Méthodes de décomposition de domaines en temps et en espace pour la résolution de systèmes d'EDOs non-linéaires

Patrice Linel

Université Claude Bernard

N° attribué par la bibliothèque
102-2011

THÈSE

pour obtenir le grade de

Docteur de Université Claude Bernard

Spécialité : **Mathématiques appliquées**

préparée au laboratoire **Institut Camille Jordan - UMR 5208**

dans le cadre de l'École Doctorale **École Doctorale Info-Math**

présentée et soutenue publiquement
par

Patrice Linel

le 05 Juillet 2011

Titre:

Méthodes de décomposition de domaines en temps et en espace pour la résolution de systèmes d'EDOs non-linéaires

Jury

M. Choi-Hong LAI,	Rapporteur
M. François-Xavier ROUX,	Rapporteur
M. Taoufik SASSI,	Rapporteur
M. François DELEBECQUE,	Jury
M. Bernhard MASCHKE,	Jury
M. Christophe PRUD'HOMME,	Jury
M. Laurent LEFEVRE,	Directeur de thèse
M. Damien TROMEUR-DERVOUT,	Directeur de thèse

Méthodes de décomposition de domaines en temps et en espace pour la résolution de systèmes d'EDOs non-linéaires

Résumé : La complexification de la modélisation multi-physique conduit d'une part à devoir simuler des systèmes d'équations différentielles ordinaires et d'équations différentielles algébriques de plus en plus grands en nombre d'inconnues et sur des temps de simulation longs. D'autre part l'évolution des architectures de calcul parallèle nécessite d'autres voies de parallélisation que la décomposition de système en sous-systèmes. Dans ce travail, nous proposons de concevoir des méthodes de décomposition de domaine pour la résolution d'EDO en temps. Nous reformulons le problème à valeur initiale en un problème aux valeurs frontières sur l'intervalle de temps symétrisé, sous l'hypothèse de réversibilité du flot. Nous développons deux méthodes, la première apparentée à une méthode de complément de Schur, la seconde basée sur une méthode de type Schwarz dont nous montrons la convergence pouvant être accélérée par la méthode d'Aitken dans le cadre linéaire. Afin d'accélérer la convergence de cette dernière dans le cadre non-linéaire, nous introduisons les techniques d'extrapolation et d'accélération de la convergence des suites non-linéaires. Nous montrons les avantages et les limites de ces techniques. Les résultats obtenus nous conduisent à développer l'accélération de la méthode de type Schwarz par une méthode de Newton. Enfin nous nous intéressons à l'étude de conditions de raccord non-linéaires adaptées à la décomposition de domaine de problèmes non-linéaires. Nous nous servons du formalisme hamiltonien à ports, issu du domaine de l'automatique, pour déduire les conditions de raccord dans le cadre l'équation de Saint-Venant et de l'équation de la chaleur non-linéaire. Après une étude analytique de la convergence de la DDM associée à ces conditions de transmission, nous proposons et étudions une formulation de Lagrangien augmenté sous l'hypothèse de séparabilité de la contrainte.

Mots clé : Complément de Schur, Décomposition de domaine en temps, Newton-Krylov, Parallélisation, Accélération non-linéaire, Condition interface

Time and space domain decomposition method for nonlinear ODE

Abstract : Complexification of multi-physics modeling leads to have to simulate systems of ordinary differential equations and algebraic differential equations with increasingly large numbers of unknowns and over large times of simulation. In addition the evolution of parallel computing architectures requires other ways of parallelization than the decomposition of system in subsystems. In this work, we propose to design domain decomposition methods in time for the resolution of EDO. We reformulate the initial value problem in a boundary values problem on the symmetrized time interval, under the assumption of reversibility of the flow. We develop two methods, the first connected with a Schur complement method, the second based on a Schwarz type method for which we show convergence, being able to be accelerated by the Aitken method within the linear framework. In order to accelerate the convergence of the latter within the non-linear framework, we introduce the techniques of extrapolation and of acceleration of the convergence of non-linear sequences. We show the advantages and the limits of these techniques. The obtained results lead us to develop the acceleration of the method of the type Schwarz by a Newton method. Finally we investigate non-linear matching conditions adapted to the domain decomposition of nonlinear problems. We make use of the port-Hamiltonian formalism, resulting from the control field, to deduce the matching conditions in the framework of the shallow-water equation and the non-linear heat equation. After an analytical study of the convergence of the DDM associated with these conditions of transmission, we propose and study a formulation of augmented Lagrangian under the assumption of separability of the constraint.

Keywords : Domain decomposition, Schur complement , Time domain decomposition, Newton-Krylov, Parallelization, Nonlinear acceleration, Interface condition

Remerciements

Je voudrais tout d'abord remercier Damien Tromeur-Dervout à qui j'exprime toute ma reconnaissance, pour la patience, la confiance, et la disponibilité dont il a su faire preuve pour m'encadrer, m'encourager et me guider dans la conduite de ces travaux en sa qualité de directeur de thèse. Je lui assure de tout mon respect et de toute ma reconnaissance pour ses précieux conseils.

Mes remerciements vont également à Laurent Lefevre, mon second directeur de thèse, à qui j'adresse toute ma gratitude sans qui ce travail n'aurait pas été possible et je tiens à lui témoigner toute ma reconnaissance pour l'intérêt qu'il a manifesté pour ce travail.

Je tiens à remercier très chaleureusement Messieurs Choi-Hong Lai, François-Xavier Roux et Taoufik Sassi qui m'ont fait l'honneur d'accepter d'être les rapporteurs de ma thèse et de compter parmi les membres du jury. Merci pour vos remarques et conseils sur l'amélioration du manuscrit.

Je remercie Messieurs Bernhard Maschke, Christophe Prud'homme et François Delebecque pour leur participation au jury et qu'ils soient assurés de ma très sincère reconnaissance.

Je désire également remercier tous les membres de l'équipe de l'ICJ à l'EPU-Lyon qui m'a accueilli et m'a fait partager leurs cultures et expériences scientifiques ; j'ai beaucoup appris à leurs côtés. C'était un réel plaisir de travailler ensemble, nous avons passé des moments mémorables. Encore merci à tous ceux que j'ai eu l'occasion de côtoyer de près ou de loin. Que tous ceux que je n'ai pas cités veuillent bien m'en excuser ; il serait difficile de les nommer ici de manière exhaustive.

Enfin, j'adresse mes remerciements à ma famille qui m'a soutenu et aidé.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La simulation numérique et le calcul scientifique jouent un rôle essentiel dans l'ensemble du champ de la science, à la fois comme instrument de description et comme instrument de prédiction. L'évolution des ressources informatiques et des besoins de conception conduisent à la formulation de systèmes d'équations différentielles(EDO/EDA) ou aux dérivées partielles(EDP) de plus en plus sophistiqués et complets. La simulation et le développement de nouvelles méthodes numériques adaptées au calcul parallèle sont devenus primordiaux.

En effet, le calcul scientifique se développe maintenant hors des communautés traditionnelles, comme la biologie et le génie des procédés qui font dorénavant appel au calcul scientifique souvent sur des modèles aux EDOs/EADs pour modéliser des phénomènes multi-physiques et multi-échelles. La complexification de la modélisation multi-physique conduit à devoir simuler des systèmes d'EDOs et EADs de plus en plus grands, en nombre d'inconnues et dont la raideur évolue avec la dynamique des solutions. Les temps de calcul pour des applications "industrielles" deviennent dès lors trop importants et nécessitent de concevoir et d'implémenter des méthodes mathématiques précises, stables et consistantes adaptées aux architectures multiprocesseurs de calcul afin de réduire les temps de restitution des résultats.

Le sujet proposé ici a pour ambition de concevoir des techniques de parallélisations pour les EDOs qui soient performantes en temps long, d'investiguer l'accélération des méthodes de décomposition de domaine, ainsi que les conditions de transmission adéquates pour des problèmes non-linéaires.

Le domaine de la résolution parallèle de systèmes d'EADs est un sujet de recherche important dans la communauté du calcul scientifique comme le démontrent les efforts pour développer des solveurs aux EDOs/EADs au Lawrence Livermore National Laboratory responsable de la suite SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers). La communauté scientifique travaille depuis longtemps sur l'adaptivité des pas de

temps et la gestion des discontinuités dans les systèmes d'EDOs, l'effort sur la parallélisation de ces systèmes est relativement récent. Les travaux de K. Burrage dans les années 1990, de parallélisation "accross the method" ne peuvent s'appliquer que sur un nombre faible de processeurs dépendant du nombre d'étapes dans les méthodes de Runge-Kutta. Du point de vue informatique, SUNDIALS propose une technique parallèle qui consiste à partitionner les inconnues du système entre les processeurs par blocs contigus, ne conduisant pas à des performances optimales surtout si la taille du système devient importante avec plusieurs dizaines de milliers d'inconnues. La stratégie de décomposition des systèmes d'équations en sous systèmes a été abordée dans les travaux de Skelboe [81, 80, 79] avec un estimateur a priori du découpage. Les méthodes de "Waveform Relaxation" s'inscrivent dans ce cadre. B. Pohl et K. Burrage proposent dans [28] une parallélisation par la résolution itérative indépendante des sous-systèmes par un processus de type Jacobi ou Gauss-Seidel. Des résultats intéressants ont été obtenus sur des problèmes particuliers où les dynamiques des solutions sont continues et non très raides. De même, les techniques parallèles $C(p, q, j)$ [42] développées par Garbey et Tromeur-Dervout sont un premier pas pour relaxer les communications entre les termes de couplage dans le cadre de systèmes aux équations dérivées partielles (EDP) modélisant des physiques couplées.

La décomposition de domaine en temps constitue une solution pour lever les difficultés et réduire les temps de restitution des résultats. Cependant, les méthodes de type multiple shooting [Deuflhard] comme Pita [Farhat] ou Parareal [Lions] sont très sensibles à la raideur du système linéarisé pour la correction des sauts de solutions aux interfaces. Dans [Guibert Tromeur-Dervout] les auteurs se sont attachés à coupler les techniques de deferred correction [Layton Minion] avec la décomposition en temps pour pallier ces instabilités numériques . Dans ce travail, nous reprenons le principe du découpage de l'intervalle de temps de simulation en sous intervalles et nous proposons des techniques différentes pour le raccord entre les sous-domaines.

Ce travail de thèse a été développé en trois axes ; le premier concerne l'adaptation de méthodes d'intégration en temps aux techniques de décomposition de domaine bien connues dans le domaine des EDPs spatiales.

Nous commencerons par une introduction aux méthodes de décomposition de domaines en espace avec recouvrement (méthode de Schwarz) ainsi que les méthodes dites sans recouvrement (méthode du complément de Schur). Les méthodes classiques d'intégration en temps sont également introduites

afin de mieux appréhender la difficulté de la décomposition en temps. Les méthodes de décomposition en temps sont basées sur le découpage de l'intervalle de temps. La difficulté pour la mise en oeuvre des méthodes de décomposition de domaine est l'absence de conditions de bord en fin d'intervalle. En effet, ces techniques ont été développées à partir de problèmes aux valeurs frontières. Dans le chapitre 2, nous exposons les idées qui nous ont permis de contourner cette difficulté en proposant deux méthodes, une première basée sur une méthode de type Schwarz et la seconde apparentée à une méthode de complément de Schur. Nous mettons en évidence les performances de ces méthodes sur calculateurs parallèles en comparaison avec la méthode pararéal. Les limites des méthodes développées dans le cadre des équations non-linéaires nous conduisent vers notre deuxième axe.

En effet, dans le cadre d'équations non-linéaires, nos résultats numériques montrent une dégradation des performances, les techniques mises en place dans le cadre linéaire ne s'appliquant pas directement. La partie II s'articule autour de l'étude des méthodes d'accélération de la convergence des suites non-linéaires. Le chapitre 3 introduit les techniques d'extrapolation et de l'accélération de la convergence des suites. Les algorithmes concernant les suites non-linéaires scalaires et vectoriels sont présentés avant d'être implémentés dans le chapitre 4. Nous montrons les avantages et les limites de ces techniques au travers des différents résultats obtenus. D'abord combiné avec la méthode itérative de décomposition de domaine de Schwarz, puis avec celle du complément de Schur, nous étudions l'accélération de la méthode de type Schwarz par une méthode de Newton. La satisfaction des contraintes égalitaires de la méthode du complément de Schur, effectuée par une méthode de Newton, est transformée en une suite d'itérés par une méthode de point fixe sur laquelle le comportement des algorithmes est étudié.

Le troisième axe est porté sur la recherche de conditions de transmission non-linéaires pour les algorithmes de décomposition de domaine et sera développé dans la partie III. Ce travail est effectué en collaboration avec l'équipe de dynamique des procédés du Laboratoire d'Automatique et de Génie des Procédés - UMR 5007 UCB-CNRS qui développe des modèles dynamiques structurés pour la simulation numérique, l'estimation paramétrique, l'observation et la commande de procédés complexes multi-physiques et multi-échelles. Le chapitre 5 introduit le formalisme hamiltonien à ports ainsi que les différents éléments de base du bondgraph. La méthodologie est mise en oeuvre au travers de deux exemples, hyperbolique et parabolique, par le développement d'une bibliothèque d'éléments prenant en compte

l'implicitation du problème.
Le chapitre 6 est consacré à l'étude de conditions de raccord pour la décomposition de domaine appliquée à l'équation de Saint-Venant. Le formalisme hamiltonien à ports est utilisé pour déduire des conditions de transmission pour une méthode de type Schwarz en espace.
Dans la continuité de la recherche de conditions de raccord non-linéaires déduites du formalisme hamiltonien, le chapitre 7 apporte des résultats complémentaires avec un travail sur la décomposition de domaine en espace de l'équation de la chaleur non-linéaire.

Première partie

Décompositions de domaine en temps

Table des matières

Chapitre 1

Introduction aux méthodes d'intégration en temps et de décompositions de domaine

Dans ce chapitre, nous ferons une introduction aux méthodes d'intégration des EDOs, puis nous donnerons les bases des méthodes de décomposition de domaines en espace. Cette introduction permettra de mieux aborder le problème de la décomposition de domaine pour les EDOs du chapitre suivant.

Table des matières

1.1 Intégration des EDOs

Nous n'introduirons que les méthodes à un pas car les méthodes développées dans le chapitre suivant ne dépendent que de la propriété de symétrie du schéma d'intégration. De façon générale, le problème que nous considérons dans ce travail s'écrit sous forme de systèmes d'équations différentielles ordinaires non linéaires

On considère une fonction continue $f : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. Pour $y_0 \in \mathbb{R}$ donné, on cherche $y : t \in \mathbb{R}_+ \to y(t) \in \mathbb{R}$ qui satisfait le problème suivant, appelé

problème de Cauchy :

$$\begin{cases} \dot{y} = f(t, y) \quad t \in]T_0, T] \\ y(T_0) = y_0 \end{cases} \tag{1.1}$$

où la variable t représente souvent le temps, $y(T_0)$ la condition initiale et $\dot{y} = \dfrac{dy(t)}{dt}$.

Numériquement on approche la dérivée $\dot{y}(t)$ par des schémas d'intégration (différences finies). Le plus simple des schémas est celui formulé par Euler :

$$\begin{cases} y_0 = y(T_0) \\ \dfrac{y_{n+1} - y_n}{h} = f(t_n, y_n) \quad \text{pour } n = 0, 1, 2, .. \end{cases} \quad avec \;\; t_n = T_0 + nh \tag{1.2}$$

Ce schéma utilise un pas de temps constant h pour calculer les approximations $y_n \simeq y(t_n)$ les unes après les autres, de la solution en partant de $y_0 = y(T_0)$. Cette méthode est dite explicite car elle ne fait intervenir dans le calcul de y_{n+1} que des quantités connues à l'itération n. Les schémas faisant intervenir des quantités inconnues dans le calcul du pas de temps courant, sont dit implicites. Le schéma d'Euler implicite s'écrit :

$$\begin{cases} y_0 = y(T_0) \\ y_{n+1} = y_n + hf(t_n, y_{n+1}) \quad \text{pour } n = 0, 1, 2, ... \end{cases} \tag{1.3}$$

Ce schéma requiert alors la résolution d'un système d'équations non-linéaires, mais bénéficie d'une stabilité inconditionnelle[51].

Le schéma du point milieu implicite fait intervenir y_n et y_{n+1} dans la fonction f

$$\begin{cases} y_0 = y(T_0) \\ y_{n+1} = y_n + hf(\dfrac{y_n + y_{n+1}}{2}) \quad \text{pour } n = 0, 1, 2, ... \end{cases} \tag{1.4}$$

Ce schéma à la propriété d'être symétrique, c'est-à-dire que le schéma reste le même en échangeant $y_n \leftrightarrow y_{n+1}$ et $h \leftrightarrow -h$ (Nous détaillerons cette notion dans le chapitre 2)

D'autres méthodes d'approximation de solutions d'équations différentielles importantes sont celles de Runge-Kutta (1895-1901). La première méthode de Runge-Kutta (RK) *implicite* fut introduite par Cauchy (1824). Les méthodes de Runge-Kutta implicites sont introduites par Butcher [29].

Définition 1.1. Soient b_i, a_{ij} $(i = 1, \cdots, s)$ réels, et c_i défini par

$$c_i = \sum_{j=1}^{i-1} a_{ij}$$

La méthode

$$\begin{cases} k_i = f\left(x_0 + c_i h, y_0 + \sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, \ldots, s \\ y_1 = y_0 + h \sum_{i=1}^{s} b_i k_i \end{cases} \tag{1.5}$$

est appelée *méthode implicite de Runge-Kutta à s-étapes* , car les équations en k_i sont implicites.

Définition 1.2. Une méthode de RK (1.5) est d'ordre p si pour un problème suffisamment régulier,

$$||y(t_0 + h) - y_1|| \leq K h^{p+1} \tag{1.6}$$

Autrement dit, la série de Taylor de la solution exacte $y(t_0 + h)$ et y_1 coïncident jusqu'au terme h^p (inclus).

Les méthodes de Runge-Kutta implicites sont caractérisées par un triplet (A, b, c) où A est la matrice carrée $(a_{ij})_{i,j=1,\ldots,s}$ de dimension $s \times s$, b et c sont respectivement les vecteurs de dimension s, $b^T = (b_1, \ldots, b_s)$ et
$c = (c_1, \ldots, c_s)^T$.

Ces méthodes sont représentées sous forme d'un tableau [29] :

$$\begin{array}{c|ccc} c_1 & a_{11} & \ldots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \ldots & a_{ss} \\ \hline & b_1 & \ldots & bs \end{array} \tag{1.7}$$

Remarque 1.1. En posant $g_i = y_0 + \sum_{j=1}^{s} a_{ij} k_j$, les méthodes de RK sont généralement écrites

$$\begin{cases} g_i = y_0 + h \sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, g_j), \quad i = 1, \ldots, s \\ y_1 = y_0 + h \sum_{j=1}^{s} b_j f(x_0 + c_j h, g_j) \end{cases} \tag{1.8}$$

Il est alors possible d'écrire ces méthodes sous la forme matricielle suivante

$$\begin{cases} G = I_s \otimes y_0 + h(A \otimes I_s)F(G) \\ y_1 = y_0 + h(b^T F(G)) \end{cases} \tag{1.9}$$

en posant

$$G = \begin{pmatrix} g_1 \\ \vdots \\ g_s \end{pmatrix}, \qquad F(G) = \begin{pmatrix} f(x_0 + c_1 h, g_1) \\ \vdots \\ f(x_0 + c_s h, g_s) \end{pmatrix}$$

avec l'opérateur tensoriel $\otimes$ défini par

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & \dots & a_{1m}B \\ a_{21}B & \dots & \dots & a_{2m}B \\ \vdots & & & \vdots \\ a_{n1}B & \dots & \dots & a_{nm}B \end{pmatrix}$$

Ces méthodes itératives sont par nature séquentielles, c'est-à-dire qu'une première estimation de la solution est utilisée pour calculer une seconde estimation et, ainsi de suite. Des travaux on été effectués concernant la parallélisation des méthodes de Runge-Kutta [50] mais ces méthodes restent limitées par le nombre d'étapes de la méthode de Runge-Kutta. Cela nous amène à investiguer les méthodes de décomposition de domaines en espace pour la décomposition en temps afin de paralléliser l'intégration des EDOs. Dans la section suivante, nous faisons un rappel des méthodes classiques conçues initialement pour des problèmes spatiaux.

1.2 Décompositions de domaine

L'une des réponses à la problématique de parallélisation d'un problème, se trouve dans les méthodes de décomposition de domaines. L'idée est de découper le domaine de calcul en plusieurs sous-domaines. Il existe deux grandes familles de méthodes pour ce découpage par sous-domaines :

1. Les méthodes avec recouvrement
2. Les méthodes sans recouvrement

Nous illustrerons ces deux types de méthodes appliquées aux problèmes en espace pour des opérateurs linéaires. Parmi les méthodes sans recouvrement, il existe les méthodes de types Schwarz et celles de type complément de Schur. Nous introduirons pour les méthode de type Schwarz celles avec recouvrement.

1.2.1 Méthodes avec recouvrement

La méthode de Schwarz[72, 83]

On recherche la solution du problème de Poisson représenté par l'opérateur $\mathcal{L}$ sur le domaine Ω suivant :

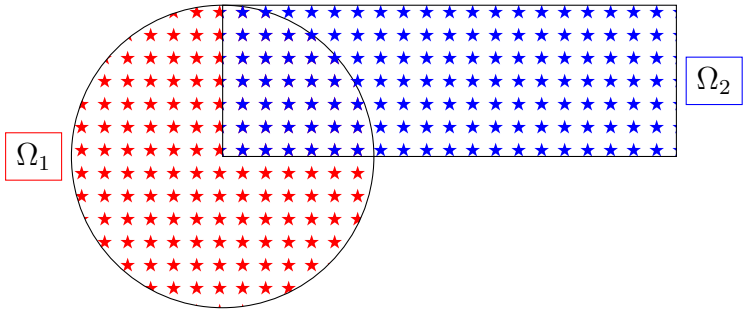


FIGURE 1.1 – Décomposition de domaine $\Omega = \Omega_1 \cup \Omega_2$

$$\begin{cases} \mathcal{L}u = f \quad dans \quad \Omega \\ u = 0 \quad sur \quad \partial\Omega \end{cases} \tag{1.10}$$

La méthode de Schwarz est une méthode itérative. On définit une solution initiale sur tout le domaine, et sur le bord des sous-domaines. Pour chaque sous-domaine, on a un problème de Poisson bien défini grâce aux conditions de Dirichlet. En résolvant chaque sous-domaine on obtient de nouvelles valeurs de bord. Puis on réitère ce processus jusqu'à convergence.

Le découpage du domaine Ω en un sous-domaine rectangulaire Ω_1 et un sous-domaine circulaire Ω_2. On écrit la méthode de Schwarz pour un opérateur $\mathcal{L}$ linéaire :

– Étape 1 : on résout

$$❶ \begin{cases} \mathcal{L}u_1^{2n+1} = f \quad dans \quad \Omega_1 \\ u_1^{2n+1} = 0 \quad sur \quad \partial\Omega \cap \Omega_1 \\ u_1^{2n+1} = u_2^{2n} \quad sur \quad \partial\Omega_2 \cap \partial\Omega_1 \end{cases} \tag{1.11}$$

– Étape 2 : on résout

$$❷ \begin{cases} \mathcal{L}u_2^{2n+2} = f \quad dans \quad \Omega_2 \\ u_2^{2n+2} = 0 \quad sur \quad \partial\Omega \cap \Omega_2 \\ u_2^{2n+2} = u_1^{2n+1} \quad sur \quad \partial\Omega_1 \cap \partial\Omega_2 \end{cases} \tag{1.12}$$

– Si la convergence n'est pas atteinte à un ϵ près, on effectue une itération supplémentaire.

Cette manière d'itéré est la méthode de Schwarz multiplicative. Il existe une version dite additive, où tous les sous-domaines sont résolus avant la mise à jour des conditions interfaces.

Algorithm 1 Algorithme de Schwarz multiplicatif

1: **while** $||u_1^{2n+1} - u_1^{2n-1}||_2 \geq eps$ **do**
2: Résoudre le problème ❶ à l'itération $2n+3$
3: Échange de la condition de Dirichlet u_1^{2n+3}
4: Résoudre le problème ❷ à l'itération $2n+4$
5: Échange de la condition de bord de Dirichlet
6: **end while**

Dans le cas d'opérateur vérifiant le principe du maximum (décroissance de la solution par rapport aux valeurs de bord), la vitesse de convergence de ces algorithmes dépend de la taille du recouvrement. Plus il est important et plus la convergence sera rapide.[72, 83]

1.2.2 Méthodes sans recouvrement

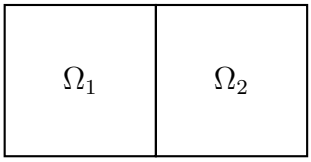
Soit le problème

$$\begin{cases} \Delta u = f \quad sur \quad \Omega = \Omega_1 \cup \Omega_2 \\ u = 0 \quad sur \quad \partial\Omega \end{cases} \tag{1.13}$$

Découpons le domaine Ω en deux sous-domaines. Nous rajoutons les conditions d'égalité de la solution et de la dérivée normale aux interfaces, ces conditions sont dites conditions de raccord.

$$\begin{cases} \Delta u_1 = f_1 \quad sur \quad \Omega_1 \\ u_1 = 0 \quad sur \quad \partial\Omega_1 \backslash \Gamma \\ u_1 = u_2 \quad sur \quad \Gamma = \partial\Omega_1 \cap \partial\Omega_2 \\ \dfrac{\partial u_2}{\partial n_2} = -\dfrac{\partial u_1}{\partial n_1} \overset{def}{=} \lambda_\Sigma \quad sur \quad \Gamma \Delta u_2 = f_2 \quad sur \quad \Omega_2 \\ u_2 = 0 \quad sur \quad \partial\Omega_2 \backslash \Gamma \end{cases} \tag{1.14}$$

En numérotant d'abord les inconnues des domaines puis celles des interfaces entre sous-domaines, on obtient le problème suivant sous forme matricielle équivalent à (1.13).



On obtient une matrice telle que :

$$\begin{pmatrix} A_1 & & C_1 \\ & A_2 & C_2 \\ B_1 & B_2 & A_\sum \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_\sum \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_\sum \end{pmatrix} \tag{1.15}$$

où $\sum = \partial\Omega_1 \cap \Omega_2 = \partial\Omega_2 \cap \Omega_1$ représente la frontière entre sous-domaine. Les u_k sont les inconnues intérieures au domaine k, et $u_\sum$ les inconnues interface aux domaines. Alors A_k représente la discrétisation de l'opérateur linéaire pour les inconnues intérieures du domaine k. Les matrices C_k, respectivement B_k, représentent l'influence du bord sur l'intérieur et respectivement l'inverse. Le second membre f_k est la contribution du second membre pour les inconnues intérieures, tandis que $f_\sum$ est la contribution du second membre pour les inconnues interfaces.

Méthode de Schur primale

On a les équations suivantes :

$$A_k u_k + C_k u_\sum = f_k, \quad k = 1, 2 \tag{1.16}$$

et

$$\sum_{k=1}^{2} B_k u_k + A_\sum u_\sum = f_\sum \tag{1.17}$$

A_k est la même matrice quelque soit le domaine Ω_k. On peut écrire $f_\sum$ comme étant la somme des contributions des deux domaines $f_{1,\sum} + f_{2,\sum}$, et de même pour $A_\sum = A_{1,\sum} + A_{2,\sum}$.
Considérons que les inconnues interfaces sont les mêmes, $u_{1,\sum} = u_{2,\sum} = u_\sum$ on peut alors éliminer les inconnues locales en exprimant u_k à partir de l'équation (1.16),

$$u_k = A_k^{-1}(f_k - C_k u_\sum) \tag{1.18}$$

En reportant ces valeurs dans l'équation (1.17) , on obtient une équation en $u_\sum$,

$$\begin{cases} S_k = A_{k,\sum} - B_k A_k^{-1} C_k \\ g_{k,\sum} = f_{k,\sum} - B_k A_k^{-1} f_k \end{cases} \tag{1.19}$$

$$\begin{cases} S = S_1 + S_2 \\ g_\sum = g_{1,\sum} + g_{2,\sum} \end{cases} \tag{1.20}$$

$$S u_\sum = g_\sum \tag{1.21}$$

Dès que l'on connait $u_\sum$,en résolvant (1.21), les solutions intérieures de chaque domaine sont données par $u_k = A_k^{-1}(f_k - C_k u_\sum)$.

La méthode se résume par le système linéaire suivant :

$$\begin{pmatrix} A_1 & 0 & C_1 \\ 0 & A_2 & C_2 \\ 0 & 0 & S \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_\sum \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ g_\sum \end{pmatrix} \tag{1.22}$$

On résout alors ce système par une méthode de descente de type Krylov. Le produit matrice-vecteur ainsi que le calcul du second membre peut se faire en parallèle. Pour chaque produit-matrice vecteur , il y a un système linéaire local à résoudre ; ce système est petit en comparaison avec le problème initial.

Méthode de Schur duale

Chaque domaine numérote les noeuds de l'interface, l'inconnue n'est plus commune aux sous-domaines. La continuité de la solution à l'interface est alors considérée comme une contrainte. En suivant une démarche analogue à la méthode primale précédente [83], et en prenant en compte la contrainte d'avoir $u_{1,\sum} = u_{2,\sum}$, nous poserons

$$\lambda_\sum = B_1 u_1 + A_{1,\sum} u_{1,\sum} - f_{1,\sum} = -(B_2 u_2 + A_{2,\sum} u_{2,\sum} - f_{2,\sum}) \tag{1.23}$$

Nous utilisons l'expression des u_k intérieurs (1.18), et en remplaçant dans l'équation (1.23) on obtient

$$S_1 u_{1,\sum} = g_{1,\sum} + \lambda_\sum \tag{1.24}$$

Soit la reformulation de la contrainte égalité $u_{1,\sum} = u_{2,\sum}$ comme

$$S_1^{-1}(g_{1,\sum} + \lambda_\sum) - S_2^{-1}(g_{2,\sum} - \lambda_\sum) = 0 \tag{1.25}$$

donc $\lambda_\sum$ est calculée comme la solution de l'équation suivante :

$$(S_1^{-1} + S_2^{-1})\lambda_\sum = S_2^{-1} g_{2,\sum} - S_1^{-1} g_{1,\sum} \tag{1.26}$$

Une fois que l'on connait $\lambda_\sum$, les solutions locales se calculent en résolvant les problèmes (1.27). Sous forme matricielle :

$$\begin{pmatrix} A_k & C_k \\ B_k & A_{k,\sum} \end{pmatrix} \begin{pmatrix} u_k \\ u_{k,\sum} \end{pmatrix} = \begin{pmatrix} f_k \\ f_{1,\sum} + \lambda_\sum \end{pmatrix} \tag{1.27}$$

Méthode Neumann-Dirichlet

Il existe aussi des méthodes de type Schwarz sans recouvrement. Ces méthodes ont été d'abord introduites dans [65, 66, 15]. Considérons deux sous-domaines $\Omega_i, \quad i = 1, 2$, soit la solution initiale $\partial u^0_{\sum}$, on résout premièrement le problème sur Ω_1 avec la condition de Neumann $\partial u^0_{\sum}$ sur $\sum$, puis le problème sur Ω_2 avec la condition de Dirichlet sur $\sum$ déterminé par la solution obtenue sur Ω_1 à l'étape précédente. La nouvelle condition de transmission $\partial u^1_{\sum}$ pour le domaine Ω_1 est la derivée normale calculée en $\sum$ sur Ω_2. On peut écrire alors

$$\begin{cases} \Delta u_1^{n+1/2} = f \in \Omega_1, \\ u_1^{n+1/2} = 0 \quad on \quad \partial\Omega_1 \ \sum, \\ \dfrac{\partial u_1^{n+1/2}}{\partial n_1} = -\dfrac{\partial u_2^n}{\partial n_2} \quad on \quad \sum \end{cases} \qquad \begin{cases} \Delta u_2^{n+1} = f \in \Omega_2, \\ u_2^{n+1} = 0 \quad on \quad \partial\Omega_2 \ \sum, \\ u_2^{n+1} = u_1^n \quad on \quad \sum \end{cases} \tag{1.28}$$

Dans la suite de cette thèse, nous nous focaliserons sur les méthodes sans recouvrement. En effet celles-ci sont plus aisément adaptables du fait de la symétrie présente dans la méthode que nous développerons au chapitre suivant.

1.3 Principes généraux de la mesure de l'efficacité d'une parallélisation

On se pose alors les questions de mise en oeuvre : Quelles données sur quels processeurs ? Quelles sont les communications nécessaires ? Quelle structure de programmation est la mieux adaptée à notre problème ? Une solution pour les méthodes de décomposition de domaines présentées est d'affecter un ou plusieurs sous-domaines par processeur, en gardant à l'esprit qu'il faut minimiser au maximum les communications entre les processeurs. Cela implique de se préoccuper de la structure de donnée pour limiter les dépendances, l'efficacité de la parallélisation dépendant des parties séquentielles. En ce qui concerne la structure de programmation, elle doit être adaptée à la stratégie adoptée pour la parallélisation, les langages les plus utilisés sont Open-MP et Message Passing Interface (MPI) en Fortran ou en C/C++.

Un code numérique peut être limité par deux facteurs, le temps de restitution des résultats et une limitation du volume de données en mémoire. De plus, l'architecture de nos ordinateurs, en constante évolution, fait intervenir plusieurs coeurs au sein d'un même processeur et très récemment permet le

calcul sur GPU. Il est donc nécessaire de développer des méthodes numériques permettant de tirer partie de toute la puissance disponible en tenant compte des contraintes. Avant de commencer la parallélisation, il convient d'effectuer une analyse du code séquentiel pour déterminer son potentiel parallèle.

Idéalement, l'accélération due à la parallélisation devrait être linéaire, c'est à dire qu'en doublant le nombre de processeurs de calcul, on devrait obtenir une réduction de moitié du temps d'exécution.

Définition 1.3. On définit le Speedup par la formule suivante :

$$S_p = \frac{T_1}{T_p}$$

où p est le nombre de processeurs, T_1 est le temps d'exécution de l'algorithme séquentiel sur 1 processeur et, T_p est le temps d'exécution de l'algorithme parallèle sur p processeurs.

On parle de speedup linéaire quand $S_p = p$. Quand un algorithme a un speedup linéaire, le temps d'exécution est divisé par deux quand le nombre de processeur est doublé. On parle de speedup absolu quand T_1 est le temps d'exécution du meilleur algorithme séquentiel, et de speedup relatif quand T_1 est le temps d'exécution du même algorithme parallèle sur un processeur. L'éfficacité est une mesure de la performance définie par $E_p = \frac{S_p}{p}$. Les algorithmes ayant un speedup linéaire ont une efficacité de 1.

Dans les années 1960, Gene Amdahl a formulé une loi empirique éponyme restée célèbre. La loi d'Amdahl affirme que la petite partie du programme qui ne peut être parallélisée limite la vitesse globale du programme. Cependant les programmes d'ingénieries ou les résolutions mathématiques contiennent des parties parallélisables et des parties séquentielles non parallélisables. Cette loi stipule, qu'une fois optimisé, il existe au sein du programme une relation entre le ratio de code parallélisé et la vitesse globale d'exécution du programme. Une loi analogue est énoncée par Gustafson, elle décrit une limite du speed-up produit par la parallélisation. $S(p) = p - \alpha * (p-1)$ où p est le nombre de processeurs, S est le speed-up, et α la partie non-parallélisable du code.

Il existe plusieurs types d'architecture matérielle parallèle. L'architecture SIMD (Single Instruction, Multiple Data), est un type de machine parallèle ayant plusieurs unités de calculs qui effectuent la même opération sur plusieurs données simultanément. Ce type de machine est approprié pour exploiter le parallélisme de données, il nécessite des processeurs vectoriels pour manipuler les flux de données. Dans le modèle SPMD(Single Program

Multiple Data), plusieurs processeurs exécutent simultanément le même programme sur des données différentes. Les tâches peuvent être exécutées sur des CPUs à unité de traitement universel. Il faut également distinguer l'architecture mémoire partagée et distribuée. Sur une machine à mémoire partagée, tous les processeurs accèdent à toute la mémoire, l'échange de données entre processeurs est alors très efficace. En mémoire distribuée, la machine parallèle est généralement séparée en plusieurs boites (dit noeuds) comportant leur propre mémoire. Ainsi les processeurs des autres noeuds ne peuvent accéder directement à la mémoire, l'échange de données se fait alors par un réseau dédié (InfiniBand).

Chapitre 2

Décomposition de domaine en temps par les méthodes de type Schwarz et complément de Schur

Table des matières

2.1 Introduction

La décomposition de domaine en espace est souvent limitée par la taille du domaine à calculer ; la décomposition de domaine en temps devient une

question essentielle afin d'obtenir un parallélisme important inhérent au développement des capacités de calculs. L'approximation numérique d'EDO a été et continue d'être un champ actif de recherche, notamment le développement de nouvelles méthodes prenant en comptant l'aspect parallélisme.

L'une des premières méthodes pour la résolution d'EDO en parallèle est la méthode de tirs multiples introduite par Riley [70, 4]. De nombreuses variantes des méthodes de tirs ont été développées. Dans [41], le lien entre la méthode Pararéal[61, 63] et la méthode de tirs multiples est montré.

Dans la communauté de la décomposition de domaine, la méthode Pararéal a fait l'objet de beaucoup d'attention. L'algorithme Pararéal a pour but de découper l'intervalle de temps$[0, T]$ en p sous domaines $S_i = [T_{i-1}^+, T_i^-]$, avec $T_0 = 0$ et $T_p = T$. Deux schémas d'intégration en temps sont appliqués à l'EDO : le premier $\mathcal{F}$ intègre une grille fine et le second $\mathcal{G}$ intègre une grille grossière. L'intervalle de temps est découpé en sous domaines aux mêmes temps T_i pour la grille fine et la grille grossière, ainsi les T_i correspondent sur les deux grilles. On effectue un processus itératif pour trouver les valeurs initiales $y(T_i^+) = \lambda_i^{(0)}$ de l'EDO en chaque temps T_i^+ pour $i = 1, \ldots, p$, grâce à l'opérateur de grossier $\mathcal{G}$, puis on effectue une correction itérative donnée par :

$$\lambda_{i+1}^{(k+1)} = \mathcal{G}(T_{i+1}^-, \lambda_i^{(k+1)}) - \mathcal{G}(T_{i+1}^-, \lambda_i^{(k)}) + \mathcal{F}(T_{i+1}^-, \lambda_i^{(k)}) \tag{2.1}$$

On remarque que la méthode génère une suite de λ_{i+1} qui vérifie $\lambda_{i+1}^{(k+1)} = \mathcal{F}(T_{i+1}^-, \lambda_i^{(k)})$. Au temps T_i, l'approximation atteindra la précision de l'opérateur de grille fine $\mathcal{F}$. Dans [2], un lien est fait entre les méthodes à valeurs frontières (BVM)[27] et la méthode Pararéal.

Parmi la famille des méthodes utilisant une grille grossière et une correction par une grille fine, il existe la méthode de Pita[40]. Dans [50], un formalisme par méthode de tirs et une analyse de ces méthodes ont été faits.

La méthode PAWSAR [44, 62] est la première tentative de résoudre l'équation de la chaleur en parallèle avec une décomposition de type Schwarz combinant la discrétisation en temps et en espace. Dans cette méthode, la solution sur plusieurs pas de temps est résolue en parallèle, en faisant les itérations de Schwarz en espace et avançant en temps sans atteindre la convergence du Schwarz. Puis cette fenêtre de plusieurs pas de temps est accélérée par la technique d'accélération de la convergence d'Aitken.

Dans cette partie on s'intéressera uniquement à la décomposition de domaine en temps pour les équations différentielles ordinaires. Le développement de méthodes parallèles pour résoudre des EDOs est un problème difficile parce que les intégrateurs en temps usuels sont séquentiels. c'est à dire que la solution au temps précédent est requise dans le schéma pour obtenir la solution du pas de temps en cours. Considérons le problème de Cauchy pour

l'EDO du premier ordre suivante :

$$\begin{cases} \dot{y} = f(t, y(t)),\ t \in]0, T] \\ y(0) = \alpha_0 \in \mathbb{R}^d \end{cases} \tag{2.2}$$

L'intervalle de temps $[0, T]$ est découpé en p tranches de temps (ou sous-domaines) $S_i = [T_{i-1}^+, T_i^-]$, avec $T_0 = 0$ et $T_p = T$(voir la figure 2.1).

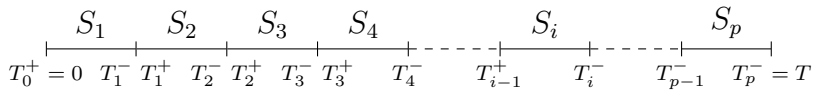


FIGURE 2.1 – Découpage en tranches de temps de l'intervalle $[0, T]$

Le but est de briser la séquentialité du processus d'intégration en temps présent dans les intégrateurs numériques. La difficulté est de raccorder les solutions $y_i(t)$ définies sur S_i aux frontières T_{i-1}^+ et T_i^-. La plupart des méthodes de décomposition de domaines en temps sont des méthodes de tirs où le saut $y_i(T_i^-) - y_{i+1}(T_i^+)$ est corrigé par une méthode de Newton avec une propagation séquentielle des corrections. La non connaissance de la solution au temps final T ne permet pas de lier ces corrections entre elles de manière déterministe. Sous certaines hypothèses sur le flux $f(t, y(t))$, l'approche proposée consiste à transformer le problème à valeur initiale (IVP) en un problème aux valeurs frontières(BVP). La méthodologie proposée est basée sur la décomposition en tranches de l'intervalle de temps. Dans toutes les méthodes de parallélisation à travers les pas, aucune ne fait intervenir l'algorithme de Schwarz exclusivement en temps. En effet il est alors nécessaire d'avoir un retour d'information du bord du sous-domaine où la solution est inconnue. Les méthodes faisant intervenir une correction à partir d'une grille fine, garde une partie séquentielle due à ce processus de correction.

De ce fait, nous avons travaillé une méthode de décomposition de domaine en temps ne présentant pas de séquentialité, à l'image des méthodes de décomposition de domaine en espace. Le problème est alors résolu en parallèle sur chaque tranche de temps en imposant des conditions de transmission(méthode itérative de Schwarz) ou des conditions de raccord (méthode du complément de Schur) aux interfaces.

Le plan de ce chapitre est le suivant, la section 2.2 présente la transformation d'un IVP en un BVP avec une symétrisation de l'intervalle de temps. La section 2.3 (respectivement 2.4) explique la décomposition de domaine de type Schwarz en temps (respectivement la décomposition en temps de type complément de Schur) pour satisfaire les conditions aux frontières entre

les tranches de temps[60]. On montre comment construire explicitement le système bloc tridiagonale satisfait par les conditions de bord des domaines de temps. Une analyse de la méthode de Schwarz est faite pour l'EDP de la chaleur. Dans la section 2.5, on discute les performances parallèles de la méthode ainsi que la comparaison avec la méthode parareal.

2.2 Transformation du problème IVP en un problème BVP sur l'intervalle de temps symétrisé

Soit $f(t, y(t)) \in \mathbb{C}^1(\mathbb{R}^+, \mathbb{R}^n)$, l'IVP avec un système d'EDO du premier ordre est transformé en une BVP d'un système d'EDO du second ordre.

$$\begin{cases} \ddot{y}(t) &= g(t, y(t)) \stackrel{def}{=} \dfrac{\partial f}{\partial t}(t, y(t)) + f(t, y(t))\dfrac{\partial f}{\partial y}(t, y(t)),\ t \in]0, T[, \\ y(0) &= \alpha_0 \\ y(T) &= \beta \end{cases} \tag{2.3}$$

La difficulté est que β n'est pas donné par le problème IVP. La transformation n'est donc pas suffisante pour briser la séquentialité du processus d'intégration sur S_{i+1} i.e. le résultat de l'intégration sur S_i est nécessaire pour définir la condition de bord pour l'intégration sur la tranche suivante S_{i+1}. L'idée principale est de symétriser l'intervalle de temps et d'appliquer intégration rétrograde sur la partie symétrisée.

Pour simplifier, considérons un seul domaine S_1. Afin de contourner la non connaissance de la solution au temps final β, on propose d'appliquer une méthode de décomposition de type Schwarz multiplicatif sans recouvrement à l'intervalle de temps symétrisé $S_1 \cup \bar{S}_1$ (où $\bar{S}_1 = [T, 0]$) de taille $2T$, avec des conditions de transmission de Neumann-Dirichlet en T (voir fig.2.2).

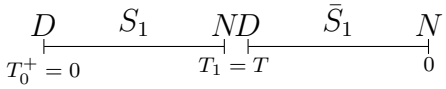


FIGURE 2.2 – Définition des sous-domaines sur l'intervalle symétrisé avec des conditions de bords de Dirichlet(D) et de Neumann(N) pour la méthode de Schwarz

Le problème est symétrisé pour obtenir des conditions de bords bien définies. La propriété de ρ-réversibilité des systèmes partitionnés justifie l'intégration rétrograde avec des schémas symétriques sur la partie symétrique

du problème. Pour l'équation différentielle autonome $\dot{y} = f(y)$ et le champ vectoriel $f(y)$ sont dits $\rho - reversible$ (ρ étant une transformation linéaire inversible dans l'espace de phase) si

$$\rho f(y) = -f(\rho y), \ \forall \ y \tag{2.4}$$

Définition 2.1. Le flux par rapport à t, est la fonction qui , à tout point y_0 de l'espace, associe la valeur $y(t)$ de la solution en partant de $y(0) = y_0$. On note Φ_t, la fonction défini par

$$\Phi_t(y_0) = y(t) \quad si \quad y(0) = y_0$$

Définition 2.2. Le flux du système, noté $\phi(y)$ est dit $\rho - reversible$ si

$$\rho \circ \phi = \phi^{-1} \circ \rho \tag{2.5}$$

Un exemple important est le système partitionné suivant :

$$\begin{cases} \dot{v} = g(u, v) \\ \dot{u} = f(u, v) \end{cases} \tag{2.6}$$

où $f(u, -v) = -f(u, v)$ et $g(u, -v) = g(u, v)$. Ici la transformation ρ est donnée par $\rho(u, v) = (u, -v)$. Alors toutes équations différentielles du second ordre $\ddot{u} = g(u)$ écrites comme $\dot{u} = v$, $\dot{v} = g(u)$ sont réversibles. Si l'équation différentielle est réversible, il faut alors que les méthodes numériques utilisées préservent la propriété de ρ-réversible de l'équation.

Méthodes d'intégration symétrique

On rappelle les résultats suivants [53],

Définition 2.3. On appelle la fonction $\phi_h : y_n \mapsto y_{n+1}$ le flux numérique ou discrétisé. La formule de la fonction ϕ est une méthode d'intégration numérique.

Définition 2.4. Pour l'intégration numérique de (2.2), une méthode numérique à un pas ϕ_h est dite symétrique ou rétrograde, si elle satisfait

$$\phi_h \circ \phi_{-h} = id \ ou \ \phi_h = \phi_{-h}^{-1} \tag{2.7}$$

où id représente la fonction identité.

Théorème 2.1. *Si une méthode numérique, appliquée à une équation différentielle $\rho - reversible$, vérifie*

$$\rho \circ \phi_h = \phi_{-h}^{-1} \circ \rho \tag{2.8}$$

alors le flux numérique ϕ_h est une fonction $\rho - reversible$ si et seulement si ϕ_h est une méthode symétrique.

Démonstration. La conséquence de l'équation (2.8) est que le flux numérique ϕ_h est réversible s'il vérifie la relation par définition $\phi_{-h} \circ \rho = \phi_h^{-1} \circ \rho$. Comme ρ est une transformation réversible, ceci est équivalent à la condition de symétrie de la méthode de la définition 2.4. □

Réciproquement, il est aussi vrai qu'une méthode symétrique est $\rho - reversible$ si et seulement si la condition de ρ-compatibilité (2.8) est vérifiée. Comparé à la condition de symétrie de la méthode, la condition (2.8) est beaucoup moins restrictive. Celle-ci est automatiquement satisfaite par la plupart des méthodes de Runge-Kutta(implicites ou explicites) . La méthode d'Euler explicite, $\phi_h(y) = y + hf(y)$, vérifie la condition pour le champ de vectoriel :

$$(\rho \circ \phi_h)(y) = \rho y + h\rho f(y) = \rho y - hf(\rho y) = \phi_{-h}(\rho y)$$

Supposons que l'on ait une équation de la forme $\ddot{y} = g(y)$, qui puisse être réécrite sous forme d'un système partitionné

$$\begin{cases} \dot{y} = z \\ \dot{z} = g(y) \end{cases} \tag{2.9}$$

et la transformation linéaire $\rho(y, z) = (y, -z)$. Considérons alors la forme à un pas du schéma de Stormer-Verlet $\begin{pmatrix} y_{n+1} \\ z_{n+1} \end{pmatrix} = \Phi_h \begin{pmatrix} y_n \\ z_n \end{pmatrix}$,

$$\begin{cases} z_{n+1/2} = z_n + hg(y_n) \\ y_{n+1} = y_n + hz_{n+1/2} \\ z_{n+1} = z_{n+1/2} + hg(y_{n+1}) \end{cases} \tag{2.10}$$

En remplaçant h par $-h$, et (y_n, z_n) par (y_{n+1}, z_{n+1}) on obtient

$$\begin{cases} \tilde{z}_{n+1/2} = z_{n+1} - hg(y_{n+1}) \\ \tilde{y}_n = y_{n+1} - h\tilde{z}_{n+1/2} \\ \tilde{z}_n = \tilde{z}_{n+1/2} - hg(\tilde{y}_n) \end{cases} \tag{2.11}$$

Avec la dernière équation du système (2.10), on a $\Rightarrow \tilde{z}_{n+1/2} = z_{n+1/2}$, alors on déduit que $\tilde{y}_n = y_n$ et finalement $\tilde{z}_{n+1} = z_{n+1}$. Par la définition 2.4 , le schéma de Stormer-Verlet est un schéma symétrique et est adapté pour l'intégration rétrograde. Il existe également des méthodes symétriques multi-pas et de Runge-Kutta [53].

Conditions de bords

En appliquant la décomposition de l'intervalle en temps de la figure2.1 à l'équation (2.3), on obtient les conditions de transmission de Dirichlet-Neumann distribuées entre les p tranches de temps S_i de $[0,T]$ et les p tranches de temps $S_{2i+p} = \overline{S}_i = \overline{[T_{i-1}^+, T_i^-]}$ de $\overline{[0,T]}$:

$$\left\{\begin{array}{rcl} 1 \leq j < p, \\ \dot{y}_j(T_j^-) = \dot{y}_{j+1}(T_j^+), \\ y_{j+1}(T_j^+) = y_j(T_j^-) \end{array}\right. \left\{\begin{array}{rcl} j = p, \\ \dot{y}_p(T_p^-) = -\dot{\overline{y}}_p(T_p^-), \\ \overline{y}_p(T_p^-) = y_p(T_p^-). \end{array}\right. \left\{\begin{array}{rcl} p > j \geq 1, \\ \dot{\overline{y}}_{j+1}(T_j^+) = \dot{\overline{y}}_j(T_j^-), \\ \overline{y}_j(T_j^-) = \overline{y}_{j+1}(T_j^+) \end{array}\right. . \tag{2.12}$$

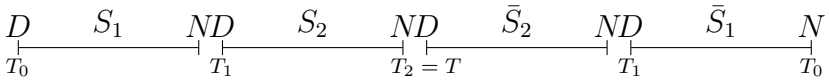


FIGURE 2.3 – Exemple du découpage pour 4 sous-domaines sur l'intervalle symétrisé avec des conditions de bords de Dirichlet(D) et de Neumann(N) pour la méthode de Schwarz

Deux méthodes nous permettent de satisfaire le raccord des solutions :

1. L'algorithme de Schwarz additif (ou multiplicatif) sans recouvrement pour lequel on peut accélérer la convergence vers la solution [43].
2. Un algorithme de Newton qui résout un système bloc tridiagonal pour une système d'EDO linéaire.

Les méthodes de tirs classiques pour la décomposition de domaine en temps [40] sont seulement basées sur des raccords de type Dirichlet entre les tranches de temps parce que le raccord de type Neumann n'est pas possible puisque la condition au temps final n'est pas connue. Notons que la complexité de calcul augmente et qu'il n'y a aucun intérêt à procéder de la sorte pour un calcul séquentiel sur un processeur.

2.3 La méthode de décomposition de domaine de type Schwarz en temps

La première approche proposée pour satisfaire le raccord des solutions est l'algorithme de Schwarz. Nous expliquerons cette méthode pour deux domaines S_1 and $\bar{S}_1$. La méthode de décomposition de domaine de Schwarz pour la m-ième itération avec comme condition initiale $\dot{\overline{y}}_1^0(T^-)$, s'écrit au niveau continu :

$$\text{❶} \quad \begin{cases} \ddot{y}_1^{2m+1} = g(t, y_1^{2m+1}) & t \in S_1 \\ y_1^{2m+1}(0) = \alpha_0 \\ \dot{y}_1^{2m+1}(T^-) = -\dot{\overline{y}}_1^{2m}(T^-) \end{cases}$$

$$\text{❷} \quad \begin{cases} \ddot{\overline{y}}_1^{2m+2} = g(t, \overline{y}_1^{2m+2}) & t \in \bar{S}_1 \\ \dot{\overline{y}}_1^{2m+2}(0) = -f(0, \alpha_0) \\ \overline{y}_1^{2m+2}(T^-) = y_1^{2m+1}(T^-) \end{cases}$$

Le critère d'arrêt est $||\dot{y}_1^{2m+2}(T^-) - \dot{y}_1^{2m}(T^-)||_2 \leq \epsilon$, avec ϵ donné.
La généralisation à $2p$ domaines de temps s'obtient en appliquant le Schwarz pour les domaines impairs puis pour les domaines pairs. On peut donc distribuer les conditions de bords entre les sous-domaines S_i de $[0,T]$ et les p domaines $\overline{S}_i = \overline{[T_{i-1}^+, T_i^-]}$ de $\overline{[0,T]}$ avec des conditions de transmission de Dirichlet-Neumann :

$$\begin{cases} \ddot{y}_{2j+1}^{2m+1}(t) & = & g(t, y_{2j+1}^{2m+1}(t)),\ t \in S_{2j+1} \\ y_{2j+1}^{2m+1}(T_{2j}^+) & = & y_{2j}^{2m}(T_{2j}^-) \\ \dot{y}_{2j+1}^{2m+1}(T_{2j+1}^-) & = & \dot{y}_{2j+2}^{2m}(T_{2j+1}^+) \end{cases} \tag{2.13}$$

$$\begin{cases} \ddot{y}_{2j+2}^{2m+2}(t) & = & g(t, y_{2j+2}^{2m+2}(t)),\ t \in S_{2j+2} \\ y_{2j+2}^{2m+2}(T_{2j+1}^+) & = & y_{2j+1}^{2m+1}(T_{2j+1}^-) \\ \dot{y}_{2j+2}^{2m+2}(T_{2j+2}^-) & = & \dot{y}_{2j+3}^{2m+1}(T_{2j+2}^+) \end{cases} \tag{2.14}$$

Le même principe de numérotation pair-impair des domaines est appliqué pour $\bar{y}_j$ sur $\bar{S}_j$. Une attention spéciale doit être apportée pour la condition Neumann sur S_p :

$$\dot{y}_p^{2m+p\%2+1}(T^-) = -\dot{\overline{y}}_p^{2m+p\%2}(T^-) avec p\%2 = mod(p,2) \tag{2.15}$$

Cette méthode a pour but d'être appliquée sur des centaines de sous-domaines.

2.3.1 Schémas de discrétisation

Dans cette section on développera les schémas numériques utilisés pour la discrétisation de l'équation (2.3). Deux cas apparaissent, le premier quand le problème est un système d'EDO, et le second quand il s'agit d'une équation aux dérivées partielles. Le cas d'un système d'EDO est classique et ne nécessite pas d'attention particulière. En revanche pour les EDP, on doit rester cohérent avec la discrétisation de la dérivée seconde en temps. Puisque l'intégration se fait en avant et en rétrograde, il faut que le schéma de discrétisation utilisé conserve la propriété de réversibilité de l'équation. La discrétisation de l'opérateur d^2/dt^2 se fait par le schéma du second ordre implicite de Stormer-Verlet avec N pas de temps réguliers de taille h ,

$$\frac{d^2y}{dt^2} \approxeq \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} \tag{2.16}$$

La décomposition de domaine de Schwarz est réalisée avec une condition de Dirichlet α en temps pour le bord gauche du domaine, et une condition de Neumann β en temps pour le bord droit. Le bord n'est pas considéré comme une inconnue du système, la condition de Dirichlet est alors écrite implicitement.

$$-2y_1 + y_2 = -\alpha + g(h, y_1) \tag{2.17}$$

On discrétise la condition de Neumann par le schéma de différences finies du second ordre suivant :

$$\frac{3}{2}y_N - 2y_{N-1} + \frac{1}{2}y_{N-2} = h\beta \tag{2.18}$$

Système linéaire

Quand le problème est linéaire, celui-ci s'écrit sous la forme d'un système matriciel. La structure de la matrice du problème dépend de la discrétisation de la dérivée seconde en temps. Le schéma a une dépendance de 3 pas de temps, la matrice a donc une structure tridiagonale par blocs, avec un traitement particulier pour tenir compte du bord Neumann lors de la résolution du système par une méthode LU bloc tridiagonale. La matrice de l'opérateur de dérivée du second ordre en temps est :

$$A = \begin{pmatrix} -2\mathbb{I}_d & 1\mathbb{I}_d & 0 & 0 & \dots & 0 \\ 1\mathbb{I}_d & -2\mathbb{I}_d & 1\mathbb{I}_d & 0 & \dots & 0 \\ 0 & 1\mathbb{I}_d & -2\mathbb{I}_d & 1\mathbb{I}_d & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 1\mathbb{I}_d & -2\mathbb{I}_d & 1\mathbb{I}_d \\ 0 & 0 & 0 & \frac{1}{2}\mathbb{I}_d & -2\mathbb{I}_d & \frac{3}{2}\mathbb{I}_d \end{pmatrix} \tag{2.19}$$

avec $\mathbb{I}_d$ représentant la matrice identité de taille d le nombre d'équations du système IVP.

La fonction second membre du système linéaire s'écrit pour N pas de temps.

$$G = \begin{pmatrix} g_1 & 0 & \dots & 0 \\ 0 & g_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \dots & 0 & g_N \end{pmatrix} \tag{2.20}$$

où g_i représente la matrice de la fonction

$$g(t,y) = \frac{\partial f}{\partial t}(t,y) + f(t,y)\frac{\partial f}{\partial y}(t,y) \tag{2.21}$$

qui est de taille d du système d'IVP.

Les conditions de bords de Dirichlet et Neumann en temps sont dans le membre de droite. $b = \begin{pmatrix} -D\mathbb{I}_d \\ 0 \\ \vdots \\ 0 \\ N\mathbb{I}_d \end{pmatrix}$.

Alors le système peut s'écrire comme $(A - G)Y = b$ avec Y le vecteur solution appartenant à $\mathbb{R}^{N\times d}$

La résolution de ce système linéaire donne la solution (non convergée) pour tous les pas de temps du sous-domaine.

Système non-linéaire

Dans le cas d'un système d'équations non-linéaires, la matrice G précédente est une fonctionnelle. Mais l'opérateur de la dérivée seconde ne change pas donc, la matrice A reste la même.

$$F(Y) = 0 = AY - b - G(Y) \tag{2.22}$$

La méthode de Newton classique pour calculer le zéro de la fonction F s'écrit pour l'itération k,

$$\frac{\partial F}{\partial Y}(Y^k)\delta^k = -F(Y^k),\ Y^{k+1} = Y^k + \delta^k \tag{2.23}$$

où $\frac{\partial F}{\partial Y}(Y^k)$ est la matrice jacobienne de F.

L'expression de la jacobienne de F est

$$J = \frac{\partial F}{\partial Y} = J_1 - J_2 = \frac{\partial AY}{\partial Y} - \frac{\partial G}{\partial Y} \tag{2.24}$$

$$g(t,y) = \frac{\partial f}{\partial t}(t,y) + f(t,y)\frac{\partial f}{\partial y}(t,y) \tag{2.25}$$

$$\begin{aligned} \frac{\partial g}{\partial y}(t,y) &= \frac{\partial^2 f}{\partial y \partial t}(t,y) + \frac{\partial}{\partial y}(f(t,y)\frac{\partial f}{\partial y}(t,y)) && (2.26)\\ &= \frac{\partial^2 f}{\partial y \partial t}(t,y) + \frac{\partial^2 f}{\partial y^2}(t,y)f(t,y) + (\frac{\partial f}{\partial y})^2 && (2.27) \end{aligned}$$

J_2 est la matrice composée des jacobiennes $\frac{\partial g}{\partial y}(t,y)$. J_1 est la jacobienne de l'opérateur linéaire discrétisé $\ddot{y}$, qui est exactement le même opérateur $J_1 = A$.

La méthode de Newton s'écrit alors ,

$$(A - J_2)\delta^k = -(AY^k - b - G(Y^k)) \tag{2.28}$$

En multipliant l'équation précédente par A^{-1} ,

$$\begin{aligned} (I - A^{-1}J_2)\delta^k &= -A^{-1}AY^k + A^{-1}b + A^{-1}g(Y^k) && (2.29)\\ &= A^{-1}g(Y^k) + Y^k - A^{-1}b && (2.30) \end{aligned}$$

La seule partie variable de (2.30) est Y^k, J_2 et $g(Y^k)$.

2.3.2 Résultats numériques

Soit l'EDO suivante :

$$\begin{cases} \dot{y}(t) = \dfrac{1}{sinh(y(t))}, t \in [0,5] \\ y(0) = 2 \end{cases} \tag{2.31}$$

En appliquant la méthode de Schwarz en temps, on obtient le problème aux frontières (2.32)

$$\begin{cases} \ddot{y}(t) = -\dfrac{cosh(y(t))}{sinh(y(t))^3} \\ y(0) = 2, \quad \dot{y}(0) = -\dfrac{1}{sinh(2)} \end{cases} \tag{2.32}$$

La figure 2.4, respectivement 2.6, représentent la solution directe et rétrograde de l'EDO (2.32) obtenue par l'algorithme de décomposition de Schwarz pour deux, respectivement quatre domaines. L'intervalle [0,T] est discrétisé en 50 points.

Les figures 2.5 et 2.7 représentent la convergence de l'algorithme de décomposition de Schwarz associée aux deux cas précédent.

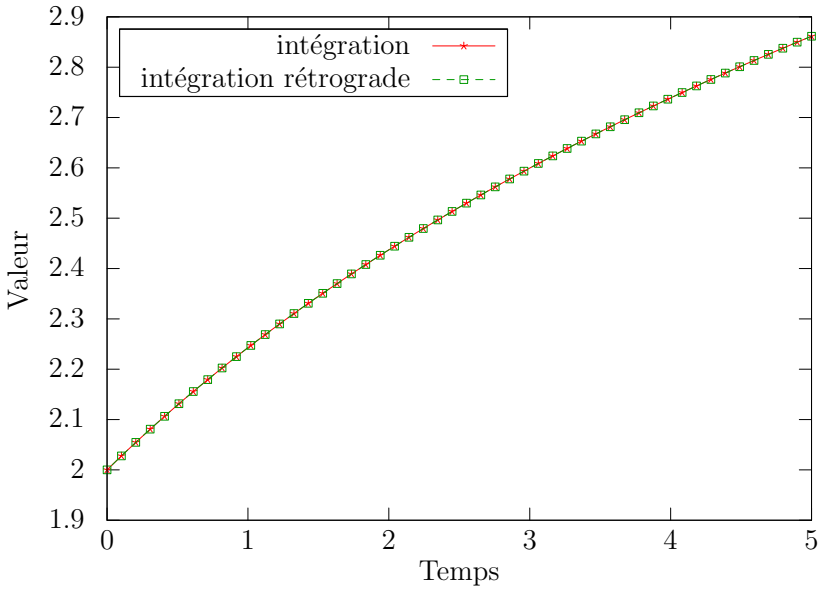


FIGURE 2.4 – Solution directe et rétrograde obtenue par l'algorithme de Schwarz en temps sur deux sous-domaines pour le problème (2.32)

2.3.3 Accélération de la convergence par la technique d'Aitken-Schwarz

Plaçons-nous dans le cas d'un problème linéaire, l'opérateur d'erreur à une convergence ou divergence linéaire qui peut être accéléré par la technique d'Aitken. Cette technique est présentée, ici, car il s'agit de l'accélération de

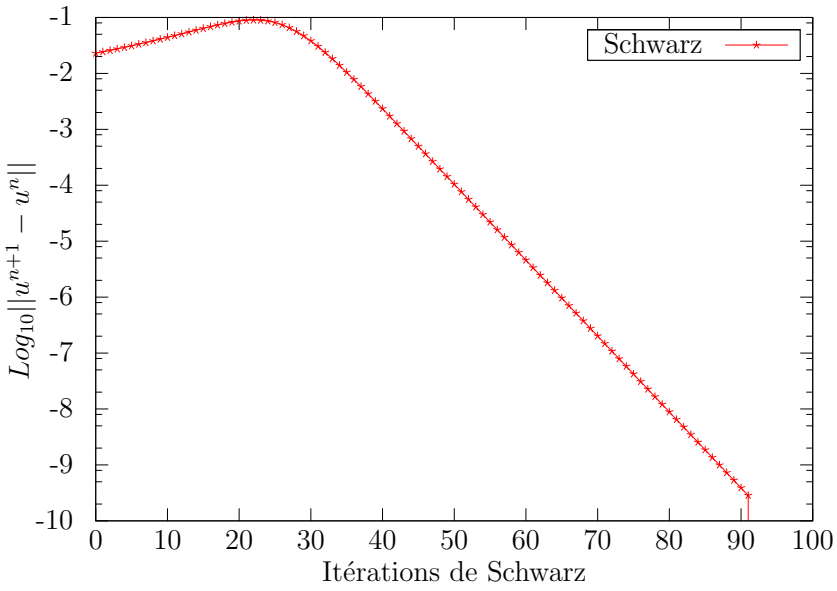


FIGURE 2.5 – Convergence de l'algorithme de Schwarz en temps sur deux sous-domaines pour le problème (2.32)

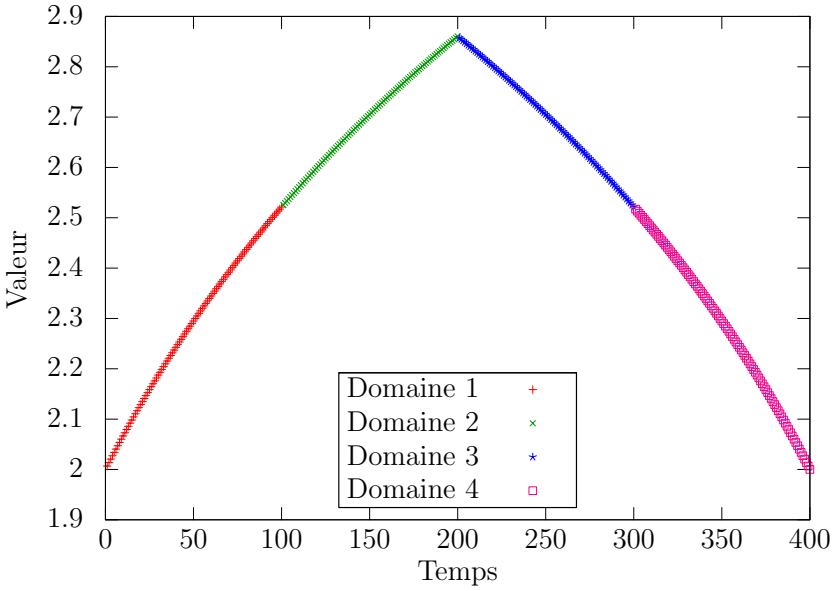


FIGURE 2.6 – Solution directe et rétrograde obtenue par l'algorithme de Schwarz en temps sur quatre sous-domaines pour le problème (2.32)

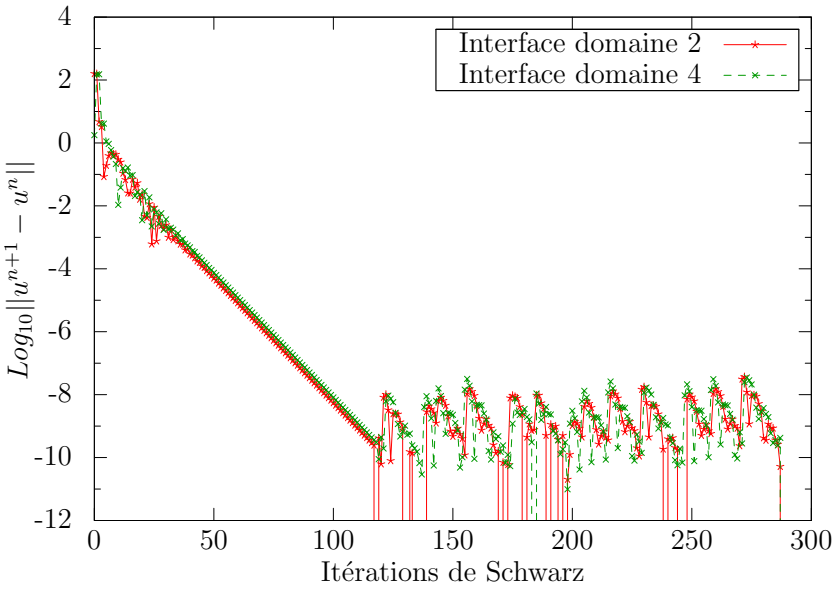


FIGURE 2.7 – Convergence de l'algorithme de Schwarz en temps sur quatre sous-domaines pour le problème (2.32)

suites linéaires, cependant le procédé d'Aitken est détaillé en section 3.1.2. Considérons, au niveau discret, le schéma implicite symétrique de Störmer-Verlet :

$$y_{i,n+1} - 2y_{i,n} + y_{i,n-1} = h^2 g(t_n, y_{i,n}) \tag{2.33}$$

où $y_{i,n}$ est l'approximation de la solution $y(t_n) = y_i(t_n)$ à $t_n \in S_i$ et h le pas de temps constant définit par la discrétisation de S_i par $N+1$ points.

La prochaine étape de la méthode, est de réduire le coût de calcul en diminuant le nombre d'itérations de l'algorithme de Schwarz.

Pour les systèmes linéaires d'EDO, on peut démontrer la convergence ou divergence linéaire de la méthode de Schwarz :

Théorème 2.2. *Soit $\dot{y} = Ay + B, y(0) = \alpha_0 \in \mathbb{R}^d$ un système d'EDO avec A matrice diagonalisable, $B \in \mathbb{R}^d$, alors l'algorithme de Schwarz en temps, appliqué à l'équation du second ordre* (2.3) *sur l'intervalle de temps de longueur $2T$, converge linéairement. La convergence vers la condition de bord exacte peut être accélérée par la technique d'accélération de la convergence d'Aitken.*

Démonstration. Mettons nous dans le cas de deux domaines de temps avec S_1 et $\bar{S}_1$. En utilisant la propriété de diagonalisation de A, il existe une matrice inversible U tel que $U^{-1}A^2U = \Lambda \overset{def}{=} diag(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_d)$. Alors le système d'origine est transformé en un système de d EDO (2.34) où $y = U\tilde{x}$, $c = U^{-1}Ab$,

$\gamma = U^{-1}\alpha_0$, $\beta = -U^{-1}f(0,\alpha_0)$ et en posant $\lambda_i = -(2+\tilde{\lambda}_i h^2)$.

$$\tilde{x}_{p,j}^{i+1,2k+p} + \lambda_j \tilde{x}_{p,j}^{i,2k+p} + \tilde{x}_{p,j}^{i-1,2k+p} = c_{j,p},\ 1 \le i \le N-1,\ 1 \le j \le d,\ p = \{1,2\} \tag{2.34}$$

associé aux conditions de bords :

$$\begin{cases} \tilde{x}_{1,j}^{0,2k+1} = \gamma_j \\ \tilde{x}_{1,j}^{N,2k+1} - \tilde{x}_{1,j}^{N-1,2k+1} = -\tilde{x}_{2,j}^{0,2k} + \tilde{x}_{2,j}^{1,2k} \end{cases}, \begin{cases} \tilde{x}_{2,j}^{0,2k+2} = \tilde{x}_{1,j}^{0,2k+1} \\ \tilde{x}_{2,j}^{N,2k+2} - \tilde{x}_{2,j}^{N-1,2k+1} = \beta_{2,j} \end{cases} \tag{2.35}$$

Par souci de simplicité, on utilisera par la suite une discrétisation du premier ordre pour la condition de bord de Neumann. Une discrétisation d'ordre élevé changera le calcul mais pas la convergence linéaire du Schwarz. Soit $X_{1,2}$ la solution de l'équation caractéristique de la récurrence linéaire (2.34) : $X^2 + \lambda_j X + 1 = 0$. Alors l'erreur pour le domaine S_1 à l'itération $2k+1$ par rapport à l'erreur à $2k-1$ s'écrit :

$$e_{1,j}^{N,2k+1} = \underbrace{\frac{(X_1^N - X_2^N)}{(X_1^N - X_1^{N-1}) - (X_2^N - X_2^{N-1})}\, k_j}_{\rho_{1,j}}\, e_{1,j}^{N,2k-1} \tag{2.36}$$

avec $k_j = \left(1 - \dfrac{(X_1^N - X_1^{N-1})X_2^1 - (X_2^N - X_2^{N-1})X_1^1}{(X_1^N - X_1^{N-1}) - (X_2^N - X_2^{N-1})}\right)$. Le facteur d'amplification $\rho_{j,1}$ ne dépend pas de l'itérée du Schwarz, par conséquent la convergence ou la divergence de l'erreur à la frontière du temps final T^- est linéaire et l'accélération de la convergence d'Aitken (2.37), dont le principe général est rappelé en section 3.1.2, peut être appliquée composante par composante.

$$\tilde{x}_{1,j}^{N,\infty} = (1-\rho_{1,j})^{-1}(\tilde{x}_{1,j}^{N,2k+1} - \rho_{1,j}\tilde{x}_{1,j}^{N,2k-1}) \tag{2.37}$$

L'équation (2.36) peut être généralisée par

$$\tilde{e_j}^{2k+1} = \mathbb{P}_j \tilde{e_j}^{2k-1} \tag{2.38}$$

où $\tilde{e_j}$ est l'erreur du mode j au $p-1$ conditions de bord de Dirichlet des domaines pairs et $\mathbb{P}_j$ ne dépend pas de l'itération de Schwarz. En pratique, le système d'EDO n'est pas diagonalisé et l'accélération est réalisée avec une matrice $\mathbb{P}$ reliant toutes les composantes de la solution au bord Dirichlet des domaines pairs. La matrice $\mathbb{P}$ est bloc diagonale des blocs $\mathbb{P}_j$, peut être calculée numériquement à partir de $d+2$ itérations de Schwarz. La matrice vérifie la relation $\tilde{e}^{2k+1} = \mathbb{P}\tilde{e}^{2k-1}$ où $\tilde{e}$ est l'erreur de la solution aux $p-1$ conditions de bords de Dirichlet des domaines noirs. □

La construction de la matrice $\mathbb{P}$ à partir des erreurs du Schwarz nécessite un nombre d'erreur égal à la taille de la matrice plus une. On peut écrire la relation (2.38) pour la première erreur (2.39) et la seconde erreur (2.40). On déduit alors la première colonne (2.41) de $\mathbb{P}$ à partir de $s+1$ erreurs.

$$\begin{pmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,s} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,s} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{s,1} & \rho_{s,2} & \cdots & \rho_{s,s} \end{pmatrix} \begin{pmatrix} e_1^{n+1} \\ e_2^{n+1} \\ \vdots \\ e_s^{n+1} \end{pmatrix} = \begin{pmatrix} e_1^{n+2} \\ e_2^{n+2} \\ \vdots \\ e_s^{n+2} \end{pmatrix} \tag{2.39}$$

$$\begin{pmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,s} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,s} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{s,1} & \rho_{s,2} & \cdots & \rho_{s,s} \end{pmatrix} \begin{pmatrix} e_1^{n+2} \\ e_2^{n+2} \\ \vdots \\ e_s^{n+2} \end{pmatrix} = \begin{pmatrix} e_1^{n+3} \\ e_2^{n+3} \\ \vdots \\ e_s^{n+3} \end{pmatrix} \tag{2.40}$$

$$\begin{pmatrix} e_1^{n+1} & e_2^{n+1} & \cdots & e_s^{n+1} \\ e_1^{n+2} & e_2^{n+2} & \cdots & e_s^{n+2} \\ \vdots & \vdots & \vdots & \vdots \\ e_1^{n+s} & e_2^{n+s} & \cdots & e_s^{n+s} \end{pmatrix} \begin{pmatrix} \rho_{1,1} \\ \rho_{1,2} \\ \vdots \\ \rho_{1,s} \end{pmatrix} = \begin{pmatrix} e_1^{n+2} \\ e_1^{n+3} \\ \vdots \\ e_1^{n+s+1} \end{pmatrix} \tag{2.41}$$

Méthode plus efficace pour l'accélération d'Aitken

Considérons la méthode de Schwarz multiplicative sur 10 domaines, avec 5 domaines sur l'intervalle $[0,T]$ et 5 domaines sur $\overline{[T,0]}$. On peut écrire une matrice représentant l'influence de l'opérateur sur les conditions de bords. Pour cela, on se place dans la base canonique et l'on observe comment une colonne de la base est transformée par l'opérateur de chaque domaine au cours d'une itération de l'algorithme. La base représente l'erreur du Schwarz à l'interface. Pour pouvoir observer les modifications de l'erreur par l'algorithme de Schwarz, il faut annuler les conditions de bords de l'équation. En effet, l'algorithme de Schwarz est appliqué à l'erreur, pour le premier vecteur de la base canonique (ie. La condition de Neumann $N_1 = 1$ et zéro partout ailleurs.), seule la condition de bord de Dirichlet D_2 sera affectée puisque c'est un voisin direct. En appliquant la technique pour toutes les colonnes de la base canonique, on obtient la matrice $\mathbb{Q}$(2.42).

Puisque les domaines sont géométriquement identiques, et que l'opérateur local est le même pour chaque domaine, les coefficients ρ_{NN}, ρ_{ND},ρ_{DN},ρ_{DD} sont les mêmes pour tous les domaines (où N est pour Neumann, et D pour Dirichlet). Ces coefficients représentent l'influence du premier bord sur le second, selon le numéro de la ligne et de la colonne.

$$\mathbb{Q} = \begin{array}{c} \\ N_1 \\ D_2 \\ N_2 \\ D_3 \\ N_3 \\ D_4 \\ N_4 \\ D_5 \\ N_5 \\ D_6 \\ N_6 \\ D_7 \\ N_7 \\ D_8 \\ N_8 \\ D_9 \\ N_9 \\ D_{10} \end{array} \begin{array}{c} \begin{array}{cccccccccccccccccc} N_1 & D_2 & N_2 & D_3 & N_3 & D_4 & N_4 & D_5 & N_5 & D_6 & N_6 & D_7 & N_7 & D_8 & N_8 & D_9 & N_9 & D_{10} \end{array} \\ \begin{pmatrix} 0 & \rho_{DN} & \rho_{NN} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \rho_{ND} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho_{DN} & \rho_{NN} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \rho_{DD} & \rho_{ND} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho_{DN} & \rho_{NN} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho_{DD} & \rho_{ND} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DN} & \rho_{NN} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho_{DD} & \rho_{ND} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\rho_{DN} & -\rho_{NN} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DD} & \rho_{ND} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DN} & \rho_{NN} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DD} & \rho_{ND} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DN} & \rho_{NN} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DD} & \rho_{ND} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DN} & \rho_{NN} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DD} & \rho_{ND} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DN} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DD} & \rho_{ND} & 0 \end{pmatrix} \end{array} \tag{2.42}$$

$$\mathbb{P} = \begin{array}{c} \\ D_2 \\ N_2 \\ D_4 \\ N_4 \\ D_6 \\ N_6 \\ D_8 \\ N_8 \\ D_{10} \end{array} \begin{array}{c} \begin{array}{ccccccccc} D_2 & N_2 & D_4 & N_4 & D_6 & N_6 & D_8 & N_8 & D_{10} \end{array} \\ \begin{pmatrix} \rho_{ND}\rho_{DN} & \rho_{ND}\rho_{NN} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \rho_{DD}\rho_{DN} & \rho_{ND}\rho_{DN} & \rho_{DN}\rho_{NN} & {\rho_{NN}}^2 & 0 & 0 & 0 & 0 & 0 \\ {\rho_{DD}}^2 & \rho_{DD}\rho_{ND} & \rho_{ND}\rho_{DN} & \rho_{ND}\rho_{NN} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho_{DD}\rho_{DN} & \rho_{ND}\rho_{DN} & -\rho_{DN}\rho_{NN} & -{\rho_{NN}}^2 & 0 & 0 & 0 \\ 0 & 0 & {\rho_{DD}}^2 & \rho_{DD}\rho_{ND} & -\rho_{ND}\rho_{DN} & -\rho_{ND}\rho_{NN} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho_{DD}\rho_{DN} & \rho_{ND}\rho_{DN} & \rho_{DN}\rho_{NN} & {\rho_{NN}}^2 & 0 \\ 0 & 0 & 0 & 0 & {\rho_{DD}}^2 & \rho_{DD}\rho_{ND} & \rho_{ND}\rho_{DN} & \rho_{ND}\rho_{NN} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \rho_{DD}\rho_{DN} & \rho_{ND}\rho_{DN} & \rho_{DN}\rho_{NN} \\ 0 & 0 & 0 & 0 & 0 & 0 & {\rho_{DD}}^2 & \rho_{DD}\rho_{ND} & \rho_{ND}\rho_{DN} \end{pmatrix} \end{array} \tag{2.43}$$

En élevant la matrice $\mathbb{Q}$ au carré puis en prenant la restriction de celle-ci aux composantes des domaines pairs, on obtient la matrice $\mathbb{P}$ pour une itération de l'algorithme de Schwarz multiplicatif. Comme dans la section précédente, la matrice $\mathbb{P}$ satisfait l'équation (2.38). La dépendance de données dans la matrice est toujours de 4, ce qui se traduit par le fait que l'erreur sur une condition affectera seulement quatre autres conditions interfaces après une itération du Schwarz. La matrice a une structure bloc tridiagonal comme le montre l'équation (2.43).

Cette approche est plus efficace que la précédente qui construit la matrice pleine à partir des erreurs. En effet, on économise le calcul de $s = d \times (p-1)$ itérations de Schwarz. Dans le cas présent, avec un découpage homogène, les domaines ont les mêmes coefficients ρ_{ij}, alors seulement 4 coefficients sont donc nécessaires à la construction de la matrice d'accélération $\mathbb{P}$. Le calcul de ces derniers requiert deux résolutions du problème local. Pour un système d'EDO, les coefficients de la matrice deviennent des blocs de la taille du système. La structure de la matrice $\mathbb{Q}$ et celle de la matrice $\mathbb{P}$ restent les mêmes par blocs. Quand la taille de l'interface devient grande, le calcul des coefficients peut être parallélisé en distribuant les composantes de la base sur p processeurs.

Test numérique pour la construction de la matrice $\mathbb{P}$

On applique la méthode d'Aitken-Schwarz en temps au système linéaire d'EDO (2.44) avec des conditions initiales bien choisies, afin d'illustrer la convergence linéaire de la méthode.

$$\begin{cases} \dot{y} = \begin{pmatrix} 0 & -a & b & -c \\ a & 0 & -b & -c \\ -b & b & 0 & -d \\ c & c & d & 0 \end{pmatrix} \\ a = \dfrac{k_1}{\sqrt{2}}, b = \dfrac{k_1}{2}, c = \dfrac{k_1}{2}, d = \dfrac{k_2}{\sqrt{2}} \\ y(0) = (1.0606, 03535, 0.5, 0.7071)^t \end{cases} \tag{2.44}$$

La figure 2.8 montre la convergence linéaire de la méthode de Schwarz pour le système linéaire d'EDO (2.44), avec l'accélération d'Aitken construite à partir des erreurs. Commençant par une erreur de l'ordre de 1 à la première itération, l'accélération d'Aitken donne la solution à l'interface avec une précision supérieur à 10^{-8} après 6 itérations de Schwarz.

Dans la section suivante, nous analyserons le comportement de l'algorithme de Schwarz en temps pour l'équation de la chaleur linéaire.

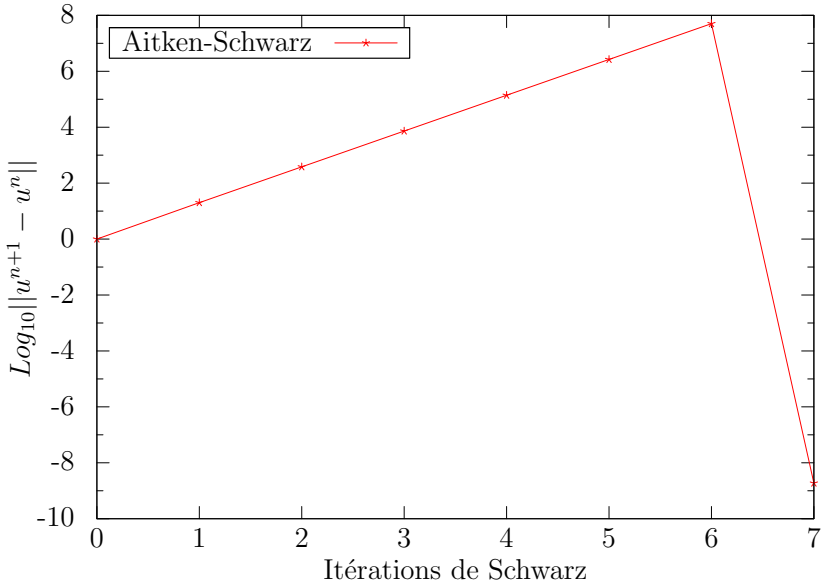


FIGURE 2.8 – Convergence de l'algorithme Aitken-Schwarz appliqué à un système d'EDO modélisant l'oscillateur harmonique pour deux domaines S_1 et $\bar{S}_1$

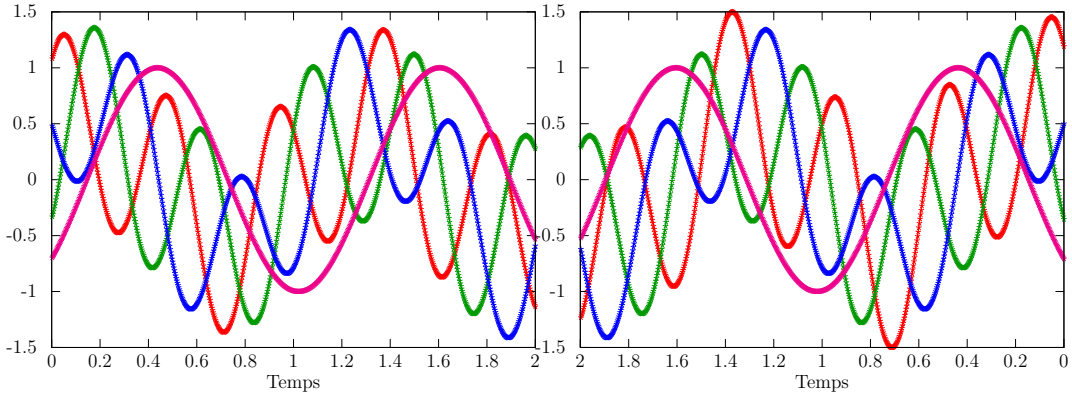


FIGURE 2.9 – Solution directe et rétrograde obtenue par l'algorithme Aitken-Schwarz pour le problème de l'oscillateur harmonique sur $[0, 2]$.

2.3.4 Analyse de la convergence de l'algorithme de Schwarz en temps pour l'EDP de la chaleur linéaire

Soit l'équation de la chaleur en 1D sur $x \in [0, \pi]$ avec des conditions de bords homogènes de Dirichlet :

$$\begin{aligned} \frac{\partial y}{\partial t}(x,t) &= \frac{\partial^2 y}{\partial x^2}(x,t) = f(y(x,t),x,t),\ (x,t) \in [0,\pi]\times]0,T], && (2.45)\\ y(0,t) &= y(\pi,t) = 0, \forall t \in [0,T], && (2.46)\\ y(x,0) &= \alpha_0(x), \forall x \in [0,\pi]. && (2.47) \end{aligned}$$

L'IVP en temps est transformé pour obtenir le problème aux frontières en temps suivant :

$$\begin{aligned} \frac{\partial^2 y}{\partial t^2}(x,t) &= \frac{\partial^4 y}{\partial x^4}(x,t),\ (x,t) \in [0,\pi]\times]0,T], && (2.48)\\ y(x,0) &= \alpha_0(x), \forall x \in [0,\pi], && (2.49)\\ \frac{\partial y}{\partial t}(x,T) &= \beta && (2.50)\\ y(0,t) &= y(\pi,t) = 0, \forall t \in [0,T], && (2.51)\\ \text{avec} &\quad \text{une condition } \pi\text{-périodique en } x\ . && (2.52) \end{aligned}$$

On applique alors le Schwarz en temps sur l'intervalle symétrisé avec comme condition de Neumann $\frac{\partial \bar{y}_1}{\partial t}(x,0) = -f(\alpha_0(x),x,0)$ on $\bar{S}_1$. Pour la discrétisation, on utilise le schéma implicite symétrique de Störmer-Verlet, avec un pas de temps h constant comme suit :

$$\begin{aligned} y_{i,j}^{(n+1)} - 2y_{i,j}^{(n)} + y_{i,j}^{(n-1)} &= \frac{h^2}{\Delta x^4}(y_{i,j-2}^{(n)} - 4y_{i,j-1}^{(n)} + 6y_{i,j}^{(n)} \cdots \\ &\quad -4y_{i,j+1}^{(n)} + y_{i,j+2}^{(n)}), \forall 1 \le j \le N-1 && (2.53)\\ y_{i,0}^{(n)} = y_{i,N}^{(n)} = 0, y_{i,-k}^{(n)} &= y_{i,N-k}^{(n)},\ y_{i,N+k}^{(n)} = y_{i,k}^{(n)}, \forall 1 \le k \le N-1 && (2.54) \end{aligned}$$

où $y_{i,j}^{(n)}$ est l'approximation de la solution $y(t_n) = y_i(t_n, x_j)$ à $t_n = T_{i-1}^+ + nh$ et $x_j = j\Delta x$, $\Delta_x = \frac{\pi}{N}$. Dans la suite, on développera l'analyse de Fourier de cet algorithme de Schwarz en temps au niveau continu afin de montrer que sa convergence est linéaire (i.e. le taux de convergence est constant et ne dépend pas de l'itération).

Analyse de Fourier de la décomposition en temps par Schwarz

Pour garder les calculs simples, considérons le cas de deux domaines S_1 et $\bar{S}_1$. Le cas de plusieurs sous-domaines est l'extension immédiate des calculs

suivants. Soit $\hat{y}_k$ (respectivement $\hat{\alpha}_k$ et $\hat{\xi}_k$) les k^{ieme} mode de la décomposition de Fourier de la solution dans la direction x. (respectivement $\alpha_0(x)$ et $-f(0,\alpha_0(x))$ sur $[0,\pi]$ et sur les domaines S_1 et $\bar{S}_1$). Le mode satisfait l'EDO suivante :

$$\ddot{\hat{y}}_{k,i}^{2m+i}(t) = k^4 \hat{y}_{k,i}^{2m+i}(t),\ t \in S_1 \text{ for } i=1,\ t \in \bar{S}_1 \text{ for } i=2 \tag{2.55}$$

$$\hat{y}_{k,1}^{2m+1}(0) = \hat{\alpha}_k, \text{ and } \dot{\hat{y}}_{k,1}^{2m+1}(T) = \hat{\gamma}_k^{2m} \tag{2.56}$$

$$\hat{y}_{k,2}^{2m+2}(T) = \hat{\eta}_k^{2m+1}, \text{ and } \dot{\hat{y}}_{k,2}^{2m+2}(0) = \hat{\xi}_k \tag{2.57}$$

où $\hat{\gamma}_k^{2m}$ et $\hat{\eta}_k^{2m+1}$ sont les coefficients de Fourier des conditions de transmission de Neumann et Dirichlet. L'erreur de l'algorithme de Schwarz s'écrit :

$$\hat{e}_{k,i}^{2m+i}(t) = C_{1,i}^{2m+i}\exp(k^2 t) + C_{2,i}^{2m+i}\exp(-k^2 t),\ i=\{1,2\} \tag{2.58}$$

Notons $\hat{e}_{\eta,k}^{2m+1}$ et $\hat{e}_{\gamma,k}^{2m}$ l'erreur aux conditions interfaces entre la solution itérée et la solution convergée. Les constantes $C_{1,i}^{2m+i}$ et $C_{2,i}^{2m+i}$ sont définies comme les solutions des systèmes suivants :

$$\begin{pmatrix} C_{1,1}^{2m+1} \\ C_{2,1}^{2m+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ k^2\exp(k^2T) & -k^2\exp(-k^2T) \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \hat{e}_{\gamma,k}^{2m} \end{pmatrix} \tag{2.59}$$

$$\begin{pmatrix} C_{1,2}^{2m+2} \\ C_{2,2}^{2m+2} \end{pmatrix} = \begin{pmatrix} \exp(k^2T) & \exp(-k^2T) \\ k^2 & -k^2 \end{pmatrix}^{-1} \begin{pmatrix} \hat{e}_{\eta,k}^{2m+1} \\ 0 \end{pmatrix} \tag{2.60}$$

La suite $(\hat{e}_{\eta,k}^{2m+1})$ correspondant à l'erreur de la condition de Dirichlet en T dans la méthode de Schwarz en temps satisfait l'équation suivante :

$$\hat{e}_{\eta,k}^{2m+3} = \hat{e}_{\eta,k}^{2m+1}\tanh^2(k^2T) = \rho_{\eta,k}\hat{e}_{\eta,k}^{2m+1} \tag{2.61}$$

Pour $\hat{e}_{\gamma,k}$ le facteur d'amplification $\rho_{\gamma,k}$ est le même. On note que le facteur d'amplification de l'itéré de Schwarz est proche de 1 pour de grandes valeurs de k^2T causant la convergence lente de l'algorithme. Néanmoins le mode k de Fourier a un comportement en $\hat{y}_k(t) = \exp(-k^2t)\hat{y}_k(0)$ pour l'équation de la chaleur. Donc pour de grandes valeurs de k^2T, le facteur d'amplification de l'algorithme de Schwarz sera compensé par la décroissance exponentielle $-k^2T$ du mode de Fourier. Par conséquent, les grands modes n'auront pas d'influence à l'interface. Cette propriété impose d'initialiser l'algorithme de Schwarz avec des conditions de Neumann homogènes à l'interface.

Pour des petites valeurs de k^2T, comme la convergence du Schwarz en temps Eq. (2.61) est purement linéaire, la convergence de l'algorithme vers la bonne solution aux interfaces des sous-domaines peut être accélérée par la technique d'Aitken[43].

L'accélération est calculée par la formule :

$$\hat{y}^{\infty}_{k,2} = (1-\rho_k)^{-1}(\hat{y}^{2m+3}_{k,2} - \rho_{\eta,k}\hat{y}^{2m+1}_{k,2}) \tag{2.62}$$

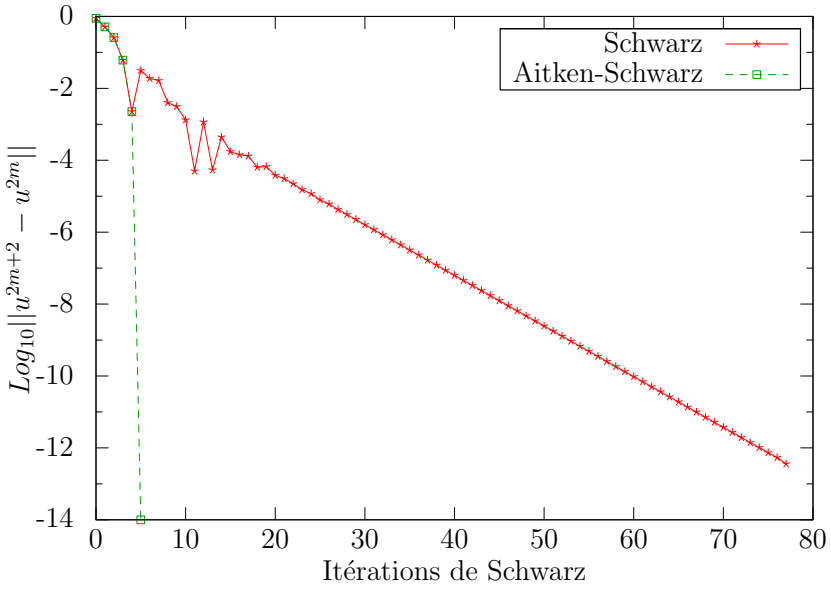


FIGURE 2.10 – Convergence de l'algorithme Aitken-Schwarz pour quatre sous-domaines S_1, S_2 et $\bar{S}_1, \bar{S}_2$ pour le mode $k = 1$ de Fourier de l'équation de la chaleur dans l'espace de Fourier.

La figure 2.10 présente les résultats numériques de la convergence du Schwarz en temps et de celle accélérée par la technique d'Aitken pour le problème (2.55) discrétisé par le schéma de Störmer-Verlet implicite. Les paramètres sont de quatre domaines S_1, S_2 et $\bar{S}_1, \bar{S}_2$, un pas de temps $h = 0.01$ de 0 jusqu'à $T = 1$ et $k = 1$. Les différences entre deux itérés consécutifs des conditions interfaces des domaines pairs (ici trois conditions interfaces) ont été utilisées pour effectuer la technique d'Aitken, menant à la précision numérique de 10^{-14} en 4 itérations. Le nombre d'itérations nécessaires pour appliquer l'accélération de la convergence peut être réduit à deux, comme présenté dans la section suivante. Nous résoudrons le problème (2.45)-(2.47) avec l'algorithme de décomposition de domaine de Schwarz en temps dans l'espace physique, avec comme condition initiale $\alpha_0(x) = (x-\pi)x\exp(-2(x-\pi)^2)$, ce choix permet de respecter les conditions homogènes tout en couplant plusieurs modes dans la décomposition de Fourier.

Résultats numériques dans l'espace physique

Soit le système discrétisé (2.53)-(2.54). On procède en collectant les conditions interfaces pour les domaines pairs mises bout à bout dans un vecteur pour chaque itération de Schwarz. Les vecteurs correspondant à plusieurs itérations sont transformés dans l'espace de Fourier. L'opérateur de l'équation de la chaleur étant séparable, chaque mode de Fourier est accéléré indépendamment des autres. Le vecteur erreur sur les conditions interfaces satisfait pour chaque mode un système linéaire $(2p-1)\times(2p-1)$ $\tilde{\hat{e}}_k^{2m+2} = \mathbb{P}_k \tilde{\hat{e}}_k^{2m}$ (pour quatre domaines sur l'intervalle de temps symétrisé, le vecteur d'erreur est $\tilde{\hat{e}}_k^{2m+2} = (\hat{e}_{\eta,k,2}^{2m+2}, \hat{e}_{\gamma,k,2}^{2m+2}, \hat{e}_{\eta,k,4}^{2m+2})^T$, où l' indice i de $\hat{e}_{.,k,i}$ représente le numéro du domaine). Chaque colonne de la matrice $\mathbb{P}_k$ représente l'influence d'une composante de $\tilde{\hat{e}}_k$ sur les autres. Si les domaines S_i ont mêmes longueurs, la matrice $\mathbb{P}_k$ possède une structure régulière, et elle peut être construite sur un seul domaine pour obtenir les quatre coefficients constants. Dans la pratique, le calcul des coefficients est fait en ajoutant deux seconds membres au problème local, chacun d'entre eux correspondant au problème homogène avec l'une des deux conditions interfaces étant le k^{ieme} mode de Fourier et l'autre condition mise à zéro. Le calcul des $\mathbb{P}_k$ peut être distribué sur les p domaines, chaque domaine ne calculant qu'un seul mode. Soit $\tilde{\hat{y}}_k^{\infty}$, où k est le mode de Fourier, le vecteur des conditions interfaces convergées. L'accélération de la convergence du Schwarz est effectuée par la technique d'Aitken vectorielle : $\tilde{\hat{y}}_k^{\infty} = (Id - \mathbb{P}_k)^{-1}(\tilde{\hat{y}}_k^{2m+2} - \mathbb{P}_k \tilde{\hat{y}}_k^{2m})$. On applique la technique seulement pour les modes k de Fourier correspondant aux composantes de $\tilde{\hat{e}}_k^{2m+2}$ plus grandes que 1.10^{-5}. La figure 2.11 montre la convergence de l'algorithme de décomposition de domaine de Schwarz en temps pour le problème de l'équation de la chaleur (2.53)-(2.54) sur quatre domaines $S_1, S_2, \bar{S}_2$ et $\bar{S}_1$. Les paramètres pour ce test sont $N = 40$, $h = 8.10^{-4}$, $T = 0.1$. L'accélération d'Aitken permet d'atteindre, en seulement deux itérations, la précision numérique de 10^{-5} ce qui correspond à la précision numérique de l'approximation par Fourier avec la solution exacte. Ce résultat est concordant avec l'approximation du deuxième ordre des schémas en temps et en espace et des valeurs des paramètres. La figure 2.12 montre la solution calculée sur l'intervalle symétrisé.

La méthode de décomposition de domaine de Schwarz en temps a été développé pour des EDO et EDP. Celle-ci est la première approche visant à satisfaire les conditions de raccords. Une autre approche consiste à appliquer une méthode de complément de Schur, expliquée dans la section suivante.

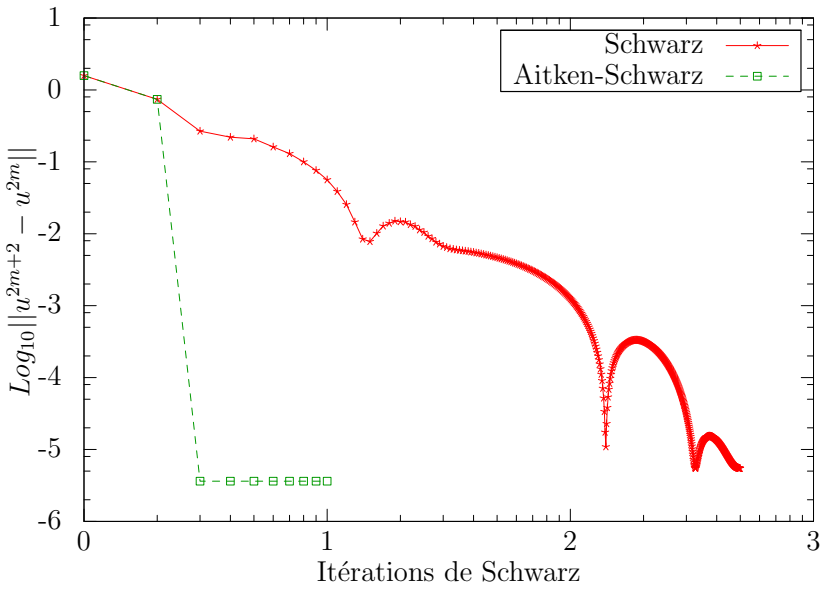


FIGURE 2.11 – Convergence de l'algorithme d' Aitken-Schwarz pour quatre domaines S_1, S_2 et $\bar{S}_1, \bar{S}_2$ pour l'équation de la chaleur dans l'espace physique.

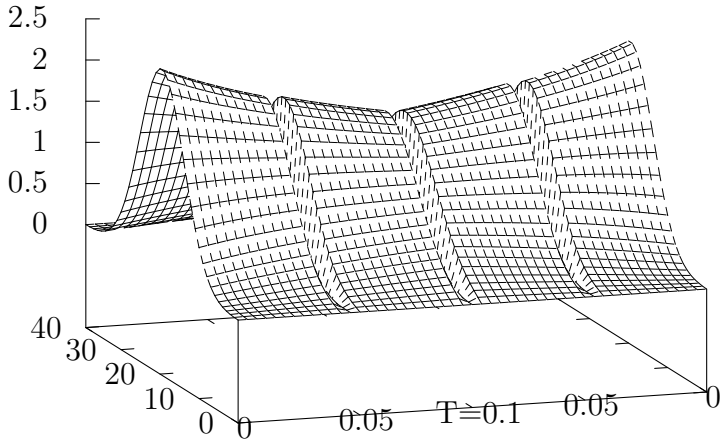


FIGURE 2.12 – La solution de l'équation de la chaleur obtenue par l'algorithme Aitken-Schwarz sur quatre sous-domaines

2.4 Décomposition de domaine en temps par le complément de Schur

Considérons l'intégration de (2.3) comme un système IVP partitionné (2.9), et le schéma de Störmer-Verlet explicite (2.10) avec le pas de temps h qui s'écrit sous forme condensée :

$$\begin{cases} y_{n+1} = y_n + hz_n + \dfrac{h^2}{2} g(t_n, y_n) \\ z_{n+1} = z_n + \dfrac{h}{2} g(t_n, y_n) + \dfrac{h}{2} g(t_{n+1}, y_{n+1}) \end{cases} \tag{2.63}$$

Ce schéma étant symétrique il permet l'intégration en avant (indicé F) ou rétrograde (indicé B) jusqu'au milieu de S_i (respectivement $\overline{S}_i$) (voir la figure 2.13).

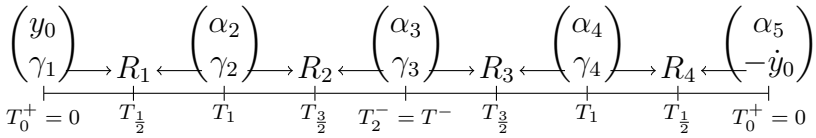


FIGURE 2.13 – Schéma d'intégration en temps direct et rétrograde avec les conditions de raccords pour 4 sous-domaines pour l'intervalle symétrisé

Les raccords R_i de ces deux solutions à $T_{i-\frac{1}{2}} = (T_{i-1}^+ + T_i^-)/2$ relient la condition de bord de Dirichlet $\alpha_i, i = 2, \dots 2p$, et celle de Neumann γ_i, $i = 1, \dots, 2p-1$ comme étant la racine de la fonction non linéaire $F(\gamma_1, \alpha_2, \gamma_2, \dots, \alpha_{2p-1}, \gamma_{2p-1}, \alpha_{2p})$. En partant des conditions de bords γ_i et α_i en T_i^+, l'intégration en avant vers $T_{i+\frac{1}{2}}$ donne $H_F^i(T_{i+\frac{1}{2}}, \gamma_i, \alpha_i)$ tandis que l'intégration rétrograde vers $T_{i-\frac{1}{2}}$ donne $H_B^i(T_{i-\frac{1}{2}}, \gamma_i, \alpha_i)$. La i^{me} composante vectorielle de F peut être calculée indépendamment des autres composantes et s'écrit pour $i = 1, \dots, 2p-1$:

$$F_i(\alpha_{i-1}, \gamma_{i-1}, \alpha_i, \gamma_i) = H_F^{i-1}(T_{i-1+\frac{1}{2}}, \gamma_{i-1}, \alpha_{i-1}) - H_B^i(T_{i-\frac{1}{2}}, \gamma_i, \alpha_i) \tag{2.64}$$

F représente la condition de raccord et peut être assimilée à la méthode du complément de Schur puisqu'un système doit être résolu sur les conditions de bords.

2.4.1 Cas d'un problème linéaire

Pour les systèmes linéaires d'EDO, on a le résultat suivant :

Théorème 2.3. *Soit $\dot{y} = Ay + b, y(0) = \alpha_0 \in \mathbb{R}^d$ un système d'EDO avec A diagonalisable, alors l'algorithme du complément de Schur revient à résoudre un système linéaire tri-diagonale par bloc liant les conditions de bords de Dirichlet et de Neumann des domaines entre elles.*

Démonstration. Sans perte de généralité aucune, on considère l'IVP $\dot{y} = \lambda y$, $y(0) = \alpha_0$, $\lambda \in \mathbb{R}$. Le système d'EDO

$$\begin{cases} \dot{y}_i = z_i \\ \dot{z}_i = \lambda^2 y_i & t \in S_i \\ y_i(T_{i-1}^+) = \alpha_i, z_i(T_{i-1}^+) = \gamma_i \end{cases} \tag{2.65}$$

admet la solution continue $(y_i(t), z_i(t))^T = \alpha_i G_F^i(t) + \gamma_i H_F^i(t)$ où

$$H_F^i(t) = \frac{1}{2|\lambda|} \begin{pmatrix} e^{|\lambda|(t-T_{i-1}^+)} - e^{-|\lambda|(t-T_{i-1}^+)} \\ e^{|\lambda|(t-T_{i-1}^+)} + e^{-|\lambda|(t-T_{i-1}^+)} \end{pmatrix}, \tag{2.66}$$

$$G_F^i(t) = \frac{1}{2} \begin{pmatrix} e^{|\lambda|(t-T_{i-1}^+)} + e^{-|\lambda|(t-T_{i-1}^+)} \\ e^{|\lambda|(t-T_{i-1}^+)} - e^{-|\lambda|(t-T_{i-1}^+)} \end{pmatrix} \tag{2.67}$$

Alors les $2p-1$ conditions de raccords de la solution lient les conditions de bords par le système tridiagonal par bloc suivant $i = 1, \ldots, 2p-1$:

$$G_F^{i-1}(T_{i-\frac{1}{2}})\alpha_{i-1} + H_F^{i-1}(T_{i-\frac{1}{2}})\gamma_{i-1} - G_B^i(T_{i-\frac{1}{2}})\alpha_i - H_B^i(T_{i-\frac{1}{2}})\gamma_i = 0 \tag{2.68}$$

□

En excluant le temps $T_{i\pm\frac{1}{2}}$ de la notation, la figure 2.14 montre la structure tridiagonale par bloc de (2.68).

$$\begin{pmatrix} H_F^0 & -G_B^1 & -H_B^1 & 0 & 0 & 0 & 0 & 0 \\ 0 & G_F^2 & H_F^2 & -G_B^3 & -H_B^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & G_F^4 & H_F^4 & -G_B^5 & -H_B^5 & 0 \\ 0 & 0 & 0 & 0 & 0 & G_F^6 & H_F^6 & -G_B^7 \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \alpha_2 \\ \gamma_2 \\ \alpha_3 \\ \gamma_3 \\ \alpha_4 \\ \gamma_4 \\ \alpha_5 \end{pmatrix} = \begin{pmatrix} -y_0 G_F^0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dot{y}_0 H_B^7 \end{pmatrix}$$

FIGURE 2.14 – Structure bloc tridiagonale du système linéaire représentant les conditions de raccords liant les valeurs frontières des sous-domaines

Remarque 2.1. Numériquement,les matrices $G_F^i(T_{i-\frac{1}{2}})$, $H_F^i(T_{i-\frac{1}{2}})$, $G_B^{i+1}(T_{i+\frac{1}{2}})$, $H_B^{i+1}(T_{i+\frac{1}{2}})$ sont construites en utilisant le schéma d'intégration numérique en utilisant la technique de la variation de la constante.

2.4.2 Cas des problèmes non-linéaires

Pour un problème non linéaire, la méthode développée précédemment ne peut être directement appliquée. La non-linéarité des problèmes locaux est prise en charge par l'intégrateur numérique de l'IVP, qui peut être une méthode de Runge-Kutta ou une méthode de Runge-Kutta partitionnée avec la propriété de réversibilité. La difficulté est alors déplacée sur les conditions de raccords entre les sous-domaines intégrant en avant et en rétrograde. La fonction F (2.69), qui représente le système de conditions de raccords aux frontières est non linéaire. Il faut utiliser une méthode de Newton pour la solution satisfaisant la nullité de F.

$$F\left(\begin{pmatrix}\gamma_1\\ \alpha_2\\ \gamma_2\\ \vdots\\ \alpha_{n-1}\\ \gamma_{n-1}\\ \alpha_n\end{pmatrix}\right) = \begin{pmatrix} W_1(t=R_1, y_0, \gamma_1) & - & W_2(t=R_1, \alpha_2, \gamma_2) \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ W_{n-1}(t=R_n, \alpha_{n-1}, \gamma_{n-1}) & - & W_n(t=R_n, \alpha_n, -\dot{y}_0) \end{pmatrix} = 0 \tag{2.69}$$

où $W_i(t, \alpha, \gamma)$ donne la solution $\begin{pmatrix} y \\ z \end{pmatrix}$ au temps t du i-ème domaine.

Dans le cadre d'un problème non-linéaire, nous cherchons maintenant à savoir comment se comporte la méthode en fonction de la précision de la solution initiale. Le problème linéaire pour ce test est le système Lotka-volterra sur 8 domaines. La solution initiale est calculée à partir de la méthode ode23s de Matlab avec différente tolérance $(10^{-2}, 10^{-3}, 10^{-4}, 10^{-6})$. Celle-ci correspond au couple solution dérivée (α_i, γ_i) de la figure 2.13.

La solution du problème de Lotka-Volterra par la méthode du complément de Schur est donnée par la figure 2.15, tandis que la convergence de la méthode de Newton pour la résolution du zéro de la fonction (2.69) est donnée par la figure 2.17.

On remarque alors que pour une solution initiale assez proche de la solution finale à 10^{-3}, l'algorithme de Newton converge. Il est évident que plus la précision de la solution initiale est élevée, plus le nombre d'itérations de la méthode de Newton sera petit. Nous remarquons que la solution initiale à 10^{-12} n'est pas la solution de la minimisation de la fonctionnelle. En effet la discrétisation et l'intégration étant différentes , la solution n'est pas la même à cause de l'erreur de discrétisation. Cependant en regardant la phase , on se rend compte que l'intégration est correcte.

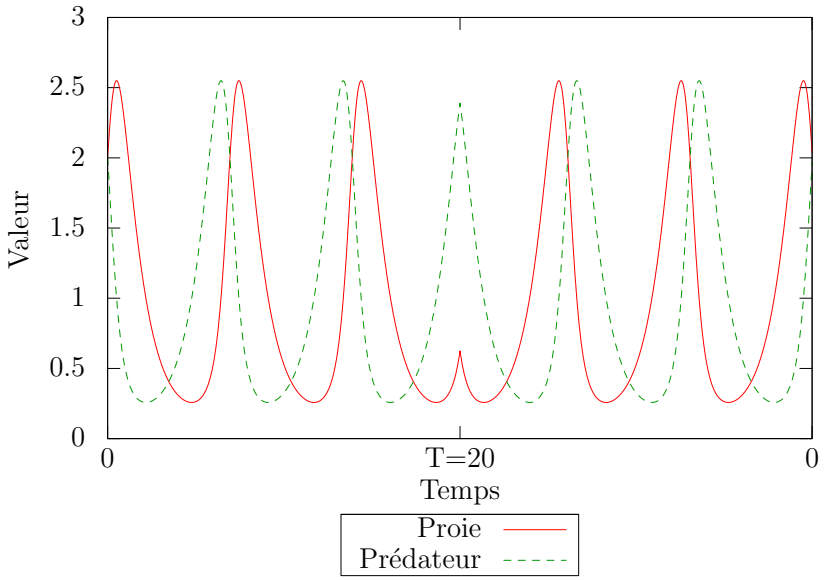


FIGURE 2.15 – Solution du problème de Lotka-Volterra obtenue par l'algorithme du complément de Schur sur 8 sous-domaines avec $T = 20$, avec 100000 pas de temps

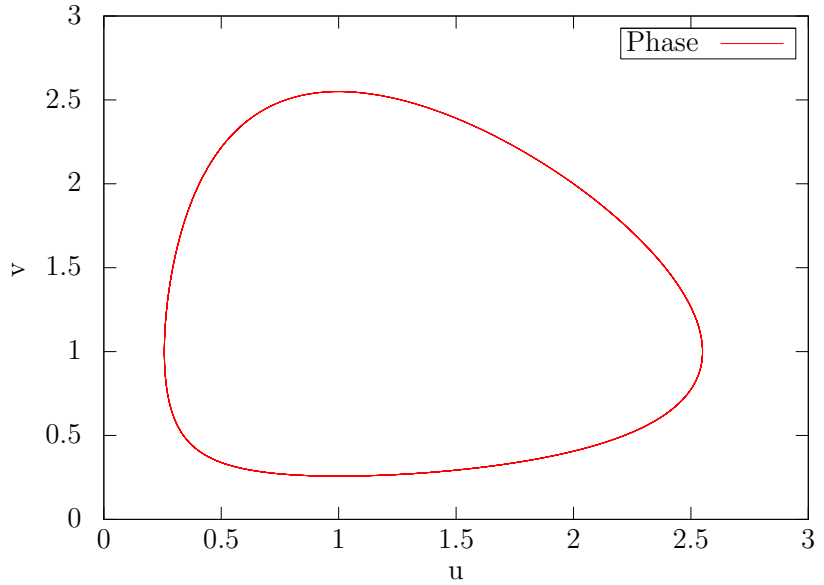


FIGURE 2.16 – Portrait de phase du problème de Lotka-Volterra obtenu du complément de Schur sur 8 domaines avec $T = 20$ avec 100000 pas de temps

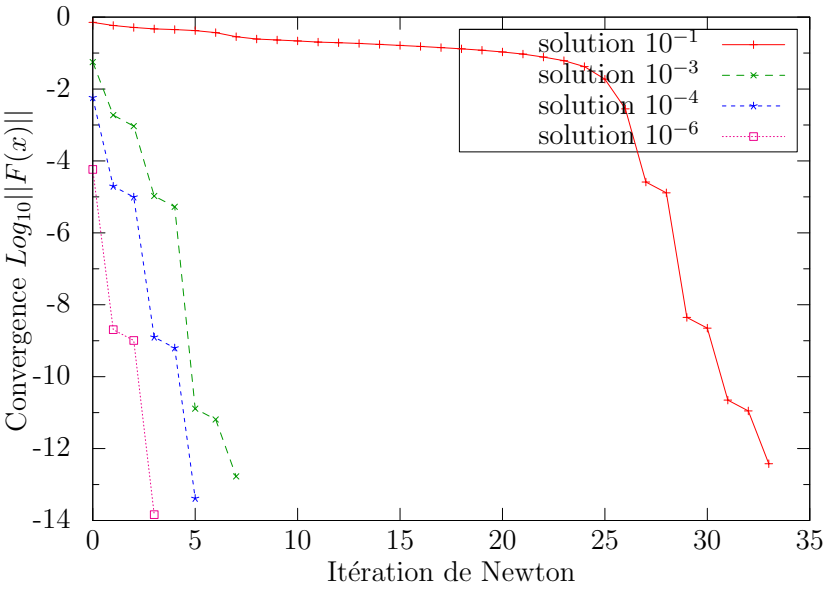


FIGURE 2.17 – Comparaison de la convergence de la méthode du complément de Schur en fonction de la précision de la solution initiale

Dans la section suivante, nous étudions les performances parallèles des deux méthodes de décomposition de domaines en temps que nous venons de développer.

2.5 Performances Parallèles

Dans cette section, nous discutons du comportement parallèle et des performances parallèles associées à la méthode de décomposition de domaine d' Aitken-Schwarz en temps et de la méthode du complément de Schur pour résoudre des systèmes d'EDO linéaires.

Les calculateurs utilisés ont les caractéristiques reportées dans la table 2.1.

Calculateur	Nom	noeuds/ machines	procs/ coeurs	mémoire par coeur	communication
A	SGI-Altix 350	8	2 IA64 1.5Ghz/6Mo	2Go	Numalink
B	SGI Altix ICE	16	2 E5472 3Ghz	4Go	Intel X5472
C	Dell R610	4	E7330 2.4Ghz	8Go	-

TABLE 2.1 – Caractéristiques des calculateurs

Tout d'abord, considérons le système d'EDO suivant, (2.44), (2.71) avec $t \in [0, 40]$, et (2.70) :

$$\begin{cases} \dot{y}_1 = 2y_1 + y_2, \\ \dot{y}_2 = -4y_1 - 3y_2, t \in [0, 5] \text{ avec conditions initiales} \end{cases} \tag{2.70}$$

$$\begin{cases} \dot{y}_1 = -k_1 y_2 \\ \dot{y}_2 = k_1 y_1 \\ \dot{y}_3 = -k_2 y_4 \\ \dot{y}_4 = k_2 y_3 \end{cases} \tag{2.71}$$

Le nombre global de pas de temps sur $[0, T]$, pour résoudre le problème (2.70) avec le schéma de Störmer-Verlet implicite, est de 10^5 et pour le Störmer-Verlet explicite utilisé pour la méthode de Schur le nombre de pas de temps est de 10^7 (respectivement $2\,10^7$ pour le problème (2.44)).

# processus	1	2	4	8	16	32	64	128	256
Calculateur A temps cpu (s)									
Schwarz Eq (2.70)	-	8.3	2.72	1.4	1.1	-	-	-	-
Schur Eq (2.70)	1.74	17	5.25	2.7	1.34	-	-	-	-
Schur Eq (2.44)	5.4	61	22.5	12.2	5.7	-	-	-	-
Calculateur B temps cpu (s)									
Schwarz Eq (2.70)	-	6.28	1.82	3.4	10	13	170	-	-
Schur Eq (2.70)	0.54	2.63	1.60	1.36	0.7	0.35	0.18	0.084	0.04
Schur Eq (2.44)	1.63	13.9	9	8.7	4.4	2.2	1.11	0.6	0.3

TABLE 2.2 – Temps cpu (s) en fonction du nombre de domaines/processeurs pour la méthode d'Aitken-Schwarz et la méthode du complément de Schur pour un système d'EDOs linéaire .

Le tableau 2.2 regroupe les performances en temps cpu obtenues sur les calculateurs A et B pour la méthode d'Aitken-Schwarz en temps avec la matrice $\mathbb{P}$ construite à partir des erreurs interfaces, ainsi que pour la méthode du complément de Schur pour résoudre les problèmes (2.44) et (2.70). Le nombre de domaines sur l'intervalle symétrisé correspond au nombre de processeurs. Le cas 1 processeur est le temps de l'intégration avec le schéma Störmer-Verlet explicite sur $[0, T]$.

Ce tableau montre que :

– l'algorithme d'Aitken-Schwarz a une bonne efficacité parallèle sur le calculateur A jusqu'à 16 processeurs. Sur le calculateur B, l'efficacité

parallèle décroit drastiquement pour le problème (2.70) et il n'y a pas de gain en utilisant plus de 8 processeurs. Les communications globales pour rassembler la matrice $\mathbb{P}$, ainsi que du bruit numérique dû à la technique de construction de la matrice, sont responsables de cette baisse de performance. Néanmoins, sur les deux calculateurs la méthode d'Aitken-Schwarz en temps n'est pas assez performante pour rivaliser avec la résolution de l'IVP.

– le complément de Schur a de bonnes performances parallèles jusqu'à 256 processeurs. Par rapport à la résolution de l'IVP, la méthode devient compétitive pour plus de 16 processeurs sur le calculateur A. Avec 256 processus sur le calculateur B, la méthode atteint un speed-up de 14 pour le problème (2.70), tandis qu'il est de 5 pour le problème (2.44). On note alors que le speed-up de 14 est obtenu "par rapport au calcul sur 2 processus" mais dans ce cas le calcul est double pour résoudre l'EDO du second ordre, cela détériore la performance comparée à la résolution de l'IVP.

TABLE 2.3 – Tableau de performance pour la méthode du complément de Schur

Calculateur C temps cpu (s)							
#processus	2	4	8	16	32	64	128
	16.45	7.92	3.95	2.03	1.1	0.55	0.3

Les résultats suivants concernent l'amélioration de l'implémentation parallèle de ces méthodes. Pour la méthode d'Aitken-Schwarz, la matrice $\mathbb{P}$ est construite à partir des quatre coefficients blocs. L'amélioration pour le complément de Schur vient essentiellement d'une ré-implémentation du schéma d'intégration de Störmer-Verlet, par rapport aux premiers tests du tableau 2.2. Le tableau 2.3 récapitule les performances obtenues pour la méthode du complément de Schur sur le calculateur C. Le temps cpu est divisé par deux quand on multiplie le nombre de processeurs(domaines) par deux. Rappelons que le domaine d'intégration est symétrisé et donc la résolution globale se fait avec 4.10^7 pas de temps. Donc pour 128 processeurs l'intervalle $[0, T]$ est divisé par seulement 64, les 64 autres étant répartis sur la partie symétrique. Le temps cpu pour intégrer avec un processeur (2.3) le problème de l'oscillateur (2.71) avec le Störmer-Verlet explicite est de 1.57s. Par comparaison, le temps pour l'intégration de l'équation (2.71) avec RK2 est $1.93s$.

Le tableau 2.4 montre les temps cpu obtenus pour la méthode de Schwarz en temps avec l'accélération d'Aitken pour le problème (2.71) sur le calculateur C. Les résultats de la première ligne sont pour 2.10^7 pas de temps sur

TABLE 2.4 – Tableau de performance pour la méthode d'Aitken-Schwarz optimale

Calculateur C temps cpu (s)						
nombre de pas de temps /#processus	2	8	16	32	64	128
2.10^7	439	110	70	35	17.5	9.15
2.10^6	43	11.3	7	3.35	1.75	0.9

$[0, T]$, et la seconde ligne les temps cpu pour 2.10^6 pas de temps. En comparaison avec la méthode du complément de Schur, la méthode de Schwarz en temps est 30 fois plus lente. On nuance ce résultat par le fait que l'implémentation faite n'est pas totalement parallèle ; Dès que la résolution du problème pour la construction des coefficients blocs de la matrice $\mathbb{P}$ est assez importante, on peut envisager de distribuer ce calcul sur les processeurs.

Nous avons également comparé nos méthodes avec la méthode parareal.

2.5.1 Comparaison avec la méthode parareal pour un système d'EDO linéaire

Afin d'effectuer une comparaison honnête de la méthode du complément de Schur et la méthode parareal, il faut trouver les meilleurs paramètres pour lesquels ces méthodes se comportent le mieux. La méthode parareal a été testée avec deux opérateurs, d'abord un Euler implicite pour $\mathcal{G}$ et un Euler explicite pour $\mathcal{F}$. Puis avec le schéma de Runge-Kutta explicite d'ordre 2 (RK2) pour $\mathcal{G}$ et $\mathcal{F}$ afin d'avoir de l'ordre 2 en temps et donc avoir la même précision que le complément de Schur qui utilise le schéma de Stormer-Verlet explicite.

La grille fine de parareal est discrétisée avec 2.10^7 pas de temps sur $[0, T]$. La grille grossière est définie par un paramètre de ratio variable par rapport à la grille fine. Le nombre de tranches de temps est défini égal au nombre de pas de temps de la grille grossière. Chaque processeur gère le même nombre de tranches.

Le tableau 2.5 présente les performances de parareal de 1 à 128 processus avec l'Euler implicite (respectivement explicite) pour $\mathcal{G}$ (respectivement $\mathcal{F}$). Les résultats montrent que l'algorithme divise le temps par deux en multipliant le nombre de processeurs par deux pour tous les ratios sauf le ratio 10. Pour 128 processus, le meilleur temps cpu est obtenu pour un ratio de 1000 mais il est le double de celui du complément de Schur. On remarque que le nombre d'itérations croit avec la décroissance du nombre de tranches. Ce résultat paraît logique puisque les corrections sont plus petites quand le

ratio entre les grilles est faible.

Calculateur C temps cpu (s) et (itérations)								
Ratio / #Processus	1	2	4	8	16	32	64	128
100000	132	66	34	17.9	10.9	5.22	3.1	1.55
	(53)	(53)	(53)	(53)	(53)	(52)	(52)	(52)
10000	46.65	23.8	12.15	7.37	5.50	2.8	1.4	0.7
	(18)	(18)	(18)	(19)	(19)	(19)	(19)	(19)
1000	25.2	13	7.96	5.20	4.59	2.28	1.18	**0.6**
	(9)	(9)	(10)	(10)	(10)	(10)	(10)	(10)
100	20.6	11.9	8.7	6.9	7.08	3.8	2.2	1.3
	(6)	(6)	(7)	(7)	(7)	(7)	(7)	(7)
10	33.25	31.4	26.9	25.8	27.5	23.2	21.5	-
	(4)	(5)	(5)	(5)	(5)	(5)	(5)	-

TABLE 2.5 – Performance de la méthode parareal avec comme opérateurs Euler implicite-explicite

Le tableau 2.6 résume les résultats pour parareal avec le schéma explicite RK2 pour $\mathcal{F}$ et $\mathcal{G}$, les tests sont faits avec $2,10^7$ pas de temps sur la grille fine. Il révèle, qu'avec un meilleur ordre de discrétisation la méthode parareal nécessite moins d'itération pour les mêmes ratios, ce qui implique de meilleures performances parallèles. Cependant, la méthode reste sensible au paramètre ratio, celle-ci donne de meilleures performances pour un ratio de 1000 dans notre cas.

Calculateur C temps cpu (s) et (itérations)					
Ratio/#Processus	1	4	16	64	128
10000	28.65	9.25	5.70	1.16	**0.55**
	6	7	7	7	7
1000	16	5.43	3.61	0.97	**0.5**
	3	4	4	5	5
100	16.26	4.58	4.20	1.20	0.73
	3	4	4	5	5
10	15.10	8.25	9	4	3.13
	2	3	4	4	4

TABLE 2.6 – Performance de la méthode parareal avec les intégrateurs RK2 explicites

Le meilleur temps cpu $0.55s$ est atteint pour un ratio de 1000 pour 128

processus avec le schéma RK2 explicite pour les opérateurs. Comparativement, la décomposition de domaine du complément de Schur donne comme meilleur temps cpu $0.3s$ pour le même nombre de processus voir table 2.3 . L'écart de performance entre parareal et le complément de Schur peut être expliqué par le fait que le schéma de Stormer-Verlet explicite coûte moins en calcul par rapport au RK2. D'ailleurs, pour obtenir les meilleures performances avec la méthode parareal et être compétitif vis-à-vis du complément de Schur pour ce système linéaire d'EDO, il faut avoir une estimation du ratio optimal entre les grilles fines et grossières. La situation peut être différente pour des problèmes non-linéaires où le complément de Schur doit utiliser un algorithme de Newton pour satisfaire les conditions de raccords.

2.6 Conclusion

Une adaptation de la méthode d'Aitken-Schwarz pour l'intégration en temps a été présentée, par la symétrisation de l'intervalle de temps et en transformant l'IVP en un BVP. La convergence ou divergence linéaire de la méthode de Schwarz est démontrée. Cela permet l'accélération de la convergence du Schwarz par la technique d'Aitken. Une méthode pour l'application du procédé d'Aitken plus efficace a été développée. Cependant, pour être compétitif avec les schémas d'intégration classique, une méthode basée sur le complément de Schur pour satisfaire les raccords interfaces a été investiguée. Les résultats parallèles montrent que de telles méthodes deviennent intéressantes quand un grand nombre de processeurs est disponible. En effet, les deux méthodes développées montrent une très bonne scalabilité et un "speed-up" linéaire. La meilleure méthode reste le complément de Schur, pour un problème linéaire elle rivalise avec la méthode pararéal.
Les méthodes développées peuvent devenir intéressantes dans le cadre d'une parallélisation à deux niveaux où les macros domaines distribués sur des noeuds de calculs distants (ou grille) seraient gérés par une méthode présentée ci-dessus, et la résolution des sous-domaines d'un macro-domaine se ferait par une technique de type multiple shooting ou pararéal.

Deuxième partie

Accélération de la convergence des méthodes de décomposition de domaines pour des problèmes non-linéaires

Table des matières

Chapitre 3

Technique d'extrapolation et accélération de la convergence

Table des matières

3.1 Introduction

Dans le cadre de notre problématique générale, nous avons développé des méthodes parallèles pour la résolution d'EDO dans le chapitre 2. Nous avons introduit une méthode de décomposition de type Schwarz en temps qui a pu être validée sur plusieurs systèmes d'EDOs, linéaires et non linéaires. Cependant, il est apparu que la convergence de cette méthode peut se révéler très lente. C'est le cas de beaucoup de méthodes en analyse numérique, par exemple les méthodes itératives pour résoudre les systèmes d'équations, les méthodes de discrétisation, etc. Très souvent la convergence de ces suites est

lente, ce qui rend leur utilisation limitée. Nous avons également adapté les précédents travaux concernant l'accélération de la convergence de la décomposition de domaine de type Schwarz. Le procédé Δ^2 d'Aitken détaillé dans [43], apparaît comme une bonne méthode d'accélération adaptée à beaucoup de cas de décomposition de domaine. Cependant celle-ci n'est exacte que pour les suites linéaires scalaires ou vectorielles, convergentes ou divergentes. Dans le cas où la décomposition de domaine de type Schwarz pour des problèmes non linéaires produit des suites non-linéaires, le procédé d'Aitken peut être itéré, mais alors sans garantie de convergence. L'extension naturelle du procédé d'Aitken aux suites non linéaires nous conduit à nous intéresser à la transformation de Shanks et à ces algorithmes dérivés. Il existe cependant d'autres techniques comme les séries temporelles en statistique, qui permettent de prévoir le comportement futur de données stochastiques présentant du bruit et/ou de la saisonnalité. Cependant un lien existe entre les séries temporelles et l'ϵ-algorithme, notamment dans [7, 8, 9] où les paramètres du processus bien connu ARMA sont calculés par l'ϵ-algorithme. Malgré la complétude de ces techniques statistiques, nous ne les avons pas investiguées car nos données issues d'un processus déterministe ne présentent pas à priori de bruit. De plus nous sommes intéressés non par la prédiction des valeurs de la suite mais par la limite de celle-ci.

Dans ce chapitre, nous nous intéresserons aux techniques d'accélération de la convergence de suites non linéaires [85, 22, 25, 23]. Nous verrons d'abord leurs propriétés scalaires puis vectorielles, enfin nous les appliquerons à notre méthode de décomposition de Schwarz en temps.

Soit une suite (S_n) qui converge vers S. Les méthodes d'accélération de la convergence consistent en la transformation d'une suite (S_n) convergent lentement en une nouvelle suite (T_n) convergente vers la même limite S plus rapidement. Parmi les méthodes de transformation de suites, les plus connues sont celles de Richardson et d'Aitken Δ^2. Ces méthodes sont dites méthodes d'extrapolation, l'idée est d'extrapoler les termes $S_n, S_{n+1}, \ldots, S_{n+k}$.

3.1.1 Méthode d'extrapolation

Définition 3.1. Soient (u_n) et (v_n) deux suites de nombres réels qui tendent vers zéro lorsque n tend vers l'infini :

- Si $\exists N > 0$ et $C > 0$ tel que $\forall n > N$ on a $|v_n| < C|u_n|$ alors on écrit $v_n = O(u_n)$
- Si $\forall \epsilon > 0$,$\exists N$ tel que $\forall n > N$ on a $|v_n| < \epsilon|u_n|$ alors on écrit $v_n = o(u_n)$

Définition 3.2. On dit que (V_n) converge plus vite que (S_n) si :

$$\lim_{n\to\infty} \frac{V_n - S}{S_n - S} = 0$$

Définition 3.3. Soit T une méthode qui permet de transformer la suite (U_n) en une suite (V_n) qui converge également vers zéro. Si $V_n = o(U_n)$ on dit que l'on a accéléré la convergence et que la méthode T est une méthode d'accélération de la convergence.

Soit $\mathcal{C}$ l'espace des suites convergentes de nombres réels muni de la norme $||S|| = \sup_n |S_n|$, alors $\mathcal{C}$ est un espace de banach. On appellera T un procédé de sommation , une application linéaire de l'espace $\mathcal{C}$ dans l'espace des suites, qui transforme S_n en une autre suite (V_n) tel que $\lim_{n\to\infty} V_n = \lim_{n\to\infty} S_n$ quelque soit $(S_n) \in \mathcal{C}$.

Il existe de nombreux procédés de sommation, on citera trois procédés de sommation importants : le procédé de Hölder, le procédé de Césaro et les méthodes d'Euler [25, 23]. Une méthode intéressante qui peut être vue comme un procédé de sommation est la méthode d'extrapolation de Richardson.

3.1.2 Procédé d'Aitken

Le procédé d'Aitken permet l'extrapolation des suites linéaires de manière exacte. On donnera des théorèmes qui permettront par la suite de faire le lien avec les algorithmes pour les suites non-linéaires. Soit une suite (S_n) à convergence lente, et la notation $\Delta S_n = S_{n+1} - S_n$. Le procédé d'Aitken consiste à transformer S_n en une autre suite Q_n définit par

$$Q_n = \frac{S_{n+2}S_n - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n} = S_{n+1} - \frac{\Delta S_n \Delta S_{n+1}}{\Delta^2 S_n} = S_{n+1} - \frac{\Delta S_{n+1}}{\dfrac{\Delta S_{n+1}}{\Delta S_n} - 1} \quad (3.1)$$

d'où le théorème suivant

Théorème 3.1. *Si on applique le procédé d'Aitken à une suite (S_n) qui converge vers S et si*

$$\lim_{n\to\infty} \frac{S_{n+1} - S}{S_n - S} = \lim_{n\to\infty} \frac{\Delta S_{n+1}}{\Delta S_n} = a \neq 1$$

alors la suite (Q_n) converge vers S plus rapidement que la suite (S_n).

Démonstration. En utilisant la définition 3.2, on a que

$$\lim_{n\to\infty} \frac{S_{n+1} - \dfrac{\Delta S_{n+1}}{\dfrac{\Delta S_{n+1}}{\Delta S_n} - 1} - S}{S_{n+1} - S} = 0 \tag{3.2}$$

$$\lim_{n\to\infty} \frac{S_{n+2} - S_{n+1}}{S_{n+1} - S} \times \frac{1}{\dfrac{\Delta S_{n+1}}{\Delta S_n} - 1} = 1 \tag{3.3}$$

$$\lim_{n\to\infty} \frac{\dfrac{S_{n+2} - S}{S_{n+1} - S} - 1}{\dfrac{\Delta S_{n+1}}{\Delta S_n} - 1} = 1 \tag{3.4}$$

Si la condition du théorème est satisfaite, la suite (Q_n) converge vers S et cela plus vite que S_{n+1}. □

Théorème 3.2. *Une condition nécessaire et suffisante pour que $Q_n = S$ pour tout $n > N$ est que la suite (S_n) vérifie :*

$$a_0(S_n - S) + a_1(S_{n+1} - S) = 0$$

avec $a_0 + a_1 \neq 1$ pour tout $n > N$

ou bien le théorème équivalent

Théorème 3.3. *Une condition nécessaire et suffisante pour que $Q_n = S$ pour tout $n > N$ est que $S_n = S + \alpha q^n$ avec $q \neq 1$ pour tout $n > N$*

Remarque 3.1. D'après le théorème précédent, on remarque que l'accélération de la suite (S_n) par le procédé d'Aitken est valable, que la suite soit convergente ou divergente.

3.1.3 Méthode d'extrapolation de Richardson

Soit (S_n) la suite initiale convergeant vers S, et soit x_n une suite auxiliaire non nulle qui converge vers zéro telle qu' il n'existe pas de couple (k, p) tels que $x_k = x_p$. Le procédé de Richardson consiste à construire les quantités $T_k^{(n)}$ à partir du schéma de Neville-Aitken,

$$\begin{cases} T_0^{(n)} = S_n \\ T_{k+1}^{(n)} = \dfrac{x_n T_k^{(n+1)} - x_{n+k+1} T_k^{(n)}}{x_n - x_{n+k+1}} \end{cases} \tag{3.5}$$

$T_k^{(n)}$ représente la valeur en $x = 0$ du polynôme d'interpolation de degré k qui passe par les $k+1$ couples (x_n, S_n),(x_{n+1}, S_{n+1}),...,(x_{n+k}, S_{n+k}). Comme la suite (x_n) est indépendante des $T_k^{(n)}$, il est évident que $T_{k+1}^{(n)}$ est une combinaison linéaire de $T_k^{(n+1)}$ et $T_k^{(n)}$. On en déduit alors que le procédé de Richardson est une transformation linéaire de suite.

Le choix de la suite (x_n) est un paramètre important de ce procédé. On définit l'opérateur Δ tel que :

$$\begin{cases} \Delta^0 S_n = S_n \\ \Delta^{k+1} S_n = \Delta^k S_{n+1} - \Delta^k S_n \end{cases} \tag{3.6}$$

Dans le cas $x_n = \Delta S_n$, on obtient alors un algorithme qui n'est plus linéaire :

$$\begin{cases} T_0^{(n)} = S_n \\ T_{k+1}^{(n)} = \dfrac{\Delta S_n T_k^{(n+1)} - \Delta S_{n+k+1} T_k^{(n)}}{\Delta S_n - \Delta S_{n+k+1}} \end{cases} \tag{3.7}$$

Pour cette méthode, on a le théorème suivant :

Théorème 3.4. *Pour l'algorithme précédent une condition nécessaire et suffisante pour que $T_k^{(n)} = S \ \forall n > N$ est que la suite (S_n) vérifie :*

$$a_0(S_n - S) + a_1 \Delta S_n + \ldots + a_k (\Delta S_n)^k = 0$$

tel que $a_0, a_1, \ldots, a_k$ non nuls.

Démonstration. Puisque $T_k^{(n)}$ est le polynôme de degré k qui en $x = 0$ vaut S, on peut écrire le polynôme :

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \tag{3.8}$$

On peut écrire un système que doit vérifier p en passant par k point :

$$\begin{pmatrix} x_n^k & \ldots & x_n & 1 \\ x_{n+1}^k & \ldots & x_{n+1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n+k}^k & \ldots & x_{n+k} & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} S_n \\ S_{n+1} \\ \vdots \\ S_{n+k} \end{pmatrix} \tag{3.9}$$

On cherche $T_k(n) = a_0$ donc l'on peut écrire $T_k^{(n)}$ sous forme de déterminant :

$$T_k^{(n)} = \frac{\begin{vmatrix} x_n^k & \dots & x_n & S_n \\ x_{n+1}^k & \dots & x_{n+1} & S_{n+1} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n+k}^k & \dots & x_{n+k} & S_{n+k} \end{vmatrix}}{\begin{vmatrix} x_n^k & \dots & x_n & 1 \\ x_{n+1}^k & \dots & x_{n+1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n+k}^k & \dots & x_{n+k} & 1 \end{vmatrix}} = S \tag{3.10}$$

En considérant que $x_n = \Delta S_n$ on obtient la condition nécessaire

$$\begin{vmatrix} S_n & S_{n+1} & \dots & S_{n+k} \\ \Delta S_n & \Delta S_{n+1} & \dots & \Delta S_{n+k} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta S_n^k & \Delta S_{n+1}^k & \dots & \Delta S_{n+k}^k \end{vmatrix} = \begin{vmatrix} S & S & \dots & S \\ \Delta S_n & \Delta S_{n+1} & \dots & \Delta S_{n+k} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta S_n^k & \Delta S_{n+1}^k & \dots & \Delta S_{n+k}^k \end{vmatrix} \tag{3.11}$$

Ou encore :

$$\begin{vmatrix} S_n - S & \dots & S_{n+k} - S \\ \Delta S_n & \dots & \Delta S_{n+k} \\ \vdots & \vdots & \vdots \\ (\Delta S_n)^k & \dots & (\Delta S_{n+k})^k \end{vmatrix} = 0 \tag{3.12}$$

qui n'est satisfaite que si il existe $a_0, a_1, \dots, a_k$ non nuls tels que $a_0(S_n - S) + a_1 \Delta S_n + \dots + a_k (\Delta S_n)^k = 0$. □

La démonstration précédente amène directement un lien avec la généralisation de la transformation de Shanks.

3.1.4 Transformation de Shanks

Dans [76], Shanks introduit la transformation suivante. Soit une suite (s_n) avec la série

$$S = \sum_{m=0}^{\infty} s_m \tag{3.13}$$

La somme partielle S_n est définie par

$$S_n = \sum_{m=0}^{n} s_m \tag{3.14}$$

La suite (S_n) converge vers S quand $n \to \infty$. La transformation de Shanks $Shk(S_n)$ de la suite S_n est définie par

$$Sh_k(S_n) = \frac{S_{n+1}S_{n-1} - S_n^2}{S_{n+1} - 2S_n + S_{n-1}} \tag{3.15}$$

qui devient alors une nouvelle suite qui en général converge plus rapidement que la suite S_n.

Remarque 3.2. La forme de la transformation de Shanks ressemble fortement au procédé Δ^2 d'Aitken. Cependant le procédé d'Aitken transforme la suite (s_n) alors que la transformation de Shanks opère sur les sommes partielles S_n.

Le procédé d'Aitken peut s'écrire sous forme de déterminant :

$$Q_n = \frac{\begin{vmatrix} S_n & S_{n+1} \\ \Delta S_n & \Delta S_{n+1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ \Delta S_n & \Delta S_{n+1} \end{vmatrix}} \tag{3.16}$$

La transformation de Shanks est basée sur le fait que les sommes partielles peuvent souvent être approximées par

$$S_n = S + \alpha q^n \tag{3.17}$$

avec $|q| < 1$ pour que la suite converge vers S quand $n \to \infty$. On peut écrire un système à 3 équations 3 inconnues (S,α,q) :

$$\begin{cases} S_{n-1} = S + \alpha q^{n-1} \\ S_n = S + \alpha q^n \\ S_{n+1} = S + \alpha q^{n+1} \end{cases} \tag{3.18}$$

En résolvant pour S on obtient la formule $Sh_k(S_n)$ (3.15). La généralisation de la transformation de Shanks intervient pour des suites de la forme

$$S_n = S + \sum_{p=1}^{k} \alpha_p q_p^n \tag{3.19}$$

ou sous la forme

$$\sum_{p=0}^{k} \alpha_p (S_{n+p} - S) = 0 \ \forall n, \quad avec \sum_{p=0}^{k} \alpha_p \neq 0 \tag{3.20}$$

En écrivant les relations sous forme d'un système, et on peut déduire par la technique des déterminants S

$$S = \frac{\begin{vmatrix} S_n & \dots & S_{n+k} \\ \Delta S_n & \dots & \Delta S_{n+k} \\ \vdots & \dots & \vdots \\ \Delta S_{n+k-1} & \dots & \Delta S_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ \Delta S_n & \dots & \Delta S_{n+k} \\ \vdots & \dots & \vdots \\ \Delta S_{n+k-1} & \dots & \Delta S_{n+2k-1} \end{vmatrix}} \tag{3.21}$$

Cette formule a été découverte par Jacobi(1834) puis reprise par Schmidt[75] On peut aussi écrire la transformation de Shanks $Sh_k(S_n)$ sous forme de déterminant de Hankel :

$$Sh_k(S_n) = S = \frac{H_{k+1}(S_n)}{H_k(\Delta^2 S_n)} \tag{3.22}$$

avec $H_k(u_n)$ le déterminant de Hankel définit par

$$H_k(u_n) = \begin{vmatrix} u_n & \dots & u_{n+k-1} \\ u_n & \dots & u_{n+k} \\ \vdots & \dots & \vdots \\ u_{n+k-1} & \dots & u_{n+2k-2} \end{vmatrix} \tag{3.23}$$

Théorème 3.5. *Une condition nécessaire et suffisante pour que* $Sh_k^{(n)} = S$ $\forall n > N$ *est que la suite* (S_n) *vérifie :*

$$\sum_{p=0}^{k} \alpha_p (S_{n+p} - S) = 0 \ \forall n, \quad avec \sum_{p=0}^{k} \alpha_p \neq 0$$

Parmi les trois méthodes que nous venons de présenter, nous choisirons l'une ou l'autre suivant les propriétés de la suite à accélérer.

Dans le cas d'une suite linéaire, notre choix se portera sur le procédé d'accélération d'Aitken, car celui-ci accélère toute suite linéaire exactement vers sa limite. Pour les suites non-linéaires, le procédé d'Aitken n'est plus exact.

Le procédé d'extrapolation de Richardson étant basé sur une interpolation polynomiale ne peut accélérer exactement la convergence que des suites de la forme $S_n = S + \sum_{i=1}^{k} a_i x_n^i$ [23] et permet de faire le lien avec la transformation de Shanks. Cette dernière est équivalente au procédé d'Aitken si la suite est linéaire ; par contre, elle est conçue pour accélérer des suites non linéaires.

Le calcul de S par la transformation de Shank nécessite le calcul de deux déterminants, ce qui peut devenir très coûteux. Ce calcul peut être effectué par le E-algorithme [24] ; cependant, la méthode la plus utilisée est l'ϵ-algorithme.

3.2 L'ϵ-algorithme scalaire

L'algorithme trouvé par Wynn[86] permet de calculer les déterminants de la transformation de Shanks,en calculant les $\epsilon_k^{(n)}$ définis par :

$$\begin{cases} \epsilon_{-1}^{(n)} = 0, \ \epsilon_0^{(n)} = S_n \\ \epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + \dfrac{1}{\epsilon_k^{(n+1)} - \epsilon_k^{(n)}} \end{cases} \tag{3.24}$$

En plaçant les quantités ϵ dans un tableau, on obtient la figure 3.1. La relation (3.24) lie les quantités situées aux sommets d'un losange. Les flèches indiquent les trois quantités à partir desquelles est calculée la quantité cible.

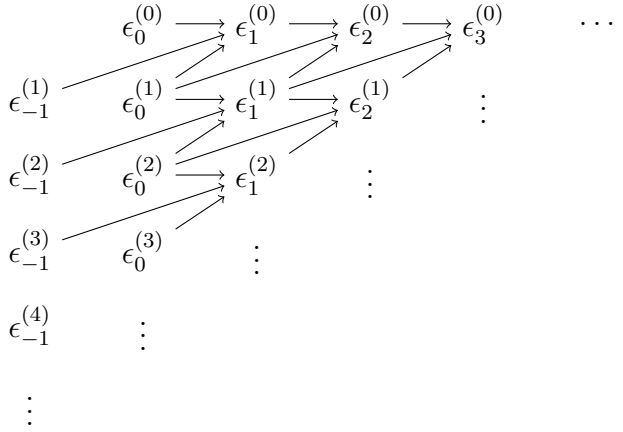


FIGURE 3.1 – Tableau de la construction des quantités ϵ l'ϵ-algorithme scalaire

L'ϵ-algorithme est relié à la transformation de Shanks par les relations

$$\epsilon_{2k}^{(n)} = Sh_k(S_n) \quad \epsilon_{2k+1}^{(n)} = \frac{1}{Sh_k(\Delta S_n)}$$

On les démontre par récurrence, en écrivant les relations de l'ϵ algorithme sous forme de déterminants, et en utilisant les propriétés sur les déterminants. Les nombres $\epsilon_{2k+1}^{(n)}$ sont des calculs intermédiaires, et l'écriture sous forme déterminants de Hankel découle directement des relations précédentes :

$$\epsilon_{2k+1}^{(n)} = \frac{H_k(\Delta^3 S_n)}{H_{k+1}(\Delta S_n)} = \frac{1}{Sh_k(\Delta S_n)} \tag{3.25}$$

et pour $\epsilon_{2k+2}^{(n)}$

$$\epsilon_{2k+2}^{(n)} = \epsilon_{2k}^{(n)} - \frac{[H_{k+1}(\Delta S_n)]^2}{H_k(\Delta^2 S_n) H_{k+1}(\Delta^2 S_n)} \tag{3.26}$$

En effectuant des éliminations dans la relation de l'ϵ-algorithme, Wynn[88] a obtenu la propriété suivante :

Propriété 3.1.

$$[\epsilon_{k+2}^{(n-1)} - \epsilon_k^{(n)}]^{-1} - [\epsilon_k^{(n)} - \epsilon_{k-2}^{(n+1)}]^{-1} = [\epsilon_k^{(n+1)} - \epsilon_k^{(n)}]^{-1} - [\epsilon_k^{(n)} - \epsilon_k^{(n-1)}]^{-1}$$

Cette propriété permet de passer de colonne paire à colonne paire dans le tableau. Elle peut se représenter sous forme d'un losange

$$\begin{array}{ccc} & N = \epsilon_k^{(n-1)} & \longrightarrow & E = \epsilon_{k+2}^{(n-1)} \\ \nearrow & & \nearrow & \\ & C = \epsilon_k^{(n)} & & \\ W = \epsilon_{k-2}^{(n+1)} & \longrightarrow & S = \epsilon_k^{(n+1)} & \end{array} \tag{3.27}$$

et donc la formule devient :

$$(C - N)^{-1} + (C - S)^{-1} = (C - W)^{-1} + (C - E)^{-1}$$

Nous proposons quelques définitions et théorèmes de convergence pour l'ϵ-algorithme dont les démonstrations peuvent être retrouvées dans [22]. Notre champ d'application produit, la plupart du temps, des suites monotones ou oscillantes. On ne s'intéressera qu'aux théorèmes se rapportant à cette classe de suite.

Définition 3.4. On dit que la suite (S_n) est totalement monotone si $(-1)^k \Delta^k S_n >= 0$ pour $n, k = 0, 1 \ldots$. On écrira $(S_n) \in TM$

Définition 3.5. On dit que la suite (S_n) est totalement oscillante si la suite $((-1)^n S_n)$ est totalement monotone. On écrira $(S_n) \in TO$

Théorème 3.6. *Soit $f(t) = \sum_{k=0}^{\infty} c_k t^k$ une série de puissances de rayon de convergence R et telle que $c_k >= 0$ pour tout k et soit $(S_n) \in TM$. Si $S_0 < R$ alors $(f(S_n) \in TM$.*

Lemme 3.1. *Toute suite totalement monotone est convergente*

Lemme 3.2. *Si on applique l'ϵ-algorithme à $(S_n) \in TM$ alors : $\epsilon_{2k}^{(n)} >= 0$ et $\epsilon_{2k+1}^{(n)} <= 0$ pour $n, k = 0, 1, \ldots$*

Lemme 3.3. *Si on applique l'ϵ-algorithme à $(S_n) \in TM$ alors : $0 <= \epsilon_{2k+2}^{(n)} <= \epsilon_{2k}^{(n)}$ pour $n, k = 0, 1, \ldots$*

3.2.1 Divers algorithmes d'accélération

Le procédé d'Overholt

Le procédé d'Overholt consiste à fournir des approximations d'ordre de plus en plus élevé de la limite S des suites (S_n) telles que :

$$S_{n+1} - S = \sum_k \alpha_k (S_n - S)^k \quad \forall n$$

où un nombre fini ou infini de coefficients α_k peut être nuls. Sans rentrer dans les détails techniques de construction, on obtient le procédé d'overholt :

$$\begin{cases} V_0^{(n)} = S_n \quad n = 0, 1, \ldots \\ V_{k+1}^{(n)} = \dfrac{(\Delta S_{n+k})^{k+1} V_k^{(n+1)} - (\Delta S_{n+k+1})^{k+1} V_k^{(n)}}{(\Delta S_{n+k})^{k+1} - (\Delta S_{n+k+1})^{k+1}} \end{cases} \tag{3.28}$$

ρ-algorithme

Le ρ-algorithme consiste à faire passer une fraction rationnelle d'interpolation dont le numérateur et le dénominateur sont des polynômes de degré k par les $2k+1$ couples de points $(x_n, S_n), \ldots, (x_{n+2k}, S_{n+2k})$, à l'aide de la formule d'interpolation de Thiele (voir [69]) puis à calculer la valeur de cette fraction rationnelle en $x = \infty$. L'interpolation de Thiele, par rapport à une interpolation polynomiale présente l'avantage d'être bornée à l'infini et de garder le comportement asymptotique.

Définition 3.6. On appelle différences réciproques les quantités

$$\rho_0^{(n)} = S_n$$

$$\rho_1^{(n)} = \frac{x_n - x_{n+1}}{\rho_0^{(n)} - \rho_0^{(n+1)}}$$

$$\rho_2^{(n)} = \frac{x_n - x_{n+2}}{\rho_1^{(n)} - \rho_1^{(n+1)}} + \rho_0^{(n+1)}$$

$$\rho_k^{(n)} = \frac{x_n - x_{n+k}}{\rho_{k-1}^{(n)} - \rho_{k-1}^{(n+1)}} + \rho_{k-2}^{(n+1)}$$

Le ρ-algorithme s'écrit

$$\rho_{-1}^{(n)} = 0 \quad \rho_0^{(n)} = S_n \quad n = 0, 1, \ldots$$

$$\rho_{k+1}^{(n)} = \rho_{k-1}^{(n+1)} + \frac{x_{k+n+1} - x_n}{\rho_k^{(n+1)} - \rho_k^{(n)}} \quad n, k = 0, 1, \ldots$$

La théorie montre que la suite (x_n) doit tendre vers l'infini quand n tend vers l'infini.

Le ρ-algorithme est peu utilisé en pratique, il est introduit ici pour être complet dans les comparaisons que nous réaliserons sur nos exemples issus de la décomposition de domaine.

Généralisation de l'ϵ-algorithme

Il existe deux généralisations de l'ϵ-algorithme. Ces généralisations consistent à l'introduction d'une suite paramètre x_n. La première généralisation est :

$$\begin{cases} \epsilon_{-1}^{(n)} = 0 \ , \ \epsilon_0^{(n)} = S_n \ , \ n = 0, 1, \ldots \\ \epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + \dfrac{\Delta x_n}{\epsilon_k^{(n+1)} - \epsilon_k^{(n)}} \quad n, k = 0, 1, \ldots \end{cases} \tag{3.29}$$

La deuxième généralisation est donnée par :

$$\begin{cases} \epsilon_{-1}^{(n)} = 0 \ , \ \epsilon_0^{(n)} = S_n \ , \ n = 0, 1, \ldots \\ \epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + \dfrac{\Delta x_{n+k}}{\epsilon_k^{(n+1)} - \epsilon_k^{(n)}} \quad n, k = 0, 1, \ldots \end{cases} \tag{3.30}$$

Dans [22] et [21] C.Brezinski montre que ces deux généralisations de l'ϵ-algorithme se comportent mieux que l'algorithme original pour certains exemples. Cependant, le choix de la suite x_n de paramètre reste à déterminer.

Théorème 3.7. *Pour les première et seconde généralisations de l'ϵ-algorithme, si* $\lim_{n\to\infty} \frac{\Delta x_{n+1}}{\Delta x_n} \neq \lim_{n\to\infty} \frac{\Delta S_{n+1}}{\Delta S_n}$ *alors,*
$\lim_{n\to\infty} \epsilon_2^{(n)} = \lim_{n\to\infty} S_n$.

Ce théorème montre qu'avec les généralisations de l'ϵ-algorithme, il est possible de traiter les suites à convergence logarithmique c'est-à-dire telle que :

$$\lim_{n\to\infty} \frac{\Delta S_{n+1}}{\Delta S_n} = \lim_{n\to\infty} \frac{S_{n+1} - S}{S_n} - S = 1$$

Ce sont des suites difficiles à accélérer , le procédé Δ^2 d'Aitken peut ne pas converger [19].

θ-algorithme

Le θ-algorithme fait le lien entre l'ϵ-algorithme et le ρ-algorithme. En partant des relations de l'ϵ algorithme réécrites sous la forme :

$$\begin{cases} \epsilon_{-1}^{(n)} = 0, \ \epsilon_{-1}^{(n)} = S_n \\ \epsilon_{k+1}^{(n)} = \epsilon_{k+1}^{(n)} + D_k^{(n)} \end{cases} \quad avec \quad D_k^{(n)} = \frac{1}{\epsilon_k^{(n+1)} - \epsilon_k^{(n)}} \tag{3.31}$$

Puis en utilisant le théorème suivant :

Théorème 3.8. *Supppons que* $\lim_{n\to\infty} \epsilon_{2k+2}^{(n)} = \lim_{n\to\infty} \epsilon_{2k}^{(n)}$. *Une condition nécessaire et suffisante pour que*

$$\lim_{n\to\infty} \frac{\Delta \epsilon_{2k+2}^{(n)}}{\Delta \epsilon_{2k}^{(n+1)}} = 0$$

est que :

$$\lim_{n\to\infty} \frac{\Delta D_{2k+1}^{(n)}}{\Delta \epsilon_{2k}^{(n+1)}} = -1$$

Il n'y a accélération de la convergence en passant de la colonne $2k$ à la colonne $2k+2$ que si la condition du théorème est respectée. Dans le cas contraire, on introduit un paramètre w_k dans les relations paires $2k+2$ de l'algorithme, la relation $\epsilon_{2k+1}^{(n)}$ restant la même. Pour satisfaire la condition du théorème, $w_k = -\lim_{n\to\infty} \frac{\Delta \epsilon_{2k}^{(n+1)}}{\Delta \epsilon_{2k+2}^{(n)}}$. En remplaçant w_k par $w_k^{(n)} = -\frac{\Delta \epsilon_{2k}^{(n+1)}}{\Delta \epsilon_{2k+2}^{(n)}}$

que l'on obtient les relations du θ-algorithme :

$$\begin{cases} \theta_{-1}^{(n)} = 0, \quad \theta_0^{(n)} = S_n \quad n = 0, 1, \dots \\ \theta_{2k+1}^{(n)} = \theta_{2k-1}^{(n+1)} + D_{2k}^{(n)} \quad n = 0, 1, \dots \\ \theta_{2k+2}^{(n)} = \dfrac{D_{2k+1}^{(n+1)}\theta_{2k}^{(n+1)} - D_{2k+1}^{(n)}\theta_{2k}^{(n+2)}}{D_{2k+1}^{(n+1)} - D_{2k+1}^{(n)}} \end{cases} \quad avec \;\; D_k^{(n)} = \frac{1}{\theta_k^{(n+1)} - \theta_k^{(n)}} \tag{3.32}$$

Les exemples montrent que le θ-algorithme se comportent comme le meilleur des deux algorithmes pour une suite donnée, et même parfois mieux [39][78].

3.3 Cas de suites vectorielles

Maintenant que l'on a connaissance des principaux algorithmes d'accélération de la convergence, il convient de se demander comment ceux-ci s'étendent à des suites vectorielles ou matricielles, qui sont beaucoup plus courantes en analyse numérique. On se concentrera sur les différentes variantes de l'ϵ-algorithme et sur la version topologique du θ-algorithme.

3.3.1 l'ϵ-algorithme vectoriel

Contrairement à l'ϵ-algorithme scalaire, qui est issu de la transformation de Shanks et du procédé d'Aitken avec toute la théorie mathématique associée, l'écriture de l'ϵ-algorithme vectoriel n'est pas basée la transformation de Shanks et l'écriture sous forme de déterminant. L'ϵ-algorithme vectoriel a été proposé par Wynn[87] en définissant l'inverse d'un vecteur, quantité qui intervient dans les relations de l'ϵ-algorithme scalaire (3.24).

Soit S un vecteur de $\mathbb{C}^p$ et (S_n) une suite de vecteurs de $\mathbb{C}^p$, alors les règles de cet algorithme sont les suivantes :

$$\begin{cases} \epsilon_{-1}^{(n)} = 0 \;\; , \;\; \epsilon_0^{(n)} = S_n \;\; , \;\; n = 0, 1, \dots \\ \epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+1)} + (\Delta\epsilon_k^{(n)})^{-1} \;\; , \; n = 0, 1, \dots \end{cases} \tag{3.33}$$

où l'inverse y^{-1} d'un vecteur $y \in \mathbb{C}^p$ est défini par :

$$y^{-1} = \frac{\bar{y}}{(y, y)}$$

$\bar{y}$ est le vecteur complexe conjugué de y et (y, y) est le produit scalaire dans $\mathbb{C}^p$.

Le résultat fondamental de la théorie de l'ϵ-algorithme vectoriel a été donné par Wynn et démontré par Mc Leod[67] :

Théorème 3.9. *Soit $S \in \mathbb{C}^p$ et soit (S_n) une suite de vecteurs de $\mathbb{C}^p$ telle que :*

$$\sum_{i=0}^{k} a_i(S_{n+i} - S) = 0 \qquad n = 0, 1, \ldots$$

où les a_i sont des nombres réels avec $a_k \neq 0$. Si on applique l'ϵ-algorithme vectoriel à cette suite alors :

$$\epsilon_{2k}^{(n)} = S \text{ pour } n = 0, 1, \ldots \text{ si } \sum_{i=0}^{k} a_i \neq 0$$

et :

$$\epsilon_{2k}^{(n)} = 0 \text{ pour } n = 0, 1, \ldots \text{ si } \sum_{i=0}^{k} a_i = 0$$

Remarque 3.3. La condition du théorème précédent est une condition suffisante alors que la condition du théorème 3.5 équivalent pour l'ϵ-algorithme est nécessaire et suffisante. $\epsilon_{2k}^{(n)} = S$ n'induit pas que la suite (S_n) vérifie la relation $\sum_{i=0}^{k} a_i(S_{n+i} - S) = 0$.

Un théorème intéressant sur la solution d'un système linéaire par l'ϵ-algorithme :

Théorème 3.10. *Si on applique l'ϵ-algorithme vectoriel à la suite de vecteurs (S_n) produit par $S_{n+1} = AS_n + b$ avec S_0 donné où A est une matrice carrée réelle telle que $I - A$ soit inversible alors :*

$$\epsilon_{2m}^{(n)} = S \quad \text{pour} \quad n = 0, 1, \ldots$$

où $x = (I - A)^{-1}b$ et m le degré du polynôme minimal de A pour $S_0 - S$.

Remarque 3.4. L'ϵ-algorithme normé est la généralisation de l'ϵ vectoriel. Les relations restent les mêmes, seule la définition de l'inverse d'un vecteur change. L'inverse est donc défini par $y^{-1} = \dfrac{\bar{y}}{||y||^2}$ où $||y||$ désigne n'importe quelle norme vectorielle de y.

Remarque 3.5. De manière analogue à la démarche précédente, Wynn a défini l'ϵ-algorithme matriciel. Il s'agit d'un ϵ-algorithme qui s'applique à des matrices carrées. La règle de l'ϵ scalaire fait intervenir une inversion de quantité, ici ce sera simplement l'inverse d'une matrice. L'inconvénient est que l'inversion de matrices peut rapidement devenir coûteuse. Une autre approche consiste à mettre les colonnes de la matrice bout à bout dans un vecteur, et à utiliser les versions vectorielles de l'ϵ-algorithme. Cette technique peut être utilisée lorsque l'on résout un problème 3D dont l'interface des domaines à accélérer est en 2D (matricielle).

Pour pallier le manque de théorie dû au fait que l'ϵ-algorithme vectoriel ne peut s'écrire sous forme de déterminant, Brezinski [17] a développé l'ϵ-algorithme topologique.

3.3.2 L'ϵ-algorithme topologique

Cet algorithme a été construit en partant d'une théorie proche de celle de Shanks pour pouvoir aboutir aux règles de l'algorithme. Soit (S_n) une suite d'éléments d'un espace vectoriel E sur $\mathbb{K}$ et E' son dual topologique. Il faut donc définir l'inverse d'un élément de E :

Définition 3.7. Soit $a \in E$ et $b \in E'$ dual topologique de E, tels que $< b, a > \neq 0$. On appelle inverse du couple $(a, b) \in E \times E'$ le couple $(b^{-1}, a^{-1}) \in E \times E'$ défini par :

$$a^{-1} = \frac{b}{< b, a >} \quad b^{-1} = \frac{a}{< b, a >} \ , \ a^{-1} \in E, b^{-1} \in E$$

On dira également que a^{-1} est l'inverse de a par rapport à b et réciproquement.

Les propriétés de l'inverse sont :

Propriété 3.2.

$$\begin{aligned} < a^{-1}, a >= 1, \\ < b, b^{-1} >= 1, \\ < a^{-1}, b^{-1} >= \frac{1}{< b, a >}, \\ (a^{-1})^{-1} = a. \end{aligned}$$

Pour pouvoir écrire l'ϵ-algorithme topologique, on reprend la démarche qui fait intervenir la transformation de Shanks. Supposons que la suite (S_n) vérifie :

$$\sum_{i=0]}^{k} a_i (S_{n+i} - S) = 0, \ \forall n \ \ avec \ \ \sum_{i=0]}^{k} a_i \neq 0 \quad et \ a_i \in \mathbb{K}$$

sans restreindre la généralité on peut supposer que $\sum_{i=0]}^{k} a_i = 1$, donc on peut écrire le système :

$$\begin{cases} a_0 + a_1 + \ldots + a_k = 1 \\ a_0 S_n + a_1 S_{n+1} + \ldots + a_k S_{n+k} = S \\ a_0 S_{n+1} + a_1 S_{n+2} + \ldots + a_k S_{n+k+1} = S \\ \ldots\ldots\ldots \\ a_0 S_{n+k} + a_1 S_{n+k+1} + \ldots + a_k S_{n+2k} = S \end{cases} \tag{3.34}$$

Ce qui peut s'écrire encore

$$\begin{cases} a_0 + a_1 + \ldots + a_k = 1 \\ a_0\Delta S_n + a_1\Delta S_{n+1} + \ldots + a_k\Delta S_{n+k} = 0 \\ a_0\Delta S_{n+1} + a_1\Delta S_{n+2} + \ldots + a_k\Delta S_{n+k+1} = 0 \\ \ldots\ldots\ldots \\ a_0\Delta S_{n+k-1} + a_1\Delta S_{n+k} + \ldots + a_k\Delta S_{n+2k-1} = 0 \\ a_0 S_{n+k} + a_1 S_{n+k+1} + \ldots + a_k S_{n+2k} = S \end{cases} \tag{3.35}$$

La dernière relation du système d'équation (3.35) peut être remplacée par

$$a_0 S_n + a_1 S_{n+1} + \ldots + a_k S_{n+k} = S \tag{3.36}$$

Soit$y' \in E'$ avec les $k+1$ équations du système , on a :

$$\begin{cases} a_0 + a_1 + \ldots + a_k = 1 \\ a_0 < y', \Delta S_n > + \ldots + a_k < y', \Delta S_{n+k} >= 0 \\ \ldots\ldots\ldots \\ a_0 < y', \Delta S_{n+k-1} > + \ldots + a_k < y', \Delta S_{n+2k-1} >= 0 \end{cases} \tag{3.37}$$

On peut alors résoudre le système linéaire par la transformation de Shanks généralisée :

$$S = \frac{\begin{vmatrix} S_n & \ldots & S_{n+k} \\ < y', \Delta S_n > & \ldots & < y', \Delta S_{n+k} > \\ \vdots & \ldots & \vdots \\ < y', \Delta S_{n+k-1} > & \ldots & < y', \Delta S_{n+2k-1} > \end{vmatrix}}{\begin{vmatrix} 1 & \ldots & 1 \\ < y', \Delta S_n > & \ldots & < y', \Delta S_{n+k} > \\ \vdots & \ldots & \vdots \\ < y', \Delta S_{n+k-1} > & \ldots & < y', \Delta S_{n+2k-1} > \end{vmatrix}} \tag{3.38}$$

On définit le déterminant de Hankel généralisé par :

$$\tilde{H}_{k+1}^{(n)}(S_n) = \begin{vmatrix} S_n & \ldots & S_{n+k} \\ < y', \Delta S_n > & \ldots & < y', \Delta S_{n+k} > \\ \vdots & \ldots & \vdots \\ < y', \Delta S_{n+k} > & \ldots & < y', \Delta S_{n+2k} > \end{vmatrix} k = 0, 1, \ldots \tag{3.39}$$

On note $H_{k+1}^{(n)}(S_n) =< y', \tilde{H}_{k+1}^{(n)}(S_n) >$, alors on a :

$$e_k(S_n) = \frac{\tilde{H}_{k+1}^{(n)}(S_n)}{H_k^{(n)}(\Delta^2 S_n)}$$

Les relations de l'ϵ-algorithme topologique sont alors :

$$\begin{cases} \epsilon_{-1}^{(n)} = 0 \in E' , \qquad \epsilon_0^{(n)} = S_n \in E , \quad n = 0, 1, \ldots \\ \epsilon_{2k+1}^{(n)} = \epsilon_{2k-1}^{(n+1)} + [\Delta\epsilon_{2k}^{(n)}]^{-1} \quad n, k = 0, 1, \ldots \\ \text{avec} \quad [\Delta\epsilon_{2k}^{(n)}]^{-1} = \dfrac{y'}{< y', \Delta\epsilon_{2k}^{(n)} >} \quad et \quad y'^{-1} = \dfrac{\Delta\epsilon_{2k}^{(n)}}{< y', \Delta\epsilon_{2k}^{(n)} >} \quad \text{où } y' \in E' \\ \epsilon_{2k+2}^{(n)} = \epsilon_{2k}^{(n+1)} + [\Delta\epsilon_{2k+1}^{(n)}]^{-1} \quad n, k = 0, 1, \ldots \\ \text{avec} \quad [\Delta\epsilon_{2k+1}^{(n)}]^{-1} = \dfrac{y'}{< \Delta\epsilon_{2k+1}^{(n)}, y'^{-1} >} = \dfrac{\Delta\epsilon_{2k}^{(n)}}{< \Delta\epsilon_{2k+1}^{(n)}, \Delta\epsilon_{2k}^{(n)} >} \end{cases} \tag{3.40}$$

La deuxième généralisation remplace la dernière ligne par

$$[\Delta\epsilon_{2k+1}^{(n)}]^{-1} = \frac{\Delta\epsilon_{2k}^{(n+1)}}{< \Delta\epsilon_{2k+1}^{(n)}, \Delta\epsilon_{2k}^{(n+1)} >} \tag{3.41}$$

Il existe une forme topologique du θ-algorithme qui est construite à partir de l'ϵ-algorithme topologique, de manière analogue au θ-algorithme scalaire.

3.3.3 θ-algorithme topologique

Les règles du θ-algorithme topologique s'obtiennent de la même façon que celle du θ-algorithme scalaire ; en partant des règles de l'ϵ-algorithme topologique, on introduit le paramètre w_k paramètre pour satisfaire la condition d'accélération de la convergence entre deux colonnes paires. On obtient alors les relations du θ-algorithme topologique suivant :

$$\begin{cases} \theta_{-1}^{(n)} = 0, \quad \theta_0^{(n)} = S_n \quad n = 0, 1, \ldots \\ \theta_{2k+1}^{(n)} = \theta_{2k-1}^{(n+)} + D_{2k}^{(n)} \quad n = 0, 1, \ldots \\ \theta_{2k+2}^{(n)} = \theta_{2k}^{(n+1)} + w_k^{(n)} D_{2k+1}^{(n)} \\ avec \quad D_{2k}^{(n)} = \dfrac{y}{(y, \theta_k^{(n+1)} - \theta_k^{(n)})} \\ avec \quad \dfrac{\theta_{2k}^{(n+1)} - \theta_{2k}^{(n}}{(\theta_{2k+1}^{(n+1)} - \theta_{2k+1}^{(n}, \theta_{2k}^{(n+1)} - \theta_{2k}^{(n)})} \\ etw_k^{(n)} = -\dfrac{(z, \theta_{2k}^{(n+2)} - \theta_{2k}^{(n+1)})}{(z, D_{2k+1}^{(n+1)} - D_{2k+1}^{(n}} \end{cases} \tag{3.42}$$

Chapitre 4

Application aux méthodes de décomposition de domaines en temps

Table des matières

4.1 Comportement numérique

Nous allons maintenant détailler la mise en œuvre de ces algorithmes et les difficultés que l'on peut rencontrer. On parlera de la stabilité des algorithmes et la mise en place de règles particulières pour empêcher la propagation des erreurs.

4.1.1 Stabilité numérique

Wynn a étudié la stabilité de l'ϵ-algorithme pour certains types de suites. Il s'agit de comprendre le comportement de l'ϵ-algorithme quand une erreur

est introduite sur une valeur de la suite. En introduisant $\delta_{k-1}^{(n+1)}$ une erreur relative sur $\epsilon_{k-1}^{(n+1)}$, celle-ci se propage alors aux autres quantités du losange. Wynn a trouvé les relations suivantes ,

$$\begin{cases} \delta_k^{(n)} = -\dfrac{\epsilon_{k-1}^{(n+1)}}{\epsilon_k^{(n)}(\epsilon_{k-1}^{(n+1)} - \epsilon_{k-1}^{(n)})^2}\delta_{k-1}^{(n+1)} \\ \delta_k^{(n+1)} = \dfrac{\epsilon_{k-1}^{(n+1)}}{\epsilon_k^{(n+1)}(\epsilon_{k-1}^{(n+2)} - \epsilon_{k-1}^{(n+1)})^2}\delta_{k-1}^{(n+1)} \end{cases} \tag{4.1}$$

A partir de ces relations, il a trouvé la relation suivante pour la suite particulière de la forme $S_n = S + \sum_{i=1}^{\infty} a_i \lambda_i^n$ avec $1 > \lambda_1 > \ldots > 0$:

$$\begin{cases} \delta_{2k+1}^{(n)} = \dfrac{\lambda_{k+1}}{1-\lambda_{k+1}}\delta_{2k}^{(n+1)} \\ \delta_{2k+2}^{(n+1)} = -\left(\dfrac{\lambda_{k+1}}{1-\lambda_{k+1}}\right)^2 \delta_{2k}^{(n+1)} \end{cases} \tag{4.2}$$

La stabilité dépend alors de la proximité de λ_{k+1} et de 1. Pour les suites oscillantes du type $S_n = S + (-1)^n \sum_{i=1}^{\infty} a_i \lambda_i^n$ avec $1 > \lambda_1 > \ldots > 0$, l'$\epsilon$-algorithme est inconditionnellement stable. Il n'y a pas de règle universelle pour la stabilité de l'algorithme, elle dépendra des propriétés de la suite accélérée. Dans les relations de l'ϵ-algorithme, il y a une division ; celle-ci peut rendre l'algorithme instable si la quantité se rapproche de 0. Pour contourner, ce problème Wynn et Cordelier ont trouvé des règles particulières.

Règle particulière de Wynn[88]

Pour les algorithmes du même type que l'ϵ-algorithme, on peut avoir deux approximants très proches. Considérons la forme scalaire de l'ϵ-algorithme, la règle que nous verrons ici peut être appliquée aux généralisations de l'ϵ-algorithme, au θ-algorithme également.

Supposons que $\epsilon_{k-2}^{(n+1)} = \epsilon_{k-2}^{(n+2)} = b$, si les quantités au numérateur sont non nulles, alors $\epsilon_{k-2}^{(n+1)}$ devient infinie. Donc $\epsilon_k^{(n)}$ et $\epsilon_k^{(n+1)}$ sont égaux à b et $\epsilon_{k+1}^{(n)}$ est indéterminée. Ce cas se schématise par la figure suivante 4.1. Cette situation engendre une grande perte de précision de l'algorithme due à l'annulation qui intervient dès que $\epsilon_{k-2}^{(n+1)}$ et $\epsilon_{k-2}^{(n+2)}$ sont très proche. Pour éviter cette indétermination sur $\epsilon_{k+1}^{(n)}$, on remplace la règle habituelle par la

$$\begin{array}{ccccc} & & \epsilon_{k-1}^{(n)} \equiv N & & \\ & \epsilon_{k-2}^{(n+1)} = b & & \epsilon_{k}^{(n)} = b & \\ \epsilon_{k-3}^{(n+2)} \equiv W & & C \equiv \epsilon_{k-1}^{(n+1)} = \infty & & E \equiv \epsilon_{k+1}^{(n)} =? \\ & \epsilon_{k-2}^{(n+2)} = b & & \epsilon_{k}^{(n+1)} = b & \\ & & \epsilon_{k-1}^{(n)} \equiv S & & \end{array}$$

FIGURE 4.1 – Illustration d'une situation de blocage dans l'ϵ-algorithme

règle particulière de Wynn,qui s'écrit :

$$\begin{cases} E = a(1 + \dfrac{a}{C})^{-1} \\ a = S(1 - \dfrac{S}{C})^{-1} + N(1 - \dfrac{N}{C})^{-1} - W(1 - \dfrac{W}{C})^{-1} \end{cases} \tag{4.3}$$

à partir de la notation points cardinaux de la figure 4.1. Le principe est en fait d'éviter les calculs intermédiaires et de ne calculer $\epsilon_{k+1}^{(n)}$ qu'à partir des ϵ des colonnes de mêmes parités qui ne sont pas affectés par l'annulation. Quand on est dans un cas d'annulation où $\epsilon_{k-2}^{(n+1)} = \epsilon_{k-2}^{(n+2)}$, alors $C = \infty$ et la règle simplifiée est :

$$\epsilon_{k+1}^{(n)} = \epsilon_{k-1}^{(n+2)} + \epsilon_{k-1}^{(n)} - \epsilon_{k-3}^{(n+2)} \tag{4.4}$$

On écrit similairement la règle pour le ρ-algorithme.

Règle particulière de Cordellier[31, 32]

La règle de Wynn n'est valable que pour les algorithmes scalaires. Cordelier a trouvé une règle particulière pour l'ϵ-algorithme vectoriel :

$$E = \frac{\dfrac{N}{||C-N||^2} + \dfrac{S}{||C-S||^2} - \dfrac{W}{||C-W||^2} + aC}{\dfrac{1}{||C-N||^2} + \dfrac{1}{||C-S||^2} - \dfrac{1}{||C-W||^2} + a} \tag{4.5}$$

avec

$$a = \frac{||N-W||^2}{||C-W||^2||C-N||^2} + \frac{||S-W||^2}{||C-W||^2||C-S||^2} - \frac{||N-S||^2}{||C-S||^2||C-N||^2} \tag{4.6}$$

La norme $||x-y||$ est définie par $||x-y||^2 = (x-y, x-y)$. Le principe de cette régle est de remplacer une addition non stable numériquement par une

division qui n'affecterait que la dernière décimale. Il est évident que cette règle peut s'appliquer dans le cas scalaire.

La section suivante présentera les techniques relatives à la mise en œuvre informatique de ces algorithmes.

4.1.2 Programmation des algorithmes

Nous parlerons essentiellement de la technique du losange mobile et des conditions d'arrêt de l'ϵ-algorithme.

Quand on applique l'ϵ-algorithme à $2k$ termes d'une suite scalaire, l'algorithme produit un tableau comme suit :

$$\begin{array}{llllll}
\epsilon_{-1}^{(0)} = 0 & \epsilon_0^{(0)} = S_0 & \epsilon_1^{(0)} & \epsilon_2^{(0)} & \dots & \epsilon_{2k}^{(0)} \\
\epsilon_{-1}^{(1)} = 0 & \epsilon_0^{(1)} = S_1 & \epsilon_1^{(1)} & \vdots & \iddots & \\
\epsilon_{-1}^{(2)} = 0 & \epsilon_0^{(2)} = S_2 & \vdots & \epsilon_2^{(2k-2)} & & \\
\vdots & \vdots & \epsilon_1^{(2k-1)} & & & \\
\epsilon_{-1}^{(2k)} = 0 & \epsilon_0^{(2k)} = S_{2k} & & & &
\end{array}$$

TABLE 4.1 – Tableau de calcul de l'ϵ-algorithme

En termes de stockage cela représente donc $2k^2$ double quand on stocke la matrice pleine. La structure est clairement diagonale supérieure, on pourrait faire un stockage plus adapté que le plein mais cela compliquerait l'algorithme. Il y a donc l'inconvénient de prendre beaucoup de place mémoire ; de plus, quand on veut rajouter des termes à la suite au fur et à mesure, cela demande de prévoir assez grand ou de faire de la réallocation. Cette manière de programmer l'algorithme pourrait convenir pour des suites scalaires mais deviendrait rapidement coûteuse en stockage pour des cas vectoriels surtout quand $2k$ devient grand. Il devient alors nécessaire de pouvoir implémenter l'algorithme de manière à utiliser le moins de stockage possible et ainsi de garder tout l'intérêt de ne pas avoir à calculer les déterminants.

La technique du losange mobile permet de pallier ces défauts. Le principe est de conserver le minimum d'informations permettant d'avancer dans le tableau de l'ϵ-algorithme. On sait que le calcul se fait en losange, donc les termes nécessaires se trouvent sur la diagonale courante et la diagonale précédente. Lorsque l'on avance sur la diagonale courante, il apparaît que les éléments déjà utilisés de la diagonale précédente sont inutiles. La figure 4.2 schématise la technique dite du losange mobile :

Les doubles flèches représentent l'état du vecteur de stockage des valeurs de la diagonale du tableau. On se place dans la situation où le calcul doit être

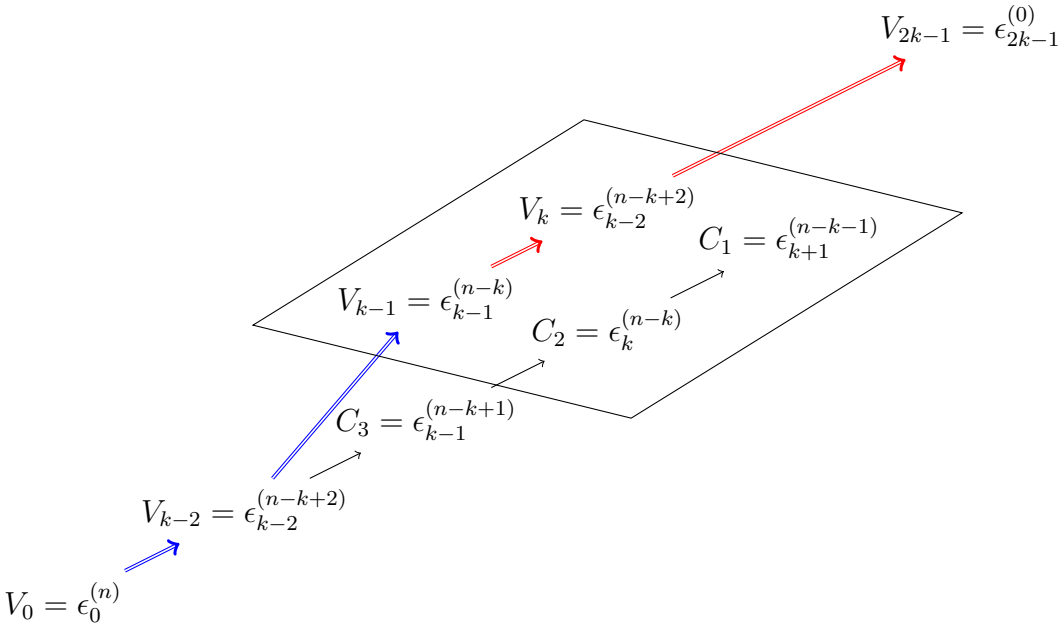


FIGURE 4.2 – Illustration de la technique du losange mobile pour le calcul des quantités ϵ

effectué pour la prochaine diagonale. Au fur et à mesure que le losange se déplace pour calculer les valeurs de la nouvelle diagonale, celles-ci sont remplacées dans le vecteur. C_2, C_3 servent au stockage temporaire des nouvelles valeurs tant que les anciens V_k et V_{k-1} sont requis pour le calcul de C_1. Pour faire avancer le losange, C_3 est stocké dans V_{k-1}, C_2 dans C_3 et C_1 dans C_2. Toutes les quantités sont disponibles pour calculer $\epsilon_{k+2}^{(n-k-2)}$ qui sera stocké dans C_1.

Cette technique permet plus aisément d'appliquer l'ϵ-algorithme parallèlement aux itérations de la suite. Le calcul de l'extrapolation peut être donc dévoué à un processeur ; cependant, il passerait la plupart de son temps à attendre les nouvelles valeurs de la suite(en particulier pour la méthode de Schwarz). Il est naturel de concevoir que c'est la suite devant être accélérée qui pilote la technique d'extrapolation. Cependant, il s'avère beaucoup plus facile à mettre en œuvre de voir les choses comme si c'était la technique d'accélération qui demande de nouveaux itérés à la suite. Ainsi quand le critère d'arrêt est satisfait la génération des termes de la suite s'arrête. Il n'est pas d'un grand intérêt de paralléliser l'extrapolation, elle peut être conservée sur un des processeurs en charge de calculer les termes de la suite.

4.2 Application à la méthode de Schwarz

La méthode de décomposition de domaine de Schwarz, développée au chapitre 2, produit une suite (V_n) de solutions interfaces entre les sous-domaines. Dès que l'interface entre sous-domaines ne se réduit plus à un point, la suite (V_n) n'est plus scalaire mais vectorielle.

Afin d'étudier le comportement de l' ϵ-algorithme avec notre méthode de décomposition de domaine de Schwarz en temps dans le cas d'opérateur non linéaire, considérons le problème à valeur initiale suivant :

$$\begin{cases} \dot{y} = exp(y) \\ y(0) = -2 \end{cases} \tag{4.7}$$

En appliquant l'algorithme de Schwarz en temps présenté à la section 2.3, on obtient un problème aux conditions limites non linéaire pour chaque sous-domaine. Pour résoudre chaque problème local, on applique une méthode de type Newton [35]. Cela implique donc qu'il faut trouver une bonne solution initiale pour s'assurer de la convergence du Newton. Il est important de remarquer que la convergence de la méthode de Schwarz n'est pas requise pour que l'ϵ-algorithme converge vers la limite de la suite. Cependant une mauvaise résolution du problème local sur les sous-domaines, dûe à la non convergence du Newton, entrainera une mauvaise résolution globale de la méthode de Schwarz. La suite produite par l'algorithme de Schwarz n'étant plus la même, l'accélération de la suite est perturbée, et l'ϵ-algorithme ne pourra converger vers la bonne limite.

On va appliquer les méthodes d'accélération de la convergence à la suite de type $V_{n+1} = F(V_n)$ produite par les échanges des solutions aux interfaces dans l'algorithme de Schwarz. V_n représente la condition de Dirichlet du second domaine, et la fonction F est une itération du Schwarz multiplicatif 2.13. V_{n+1} correspond alors à la condition de Dirichlet calculée en $t = T$ dans le premier domaine. La fonction F n'est pas connue analytiquement en général, on a accès seulement au résultat de son application.

Alors deux stratégies s'offrent à nous pour réaliser l'accélération de cette suite (V_n) formée par F.

1. Une première approche consiste à collecter k termes de la suite (V_n) jusqu'à la convergence imposée par le critère d'arrêt :

$$|1 - \frac{\epsilon^{(3)}_{k-3}}{\epsilon^{(1)}_{k-1}}| < \epsilon. \tag{4.8}$$

où ϵ est fixé. On obtient la solution finale de l'algorithme de Schwarz en

effectuant une nouvelle itération en ayant injecté la condition accélérée par l'ϵ-algorithme.

2. Une autre manière d'appliquer l'ϵ-algorithme, due à C. Brezinski [20], consiste à appliquer l'ϵ-algorithme sur $2m-r$ itérés de base d'un système non linéaire à p équations et p inconnues. En prenant $x_0 \in \mathbb{R}^p$ fixé,

$$\begin{cases} V_0 = x_n \in \mathbb{R}^p \\ V_k = F(V_{k-1}), k = 1, \cdots, 2m-r \end{cases} \tag{4.9}$$

Alors la 'prédiction' $x_{n+1} = \epsilon_{2m-r}^{(r)}$ donnée par l'ϵ-algorithme à partir des itérations de base est injectée dans (4.9), puis de nouveau on calcule des itérations de base. Le paramètre m représente le degré du polynôme minimal de la matrice jacobienne notée $F'(x)$ pour le vecteur $x_n - x$, et r est la multiplicité de la racine en zéro de ce polynôme. Il a été prouvé [20] que la convergence de cette méthode est quadratique si la matrice $I - F'(x)$ est inversible et x est un point fixe de F, c'est à dire $||x_{n+1} - x|| = O(||x_n - x||^2)$.

Nous allons maintenant illustrer ces deux techniques sur des problèmes EDO avec notre algorithme de Schwarz en temps. On commencera par un exemple scalaire , puis on montrera le comportement de l'ϵ-algorithme dans des cas vectoriels.

Exemple scalaire

Considérons le problème à valeur initiale (4.7) avec $T = 5$, on le transforme en un problème aux limites, puis on applique l'algorithme de Schwarz en temps pour deux domaines avec 200 pas de temps pour chaque domaine.

La figure 4.3 compare l'accélération de la convergence du Schwarz par l'ϵ-algorithme et par la méthode Aitken-Schwarz [43]. La méthode de Schwarz en temps converge vers la bonne solution à $||V_{n+1} - V_n|| = 1.\,10^{-10}$ en 160 itérations.

On ne peut démarrer le procédé d'Aitken qu'à partir de la 30ème itération quand la convergence devient 'linéaire'. Alors en 4 applications du procédé d'Aitken, on obtient une solution extrapolée qui donne une convergence à $1.\,10^{-10}$. Chaque application d'Aitken nécessite 3 itérations de l'algorithme de Schwarz. Le problème étant scalaire, la construction de la matrice $\mathbb{P}$ s'effectue à partir de l'erreur à l'interface. Il faut donc 3 solutions interface qui donnent deux erreurs et un coefficient pour $\mathbb{P}$. Cette méthode apporte déjà un gain conséquent par rapport au Schwarz classique. Cependant, en général,

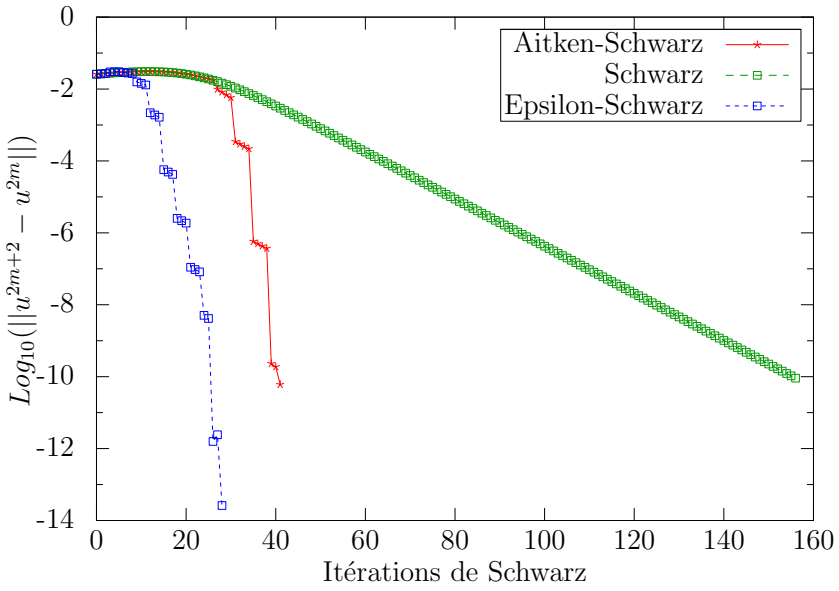


FIGURE 4.3 – Convergence des méthodes de Schwarz Aitken-Schwarz et ϵ-Schwarz pour 2 domaines en temps S_1 et $\bar{S}_1$

la convergence des problèmes non linéaires ne devient pas linéaire à partir d'un certain nombre d'itérations.

L'ϵ-algorithme (4.9) montre un meilleur comportement ; il peut être appliqué avec beaucoup moins d'itérations de Schwarz. La méthode employée ici est la deuxième, dès que les $2m - r$ itérations de base ont été effectuées, on applique l'extrapolation qui est réinjectée dans l'algorithme de Schwarz, dans notre cas le nombre d'itérations de base nécessaire est 3. En termes de coût de calcul, il faut considérer deux aspects, le nombre de résolutions locales nécessaires (intimement lié au nombre d'itérations) et le coût de calcul de l'extrapolation. L'ϵ-algorithme converge en moins de 25 itérations avec un moindre coût de calcul comparé à la méthode Aitken-Schwarz ; il effectue moins de résolution locale et le calcul de l'extrapolation par la formule (3.24) est moins coûteuse que l'inversion de matrice de la méthode Aitken-Schwarz.

La méthode Aitken-Schwarz implique la construction d'une matrice $\mathbb{P}$ d'accélération satisfaisant la relation $e_{n+1} = \mathbb{P}e_n$ où e_n représente l'erreur de deux itérations successives aux interfaces(voir section 2.3.3). Cette construction de matrice déduite à partir des erreurs n'apparaît pas dans l'ϵ-algorithme, qui lui n'a besoin que de la suite de solutions à l'interface produite par l'algorithme de Schwarz.

Exemple vectoriel

Considérons maintenant le problème de Lotka-Volterra on $t \in [0, 3]$ suivant :

$$\begin{cases} \dot{u} = uv - u \\ \dot{v} = v - uv \\ u(0) = v(0) = 2 \end{cases} \tag{4.10}$$

Ce système d'équations à valeur initiale nous servira de cas test pour l'accélération de la convergence dans le cas d'une suite vectorielle. Dans la suite, les notations utilisées pour la décomposition de domaine de Schwarz seront celles de la section 2.3. La figure 4.4 représente la solution de ce problème avec la méthode de Schwarz en temps .

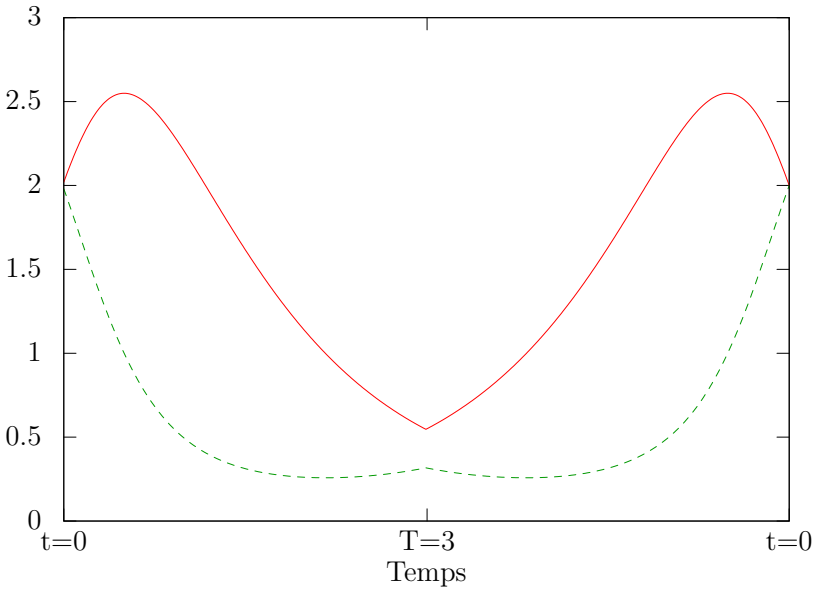


FIGURE 4.4 – Solution du Lotka-Volterra avec l'algorithme de Schwarz en temps pour deux sous-domaines S_1 et $\bar{S}_1$

La figure 4.5 montre la comparaison de la convergence de la méthode de Schwarz, celle Aitken-Schwarz et celle de l'ϵ-algorithme par la première méthode, lorsque le procédé d'accélération est appliqué à partir de la 400 ième itération. Ce chiffre de 400 est basé sur le critère d'arrêt (4.8) à $1.\,10^{-7}$ pour le ϵ-algorithme ce qui se traduit par une valeur du critère d'arrêt de la méthode de Schwarz à 10^{-5}. On injecte alors la solution interface extrapolée par le procédé d'accélération d'Aitken ou par l'ϵ-algorithme dans la méthode de Schwarz en temps. Ce procédé est répété si l'algorithme de Schwarz n'a pas satisfait son critère de convergence. Cette figure montre :

– La convergence du Schwarz en temps pour ce problème est très lente et, plus de 1000 itérations sont nécessaires pour atteindre une erreur à 1.10^{-10}.
– Deux applications de l'ϵ-algorithme sont nécessaires pour atteindre la convergence de $1.\,10^{-10}$ pour l'algorithme de Schwarz en temps.
– L'application de la méthode Aitken-Schwarz à partir de la 400ième itération commence à converger, mais dès la deuxième application l'extrapolation n'est plus correcte ce qui fait diverger la méthode. Ceci illustre la limite de la méthode Aitken-Schwarz pour des problèmes non linéaires.

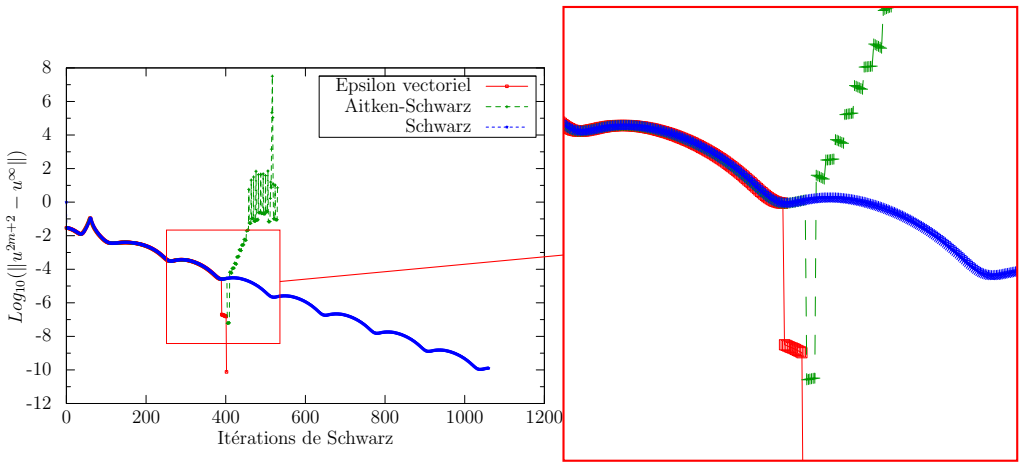


FIGURE 4.5 – Comparaison de la convergence des méthodes de Schwarz Aitken-Schwarz et ϵ-Schwarz(vectoriel) en temps appliquées au problème de Lotka-Volterra

La vitesse de convergence de l'ϵ-algorithme peut être améliorée en étudiant la stabilité numérique [23](quand on utilise la méthode 2). L'instabilité numérique des itérations de base de la suite originale, peut avoir deux conséquences sur la suite (x_n) générée par l'algorithme 4.9, soit une non convergence de la suite produite par (4.9), ou alors une convergence sans assurance que celle-ci soit quadratique.

Considérons des itérations de base de l'algorithme 4.9. Elles peuvent être mises sous la forme suivante avec $h = 1$:

$$V_{k+1} = V_k + h(F(V_k) - V_k) \tag{4.11}$$

En tenant compte des erreurs d'arrondis, l'itéré n'est pas V_k exactement mais $\bar{V_k} = V_k + e_k$, et en dérivant F au voisinage du point fixe x on a,

$$e_{k+1} = [I + h(F'(V_k) - I)]e_k + o(e_k) \tag{4.12}$$

On obtiendra alors un critère de stabilité si la norme de e_k n'augmente pas avec les itérations. En négligeant $o(e_k)$, une condition pour la stabilité est que les valeurs propres de la matrice $I + h(F'(x) - I)$ doivent être plus petites que 1. En notant λ_i les valeurs propres de $F'(x)$, et à condition que toutes les parties réelles de λ_i aient le même signe, il est possible de trouver un h qui satisfait

$$|1 + h(\lambda_i - 1)| < 1 \tag{4.13}$$

.

Appliquons cette technique de stabilité au problème de Lotka-Volterra (4.10) et, comparons la convergence de l'ϵ-algorithme avec différentes valeurs de h. En estimant numériquement les valeurs propres de $F'(x)$ en x_0, on peut trouver le paramètre h. Dans notre cas on obtient $\lambda_i = 0.9808 \pm 0.0238i$, on en déduit que le paramètre optimal est proche de $h = 5$ pour les premières itérations de Schwarz. Au cours des itérations, ce paramètre est amené à évoluer puisqu'il faut évaluer la jacobienne.

Dans le cas d'une fonction F scalaire, la dérivée $F'(x)$ peut s'approximer par $\dfrac{F(v_1) - F(v_0)}{v_1 - v_0}$. Il est alors possible de définir à partir de l'équation (4.13), le paramètre h à chaque renouvellement des itérations de base :

$$0 > h(\frac{F(v_1) - F(v_0)}{v_1 - v_0} - 1) > -2 \tag{4.14}$$

$$0 > h > \frac{-2(v_1 - v_0)}{(F(v_1) - F(v_0)) - (v_1 - v_0)} \tag{4.15}$$

La figure 4.6 résume les résultats de convergence obtenus avec différentes valeurs du paramètre h.

La figure 4.7, montre la comparaison du comportement des différents ϵ-algorithmes vectoriels appliqués à la méthode de Schwarz pour le problème de Lotka-Volterra (4.10). Afin de réaliser cette étude, nous avons conservé les valeurs de la suite générées par le Schwarz jusqu'à convergence de ce dernier. Nous avons les résultats pour 3 versions différentes de l'algorithme : l'ϵ-algorithme vectoriel, l'ϵ-algorithme généralisé, et l'ϵ-algorithme topologique. Le θ-algorithme a été testé mais il ne donne aucun résultat exploitable pour les paramètres testés. L'ϵ-algorithme vectoriel n'inclut pas de paramètre contrairement aux versions généralisée et topologique. La suite de paramètres pour l'ϵ généralisé vectoriel est (n^2), le vecteur paramètre pour l'ϵ-topologique est pris à e^y avec $y_i,\ i = 1, .., n$. La figure 4.7 montre la convergence propre de chaque algorithme en $||u^{2m+2} - u^{2m}||_2$, tandis que la figure 4.8 montre la convergence de chaque algorithme par rapport à la valeur finale de la suite , c'est à dire $||u^{2m+2} - u^{\infty}||_2$. L'$\epsilon$-algorithme vectoriel oscille beaucoup mais

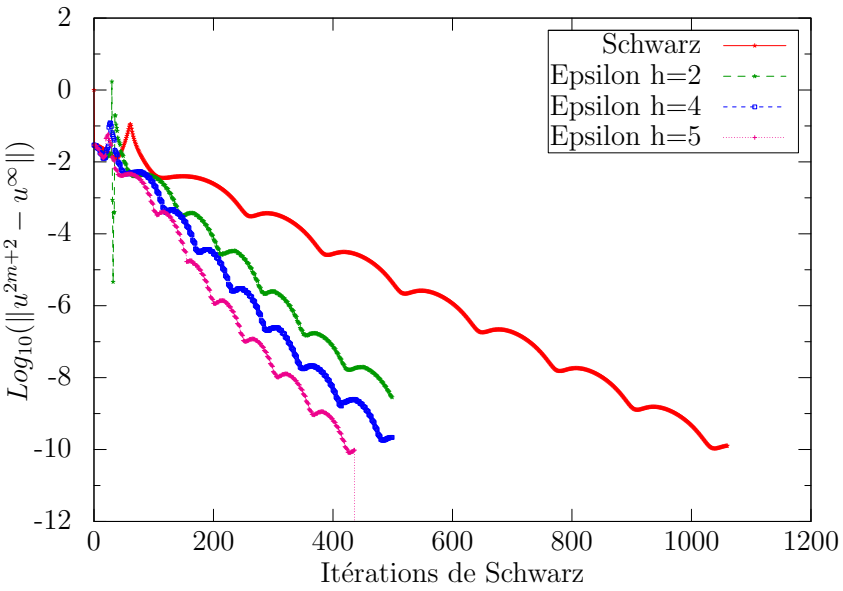


FIGURE 4.6 – Comparaison de la convergence de l'algorithme ϵ-Schwarz pour différentes valeurs du paramètre h sur le problème de Lotka-Volterra avec deux sous-domaines

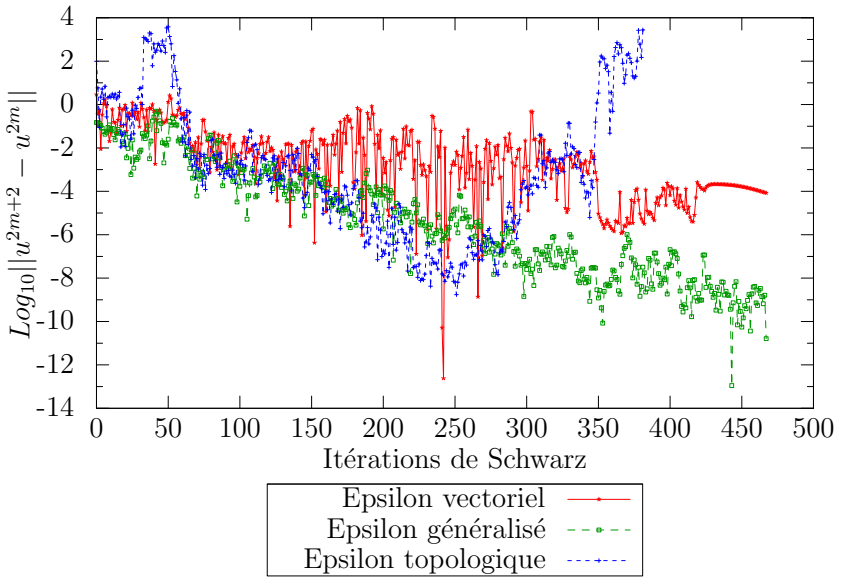


FIGURE 4.7 – Comparaison de la convergence entre deux itérés successifs pour la méthode de Schwarz en temps pour les techniques d'accélération ϵ-algorithmes vectoriel, généralisé et topologique

ne converge pas en deçà de 1.10^{-5}, cependant en comparaison avec la suite originale l'accélération produite à itération égale est non négligeable. Nous pouvons remarquer le même comportement pour la convergence par rapport à la solution finale sur la figure 4.8.

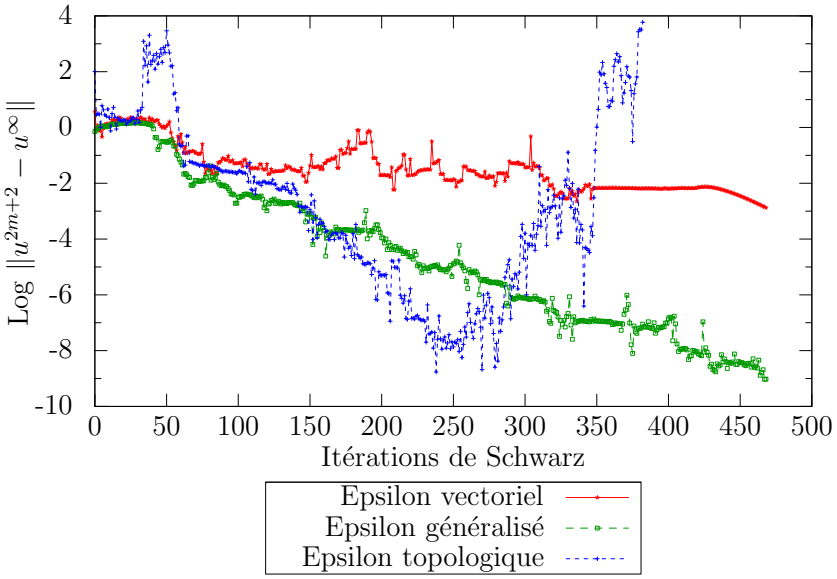


FIGURE 4.8 – Comparaison de la convergence pour la méthode de Schwarz en temps pour les techniques d'accélération ϵ-algorithmes vectoriel, généralisé et topologique par rapport à la solution exacte

4.2.1 Accélération de la méthode de Schwarz par une méthode de Newton

Nous avons vu que nous pouvons accélérer les suites non linéaires de la forme $V^{n+1} = F(V^n)$ avec les différentes versions de l'ϵ-algorithme. Une méthode bien connue pour résoudre les équations non-linéaires est la méthode de Newton. En écrivant la fonction f non linéaire sous la forme $f(V) = V^n - F(V^n) = 0$, la méthode de Newton s'écrit :

$$\begin{cases} f'(V^k)\delta^k = -f(V^k) \\ V^{k+1} = V^k + \delta^k \end{cases} \tag{4.16}$$

L'inconvénient majeur, par rapport aux algorithmes de type *epsilon*, est qu'il faut construire et inverser la jacobienne de f.

Appliquons maintenant cette méthode à l'algorithme de Schwarz en temps. Notons $g_i^N(V_{i-1}^n, V_i^n)$, respectivement $g_i^D(V_{i-1}^n, V_i^n)$, la fonction qui

en fonction des conditions de bords de Dirichlet et Neumann à l'itération n renvoie, après la résolution locale du domaine i, la mise à jour de la condition de Neumann pour le domaine $i-1$, respectivement la condition de Dirichlet pour le domaine $i+1$. On peut ainsi écrire la suite des solutions interfaces en fonction des g_i sous forme d'une suite $V_{n+1} = G(V_n)$.

$$\begin{cases} V_1^{n+1} = g_1^D(cdtbord, g_2^N(V_1^n, V_2^n)) \\ V_2^{n+1} = g_3^N(g_2^D(V_1^n, V_2^n), g_4^N(V_3^n, V_4^n)) \\ V_3^{n+1} = g_3^D(g_2^D(V_1^n, V_2^n), g_4^N(V_3^n, V_4^n)) \\ V_4^{n+1} = g_5^N(g_4^D(V_3^n, V_4^n), g_6^N(V_5^n)) \\ V_5^{n+1} = g_5^D(g_4^D(V_3^n, V_4^n), g_6^N(V_5^n, cdtbord)) \end{cases} \tag{4.17}$$

Nous constatons que la dépendance de données est de quatre au maximum, cela pour une itération de l'algorithme de Schwarz. Il est possible de décider que la fonction F correspond à plusieurs itérations de l'algorithme de Schwarz, ainsi la composante V_i^{n+1} est influée par toutes les autres interfaces, la contrepartie est une complexification de la fonction F et par conséquent un temps de construction plus long et un remplissage de la matrice jacobienne calculée par la méthode de Newton. Nous verrons à l'usage si une telle pratique est bénéfique, ou alors s'il est avantageux de garder la dépendance entre les interfaces au minimum. Le tableau 4.2 présente les résultats de l'étude de la complexité de la fonction F liée aux itérations de Schwarz pour le problème de Lotka-Volterra sur deux domaines. Deux critères sont importants pour bien évaluer l'influence de ces variations, le nombre d'évaluations de la fonction de Newton et la précision atteinte par ce dernier. Les résultats de ce premier tableau révèle clairement que la meilleure pratique est d'une seule itération de Schwarz pour la fonction F. Tout d'abord en termes de précision, les deux meilleures sont obtenues pour 1 et 2, mais pour 2 itérations il y a 30 évaluations de la fonction de Newton, ce qui globalement donne 60 itérations de Schwarz donc le 1 reste le meilleur choix. Pour 5 itérations de Schwarz, la précision n'est que de 1.10^{-8} pour 25 évaluations de la fonction de Newton, soit 100 itérations de Newton.

Le tableau 4.3, présente les résultats pour l'étude avec le problème exponentiel sur quatre domaines. L'analyse des résultats est similaire, la meilleure précision de 1.10^{-12} est atteinte pour 45 évaluations de la fonction de Newton ne comportant qu'une itération de Schwarz. Il n'y a quasiment pas de diminution du nombre d'évaluations de la fonction F en augmentant le nombre d'itérations de Schwarz contrairement à l'exemple précédent. Donc pour 5 itérations de Schwarz par évaluation de fonction F on ne gagne ni en nombre d'évaluations ni en précision, par contre globalement cela augmente le nombre d'itérations de Schwarz. La meilleure pratique consiste à garder la dépen-

itération de Schwarz	évaluation fonction Newton	précision atteinte
1	50	1e-12
2	30	1e-14
3	30	1e-8
4	25	1e-8
5	25	1e-8

TABLE 4.2 – Tableau de comparaison de l'influence du nombre d'itérations de Schwarz pour construire la jacobienne pour le problème de Lotka-Volterra sur deux domaines

dance de données minimale pour la jacobienne, par la suite on verra que la structure de celle-ci est connue.

itération de Schwarz	évaluation fonction Newton	précision atteinte
1	45	1e-12
2	45	1e-11
3	40	1e-9
4	45	1e-9
5	41	1e-10

TABLE 4.3 – Tableau de comparaison de l'influence du nombre d'itérations de Schwarz pour construire la jacobienne pour le problème exponentiel 4.7 sur quatre domaines

On remarque également une similitude dans le procédé d'application de l'accélération par Newton et de l'ϵ-algorithme. La construction de la jacobienne peut être vue comme les itérations de base nécessaires à la deuxième technique d'application de l'ϵ-algorithme. Et la réinjection de la solution calculée est pour la méthode de Newton la mise à jour de la solution initiale après acceptation de la réduction du résidu sur f. Il est évident que la fonction F ne doit pas changer pendant la génération des itérations de base, autrement dit la convergence de l'algorithme de Schwarz doit être indépendante de l'itération n.

La figure 4.9, montre la convergence de la méthode de Newton pour la résolution du problème 4.7 par l'algorithme de Schwarz en temps sur 4 sous-domaines. L'algorithme de Newton converge en 12 itérations à 1.10^{-14}, l'ϵ-algorithme converge en 35 itérations en mode adaptatif, tandis qu'avec la réinjection il atteint la convergence en 19 itérations. Cependant, il faut tenir compte du nombre d'évaluations de fonctions entre chaque itération de la mé-

thode, ce qui correspond au nombre total d'itérations de Schwarz nécessaire. Cette comparaison pour les 3 méthodes utilisées est montrée par la figure 4.10. On constate que malgré le faible nombre d'itérations de la méthode ϵ avec réinjection, celle-ci nécessite un nombre important d'itérations (environ 150) de Schwarz. La méthode adaptative où l'accélération est calculée en même temps que les itérations de Schwarz, montre une très bonne performance et totalise une quarantaine d'itérations de Schwarz. Cette dernière fait jeu égal sur ce problème avec l'accélération par la méthode de Newton.

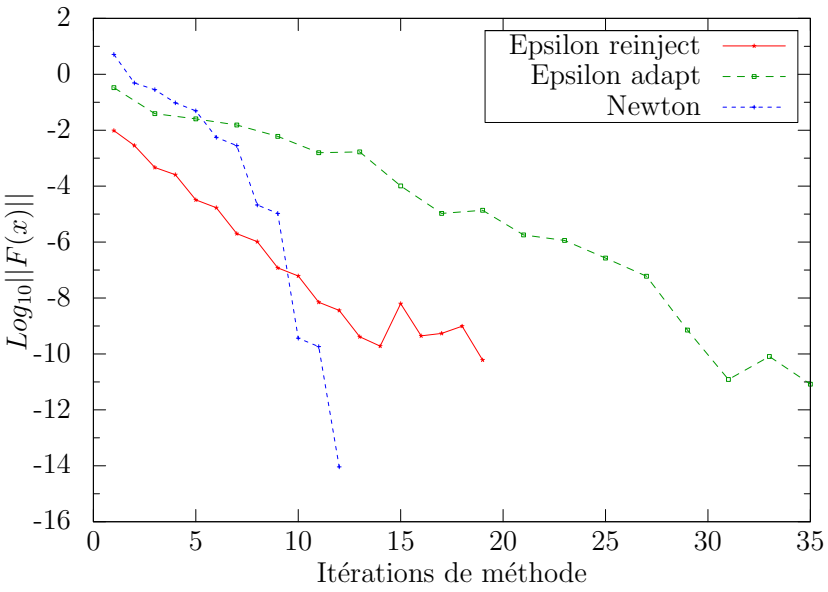


FIGURE 4.9 – Comparaison de la convergence de la méthode de Schwarz pour l'extrapolation par la méthode de Newton et par ϵ-algorithme par rapport au nombre d'itérations de Newton pour le problème de 4.7 sur quatre sous-domaines

.

Nous avons rencontré des difficultés pour tester cette méthode sur un grand nombre de processeurs ; Il faut une bonne solution initiale qui permet la résolution non-linéaire des sous-domaines sans problème. Le contraire entraîne un changement dans la fonction F au cours de ses évaluations pour construire la jacobienne, et cela empêche la convergence de la méthode de Newton. D'autre part, la construction de la jacobienne peut s'avérer très coûteuse. Pour essayer de pallier ce défaut, on pourra s'intéresser aux variantes de type 'jacobian-free' de la méthode de Newton[35].

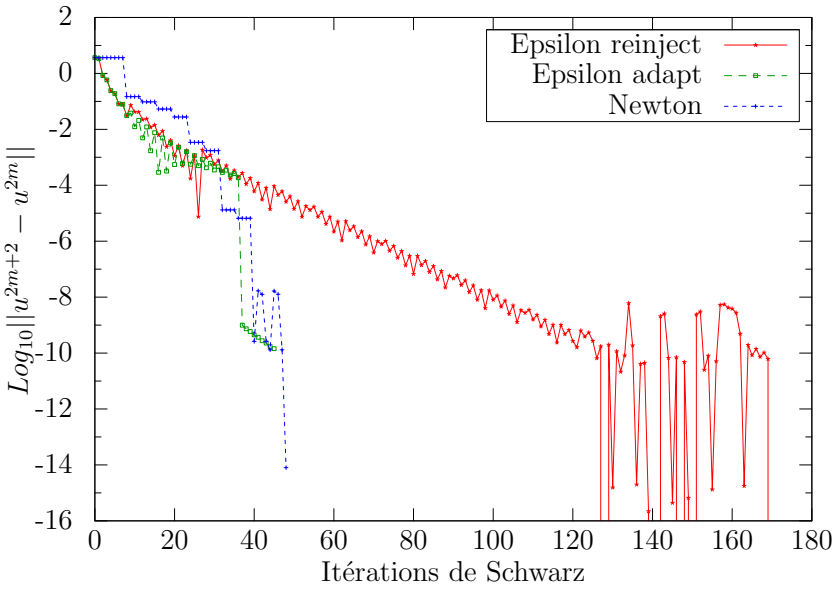


FIGURE 4.10 – Comparaison de la convergence de la méthode de Schwarz pour l'extrapolation par la méthode de Newton et par ϵ-algorithme par rapport au nombre d'itérations de Schwarz pour le problème de 4.7 sur quatre sous-domaines

.

4.2.2 Comportement linéaire

Dans le cas où la suite produite par l'algorithme de Schwarz est linéaire, il suffit alors d'une seule itération de Newton pour obtenir la solution interface. Nous avons testé cette approche de l'accélération de la convergence du Schwarz par la méthode de Newton, pour le problème linéaire de l'oscillateur harmonique. Le but, au-delà de la convergence certaine de la méthode, est de faire le lien avec la méthode d'Aitken-Schwarz. Dans la méthode de Newton, la clé est la jacobienne, il convient donc d'observer celle-ci pour le problème sur deux domaines donc une seule interface à accélérer de taille 4. La jacobienne J obtenue est la matrice diagonale suivante :

$$J = \begin{pmatrix} 125.47672 & 0 & 0 & 0 \\ 0 & 125.47672 & 0 & 0 \\ 0 & 0 & 1.6844342 & 0 \\ 0 & 0 & 0 & 1.6844342 \end{pmatrix} \tag{4.18}$$

La matrice P obtenue par la méthode Aitken-Schwarz est :

$$P = \begin{pmatrix} -124.47672 & 0 & 0 & 0 \\ 0 & -124.47672 & 0 & 0 \\ 0 & 0 & -0.6844342 & 0 \\ 0 & 0 & 0 & -0.6844342 \end{pmatrix} \tag{4.19}$$

On remarque donc que la jacobienne n'est rien d'autre que (Id-P) ! La démonstration s'écrit en partant de la forme condensée du Newton :

$$x^{\infty} = x^0 - J^{-1}F(x^0) \tag{4.20}$$

Nous sommes dans le cas linéaire donc la jacobienne est constante et notée J , l'algorithme donne la solution en une itération d'où le x^{∞}. La notation x^0 avec l'indice du haut pour les itérations de Newton est ici simplifiée en notant simplement les indices de Schwarz x_0. La formule du Newton devient

$$x_{\infty} = x_0 - J^{-1}F(x_0) \tag{4.21}$$

Selon la définition de la fonction F on sait que $F(x_0) = (x_1 - x_0)$,donc

$$x_{\infty} = x_0 - J^{-1}(x_1 - x_0) \tag{4.22}$$

En partant de la formule d'accélération d'Aitken, on obtient :

$$x_\infty = (Id - P)^{-1}(x_1 - Px_0) \tag{4.23}$$
$$(Id - P)x_\infty = (x_1 - Px_0) \tag{4.24}$$
$$x_\infty - Px_\infty + Px_0 = x_1 \tag{4.25}$$
$$(x_\infty - x_0) - P(x_\infty - x_0) = (x_1 - x_0) \tag{4.26}$$
$$(Id - P)(x_\infty - x_0) = (x_1 - x_0) \tag{4.27}$$
$$x_\infty = x_0 + (Id - P)^{-1}(x_1 - x_0) \tag{4.28}$$

On peut donc identifier J à $(Id - P)$.

On a montré l'équivalence entre l'accélération de la convergence du Schwarz par le procédé d'Aitken et l'accélération par la méthode de Newton, et que la construction de la matrice $(Id - P)$ revenait à construire la jacobienne de F. Une remarque importante ici, est que l'on améliore la formule d'Aitken en remplaçant la multiplication matrice-vecteur par une opération addition/soustraction vectorielle beaucoup moins coûteuse. Cependant la technique d'Aitken avec la construction optimale de la matrice P développée au chapitre 2 reste plus rapide que l'implémentation actuelle de la méthode de Newton. En effet, la jacobienne est approximée par différences finies, et est construite pleine par extraction à partir de la fonction F en parcourant la base canonique ; une amélioration pourrait être de se tourner vers les méthodes de Krylov matrix free pour la résolution du système linéaire. Cependant la convergence de la méthode de krylov est très sensible au conditionnement de la matrice (cad le spectre de valeur propre). Sans préconditionnement adéquat cette méthode peut se révéler moins performante. On a discuté dans le cadre de la méthode Newton, du couple extraction et résolution du système faisant intervenir la matrice jacobienne. Pour la méthode d'Aitken, tout le travail a été porté sur la construction optimale de la matrice P étant donnée sa structure creuse. Le système linéaire étant ensuite résolu par une méthode de LU.

L'idée est alors de coupler les deux méthodes pour en prendre la meilleure ; la construction de la matrice P se fait par l'analyse et l'amélioration de la technique d'Aitken. Ainsi on obtient une construction optimale de la jacobienne $J = Id - P$, et la formule d'extrapolation utilisée sera celle de la méthode de Newton. Il reste maintenant à comparer cette méthode améliorée à l'application de l'ϵ-algorithme,dont on rappelle que l'application à une suite linéaire donne la solution en deux itérations de l'ϵ-algorithme, correspondant à $\epsilon_2^{(0)}$ de la figure 3.1.

4.3 Application à la méthode du complément de Schur

Dans le chapitre 1, nous avons vu deux méthodes permettant de satisfaire les conditions de raccords dans la décomposition de domaine. La première, la décomposition de domaine de type Schwarz produit clairement une suite d'itérés aux interfaces. Dans la section précédente nous avons étudié l'application des méthodes d'extrapolation non linéaire à la suite générée par le Schwarz. La deuxième méthode présentée est une technique basée sur le complément de Schur. Pour des systèmes d'ODEs linéaires la méthode se réduit à la résolution d'un système linéaire sur les inconnues, mais dans le cas où le système est non linéaire, la fonction F (2.69) doit être résolue par une méthode de Newton. On voudrait appliquer les techniques décrites dans le chapitre précédent au complément de Schur. Or le système à résoudre s'écrit sous forme $F(x) = 0$, pour l'application des ϵ-algorithmes le système doit s'écrire sous forme d'une suite $x_{n+1} = f(x_n)$. On réécrit alors la fonction F comme :

$$x_{n+1} = x_n + \gamma F(x_n) \tag{4.29}$$

où γ est un nombre non nul ou une matrice carrée inversible. Dès lors, il est possible d'accélérer ce point fixe par l'ϵ-algorithme. Nous comparerons l'efficacité de ces algorithmes avec les deux méthodes d'extrapolation données pour le problème (2.32) pur $t = [0, 5]$ discrétisé avec 200 pas de temps. La figure 4.11 (respectivement 4.12) montre la convergence de la suite vectorielle originale ainsi que l'accélération par l'ϵ-algorithme vectoriel, topologique et généralisé pour 4 domaines (resp. 8 domaines).

Parmi les 3 algorithmes testés, l'ϵ-algorithme vectoriel se montre le moins performant des trois. Le plus efficace est l'ϵ-topologique qui converge en un peu plus de 200 itérations à 1.10^{-11}. L'ϵ-généralisé lui montre un comportement intéressant car il suit de près la convergence de la suite originale, cependant il accélère cette dernière puisqu'il converge à 1.10^{-11} en 450 itérations. L'ϵ-algorithme vectoriel suit le comportement de l'ϵ topologique jusqu'à 1.10^{-8} puis oscille autour de cette valeur, et n'atteint pas la précision Le θ-algorithme a été testé avec cette configuration mais il ne donne pas de résultats concluants.

La figure 4.12, montre les résultats pour le même problème mais avec un découpage sur 8 processeurs. Avec 8 domaines c'est encore l'ϵ-algorithme topologique qui donne la meilleure convergence et qui réussit à accélérer la suite jusqu'à 1.10^{-8}. L'ϵ-algorithme généralisé, montre le même comportement qu'avec 4 domaines, il suit de près la courbe de la suite originale, en accélérant légèrement. Quant à l'ϵ-algorithme vectoriel, il a une convergence

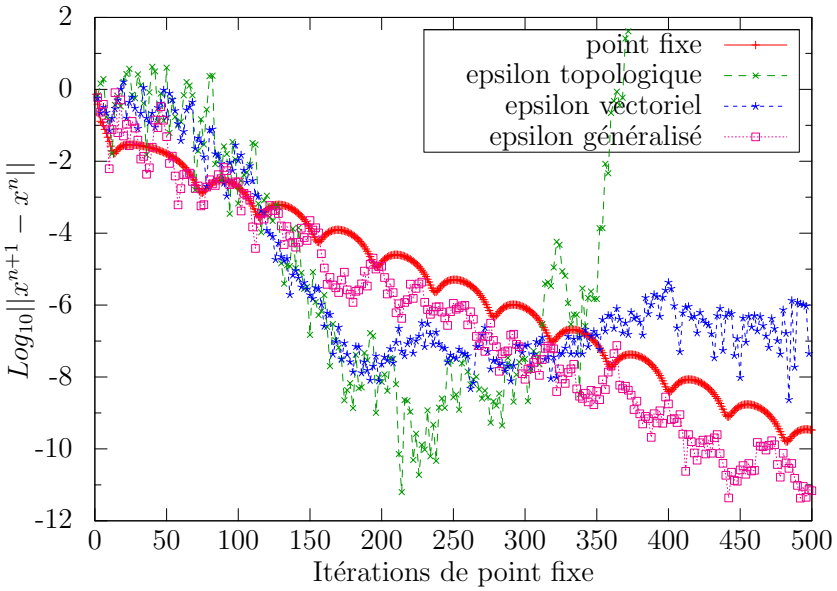


FIGURE 4.11 – Comparaison de l'accélération de la méthode du point fixe de la fonction du complément de Schur des ϵ-algorithmes vectoriel, topologique et généralisé sur 4 sous-domaines

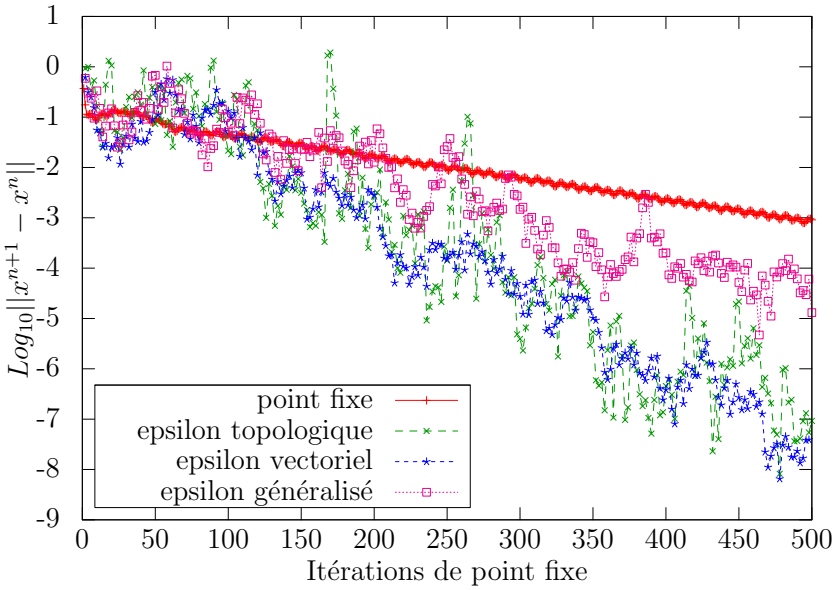


FIGURE 4.12 – Comparaison de l'accélération de la méthode point fixe de la fonction du complément de Schur des ϵ-algorithmes vectoriel, topologique et généralisé sur 8 sous-domaines

proche de l'ϵ topologique jusqu'à 1.10^{-8}

Nous avons testé la première façon d'appliquer les méthodes d'extrapolation de la convergence pour notre algorithme de Schur en récupérant 500 itérés de la suite formée par le point fixe. Les figures suivantes montrent le comportement de ces mêmes algorithmes en utilisant la deuxième méthode, où dès que possible on réintroduit l'extrapolation dans la suite. La figure 4.13 (respectivement 4.14) résume les résultats de convergence des ϵ-algorithmes vectoriel, topologique et généralisé. Avec cette technique d'application, le

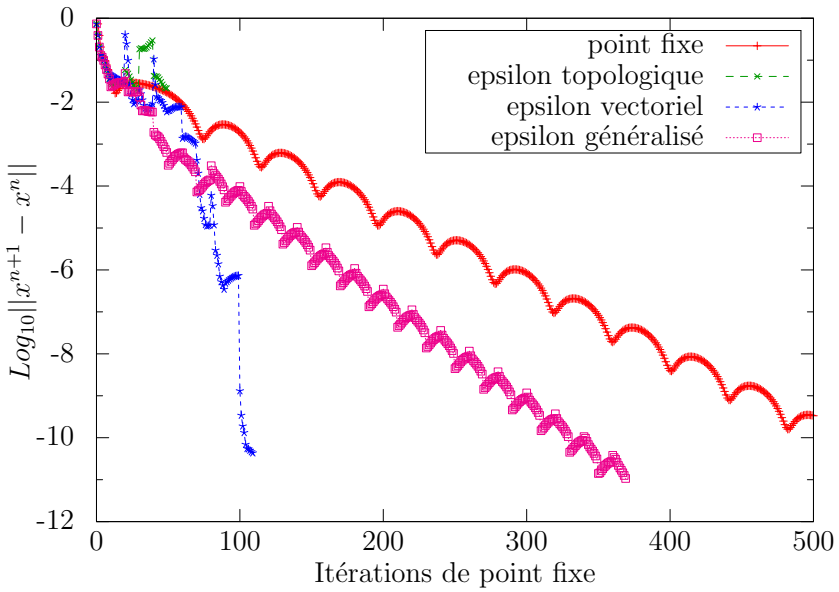


FIGURE 4.13 – Comparaison de l'accélération du point fixe de la fonction du complément de Schur des ϵ-algorithmes vectoriel, topologique et généralisé en réinjectant sur 4 sous-domaines

comportement est totalement différent. La figure 4.13 montre que pour 4 domaines , l'algorithme qui se comporte le mieux est l'ϵ-algorithme vectoriel et converge en 100 itérations. L'ϵ-algorithme généralisé converge en un peu moins de 400 itérations. Cette fois-ci, l'ϵ-topologique ne donne pas de résultats probants et diverge dès les premières itérations.

Pour 8 domaines la figure 4.14 montre que la meilleure méthode sur 4 domaines, n'accélère pas la suite originale pour 8 domaines. L'ϵ-algorithme vectoriel n'extrapole pas de bonnes solutions et finit par diverger au bout de 450 itérations. Le seul à converger est l'ϵ-algorithme généralisé, tandis que l'ϵ-topologique se montre totalement inefficace dès les premières itérations.

En comparant ces deux techniques d'application de l'extrapolation par l'ϵ-algorithme, on remarque que la plus efficace à priori est la première. En effet,

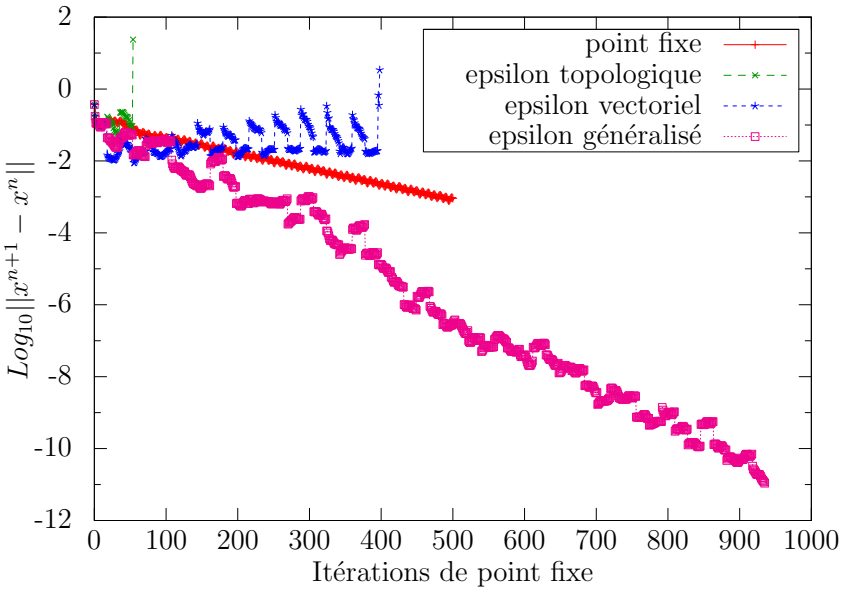


FIGURE 4.14 – Comparaison de l'accélération du point fixe de la fonction du complément de Schur des ϵ-algorithmes vectoriel, topologique et généralisé en réinjectant sur 8 sous-domaines

la meilleure méthode, l'ϵ-topologique converge en 200 itérations, alors que la meilleure méthode pour la deuxième technique converge en 380 itérations. Cependant, en se référant à l'étude de la stabilité numérique de la section 4.1, il est possible pour la technique réinjection d'améliorer la convergence de l'algorithme en ajustant le paramètre h. La figure 4.15 montre l'influence de ce paramètre sur la convergence de l'ϵ-algorithme généralisé pour 4 domaines.

La courbe du paramètre $h = 1$ correspond à l'ϵ-algorithme généralisé normal. Globalement, il est clair que la technique d'extrapolation de la convergence apporte un gain d'une centaine d'itérations à convergence égale. Cependant, entre la meilleure convergence, obtenue avec le paramètre $h = 4$, et la convergence classique ($h = 1$) il y a environ 150 itérations. Cette figure, montre que ce paramètre joue un rôle important dans la convergence de l'algorithme utilisé quand on applique la technique avec ré-injection de l'extrapolation dans la suite. Le paramètre est relié aux valeurs propres de la jacobienne de la fonction suite ; afin d'obtenir la convergence optimale il faudrait calculer ce paramètre h à chaque réinjection (ce qui alourdit considérablement l'algorithme), on peut donc espérer améliorer de façon significative l'efficacité de l'ϵ-algorithme.

Cependant, l'application de l'ϵ-algorithme n'est pas en mesure, dans l'état actuel de l'implémentation, de rivaliser avec la résolution du complément de

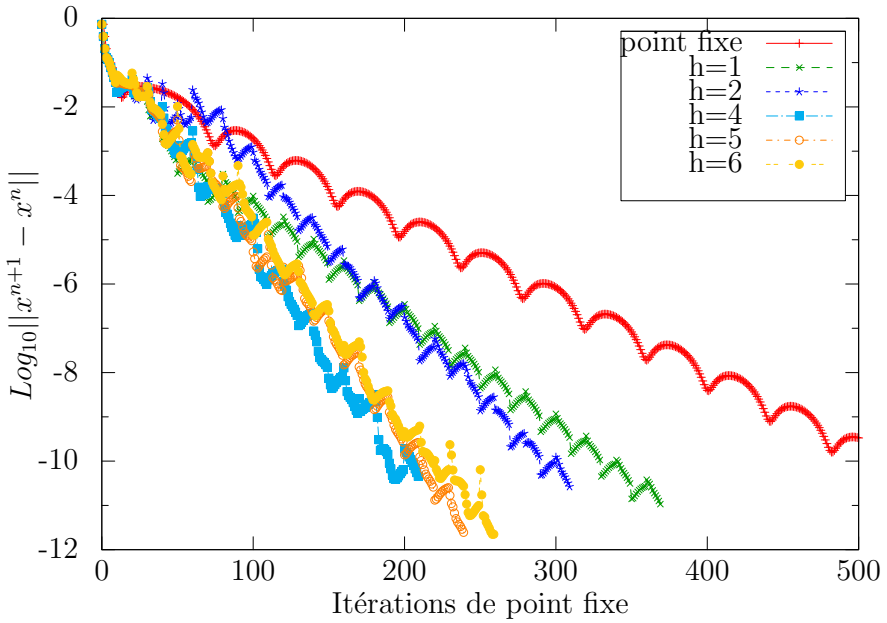


FIGURE 4.15 – Influence du paramètre h sur 'accélération du point fixe de la fonction du complément de Schur des ϵ-algorithmes topologique et généralisé en réinjectant sur 4 sous-domaines

valeurs propres initiales	0.29790853 + 0.37562208i , 0.92669550 + 0.01385364i
valeurs propres finales	0.33506807 + 0.32157447i , 0.95880058 + 0.07391136i

TABLE 4.4 – Valeurs propres de la jacobienne de F pour notre problème à l'instant initial et à convergence.

Procs	Évalutions	Itérations	Temps
4	96	23	0.5
8	73	10	0.13
16	137	10	0.26
32	265	10	0.15

TABLE 4.5 – Tableau récapitulatif du comportement du Newton pour le Schur

Schur par une méthode de Newton pour nos problèmes. Nous utilisons la méthode de Newton NLEQRES basée sur le résidu qui inclut la stratégie adaptative de région de confiance [35]. Cette stratégie permet une adaptation du λ garantissant une réduction du résidu à chaque itération du Newton. La convergence de cette méthode pour notre problème sur 4,8 et 16 processeurs (en sous-domaines) est présentée sur la figure 4.16.

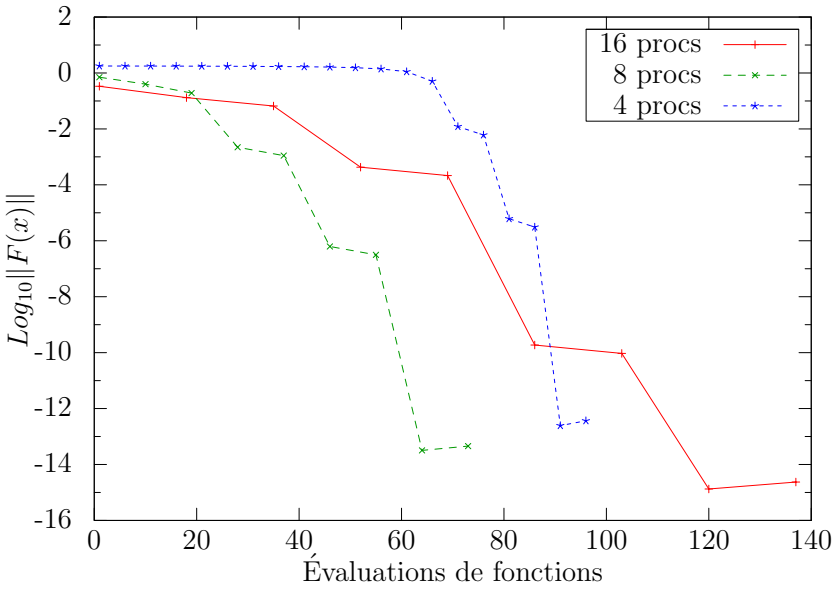


FIGURE 4.16 – Comparaison en terme d'évaluations de fonctions de la convergence de la méthode du complément de Schur en temps résolu par la méthode de Newton en fonction du nombre de sous-domaines

La convergence a été tracée en rapport avec le nombre d'évaluations de la fonction. Pour 4 processeurs, la méthode arrive à convergence en 96 évaluations de fonctions et 20 itérations de Newton. Pour 8 processeurs, la

convergence est atteinte en 9 itérations de Newton pour 73 évaluations de la fonction. Sur 16 processeurs, la méthode de Newton converge encore en 9 itérations, mais la résolution du système de la jacobienne demande plus d'évaluations de fonctions ; ce qui explique les 137 évaluations nécessaires.

Dans notre cas, la résolution par la méthode de Newton reste de loin plus efficace que la méthode d'extrapolation, puisque celle-ci ne parvient pas à convergence en moins de 200 itérations soit près du double.

4.4 Conclusion

L'objectif de cette partie était d'explorer les méthodes d'extrapolation non-linéaire afin d'améliorer la convergence de nos algorithmes de décomposition de domaine. Après un rappel de la théorie de méthodes d'accélération de la convergence, l'algorithme majeur de cette étude, l'ϵ-algorithme a été introduit comme l'algorithme permettant le calcul des déterminants de la transformation de Shanks pour des suites scalaires. Des généralisations de l'ϵ-algorithme ainsi que d'autres algorithmes (ρ et θ) ont été introduits afin d'être le plus complet possible. Nos méthodes de décomposition de domaines génèrent en général des suites vectorielles. L'extension de l'ϵ-algorithme aux suites vectorielles, introduite par Wynn, ne permet pas de garder la relation avec la transformation de Shanks. En effet, l'algorithme de l'ϵ-vectoriel a été adapté en définissant l'inverse d'un vecteur. C.Brezinski a alors crée l'ϵ-topologique qui conserve les propriétés de la transformation de Shanks pour le cas vectoriel.

Ainsi nous avons appliqué d'abord ces techniques d'accélération à notre algorithme de Schwarz en temps développé dans la partie 1. Deux techniques d'application s'offrent à nous, soit en attendant qu'il y ait suffisamment de termes de la suite pour atteindre la convergence avec l'algorithme, soit en modifiant la suite originale en réinjectant le résultat d'une extrapolation d'un certain nombre de termes. Les résultats numériques ont permis de comparer le comportement des différents algorithmes entre eux, ainsi qu'avec la méthode de Newton-Schwarz. Cette dernière a été développée en réécrivant le Schwarz comme une fonction devant s'annuler à convergence. La méthode de Newton peut alors être vue comme une méthode d'accélération de la convergence du Schwarz. Nous avons également appliqué ces techniques d'extrapolation à la méthode du complément de Schur.

Dans le cas de problème non-linéaire, celle-ci a d'abord été présentée dans la partie 1 mise sous forme d'une fonction à annuler par une méthode de Newton. Afin d'appliquer les algorithmes de cette partie, nous avons transformé cette fonction en une fonction générant une suite qui tend vers zéro en l'infini.

Les résultats ont été comparés avec ceux obtenus par la méthode de Newton.

Nos expériences numériques ont montré que l'efficacité numérique des différentes variantes de l'ϵ-algorithme dépend de la suite (varie avec le problème et le nombre de sous-domaines) et de la méthode d'application de l' ϵ-algorithme. On déduit qu'il n'y a pas un algorithme universel pour accélérer nos décompositions de domaine telles qu'elles sont formulées pour un problème non linéaire général. Une solution peut être de tester en parallèle chaque méthode pour déterminer celle qui se comporte le mieux dans une configuration donnée. Une autre hypothèse pourrait être que le choix de raccords "linéaires" comme Neumann-Dirichlet pour la décomposition de domaine de problèmes non linéaires, n'est peut être pas le choix optimal pour produire des suites non linéaires qui se prêteraient bien à ce type d'accélération. Ceci nous amène à considérer, dans la partie suivante, l'étude de conditions de raccords non linéaires pour la décomposition de domaine de problèmes non linéaires.

Troisième partie

Formalisme hamiltonien à ports et développement de conditions de transmission non-linéaires pour la décomposition de domaine

Table des matières

Chapitre 5

Développement d'une bibliothèque bondgraph

Table des matières

Introduction

Les résultats de l'accélération de suites non-linéaires n'ont pas permis d'avoir une réponse "tranchée" sur le bénéfice de l'accélération de ces suites. Ceci nous amène à investiguer les conditions de raccords non linéaires pour la décomposition de domaine pour les problèmes non-linéaires.

Notre approche a consisté, dans un premier temps, à acquérir la méthodologie bond-graph et des structures mathématiques hamiltoniennes à port qui s'y affèrent, développées dans le domaine de l'automatique, par le Laboratoire d'Automatique et de Génie des procédés qui permette de conserver la puissance (produit de l'effort par le flux) et de la transposer dans le domaine de la décomposition de domaine. L'idée est de préserver les quantités physiques invariantes entre les sous-domaines.

Nous allons dans ce chapitre 5, expliquer le formalisme hamiltonien à ports des systèmes dynamiques en dimension finie, avec la mise au point

d'une bibliothèque utilisant le logiciel SUNDIALS et intégrant les éléments de bases au travers de deux exemples, l'un linéaire, la ligne de transmission et l'autre, non linéaire, la colonne d'adsorption. Ces travaux effectués dans le cadre du projet ANR PARADE ont permis de briser la causalité de la formulation hamiltonienne à ports dans les schémas bond-graph.

Dans le chapitre 6, nous appliquons cette technique au problème non linéaire de Saint-Venant requérant le schéma d'intégration en espace et en temps de Preissman non inclus dans SUNDIALS.

Dans le chapitre 7, nous faisons une analyse de la convergence des suites non linéaires pour le problème de l'équation de la chaleur non linéaire stationnaire. Nous montrons que les conditions de raccord de type Neumann-Dirichlet ne sont pas appropriées dans ce cas. Ceci nous amène à proposer des conditions de raccords non linéaires imposées par le formalisme hamiltonien. Enfin nous proposons une approche de type multiplicateur de Lagrange où la contrainte (séparable) des raccords non linéaires est dualisée.

5.1 Formulation hamiltonienne à ports des systèmes dynamiques de dimension finie

Dans ce chapitre l'objectif est de créer une base afin de développer et tester des modèles bondgraphs compliqués. Pour cela il est nécessaire d'extraire la généricité des éléments de base des bondgraphs afin de pouvoir les interconnecter entre eux. Deux exemples illustreront l'utilisation de la bibliothèque bondgraph.

La modélisation hamiltonienne des systèmes mécaniques est donnée par les équations dites de Hamilton qui sont obtenues à partir des équations d'Euler-Lagrange en utilisant une transformation de Legendre. Physiquement elles représentent une condition nécessaire d'optimalité d'une fonctionnelle le long de la dynamique d'un système donné [1, 3]. L'équation d'Euler-Lagrange d'un système mécanique est donnée comme suit :

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}(q,\dot{q})\right) - \frac{\partial \mathcal{L}}{\partial q}(q,\dot{q}) = \tau \tag{5.1}$$

où q est la coordonnée généralisée du système, $\mathcal{L}$ le lagrangien qui est égal à la différence entre l'énergie cinétique E_c et l'énergie potentielle E_p ($\mathcal{L} = E_c - E_p$) et τ est la force généralisée agissant sur le système. L'énergie cinétique d'un système mécanique est généralement donnée comme suit :

$$E_c = \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{5.2}$$

où $M(q) > 0$ est la matrice d'inertie du système. On définit le moment généralisé p et le Hamiltonien par :

$$\begin{cases} p = \dfrac{\partial \mathcal{L}}{\partial \dot{q}} \\ H = \dot{q}p - \mathcal{L} \end{cases} \tag{5.3}$$

qui ne sont qu'une transformation de Legendre du lagrangien $\mathcal{L}$ [3]. Pour un système mécanique dont l'énergie cinétique est donnée par l'expression (5.2), le moment généralisé sera simplement :

$$p = M(q)\dot{q} \tag{5.4}$$

Ainsi le Hamiltonien aura pour expression :

$$\begin{aligned} H(q,p) &= \dot{q}p - \mathcal{L} && (5.5)\\ &= \dot{q}M(q)\dot{q} - (\frac{1}{2}\dot{q}^T M(q)\dot{q} - E_p(q)) && (5.6)\\ &= \frac{1}{2}\dot{q}^T M(q)\dot{q} + E_p(q) && (5.7)\\ &= E_c + E_p && (5.8) \end{aligned}$$

Cette dernière équation montre bien que le Hamiltonien n'est autre que l'énergie totale du système. Ainsi, l'équation d'Euler-Lagrange qui représente une équation différentielle de second ordre peut être réécrite sous la forme d'un système d'équations différentielles du premier ordre :

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \dfrac{\partial H}{\partial q} \\ \dfrac{\partial H}{\partial p} \end{bmatrix} + \begin{bmatrix} 0 \\ \tau \end{bmatrix} \tag{5.9}$$

le vecteur $[\dfrac{\partial H}{\partial q}, \dfrac{\partial H}{\partial p}]^T$ n'est autre que le vecteur des efforts donnés comme les dérivées de l'énergie totale par rapport aux variables q et p. Ce sont des variables intensives. Les dérivées $(\dot{q}, \dot{p})$ définissent des variables extensives qu'on appelle variables de flux. Les variables de flux et d'effort sont des variables duales (conjuguées) appelées aussi variables de puissance car leur produit est homogène à une puissance.
En calculant la variation d'énergie totale du système, on obtient :

$$\frac{dH}{dt} = \begin{bmatrix} \dfrac{\partial H}{\partial q} & \dfrac{\partial H}{\partial p} \end{bmatrix}^T \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \dfrac{\partial H}{\partial q} \\ \dfrac{\partial H}{\partial p} \end{bmatrix} + \frac{\partial H}{\partial p}^T \tau \tag{5.10}$$

On note

$$J = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{5.11}$$

J est une matrice antisymétrique ($J^T = -J$). Cela implique donc :

$$\forall x \in \mathbb{R}^2 : x^T J x = 0 \tag{5.12}$$

Finalement la variation d'énergie du système est :

$$\frac{dH}{dt} = \frac{\partial H}{\partial p}^T \tau = \dot{q}^T \tau \tag{5.13}$$

c'est-à-dire le produit de la vitesse et des forces extérieures. Ce produit définit bien la puissance extérieure communiquée au système. Donc l'équation (5.13) implique que l'accroissement de l'énergie du système est égal à la puissance qui lui est transmise par les forces extérieures. On en déduit que si la force généralisée (τ) est l'entrée (u) du système, alors la vitesse $\dot{q} = \frac{\partial H}{\partial p}$ est la sortie duale (y) du système. Cela nous permet d'écrire le bilan (5.13) :

$$\frac{dH}{dt} = y^T u \tag{5.14}$$

On peut écrire alors le système dynamique sous la forme :

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tau \tag{5.15}$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial H}{\partial q} \\ \frac{\partial H}{\partial p} \end{bmatrix} \tag{5.16}$$

Le modèle (5.15,5.16) considéré est celui d'un système conservatif (et passif) pour l'entrée u et la sortie y et pour une fonction d'énergie $H(q,p)$ [74].

$J^T(x) = -J(x) \in \mathbb{R}^{n \times n}$ est la matrice d'interconnexion qui définit les couplages inter-domaines à l'intérieur du système et $g(x)$ est la matrice d'entrée du système. La représentation des systèmes physiques est appelée représentation hamiltonienne à ports.

Afin de pouvoir considérer une dissipation dans la formulation hamiltonienne à ports, on ajoute sur certains ports (notés (u_R, y_R)) du système des éléments

dissipatifs définis par une relation constitutive entre les variables d'effort et de flux comme suit :

$$u_R = -F(y_R) \quad \text{avec} \quad y_R^T F(y_R) \geq 0 \tag{5.17}$$

Formulation hamiltonienne à ports implicite Dans la procédure de modélisation d'un système physique, on traduit par des équations mathématiques les contraintes liant entre elles les variables d'état du système. Ces contraintes s'expriment par des systèmes d'équations différentielles linéaires ou non, et généralement implicites (i.e. $f(x, \dot{x}, t) = 0$). Il n'est pas toujours aisé, ni même possible, de rendre explicite (i.e. $\dot{x} = f(x, t)$) ce type d'équations issues de la physique qui n'ont pas nécessairement été développées dans une approche de type "système". Il s'avère donc important de généraliser la modélisation hamiltonienne à ports pour les systèmes dits implicites. Pour cela, on introduit la notion de structure de Dirac permettant de généraliser la structure d'interconnexion définie par la matrice $J(x)$.

Une structure de Dirac (notée $\mathcal{D}$) représente la structure d'interconnexion des différents éléments composant un système physique [33, 36, 34]. Une structure de Dirac peut être définie par différentes représentations : noyau et image, entrées-sorties, flux contraints, efforts contraints et canoniques.

Définition 5.1 (Représentation noyau de la structure de Dirac)**.** Soient $\mathcal{F}$ et $\mathcal{E}$ les espaces vectoriels des variables de ports conjuguées. Supposons $\mathcal{F} = \mathcal{E} = \mathbb{R}^K$ où K est le nombre fini de ports du système. Soit l'ensemble

$$\mathcal{D} = \{(f, e) \in \mathcal{F} \times \mathcal{E} : Ff + Ee = 0\}$$

où $F, E \in \mathbb{R}^{K \times K}$. Alors $\mathcal{D}$ définit une structure de Dirac sur l'espace des variables de ports $\mathcal{F} \times \mathcal{E}$ si et seulement si :

1. $EF^T + FE^T = 0$
2. $rang[F : E] = K$

La première condition impose que la puissance totale entrant dans la structure est nulle à tout instant. La deuxième condition fait en sorte qu'il soit toujours possible, à l'aide des équations de la structure, de déterminer une moitié des variables de ports en fonction de l'autre moitié. Il est donc nécessaire d'imposer soit l'effort, soit le flux, à chaque port de la structure de Dirac dont les équations permettent alors de calculer la valeur de la variable conjuguée (à celle imposée) à chaque port.

Proposition 5.1. *Soient $\mathscr{F}$ et $\mathscr{E}$ respectivement l'espace des variables de flux et d'efforts. Soit $\mathscr{D}$ un sous-espace de $\mathscr{F} \times \mathscr{E}$, alors $\mathscr{D} = (e^q, e^p, f^q, f^p, e^0_\partial, e^L_\partial, f^0_\partial, f^L_\partial)$ tel que*

$$\begin{cases} \begin{pmatrix} f^p \\ f^q \end{pmatrix} = \begin{pmatrix} 0 & -d \\ -d & 0 \end{pmatrix} \begin{pmatrix} e^p \\ e^q \end{pmatrix} \\ \begin{pmatrix} f^0_\partial \\ f^L_\partial \\ e^0_\partial \\ e^L_\partial \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} e^p(t,0) \\ e^p(t,L) \\ e^q(t,o) \\ e^q(t,L) \end{pmatrix} \end{cases} \tag{5.18}$$

définit une structure de Dirac i.e. $\mathscr{D} = \mathscr{D}^\perp$, par rapport au produit de puissance $\ll . \gg$.

Un système physique peut être vu comme un ensemble d'éléments de natures différentes (éléments d'accumulation d'énergie ou de dissipation ...) connectés entre eux par une structure géométrique qui conserve la puissance mise en échange (Fig.5.1). Cette structure est représentée analytiquement par une structure de Dirac définie sur l'espace des variables de flux $\mathcal{F} = \mathcal{F}_\mathcal{S} \times \mathcal{F}_\mathcal{R} \times \mathcal{F}_\mathcal{P}$, et l'espace des variables d'effort $\mathcal{E} = \mathcal{E}_\mathcal{S} \times \mathcal{E}_\mathcal{R} \times \mathcal{E}_\mathcal{P}$. Dans ces espaces de variables des ports, $\mathcal{F}_\mathcal{S}$ (respectivement $\mathcal{E}_\mathcal{S}$) représente l'espace de variables de flux (respectivement d'efforts) connectées aux éléments qui accumulent de l'énergie, $\mathcal{F}_\mathcal{R}$ (respectivement $\mathcal{E}_\mathcal{R}$) représente l'espace des variables de flux (respectivement d'efforts) connectées aux éléments qui dissipent de l'énergie, $\mathcal{F}_\mathcal{P}$ (respectivement $\mathcal{E}_\mathcal{P}$) représente l'espace des variables de flux (respectivement d'efforts) par lesquelles le système échange de l'énergie avec l'environnement extérieur. Les équations constitutives des différents éléments peuvent alors être vues comme des équations de fermeture entre les variables de port de la structure de Dirac. Par exemple, les variables de flux et d'effort des éléments d'accumulation sont liées par des équations dynamiques du type :

$$f_S = -\dot{x}(t) = -\frac{dx}{dt}(t) \tag{5.19}$$

$$e_S = \frac{\partial H}{\partial x}(x(t)) \tag{5.20}$$

Le signe affectant le flux est conventionnel. Il est défini de manière à ce que la puissance sortante de l'élément d'accumulation $-\frac{\partial H}{\partial x}\dot{x}(t)$ et entrante dans la structure puisse s'écrire $e_S f_S$. La dynamique du système peut donc être décrite par la condition d'appartenance des variables de ports à la structure

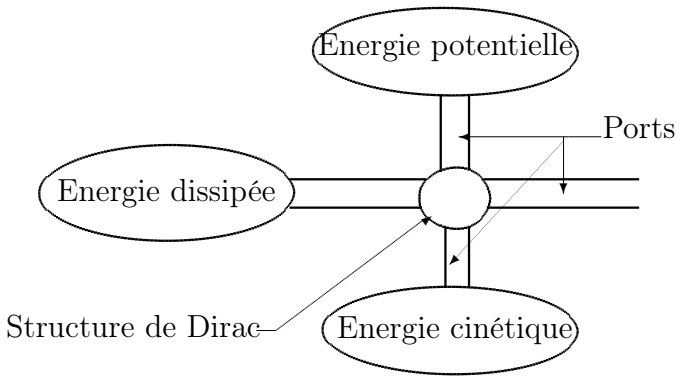


FIGURE 5.1 – Décomposition en sous-systèmes des systèmes physiques

de Dirac :

$$(-\dot{x}(t), f_R, f_P, \frac{\partial H}{\partial x}(x(t)), e_R, e_P) \in \mathcal{D} \subset (\mathcal{F}_\mathcal{S} \times \mathcal{F}_\mathcal{R} \times \mathcal{F}_\mathcal{P}) \times (\mathcal{E}_\mathcal{S} \times \mathcal{E}_\mathcal{R} \times \mathcal{E}_\mathcal{P}) \tag{5.21}$$

où les variables (f_R, e_R) sont reliées par les équations constitutives des éléments dissipatifs (éventuellement implicites) et où les variables de ports e_P ou f_P sont déterminées par la réalisation physique particulière de l'échange de puissance avec l'environnement.

La table 5.1 résume quelques exemples de variables effort et flux dans différents domaines de la physique.

Mécanique translation	Force $F(t)$	Vitesse $V(t)$
Mécanique rotation	Couple $\tau(t)$	Vitesse angulaire $\omega(t)$
Hydraulique	Pression $p(t)$	Débit $Q(t)$
Électrique	Tension $U(t)$	Courant $I(t)$

TABLE 5.1 – Quelques exemples de variables effort et flux

5.2 Les éléments du langage Bondgraph

On définit dans cette partie l'ensemble des multiports nécessaires pour la modélisation [55] d'un système physique d'une manière générique en utilisant les variables généralisées d'effort et de flux.

Ces éléments de base se classifient en :

- trois éléments passifs (**R** ,**C** et **I**) ;
- deux éléments actifs (**Se** et **Sf**)
- quatre éléments (**1** , **0** , **TF** et **GY**).

Les éléments sont dits passifs car ils transforment la puissance qui leur est fournie en puissance dissipée (élément **R**), stockée sous forme d'énergie potentielle (élément **C**) ou cinétique (élément **I**). Ces éléments sont généralement à 1 port dans le cas des systèmes monoénergétiques car ils reçoivent de la puissance par un seul lien. Comme la puissance est fournie à ces éléments, la demi-flèche est orientée vers l'élément.

L'élément **R** à 1 port modélise tout phénomène qui dissipe de l'énergie : la puissance qui lui est fournie est transformée en énergie thermique dissipée.

L'élément **C** à 1 port modélise tout système qui transforme la puissance qui lui est fournie en énergie potentielle stockée. La loi générique qui le caractérise est une relation dynamique liant l'intégrale du flux (soit le déplacement généralisé) et l'effort.

L'élément **I**(inertie) est aussi un élément de stockage d'énergie. L'élément **I** transforme la puissance qui lui est fournie en énergie cinétique pour les systèmes mécaniques et magnétiques pour les systèmes électriques. La loi générique qui le caractérise est une relation dynamique liant l'intégrale de l'effort (soit l'impulsion) et le flux.

Les éléments dits actifs sont ceux qui fournissent de la puissance au système. On distingue les sources d'effort **Se** et de flux **Sf** .

Ces éléments, notés **0** ,**1**, **TF** et **GY** servent à coupler les éléments **R**, **C** et **I** pour construire la structure de jonctions du système à modéliser. Ils sont tous conservatifs de puissance. Pour reproduire l'architecture du système à modéliser, les éléments **R** ,**C** et **I** sont couplés entre eux par des jonctions **0** lorsqu'ils sont soumis à un même effort et par des jonctions **1** lorsqu'ils sont soumis à un même flux. La pondération sur les flux (ou les efforts) se fait avec un signe (+) ou (-) suivant que la demi-flèche est entrante dans la jonction **1** (ou **0**) ou sortante.

Les phénomènes de transformation de puissance sont représentés par les éléments **TF** (transformateur) et **GY** (gyrateur) définis par :

$$\mathbf{TF} \quad : \quad \begin{cases} e_1 = m e_2 \\ f_2 = m e_1 \end{cases} \tag{5.22}$$

$$\mathbf{GY} \quad : \quad \begin{cases} e_1 = r f_2 \\ e_2 = r f_1 \end{cases} \tag{5.23}$$

5.2.1 Mise en équation implicite pour l'utilisation de SUNDIALS

Dans SUNDIALS, la résolution se fait de manière implicite en réécrivant le problème sous la forme $F(t, \dot{u}, u) = 0$. Le problème ne s'exprime alors plus sous forme de causalité mais uniquement par les relations algébriques entre les variables. La résolution de ce problème se fait donc par une méthode de type Newton avec une discrétisation BDF de la variable en temps. La condition initiale doit être choisie avec soin ; en effet, la convergence du Newton est très sensible à la condition initiale, et cela affecte également la vitesse de convergence de ce dernier. Pour pallier ce problème d'initialisation, les variables sont initialisées grâce à la causalité du bondgraph.

$$F(t, \dot{u}, u) = \big(M\big)_{n\times n} \begin{pmatrix} \dot{u_1} \\ \vdots \\ \dot{u_n} \end{pmatrix} - \big(A\big)_{n\times n}(t) \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = 0 \tag{5.24}$$

Chaque élément bondgraph est décrit avec le prototype générique suivant :

```
subroutine element (var_entree,var_sortie,fct_discret,residu,...
  params(optionnel))
residu=var_sortie - fct_discret(var_entree)
end subroutine
```

Ainsi la fonction $F(t, \dot{u}, u) = 0$ que l'on demande à SUNDIALS de minimiser est une succession d'appels des différents éléments afin de constituer le résidu général.

La bibliothèque a été développée avec une discrétisation par collocation. Cependant n'importe quelle discrétisation peut être utilisée à condition d'être compatible avec la taille des variables. Nous illustrons, au travers de deux exemples, les transformations apportées par la bibliothèque bondgraph.

5.2.2 Exemple avec la ligne de transmission

Dans l'exemple qui suit, nous allons illustrer la notion de structure de Dirac en 1D sur le cas d'une ligne de transmission idéale i.e sans dissipation à une dimension d'espace sur l'intervalle [0, L]. La formulation mathématique de ce problème est donnée par des équations aux dérivées partielles hyperboliques connues sous le nom d'équation des télégraphistes, notre objectif étant d'illustrer la reformulation sous forme hamiltonienne à ports qui en est faite, et d'écrire la structure de Dirac en dimension infinie [38]. Les équations des

télégraphistes découplées s'écrivent

$$\begin{cases} \dfrac{\partial q}{\partial t} = -\dfrac{\partial I}{\partial z} \\ \dfrac{\partial p}{\partial t} = -\dfrac{\partial V}{\partial z} \end{cases} \tag{5.25}$$

où $q(t,z)$ représente la charge, $p(t,z)$ le flux, I le courant électrique et V la différence de potentiel. Avec les identifications suivantes :

- f_q et f_p sont des 1-formes représentant respectivement la charge dq/dt, et le flux infinitésimaux dp/dt.
- e_q et e_p des 0-formes représentant respectivement la différence de potentiel et le courant électrique.
- e_∂^0 et e_∂^L sont les valeurs aux bords de la différence de potentiel respectivement à gauche et à droite de la ligne considérée.
- f_∂^0 et f_∂^L sont les valeurs aux bords du courant électrique respectivement à gauche et à droite.

La représentation hamiltonienne des équations est donnée par

$$\begin{pmatrix} f_q \\ f_p \end{pmatrix} = \begin{pmatrix} 0 & -\dfrac{\partial}{\partial z} \\ -\dfrac{\partial}{\partial z} & 0 \end{pmatrix} \begin{pmatrix} eq \\ e_p \end{pmatrix} \tag{5.26}$$

Avec les variables de bords associées

$$\begin{cases} e_\partial^0 = e_q(t,0), e_\partial^L = e_q(t,L) \\ f_\partial^0 = e_p(t,0), f_\partial^L = e_p(t,L) \end{cases} \tag{5.27}$$

Nous nous basons sur le problème de la ligne de transmission suivant :

$$\begin{cases} \mathbb{C} \begin{cases} \dfrac{dq}{dt} = f_q \\ e_p = mat_1 * q \end{cases} \\ \mathbb{GY} \begin{cases} \begin{pmatrix} f_q \\ f_p \end{pmatrix} = Jmat \begin{pmatrix} e_q \\ e_p \end{pmatrix} \end{cases} \\ \mathbb{C} \begin{cases} \dfrac{dp}{dt} = f_p \\ e_q = mat_2 * p \end{cases} \end{cases} \tag{5.28}$$

Dans un objectif de généricité , la discrétisation utilisée dans les éléments est un paramètre des routines. L'élément $\mathbb{C}$ est donc caractérisé en entrée par la variable de flux et la matrice de discrétisation. En sortie, on récupère la variable d'effort.

```
subroutine Celem_implicite(e_out,x,fct,res)
```

L'élément $\mathbb{GY}$ est caractérisé en entrée par les 2 variables d'effort et la matrice $jmat$ de discrétisation. Les variables de sortie sont celles des flux.

```
subroutine GYelem_implicite(e1_in,e2_in,e1_0,e2_0,f1_out,
                            f2_out,jmat,res_1,res_2)
```

Pour utiliser ces deux éléments, il est donc nécessaire d'initialiser les matrices de discrétisation $Jmat$,mat_1 et mat_2 par la méthode souhaitée (éléments finis, éléments spectraux, etc.). Puis l'on assemble les éléments entre eux conformément au bondgraph voulu.

Le LAGEP résolvait ce problème de manière explicite. L'intégration de x_1 et x_2 était réalisée par un Euler explicite au temps t_n, puis le calcul des efforts e correspondant était fait. La mise à jour des flux f permettait le passage au temps t_{n+1}. L'intégration par un Stormer-Verlet explicite à été mise en place dans les travaux de R.Moulla [71]

Ceci se traduit par les appels successifs suivants des modules : Dans SUNDIALS, la résolution se fait de manière implicite en réécrivant le problème sous la forme $F(t, \dot{u}, u)$:

De l'exemple ci-dessus, on obtient le système :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{e_1} \\ \dot{e_2} \end{pmatrix} - \begin{pmatrix} 0 & 0 & Jmat11 & Jmat12 \\ 0 & 0 & Jmat21 & Jmat22 \\ mat_1 & 0 & -1 & 0 \\ 0 & mat_2 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ e_1 \\ e_2 \end{pmatrix} = 0$$

La programmation de la fonction $F(t, \dot{u}, u)$ se fait à partir des éléments bondgraph($\mathbb{C}$...) et de la discrétisation associée. Puis cette fonction est passée à SUNDIALS pour la résolution en temps.

```
subroutine transline_sundial (u,up,res)
Appel de l'élément GYelem_implicite
Appel de l'élément Celem_implicite
Appel de l'élément Celem_implicite
end subroutine
```

L'appel est alors le suivant :

```
call init_transline(mat_1,mat_2,jmat)
```

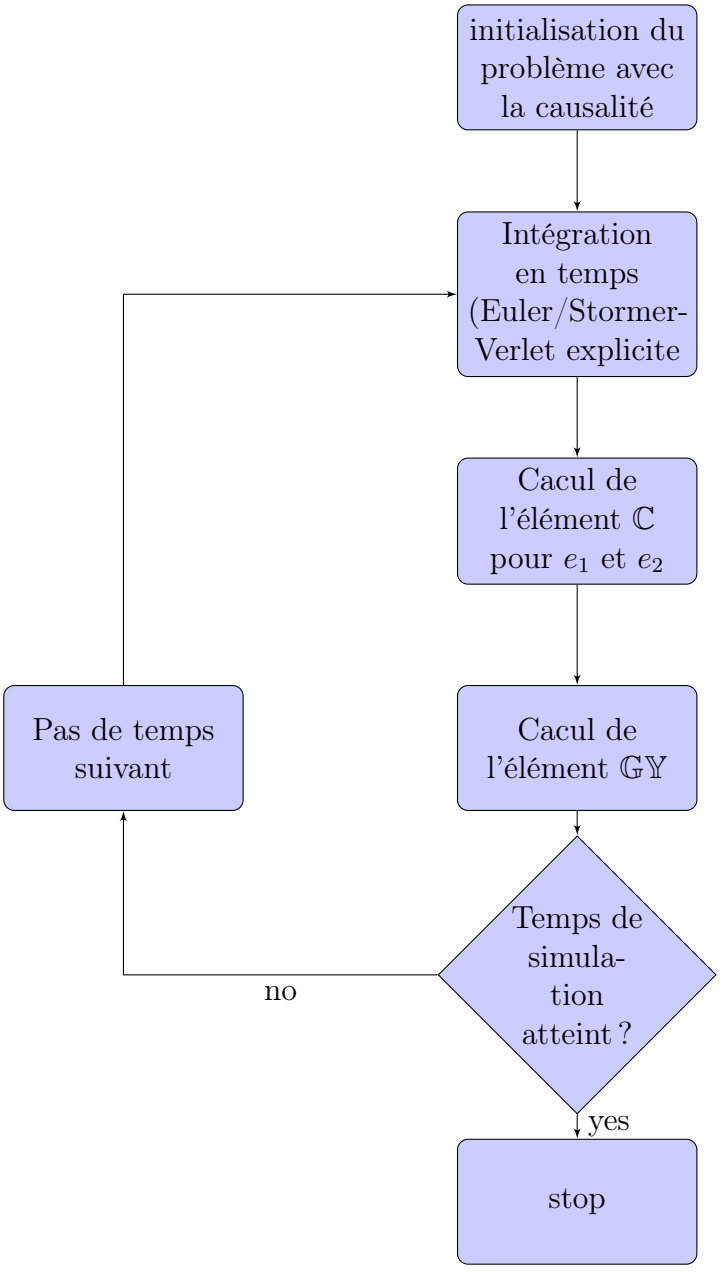


FIGURE 5.2 – Résolution classique du bondgraph faisant apparaître la causalité pour l'initialisation du problème et la séquentialité de la résolution des éléments bondgraphs

```
! Initialisation cohérente du premier vecteur pour
  le processus implicite

x(1:n)=1d0
x(n+1:2*n)=1d0
x(2*n+1:3*n)=matmul(mat_1,x(1:n))
x(3*n+1:4*n)=matmul(mat_2,x(n+1:2*n))

print *,'sundials conservation avant:',
dot_product(x(1:n),matmul(mat_1,x(1:n)))+
dot_product(x(n+1:2*n),matmul(mat_2,x(n+1:2*n)))
neq=24
!call time_integration(x,deltat,deltat,transline_sundial2)

call fida_solver(neq,x,xp) !Resolution par fida dans SUNDIALS

print *,'sundials conservation avant:',
dot_product(x(1:n),matmul(mat_1,x(1:n)))+
dot_product(x(n+1:2*n),matmul(mat_2,x(n+1:2*n)))
```

La fonction fida_solver appelle une fonction fidaresfun qui calcule $F(t, \dot{u}, u)$.

Pour la réalisation de cet exemple, il a fallu mettre en place une discrétisation polynomiale. Pour cela, on a créé les opérations de base (addition , multiplication, soustraction) et l'opérateur de dérivation pour les polynômes. L'intégration est faite par une formule de quadrature sur les points de Gauss(formule Gauss-Lobatto).

On peut vérifier que l'intégration du problème s'est bien déroulée en calculant l'énergie stockée dans la ligne définie par $x_1.mat_1x_1 + x_2.mat_2x_2$.
C'est une équation de transport hyperbolique, donc le calcul numérique doit préserver ce caractère hyperbolique qui conserve l'énergie le long de la ligne.

Processus explicite	Processus implicite par sundial
Etat initial : 1.97181343455643	Etat initial 1.97181343455643
Etat final : 1.97429905560194	Etat final : 1.97180122450567

Ces résultats sont également liés à la précision demandée au solveur fida de SUNDIALS.

5.2.3 Exemple avec la colonne d'adsorption

Afin d'illustrer un modèle parabolique dissipatif, considérons la réduction et la simulation du processus de séparation d'éléments chimiques par diffusion-adsorption discuté dans [5]. L'adsorption est un phénomène de surface par l'intermédiaire duquel des molécules à l'état gazeux ou liquide sont fixées sur un substrat solide par des liaisons covalentes. Ce phénomène joue un rôle majeur dans de nombreux processus industriels, notamment pour la séparation de molécules de différentes natures.
La démarche est basée sur le modèle de Stefan-Maxwell qui décrit l'adsorption comme une diffusion entre particules. Les forces thermodynamiques responsables de ce phénomène, et donc de l'adsorption, sont représentées par le potentiel chimique μ qu'on peut voir comme la variable conjuguée du flux de particules N. Ainsi on peut montrer que les équations régissant le système peuvent s'écrire :

$$N = -\frac{*qD}{RT} d\mu \tag{5.29}$$

où N est le flux de particules identifié comme une 2-forme, q est la concentration de particules vue comme une 3-forme et μ le potentiel chimique qui est une 0-forme. D, R et T étant respectivement le facteur de diffusion, la constante des gaz parfaits et la température.
Cette équation est augmentée d'une relation de conservation du nombre de particules total qui s'écrit

$$\dot{q} = -dN \tag{5.30}$$

où le point désigne la dérivée par rapport au temps.
Par ailleurs, le potentiel chimique est donné par l'expression suivante :

$$\mu = \mu^0(T, P) + RTln(\frac{*q}{P(k_1 - k_2 * q)}) \tag{5.31}$$

Maintenant il s'agit de reformuler ce modèle en utilisant l'approche des systèmes hamiltoniens à ports.
Considérons les espaces de flux $\mathscr{F}$ et d'efforts $\mathscr{E}$, avec les identifications suivantes :

$$\begin{aligned} f^q &= \dot{q} = dN \\ e^p &= -d\mu \\ e^q &= \mu \\ f^p &= N \end{aligned} \tag{5.32}$$

$$\begin{pmatrix} f_q \\ e_p \end{pmatrix} = \begin{pmatrix} 0 & d \\ d & 0 \end{pmatrix} \begin{pmatrix} f_p \\ e_q \end{pmatrix} \tag{5.33}$$

avec les variables de bords

$$\begin{pmatrix} f_\partial \\ e_\partial \end{pmatrix} = \begin{pmatrix} -e_{q\partial} \\ f_{p\partial} \end{pmatrix} \tag{5.34}$$

on peut reconnaître l'expression de la structure de Dirac définie en (5.18). L'élément bondgraph C, permet de calculer q à partir de f_q et de e_q à partir de q par l'équation thermodynamique (5.31). La dissipation est représentée par la relation R, qui permet le calcul de e_p à partir de f_p par l'équation (5.29).

Remarque 5.1. Dans ce modèle de diffusion-adsorption, le choix des variables d'efforts et de flux a été fait tel que le système exhibe une structure de Dirac (5.18) ayant une représentation analogue au modèle de la ligne de transmission. Dans une vision modulaire, ceci offre l'avantage d'avoir la même représentation de la structure mathématique pour différents modèles physiques. Cependant, du point de vue de la simulation, pour des raisons de causalité, la résolution du système dynamique pour ce modèle de diffusion-adsorption ainsi formulé, exclut une résolution séquentielle (déduire chaque variable à partir d'une autre), mais nécessite une résolution simultanée des relations constitutives en les considérant comme une contrainte algébrique. Cette résolution peut être faite à l'aide d'outils comme SUNDIALS.

La fonction $F(t, u, \dot{u})$, pour le module fida de SUNDIALS, est formée par les éléments $\mathbb{G}$,$\mathbb{C}$,$\mathbb{R}$, et s'écrit :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{q} \\ \dot{f}_q \\ \dot{f}_p \\ \dot{e}_q \\ \dot{e}_p \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & G_1 & 0 & 0 \\ 0 & 0 & 0 & G_2 & -1 \\ Cmat & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & R \end{pmatrix} \begin{pmatrix} q \\ f_q \\ f_p \\ e_q \\ e_p \end{pmatrix} = 0$$

La fonction implicite pour l'utilisation de la bibliothèque SUNDIALS est alors :

```
subroutine adsorp_sundials(y,yp,res)

call Celem_implicite(eq,q,fct_C,res)

call G1(eq,res)

call Relem_implicite(fp,ep,fct_R,res)

call G2(fp,res(1:n-1))
```

Les fonctions fct_R,fct_C sont les fonctions de discrétisation (ici non linéaire).

5.2.4 Conclusion

Notre travail a consisté à traduire le langage des bondgraphs dans le formalisme de la bibliothèque SUNDIALS afin de définir un plugin regroupant les éléments de base bondgraph. Par ailleurs l'écriture dans le formalisme de SUNDIALS permet, de part l'implicitation, de ne plus avoir de condition de stabilité dans le schéma d'intégration d'une part et d'autre part permet d'obtenir un schéma acausal. Le formalisme hamiltonien va nous permettre de connaître les conditions de raccords pour nos méthodes de décomposition de domaines. Cependant nous ne souhaitons pas effectuer la résolution de nos problèmes avec ce formalisme mais utiliser les techniques classiques. Nous illustrerons cette idée dans les deux prochains chapitres avec un exemple d'écoulement de Saint-Venant, que nous discrétiserons par un schéma spécifique en temps et en espace. Un dernier exemple traitant de la mise au point de condition de raccords pour l'équation de la chaleur non-linéaire sera développé.

Chapitre 6

Équation de Saint-venant

Table des matières

Notre but est de faire de la décomposition de domaine en espace qui à l'interface conserve les propriétés de l'approche hamiltonienne à ports.

Les écoulements gravitaires à surface libre se produisent sous l'effet de la gravité, le fluide étant en contact partiel avec le contenant. On retrouve ces écoulements à surface libre dans les rivières et les fleuves mais aussi dans les canaux construits par l'homme pour des besoins d'irrigation, de drainage.

6.1 Equations de Saint-Venant : cas uni-dimensionnel (1D)

Les équations de Saint-Venant décrivent la dynamique des tirants et débits d'eau dans un écoulement à surface libre. Elles ont été établies en 1871 par Adhémar Barré de Saint Venant [73]. Elles sont aujourd'hui un modèle de référence pour décrire l'écoulement dans un canal à surface libre, dans une rivière naturelle [48]. Ces équations sont obtenues pour des écoulements

dits filaires d'un fluide incompressible ; la vitesse est supposée uniforme sur la section perpendiculaire à la direction de l'écoulement, la répartition de la pression est hydrostatique, les niveaux d'eau sont négligeables devant la longueur total du canal. Les forces internes de viscosité sont négligées par rapport aux forces de frottement du fluide avec le fond et les berges du canal.

Nous considérons un canal rectangulaire à surface libre de pente I, de longueur L et de largeur B comme celui présenté dans la figure (6.1). Ce canal est délimité par une vanne en amont et une autre en aval. Les équations de Saint-Venant peuvent être écrites soit en débits ($Q(x,t)$) et hauteurs d'eau ($h(x,t)$) soit en hauteurs d'eau ($h(x,t)$) et vitesse d'écoulements ($v(x,t)$). Les deux formulations sont données ci-après :

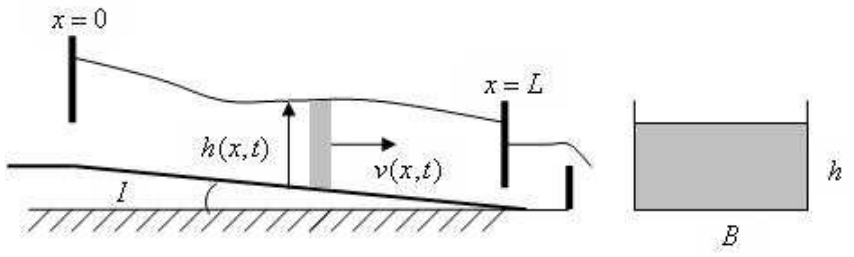


FIGURE 6.1 – Vue longitudinale (à gauche) et latérale (à droite) d'un canal à surface libre rectangulaire

Formulation en débits $\mathbf{Q(x,t)}$ et hauteurs d'eau $\mathbf{h(x,t)}$

$$\frac{\partial}{\partial t}\begin{pmatrix} h(x,t) \\ Q(x,t) \end{pmatrix} = -\frac{\partial}{\partial x}\begin{pmatrix} \frac{Q}{B} \\ \frac{Q^2}{Bh} + \frac{1}{2}gBh^2 \end{pmatrix} + \begin{pmatrix} 0 \\ gBh(I - J(Q,h)) \end{pmatrix} \quad (6.1)$$

La variable x représente l'espace, t est la variable du temps, $h(x,t)$ la hauteur du fluide, $Q(x,t)$ le débit volumique, et g l'accélération de la gravité. $J(Q,h)$ est le terme traduisant les forces de frottement du fluide avec le fond et les berges. On considère le modèle dit de Manning-Strickler donné par :

$$J(Q,h) = \frac{Q|Q|}{K^2(Bh)^2(\frac{Bh}{2h+B})^{\frac{4}{3}}} \quad (6.2)$$

où K est le paramètre de Manning-Strickler[49] mesurant la rugosité du fond et des berges du canal.

Formulation en vitesses d'écoulement $\mathbf{v(x,t)}$ et hauteurs de fluide $\mathbf{h(x,t)}$ La formulation en vitesses d'écoulement $v(x,t)$ et hauteurs d'eau $h(x,t)$ peut être déduite de celle en débits et hauteurs en tenant compte de l'expression du débit en fonction de la vitesse d'écoulement et de la hauteur :

$$Q(x,t) = Bh(x,t)v(x,t) \tag{6.3}$$

En injectant cette relation dans (6.1) on obtient les équations :

$$\frac{\partial}{\partial t}\begin{pmatrix} h(x,t) \\ v(x,t) \end{pmatrix} = -\frac{\partial}{\partial x}\begin{pmatrix} hv \\ \frac{1}{2}v^2 + gh \end{pmatrix} + \begin{pmatrix} 0 \\ g(I - J(v,h)) \end{pmatrix} \tag{6.4}$$

Équilibre stationnaire des équations de Saint Venant Les équations de Saint-Venant (6.1) sont à l'équilibre lorsque la hauteur $h(x,t)$ et le débit $Q(x,t)$ ne dépendent plus du temps t. Les profils d'équilibre satisfont les conditions :

$$\begin{cases} \dfrac{\partial Q(x,t)}{\partial x} = 0 \\ -\dfrac{\partial}{\partial x}(\dfrac{Q^2}{Bh} + \dfrac{1}{2}gBh^2) + gBh(I - J(Q,h)) = 0 \end{cases} \tag{6.5}$$

A l'équilibre le profil de débit est uniforme le long du canal (il sera noté Q_e) et, le profil de hauteur $h_e(x)$ est la solution d'une équation différentielle du premier ordre qui s'écrit :

$$\left[\frac{Q_e^2}{Bh_e^2} - gBh_e\right]\frac{dh_e}{dx} + gBh_e(I - J(Q_e,h_e)) = 0 \tag{6.6}$$

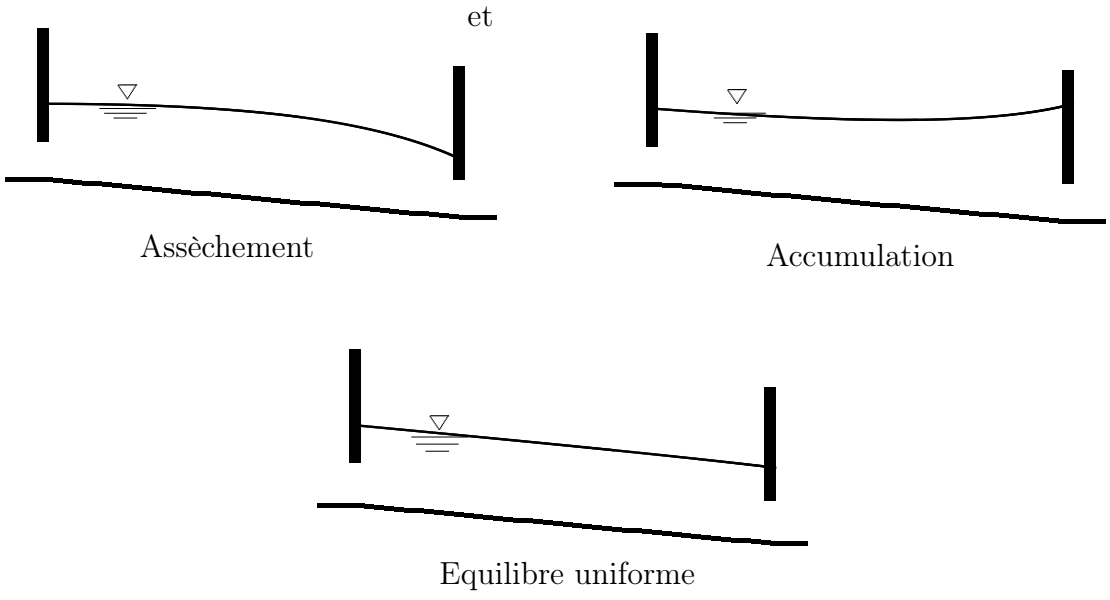
Cette écriture fait apparaître le nombre de Froude Fr défini par :

$$Fr^2 = (\frac{Q_e}{Bh_e})^2(\frac{1}{gh_e}) = \frac{v_e^2}{gh_e} \tag{6.7}$$

Ce nombre positif détermine le régime de l'écoulement en x. Il est défini par le rapport entre la vitesse d'écoulement v et la vitesse des ondes de surface $\sqrt{gh_e}$. Si $Fr < 1$ l'écoulement est dit *fluvial*. Si $Fr > 1$ l'écoulement est dit *torrentiel*. Si $Fr = 1$ (la vitesse d'écoulement et la vitesse des ondes sont égales) l'écoulement est dit *critique*. Si $Fr \neq 1$ on a :

$$\frac{dh_e}{dx}(x) = \frac{I - J(Q_e, h_e(x))}{1 - Fr^2} \tag{6.8}$$

Le profil d'équilibre peut donc être paramétré par le débit à l'équilibre Q_e et la hauteur en un point particulier du canal (généralement une des deux



f

FIGURE 6.2 – Les différents profils d'équilibre des équations de Saint-Venant

extrémités). Dans le cas fluvial on peut avoir trois types de profils (*courbes de remous*) : assèchement, accumulation, équilibre uniforme (Fig.6.2)

Le profil uniforme est obtenu pour une valeur particulière de Q_e où la force de gravité est parfaitement compensée par les forces de frottement. Cette condition s'exprime :

$$J(Q_e, h_e) = I \tag{6.9}$$

Vanne hydraulique Les vannes sont utilisées pour réguler le débit du canal et assurer sont maintien en eau à un niveau suffisant. Le débit à travers une vanne noyée est généralement modélisé par l'équation :

$$Q = \alpha B \theta(t) \sqrt{2g(h_1 - h_2)} \tag{6.10}$$

où α est le coefficient de débit de la vanne (on l'appelle aussi coefficient de contraction entre la surface d'ouverture de la vanne et la surface située en aval où les lignes de courants deviennent parallèles, souvent, $0.6 < \alpha < 0.7$), B est la largeur de la vanne, h_1 est le tirant d'eau amont, h_2 est le tirant d'eau aval, $\theta(t)$ est l'ouverture de la vanne en fonction du temps et g est l'accélération de la pesanteur (voir Fig.6.3).

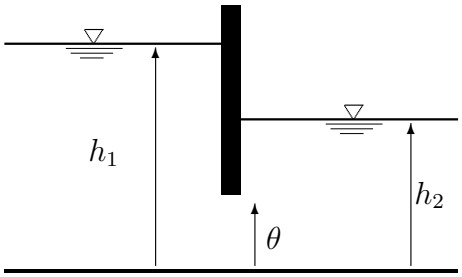


FIGURE 6.3 – Vanne noyée

6.1.1 Formulation hamiltonienne à ports des équations de Saint-Venant : cas uni-dimensionnel (1D)

Les équations de Saint-venant sont des équations de conservation de la quantité de matière et de la quantité de mouvement. Ces deux quantités sont extensives et définies dans tout le domaine spatial unidimensionnel considéré. Il est donc naturel de choisir comme variables d'énergie le volume infinitésimal et la densité linéique de quantité de mouvement. Ces deux variables sont (dans le cas considéré du canal de section rectangulaire) les 1-formes différentielles :

$$q(x,t) = Bh(x,t)dx \qquad \in \Omega^1(Z = [0, L]) \tag{6.11}$$

$$p(x,t) = \rho v(x,t)dx \qquad \in \Omega^1(Z = [0, L]) \tag{6.12}$$

La densité linéique d'énergie du fluide en écoulement est calculée comme étant la somme de la densité d'énergie potentielle et de la densité d'énergie cinétique d'un volume élémentaire :

$$\mathcal{H}_p(x,t) = \left(\int_{z_f}^{z_f+h} \rho g B \xi d\xi \right) dx \tag{6.13}$$

$$\mathcal{H}_c(x,t) = \frac{1}{2} \rho B h(x,t) v^2(x,t) dx \tag{6.14}$$

où z_f est la hauteur absolue du fond du canal. L'énergie totale est l'intégrale de la densité linéique d'énergie le long du domaine spatial Z :

$$\begin{aligned} H(t) &= \int_0^L \mathcal{H}(x,t) \\ &= \int_0^L \mathcal{H}_p(x,t) + \mathcal{H}_c(x,t) \\ &= \frac{1}{2}\int_0^L (\rho g B h^2 - 2\rho B I h g x + \rho B h v^2) dx \end{aligned} \tag{6.15}$$

De la fonction d'énergie on calcule les variables d'efforts (appelées aussi variables de co-énergie) comme dérivées variationnelles de cette fonction d'énergie.

$$\begin{aligned} e_q(x,t) &:= \delta_q H = \frac{1}{2}\rho v^2(x,t) + \rho g(h(x,t) - Ix) \\ e_p(x,t) &:= \delta_p H = Bh(x,t)v(x,t) \end{aligned} \tag{6.16}$$

Ces variables sont des 0-formes différentielles (fonctions) sur le domaine Z. Le premier effort ($e_q(x,t)$) est généralement appelé pression hydrodynamique et le deuxième ($e_p(x,t)$) débit volumique. Ces deux variables sont les variables des puissances (variables duales) car leur produit est homogène à une puissance.

Ajout de la dissipation par frottements Dans les équations de Saint-Venant les forces de frottement sont modélisées par la formule de Manning-Strickler. Elles peuvent être modélisées comme de la dissipation de la quantité de mouvement par un flux dissipatif noté f_d :

$$f_d = \rho g J(Q,h) dx \qquad \in \Omega^1(Z) \tag{6.17}$$

Ce flux de quantité de mouvement dissipé peut être réécrit par la relation constitutive suivante :

$$f_d = G_d(q,p) \wedge e_p(x,t) \tag{6.18}$$

où $G_d(q,p) \in \Omega^1(Z)$ est une 1-forme différentielle représentant la caractéristique distribuée de la dissipation et peut être assimilée à une conductance linéique. Elle est donnée explicitement par l'équation :

$$G_d(q,p) = \frac{\rho g |Q|}{K^2 (Bh)^2 (\frac{Bh}{2h+B})^{\frac{4}{3}}} dx \tag{6.19}$$

Cette conductance $G_d(q,p)$ satisfait la condition de dissipativité :

$$\int_Z e_p \wedge (G_d(q,p) \wedge e_p(x,t)) \geq 0 \quad \forall e_p \in \Omega^0(\mathcal{Z}) \tag{6.20}$$

En écrivant le bilan du flux de la quantité de mouvement on obtient :

$$f_p(x,t) + f_d(x,t) + \frac{\partial q}{\partial t}(x,t) = 0 \tag{6.21}$$

Compte tenu des équations de la structure de couplage, ce bilan peut s'écrire :

$$-\frac{\partial q}{\partial t}(x,t) = d(e_p(x,t)) + f_d(x,t) \tag{6.22}$$

On obtient finalement la formulation hamiltonienne à ports des équations de Saint-Venant avec dissipation :

$$\begin{bmatrix} -\frac{\partial q}{\partial t} \\ -\frac{\partial p}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 & d \\ d & 0 \end{bmatrix} \begin{bmatrix} \delta_q H \\ \delta_p H \end{bmatrix} + \begin{bmatrix} 0 \\ G_d \wedge \delta_p H \end{bmatrix} \tag{6.23}$$

$$\begin{bmatrix} e^0_\partial(t) \\ e^L_\partial(t) \\ f^0_\partial(t) \\ f^L_\partial(t) \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_q H_{|x=0} \\ \delta_q H_{|x=L} \\ \delta_p H_{|x=0} \\ \delta_p H_{|x=L} \end{bmatrix} \tag{6.24}$$

Les equations (6.23,6.24) définissent la formulation hamiltonienne à ports des équations de Saint-Venant avec pente et forces de frottements. Elle est équivalente à leurs formulations classiques dans (6.1,6.4). Le bilan de la puissance instantanée est, dans ce cas donné par :

$$\frac{dH}{dt} = \int_{\partial\mathcal{Z}} e_\partial \wedge f_\partial - \int_{\mathcal{Z}} e_p \wedge f_d \tag{6.25}$$

$$= \int_{\partial\mathcal{Z}} e_\partial \wedge f_\partial - \int_{\mathcal{Z}} e_p \wedge (G_d(q,p) \wedge e_p(x,t)) \tag{6.26}$$

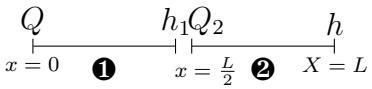
La variation de l'énergie du fluide est égale à la puissance fournie via les frontières du domaine à laquelle on retranche la puissance dissipée par frottements.

6.2 Application de la décomposition de domaine à Saint-Venant

Notre objectif est de découper le canal en deux afin d'appliquer les techniques de décomposition de domaine pour résoudre le problème sur chaque

sous-canal indépendamment. Le problème de l'écoulement sur le domaine entier est donné avec un débit et une hauteur initiale à l'entrée du canal. Pour la décomposition de domaine, il est nécessaire d'avoir des conditions de transmission entre domaines. Donc il faut transformer le problème aux valeurs initiales , en un problème aux valeurs frontières. On décide d'imposer le débit en entrée du canal et la hauteur en sortie. Tout d'abord on vérifie alors que l'on obtient bien la même solution à l'équilibre en résolvant le problème aux valeurs initiales et le problème aux valeurs frontières.

Découpage du canal en espace :



6.2.1 Discrétisation par un schéma de Preissman

Le schéma de Preissman [Preissmann and cunge, 1961 ; Liggett and Cunge, 1975 ; Cunge,Holly and Verwey, 1980] est un schéma différences finies semi-implicite. Ce schéma, très utilisé dans le domaine hydraulique depuis les années 1960, a la propriété d'être stable inconditionnellement en temps. À chaque itération, il faut alors résoudre un système d'équation algébrique non linéaire.

La discrétisation en temps et en espace se fait par un pas de temps Δt et un pas spatial Δx. On adopte la notation $f(x_i, t_j) = f_i^j$.

Le schéma s'écrit alors :

$$f(x,t) = \theta(\frac{f_{i+1}^{j+1} + f_i^{j+1}}{2}) + (1-\theta)(\frac{f_{i+1}^{j} + f_i^{j}}{2}) \tag{6.27}$$

$$\frac{\partial f}{\partial x}(x,t) = \frac{\theta}{\Delta x}(\frac{f_{i+1}^{j+1} - f_i^{j+1}}{2}) + (1-\theta)(\frac{f_{i+1}^{j} - f_i^{j}}{2}) \tag{6.28}$$

$$\frac{\partial f}{\partial t}(x,t) = \frac{\Phi}{\Delta t}(\frac{f_{i+1}^{j+1} - f_{i+1}^{j}}{2}) + (1-\Phi)(\frac{f_i^{j+1} - f_i^{j}}{2}) \tag{6.29}$$

Le paramètre Φ (respectivement θ) est un coefficient pour régler l'implicitation du schéma en espace (respectivement en temps) compris entre 0 et 1. Le schéma est inconditionnellement stable pour $\Phi > 0.5$, c'est-à-dire qu'il n'y a aucune restriction sur le pas de temps Δt et le pas d'espace Δx.

En appliquant le schéma de Preissman aux équations de Saint-Venant en débit/hauteur, on obtient :

Équation de continuité :

$$\frac{B}{2\Delta t}[(h_{i+1}^{j+1}-h_{i+1}^{j})+(h_i^{j+1}-h_i^{j})]+\frac{1}{\Delta x}[\theta(Q_{i+1}^{j+1}-Q_i^{j+1})+(1-\theta)(Q_{i+1}^{j}-Q_i^{j})]=0 \tag{6.30}$$

Équation dynamique :

$$\begin{aligned}
&\frac{1}{2\Delta t}[(Q_{i+1}^{j+1}-Q_{i+1}^{j})+(Q_i^{j+1}-Q_i^{j})]\\
&+\frac{1}{B}[\frac{\theta}{\Delta x}((Q^2/h)_{i+1}^{j+1}-(Q^2/h)_i^{j+1})+\frac{1-\theta}{\Delta x}((Q^2/h)_{i+1}^{j+1}-(Q^2/h)_i^{j+1})]\\
&+\frac{1}{2}gB[\theta(h_{i+1}^{j+1}-h_i^{j+1})+(1-\theta)(h_{i+1}^{j}-h_i^{j})]\\
&\qquad(\frac{1}{\Delta x}[\theta(h_{i+1}^{j+1}-h_i^{j+1})+(1-\theta)(h_{i+1}^{j}-h_i^{j})]-I)\\
&+\frac{1}{2}gB[\theta((hJ)_{i+1}^{j+1}-(hJ)_i^{j+1})+(1-\theta)((hJ)_{i+1}^{j}-(hJ)_i^{j})]=0
\end{aligned} \tag{6.31}$$

Afin de fermer le système, on rajoute deux équations de bords. La discrétisation des conditions de bords est faite par des conditions de Dirichlet, en amont et en aval du canal.

$$Q_1^{j+1}-valeur_1^{j+1}=0 \tag{6.32}$$
$$h_N^{j+1}-valeur_N^{j+1}=0 \tag{6.33}$$

La résolution de ce système de deux équations non-linéaires se fait par une méthode de Newton classique. La solution au temps j étant la condition initiale pour obtenir la solution au temps $j+1$.

6.2.2 Application de l'algorithme de Schwarz pour la décomposition de domaine

On se propose de mettre en place un algorithme de Schwarz sans recouvrement pour la résolution du problème de Saint-venant (6.1) sur un canal décomposé.

Les conditions à satisfaire à l'interface sont :

$$\begin{cases} h_1 = h_2(\frac{L}{2}) \\ Q_2 = Q_1(\frac{L}{2}) \end{cases} \tag{6.34}$$

L'algorithme de Schwarz multiplicatif sans recouvrement s'écrit

$$\begin{cases} \mathcal{J}(Q_1, h_1) = 0 \quad dans \quad [0, L/2] \\ Q_1^{2n+1} = 0 \quad pour \quad x = 0 \\ h_1^{2n+1} = h_2^n \quad pour \quad x = \dfrac{L}{2} \end{cases} \tag{6.35}$$

$$\begin{cases} \mathcal{J}(Q_2, h_2) = 0 \quad dans \quad [L/2, L] \\ h_2^{2n+2} = 0 \quad pour \quad x = L \\ Q_2^{2n+2} = Q_1^{2n+1} \quad pour \quad x = \dfrac{L}{2} \end{cases} \tag{6.36}$$

Algorithm 2 Algorithme de Schwarz multiplicatif

1: **while** $||Q_1^{2n+1} - Q_1^{2n-1}||_2 \geq eps$ **do**
2: Résoudre le problème ❶ à l'itération $2n+3$
3: Échange de la condition de Dirichlet Q_1^{2n+3}
4: Résoudre le problème ❷ à l'itération $2n+4$
5: Échange de la condition de bord de Dirichlet h_2^{2n+4}
6: **end while**

Nous avons effectué des tests avec la manière la plus naturelle de faire l'échange de Schwarz, c'est à dire en échangeant les variables Q et h entre les sous-domaines, comme décrit par les équations (6.36) de l'algorithme précédent. À chaque pas de temps, l'algorithme de Schwarz en espace doit avoir convergé pour continuer le processus d'intégration en temps. Nous nous servons de la solution convergée du pas de temps précédent pour l'initialisation de chaque domaine au temps courant t.

Notre démarche est de montrer l'intérêt de nouvelles conditions d'échanges pour l'algorithme de Schwarz. En effet la continuité des quantités triviales, comme la solution ou la dérivée, n'est peut être pas la plus adaptée pour le problème, en particulier, pour des problèmes non-linéaires. Dans notre cas, il faudrait que les conditions d'échanges prennent en compte la gestion des sauts dans la solution qui peuvent intervenir dans l'équation de Saint-Venant. La condition de raccord des domaines correspond en formulation hamiltonienne à ports à l'égalité des variables d'efforts à la frontière entre sous-domaines. Dans la formulation hamiltonienne développée dans la section précédente, on a les variables d'efforts (6.16), $\delta_q H$ et $\delta_p H$. Cette condition d'égalité s'écrit

$$\begin{cases} e_{q,1} = e_{q,2} \\ e_{p,1} = e_{p,2} \end{cases} \tag{6.37}$$

Cette condition d'égalité n'est pas équivalente à la continuité de la hauteur h et des débits Q. Avec la définition des variables $e_q(x,t), e_p(x,t)$ (6.16), on obtient en $x = \frac{L}{2}$

$$\begin{aligned} \frac{1}{2}\rho v_1^2(x,t) + \rho g(h_1(x,t) - Ix) &= \frac{1}{2}\rho v_2^2(x,t) + \rho g(h_2(x,t) - Ix) \quad (6.38) \\ Bh_1(x,t)v_1(x,t) &= Bh_2(x,t)v_2(x,t) \quad (6.39) \end{aligned}$$

Par définition, on sait que $v = \frac{Q}{h}$ donc la deuxième équation devient

$$Q_1 = Q_2 \quad (6.40)$$

$$\begin{aligned} \frac{1}{2}\rho \frac{{Q_1}^2}{h_1}(x,t) + \rho g(h_1(x,t) - I_1 x) &= \frac{1}{2}\rho \frac{{Q_2}^2}{h_2}(x,t) + \rho g(h_2(x,t) - I_2 x) \\ &= \gamma \quad (6.41) \end{aligned}$$

$$\frac{1}{2}\rho Q_1^2(x,t) + \rho g(h_1^3(x,t) - I_1 x h_1^2) - h_1^2\gamma = 0 \quad (6.42)$$

$$\rho g h_1^3 - h_1^2\gamma + \frac{1}{2}\rho Q_1^2(x,t) = 0 \quad (6.43)$$

La solution triviale est $h_1 = h_2$, on peut alors factoriser l'équation

$$(h_1 - h_2)(\alpha h_1^2 + \beta h_1 + \delta) = 0 \quad (6.44)$$

En identifiant les coefficients du polynôme de degré 2, on obtient

$$\alpha h_1^3 + h_1^2(\beta - \alpha h_2^2) + h_1(\delta - \beta h_2) - \delta h_2 = 0 \quad (6.45)$$

$$\begin{cases} \alpha = \rho g \\ \delta - \beta h_2 = 0 \\ \beta - \alpha h_2^2 = -e_2 \\ \delta = -\frac{1}{2}\rho \frac{Q_1^2}{h_2} \end{cases} \quad \begin{cases} \beta = -\gamma + \alpha h_2^2 = -\frac{1}{2}\rho \frac{Q_2^2}{h_2^2} \\ \delta = -h_2\gamma + \alpha h_2^2 = -\frac{1}{2}\rho \frac{Q_2^2}{h_2} \end{cases} \quad (6.46)$$

$\Delta = \beta^2 - 4\alpha\delta = (-\frac{1}{2}\rho\frac{Q_2^2}{h2})^2 + 4\rho g\frac{1}{2}\rho\frac{Q_2^2}{h2} > 0$. On trouve alors deux autres solutions :

$$h_1 = \frac{2Q_2(\frac{Q_2^2}{4} + 2gh_2^3)^{1/2} + Q_2^2}{4gh_2^2} \quad (6.47)$$

$$h_1 = \frac{-(2Q_2(\frac{Q_2^2}{4} + 2gh_2^3)^{1/2} - Q_2^2)}{4gh_2^2} \quad (6.48)$$

Il est évident que la solution négative n'est pas admissible puisque la quantité calculée est une hauteur. Nous avons testé l'échange aux frontières en utilisant la formule (6.47) sans succès. La méthode de Schwarz ne converge pas vers la solution du problème pour un temps t donné. L'utilisation des variables fournies par le formalisme hamiltonien à ports aboutit uniquement sur l'échange trivial des quantités Q et h.

6.2.3 Utilisation de la méthode de Newton pour satisfaire les contraintes

Nous nous proposons de résoudre le système non-linéaire (6.37), formé par l'égalité des variables de ports à l'interface, par une méthode de Newton. Les variables inconnues sont h_1 et Q_2 conditions de bords des sous-domaines. Les variables h_2 et Q_1 étant données par la résolution locale de chaque sous-domaine. La fonction non-linéaire à minimiser s'écrit :

$$F(h_1, v_2) = \begin{cases} h_1 v_1 - h_2 v_2 = 0 \\ 0.5(v_1^2 + g h_1) - 0.5(v_2^2 + g h_2) = 0 \end{cases} \tag{6.49}$$

La matrice jacobienne J du système de contrainte (6.49) à l'interface s'écrit :

$$J = \begin{pmatrix} -(v_2 + v_1) & (h_1 + h_2) \\ -2g & (v_2 + v_1) \end{pmatrix} \tag{6.50}$$

En posant $x = \begin{pmatrix} h \\ v \end{pmatrix}$, la méthode de Newton s'écrit :

$$\begin{cases} J\delta^k = -F(x^k) \\ x^{k+1} = x^k + \delta^k \end{cases} \tag{6.51}$$

Des variantes plus adaptées de cette méthode peuvent être utilisées [35]. Tous nos tests numériques ont été effectués sous Matlab ; nous avons donc résolu le système non-linéaire (6.49) par la méthode fsolve fournie par Matlab. A chacun de nos tests sur le canal découpé, la méthode de Newton qui permet de trouver les quantités à échanger en respectant la contrainte, donne toujours la solution triviale $h_1 = h_2$ et $Q_1 = Q_2$ du système non-linéaire .

Nous avons tenté de réaliser un choc, c'est à dire de se mettre dans une situation de propagation d'une vague dans le canal avec changement de régime , fluvial vers torrentiel. Les résultats obtenus n'ont pas été concluants, nous nous sommes donc intéressés à un problème différent que nous développerons dans le chapitre suivant.

Chapitre 7

Équation de la chaleur non-linéaire

Table des matières

Les problèmes rencontrés avec l'équation de Saint-Venant ne permettent pas d'obtenir un cas test qui montre l'intérêt des conditions de transmissions non-linéaires pour la décomposition de domaine de type Schwarz définies par le formalisme à ports.

Dans [16] J.Koko et D.Bresch présentent une condition de transmission non-linéaire à l'interface de deux écoulements de Navier-Stokes de viscosités différentes. La condition de transmission représente la couche de mélange des écoulements qui donne la tension de surface de chaque écoulement comme une fonction non-linéaire du saut des solutions à l'interface. Dans [59, 58], C.-H Lai résout l'équation de la chaleur non-linéaire entre deux sous domaines et propose une équation non-linéaire de saut à l'interface des domaines. Il propose une méthode de type point fixe pour résoudre cette équation de saut non-linéaire. Dans [77], il fait état d'un comportement de la convergence dépendant de l'équation de saut, et de plus que cette convergence

n'est pas garantie. Citons par ailleurs les travaux de Periaux et Mantel [54] qui proposent d'améliorer la convergence des problèmes à l'interface par des techniques basées sur des algorithmes génétiques.

Dans ce chapitre, nous étudierons une décomposition de domaine sur l'équation de la chaleur non-linéaire avec un coefficient de diffusion $K(T)$ qui est une fonction non-linéaire de la solution recherchée T. Ce problème est abordé par Kornhuber et al. dans [13, 12],où ils effectuent un changement de variable $\nabla u_i = K(T)\nabla T$, $u_i = \kappa_i(T_i)$ dans chaque sous-domaine pour transformer le problème local en un problème linéaire. Par contre la non-linéarité est reportée sur la condition de raccord qui devient $\kappa_1^{-1}(u_1) = \kappa_2^{-1}(u_2)$. La limitation de cette approche est la nécessité de connaître l'inverse du changement de variable "analytiquement" pour ce qui n'est pas toujours possible suivant la fonction nonlinéaire $K(T)$. L'algorithme proposé s'apparente à celui de type Schwarz avec des conditions de transmissions de Dirichlet-Neumann où la condition de Dirichlet est donnée par $u_1 = \kappa_1(\kappa_2^{-1}(u_2))$.

Notre démarche est la suivante : dans la section 7.1 nous montrons que d'imposer des conditions de raccord de type Dirichlet-Neumann sur des considérations purement algorithmiques ne fonctionne pas . Nous illustrons la non-convergence de cet algorithme sur un cas de problème non linéaire. Puis nous proposons notre approche qui consiste à nous servir du formalisme hamiltonien à ports pour déterminer les conditions de transmission adéquates pour effectuer une décomposition de domaine de type Schwarz en espace dans la section 7.2. Enfin, pour des considérations d'efficacité algorithmique, nous introduisons un multiplicateur de Lagrange afin d'imposer la contrainte d'égalité des solutions à l'interface dans la section 7.3.

7.1 Analyse de la non convergence des conditions de transmissions Neumann-Dirichlet

Nous considérons uniquement la forme stationnaire de l'équation de la chaleur non-linéaire par soucis de simplicité pour l'étude des conditions de transmission en espace.

$$\begin{cases} -div(-K(T)grad(T)) = 0 \\ T(0) = 0 \\ T(L) = 1 \end{cases} \tag{7.1}$$

Appliquons alors une décomposition de domaine du domaine

$[0, L] = [0, \Gamma] \cup [\Gamma, L]$ en deux sous-domaines et, appliquons un algorithme de résolution de type Neumann-Dirichlet :

$$\begin{cases} -div(-K(T_1^{n+1/2})grad(T_1^{n+1/2})) = 0 \\ T_1^{n+1/2}(0) = 0 \\ T_1^{n+1/2}(\Gamma) = \alpha^n = T_2^n(\Gamma) \end{cases} \tag{7.2}$$

$$\begin{cases} -div(-K(T_2^{n+1})grad(T_2^{n+1})) = 0 \\ \dfrac{\partial T_2^{n+1}}{\partial x}(\Gamma) = \beta^{n+1/2} = \dfrac{\partial T_1^{n+1/2}}{\partial x}(\Gamma) \\ T_2^{n+1}(L) = 1 \end{cases} \tag{7.3}$$

On souhaite déterminer la convergence ou la divergence de la condition de transmission $(\alpha^n, \beta^{n+1/2})$ à l'interface Γ. Comme dans les travaux de Quarteroni [72] pour la méthode de Neumann-Dirichlet sans relaxation dans le cas du Laplacien , la méthode de Neumann-Dirichlet stagne si la taille des deux domaines est identique. Dans nos tests on prendra $\Gamma = 3/5L$.

7.1.1 Test avec $K(T) = T^{-\frac{1}{2}}$

Prenons $K(T) = T^{-\frac{1}{2}}$, $L = 1$ et $\Gamma = 3/5$ à la solution analytique du problème non décomposé est alors $T(x) = x^2$. Nous allons nous intéresser au comportement de la solution sur chaque domaine en fonction de la condition de transmission α ou β. Les problèmes deviennent alors :

$$\begin{cases} -div(-K(T_1^{n+1/2})grad(T_1^{n+1/2})) = 0 \\ T_1^{n+1/2}(0) = 0 \\ T_1^{n+1/2}(\dfrac{3}{5}) = \alpha^n \end{cases} \tag{7.4}$$

$$\begin{cases} -div(-K(T_2^{n+1})grad(T_2^{n+1})) = 0 \\ \dfrac{\partial T_2^{n+1}}{\partial x}(\dfrac{3}{5}) = \beta^{n+1} \\ T_2^{n+1}(1) = 1 \end{cases} \tag{7.5}$$

La solution analytique pour le domaine 1 est :

$$T_1^{n+1/2}(x) = \frac{25}{9}x^2\alpha^n \tag{7.6}$$

On veut connaître le comportement de la condition à l'interface Dirichlet, c'est-à-dire obtenir α^n en fonction de α^{n-1} où n représente l'itération de l'algorithme Neumann-Dirichlet.

La dérivée analytique de la solution dans le domaine 1 est

$$T_1'^{n+1/2}(x) = \frac{50}{9} x\alpha^n \tag{7.7}$$

On trouve donc la valeur de la dérivée en $x = \frac{3}{5}$, $T_1'^{n+1/2}(\frac{3}{5}) = \frac{10}{3}\alpha^n$.

La solution analytique du domaine deux en fonction de la condition de Dirichlet α^n donne deux possibilités :

$$T_2^{n+1}(x) = \begin{cases} \frac{25}{4} r_1^2 x^2 + \frac{5}{2} r_1 x(-5r_1 - 2) + \frac{1}{4}(-5r_1 - 2)^2 \\ \frac{25}{4} r r_2^2 x^2 + \frac{5}{2} r r_2 x(-5 r r_2 + 2) + \frac{1}{4}(-5 r r_2 + 2)^2 \end{cases} \tag{7.8}$$

où r_1 est la première racine du polynôme $3Z^2+3Z+2\alpha$ et rr_2 est la deuxième racine du polynôme $3Z^2-3Z+2\alpha$. En regardant $T_2^{n+1}(\frac{3}{5})$, on obtient la suite (α^n) pour les deux solutions,

$$\alpha^{n+1} = f(\alpha^n) \begin{cases} 1/2 + (1/6)\sqrt{9 - 24\alpha^n} - (2/3)\alpha^n \\ 1/2 - (1/6)\sqrt{9 - 24\alpha^n} - (2/3)\alpha^n \end{cases} \tag{7.9}$$

L'analyse de ces deux suites montre qu'aucune ne permet de converger vers la solution $T(\frac{3}{5}) = \frac{9}{25}$.
Le point fixe pour la suite $1/2 - (1/6)\sqrt{9 - 24\alpha^n} - (2/3)\alpha^n$ est 0, cette possibilité même en cas de convergence ne mènera pas vers la solution attendue.

La suite $1/2 - (1/6)\sqrt{9 - 24\alpha^n} - (2/3)\alpha^n$ peut encore se réécrire $1/4 - \sqrt{1/4 - 2/3 * \alpha^n} + 1/4 - (2/3)\alpha^n$. Pour que cette suite converge vers le point fixe $\alpha_r = \frac{9}{25}$, il est nécessaire que $|f'(\alpha)| < 1$ au voisinage de la racine α_r. D'autre part, on sait que α n'est défini que sur $[0, 3/8]$ qui contient la racine α_r. L'étude de la fonction montre que $|f'(\alpha)| \geq 1$ sur l'intervalle $[0, 3/8]$. On en déduit que la méthode du point fixe ne convergera pas, et que l'algorithme de Neumann-Dirichlet ne convergera pas vers la solution du problème. On note également que l'intervalle de définition est très restreint pour la condition α^n.

On conclut qu'une condition de Neumann arbitraire permet d'itérer l'algorithme mais ne permet pas de le faire converger vers la solution du problème non décomposé.

7.2 Conditions de transmissions déduites du Formalisme Hamiltonien à port

Dans le cadre du formalisme hamiltonien à ports, l'équation de chaleur non-linéaire en 1D s'écrit :

$$\begin{cases} \dfrac{\partial u}{\partial t} = -div(J_Q) \\ J_Q = -K(T,x)\dfrac{\partial T(t,x)}{\partial x} \\ T(t,0) = 0 \\ T(t,L) = 1 \\ T(0,x) = T_0 \end{cases} \tag{7.10}$$

u étant l'énergie interne et est reliée à la variable température par la relation $du = cv(T)dT$.

En reprenant le formalisme du chapitre 5 pour ce système (7.10), on choisit la variable d'état comme étant l'énergie interne u définissant la fonction d'entropie $s(u)$ et qui permet d'obtenir la variable conjuguée $T = \dfrac{du}{ds}$. La structure de Stokes-Dirac associée à ce système s'écrit :

$$\begin{pmatrix} \dfrac{\partial u}{\partial t} \\ -F' \end{pmatrix} = \begin{pmatrix} 0 & -\dfrac{\partial}{\partial x} \\ -\dfrac{\partial}{\partial x} & 0 \end{pmatrix} \begin{pmatrix} 1/T \\ J_Q \end{pmatrix} \tag{7.11}$$

Les variables de bords associées à la structure de Stokes-Dirac sur un intervalle [a,b], s'écrivent aux bords a et b :

$$\begin{pmatrix} f_\delta \\ e_\delta \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \dfrac{ds}{du} \\ J_Q \end{pmatrix} \tag{7.12}$$

qui sont la réciproque de la température $\dfrac{ds}{du} = \dfrac{1}{T}$ et le flux de chaleur J_Q aux frontières. Les relations (7.11)(7.12) définissent la structure de Stokes-Dirac qui peut être complétée par la loi des flux $J_Q = K(T,x)T^2F'$ et on obtient alors le système complet qui décrit la conduction de la chaleur.

L'idée est, comme au chapitre précédent, de trouver des conditions d'échanges non-linéaires à l'interface qui améliorent le comportement de l'erreur et la convergence de l'algorithme. Pour cela, nous nous sommes intéressés au formalisme hamiltonien à ports utilisés dans le domaine de

l'automatique. D'après le modèle hamiltonien de l'équation (7.11)(7.12), les variables de ports sont :

$$\begin{cases} e_1 = \delta_q H = \dfrac{1}{T} \\ e_2 = \delta_p H = J_Q \end{cases} \tag{7.13}$$

Le formalisme hamiltonien à ports préconise de prendre comme variable d'échange à l'interface l'inverse de la solution $\dfrac{1}{T}$, et J_Q . Nous ne prendrons pas l'inverse de la solution mais la solution T comme donnée d'échange et le flux $K(T,x)grad(T(x))$.

7.2.1 Test avec $K(T) = T^{-\frac{1}{2}}$

En reprenant la décomposition de domaine du cas précédent avec les conditions de transmissions du formalisme hamiltonien à ports, le problème de l'équation de la chaleur non linéaire dans le deuxième domaine s'écrit :

$$\begin{cases} -div(-K(T_2^{n+1})grad(T_2^{n+1})) = 0 \\ K(T_2^{n+1})\dfrac{\partial T_2^{n+1}}{\partial x}(\dfrac{3}{5}) = \beta^{n+1/2} \\ T_2^{n+1}(L) = 1 \end{cases} \tag{7.14}$$

Calculons analytiquement le comportement de la solution avec ces nouvelles variables pour l'échange de l'algorithme de type Neumann-Dirichlet.

$$T_2^{n+1}(x) = \begin{cases} \dfrac{25}{4}x^2 - \dfrac{15}{2}x + \dfrac{9}{4} & \text{(indépendant de } \alpha^n) \\ \dfrac{25}{9}\alpha^n x^2 + \dfrac{5}{3}x\sqrt{\alpha^n}(2 - \dfrac{10}{3}\sqrt{\alpha^n}) + \dfrac{1}{4}(2 - \dfrac{10}{3}\sqrt{\alpha^n})^2 & \alpha^n \text{ réelle} \\ \dfrac{25}{9}\alpha^n x^2 - \dfrac{5}{3}x\sqrt{\alpha^n}(2 + \dfrac{10}{3}\sqrt{\alpha^n}) + \dfrac{1}{4}(2 + \dfrac{10}{3}\sqrt{\alpha^n})^2 & \alpha^n \text{ complexe} \end{cases} \tag{7.15}$$

On ne considère que la solution réelle et la solution indépendante de α^n. Elles respectent toutes les deux la condition $T_2^{n+1}(1) = 1$. On sait que la première solution ne peut pas respecter la condition

$$K(T_2^{n+1}(\frac{3}{5}))grad(T_2^{n+1})(\frac{3}{5}) = K(T_1^{n+1/2}(\frac{3}{5}))grad(T_1^{n+1/2})(\frac{3}{5}) \tag{7.16}$$

puisque elle vaut zéro en $x = \dfrac{3}{5}$. La solution réelle dépendante de α donne :

$$K(T_2^{n+1}(\frac{3}{5}))grad(T_2^{n+1})(\frac{3}{5}) = \frac{10}{3}\frac{sign(-3+2\sqrt{\alpha^n})(-2\alpha^n + 3\sqrt{\alpha^n})}{-3+2\sqrt{\alpha^n}} \tag{7.17}$$

Or $K(T_1(3/5)grad(T_1))(3/5) = \frac{10}{3}\sqrt{\alpha^n}$ qui est positive. L'expression du domaine deux n'est positive que pour $\alpha^n \in [0, 9/4]$. C'est donc cette solution qui nous donnera la convergence de l'algorithme.

En prenant la valeur de $T_2(x)$ en $x = \frac{3}{5}$ il ne reste que deux suites possibles :

$$\alpha^{n+1} = \begin{cases} 0 \\ 4/9\alpha^n - 4/3\sqrt{\alpha^n} + 1 \end{cases} \tag{7.18}$$

Dans le cas où $\alpha^n > \frac{9}{4}$ seule la première solution indépendante de α peut convenir, celle-ci a l'avantage de ramener α^{n+1} à zéro. Alors, la solution réelle dépendante de α^n sera choisie. Celle-ci crée une suite qui converge vers la solution du problème à l'interface. En effet elle a comme point fixe $\alpha_r = 9/25$. Il est facile de vérifier que la condition nécessaire de convergence $|f'(\alpha_r)| < 1$ est respectée, de plus on a que pour $x \in I$, $f(I) \subset I$ ce qui confirme la convergence du point fixe vers α_r.

La figure 7.1 montre que la solution de l'algorithme de Neumann-Dirichlet est bien celle du problème analytique. La figure 7.2 compare la convergence de l'algorithme avec les deux conditions de transmission utilisées. Ces résultats numériques sont conformes aux calculs analytiques précédents.

7.2.2 Test avec $K(T) = \frac{1}{1+T^2}$

Maintenant, prenons la fonction $K(T) = \frac{1}{1+T^2}$, la solution analytique du problème global est

$$T(x) = tan(\frac{x\pi}{4}) \tag{7.19}$$

Les solutions analytiques de l'erreur des deux sous-domaines sont données par

$$\begin{cases} T_1^{n+1/2}(x) = tan(2arctan(\alpha^n)x) \\ T_2^{n+1}(x) = tan(\beta^{n+1/2}x + C) \end{cases} \tag{7.20}$$

La condition de transmission du domaine deux est la condition de flux du modèle hamiltonien à ports. Il n'est pas possible de déterminer exactement l'erreur dans le domaine 2 à cause de la constante C. L'analyse de la convergence de l'algorithme ne peut pas être effectuée. Cependant les tests numériques sur la méthode de Schwarz avec conditions de transmissions Dirichlet et K(T)grad(T) montrent une convergence linéaire sur la figure 7.3,

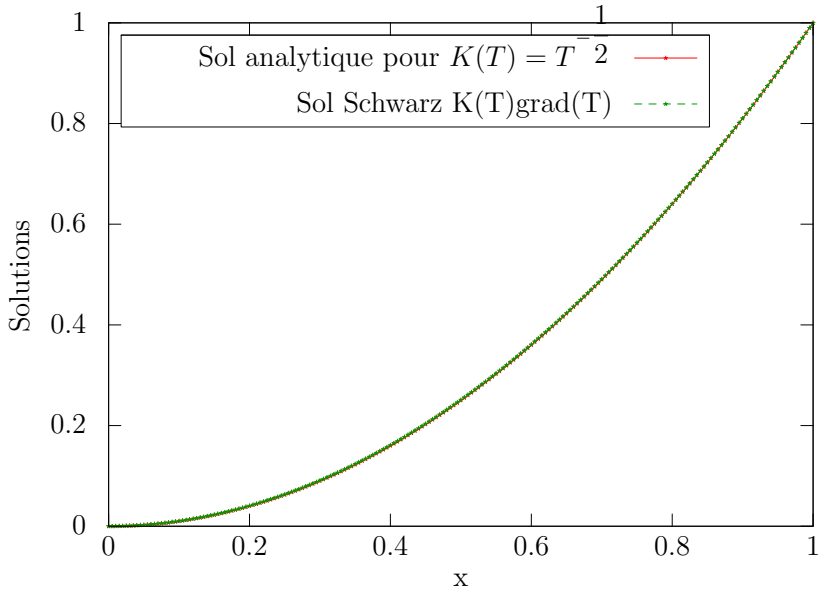


FIGURE 7.1 – Comparaison de la solution obtenue par l'algorithme de Schwarz avec les conditions de transmissions issues du formalisme hamiltonien à ports avec la solution analytique du problème (7.1) avec $K(T) = \sqrt{T}$

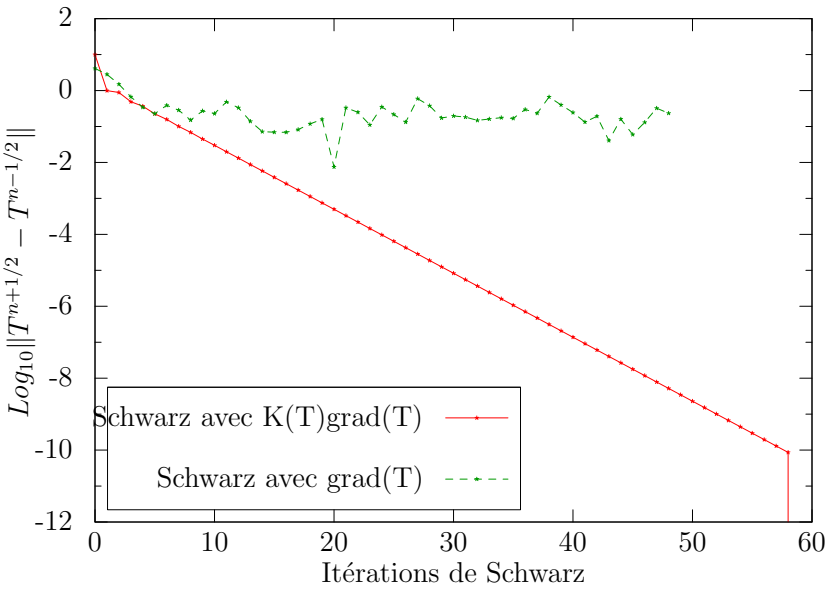


FIGURE 7.2 – Comparaison de la convergence de l'algorithme de Schwarz pour les conditions de transmissions grad(T) ou K(T)grad(T) avec $K(T) = \sqrt{T}$

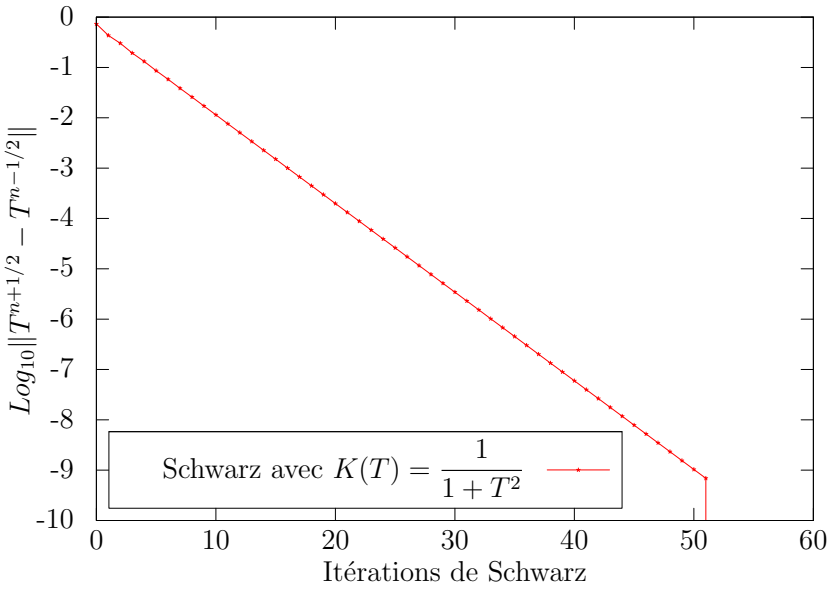


FIGURE 7.3 – Convergence de l'algorithme de Schwarz non-linéaire

sur laquelle il peut être envisagé d'utiliser les techniques du chapitre 3 pour l'accélération de la convergence.

Avec $K(T) = \frac{1}{1+T^2}$, en utilisant une condition dérivée Neumann, la résolution locale est parfois impossible, l'algorithme ne converge pas vers la solution du problème non décomposé.

7.3 Formulation de type complément de Schur dual

Dans le cas où le formalisme hamiltonien à ports ne peut être utilisé pour déduire des conditions de transmissions, une idée est de dualiser la contrainte de Dirichlet. Pour la méthode du complément de Schur cela permet de remplacer la condition obtenue par un multiplicateur de Lagrange. (Section 1.2.2 dans le cas linéaire.)

Soit la formulation faible du problème de la chaleur non-linéaire[38, 45] suivant :

$$\begin{cases} \text{Trouver } u \text{ tel que } u - u_0 \in H_0^1(\Omega), \forall v \in H_0^1(\Omega) \\ \int_\Omega K(u)\nabla u \nabla v d\Omega = \int_\Omega f.v d\Omega \end{cases} \tag{7.21}$$

avec la forme $a(u,v) = \int_\Omega K(u)\nabla u \nabla v dx$ et la forme linéaire $L(v) = \int_\Omega f.v dx$.

En posant $u = u_0 + w$ le problème variationnel peut se mettre sous la forme,

$$\begin{cases} \text{Trouver } w \in H_0^1(\Omega), \forall v \in H_0^1(\Omega) \\ a(w,v) = L(v) - a(u_0, v) \end{cases} \tag{7.22}$$

Soit la fonctionnelle $F(u)$

$$F(u) = \frac{1}{2}a(u,u) - L(u) = \frac{1}{2}\int_\Omega K(u)(\nabla u)^2 d\Omega - \int_\Omega f.u d\Omega \tag{7.23}$$

La fonctionnelle F(u) est continue puisque

$$\begin{aligned} a(u,v) &\leq ||K(u)||_{\infty,\Omega}||\nabla u||_{0,\Omega}||\nabla v||_{0,\Omega} && (7.24)\\ &\leq ||K(u)||_{\infty,\Omega}||u||_{1,\Omega}||v||_{1,\Omega} && (7.25) \end{aligned}$$

En supposant qu'il existe un ϵ tel que $|K(u)| > C_0 > \epsilon$, la coercivité se déduit en utilisant l'inégalité de Poincaré

$$\begin{aligned} a(u,u) &\geq C_0 \int_\Omega |\nabla u|^2 d\Omega && (7.26)\\ &\geq C_0\alpha|u|^2_{1,\Omega} \ \alpha \text{ constante de Poincaré} && (7.27)\\ &\geq C_0\alpha||u||^2_{0,\Omega} && (7.28) \end{aligned}$$

La forme linéaire $L(v)$ est continue sur $H_0^1(\Omega)$,

$$L(v) \leq ||f||_{0,\Omega}||v||_{1,\Omega} \tag{7.29}$$

Comme $C_0 > 0$, quand $||u||_{0,\Omega}$ tend vers l'infini, la fonction $F(u)$ tend vers l'infini. On obtient la coercivité de la fonctionnelle $F(u)$ et donc l'existence d'au moins un élément u qui minimise la fonctionnelle [30].

On peut alors décomposer $\Omega = \Omega_1 + \Omega_2$ en deux sous-domaines Ω_1 et Ω_2,

$$\begin{aligned} F(u) &= \frac{1}{2}\int_\Omega K(u)\nabla u \nabla u d\Omega - \int_\Omega f.u d\Omega && (7.30)\\ &= \frac{1}{2}\int_{\Omega_1} K(u)\nabla u \nabla u dx + \frac{1}{2}\int_{\Omega_2} K(u)\nabla u \nabla u dx - \int_{\Omega_1} f.u dx - \int_{\Omega_2} f.u dx \\ &= F_1(u_1) + F_2(u_2) && (7.31) \end{aligned}$$

De manière discrète maintenant en prenant $\Omega = [0,1](\Omega_1 = [0,1/2], \Omega_2 = [1/2,1])$ discrétisé sur n points, on définit F_1 et F_2 les fonctionnelles des problèmes locaux à minimiser sur chaque sous-domaine, en y ajoutant la condition de continuité de la solution, le problème global s'écrit :

$$\begin{cases} min\{F_1(u_1) + F_2(u_2)\} \\ u_{1,\gamma} = u_{2,\gamma} \end{cases} \tag{7.32}$$

En définissant la fonctionnelle

$$F : v = (v_1, v_2) \in (\mathbb{R}^{n/2})^2 \to F(v) = F_1(v_1) + F_2(v_2)$$

par rapport à l'ensemble

$$U = \{(v_1, v_2) \in (\mathbb{R}^{n/2})^2; \phi(v) \stackrel{\text{def}}{=} e_{n/2}^T v_1 - e_1^T v_2 = 0\}$$

définit pour une contrainte $\phi(v_1, v_2)) = \phi_1(v_1) + \phi_2(v_2)$ séparable.

D'après le théorème 7.2-2 de [30] , la fonction F admettant en u un extremum relatif par rapport à l'ensemble U, il existe un unique nombre λ tel que

$$F'(u) + \lambda\phi'(u) = 0 \tag{7.33}$$

L'équation reste vraie pour tout h tel que

$$F'(u).h + \lambda\phi'(u).h = 0 \tag{7.34}$$

L'application étant dérivable en $u \in \mathbb{R}^n$, elle possède des dérivées partielles par rapport à chacune des variables,

$$\begin{aligned} F'(u).h + \lambda\phi(u).h &= 0 && (7.35)\\ \partial_1 F(u).h_1 + \partial_2 F(u).h_2 + \lambda\partial_1\phi(u).h_1 + \lambda\partial_1\phi(u).h_1 &= 0 && (7.36)\\ h_1(\partial_1 F(u) + \lambda\partial_1\phi(u)) + h_2(\partial_2 F(u) + \lambda\partial_2\phi(u)) &= 0 && (7.37) \end{aligned}$$

pour tout $h = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \in \mathbb{R}^n$.

Le système suivant est alors déduit :

$$\begin{cases} \dfrac{\partial F_1}{\partial u_1}(u_1) + \lambda\phi'(u_1)e_{n/2} = 0 \\ \dfrac{\partial F_2}{\partial u_2}(u_2) - \lambda\phi'(u_2)e_1 = 0 \end{cases} \tag{7.38}$$

En définissant la fonction $G_k(u) = \dfrac{\partial F_k}{\partial u_k}(u_k) + \lambda\partial_k\phi(u)$ et en notant J_k la jacobienne de $G_k(u_k)$, on écrit une méthode de Newton pour résoudre chaque domaine en séparant les variables intérieures et interfaces :

$$\begin{cases} \begin{pmatrix} J_{k,i} & J_{k,c} \\ J_{k,b} & J_{k,e} + \phi''(u_k) \end{pmatrix} \begin{pmatrix} h_{k,i} \\ h_{k,e} \end{pmatrix} = \begin{pmatrix} G_{k,i}(u_k) \\ G_{k,e}(u_k) + \lambda\phi'(u_k) \end{pmatrix} \\ \begin{pmatrix} u_{k,i}^{n+1} \\ u_{k,e}^{n+1} \end{pmatrix} = \begin{pmatrix} u_{k,i}^{n} \\ u_{k,e}^{n} \end{pmatrix} - \begin{pmatrix} h_{k,i} \\ h_{k,e} \end{pmatrix} \end{cases} \tag{7.39}$$

k représente le numéro du domaine, l'indice i pour les inconnues intérieures et l'indice e pour les inconnues interfaces. λ est le multiplicateur de Lagrange à l'interface .

On cherche à exprimer de la solution aux noeuds interfaces en fonction de la solution aux noeuds intérieurs. Pour cela, il faut exprimer $h_{k,e}$ et $h_{k,i}$

$$h_{k,i} = (J_{k,i})^{-1}(G_{k,i}(u_k^n) - J_{k,c}h_{k,e}) \tag{7.40}$$

$$\begin{aligned} \lambda &= J_{1,b}h_{1,i} + (J_{1,e} + \phi_1''(u_1))h_{1,e} - G_{1,e}(u_1^n) && (7.41)\\ &= -(J_{2,b}h_{2,i} + (J_{2,e} + \phi_2''(u_2))h_{2,e} - G_{2,e}(u_2^n)) && (7.42) \end{aligned}$$

$h_{k,e}$ s'obtient en remplaçant $h_{k,i}$ dans (7.42),

$$\begin{aligned} h_{1,e} &= (J_{1,e} + \phi_1''(u_1) - J_{1,b}J_{1,i}^{-1}J_{1,c})^{-1}(-J_{1,b}J_{1,i}^{-1}G_{1,i}(u_1^n) - G_{1,e}(u_1^n) + \lambda)\\ &= S_1^{-1}(g_1 + \lambda) && (7.43) \end{aligned}$$

où $S_1 = J_{1,e} + \phi_1''(u_1) - J_{1,b}J_{1,i}^{-1}J_{1,c}$ et $g_1 = -J_{1,b}J_{1,i}^{-1}G_{1,i}(u_1^n) - G_{1,e}(u_1^n)$

$$h_{2,e} = S_2^{-1}(g_2 - \lambda) \tag{7.44}$$

La condition de continuité de la solution $u_{1,e}^{n+1} = u_{2,e}^{n+1}$ s'écrit

$$h_{1,e} = h_{2,e} + (u_{1,e}^n - u_{2,e}^n) \tag{7.45}$$

L'inconnue restante est λ donnée par

$$\lambda = (S_1^{-1} + S_2^{-1})^{-1}(S_2^{-1}g_2 - S_1^{-1}g_1) + (u_{1,e}^n - u_{2,e}^n) \tag{7.46}$$

Une fois le multiplicateur de Lagrange trouvé , la mise à jour du nouvel itéré u_k^{n+1} de la méthode de Newton se fait par le calcul de $h_{k,i}$ et $h_{k,e}$ par les équations (7.40) et (7.43). Contrairement à une approche classique consistant à linéariser le problème avant de lui appliquer une méthode de décomposition de domaine, nous effectuons un découpage en amont de la linéarisation. Notre approche s'inscrit dans la même démarche que [77], mais diffère par la condition de saut et par la méthode de réduction de celui-ci, notamment notre approche est basée sur un complément de Schur dual.

Les figures suivantes montrent les résultats de convergence de la méthode du complément de Schur pour différentes fonctions $K(u)$. Le domaine $[0, 1]$ est découpé deux sous-domaines discrétisés avec n=101 points avec un schéma aux différences finies.

Les deux figures 7.4 et 7.5 correspondant à la fonction $K(u) = \dfrac{1}{1+u^2}$, respectivement $K(u) = 1 + u + u^4$, montre la convergence de l'algorithme. 5

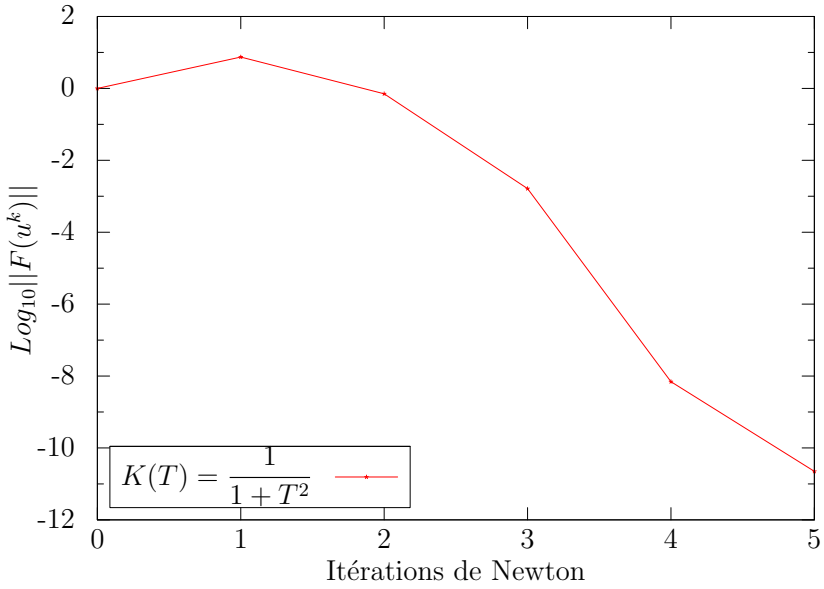


FIGURE 7.4 – Convergence des itérations de Newton pour le complément de Schur dual nonlinéaire pour l'équation (7.1) avec $K(T) = \frac{1}{1+T^2}$

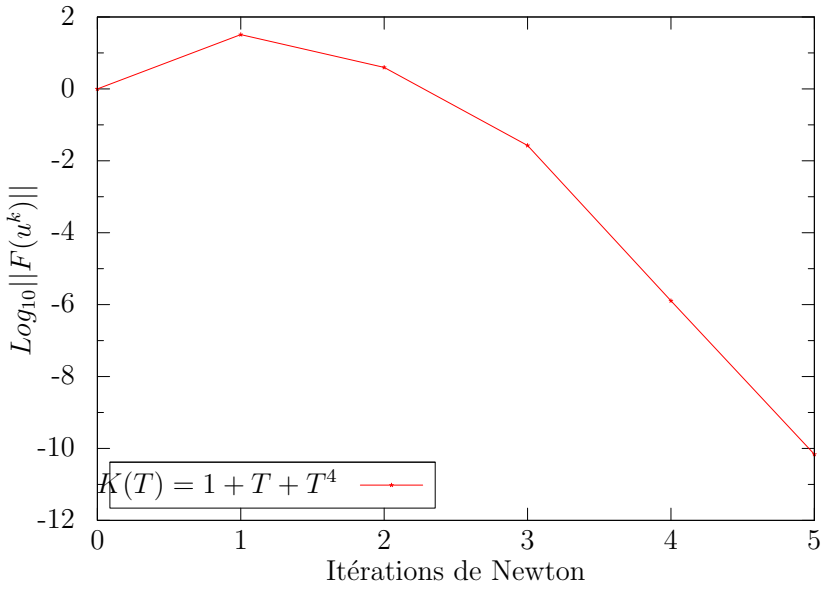


FIGURE 7.5 – Convergence des itérations de Newton pour le complément de Schur dual nonlinéaire pour l'équation (7.1) avec $K(T) = 1 + T + T^4$

itérations de Newton sont nécessaires pour atteindre une erreur à l'interface inférieure à 1.10^{-12} entre deux itérés. Ce faible nombre d'itérations est à relativiser par rapport au nombre d'évaluations de fonction nécessaires.

Nous avons également pris en exemple une fonction venant de la physique issue de [14]. Pour un gaz mono-atomique à faible densité et faible pression on peut utiliser la formule de Chapman-Enskog :

$$K(T) \simeq 1.989\,10^{-4}\frac{\sqrt{T/M}}{\sigma^2\Omega_k\,(T*)} \tag{7.47}$$

où M est sa masse moléculaire, σ la distance inter-atomique et $\Omega_k\,(T)$ l'intégrale de collision.

En utilisant le potentiel de Lennard-Jones d'interaction inter-atomique, on obtient :

$$\Omega_k\,(T*) = \frac{1.16145}{T^{*0.14874}} + \frac{0.52487}{\exp\,(0,.73207\,T^*)} + \frac{2.16178}{\exp\,(2.43787\,T^*)}$$

où $T^* = \frac{\kappa T}{\varepsilon}$ est la température réduite. Pour l'hydrogène H_2 on a les constantes suivantes : $M = 2.016$, $\frac{\varepsilon}{\kappa} = 38$ K et $\sigma = 2.915$ Å.

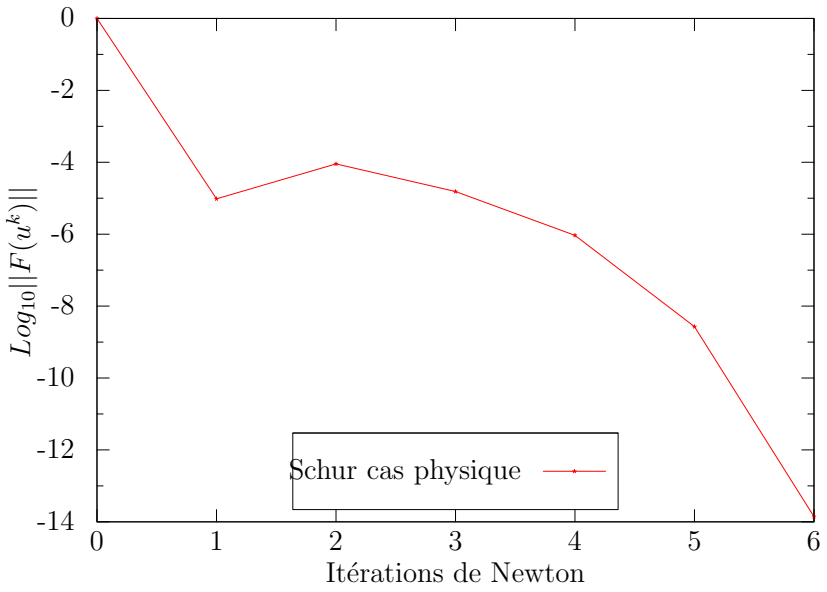


FIGURE 7.6 – Convergence des itérations de Newton pour le complément de Schur dual nonlinéaire pour l'équation (7.1) avec $K(T) = 1.989\,10^{-4}\frac{\sqrt{T/M}}{\sigma^2\Omega_k(T*)}$

La figure 7.6 représente la convergence du complément de Schur non-linéaire pour la conductivité thermique (7.47) pour l'hydrogène. L'algorithme converge en 7 itérations à 10^{-12}.

Remarque 7.1. Notons qu'en imposant une condition de Neumann dans la fonction G_k sur chaque sous-domaine, la méthode du complément de Schur dual développée ne converge pas vers la solution du problème initial.

Il est à noter que le nombre d'itérations de la méthode de Newton est identique pour le problème non découpé, la méthode est plus rapide que la méthode de type Schwarz. En effet, dans la méthode de Schwarz, la résolution par un Newton du problème local est nécessaire à chaque pas, même en considérant la solution de l'itération précédente comme solution initiale, la convergence lente de l'algorithme implique un plus grand nombre de calculs. Enfin, des techniques d'Uzawa[37, 82, 47, 56, 64] peuvent être également envisagées afin de résoudre le problème de minimisation en calculant le multiplicateur de Lagrange λ de manière itérative. Cette méthode a l'avantage de ne pas dériver la fonction de contrainte $\phi(u)$.

7.4 Conclusion

Nous avons d'abord introduit le formalisme hamiltonien à ports avant de proposer une idée basée sur ce formalisme afin de déterminer des conditions de transmissions non-linéaires pour les méthodes de décomposition de domaine de type Schwarz ou complément de Schur. Notre approche diffère de [59] essentiellement dans le choix de la condition de transmission et de la technique utilisée pour l'appliquer à la méthode de décomposition de domaine. Les avantages de la méthode du complément de Schur dual sont un meilleur temps de résolution ainsi qu'un nombre d'itérations inchangé de la méthode de Newton comparé au problème entier. La généralisation de la méthode à plusieurs sous domaines en 1D s'obtient aisément avec un système tridiagonal à résoudre sur les multiplicateurs de Lagrange. Dans de futurs travaux cette technique pourrait être adaptée à la résolution en parallèle des ODEs.

Conclusions

Dans ce travail nous nous sommes intéressés aux méthodes de décomposition de domaines en temps pour les EDOs linéaires et non-linéaires. Trois principaux axes ont été développés. Tout d'abord de nouvelles méthodes de décomposition de domaines permettant d'avoir un retour d'information de la fin de l'intervalle de temps, puis dans la continuité, nous avons abordé les méthodes d'accélération de la convergence des suites non linéaires générées par la décomposition de domaine. Ce travail considère également le formalisme hamiltonien à port afin de déterminer des conditions de transmissions non-linéaires pour les méthodes de type Schwarz en espace.

L'originalité de cette étude est l'adaptation des méthodes de décomposition de domaine, habituellement utilisées en espace , à l'intégration en temps des ODEs. Nous avons symétrisé l'intervalle de temps pour résoudre le problème de la condition de bord inconnue au temps final. La méthode du complément de Schur produit des résultats très satisfaisants pour la résolution d'ODEs linéaires. Une analyse de la méthode est effectuée en écrivant l'opérateur analytique pour des ODEs et pour l'équation de chaleur. Un autre problème lié à la convergence parfois très lente des méthodes de décomposition de domaine de type Schwarz nous amène à nous intéresser aux méthodes d'accélération de la convergence. D'abord dans le cas linéaire en utilisant les travaux sur la méthode d' Aitken-Schwarz.

Il est donc intéressant de pouvoir accélérer la convergence des méthodes dans le cas de problèmes non-linéaires. L'extrapolation et l'accélération des suites non-linéaires sont une grande partie des travaux de C.Brezinski. Dans ce cadre, nous avons testé et confronté plusieurs algorithmes dans différentes configurations. Les tests font ressortir les algorithmes basés sur l'ϵ-algorithme (vectoriel ou topologique). Nous avons également accéléré la méthode de type Schwarz par une méthode de Newton sur la suite interface. Cependant aucun algorithme n'apparait comme étant supérieur ou plus robuste. Dans le cadre de nos méthodes de décomposition de domaines, ces résultats ont montré qu'il n'existait pas d'algorithme universel. Dans de futures recherches, une perspective à envisager est l'utilisation en parallèle de tous les algorithmes

disponibles afin d'obtenir les meilleures performances.

Une autre originalité dans ce travail est la recherche de conditions de transmissions non-linéaires conservant les propriétés du problème. Nous proposons d'utiliser l'approche hamiltonienne à ports développée dans le domaine de l'automatique afin de définir des conditions de transmissions adéquates. Nos premiers tests sur l'équation de Saint-Venant n'ont pas été satisfaisants, dans le sens où on obtient systématiquement comme solution des deux conditions non-linéaires la solution triviale. Nous nous sommes orientés vers l'équation de la chaleur avec coefficient de conductivité thermique non-linéaire. Nous nous sommes appuyés premièrement sur la méthode Neumann-Dirichlet afin de montrer l'intérêt de l'approche. Puis nous avons appliqué la méthode du complément de Schur dual dans le cadre d'optimisation de problème non-linéaire. Les résultats obtenus sont intéressants du fait de la bonne convergence de la méthode du complément de Schur et de la résolution plus aisée des domaines locaux. Ces travaux sur les conditions de transmissions non-linéaires ouvrent des perspectives de recherche pour les méthodes de décomposition de domaines en temps.

Bibliographie

[1] Ralph Abraham and Jerrold E. Marsden. *Foundations of Mechanics*. second edition, 1994.

[2] Pierluigi Amodio and Luigi Brugnano. Parallel solution in time of odes : some achievements and perspectives. *Applied Numerical Mathematics*, 59(3-4) :424 – 435, 2009. Selected Papers from NUMDIFF-11.

[3] V. I. Arnol'd. *Mathematical Methods of Classical Mechanics*, volume 60. Second edition, 1989.

[4] Uri M. Ascher, Robert M. M. Mattheij, and Robert D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. SIAM, Philadelphia, 1995.

[5] Ahmed Baaiu, Françoise Couenne, Laurent Lefevre, and Yann Le Gorrec. Energy-preserving method for spatial discretization : application to an adsorption column. In Bertrand Braunschweig and Xavier Joulia, editors, *18th European Symposium on Computer Aided Process Engineering*, volume 25 of *Computer Aided Chemical Engineering*, pages 727 – 732. Elsevier, 2008.

[6] C. Batlle, F. Couenne, A. Doria-Cerezo, V. Duindam, E. Fossas, C. JAllut, L. Lefevre, Y. Le Gorrec, B. M. J. Maschke, R. Ortega, K. Schlacher, S. Stramigioli, and M. Tayakout. Port-based modeling in different domains. In *Modeling and Control of Complex Physical Systems*, pages 131–209. Springer Verlag, London, 2009.

[7] A. Berlinet. Estimating the degrees of an ARMA model. In *Lectures in computational statistics*, volume 3 of *Compstat Lectures*, pages 61–94. Physica, Heidelberg, 1984.

[8] A. Berlinet. Some useful algorithms in time series analysis. In *Alternative approaches to time series analysis (Rouen, 1982)*, volume 1 of *Travaux Rech.*, pages 95–120. Publ. Fac. Univ. Saint-Louis, Brussels, 1984.

[9] A. Berlinet. Sequence transformations as statistical tools. *Appl. Numer. Math.*, 1(6) :531–544, 1985.

[10] Alain Berlinet. Une méthode de détermination des degrés d'un modèle ARMA. *Publ. IRMA Lille*, 3(6) :exp. no. 1, 35, 1981.

[11] Alain Berlinet and Christian Francq. Identification of a univariate ARMA model. *Comput. Statist.*, 9(2) :117–133, 1994.

[12] Heiko Berninger, Ralf Kornhuber, and Oliver Sander. On nonlinear dirichlet—neumann algorithms for jumping nonlinearities. In Timothy J. Barth, Michael Griebel, David E. Keyes, Risto M. Nieminen, Dirk Roose, Tamar Schlick, Olof B. Widlund, and David E. Keyes, editors, *Domain Decomposition Methods in Science and Engineering XVI*, volume 55 of *Lecture Notes in Computational Science and Engineering*, pages 489–496. Springer Berlin Heidelberg, 2007.

[13] Heiko Berninger, Ralf Kornhuber, and Oliver Sander*. Convergence behaviour of dirichlet–neumann and robin methods for a nonlinear transmission problem. In Timothy J. Barth, Michael Griebel, David E. Keyes, Risto M. Nieminen, Dirk Roose, Tamar Schlick, Yunqing Huang, Ralf Kornhuber, Olof Widlund, and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering XIX*, volume 78 of *Lecture Notes in Computational Science and Engineering*, pages 87–98. Springer Berlin Heidelberg, 2011.

[14] Robert B. Bird, Warren E. Stewart, and Edwin N. Lightfoot. *Transport Phenomena*. John Wiley & Sons, New York, second edition, 2002.

[15] Petter E. Bjørstad and Olof B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. Numer. Anal.*, 23(6) :1097–1120, 1986.

[16] Didier Bresch and Jonas Koko. Operator-splitting and Lagrange multiplier domain decomposition methods for numerical simulation of two coupled Navier-Stokes fluids. *Int. J. Appl. Math. Comput. Sci.*, 16(4) :419–429, 2006.

[17] C. Brezinski. Généralisations de la transformation de Shanks, de la table de Padé et de l' ϵ-algorithme. *Calcolo*, 12(4) :317–360, 1975.

[18] C. Brezinski. Convergence acceleration during the 20th century. *J. Comput. Appl. Math.*, 122(1-2) :1–21, 2000. Numerical analysis 2000, Vol. II : Interpolation and extrapolation.

[19] Claude Brezinski. Accélération de suites à convergence logarithmique. *C. R. Acad. Sci. Paris Sér. A-B*, 273 :A727–A730, 1971.

[20] Claude Brezinski. Sur un algorithme de résolution des systèmes non linéaires. *C. R. Acad. Sci. Paris Sér. A-B*, 272 :A145–A148, 1971.

[21] Claude Brezinski. Conditions d'application et de convergence de procédés d'extrapolation. *Numer. Math.*, 20 :64–79, 1972/73.

[22] Claude Brezinski. *Accélération de la convergence en analyse numérique*. Lecture Notes in Mathematics, Vol. 584. Springer-Verlag, Berlin, 1977.

[23] Claude Brezinski. *Algorithmes d'accélération de la convergence*. Éditions Technip, Paris, 1978. Étude numérique, Collection Langages et Algorithmes de l'Informatique.

[24] Claude Brezinski. A survey of iterative extrapolation by the *E*-algorithm. *Skr. K. Nor. Vidensk. Selsk.*, (2) :1–26, 1989. Romberg Seminar on Quadrature/Interpolation/Extrapolation and Rational Approximations (Padé-continued Fractions) (Trondheim, 1988).

[25] Claude Brezinski and Michela Redivo Zaglia. *Extrapolation methods*, volume 2 of *Studies in Computational Mathematics*. North-Holland Publishing Co., Amsterdam, 1991. Theory and practice, With 1 IBM-PC floppy disk (5.25 inch).

[26] Claude Brezinski and Alain C. Rieu. The solution of systems of equations using the ε-algorithm, and an application to boundary-value problems. *Math. Comput.*, 28 :731–741, 1974.

[27] L. Brugnano and D. Trigiante. Boundary value methods : The third way between linear multistep and runge-kutta methods. *Computers and Mathematics with Applications*, 36(10-12) :269 – 284, 1998.

[28] Kevin Burrage and Bert Pohl. Implementing an ode code on distributed memory computers. *Computers Math. Applic*, 28 :235–252, 1994.

[29] J.C. Butcher. Implicit Runge-Kutta processes. *Math. Comp.*, 18 :50–64, 1964.

[30] Philippe G. Ciarlet. *Introduction à l'analyse numérique matricielle et à l'optimisation*. Collection Mathématiques Appliquées pour la Maîtrise. [Collection of Applied Mathematics for the Master's Degree]. Masson, Paris, 1982.

[31] F. Cordellier. Particular rules for the vector ε-algorithm. *Numer. Math.*, 27(2) :203–207, 1976/77.

[32] Florent Cordellier. *Interpolation rationnelle et autres questions. Problèmes algorithmiques et numériques.* PhD thesis, Univ. de Lille, juin 1989.

[33] T.J. Courant. Dirac manifolds. *T. Am. Math. Soc.*, pages 631–661, 1990.

[34] Morten Dalsmo and Arjan Van Der Schaft. On representations and integrability of mathematical structures in energy-conserving physical systems. *SIAM J. Control and Optimization*, 37 :54–91, 1996.

[35] Peter Deuflhard. *Newton methods for nonlinear problems*, volume 35 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2004. Affine invariance and adaptive algorithms.

[36] Irene Dorfman. *Dirac structures and integrability of nonlinear evolution equations*. John Wiley & Sons Ltd., Chichester, 1993.

[37] Howard C. Elman and Gene H. Golub. Inexact and preconditioned uzawa algorithms for saddle point problems. *SIAM J. Numer. Anal.*, 31 :1645–1661, December 1994.

[38] Alexandre Ern and Jean-Luc Guermond. *Éléments finis : théorie, applications, mise en œuvre*, volume 36 of *Mathématiques & Applications (Berlin) [Mathematics & Applications]*. Springer-Verlag, Berlin, 2002.

[39] C. Espinoza. Application de l'ϵ-alogrithme à des suites non scalaires et comparaison de quelques résultats numériques obtenues avec les ϵ ρ et θ-algorithmes. 1975.

[40] C. Farhat and M. Chandersis. Time-decomposed parallel time-integrators : theory and feasability studies for fluid, structure, and fluid structure applications. *Int. J. for Num. Methods in Engineering*, (58) :1397–1434, 2003.

[41] Martin J. Gander and Stefan Vandewalle. On the superlinear and linear convergence of the parareal algorithm. In *Domain decomposition methods in science and engineering XVI*, volume 55 of *Lect. Notes Comput. Sci. Eng.*, pages 291–298. Springer, Berlin, 2007.

[42] M. Garbey and D. Tromeur-Dervout. Méthodes Numériques et Couplage de codes pour le Calcul Distribué Distant. *Calculateur Parallèle*, 2002.

[43] M. Garbey and D. Tromeur-Dervout. On some Aitken-like acceleration of the Schwarz method. *Internat. J. Numer. Methods Fluids*, 40(12) :1493–1513, 2002. LMS Workshop on Domain Decomposition Methods in Fluid Mechanics (London, 2001).

[44] Marc Garbey. A direct solver for the heat equation with domain decomposition in space and time. In *Domain decomposition methods in science and engineering XVII*, volume 60 of *Lect. Notes Comput. Sci. Eng.*, pages 501–508. Springer, Berlin, 2008.

[45] G.Touzot et E.Lefrançois G.Dhatt. *Méthode des éléments finis*. Lavoisier, 2005.

[46] E. Gekeler. On the solution of systems of equations by the epsilon algorithm of Wynn. *Math. Comp.*, 26 :427–436, 1972.

[47] R. Glowinski, Q. V. Dinh, and J. Periaux. Domain decomposition methods for nonlinear problems in fluid dynamics. *Comput. Methods Appl. Mech. Engrg.*, 40(1) :27–109, 1983.

[48] H. Graf and M.S. Altinakar. *Hydraulique fluviale : écoulement permanent uniforme et non uniforme*. Traité de génie civil de l'école polytechnique

fédérale de Lausanne. Presses polytechniques et universitaires romandes, 2000.

[49] W.H. Graf and M.S. Altinakar. *Hydraulique fluviale - Ecoulement et phénomènes de transport dans les canaux à géométrie simple*, volume 16 of *Traité de génie civil de l'Ecole Polytechnique Fédérale de Lausanne*. Presses Polytechniques Universitaires Romandes, 2000. ISBN 978-2-88074-812-8.

[50] David Guibert. *Analyse de méthodes de résolution parallèles d'EDO/EDA raides*. PhD thesis, Université Claude Bernard - Lyon I, 09 2009.

[51] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993. Nonstiff problems.

[52] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2010. Stiff and differential-algebraic problems, Second revised edition, paperback.

[53] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006. Structure-preserving algorithms for ordinary differential equations.

[54] Bertrand Mantel Jacques Periaux and Hong Quan Chen. Intelligent interfaces of a schwarz domain decomposition method via genetic algorithms for solving nonlinear pdes : Application to transonic flows simulations, 1998.

[55] Rosenberg Ronald C. Karnopp Dean, Margolis Donald L. *System dynamics : Modeling and simulation of mechatronic systems*. Wiley, 3rd edition edition, 2000.

[56] Jonas Koko. Uzawa block relaxation domain decomposition method for a two-body frictionless contact problem. *Appl. Math. Lett.*, 22(10) :1534–1538, 2009.

[57] C. H. Lai. An acceleration technique for a nonoverlapped domain decomposition method. In *Computational and applied mathematics, I (Dublin, 1991)*, pages 289–293. North-Holland, Amsterdam, 1992.

[58] C. H. Lai. Applications of quasi-newton methods for the numerical coupling of some nonlinear problems, 97.

[59] C.-H. Lai, A. M. Cuffe, and K. A. Pericleous. A defect equation approach for the coupling of subdomains in domain decomposition methods. *Comput. Math. Appl.*, 35(6) :81–94, 1998.

[60] P. Linel and D. Tromeur-Dervout. Aitken-schwarz and schur complement methods for time domain decomposition. In *Parallel Computing : From Multicores and GPU's to Petascale*, volume 19 of *Advances in Parallel Computing*, pages 75–82. iospress, 2010.

[61] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. Résolution d'EDP par un schéma en temps "pararéel". *C. R. Acad. Sci. Paris Sér. I Math.*, 332(7) :661–668, 2001.

[62] Hatem Ltaief and Marc Garbey. A parallel Aitken-additive Schwarz waveform relaxation suitable for the grid. *Parallel Comput.*, 35(7) :416–428, 2009.

[63] Yvon Maday and Gabriel Turinici. A parareal in time procedure for the control of partial differential equations. *Comptes Rendus Mathematique*, 335(4) :387 – 392, 2002.

[64] Frédéric Magoulès and François-Xavier Roux. Lagrangian formulation of domain decomposition methods : A unified theory. *Applied Mathematical Modelling*, 30(7) :593 – 615, 2006. Parallel and Vector Processing in Science and Engineering.

[65] L. D. Marini and A. Quarteroni. An iterative procedure for domain decomposition methods : a finite element approach. In *First International Symposium on Domain Decomposition Methods for Partial Differential Equations (Paris, 1987)*, pages 129–143. SIAM, Philadelphia, PA, 1988.

[66] L. D. Marini and A. Quarteroni. A relaxation procedure for domain decomposition methods using finite elements. *Numer. Math.*, 55(5) :575–598, 1989.

[67] J. B. McLeod. A note on the ε-algorithm. *Computing (Arch. Elektron. Rechnen)*, 7 :17–24, 1971.

[68] Milan D. Mihajlović and David J. Silvester. Efficient parallel solvers for the biharmonic equation. *Parallel Comput.*, 30(1) :35–55, 2004.

[69] L. M. Milne-Thomson. *The Calculus of Finite Differences*. Macmillan and Co., Ltd., London, 1951.

[70] David D. Morrison, James D. Riley, and John F. Zancanaro. Multiple shooting method for two-point boundary value problems. *Comm. ACM*, 5 :613–614, 1962.

[71] Redha Moulla. *Méthodes géométriques pour la réduction spatiale des systèmes dynamiques*. PhD thesis, Université Lyon1, 2010.

[72] Alfio Quarteroni and Alberto Valli. *Domain decomposition methods for partial differential equations*. Numerical Mathematics and Scientific Computation. The Clarendon Press Oxford University Press, New York, 1999. Oxford Science Publications.

[73] Barre De Saint-Venant. Theorie du movement non-permanent des eaux crues des rivieres et a lintroduction des marees dans leur lit. *Acad. Sci. (Paris) Comptes rendus*, 73 :147–154, 237–240, 1871.

[74] A. J. Van der Schaft. *L2-Gain and Passivity Techniques in Nonlinear Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996.

[75] R. J. Schmidt. On the numerical solution of linear simultaneous equations by an iterative method. *Philos. Mag. (7)*, 32 :369–383, 1941.

[76] Daniel Shanks. Non-linear transformations of divergent and slowly convergent sequences. *J. Math. and Phys.*, 34 :1–42, 1955.

[77] Pericleous Koulis Siahaan Antony, Lai Choi-Hong. A nonoverlapping domain decomposition method for nonlinear physical processes. *Proc. Appl. Math. Mech.*, 7 :2140003–2140004, 2007.

[78] Avram Sidi. A convergence and stability study of the iterated lubkin transformation and the theta-algorithm. *Math. Comput.*, 72(241) :419–433, 2003.

[79] S. Skelboe. Parallel algorithms for a direct circuit simulator. In Erik Lindberg, editor, *Proceedings of the 10th European Conference on Circuit Theory and Desing*, Copenhagen, Denmark, 1991.

[80] S. Skelboe. Integration of partitioned stiff systems of ordinary differential equations. In *Proceedings of the Third International Workshop on Applied Parallel Computing in Industrial Computation and Optimization (PARA'96)*, volume 1184, pages 621–630, Lyngby, Denmark, 1996. Springer lecture Notes in Computer Science.

[81] S. Skelboe and Z. Zlatev. Exploiting the natural partioning in the numercial solution of ode systems arising in atmospheric chemistry. *Springer Lecture Notes in Computer Science*, 1996.

[82] R. Sugino, H. Imai, and N. Tosaka. Boundary element scheme with domain decomposition approach for moving interface phenomenon. In *Domain decomposition methods in sciences and engineering (Chiba, 1999)*, pages 207–214 (electronic). DDM.org, Augsburg, 2001.

[83] Andrea Toselli and Olof Widlund. *Domain decomposition methods—algorithms and theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2005.

[84] A. J. van der Schaft and B. M. Maschke. Hamiltonian formulation of distributed-parameter systems with boundary energy flow. *Journal of Geometry and Physics*, 42(1-2) :166 – 194, 2002.

[85] Jet Wimp. *Sequence transformations and their applications*, volume 154 of *Mathematics in Science and Engineering*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1981.

[86] P. Wynn. On a device for computing the $e_m(S_n)$ tranformation. *Math. Tables Aids Comput.*, 10 :91–96, 1956.

[87] P. Wynn. Acceleration techniques for iterated vector and matrix problems. *Math. Comp.*, 16 :301–322, 1962.

[88] P. Wynn. Upon systems of recursions which obtain among the quotients of the Padé table. *Numer. Math.*, 8 :264–269, 1966.