul de l'approche AMG est corrélée à l'augmentation du nombre d'itérations nécesaires pour converger. Ce constat permet d'affirmer que dans un cas où l'équilibrage de charge est parfait, l'efficacité parallèle est bonne. Ceci confirme une fois de plus l'importance de l'équilibrage de charge.

ncpu	16	24	32	40	48	56	64
solveur	118	123	105	91	70	77	64
hors-solveur	133	109	97	101	93	101	114
total	252	232	202	193	163	178	178
ratio solveur/total	0.47	0.53	0.51	0.47	0.42	0.43	0.36

TABLE 4.7 – Temps de calcul (sec.) sur 10 pas de temps pour environ 2 Million d'inconnues (pression), restriction Falgout

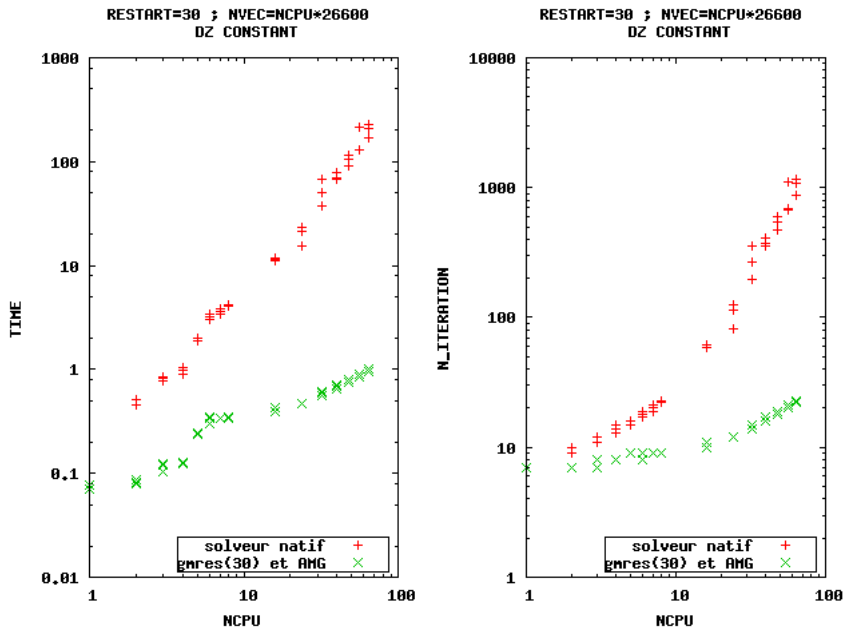


FIGURE 4.7 – Comparaison de la scalabilité entre solveur témoin et GMRES préconditionné par AMG (restriction RS 3 passes) sur le test du canal droit

4.4.5 Comparaison avec le préconditionnement ILU

A titre indicatif, voici un compte-rendu sur les problèmes de scalabilités rencontrés avec le solveur témoin décrit précédemment. L'implémentation du préconditionneur ILU sur chaque sous-domaine est issue de la bibliothèque SPARSKIT. Les solveurs FGMRES et GMRES qui s'imbriquent l'un dans l'autre sont issus du CERFACS. Nous appelons simplement cette méthode "block-ILU" dans la suite. Ces données confirment la pertinence de l'approche multigrille. Avec le même protocole, on fera varier en plus le paramètre fill-in de la méthode ILU pour chaque configuration. Les valeurs sont : 0, 2, 3, 4, 5, 6, 7, sachant que la valeur 0 correspond à l'absence de préconditionneur. Au total, cela fait $48 \times 7 = 336$ runs. La machine utilisée est jade.cines.fr.

Le tableau 4.8 contient les scaling factors. Les éléments nuls de ce tableau correspondent aux séries de données dans lesquels un des runs n'a pas pu converger dans le temps imparti. En particulier, le run à 10.52×10^6 inconnues n'a pas pu aboutir sur 8 processeurs en l'absence de factorisation block-ILU.

Le tableau 4.9 contient les temps de restitutions sur 64 processeurs. Ces valeurs permettent de voir l'ordre de grandeur du gain de performance en fonction de la précision des factorisations block-ILU et de la taille des vecteurs.

ncpu	fill-in						
	0	2	3	4	5	6	7
8	0	1.7139	1.7080	1.6288	1.6403	1.7533	1.6938
16	1.9402	1.8306	1.8254	1.8154	1.8020	1.7981	1.6786
24	1.9653	1.7421	1.8179	1.8227	1.8028	1.8337	1.7768
32	2.0181	1.9056	1.9008	1.8479	1.8231	1.8362	1.7667
40	2.0849	1.8455	1.8910	1.8635	1.8078	1.8248	1.7820
48	2.1619	2.0614	1.9487	1.9153	1.8766	1.9419	1.8245
56	2.1634	2.0130	1.9896	1.9545	1.8974	1.8187	1.8342
64	2.2257	1.9472	1.9722	1.9591	1.8850	1.8813	1.8713

TABLE 4.8 – Scaling factors pour le préconditionnement par block-ILU

fill-in	nvec					
	726827	1232355	2092196	3579396	6116733	10521175
0	18.7489	74.4953	313.676	868.052	2483.17	7730.26
2	28.7399	67.0496	220.059	626.95	1598.16	5067.24
3	13.0792	34.0089	143.456	368.804	957.475	2345.43
4	8.26494	37.1109	75.1093	240.7	766.292	1625.45
5	8.41591	32.1514	104.403	214.038	572.613	1499.98
6	8.67186	29.3002	97.4955	183.205	511.632	1567.78
7	7.79071	22.3408	74.1054	240.875	413.581	1175.01

TABLE 4.9 – Temps de restitution (sec.) avec 64 processeurs pour le préconditionnement par block-ILU

Interprétations

En analysant le tableau 4.8 ligne par ligne, nous pouvons conclure que la précision de la factorisation block-ILU améliore le scale factor, cette précision étant déterminée par la valeur de fill-in. Toutefois, cette approche est limitée par la quantité de mémoire disponible et le poids de la résolution ILU dans chaque sous-domaine. Le fill-in n'a pas pu être poussé au delà de 7 à cause du manque de mémoire sur les runs à 8 processeurs. D'autres essais ont montré que l'amélioration du scaling factor par block-ILU tient à la réduction du nombre d'itération du solveur externe FGMRES. Lorsque la valeur du fill-in est trop élevée, les temps de restitution ne sont plus représentatifs des nombres d'itérations. Une analyse colonne par colonne du tableau 4.8 montre que le changement de découpage n'a pas d'impact négatif sur le scaling factor. Le tableau 4.9 montre par ailleurs que la valeur du fill-in n'a pas besoin de dépasser 5. Enfin, nous pouvons conclure que la méthode block-ILU ne permet pas d'obtenir le linear scaling pour l'équation de Poisson.

4.5 Synthèse

Nous pouvons, à la lumière des travaux d'optimisation du procédé OCFIA, affirmer que la simulation CFD patient spécifique sans interaction fluide-structure est un calcul relativement léger. Les temps de calculs relativement courts (inférieurs à 5 heures) montrent que ce procédé d'imagerie fonctionnelle est utilisable dans un environnement où la productivité est décisive (*"on abandonne si c'est trop long"*).

Il n'est pas approprié de s'étendre ici sur la validation scientifique des images fonctionnelles issues du procédé OCFIA (validation in-vitro, in-vivo, applications potentielles). Bien que les résultats in-vitro actuels soient très satisfaisants, il reste encore beaucoup de travail avant d'atteindre le niveau de validation in-vivo. En dehors de toute considération d'ordre médicale, le fait de disposer d'un procédé de simulation rapide (résultats dans journée ou la nuit) et non-chronophage (peu de temps humain) représente en soi un atout majeur dans le travail de recherche clinique. La disponibilité de l'outil automatisé et optimisé permettra a terme d'accélérer des travaux de validation clinique où le nombre de simulations nécessaires se compte par centaines.

La synthèse sur le plan de la conception de logiciels pour le calcul intensif est la suivante :

- La performance au sens informatique est un élément qu'il faut penser en amont. La résolution de problèmes d'efficacité parallèle a posteriori (ou séquentiels, avec la problématique de minimisation des défauts de caches) peut parfois conduire au recodage complet des structures de données et par conséquent des algorithmes, sans oublier la conception de nouveaux tests de validation. C'est une situation qu'il faut essayer d'éviter.
- La performance au sens numérique (choix des algorithmes) peut aussi entraîner d'importantes modifications sur un code. Les librairies imposent de fait leurs propres structures de données (stockage des matrices creuses par exemple). Adopter une librairie (SPARSKIT, HYPRE par exemple) revient donc à accepter une contrainte forte sur la conception de son propre code. C'est le genre de chose qu'il faut aussi anticiper en amont.
- Le fait de penser un workflow complexe comme un seul programme exécutable est possible, mais en pratique très compliqué à mettre en œuvre. L'utilisation de langages de script rudimentaires tels que Bash ou Perl pour l'implémentation de l'enchaînement des tâches, le traitements des fichiers d'entrée et sortie, est un succès. La validation individuelle de chaque tâche est grandement facilitée.

Notons que le projet IAS a été abandonné : suite à la thèse de Bruno Tayllamin, le recodage du logiciel n'a pas été envisagé. L'aptitude de la communauté médicale à manipuler les maillages non structurés a été largement sous-estimée dès le départ. Nous avons assisté à un retour vers les méthodes body-fitted, qui représentent intrinsèquement mieux les conditions limites aux parois que les méthodes à frontières immergées sur maillages structurés cartésiens. De plus, des tests avec la méthode

Lattice Boltzmann (le maillage triangulaire des contours du domaine suffit) ont révélé que ce type d'algorithme répond facilement à une demande de la communauté médicale qui souhaite concilier les échelles microscopiques et macroscopiques.

Compte-tenu du coût élevé des investissements pour maintenir des infrastructures de calcul haute performance à la pointe, et du contexte concurrentiel en sciences, il s'avère que l'optimalité d'un code parallèle est un critère important. Toutefois, il est essentiel d'être en phase avec la demande des utilisateurs. La production d'un code optimisé ne doit pas prendre plus d'importance que la validation de l'adéquation d'une méthode vis à vis de la demande. Ceci dit, l'anticipation des problèmes de performances est essentiel au début du cycle de développement.

Dans cette synthèse, l'expérience acquise lors de contrats de prestations de services pour le compte de clients de la société ASA (Advanced Solutions Accelrator) a joué un rôle important. Toutefois, il est impossible de mentionner ces projets dans le cadre de ce mémoire pour des raisons liées à des engagements de confidentialité.

Chapitre 5

Conclusion générale

L'ensemble des travaux de recherche et de développements logiciels effectués depuis 2008 permet de constater plusieurs points importants.

Minimisation des communications La puissance de calcul augmente plus rapidement que la vitesse de transmission des données entre nœuds de calcul. Les processeurs GPU et les nouveaux co-processeurs Xeon-Phi confirment fortement cette tendance. Cette situation favorise l'approche asynchrone, mais il faut nuancer le point vue qui consiste à affirmer que l'asynchronisme permet de se passer d'équilibrage de charge. Sur les architectures actuelles l'asynchronisme peut faire face aux problèmes d'hétérogénéité de puissance des nœuds et de bande passante entre les nœuds. Les études effectuées sur Grid'5000 permettent d'affirmer cela. En revanche, l'asynchronisme ne permet pas de palier aux problèmes d'efficacité parallèle liés au surcoût des communications, dont la proportion par rapport au poids des calculs ira en grandissant. L'utilisation d'outils classiques d'équilibrage de charge est important dans le contexte asynchrone, car ils résolvent par défaut le problème de minimisation des communications. Autrement dit, la suppression des temps d'inactivité, conséquences des échanges de messages synchrones, n'est pas suffisant. Il faut aussi minimiser les poids des communications. L'équilibrage du nombre d'inconnues dans chaque sous-domaine favorisera aussi bien l'approche synchrone que asynchrone ; les déséquilibres de charges liés au matériel et à l'environnement dynamique persisteront.

Considération du scaling factor Dans le cadre des solveurs itératifs, l'étude de la relation du type $T = \mathcal{O}(n^s)$ entre le temps de restitution (T) et le nombre d'inconnues (n) doit être pris en considération (s est le scaling factor, voir section 4.4). C'est une approche empirique et expérimentale qui se rapproche de la notion de complexité algorithmique et qui est plus parlante dans la pratique que les notions classiques de convergence linéaire ou quadratique des algorithmes itératifs. En effet, la notion classique de convergence ne permet pas d'expliciter le nombre d'opérations nécessaires pour terminer un calcul, en atteignant une précision donnée, en fonction

uniquement du nombre d'inconnues. En revanche, on peut facilement construire la courbe du temps de restitution en fonction de ce nombre, qu'on fait varier en changeant par exemple la finesse d'un maillage. On parle donc de mesures expérimentales pour une configuration physique fixée. Le scaling factor permet de quantifier de façon empirique la capacité d'un code à supporter des modèles de plus en plus gourmands. En aucun cas, le scaling factor permet de caractériser la performance d'un algorithme de manière générale, contrairement à la notion de convergence classique. Cependant, il a le mérite de représenter assez fidèlement ce qui pourrait se passer dans la pratique de l'utilisation d'un code, dans un environnement bien délimité. L'exprérience industrielle acquise au sein d'ASA a une influence importante sur cette conclusion. Les utilisateurs ont tendance à effectuer quelques essais avec relativement peu d'inconnues avant de se lancer dans des tests plus lourds. Un algorithme ayant un scaling factor trop élevé va exiger trop de puissance de calcul supplémentaire lors du passage au modèle lourd. Le choix d'un algorithme *qui passe bien à échelle supérieure* demeure le levier agissant le mieux sur la performance.

Rôle de l'asynchronisme Les expérimentations, effectuées dans le cadre de comparaisons entre les versions synchrones et asynchrones d'un même algorithme, ont toujours montré que les performances des version asynchrones ne s'effondrent pas lorsque celles des versions synchrones s'effondrent à partir d'un certain nombre de nœuds de calcul. L'approche asynchrone intervient naturellement pour palier aux problèmes de scalabilité en fonction du nombre de nœuds de calcul. En effet, lorsque les modèles deviennent plus lourds, les utilisateurs ont tendance à employer plus de moyens de calculs, en prenant le risque de faire effondrer l'efficacité en parallèle. Dans ce contexte, une méthode asynchrone est définitivement utile si elle permet de restaurer les bonnes performances en parallèle, tout en offrant la possibilité de passer correctement à l'échelle supérieure. On parle alors de la notion de weak scaling. Le chapitre 5 de la thèse [112] a eu une influence sur cette conclusion. Il montre qu'une méthode de décomposition de domaine avec communication asynchrone exploitant GMRES sur chaque sous-domaine peut surpasser GMRES sur un domaine non décomposé. On peut donc espérer pouvoir mettre au point une méthode itérative asynchrone ayant un bon comportement en terme de weak scaling.

Persectives

L'objectif pour la poursuite des travaux de recherche et de développement consistera à implémenter des solveurs asynchrones attractifs pour le monde de l'industrie et de la recherche. Les pistes qui seront suivies sont notamment :

- l'adoption de structures de données propres aux maillages non structurés,
- l'adoption d'un test d'arrêt robuste et efficace,
- la minimisation du poids des communications,
- l'équilibrage du nombre d'inconnues par sous-domaine,

– l'adoption de méthodes de sous-domaines performantes [31, 32, 135],
– la prise en considération du scaling factor,
– la prise en considération du weak scaling.

En plus des aspects liés à la performance et aux architectures de calculateurs parallèles, un solveur asynchrone attractif doit pouvoir être facilement utilisable dans des procédés aussi complexes que la chaîne de traitement OCFIA décrite dans le chapitre 4. Compte-tenu de la variété des modèles pour lesquels l'approche asynchrone peut convenir, et du développement du Grid et Cloud Computing, les utilisateurs potentiels sont nombreux. Ceci dit, il est évident que l'approche asynchrone ne peut pas convenir à n'importe quel contexte. Les choix algorithmiques sont essentiellement guidés en amont par les aspects scientifiques du domaine de l'application, qui dépassent le domaine de compétence du numéricien. Par conséquent, il ne faut pas sous-estimer l'utilité des méthodes qui équilibrent dynamiquement la charge en fonction de la puissance des nœuds et du réseau de communication.

Bibliographie

[1] D. Chazan and W. Miranker. Chaotic relaxation. *Linear Algebra Appl.*, 2 :199–222, 1969.

[2] J.C. Miellou. Algorithmes de relaxation chaotique à retards. *RAIRO*, 1 :55–82, 1975.

[3] G.M. Baudet. Asynchronous iterative methods for multiprocessor. *J. Assoc. Comput. Mach.*, 2 :226–244, 1978.

[4] F. Robert. *Étude et utilisation de normes vectorielles en analyse numérique linéaire*. Thèse de Doctorat ès Sciences, Grenoble, 1968.

[5] F. Robert. Contraction en norme vectorielle : convergence d'itérations chaotiques. *Linear algebra and its applications*, 13 :19–35, 1975.

[6] M. Charnay. *Itérations chaotiques sur un produit d'espaces métriques*. Thèse de Doctorat, Université Claude Bernard, Lyon, 1975.

[7] 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.

[8] 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.

[9] 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.

[10] M.N. El Tarazi. Some convergence results for asynchronous algorithms. *Num. Math.*, 39 :325–340, 1982.

[11] J.C. Miellou. Itérations chaotiques à retards, étude de la convergence dans le cas d'espaces partiellement ordonnés. *CRAS Paris*, 280 :233–236, 1975.

[12] C. Jacquemard. *Contribution à l'étude d'algorithmes de relaxation à convergence monotone*. Thèse de Doctorat, Université de Besançon, 1977.

[13] M.N. El Tarazi. Algorithmes mixtes asynchrones. Étude de la convergence monotone. *Num. Math.*, 44 :363–369, 1984.

[14] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation, Numerical Methods*. Prentice Hall, Englewood Cliffs N.J., 1989.

[15] J.C. Strikwerda. A probabilistic analysis of asynchronous iteration. *Linear Algebra and its Application*, 349 :125–154, 2002.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] J.C. Miellou and P. Spitéri. Un critère de convergence pour des méthodes générales de point fixe. *M2AN*, 19(4) :645–669, 1985.

[21] J.C. Miellou. *Opérateurs para-monotones*. Thèse de Doctorat ès Sciences, I.M.A.G., Université de Grenoble, 1970.

[22] P. Comte, J.C. Miellou, and P. Spitéri. La notion de H-accrétivité, applications. *CRAS série A*, 283 :655–658, 1976.

[23] P. Benilan. *Equations d'évolution dans un espace de Banach quelconque et applications*. Thèse de Doctorat ès Sciences, Orsay, 1972.

[24] V. Barbu. *Non linear semi-groups and differential equations in Banach spaces*. Noordhoff international publishing, Gröningen, 1976.

[25] L. Giraud and P. Spitéri. Résolution parallèle de problèmes aux limites non linéaires. *M2AN*, 25 :73–100, 1991.

[26] P. Spitéri. A new characterization of M-matrices and H-matrices. *BIT Numerical Mathematics*, 43 :1019–1032, 2003.

[27] K.H. Hoffman and J. Zou. Parallel efficiency of domain decomposition methods. *Parallel Computing*, 19 :1375–1391, 1993.

[28] 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.

[29] A .Frommer, H. Schwandt, and D. Szyld. Asynchronous weighted additive Schwarz methods. *Electronic Transactions on Numerical Analysis*, 5 :48–61, 1997.

[30] A. Frommer and D. Szyld. Weighted max norms, splittings and overlapping additive Schwarz iterations. *Numer. Math.*, 83 :259–278, 1999.

[31] M.J. Gander. Optimized Schwarz methods. *SIAM Journal of Numerical Analysis*, pages 15–28, 2006.

[32] 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.

[33] D.J. Evans and W. Deren. An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations. *Parallel Computing*, 17 :165–180, 1991.

[34] P. Spitéri. Parallel asynchronous algorithms for solving boundary value problems. In M. Cosnard et al., editor, *Parallel Algorithms*, pages 73–84. North-Holland, 1986.

[35] S. Benjelloun, P. Spitéri, and G. Authié. Parallel algorithms for solving the obstacle problem. *Computational Mechanics Publ., Springer-Verlag*, 2 :275–281, 1989.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] J. Arnal, V. Migallón, and J. Penadés. Newton two stage paralle iterative methods for non linear systems. *BIT Numerical Mathematics*, 43 :849–861, 2003.

[45] M. Chau, R. Couturier, J. Bahi, and P. Spitéri. Parallel solution of the obstacle problem in grid environments. *International Journal of High Performance Computing Applications*, 25(4) :488–495, 2011.

[46] 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.

[47] E. Chajakis and S.A. Zenios. Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization. *Parallel Computing*, 17 :873–894, 1991.

[48] D. El Baz. Asynchronous implementation of relaxation and gradient algorithms for convex network flow problems. *Parallel Computing*, 19 :1019–1028, 1993.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] J. Bahi, J.C. Miellou, and K. Rhofir. Asynchronous multisplitting methods for nonlinear fixed point problems. *Numerical Algorithms*, 15 :315–345, 1997.

[58] A. Frommer and D. Szyld. Asynchronous iterations with flexible communication for linear systems. *Calculateurs Parallèles, Réseaux et Systèmes Répartis*, 10 :421–429, 1998.

[59] 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.

[60] A. Frommer and D. Szyld. On asynchronous iterations. *Journal of Computational and Applied Mathematics*, 123 :201–216, 2000.

[61] 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.

[62] 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.

[63] 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.

[64] A. Uresin and M. Dubois. Sufficient conditions for the convergence of asynchronous iterations. *Parallel Computing*, 10 :83–92, 1989.

[65] A. Uresin and M. Dubois. Parallel asynchronous algorithms for discrete data. *Journal of the association for computing machinery*, 37(3) :558–606, 1990.

[66] 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.

[67] D. Mitra. Asynchronous relaxations for the numerical solution of differential equations by parallel processors. *SIAM J. Sci. Stat. Comput.*, 8 :43–58, 1987.

[68] 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.

[69] M. Chau, C. Tauber, and P. Spitéri. Parallel schwarz alternating methods for anisotropic diffusion of speckled medical images. *Numerical Algorithms*, 51(1) :85–114, 2009.

[70] C. Tauber, S. Stute, M. Chau, P. Spitéri, S. Chalon, D. Guilloteau, and I. Buvat. Spatio-temporal diffusion of dynamic pet images. *Physics in Medicine and Biology*, 56(20) :6583, 2011.

[71] T. Garcia, M. Chau, T.T. Nguyen, D. El-Baz, and P. Spitéri. Asynchronous peer-to-peer distributed computing for financial applications. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, IPDPSW '11, pages 1458–1466, Washington, DC, USA, 2011. IEEE Computer Society.

[72] T. Garcia, M. Chau, and P. Spitéri. Synchronous and asynchronous distributed computing for financial option pricing. In B. Murgante et al., editor, *Computational Science and Its Applications - ICCSA 2011*, volume 6783 of *Lecture Notes in Computer Science*, pages 664–679. Springer Berlin Heidelberg, 2011.

[73] M. Chau, R. Couturier, J. Bahi, and P. Spitéri. Asynchronous grid computation for american options derivatives. *Advances in Engineering Software*, online :10.1016/j.advengsoft.2012.06.005, 2012.

[74] L.Ziane Khodja, M. Chau, R. Couturier, J. Bahi, and P. Spitéri. Parallel solution of american option derivatives on gpu clusters. Accepted, Computers and Mathematics with Applications, 2013.

[75] M. Chau, P. Spitéri, and H.C. Boisson. Parallel numerical simulation for the coupled problem of continuous flow electrophoresis. *International Journal for Numerical Methods in Fluids*, 55(10) :945–963, 2007.

[76] M. Chau, P. Spitéri, R. Guivarch, and H.C. Boisson. Parallel asynchronous iterations for the solution of a 3d continuous flow electrophoresis problem. *Computers & Fluids*, 37(9) :1126 – 1137, 2008.

[77] T. Garcia, M. Chau, and P. Spitéri. Computation of protein separation using a grid environment. In B.H.V. Topping P. Iványi, editor, *The Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG'11)*, Ajaccio, Corsica, France, 2011. Civil-Comp Press, Stirlingshire, UK.

[78] M. Chau, T. Garcia, and P. Spitéri. Proteins separation in distributed environment computation. In B. Murgante et al., editor, *Computational Science and Its Applications - ICCSA 2011*, volume 6783 of *Lecture Notes in Computer Science*, pages 664–679. Springer Berlin Heidelberg, 2011.

[79] M. Chau, T. Garcia, and P. Spitéri. Asynchronous grid computing for the simulation of the 3D electrophoresis coupled problem. *Advances in Engineering Software*, online :10.1016/j.advengsoft.2012.11.010, 2012.

[80] M. Chau, D. El Baz, R. Guivarch, and P. Spitéri. MPI implementation of parallel subdomain methods for linear and nonlinear convection-diffusion problems. *J. Parallel Distrib. Comput.*, 67(5) :581–591, 2007.

[81] T. T. Nguyen, D. El Baz, P. Spitéri, G. Jourjon, and M. Chau. High performance peer-to-peer distributed computing with application to obstacle problem. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium*, pages 1–8, 2010.

[82] M. Chau, R. Couturier, J. Bahi, and P. Spitéri. Parallel solution of the sequence of obstacle problems in a grid environment. In B.H.V. Topping P. Iványi, editor, *The Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG'11)*, Ajaccio, Corsica, France, 2011. Civil-Comp Press, Stirlingshire, UK.

[83] M. Chau, T. Garcia, and P. Spitéri. Parallel asynchronous schwarz alternating method for obstacle problems on grid computing. In Y. Tsompanakis B.H.V. Topping, editor, *Proceedings of the Thirteenth International Conference on Civil, Structural and Environmental Engineering Computing*, Chania, Crete, Greece, 2011. Civil-Comp Press, Stirlingshire, UK.

[84] M. Chau, T. Garcia, A. Laouar, and P. Spitéri. Subdomain solution of problem with unilateral constraints in grid environments. In *Proceedings of the 4th international conference on Data management in grid and peer-to-peer systems*, Globe'11, pages 108–119, Berlin, Heidelberg, 2011. Springer-Verlag.

[85] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.G. Talbi, and I. Touche. Grid'5000 : A large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, 20(4) :481–494, 2006.

[86] B. Yagoubi and Y. Slimani. Task load balancing strategy for grid computing. *Journal of Computer Science*, 3(3) :186–194, 2007.

[87] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7) :629–639, 1990.

[88] J. Weickert, B.M. Haar Romeny, and M.A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7(3) :398–410, 1998.

[89] C. Tauber, P. Spitéri, and H. Batatia. Iterative methods for anisotropic diffusion of speckled medical images. *Applied Numerical Mathematics*, 60 :1115–1030, 2010.

[90] D. El Baz and T.T. Nguyen. A self-adaptive communication protocol with application to high performance peer to peer distributed computing. In *Pa-*

rallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on, pages 327–333, 2010.

[91] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2-nd edition, 1999.

[92] S.V. Patankar. *Numerical heat transfer and fluid flow*. Mc Graw Hill, 1980.

[93] P. Wilmott, J. Dewyne, and S. Howison. *Option Pricing-Mathematical Models and Computation*. Oxford financial press, 1993.

[94] D. Lamberton P. Jaillet and B. Lapeyre. Variational inequalities and the pricing of american option. *Acta Applicandae Mathematicae*, 21(3) :263–289, 1990.

[95] R. Glowinski, J.L. Lions, and R. Tremolieres. *Analyse Numérique des Inéquations Variationnelles*. DUNOD, tome 1 and 2, 1976.

[96] P. Lions and B. Mercier. Approximation numérique des équations de Hamilton-Jacobi-Bellman. *RAIRO*, 14 :369–393, 1980.

[97] C. Li, J. Zeng, and S. Zhou. Convergence Analysis of Generalized Schwarz Algorithms for Solving Obstacle Problems with T-monotone Operator. *Computers - mathematics*, 48 :373–386, 2004.

[98] L. Badea and J. Wang. An Additive Schwarz Method for Variational Inequalities. *Math. Comp.*, 69 :1341–1354, 2000.

[99] X. C. Tai. Convergence Rate Analysis of Domain Decomposition Methods for Obstacle Problems. *East-West J. Numer. Anal.*, 9(3) :233–252, 2001.

[100] X. C. Tai. Rate of Convergence for Some Sonstraint Decomposition Methods for Nonlinear Variational Inequalities. *Numer. Math.*, 93 :755–786, 2003.

[101] L. Badea, X. C. Tai, and J. Wang. Convergence Rate Analysis of a Multiplicative Schwarz Method for Variational Inequalities. *SIAM J. on Numer. Analysis*, 41(3) :1052–1073, 2004.

[102] Y. A. Kuznetsov, P. Neittaanmaki, and P. Tarvainen. Block Relaxation Methods for Algebraic Obstacle Problems with M-matrices. *East-West J. Numer. Math.*, 4 :69–82, 1994.

[103] Y. A. Kuznetsov, P. Neittaanmaki, and P. Tarvainen. Schwarz methods for obstacle problems with convection-diffusion operators. In D.E. Keyes and J.C. Xu, editors, *Domain Decomposition Methods in Scientifical and Engineering Computing*, pages 251–256. AMS, 1995.

[104] X. C. Tai and P. Tseng. Convergence Rate Analysis of an Asynchronous Space Decomposition Method for Convex Minimization. *Mathematics of Computation*, 71(239) :1105–1135, 2001.

[105] M. Chau. *Algorithmes parallèles asynchrones pour la simulation numerique*. Thèse de Doctorat, Institut National Polytechnique de Toulouse, Laboratoire d'Informatique et de Mathématiques Appliquées (ENSEEIHT-IRIT), 2005.

[106] 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.

[107] M.J. Clifton. Numerical simulation of protein separation by continuous-flow electrophoresis. *Electrophoresis*, 14 :1284–1291, 1993.

[108] R.I. Issa. Solution of the implicitely discretised fluid flow equations by operator splitting. *Journal of Computational Physics*, 62 :40–65, 1986.

[109] P. K. Smolarkiewicz. A fully multidimensional positive definite advection transport algorithm with small implicit diffusion. *Journal of Computational Physics*, 54 :325–362, 1984.

[110] 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.

[111] P. K. Smolarkiewicz and W. W. Grabowski. The multidimensional positive definite advection transport algorithm : nonoscillatory option. *Journal of Computational Physics*, 86 :355–375, 1990.

[112] L. Ziane Khodja. *Résolution de systèmes linéaires et non linéaires creux sur grappes de GPUs*. Thèse de Doctorat, Université de Franche-Comté, 2013.

[113] 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.

[114] J.D.P. Donnely. Periodic chaotic relaxation. *Linear Algebra appl.*, 4 :117–128, 1971.

[115] 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.

[116] 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.

[117] 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.

[118] T.T. Nguyen. *Un environnement pour le calcul intensif pair à pair*. Thèse de Doctorat, Institut National Polytechnique de Toulouse, 2011.

[119] D. El Baz. A method of terminating asynchronous iterative algorithms on message passing systems. *Parallel Algorithms and Applications*, 9 :153–158, 1996.

[120] K.M. Chandy and L. Lamport. Distributed snapshots : determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1) :63–75, 1985.

[121] D. P. Bertsekas and J. N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithmsa survey. *Automatica*, 27(1) :3 – 21, 1991.

[122] J.M. Bahi, S. Contassot-Vivier, and R. Couturier. *Parallel iterative algorithms : from sequential to grid computing*. Chapman & Hall/CRC, Boca Raton, 2007.

[123] R. Moreno. *Simulations Numériques Vasculaires Spécifiques et Réalistes*. Thèse de Doctorat, Université Toulouse 3, 2007.

[124] R. Moreno, M. Chau, S. Jeetoo, F. Nicoud, F. Viart, A. Salvayre, and H. Rousseau. Optimised computational functional imaging for arteries. In J.M.L.M. Palma et al., editor, *High Performance Computing for Computational Science - VECPAR 2008*, volume 5336 of *Lecture Notes in Computer Science*, pages 420–429. Springer Berlin Heidelberg, 2008.

[125] R. Moreno, M. Chau, B. Tayllamin, H. Rousseau, and F. Nicoud. Correct rheology simulation on compliant thoracic aorta model : comparison between cfd and mri velocity measurements. *Computer Methods in Biomechanics and Biomedical Engineering*, 12(sup1) :195–196, 2009.

[126] M. Midulla, R. Moreno, A. Baali, M. Chau, A. Negre-Salvayre, F. Nicoud, J.P. Pruvo, S. Haulon, and H. Rousseau. Haemodynamic imaging of thoracic stent-grafts by computational fluid dynamics (CFD) : presentation of a patient-specific method combining magnetic resonance imaging and numerical simulations. *European radiology*, 22(10) :2094–102, 2012.

[127] S. Jeetoo. *Perfusion Quantitative Pulmonaire à l'IRM chez le sujet HTAP*. Thèse de Doctorat, Université Toulouse 3, 2009.

[128] A. Baali. *Scanner Somatom Definition à double énergie : application à la caractérisation lésionnelle des plaques vulnérables d'athérosclérose et à l'analyse fonctionnelle du ventricule gauche*. Thèse de Doctorat, Université Toulouse 3, 2012.

[129] B. Tayllamin. *Evaluation d'une méthode de Frontières Immergées pour les Simulations Numériques d'Ecoulements Cardiovasculaires*. Thèse de Doctorat, Université Montpellier 2, 2012.

[130] J. Ashburner, J.L. Anderson, and K.J. Friston. High-dimensional image registration using symmetric priors. *Neuroimage*, 9(6 Pt 1) :619–628, 1999.

[131] J. Ashburner, J.L. Anderson, and K.J. Friston. Image registration using a symmetric prior - in three dimensions. *Human Brain Papping*, 9(4) :212–225, 2000.

[132] C.S. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25(3) :220–252, 1977.

[133] C.S. Peskin. The immersed boundary method. *Acta Numerica*, 11 :479–517, 2002.

[134] V.E. Henson and U.M. Yang. Boomeramg : a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(5) :155–177, 2000.

[135] 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.