Méthodes de décomposition de domaine : application à la résolution de problèmes de contrôle optimal

Aïcha Bounaim

THÈSE

Présentée par

Aïcha BOUNAIM

Pour obtenir le titre de

Docteur de l'Université Joseph Fourier - Grenoble I

(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

Spécialité : **Mathématiques Appliquées**

Méthodes de décomposition de domaine : Application à la résolution de problèmes de contrôle optimal.

Date de soutenance : 25 juin 1999

Composition du Jury :

M.	P. Witomski	Président
M.	J. Pousin	Rapporteur
M.	J.-D. Benamou	Rapporteur
M.	J. Blum	Examinateur
M.	E. Blayo	Examinateur

Thèse préparée au sein du laboratoire LMC-IMAG dans le cadre du projet IDOPT(CNRS, INRIA, UJF, INPG, IMAG)

Remerciements

Cette thèse n'aurait pu aboutir sans la présence de plusieurs personnes que je tiens à remercier ici :

M. Patrick Witomski, Professeur à l'université Joseph Fourier, directeur du LMC pour la plus grande partie de ma thèse, qui a accepté, à mon grand honneur, de présider mon jury de thèse.

M. Jérôme Pousin, Professeur à l'INSA de Lyon, qui a accepté d'examiner le présent mémoire et pour l'intérêt qu'il lui a porté.

M. Jean-David Benamou, Chercheur à l'INRIA Rocquencourt, qui a accepté de rapporter sur ce mémoire. Ses remarques et ses conseils m'on aidé à en améliorer la première version.

M. Eric Blayo, maître de conférence à l'université Joseph Fourier, qui m'a fait l'honneur de participer à ce jury.

Je témoigne enfin ma sincère reconnaissance à M. Jacques Blum, Professeur à l'université Joseph Fourier et directeur du LMC, pour m'avoir proposé un sujet de thèse très intéressant (qui m'a permis de voyager!), pour m'avoir obtenu un financement et pour ses discussions fructueuses et ses encouragements tout au long de la thèse. Je le remercie plus spécialement pour la confiance dont il a fait part à mon égard et pour la liberté qu'il m'a accordée d'éprouver mes idées et de mener à bien ma recherche en m'aidant de ses conseils judicieux.

Je tiens à remercier également M. Taoufik Sassi, maître de conférence à l'INSA de Lyon, pour ses remarques et ses nombreux conseils pendant la rédaction de ce mémoire.

Un grand merci à tous ceux que j'ai connus et côtoyés au LMC durant cette thèse et qui y ont fait régner une bonne ambiance aussi bien de travail que de détente. Merci à Lemine, Hafsa, Mohamed Khachan pour leur précieuse amitié et à Hans-Emmanuel et Bruno pour les avoir "embêtés" assez souvent.

Enfin, je ne sais comment m'exprimer envers ceux qui m'ont soutenue, encouragée et aidée de très près, je pense à mon petit "grand" frère Hassan et mes chers amis Samy, Samira et la famille Boussetta.

Mes derniers et profonds remerciements vont à mes chers parents pour leur amour et pour leur confiance en moi ainsi qu'au reste de la famille: Khadija, Najia et Hania sans oublier tous ceux qui m'ont donné le goût des études (mes maîtres et mes professeurs).

Je tiens enfin à exprimer ma gratitude envers MM. Jean-Charles Gilbert et Claude Lemaréchal pour leur code numérique d'otpimisation M1QN3 (Librairie MODULOPT de l'INRIA) et le centre de Calcul du CEA de Grenoble qui nous autorisés à effectuer une partie des calculs sur le CRAY T3E.

Thèse de doctorat de l'université Joseph Fourier.

Méthodes de décomposition de domaine: Application à la résolution de problèmes de contrôle optimal.

Auteur: **Aïcha BOUNAIM.**[1]

Sous la direction du:

Professeur Jacques BLUM.[1]

Résumé: Ce travail porte sur l'étude des méthodes de décomposition de domaine et leur application pour résoudre des problèmes de contrôle optimal régis par des équations aux dérivées partielles. Le principe de ces méthodes consiste à ramener des problèmes de grande taille sur des géométries complexes en une suite de sous-problèmes de taille plus petite sur des géométries plus simples.

En considérant une décomposition sans recouvrement, l'intérêt de ces méthodes pour les problèmes de contrôle optimal réside au niveau de l'intégration de l'équation d'état, puisqu'il est possible de partitionner le problème en une suite de problèmes plus petits, quitte à contraindre les interfaces entre les sous-domaines à obéir à des conditions de raccordement afin de déduire la solution globale à partir des solutions locales. Dans une première partie, nous étudions le cas elliptique. Nous considérons simultanément la minimisation de la fonction coût et des raccordements sur les frontières entre les sous-domaines. Cette combinaison de problèmes de minimisation et de méthodes de décomposition de domaine est traitée par des techniques de Lagrangien augmenté. Nous montrons que, sur le domaine décomposé, le problème initial se réduit à la recherche d'un point-selle. Une étude des méthodes de Lagrangien nous a permis de choisir une variante d'algorithmes existants dans la littérature et de les combiner avec un algorithme de décomposition de domaine.

Dans la seconde partie, nous développons l'extension de cette approche aux problèmes de contrôle optimal régis par des systèmes paraboliques en considérant uniquement une décomposition en espace du domaine de calcul.

Dans une dernière partie, nous considérons une décomposition de domaine avec recouvrement à chaque pas de la minimisation. D'une part, nous construisons un algorithme parallèle en utilisant la méthode de Schwarz multiplicative en tant que solveur. Ceci permet de déduire naturellement l'état adjoint par transposition des systèmes directs locaux. L'algorithme global défini par la méthode de minimisation de type quasi-Newton et ce solveur de Schwarz constitue une méthode robuste de résolution du problème de contrôle optimal, mais coûteuse. D'autre part, et plus particulièrement, pour des problèmes de grande taille, l'algorithme de type quasi-Newton, combiné avec le solveur de Krylov `BiCGSTAB` préconditionné par une méthode de Schwarz additive, est plus compétitif dans la mesure où l'on obtient de bonnes performances parallèles. De nombreux résultats sont présentés pour préciser le comportement des algorithmes d'optimisation quand ils sont utilisés avec des méthodes de Schwarz.

1. **Projet IDOPT**, *commun à l'INRIA Rhône-Alpes et au Laboratoire LMC de l'IMAG (l'IMAG est une fédération d'unités de recherche de l'INPG et/ou de l'UJF, associées au CNRS), BP 53, 38041 Grenoble Cedex 9, France.*

Domain decomposition methods: Application to the solution of optimal control problems.

Abstract: This work deals with the use of domain decomposition methods to solve optimal control problems governed by partial differential equations. The problem on the whole domain decomposes into local subproblems with the cost function naturally resulting from the sum of the local cost functions provided that the perfect matching of the local solutions is preserved on both the sides of the interface between non-overlapping neighbour subdomains.

In a first part, we especially study the elliptic case: we take simultaneously into account the minimization of the cost function and the links between subdomains. These links are satisfied by ensuring the flux continuity of the direct state given explicitly on the local system whereas the continuity constraint is considered with multipliers. The resulting optimization problem is treated by augmented Lagrangian techniques and is solved thanks to a variant of algorithms existing in the literature.
In a second part, an extension to optimal control problems governed by parabolic system is developed considering only a space decomposition of the calculation domain.

In the last part, we combine Schwarz algorithms and optimal control problems using an overlapping domain decomposition at each step of the minimization process. We design a parallel algorithm based on a multiplicative Schwarz method used as a solver. The adjoint state is naturally deduced from the transposition of the local direct. This mixed algorithm constitutes a robust but expensive method to solve the optimal control problem. For particularly large problems, another algorithm combining a Quasi-Newton method and a Krylov solver (`BiCGSTAB`) preconditioned by an additive Schwarz method is proven to be more competitive since good parallel efficiencies are obtained. Results are presented to show the behaviour of the optimization solvers when Schwarz algorithms are used.

Table des matières

Table des figures

Liste des tableaux

Introduction.

Les méthodes de décomposition de domaine ont récemment suscité un grand intérêt grâce aux progrès qu'ont connus le développement et l'évolution des architectures parallèles. Nous nous intéressons à l'application de ces méthodes pour la résolution de problèmes de contrôle optimal régis par des équations aux dérivées partielles.

Nous traitons dans cette étude des méthodes itératives de décomposition de domaine avec et sans recouvrement. Le plan détaillé de la thèse est le suivant:

Dans un premier chapitre, nous exposons quelques généralités sur les méthodes de décomposition de domaine, leur historique, les motivations de leur développement et nous en donnons des exemples, à savoir, les méthodes de Schwarz: additive ou multiplicative, avec ou sans recouvrement. Nous rappelons également dans ce chapitre des résultas généraux sur les méthodes de Lagrangien et de Lagrangien augmenté que nous utilisons dans le chapitre 3.

Comme notre but est d'appliquer ces méthodes à des problèmes de contrôle optimal particuliers, nous en exposons un problème modèle au deuxième chapitre. Des rappels théoriques ainsi que les aspects pratiques de sa résolution numérique sont présentés.

L'objectif du troisième chapitre est de développer une méthode de décomposition de domaine sans recouvrement: le domaine de calcul est divisé en un ensemble de sous-domaines non recouvrants. Ensuite, nous minimisons simultanément la fonction coût et les raccordements aux interfaces entre les sous-domaines. Ces contraintes supplémentaires de continuité sont traitées par une technique de Lagrangien augmenté et on montre que le problème global se réduit à un problème décomposé de recherche de point-selle. Des algorithmes sont mis au point et étudiés dans le cas du problème modèle du chapitre 2.

Pour les applications numériques, nous avons utilisé des outils de programmation sur les machines à mémoire distribuée. Le chapitre 4. est consacré à une présentation des mécanismes du calcul parallèle ainsi que du fonctionnement de la librairie de passage de messages `MPI`.

Dans le chapitre 5., des résultats numériques sont exposés pour l'application des mé-

thodes du chapitre 3. à un problème de contrôle optimal elliptique. Nous revenons à la formulation du chapitre 3. pour une extension aux problèmes paraboliques et ceci est développé au chapitre 7. après avoir présenté des généralités sur les problèmes de contrôle optimal gouvernés par des edp paraboliques au chapitre 6.

Dans le dernier chapitre, nous présentons une variante de méthodes de décomposition de domaine avec recouvrement. Nous développons des algorithmes parallèles basés sur les algorithmes itératifs de Schwarz. Nous présentons également divers résultats numériques décrivant le comportement des algorithmes de minimisation quand ils sont combinés avec des solveurs de Krylov pour résoudre les systèmes direct et adjoint. A la fin du chapitre, nous passons brièvement à l'extension naturelle et directe aux problèmes de contrôle optimal dont l'équation d'état est une edp parabolique.

Chapitre **1 :**

Généralités et rappels.

Nous exposons ici les principes et généralités relatifs aux deux méthodes numériques qui sous-tendent notre exposé à savoir les méthodes de décomposition de domaine (MDD) et les algorithmes de Lagrangien augmenté. Nous ne donnons ici que les théorèmes et résultats généraux pour mieux situer notre travail. Nous reviendrons en détail sur les MDD et sur leur implémentation comme préconditionneur d'un solveur de Krylov au chapitre 8., et sur l'utilisation de Lagrangien augmenté dans le cadre d'un problème d'optimisation tout au long de la 1ère partie.

1.1 Méthodes de décomposition de domaine: Généralités et historique

1.1.1 Introduction

Au cours de la dernière décennie, les machines parallèles ont connu une grande évolution, en particulier en puissance de calcul et en capacité de stockage. Les simulations de problèmes physiques de l'ordre de la dizaine de millions de degrés de liberté sont désormais chose courante et la vitesse d'exécution des codes scientifiques s'exprime aujourd'hui en centaines de Mflops (Millions d'opérations flottantes par seconde). Développer des algorithmes bien adaptés à de telles machines est donc devenu impératif. Parmi les méthodes qui fournissent de bons algorithmes faciles à implémenter sur des architectures parallèles et ayant une base mathématique très riche, on trouve les méthodes de décomposition de domaine (**MDD**). Bien qu'elles soient très anciennes puisque la première remonte à la fin du XIXème siècle avec H. A. Schwarz qui lui donna son nom [Schwarz, 1890], elles ont suscité récemment un grand intérêt et la littérature abondante qui leur est consacrée en témoigne.

Tout d'abord, l'expression "**décomposition de domaine**" posséde différentes significations chez les spécialistes des équations aux dérivées partielles :

En calcul parallèle, la décomposition de domaine se référe aux techniques pour décomposer les données et cela peut ne pas dépendre des méthodes numériques utilisées.

En analyse asymptotique, décomposer le domaine veut dire diviser le domaine physique en des domaines modélisés par des équations aux dérivées partielles différentes en y ajoutant des conditions bien appropriées sur les interfaces entre les sous-domaines.

Pour les méthodes de préconditionnement, la décomposition se fait au niveau des systèmes linéaires qui résultent de la discrétisation des équations aux dérivées partielles. Par conséquent, les méthodes de décomposition de domaine deviennent plus un outil pour construire de bons préconditionneurs que la décomposition en elle même.

1.1.2 Motivations du développement des MDD

Le premier objectif des méthodes de décomposition de domaine est de trouver des solutions efficaces aux problèmes gouvernés par des équations aux dérivées partielles sur des géométries compliquées et leurs approximations sur de très fines grilles. En effet, cela engendre des systèmes linéaires de grande taille et difficiles à résoudre par des solveurs directs ou des méthodes itératives puisque de tels systèmes sont souvent mal conditionnés. Ainsi, les MDD permettent de décomposer le problème initial en des sous-problèmes de petite taille et sur des géométries plus simples. Elles constituent un compromis entre l'usage de solveurs directs pour le calcul local sur chaque sous-domaine et l'utilisation de solveurs itératifs au niveau des interfaces.
Il faudra ajouter que les méthodes de décomposition de domaine fournissent une façon très naturelle de dériver des algorithmes bien adaptés aux machines parallèles. C'est pourquoi, elles ont récemment suscité un grand intérêt du point de vue théorique et pratique. A cet effet et depuis 1987, une conférence internationale se tient annuellement pour présenter les nouveaux résultats et les diverses applications qui sont recueillis par la suite dans des "*Proceedings*" de ces conférences, *cf.* [Glowinski et al., 1988], [Chan et al., 1989], [Chan et al., 1990], [Glowinski et al., 1991], [Keyes et al., 1992], [Quarteroni et al., 1994], [Keyes et Xu, 1995], [Glowinski et al., 1996], [Bjørstad et al., 1998].
Quant aux diverses et importantes raisons qui ont conduit à la popularité de ces méthodes, cela peut être résumé dans le fait qu'elles :

- sont bien adaptées aux machines parallèles dont le développement et les caractéristiques ne cessent de s'accroître,
- possédent un intérêt mathématique intrinsèque,
- peuvent s'appliquer à des problèmes définis sur des géométries complexes,
- possédent une base théorique solide,
- facilitent l'utilisation de schémas numériques différents pour chaque sous-problème, par exemple, éléments finis, différences finies, méthodes spectrales ou de collocation...

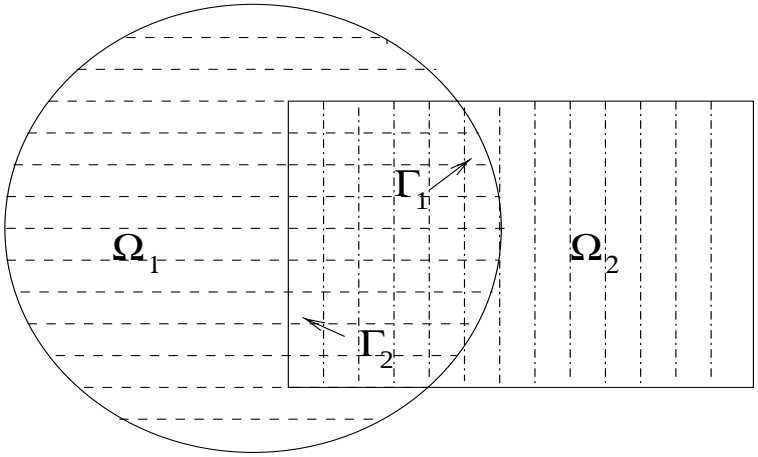

FIG. 1.1 – *La fameuse figure de Schwarz.*

- et peuvent être combinées avec une résolution par d'autres méthodes telles les méthodes multi-échelles ou le raffinement local de maillage.

Reste aux spécialistes des méthodes de décomposition de domaine à sélectionner des algorithmes performants tant au niveau de leur temps d'exécution sur machines parallèles qu'en ce qui concerne leurs propriétés de convergence.

1.1.3 Historique des méthodes de décomposition de domaine

La première méthode de décomposition de domaine fut proposée par **H. A. Schwarz** dans une preuve d'existence pour résoudre un problème de Poisson sur un domaine Ω de géométrie complexe (voir figure 1.1). La méthode alternée de Schwarz s'applique à la résolution de problèmes elliptiques sur un domaine formé d'une réunion de deux sous-domaines qui se recouvrent et on résout alternativement sur chaque sous-domaine le même problème.

Pour simplifier, nous allons nous limiter au cas d'un problème elliptique avec des conditions aux limites de Dirichlet homogènes :

$$\left\{ \begin{array}{rcll} -\Delta u & = & f \quad \text{dans} & \Omega = \Omega_1 \cup \Omega_2, \\ u & = & 0 \quad \text{sur} & \partial\Omega. \end{array} \right. \tag{1.1}$$

Avec la décomposition $\Omega = \Omega_1 \cup \Omega_2$ telle qu'elle est donnée par la figure 1.1, on initialise sur Ω_1 en fixant $u^0|_{\Gamma_1}$ et pour $n \geq 0$, on résout alternativement :

$$\left\{ \begin{array}{rcll} -\Delta u^{n+\frac{1}{2}} & = & f \quad \text{dans} & \Omega_1, \\ u^{n+\frac{1}{2}} & = & 0 \quad \text{sur} & \partial\Omega_1 \setminus \Gamma_1, \\ u^{n+\frac{1}{2}} & = & u^n \quad \text{sur} & \Gamma_1, \end{array} \right. \tag{1.2}$$

$$\begin{cases} -\Delta u^{n+1} &= f & \text{dans} \quad \Omega_2, \\ u^{n+1} &= 0 & \text{sur} \quad \partial\Omega_2 \setminus \ , _2, \\ u^{n+1} &= u^{n+\frac{1}{2}} & \text{sur} \quad \ , _2. \end{cases} \tag{1.3}$$

Ce procédé est alors répété jusqu'à ce qu'un critère d'arrêt soit vérifié, par exemple, lorsque l'on obtient :

$$\|u^{n+1}|_{\Gamma_1} - u^n|_{\Gamma_1}\| \leq \epsilon \quad \text{et} \quad \|u^{n+1}|_{\Gamma_2} - u^n|_{\Gamma_2}\| \leq \epsilon,$$

ou

$$\|u^{n+1}|_{\Omega_1} - u^n|_{\Omega_1}\| \leq \epsilon \quad \text{et} \quad \|u^{n+1}|_{\Omega_2} - u^n|_{\Omega_2}\| \leq \epsilon.$$

La convergence vers la solution globale de cet algorithme a été établie par Schwarz en utilisant le Principe du Maximum.
Ensuite, vers les années 30, Sobolev a decouvert la formulation variationnelle de l'algorithme de Schwarz pour des problèmes en élasticité linéaire [Sobolev, 1936]. Le résultat de Sobolev offre un point de départ pour utiliser l'algorithme pour des problèmes ne vérifiant pas le principe du maximum.
Puis, en 1988, P.-L. Lions donne une interprétation à cet algorithme à l'aide d'une formulation variationnelle [Lions, 1988]. Il précise que la convergence de cette méthode dépend de la taille du recouvrement δ, du pas de discrétisation h et du diamètre caractéristique des sous-domaines H. Bien entendu, plus le recouvrement δ est important et meilleure sera la convergence. Voir également [Lions, 1989, Lions, 1990].

1.2 La méthode de Schwarz classique

La méthode alternée de Schwarz classique est celle que l'on a présentée ci-dessus. On précise ici le sens des adjectifs "multiplicatif" et "additif" qui lui sont attribués.

1.2.1 Méthode de Schwarz multiplicative

Ecrivons maintenant le problème (1.1) sous forme variationnelle :

$$a(u,v) = f(v) \qquad \forall v \in V = H_0^1(\Omega) \tag{1.4}$$

où

$$a(u,v) = \int_\Omega \nabla u \nabla v dx \quad \text{et} \quad f(v) = \int_\Omega f v dx.$$

Nous allons interpréter maintenant la méthode alternée de Schwarz en termes de de projection d'erreur. Pour cela, nous introduisons les formes bilinéaires a_1 et a_2, qui sont les restrictions de a aux sous-espaces V_1 et V_2 :

$$\begin{array}{rcl} a_1(u,v) = a(u,v) \quad \forall u,\ v \in V_1 & = & \{u \in V | u|_{\Omega \setminus \Omega_1}\}, \\ a_2(u,v) = a(u,v) \quad \forall u,\ v \in V_2 & = & \{u \in V | u|_{\Omega \setminus \Omega_2}\}, \end{array}$$

Avec ces notations et en considérant les différences $u^{n+1} - u^{n+\frac{1}{2}}$ et $u^{n+\frac{1}{2}} - u^n$, l'algorithme proposé par Schwarz, sous forme variationnelle, s'écrit :

$$\left\{ \begin{array}{lcll} a(u^{n+\frac{1}{2}} - u^n, v) & = & f(v) - a(u^n, v), & u^{n+\frac{1}{2}} - u^n \in V_1, \quad \forall v \in V_1, \\ a(u^{n+1} - u^{n+\frac{1}{2}}, v) & = & f(v) - a(u^{n+\frac{1}{2}}, v), & u^{n+1} - u^{n+\frac{1}{2}} \in V_2, \quad \forall v \in V_2. \end{array} \right. \tag{1.5}$$

En définissant l'erreur e^n par $e^n = u - u^n$, avec u est la solution unique (par le théorème de Lax-Milgram) de (1.4) , on aboutit à :

$$\left\{ \begin{array}{lcl} a((u^{n+\frac{1}{2}} - u^n) - e^n, v) & = & 0, \quad \forall v \in V_1, \\ a((u^{n+1} - u^{n+\frac{1}{2}}) - e^{n+\frac{1}{2}}, v) & = & 0, \quad \forall v \in V_2. \end{array} \right. \tag{1.6}$$

A chaque demi-itération, $u^{n+1} - u^{n+\frac{1}{2}}$ et $u^{n+\frac{1}{2}} - u^n$ ne sont autres que les projections (relativement au produit scalaire induit par $a(.,.)$) de l'erreur sur V_1 et V_2 respectivement. Et les équations (1.6) s'écrivent donc :

$$\left\{ \begin{array}{lcl} u^{n+\frac{1}{2}} & = & u^n + P_{a_1}.e^n, \\ u^{n+1} & = & u^{n+\frac{1}{2}} + P_{a_2}.e^{n+\frac{1}{2}}, \end{array} \right. \tag{1.7}$$

ou encore,

$$\left\{ \begin{array}{lcl} e^{n+\frac{1}{2}} & = & (I - P_{a_1}).e^n, \\ e^{n+1} & = & (I - P_{a_2}).e^{n+\frac{1}{2}}. \end{array} \right. \tag{1.8}$$

D'où finalement,

$$e^{n+1} = (I - P_{a_2})(I - P_{a_1}).e^n. \tag{1.9}$$

Ce qui permet d'interpréter l'algorithme de Schwarz alterné en termes de projections et corrections successives de l'erreur e^n sur chacun des sous-espaces V_i. La relation (1.9) se généralise facilement dans le cas d'une décomposition en m sous-domaines à :

$$e^{n+1} = \Pi_{i=m}^{i=1}(I - P_{a_i}).e^n. \tag{1.10}$$

et on retrouve donc l'adjectif "**multiplicatif**" qu'on attribue à la méthode alternée de Schwarz.

Cette "*correction successive de l'erreur*" s'interpréte algébriquement comme une itération de Gauss-Seidel par blocs où chaque bloc est relatif à l'opérateur restreint sur un sous-domaine. Le degré de parallélisme dans ce cas est moins important puisque chaque processeur (ou sous-domaine) i doit attendre que la correction soit faite sur les domaines Ω_1, ...Ω_{i-1} avant d'entamer son propre calcul.

1.2.2 Méthode de Schwarz additive

Une autre façon de transformer l'algorithme et accroître son efficacité parallèle réside en l'écriture d'une nouvelle version "additive" qui revient à projeter l'erreur e^n simultanément sur chaque sous-domaine et à cumuler par la suite les corrections locales sur le domaine global. Au niveau algorithmique, cela consiste à remplacer la condition de Dirichlet $u^{n+1} = u^{n+\frac{1}{2}}$ sur $,_2$ par $u^{n+1} = u^n$ sur $,_2$. Sous la forme variationnelle, l'algorithme devient :

$$e^{n+1} = (I - \sum_{i=m}^{i=1} P_{a_i}).e^n. \tag{1.11}$$

D'où le nom d' "**additif**" donné à cette méthode et qui a été proposée au départ par M. Dryja et O. Widlund [Dryja et Widlund, 1989]. Ses propriétés sont étudiées dans [Bjørstad, 1987] et [Bramble et al., 1990]. Il faut souligner que la convergence de la méthode de Schwarz additive est moins bonne que sa version multiplicative. En termes algébriques, elle correspond à une méthode de Jacobi par blocs où chaque bloc est relatif à l'opérateur restreint sur un sous-domaine.

Remarque 1.2.1 *Dans l'algorithme de Schwarz, les conditions aux limites Dirichlet sur les bords $,_i$, $i = 1,2$ peuvent être remplacées par une combinaison de conditions aux limites Dirichlet et Neumann, ce qui s'écrit :*

$$\begin{cases} u^{n+\frac{1}{2}} + \alpha\dfrac{\partial}{\partial\eta}u^{n+\frac{1}{2}} = u^n + \alpha\dfrac{\partial}{\partial\eta}u^n & sur \quad ,_1, \\ u^{n+1} + \alpha\dfrac{\partial}{\partial\eta}u^{n+1} = u^{n+\frac{1}{2}} + \alpha\dfrac{\partial}{\partial\eta}u^{n+\frac{1}{2}} & sur \quad ,_2. \end{cases} \tag{1.12}$$

■

Ceci nous amène à parler de l'autre variante des méthodes de décomposition de domaine dites : **Méthodes de décomposition de domaine sans recouvrement**.
En effet, quand le recouvrement entre Ω_1 et Ω_2 tend vers 0 (au sens d'une distance quelconque), c'est à dire que les deux sous-domaines n'ont qu'une interface en commun, les conditions ci-dessus (1.12) sont appelées "*conditions de transmission de type Robin ou mixtes*".

1.2.3 Version sans recouvrement de la méthode de Schwarz: Algorithme de Lions

Nous exposons l'algorithme de décomposition de domaine sans recouvrement de P.-L. Lions, qui est inspiré de la méthode originale de Schwarz ci-dessus.
Si nous considérons le même problème (1.1) et dans le cas d'une décomposition multidomaine, l'algorithme de Lions [Lions, 1990] s'écrit :

Algorithme de Lions

Initialisation: Pour tout $i : 1 \leq i \leq m$, soient u_i^0 choisis arbitrairement dans $H^2(\Omega_i)$ et s'annulant sur $\partial\Omega \cap \partial\Omega_i$.

Itération: Pour tout $n \geq 0$ et $i : 1 \leq i \leq m$, les suites $(u_i^n)_{1\leq i\leq m}$ sont calculées par :

$$\left\{\begin{array}{lcll} -\Delta u_i^{n+1} & = & f_i & \text{dans } \Omega_i, \\ u_i^{n+1} & = & 0 & \text{sur } \partial\Omega \cap \partial\Omega_i, \\ \dfrac{\partial u_i^{n+1}}{\partial \eta_{ij}} + \lambda_{ij} u_i^{n+1} & = & -\dfrac{\partial u_j^n}{\partial \eta_{ji}} + \lambda_{ji} u_j^n & \text{sur } \sigma_{ij} = \partial\Omega_i \cap \partial\Omega_j, j \neq i. \end{array}\right. \tag{1.13}$$

où $\eta_{ij} = -\eta_{ji}$ est le vecteur normal extérieur à Ω_i en la frontière non vide avec son voisin Ω_j et $\lambda_{ij} = \lambda_{ji}$ sont des réels strictement positifs.

Nous énonçons également un théorème de convergence de cet algorithme dont la démonstration est détaillée dans [Lions, 1990].

Théorème 1.2.1 *Pour tout $i : 1 \leq i \leq m$, u_i^n converge vers $u_i|_{\Omega_i}$ dans $H^1(\Omega_i)$ faiblement et en particulier, $u_i^n|_{\sigma_{ij}}$ converge vers $u|_{\sigma_{ij}}$ faiblement dans $H^{\frac{1}{2}}(\sigma_{ij})$. De plus, $\frac{1}{2}(u_i^{n+1} + u_j^n)$ converge vers $u|_{\sigma_{ij}}$ faiblement dans $H^{\frac{1}{2}}(\sigma_{ij})$, pour tout $j \neq i$.*

Remarque 1.2.2 *Dans cet algorithme de Lions, la résolution des systèmes locaux se fait en parallèle : à l'itération courante, chaque sous-domaine a besoin d'informations à l'itération précédente, de ses voisins avec lesquels il a des frontières non vides. Plus loin, nous reviendrons sur les notations de cette dernière partie.*

1.3 Rappels sur les méthodes de Lagrangien et de Lagrangien augmenté

1.3.1 Rappels et définitions

Plusieurs problèmes en physique, mécanique, économie...etc peuvent être écrits sous la forme :

$$(P) \qquad Min_{v\in V}(F(Bv) + G(v)), \tag{1.14}$$

où

- V et H sont deux espaces de Hilbert,

- B est une application linéaire de V sur H,
- F et G sont deux fonctionnelles convexes, propres, semi-continues inférieurement sur H et V respectivement.

L'idée principale de cet algorithme est basée sur l'équivalence immédiate entre le problème (P) et

$$(\tilde{P}) \qquad Min_{(v,q)\in W}(F(q)+G(v)), \tag{1.15}$$

avec $W=\{(v,q)\in V\times H, Bv-q=0\}$.
q est donc une variable supplémentaire liée à v par la contrainte linéaire $Bv=q$. On a ainsi découplé F et B et simplifié le caractère non-linéaire de (P).

Lagrangien et Lagrangien augmenté associés à $(\tilde{P})$

Soit $(v,q,\mu)\in V\times H\times H$ et on définit **le Lagrangien** $\mathcal{L}$ associé au problème $(\tilde{P})$ par

$$\mathcal{L}(v,q,\mu)=F(q)+G(v)+(\mu,Bv-q). \tag{1.16}$$

Et pour tout $r\geq 0$, **le Lagrangien augmenté** est donné par

$$\mathcal{L}_r(v,q,\mu)=F(q)+G(v)+(\mu,Bv-q)+\frac{r}{2}|Bv-q|^2. \tag{1.17}$$

Remarque 1.3.1 *Le dernier terme de la fonctionnelle $\mathcal{L}_r$ est le terme "augmenté" ou "pénalisateur"; il permet en fait, de retrouver la solution exacte du problème, quand elle existe, sans faire tendre le coefficient r vers l'infini.*

Propriétés des points-selles de $\mathcal{L}$ et $\mathcal{L}_r$

Tout d'abord, nous donnons la définition d'un point-selle du Lagrangien $\mathcal{L}$:

Définition 1.3.1 *Un élément (u,p,λ) de $V\times H\times H$ est un* **point-selle** *de $\mathcal{L}$ s'il vérifie*

$$\mathcal{L}(u,p,\mu)\leq\mathcal{L}(u,p,\lambda)\leq\mathcal{L}(v,q,\lambda), \tag{1.18}$$

$$\text{pour tout } (v,q,\mu) \text{ dans } V\times H\times H. \tag{1.19}$$

Ce qui implique,

$$\mathcal{L}(u,p,\mu)=Sup_{\mu\in H}Inf_{(v,q)\in V\times H}\mathcal{L}(v,q,\mu). \tag{1.20}$$

Le théorème suivant, dont la démonstration détaillée est dans [Fortin et Glowinski, 1982], donne la relation entre les points-selles de $\mathcal{L}$ et ceux de $\mathcal{L}_r$:

Théorème 1.3.1 *Soit (u,p,λ) un point-selle de $\mathcal{L}$ sur $V\times H\times H$, alors (u,p,λ) est point-selle de $\mathcal{L}_r$, et réciproquement. De plus, u est solution de (P) et on a $Bu=p$.*

1.3.2 Algorithmes de calcul de points-selles de $\mathcal{L}_r$

Les algorithmes utilisés pour calculer les points-selles de $\mathcal{L}$ ou de $\mathcal{L}_r$ sont connus sous le nom d'algorithmes d'Uzawa et sont dûs à Arrow, Harwicz et Uzawa [Arrow et al., 1958] et à Glowinski, Lions et Trémolières [Glowinski et al., 1976]. Les plus utilisés sont `ALG1`, `ALG2` et `ALG3` donnés dans [Fortin et Glowinski, 1982] et [Glowinski et Le Tallec, 1989] et que nous présentons ci-dessous dans le cas du problème (P).

Tout d'abord, nous donnons un premier algorithme et qui est le plus simple des algorithmes de type Uzawa pour calculer un point-selle (u, p, λ), quand il existe. Il est construit de la manière suivante:

Algorithme `ALG1`

- Initialisation: $\lambda^0 \in H$ arbitrairement donné,

- λ^n étant connu, on calcule u^n et p^n par

$$\begin{aligned} &\mathcal{L}_r(u^n, p^n, \lambda^n) \leq \mathcal{L}_r(v, q, \lambda^n), \quad \forall (v, q) \in V \times H \\ &(u^n, p^n) \in V \times H \end{aligned} \tag{1.21}$$

- Mise à jour du multipicateur de Lagrange:

$$\lambda^{n+1} = \lambda^n + \frac{\partial \mathcal{L}_r}{\partial \mu}(u^n, p^n, \lambda^n) \tag{1.22}$$

avec $\dfrac{\partial \mathcal{L}_r}{\partial \mu}(u^n, p^n, \lambda^n) = Bu^n - p^n$.

Nous présentons également un deuxième algorithme noté `ALG2`:

Algorithme `ALG2`

- Initialisation: p^0 et λ^1 choisis arbitrairement dans H,

- p^{n-1} et λ^n étant connus, on calcule successivement u^n, p^n et λ^{n+1} par

$\star$ Calcul de u^n :

$$\begin{aligned} &\mathcal{L}_r(u^n, p^{n-1}, \lambda^n) \leq \mathcal{L}_r(v, p^{n-1}, \lambda^n), \forall v \in V, \\ &u^n \in V. \end{aligned} \tag{1.23}$$

$\star$ Calcul de p^n :

$$\begin{aligned} &\mathcal{L}_r(u^n, p^n, \lambda^n) \leq \mathcal{L}_r(u^n, q, \lambda^n), \forall q \in H, \\ &p^n \in H. \end{aligned} \tag{1.24}$$

⋆ Mise à jour du multiplicateur λ^n :

$$\lambda^{n+1} = \lambda^n + \rho_n(Bu^n - p^n), \quad \rho_n > 0. \tag{1.25}$$

Les systèmes (1.23) et (1.24) de `ALG2` représentent en fait une relaxation avec un seul balayage du système (1.21).

Une variante de l'algorithme `ALG2` peut être obtenue en échangeant les rôles des variables v et q de $\mathcal{L}_r$. Pour avoir des rôles symétriques de ces deux variables, on peut faire une première mise à jour de λ entre les systèmes (1.23) et (1.24). Il en résulte l'algorithme `ALG3`:

Algorithme `ALG3`

- Initialisation: p^0 et λ^1 choisis arbitrairement dans H,

- p^{n-1} et λ^n étant connus, on calcule successivement u^n, $\lambda^{n+\frac{1}{2}}$, p^n et λ^{n+1} par

⋆ calcul de u^n :

$$\begin{array}{l} \mathcal{L}_r(u^n, p^{n-1}, \lambda^n) \leq \mathcal{L}_r(v, p^{n-1}, \lambda^n), \forall v \in V, \\ u^n \in V. \end{array} \tag{1.26}$$

⋆ 1ère mise à jour de λ :

$$\lambda^{n+\frac{1}{2}} = \lambda^n + \rho_n(Bu^n - p^{n-1}), \quad \rho_n > 0. \tag{1.27}$$

⋆ calcul de p^n :

$$\begin{array}{l} \mathcal{L}_r(u^n, p^n, \lambda^{n+\frac{1}{2}}) \leq \mathcal{L}_r(u^n, q, \lambda^{n+\frac{1}{2}}), \forall q \in H, \\ p^n \in H. \end{array} \tag{1.28}$$

⋆ 2ème mise à jour de λ :

$$\lambda^{n+1} = \lambda^{n+\frac{1}{2}} + \rho_n(Bu^n - p^n), \quad \rho_n > 0. \tag{1.29}$$

1.3.3 Résultats de convergence

Nous énonçons un théorème de convergence de l'algorithme `ALG2` dont la preuve détaillée est donnée dans [Glowinski et Le Tallec, 1989] ou [Fortin et Glowinski, 1982]. Nous donnerons après les quelques différences d'hypothèses pour les algorithmes `ALG1` et `ALG3`.

Théorème 1.3.2 *Supposons vérifiées les hypothèses suivantes :*

(i) Le Lagrangien augmenté $\mathcal{L}_r$ admet un point-selle.

(ii) La fonction $\mathcal{L}_r(.,.;\mu)$ est coercive sur $V \times H$ pour tout μ fixé, différentiable sur V pour tout q et μ fixés et différentiable sur H pour tout v et μ fixés.

(iii) Ou bien F est uniformément convexe sur les bornés de H avec B injective et ImB fermé, ou bien G est uniformément convexe sur tous les bornés de V.

Alors, si ρ_n vérifie

$$0 < \rho_n = \rho < \frac{1+\sqrt{5}}{2} r, \tag{1.30}$$

la suite (u^n, p^n, λ^n) calculée par `ALG2` *est bien définie et on a, quand n tend vers l'infini,*

$$\begin{array}{lcl} F(p^n) + G(u^n) & \longrightarrow & F(p) + G(u) \quad (= J(u)), \\ u^n & \longrightarrow & u \quad \textit{fortement dans } V, \\ p^n & \longrightarrow & p \quad \textit{fortement dans } H, \\ \lambda^n & \longrightarrow & \lambda^\star \quad \textit{faiblement dans } H, \ (u, p, \lambda^\star) \quad \textit{étant le point-selle de } \mathcal{L}_r. \end{array}$$

Remarque 1.3.2 *Les propriétés de convergence d'*`ALG1` *sont les mêmes que celles d'*`ALG2`*, avec la condition sur ρ_n (1.30) remplacée par*

$$0 < \alpha_0 \leq \rho_n \leq \alpha_1 < 2r. \tag{1.31}$$

D'autre part, avec le choix optimal $\rho_n = \rho = r$, on montre la convergence d'`ALG3`*. De plus, des tests numériques montrent qu'* `ALG3` *est moins robuste que les deux autres algorithmes, mais, il peut être plus rapide.*

■

Nous avons vu que la convergence des algorithmes de Lagrangien augmenté nécessite l'existence d'un point-selle. Plus précisément, la recherche d'un point-selle revient à chercher un multiplicateur de Lagrange et en dimension infinie, cela s'avère très délicat. Nous avons le théorème suivant qui donne des conditions suffisantes d'existence d'un multiplicateur pour $\mathcal{L}_r$ (ou $\mathcal{L}$) et dont la preuve se trouve dans [Ekeland et Temam, 1974] :

Théorème 1.3.3 *Des conditions suffisantes pour assurer l'existence d'un multiplicateur se résumeraient en :*

- *(P) admet une solution,*
- *$\exists u_0 \in V$ telque $J(u_0) < \infty$,*
- *F est continue en u_0. (C'est la condition la plus difficile à vérifier).*

Par ailleurs, en dimension finie, l'existence d'un point-selle est immédiate puisque l'on minimise une fonctionnelle sous une contrainte linéaire d'égalité. Il suffit seulement que le problème (P) admette une solution (voir [Fortin et Glowinski, 1982] pour d'autres remarques). ■

Nous présentons dans le paragraphe suivant un résultat, dû à Glowinski et Le Tallec, qui regroupe des méthodes de décomposition de domaine et de Lagrangien augmenté.

1.4 Illustration d'une MDD pour un problème elliptique avec Lagrangien augmenté

Dans [Glowinski et Le Tallec, 1990], on se propose de résoudre le problème elliptique avec des conditions aux limites de Dirichlet :

$$\left\{ \begin{array}{rcl} -\Delta u & = & f \quad \text{dans} \quad \Omega, \\ u & = & 0 \quad \text{sur} \quad \partial\Omega. \end{array} \right. \tag{1.32}$$

Lorsqu'on décompose l'ouvert borné Ω en deux sous-domaines Ω_1 et Ω_2 non-recouvrants dont on note Σ l'interface commune, on remarque que le système elliptique (1.32) peut être reformulé par le problème de minimisation suivant

$$\left\{ \begin{array}{l} \text{Minimiser} \quad \displaystyle\sum_{i=1}^{2} \left[\int_{\Omega_i} (\frac{1}{2}|\nabla v_i|^2 - f v_i) dx \right] \\ v_i \in H^1(\Omega_i), \\ v_i = 0 \quad \text{sur } \partial\Omega, \\ v_1 = q \text{ et } v_2 = q \quad \text{sur } \Sigma. \end{array} \right.$$

Un Lagrangien augmenté est associé à ce nouveau problème qui se réduit donc à la recherche d'un point-selle de

$$\begin{aligned} \mathcal{L}_r(v_i, q; \mu_i) = \sum_{i=1}^{2} \Bigg[& \int_{\Omega_i} (\frac{1}{2}|\nabla v_i|^2 - f v_i) dx + \\ & \int_{\Sigma} \mu_i (v_i - q) d\sigma + \frac{r}{2} \int_{\Sigma} |v_i - q|^2 d\sigma \Bigg] \end{aligned} \tag{1.33}$$

On utilise l'algorithme `ALG3` de la section précédente pour résoudre le nouveau problème. En éliminant les multiplicateurs de Lagrange λ_i et la variable supplémentaire q de cet algorithme, on retrouve exactement la méthode alternée de Schwarz sans recouvrement proposée par Lions [Lions, 1990].

Chapitre **2 :**

Un problème-modèle de contrôle optimal.

Dans ce chapitre, nous nous intéressons à l'étude d'un problème de contrôle optimal. Les généralités et les problèmes d'existence sont présentés. Enfin, les aspects de résolution de tels problèmes sont détaillés.
Sont également exposées des généralités sur les problèmes de contrôle optimal régis par des équations aux dérivées partielles elliptiques dont la théorie est présentée par J.-L. Lions [Lions, 1968] et nous donnons quelques résultats fondamentaux. Ensuite, pour parler des algorithmes de résolution, en général des algorithmes de descente et donc nécessitant la connaissance du gradient de la fonctionnelle à minimiser, nous allons nous consacrer au calcul exact de ce gradient. Pour cela, nous introduisons un état adjoint par le biais d'une formulation Lagrangienne du problème de contrôle que nous étudions.

2.1 Position du problème

2.1.1 Qu'est-ce qu'un problème de contrôle ?

Dans un premier temps, nous allons donner une définition du problème auquel nous allons nous intéresser dans ce rapport : **Problème de contrôle optimal**. Une manière de le formuler est la suivante :

Etant donnée une observation y_d, peut-on trouver un contrôle admissible u tel que $y(u) = y_d$ où $y(u)$ est la solution de l'équation d'état du problème dépendant du contrôle?

La formulation du problème de contrôle optimal repose sur une fonctionnelle J à valeurs réelles dite **fontion coût** ou **critère** qui minimise l'écart entre $y(u)$ et y_d au sens de la norme L^2, à laquelle on ajoute un terme qui assure la régularité du contrôle et l'unicité de la solution.

Concrètement, le problème est donné par :

Trouver $u \in \mathcal{U}_{ad}$ tel que :

$$J(u) = \inf_{v \in \mathcal{U}_{ad}} J(v)$$

avec J définie par:

$$J(v) = \frac{1}{2}\left(\int_\Omega |y(v) - y_d|^2 dx + \nu \int_\Omega |v|^2 dx\right)$$

C'est un problème de minimisation. Dans le paragraphe suivant, on en donnera les propriétés de base, qui nous seront utiles dans toute la suite de cette étude.

2.1.2 Des notions et résultats théoriques en optimisation

Dans ce paragraphe, nous citons quelques résultats sur l'existence, l'unicité des solutions ainsi que leurs caractérisations. Nous ne démontrons pas tous ces résultats et pour plus de détails, nous renvoyons le lecteur aux références suivantes [Céa, 1971] et [Faurre, 1986].
Alors, soit un problème d'optimisation dans son cadre plus général :

$$J(u) = \inf_{v \in K} J(v), \quad u \in K \tag{2.1}$$

où K est un convexe fermé non vide d'un espace de Hilbert V sur $\mathbb{R}$ et J est une fonctionnelle définie sur K à valeurs dans $\mathbb{R}$.
Tout d'abord, il faut noter l'importance de l'hypothèse de convexité de la fonctionnelle J en optimisation. En effet, nous avons la proposition :

Proposition 2.1.1 *Si la fonctionnelle J est convexe, alors tout minimum local est un minimum global. Si de plus, J est strictement convexe, le minimum, quand il existe, est unique.*

2.1.3 Sur l'existence d'un optimum

Le théorème suivant donne un résultat général sur l'existence de minima :

Théorème 2.1.1 *Si la fonctionnelle J est convexe, semi-continue inférieurement et si $\lim_{\|v\| \to +\infty} J(v) = +\infty$ ou bien le convexe fermé K est borné, alors J admet un minimum.*

Avant de donner les caractérisations des optima, nous rappelons les définitions suivantes :

Définition 2.1.1 *La dérivée de la fonctionnelle J au point v dans la direction w est donnée par :*

$$J'(v,w) = \lim_{\theta \to 0} \frac{J(v+\theta w) - J(v)}{\theta} \tag{2.2}$$

Définition 2.1.2 *On dira que J est Gâteaux-dérivable si $w \mapsto J'(v,w)$ de la définition.2.1.1 est linéaire continue. Dans ce cas, la Gâteaux-dérivée de J est définie par :*

$$(J'(v), w) = J'(v,w) \tag{2.3}$$

Définition 2.1.3 *Si on a :*

$$J(v+w) = J(v) + (J'(v), w) + o(w) \ \text{où} \ \lim_{||w|| \to 0} \frac{o(w)}{||w||} = 0, \tag{2.4}$$

on dira que J est Fréchet-dérivable et de Fréchet-dérivée $J'(v)$ au point v.

Par ailleurs, les Fréchet et Gâteaux-dérivées sont liées par :

Proposition 2.1.2 *La Fréchet-dérivabilité entraine la Gâteaux-dérivabilité.*
La réciproque est vraie à condition que la Gâteaux-dérivée $J'(v)$ soit continue en v.

On va donc pouvoir énoncer un théorème important en optimisation convexe :

Théorème 2.1.2 *Soit $J : K \longrightarrow \mathbb{R}$ une fonctionnelle convexe et Gâteaux-dérivable. Un élément u de K est un minimum de J si et seulement si :*

$$\begin{cases} (J'(u), v-u) & \geq 0 \quad \forall v \in K \quad \textit{cas avec contraintes : } \underline{\textit{Inéquation d'Euler}} \\ & \textit{ou} \\ J'(u) & = 0 \quad K = V \quad \textit{cas sans contraintes : } \underline{\textit{Equation d'Euler}} \end{cases}$$

2.2 Problème-modèle de contrôle optimal

2.2.1 Exemple: Contrôle distribué

Pour simplifier cette étude et pour ne pas avoir de lourdes notations, nous considérons un problème de contrôle optimal distribué régi par une équation aux dérivées partielles elliptique et donné par :

$$(\mathcal{P}) \qquad J(u) = \inf_{v \in \mathcal{U}_{ad}} J(v), \quad u \in \mathcal{U}_{ad} \tag{2.5}$$

où

$$J(v) = \frac{1}{2}\left(\int_\Omega |y(v) - y_d|^2 dx + \nu \int_\Omega |v|^2 dx\right),$$

- Ω est un ouvert convexe borné de $\mathbb{R}^q$ ($q = 2$ ou 3),
- , est la frontière de Ω supposée suffisamment régulière,
- $\mathcal{U}_{ad}$, l'espace des contrôles admissibles, est un convexe fermé de $L^2(\Omega)$,
- y_d l'état désiré, est pris dans $L^2(\Omega)$,
- ν est un réel strictement positif.
- Pour $f \in L^2(\Omega)$, $y(v)$ est la solution de l'équation d'état :

$$(\mathcal{D}ir) \qquad \begin{cases} -\Delta y(v) = f + v & \text{dans} \quad \Omega, \\ y(v) = 0 & \text{sur} \quad , = \partial\Omega. \end{cases} \tag{2.6}$$

Proposition 2.2.1 *Le système $(\mathcal{D}ir)$ admet une solution unique $y(v)$. De plus, l'application $u \mapsto y(u)$ est affine continue de $\mathcal{U}_{ad}$ sur $H^1(\Omega)$.*

Proposition 2.2.2 *Le problème $(\mathcal{P})$ admet une solution unique $u \in \mathcal{U}_{ad}$.*

Preuve : On montre facilement que la fonctionnelle J est continue et strictement convexe. De plus, $\mathcal{U}_{ad}$ est un convexe fermé supposé non vide de $L^2(\Omega)$.
D'autre part, $\lim_{\|v\| \to +\infty} J(v) = +\infty$, alors, d'après le théorème 2.1.1, J admet un minimum unique $u \in \mathcal{U}_{ad}$.

2.2.2 Interprétation Lagrangienne

On remarque que la fonctionnelle J dépend explicitement de v par l'intermédiaire du terme de régularisation et implicitement par le premier terme "$y(v)$".
On peut donc considérer v et y comme indépendants en assimilant les équations liant v et y à des contraintes d'un type particulier. Ainsi, on peut poser $J(v) \overset{déf}{=} J(v, y(v)) = J(v, y)$ de telle sorte que J opère sur $\mathcal{U}_{ad} \times H^1(\Omega)$.
Et le problème $(\mathcal{P})$ s'écrit autrement, soit

$$(\mathcal{P}bis) \qquad \begin{cases} \text{Minimiser} \quad J(v,y) \\ \\ v \in \mathcal{U}_{ad},\ y \in H^1(\Omega), \\ y \text{ et } v \text{ liés par } (\mathcal{D}ir). \end{cases} \tag{2.7}$$

Nous sommes donc devant un problème d'optimisation dont les variables sont v et y soumis à la contrainte donnée par $(\mathcal{D}ir)$ à laquelle nous allons associer un multiplicateur de Lagrange. On associe alors à $(\mathcal{P}bis)$ le Lagrangien $\mathcal{L}$ défini par :
$\mathcal{L} : \mathcal{U}_{ad} \times H^1(\Omega) \times H^1(\Omega) \longrightarrow \mathbb{R}$ et tel que :

$$\mathcal{L}(v,z,q) = J(v,z) - a(z,q) + (f,q) + \int_\Omega vq dx$$

On rappelle donc la définition d'un point-selle de $\mathcal{L}$:

Définition 2.2.1 *(u,y,p) est un point-selle de $\mathcal{L}$ si et seulement si :*

$$\mathcal{L}(u,y,q) \leq \mathcal{L}(u,y,p) \leq \mathcal{L}(v,z,p) \quad \forall v \in \mathcal{U}_{ad},\ \forall (z,q) \in (H^1(\Omega))^2.$$

La proposition suivante permet de ramener le problème $(\mathcal{P})$ à la recherche d'un point-selle du Lagrangien $\mathcal{L}$:

Proposition 2.2.3 *Si (u,y,p) est un point-selle de $\mathcal{L}$, alors, u est la solution du problème $(\mathcal{P})$.*

Preuve : Soit (u,y,p) un point-selle de $\mathcal{L}$, il vérifie donc

$$\mathcal{L}(u,y,q) \ \leq \ \mathcal{L}(u,y,p) \ \leq \ \mathcal{L}(v,z,p) \qquad \forall v \in \mathcal{U}_{ad},\ \forall (z,q) \in (H^1(\Omega))^2. \tag{2.8}$$

Dans l'inégalité de gauche de (2.8), si nous remplaçons $\mathcal{L}(.,.,.)$ par son expression, nous obtenons

$$J(u,y) - a(y,q) + (f,q) + \int_\Omega vq dx \leq J(u,y) - a(y,p) + (f,p) + \int_\Omega up dx.$$

Soit,

$$a(y,q-p) - (f,q-p) + \int_\Omega u(q-p)dx \leq 0 \qquad \forall q \in H^1(\Omega).$$

En posant successivement $q = p + \tilde{q}$ et $q = p - \tilde{q}$ où $\tilde{q}$ est quelconque dans $H^1(\Omega)$, nous obtenons

$$a(y,\tilde{q}) = (f,\tilde{q}) + \int_\Omega u\tilde{q}dx \leq 0 \text{ et } a(y,\tilde{q}) = (f,\tilde{q}) + \int_\Omega u\tilde{q}dx \geq 0.$$

Donc y est la solution de $(\mathcal{D}ir)$.

D'autre part, de l'inégalité de droite de (2.8), il vient

$$J(u,y) - a(y,p) + (f,p) + \int_\Omega updx \leq J(v,z) - a(z,p) + (f,p) + \int_\Omega vpdx.$$

Comme y est solution de $(\mathcal{D}ir)$, il reste alors

$$J(u,y) \ \leq \ J(v,z) - a(z,p) + (f,p) + \int_\Omega vpdx, \qquad \forall (v,z) \in \mathcal{U}_{ad} \times H^1(\Omega). \tag{2.9}$$

Ce qui est équivalent à

$$\begin{aligned} &J(u,y) \ \leq \ J(v,z) \\ &\forall (v,z) \in \mathcal{U}_{ad} \times H^1(\Omega) \text{ tel que } \ a(z,p) = (f,p) + \int_\Omega vpdx, \end{aligned} \tag{2.10}$$

ou encore

$$\begin{aligned} &J(u,y) \ \leq \ J(v,z) \\ &\forall (v,z) \in \mathcal{U}_{ad} \times H^1(\Omega) \quad \text{tel que } \ v \text{ et } z \text{ liés par } \ (\mathcal{D}ir). \end{aligned} \tag{2.11}$$

2.2.3 Caractérisation d'un point-selle de $\mathcal{L}$

Comme $\mathcal{L}$ est différentiable et convexe par rapport à chacune de ses variables v et z, (2.8) est équivalente à

$$(i) \qquad \left(\frac{\partial \mathcal{L}}{\partial q}(v,z,q)|_{(u,y,p)}, \ \varphi \right) \ = \ 0, \qquad \forall \varphi \in H^1(\Omega), \tag{2.12}$$

$$(ii) \qquad \left(\frac{\partial \mathcal{L}}{\partial z}(v,z,q)|_{(u,y,p)}, \ \varphi \right) \ = \ 0, \qquad \forall \varphi \in H^1(\Omega), \tag{2.13}$$

$$(iii) \ \left(\frac{\partial \mathcal{L}}{\partial v}(v,z,q)|_{(u,y,p)}, \ v-u \right) \ \geq \ 0, \qquad \forall v \in \mathcal{U}_{ad}. \tag{2.14}$$

Ce qui donne les systèmes suivants:

$$
\begin{cases}
(i)\ \underline{\textbf{Etat direct}} & \begin{cases} -\Delta y(u) = f+u & \text{dans } \Omega, \\ y(u) = 0 & \text{sur } ,, \end{cases} \\
(ii)\ \underline{\textbf{Etat adjoint}} & \begin{cases} -\Delta p(u) = y(u)-y_d & \text{dans } \Omega, \\ p(u) = 0 & \text{sur } ,, \end{cases} \\
(iii)\ \underline{\textbf{Condition d'optimalité}} & \int_\Omega (p(u)+\nu u)(v-u)dx \geq 0, \quad \forall v \in \mathcal{U}_{ad} \\
& \text{ou} \qquad p+\nu u = 0 \text{ cas sans contraintes}
\end{cases} \tag{2.15}
$$

Les équations (2.15) forment le "**système d'optimalité**" du problème de contrôle optimal $(\mathcal{P})$.

Proposition 2.2.4 *Pour que (u,y,p) soit un point-selle de $\mathcal{L}$ sur $\mathcal{U}_{ad}\times V\times V$, il faut et il suffit qu'il soit solution du système* (2.15).

Preuve : La condition nécessaire est donnée par la caractérisation ci-dessus. la réciproque repose sur la convexité du Lagrangien $\mathcal{L}$ en v et z.

Remarque 2.2.1 *L'inéquation (resp. l'équation) (iii) du système* (2.15) *n'est autre que l'inéquation (resp. l'équation) d'Euler relative au problème d'optimisation* $(\mathcal{P})$.

Ce qui nous amène a énoncer le lemme suivant qui donne le calcul direct du gradient de la fonction coût J en u :

Lemme 2.2.1 *Le gradient de la fonction coût J en u est donné par*

$$J'(u) = p+\nu u. \tag{2.16}$$

Soit u et v dans $L^2(\Omega)$, comme $y(u)$ est une fonction affine continue de u, on a

$$(J'(u), v-u) = \int_\Omega (y(u)-y_d)(y(v)-y(u))dx + \nu\int_\Omega u(v-u)dx. \tag{2.17}$$

Nous allons réécrire le premier terme de cette égalité. Pour cela, nous utilisons les formulations variationnelles des états direct et adjoint du système d'optimalité (2.15), c'est à dire,

$$a(y(u),\varphi) = (f+u,\varphi), \quad \forall\varphi\in V, \tag{2.18}$$

$$a(p,\phi) = (y(u)-y_d,\phi), \quad \forall\phi\in V. \tag{2.19}$$

On pose $\varphi = p$ dans (2.18) pour $y(u)$ et $y(v)$ et on fait la différence, il résulte

$$a(y(v)-y(u),p) = (v-u,p). \tag{2.20}$$

Ensuite, on prend $\phi = y(v) - y(u)$ dans (2.19), il vient

$$a(y(v) - y(u), p) = (y(u) - y_d, y(v) - y(u)). \tag{2.21}$$

Donc,

$$(J'(u), v - u) = (p + \nu u, v - u) \tag{2.22}$$

Remarque 2.2.2

$$\frac{\partial}{\partial v}\mathcal{L}(v, z, q)|_{(u,y,p)} \;=\; J'(u) = \; p + \nu u \tag{2.23}$$

Remarque 2.2.3 *Dans le cas sans contraintes, c'est à dire, $\mathcal{U}_{ad} = L^2(\Omega)$, le contrôle optimal est donné par :*

$$u = -\frac{p}{\nu}$$

On obtient alors un système couplé en y et p.

2.3 Résolution numérique et calcul du gradient

Nous avons vu que l'approche Lagrangienne permet de ramener le problème de contrôle optimal à une recherche de point-selle. Cette approche permet également de calculer l'expression du gradient de la fonctionnelle J à partir de l'état adjoint (ou le multiplicateur de Lagrange associé à la contrainte donnée par l'équation d'état). Nous développons ainsi numériquement un algorithme rapide de calcul de gradient de la fonctionnelle coût J à minimiser. Indépendamment de la nature de l'algorithme d'optimisation choisi, nous résumons les étapes de calcul de la fonction coût en un point (ou un contrôle) u^n : $J(u^n)$ ainsi que son gradient $\nabla J(u^n)$ dans le diagramme de la fugure 2.1.

2.3.1 Algorithmes de minimisation

Pour résoudre les problèmes de contrôle optimal, on a besoin d'algorithmes efficaces de minimisation. Ces algorithmes reposent très souvent sur les méthodes de descente.
La fonctionnelle à minimiser dans les problèmes de contrôle que nous allons considérer, étant quadratique, un premier algorithme très efficace est l'algorithme de Gradient Conjugué.

Algorithme de gradient conjugué

Si on considère le cas simple de fonction quadratique J définie par :

$$J(x) = \frac{1}{2}(Ax, x) - (b, x) \tag{2.24}$$

où A est une matrice $M \times M$ définie positive et $b \in \mathbb{R}^M$, le problème de minimisation étant donné par :

$$\inf_{x \in \mathbb{R}^M} J(x) = J(x^*), \quad x^* \in \mathbb{R}^M. \tag{2.25}$$

L'algorithme de gradient conjugué s'écrit :

Etape 0 : **Initialisation**

$$\begin{aligned} &x^0 \in \mathbb{R}^M \text{donné arbitrairement,} \\ &g^0 = Ax^0 - b \\ &d^0 = g^0 \end{aligned}$$

Puis , pour $n \geq 0$, x^n, g^n et d^n connus, on calcule x^{n+1}, g^{n+1} et d^{n+1} par :

Etape 1 : **Descente**

$$\left\{ \begin{array}{l} \text{Trouver } \rho_n \in \mathbb{R} \text{ tel que} \\ J(x^n - \rho_n d^n) \leq J(x^n - \rho d^n), \ \forall \rho \in \mathbb{R} \end{array} \right. \tag{2.26}$$

$$\begin{aligned} x^{n+1} &= x^n - \rho_n d^n \\ g^{n+1} &= Ax^{n+1} - b \end{aligned}$$

Etape 2 : **Nouvelle direction de descente**

$$d^{n+1} = g^{n+1} + \gamma_n d^n \tag{2.27}$$

$$\text{où} \quad \gamma_n = \frac{(g^n, d^n)}{(Ad^n, d^n)} \tag{2.28}$$

Faire $n = n + 1$ et aller à (2.26)

Remarque 2.3.1 *La méthode de Gradient Conjugué converge en un nombre fini d'itérations (au plus M itérations dans le cas de* (2.24)*).*

■

Nous présentons maintenant une autre classe de méthodes de minimisation :

Méthode de quasi-Newton

Le principe des méthodes de quasi-Newton est d'utiliser les variations du gradient de la fonctionnelle à minimiser entre deux itérés successifs; on obtient donc de l'information

sur le Hessien de cette fonctionnelle.
En général, une méthode de quasi-Newton est basée sur l'itération du processus :

$$\begin{cases} x^{k+1} = x^k - \rho_k H_k^{-1} g_k \\ \\ H_{k+1}^{-1} = \mathcal{A}(H_k^{-1}, y_k, s_k) \end{cases} \tag{2.29}$$

où $y_k = g_{k+1} - g_k$, $g_k = \nabla J(x_k)$ est la différence des gradients entre deux itérés successifs, $s_k = x_{k+1} - x_k$ est le pas en x, ρ_k est un pas calculé par minimisation de J dans la direction $d_k = -H_k^{-1} g_k$ et $\mathcal{A}$ est un algorithme de mise à jour de H_k^{-1} en fonction de y_k et s_k, où H_k est le Hessien de la fonctionnelle J au point x_k: $H_k = \nabla^2 J(x_k)$.
La formule BFGS est l'une des méthodes de mise à jour qui donne de bons résultats :

$$H_{k+1} = H_k + \frac{y_k \otimes y_k}{\langle y_k, s_k \rangle} - \frac{(H_k s_k) \otimes (H_k s_k)}{\langle H_k s_k, s_k \rangle}$$

en ayant utilisé la notation :

$$u \otimes v : \mathbb{R}^n \longrightarrow \mathbb{R} \text{ et } w \mapsto (u \otimes v)w = \langle v, w \rangle u$$

avec $\langle ., . \rangle$ est le produit scalaire ordinaire. Ces formules ont la propriété de conserver la définie-positivité de H_k si et seulement si $\langle y_k, s_k \rangle$ est positif.

Pour nos tests numériques, nous allons utiliser une implémentation de cette méthode réalisée dans le cadre du projet `MODULOPT` de l'INRIA par J.-C. Gilbert et C. Lemaréchal [Gilbert et Lemaréchal, 1989].

Conclusion

Après avoir posé et étudié cet exemple de problème de contrôle, on va voir dans le chapitre qui suit comment on va le reformuler une fois ayant décomposé le domaine Ω.

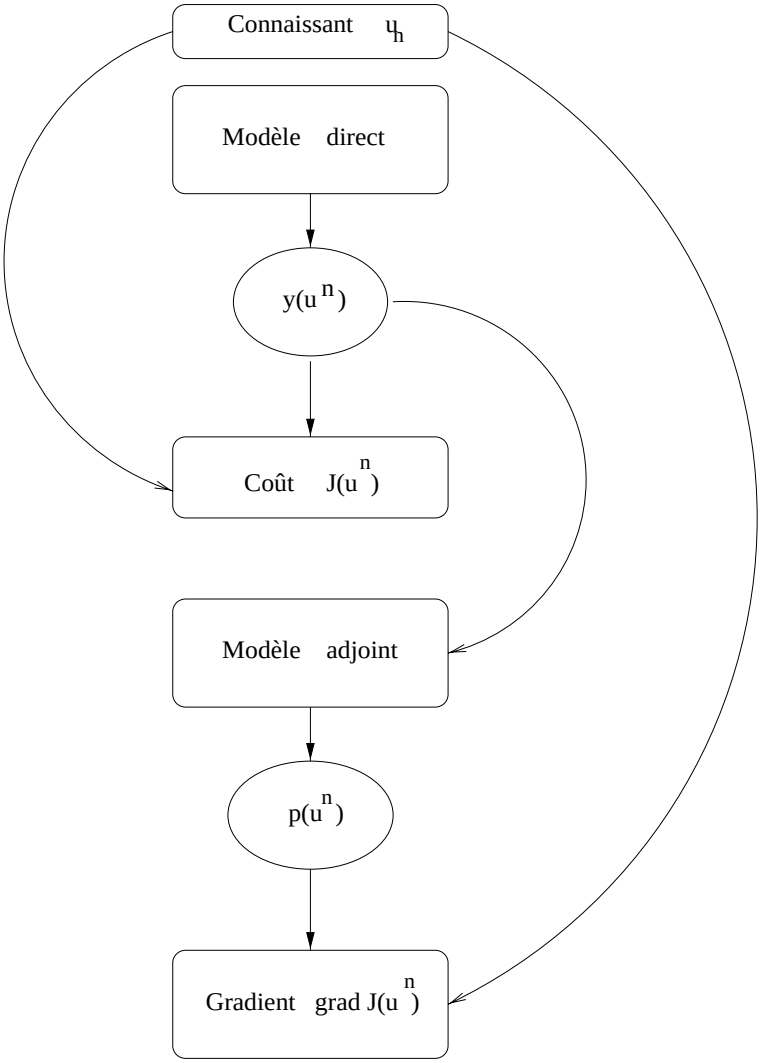

FIG. 2.1 – *Schéma de calcul du gradient de la fonction coût J.*

Chapitre **3 :**

Méthode de décomposition de domaine pour le problème de contrôle optimal.

Nous développons dans ce chapitre une méthode de décomposition de domaine **sans recouvrement** et nous donnons une nouvelle approche par l'intermédiaire d'un Lagrangien augmenté que nous associons au problème de contrôle étudié au chapitre précédent sur le domaine décomposé. Pour la résolution de notre nouveau problème, des algorithmes sont mis au point, étudiés et analysés.

Motivation

Comme nous l'avons vu au premier chapitre, les méthodes de décomposition de domaine permettent désormais, moyennant l'utilisation d'ordinateurs parallèles, de résoudre des problèmes de plus en plus complexes issus de la mécanique des fluides, de la météorologie et plus généralement des problèmes modélisés par des équations aux dérivées partielles (EDP). Dans ce chapitre, nous évoquons les problèmes de contôle optimal relevant du domaine de la météo ou l'océanographie qui demandent à l'heure actuelle de considérables temps de calcul sur des machines même très rapides telles que le `CRAY`.
Pour l'étude des problèmes de contrôle optimal gouvernés par des EDP, notre motivation repose donc sur le profit que nous pouvons tirer des propriétés de ces méthodes et sur la possibilité d'utiliser des machines multiprocesseurs dont le progrès et les développements sont en forte croissance.
Avant d'entamer cette étude, nous présentons ce qui a été déjà fait dans ce domaine.

3.1 Rappels bibliographiques

Quant à l'application des méthodes de décomposition de domaine aux problèmes de contrôle optimal régis par des équations aux dérivées partielles, plus spécialement des problèmes stationnaires, on passe en revue des travaux ayant été faits jusqu'à présent :

Tout d'abord, ce furent Bensoussan, Glowinski et Lions [Bensoussan et al., 1973], les premiers en 1973 à proposer une méthode basée sur des techniques de décomposition-coordination:
Procédant par une décomposition sans recouvrement, la résolution du problème initial est ramenée à la résolution d'une suite de sous-problèmes de même nature sur les sous-domaines moyennant une formulation par un "*Lagrangien généralisé*". Un algorithme a été proposé et des résultats ont été obtenus dans le cas d'une décomposition en deux sous-domaines et la généralisation à une décomposition en plusieurs sous-domaines s'en déduisait automatiquement (formellement).
A partir de 1994, J.-D. Benamou [Benamou, 1994, Benamou, 1996] a repris l'étude des problèmes de contrôle optimal avec une décomposition du domaine non recouvrante. Il a imposé sur les interfaces entre les sous-domaines, les conditions de transmission introduites par P.-L. Lions dans [Lions, 1988, Lions, 1990]. Ces conditions ont été prises en compte dans les systèmes locaux, ce qui a permis de réduire le problème sur le domaine global à une suite de sous-problèmes de contrôle optimal locaux *couplés en état direct-état adjoint*.
Cette même idée a été appliquée par Benamou au contrôle optimal de l'équation de Helmholtz en collaboration avec Després [Benamou et Després, 1997], la preuve de convergence de l'algorithme proposé étant inspirée de la technique exposée par Després dans [Després, 1991] et [Després, 1993].

D'autre part et dans un autre contexte, rappelons que les problèmes de contrôle optimal relèvent d'une catégorie de problèmes d'optimisation avec ou sans contraintes. Plus précisèment, nous minimisons une fonctionnelle sous la contrainte d'une équation aux dérivées partielles. A l'instar de cette précision, nous mentionnons qu'un travail très détaillé a été effectué par X.-C. Tai [Tai, 1995a, Tai, 1995b] pour l'application des méthodes de décomposition de domaine à une classe générale de problèmes d'optimisation. Ces méthodes sont basées surtout sur une décomposition de la fonctionnelle à minimiser ($J(v) = \sum_{i=1}^{m} J_i(v)$) ou bien sur la décomposition de l'espace de minimisation ($V = \sum_{i=1}^{m} V_i$) sous certaines hypothèses à vérifier. On trouve aussi plus particulièrement dans les travaux de M. Espedal et X.-C. Tai [Espedal et Tai, 1997] et K. Kunisch et X.-C. Tai [Kunisch et Tai, 1997] une application de ces décompositions d'espace ou de la fonctionnelle, aux problèmes inverses.

Plus récemment, en 1998, J.-L. Lions a repris l'idée des méthodes de décomposition de domaine pour les problèmes de contrôle optimal dans une série de travaux: il développe une méthode de décomposition de domaine avec recouvrement basée sur une nouvelle formulation en introduisant des termes de pénalisation.
Plus précisément, le terme de pénalisation intervient au niveau de la formulation variationnelle de l'équation d'état du problème et on montre ensuite que quand le coefficient de pénalisation tend vers zéro, on obtient à convergence, la solution globale à partir des solutions locales. La contrainte qu'il faut assurer est celle de la continuité des contrôles

sur les frontières des recouvrements entre les sous-domaines, voir [Lions, 1998a]. Dans le même contexte, il donne une approche de la stabilisation et du contrôle de systèmes distribués par des algorithmes parallèles: il introduit un nouvel algorithme SPA (Stabilization Parallel Algorithm) [Lions, 1998c] ainsi que son application aux systèmes hyperboliques et aux systèmes de Petrovsky [Lions, 1998b].

D'autre part, dans [Lions et Pironneau, 1998a] et [Lions et Pironneau, 1998b], Lions et Pironneau proposent, en considérant toujours une décomposition de domaine sans recouvrement, une autre idée basée cette fois sur l'introduction de multiplicateurs dans la formulation variationnelle des systèmes directs restreints à chaque sous-domaine. Ces multiplicateurs ou "contrôles virtuels" sont ensuite pris en compte dans le nouveau problème décomposé par un terme de régularisation et on minimise ensuite, simultanément par rapport aux contrôles (les vrais et les virtuels). Pour cela, on suppose que l'existence des multiplicateurs assure la continuité des états sur les bords (distincts du bord du domaine global) des recouvrements entre les sous-domaines.
Et enfin, dans [Leugering, 1997a], [Leugering, 1997b] et [Lagnese et Leugering, 1998], nous trouvons des travaux de Lagnese et Leugering qui peuvent être directement reliés aux méthodes de décomposition de domaine et aux problèmes de contrôle optimal.

■

Nous précisons maintenant les notations et les résultats préliminaires qui nous seront utiles dans toute la suite.

3.2 Préliminaires et notations

Soit Ω un domaine ouvert borné de $\mathbb{R}^d$ $(d \geq 2)$, on va considérer une décomposition sans recouvrement de Ω en m sous-domaines; c'est à dire que les sous-domaines ne communiquent que par leur éventuelle interface commune, voir (fig. 3.1).

$$\Omega = \bigcup_{i=1}^{m} \Omega_i \cup \Sigma, \quad , _i = \partial\Omega \cap \partial\Omega_i, \ i = 1, ..., m, \tag{3.1}$$

$$\sigma_{ij} = \partial\Omega_i \cap \partial\Omega_j, \ i = 1, ..., m, \ j \neq i, \quad \Sigma = \bigcup_{1 \leq i \neq j \leq m} \sigma_{ij}, \tag{3.2}$$

$$I = \{i \in \mathbb{N} \ et \ 1 \leq i \leq m\}. \tag{3.3}$$

avec les définitions suivantes:

- Ω_i : le sous-domaine i étant un ouvert de $\mathbb{R}^d$ et les Ω_i sont disjoints deux à deux,
- σ_{ij} : l'interface (resp. les interfaces) entre Ω_i et Ω_j s'il a un voisin (resp. si le domaine i a plusieurs voisins) étant un sous-ensemble de $\mathbb{R}^{d-1}$ et Σ est la réunion de toutes les interfaces,

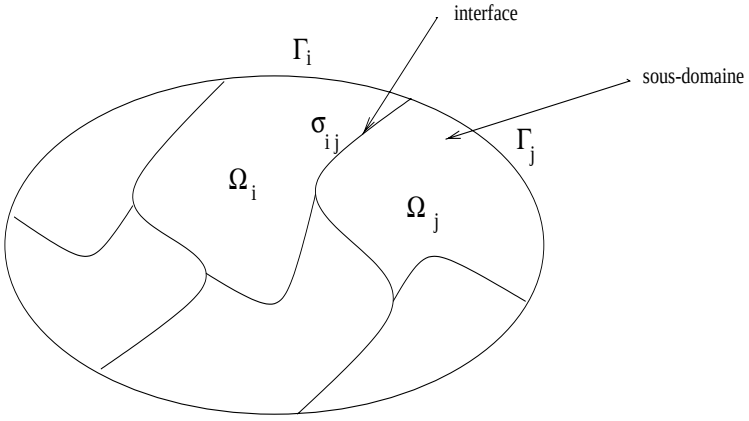

FIG. 3.1 – *Domaine de* $\mathbb{R}^2 \quad (m = 6)$.

- $,_i$: est le bord de Ω_i commun avec le bord du domaine global.

Nous rappelons que le problème de contrôle étudié dans le chapitre précédent est donné par:

$$(\mathcal{P}) \qquad \inf_{v \in \mathcal{U}_{ad}} J(v), \quad u \in \mathcal{U}_{ad} \tag{3.4}$$

où

$$J(v) = \frac{1}{2}\left(\int_\Omega |y(v) - y_d|^2 dx + \nu \int_\Omega |v|^2 dx\right)$$

et $y(v)$ est la solution du système :

$$(\mathcal{D}ir) \qquad \begin{cases} -\Delta y(v) &= f + v \quad \text{dans} \quad \Omega, \\ \\ y(v) &= 0 \qquad \text{sur} \quad , = \partial\Omega. \end{cases} \tag{3.5}$$

Dans un premier temps, nous avons la proposition suivante qui donne les conditions à respecter afin d'avoir une équivalence entre la solution de (3.5) sur le domaine global et ses restrictions sur les sous-domaines resultant de la décomposition :

Proposition 3.2.1 *La restriction sur le sous-domaine Ω_i de y, la solution de l'équation d'état, est égale à la fonction $y_i \in H^1(\Omega_i)$ si et seulement si les y_i vérifient :*
pour tout $i \in I$,

$$\begin{cases} -\Delta y_i &= f_i + v_i \quad dans \quad \Omega_i, \\ \\ y_i &= 0 \qquad\quad sur \quad , _i, \end{cases} \tag{3.6}$$

en imposant les conditions de continuité aux interfaces :

pour $i,\ j \in I,\ i \neq j$ et $\sigma_{ij} \neq \emptyset$,

$$y_i = y_j \quad sur \quad \sigma_{ij}, \qquad j \neq i \tag{3.7}$$

$$\frac{\partial y_i}{\partial \eta_{ij}} = -\frac{\partial y_j}{\partial \eta_{ji}} \quad sur \quad \sigma_{ij}, \qquad j \neq i \tag{3.8}$$

où f_i et v_i sont les restrictions respectives de f et v sur Ω_i et les v_i étant fixés et η_{ij} est la normale extérieure au sous-domaine Ω_i en σ_{ij}, nous avons $\eta_{ij} = -\eta_{ji}$.

Preuve : Le passage du domaine global aux sous-domaines est évident.
D'autre part, soient y_i, $i = 1, ..., m$, les solutions locales telles que les contraintes de continuité (3.7) sont vérifées. Alors, il suffit de poser :

$$y(x) = \sum_{i=1}^{m} y_i(x)\chi_i(x), \tag{3.9}$$

où, χ_i est la fonction caractéristique de Ω_i. Puis, il est clair que y est bien dans $H^1(\Omega)$.

Remarque 3.2.1 *Il faut noter que les deux équations* (3.7) *et* (3.8) *assurent "* **le bon raccordement** *" de la solution sur les interfaces. Autrement dit, elles assurent la continuité de la fonction y solution du système direct sur les interfaces σ_{ij} ainsi que la continuité du flux $\dfrac{\partial y}{\partial \eta_{ij}}$.*

3.3 Remarques sur les conditions de transmission

Pour bien illustrer les conditions de transmission, examinons le cas d'une décomposition en deux sous-domaines.
Soit $y \in H^1(\Omega)$ la solution de $(\mathcal{D}ir)$ et considérons les notations de la section 3.2 dans le cas où $m = 2$ ce qui revient à $I = \{1, 2\}$.
Prenons maintenant $y_1 \in H^1(\Omega_1)$, $y_2 \in H^1(\Omega_2)$, alors, d'après la proposition 3.2.1, pour que $y_1 = y|_{\Omega_1}$ et $y_2 = y|_{\Omega_2}$, il faut et il suffit que :

$$\begin{cases} -\Delta y_1 &= f_1 + v_1 \quad \text{dans} \quad \Omega_1, \\ y_1 &= 0 \qquad\quad \text{sur} \quad \ , _1, \end{cases} \tag{3.10}$$

$$\begin{cases} -\Delta y_2 &= f_2 + v_2 \quad \text{dans} \quad \Omega_2, \\ y_2 &= 0 \qquad\quad \text{sur} \quad \ , _2, \end{cases} \tag{3.11}$$

en y ajoutant les conditions sur l'interface Σ:

$$y_1 = y_2 \quad \text{sur} \quad \Sigma, \tag{3.12}$$

$$\frac{\partial y_1}{\partial \eta_{12}} = \frac{\partial y_2}{\partial \eta_{12}} \quad \text{sur} \quad \Sigma. \tag{3.13}$$

Nous précisons que pour $i = 1, 2$, $y_i|_{\ \Sigma}$ et $\dfrac{\partial y_i}{\partial \eta_{ij}}$ $(i \neq j)$ sont respectivement dans $H^{\frac{1}{2}}(\Sigma)$ et $H^{-\frac{1}{2}}(\Sigma)$, voir [Lions et Magenes, 1968]).

Remarque 3.3.1 *En effectuant une combinaison linéaire des conditions limites aux interfaces* (3.12), (3.13)*, on obtient de nouvelles conditions qui lui sont équivalentes à condition que les paramètres α_i, β_i ci-dessous vérifient $\alpha_1\beta_2 \neq \alpha_2\beta_1$. Ainsi, les états directs locaux y_i, $i = 1, 2$, deviennent solutions respectives de :*

$$\begin{cases} -\Delta y_1 &= f_1 + v_1 \quad \textit{dans} \quad \Omega_1, \\ y_1 &= 0 \qquad\quad \textit{sur} \quad \ , _1, \\ \alpha_1 y_1 + \beta_1 \dfrac{\partial y_1}{\partial \eta_{12}} = \alpha_1 y_2 - \beta_1 \dfrac{\partial y_2}{\partial \eta_{21}} & \textit{sur} \quad \Sigma, \end{cases} \tag{3.14}$$

$$\begin{cases} -\Delta y_2 &= f_2 + v_2 \quad \textit{dans} \quad \Omega_2, \\ y_2 &= 0 \qquad\quad \textit{sur} \quad \ , _2, \\ \alpha_2 y_2 + \beta_2 \dfrac{\partial y_2}{\partial \eta_{21}} = \alpha_2 y_1 - \beta_2 \dfrac{\partial y_1}{\partial \eta_{12}} & \textit{sur} \quad \Sigma. \end{cases} \tag{3.15}$$

Remarque 3.3.2 *Les conditions ci-dessus sont dites* **des conditions de transmission mixtes** *et peuvent s'écrire dans le cas d'une décomposition en plusieurs sous-domaines, avec k un réel strictement positif:*

$$\begin{cases} \dfrac{\partial y_i}{\partial \eta_{ij}} + k y_i &= \dfrac{\partial y_j}{\partial \eta_{ij}} + k y_j \quad \textit{sur} \quad \sigma_{ij}, \quad j \neq i, \\ \dfrac{\partial y_i}{\partial \eta_{ji}} + k y_i &= \dfrac{\partial y_j}{\partial \eta_{ji}} + k y_j \quad \textit{sur} \quad \sigma_{ij}, \quad j \neq i. \end{cases} \tag{3.16}$$

Proposition 3.3.1 *Les conditions d'interfaces* (3.16) *sont équivalentes aux contraintes séparées* (3.7) *et* (3.8) *et elles traduisent que les sauts* $[\frac{\partial y_i}{\partial \eta_{ij}}+ky_i]$ *et* $[\frac{\partial y_i}{\partial \eta_{ij}}-ky_i]$ *s'annulent aux frontières entre les sous-domaines.*

3.3.1 Reformulation du problème $(\mathcal{P})$

En divisant le domaine Ω en sous-domaines non recouvrants, la décomposition de la fonctionnelle J paraît tout à fait naturelle. Elle peut s'écrire en fonction des v_i et y_i, solutions des systèmes (3.6), (3.7) et (3.8) :

$$\begin{aligned} J_{dec}(v_I, y_I) &= \sum_{i=1}^{m} \frac{1}{2}\Big(\int_{\Omega_i} |y_i(v_I) - y_{d,i}|^2 dx + \nu \int_{\Omega_i} |v_i|^2 dx\Big) \\ &= \sum_{i=1}^{m} J^i_{dec}(v_i, y_i) \end{aligned}$$

où J^i_{dec} est la restriction de la fonctionnelle J sur le $i$ème sous-domaine.
On peut montrer facilement l'équivalence :

Minimiser $J(v)$

$\Big\|$ $v \in \mathcal{U}_{ad}$,
$\Big\|$ v et y liés par (3.5).

$\Longleftrightarrow$

Minimiser $J_{dec}((v_i)_{i\in I}, (y_i)_{i\in I})$

$\Big\|$ pour $i \in I$ et $j \neq i$:
$\Big\|$ $v_I \in \mathcal{U}_{ad}$,
$\Big\|$ v_i et y_i liés par (3.6),
$\Big\|$ $y_i = y_j$ sur σ_{ij},
$\Big\|$ $\dfrac{\partial y_i}{\partial \eta_{ij}} = \dfrac{\partial y_j}{\partial \eta_{ij}}$ sur σ_{ij}.

En effet, d'une part, prenons la contrainte qui lie l'état direct y au contrôle v et qui est donnée par le système (3.5) du paragraphe 3.2, d'après la proposition 3.2.1. Elle est équivalente à la suite des contraintes sur chaque sous-domaine définie par (3.6), (3.7) et (3.8). D'autre part, en posant $\mathcal{U}^i_{ad} = \mathcal{U}_{ad}|_{\Omega_i}$, la réécriture de la fonctionnelle J comme somme de fonctions coût locales est évidente.
Par conséquent, notre problème sur le domaine global Ω est décomposé en une suite de problèmes sur chaque sous-domaine. Plus loin, nous verrons comment une nouvelle forme de problèmes locaux en sera déduite.

L'idée à la base de la méthode de décomposition de domaine proposée ici, est d'**introduire des variables supplémentaires** ω_{ij} afin d'assurer la continuité de la dérivée normale et telles que :

$$(Dir)_i \qquad \begin{cases} -\Delta y_i &= f_i + v_i \quad \text{dans} \quad \Omega_i, \\ y_i &= 0 \qquad \text{sur} \quad , i, \\ \dfrac{\partial y_i}{\partial \eta} &= \omega_{ij} \qquad \text{sur} \quad \sigma_{ij},\ i<j, \end{cases} \tag{3.17}$$

avec

$$\omega_{ij} = \omega_{ji} \in H^{-\frac{1}{2}}(\sigma_{ij}), \quad j \neq i \text{ pour } \ i = 1, ..., m. \tag{3.18}$$

et on pose la convention de notation : $\eta = \eta_{min(i,j),max(i,j)}$.

Dans une décomposition quelconque de Ω, chaque sous-domaine Ω_i posséde des interfaces σ_{ij} avec ses voisins Ω_j ($i<j$ ou $i>j$).
Donc, avec la notation ci-dessus, nous avons :

- Si $i<j$, alors, $\eta = \eta_{ij}$ et donc $\dfrac{\partial y_i}{\partial \eta_{ij}} = \omega_{ij}$,

- Si $i>j$, alors, $\eta = \eta_{ji} = -\eta_{ij}$ et donc, $\dfrac{\partial y_i}{\partial \eta_{ij}} = -\omega_{ij}$.

Il résulte alors, pour la formulation variationnelle du système $(Dir)_i$, que

$$\begin{aligned} \int_{\Omega_i} \nabla y_i \nabla \varphi_i dx &= \int_{\Omega_i} (f_i + v_i)\varphi_i dx + \sum_{(j,i<j:\sigma_{ij}\neq\emptyset)} \int_{\sigma_{ij}} \omega_{ij}\varphi_i d\sigma_{ij} \\ &- \sum_{(j,i>j:\sigma_{ij}\neq\emptyset)} \int_{\sigma_{ij}} \omega_{ij}\varphi_i d\sigma_{ij}. \end{aligned} \tag{3.19}$$

pour tout $\varphi_i \in H^1(\Omega_i)$ et $\varphi_i = 0$ sur $\partial\Omega$.

Par suite, les systèmes directs locaux seront découplés au niveau de la continuité du flux aux interfaces.

Remarque 3.3.3 *Introduire une condition limite de type Neumann dans le système* (3.17) *par l'intermédiaire de la variable* ω_{ij} *est très proche d'une méthode de "Schur dual" [Roux, 1989] pour résoudre un problème elliptique. Dans notre cas, comme l'équation aux dérivées partielles est elle-même une contrainte du problème de contrôle optimal, la condition de continuité de la dérivée normale seule n'est pas suffisante: il faut de plus imposer la continuité de la fonction sur les interfaces.*

■

Dans toute la suite, on va s'intéresser à la décomposition de notre problème de contrôle et plus précisément à une méthode pour le résoudre efficacement.

3.4 Formulation Lagrangienne

Au paragraphe précédent, la décomposition du domaine a donné lieu à une nouvelle formulation du problème de contrôle qui garde par ailleurs la structure d'un problème d'optimisation avec contraintes. La fonctionnelle J décomposée est alors fonction des variables v_i, y_i et implicitement des ω_{ij} puisqu'elles interviennent indirectement dans le système (3.17). Le problème $(\mathcal{P})$ devient

$$(\mathcal{P})_{dec} \quad \left\{ \begin{array}{l} \text{Minimiser } J_{dec}(v_I, y_I) \\ \\ \left\| \begin{array}{ll} v_I \in \mathcal{U}_{ad} & \\ v_i,\ \omega_{ij} \text{ et } y_i \text{ liés par } (Dir)_i & \underline{i = 1, ..., m} \\ y_i \ = \ y_j \text{ sur } \sigma_{ij}, \quad j \neq i. & \end{array} \right. \end{array} \right.$$

Pour traiter les contraintes de ce nouveau problème de minimisation, introduisons des multiplicateurs de Lagrange relativement à chaque contrainte et réécrivons notre problème en 3 étapes:

3.4.1 1ère étape: Définition du Lagrangien pour le problème décomposé

Nous présentons dans un premier temps les notations qui nous seront utiles pour la suite:

Notations

$$I = \{i \in \mathbb{N} \ ; \ 1 \leq i \leq m\}, \qquad J_i = \{j \in I \ ; \ i \neq j \ \text{ et } \ \sigma_{ij} \neq \emptyset\},$$

$$J_i^+ = \{j \in I \ ; \ i > j \ \text{ et } \ \sigma_{ij} \neq \emptyset\}, \qquad J_i^- = \{j \in I \ ; \ i < j \ \text{ et } \ \sigma_{ij} \neq \emptyset\},$$

$$\mathcal{U} = \Pi_{i \in I} \, \mathcal{U}_{ad}^i,$$

$$V = \Pi_{i \in I} \, V_i, \text{ où } V_i = \{\varphi_i \in H^1(\Omega_i); \varphi_i = 0 \ mboxsur \ \partial\Omega\},$$

$$W_{ij} = Tr V_i \mid_{\sigma_{ij}},$$

$$W^- = \Pi_{i \in I} \, \Pi_{j \in J_i^-} \, W_{ij},$$

$$W = \Pi_{i \in I} \, \Pi_{j \in J_i} \, W_{ij}.$$

Dans toute la suite, on va utiliser les notations (v_I, y_I) à la place de $((v_i)_{i\in I}, (y_i)_{i\in I})$ et ω_J pour $(\omega_{ij})_{i,j\in J_i}$.

D'une part, on associe à la contrainte donnée par le système $(Dir)_i$, un multiplicateur de Lagrange p_i et qui sera l'état adjoint local. D'autre part, un multiplicateur de Lagrange λ_{ij} sera associé à la contrainte linéaire de continuité de y sur l'interface σ_{ij}, c'est à dire, $y_i = y_j$ sur σ_{ij}. Et on définit le Lagrangien $\mathcal{L}$ par

$$\mathcal{L}(v_I, y_I; p_I, \omega_J, \lambda_J) = \sum_{i\in I} J^i_{dec}(v_i, y_i) - \sum_{i\in I}\left(\int_{\Omega_i} \nabla y_i \nabla p_i dx - \int_{\Omega_i} (f_i + v_i) p_i dx\right)$$
$$+ \sum_{i\in I}\sum_{j\in J_i^-} \int_{\sigma_{ij}} \omega_{ij}(p_i - p_j) d\sigma_{ij} + \sum_{i\in I}\sum_{j\in J_i^-} (\lambda_{ij}, y_i - y_j)_{ij}$$

Remarque 3.4.1 *Le produit scalaire $(.,.)_{ij}$ sur l'espace des traces sur σ_{ij} peut être pris égal à celui de $L^2(\sigma_{ij})$, c'est à dire*

$$(\mu, \mu')_{ij} = \int_{\sigma_{ij}} \mu\mu' d\sigma_{ij}.$$

Néanmoins, pour des raisons de convergence, ce choix sera rediscuté plus loin lors de l'élaboration d'algorithmes de recherche de point-selle.

Remarque 3.4.2 *Les 2ème et 3ème sommations de l'expression de $\mathcal{L}$ proviennent de la formulation variationnelle (3.19). En effet, en les sommant sur tous les sous-domaines, les termes d'interfaces s'écrivent*

$$\sum_{i\in I}\sum_{j\in J_i^-} \int_{\sigma_{ij}} \omega_{ij} p_i d\sigma_{ij} - \sum_{i\in I}\sum_{j\in J_i^+} \int_{\sigma_{ij}} \omega_{ij} p_i d\sigma_{ij}. \tag{3.20}$$

Or, la 2ème sommation peut s'écrire aussi

$$\sum_{i\in I}\sum_{j\in J_i^+} \int_{\sigma_{ij}} \omega_{ij} p_i d\sigma_{ij} = \sum_{j\in I}\sum_{j,j<i} \int_{\sigma_{ij}} \omega_{ij} p_i d\sigma_{ij}$$
$$(\text{ un changement d'indice et } \omega_{ij} = \omega_{ji}) \quad = \quad \sum_{i\in I}\sum_{j,i<j} \int_{\sigma_{ij}} \omega_{ij} p_j d\sigma_{ij}.$$

3.4.2 2ème étape : Changement de rôle des variables ω_{ij}

Remarque 3.4.3 *Dans l'expression de $\mathcal{L}_r$, les ω_{ij} peuvent être interprétés comme des multiplicateurs de Lagrange "***implicites***" pour la continuité de l'état adjoint sur σ_{ij} puisqu'ils sont affectés au saut des p_i sur les interfaces.*

A l'aide de cette dernière remarque, nous pouvons réécrire le Lagrangien $\mathcal{L}$ en donnant un nouveau rôle aux variables supplémentaires ω_{ij}, c'est à dire, nous allons avoir

$$\mathcal{L}(\quad \underbrace{v_I, y_I}_{\text{variables de minimisation}} \quad ; \quad \underbrace{p_I, \omega_J, \lambda_J}_{\text{multiplicateurs}} \quad)$$

3.4.3 3ème étape : Définition du Lagrangien augmenté

Pour appliquer la méthode de Lagrangien augmenté à notre problème $(\mathcal{P}_{dec})$, la pénalisation va porter uniquement sur la contrainte de continuité sur l'interface σ_{ij}. On aura donc considéré "implicitement" une partie des contraintes imposée par la décomposition du domaine via le Lagrangien alors que la continuité du flux est donnée explicitement dans le système direct local.

Nous ramenons le problème $(\mathcal{P}_{dec})$ à une recherche de point-selle. Ce dernier, quand il existe, est le même pour le Lagrangien et le Lagrangien augmenté.

Soit r un réel positif, nous définissons le Lagrangien augmenté $\mathcal{L}_r$ associé au problème décomposé par :

$$\begin{aligned}\mathcal{L}_r(v_I, y_I; p_I, \omega_J, \lambda_J) = \sum_{i\in I} J^i_{dec}(v_i, y_i) - \sum_{i\in I}\left(\int_{\Omega_i} \nabla y_i \nabla p_i dx - \int_{\Omega_i} (f_i + v_i) p_i dx\right) \\ + \sum_{i\in I}\sum_{j\in J_i^-} \int_{\sigma_{ij}} \omega_{ij}(p_i - p_j) d\sigma_{ij} + \sum_{i\in I}\sum_{j\in J_i^-}\left((\lambda_{ij}, y_i - y_j)_{ij} + \frac{r}{2}|y_i - y_j|^2_{ij}\right)\end{aligned}$$

3.5 Relation entre le problème global et le problème décomposé

Une fois que nous avons décomposé le domaine, les questions suivantes se posent :

1. Quel lien existe-il entre le problème sur le domaine global et le même problème après la décomposition sans recouvrement de ce même domaine?
2. Qu'avons-nous gagné de cette décomposition?

Dans ce paragraphe, nous ne considérons que le cas simple de produit scalaire L^2 sur l'interface :

$$(\mu, \mu')_{ij} = \int_{\sigma_{ij}} \mu\mu' d\sigma_{ij}$$

Avant de donner une proposition qui répond à ces questions, nous supposons que les hypothèses suivantes sont vérifiées :

Comme $\mathcal{U}_{ad}$ est un convexe fermé de $L^2(\Omega)$, il suffit donc de préserver cette propriété quand nous passons aux restrictions sur les sous-domaines Ω_i, c'est à dire, $\mathcal{U}_{ad}^i = \mathcal{U}_{ad}|_{\Omega_i}$ doit être un convexe fermé de $\mathcal{U}^i = \mathcal{U}|_{\Omega_i}$.
On suppose aussi

$$\forall v \in \mathcal{U}_{ad} \quad \text{alors } v_i = v|_{\Omega_i} \in \mathcal{U}_{ad}^i.$$

Et réciproquement, si nous posons : $v = (v_i)_{i \in I}$ où $v_i \in \mathcal{U}_{ad}^i$, alors, $v \in \mathcal{U}_{ad}$, autrement, $v|_{\Omega_i} = v_i$, pour tout $i \in I$.

Proposition 3.5.1 *Si $(u_I, y_I; p_I, \omega_J, \lambda_J)$ est un point-selle de $\mathcal{L}_r$, quand il existe, alors, pour $i \in I$,*

$$u_i = u|_{\Omega_i}, \quad y_i = y|_{\Omega_i}, \quad p_i = p|_{\Omega_i},$$

où $(u,\ y;\ p)$ est le point-selle du Lagrangien associé au problème $(\mathcal{P})$ et u est le minimum de J.

Preuve : Pour démontrer cette proposition, explicitons la caractérisation d'un point-selle de $\mathcal{L}_r$, s'il existe, que nous allons noter :

$$(p.s) \ = \ (u_I, y_I, p_I, \omega_J, \lambda_J) \tag{3.21}$$

c'est à dire

$$\mathcal{L}_r(u_I, y_I, q_I, \omega_J, \mu_J) \ \leq \ \mathcal{L}_r(u_I, y_I, p_I, \omega_J, \lambda_J) \ \leq \ \mathcal{L}_r(v_I, z_I, p_I, d\omega_J, \lambda_J) \tag{3.22}$$

pout tout $(v_I,\ z_I,\ q_I,\ d\omega_J,\ \mu_J) \ \in \ \mathcal{U} \times V^2 \times W^- \times W^-$.

Tel qu'il a été défini, $\mathcal{L}_r$ est différentiable en chacune de ses variables, alors,
pour tout $i \in I,\ j \in J_i$, (3.22) est équivalente à :

$$(\frac{\partial \mathcal{L}_r}{\partial p_i}(p.s), \varphi_i) \ = \ 0 \quad \forall \varphi_i \in V_i, \tag{3.23}$$

$$(\frac{\partial \mathcal{L}_r}{\partial y_i}(p.s), \varphi_i) \ = \ 0 \quad \forall \varphi_i \in V_i, \tag{3.24}$$

$$(\frac{\partial \mathcal{L}_r}{\partial v_i}(p.s), v_i - u_i) \ \geq \ 0 \quad \forall v_i \in \mathcal{U}_{ad}^i, \tag{3.25}$$

$$(\frac{\partial \mathcal{L}_r}{\partial \lambda_{ij}}(p.s), d\lambda) \ = \ 0 \quad \forall\ d\lambda \in W^-, \qquad j \in J_i^- \tag{3.26}$$

$$(\frac{\partial \mathcal{L}_r}{\partial \omega_{ij}}(p.s), d\omega) \ = \ 0 \quad \forall\ d\omega \in W^-, \qquad j \in J_i^-. \tag{3.27}$$

Ou encore,

– **<u>équation en y_i :</u>**

$$\begin{aligned}\int_{\Omega_i} \nabla y_i \nabla \varphi_i dx &= \int_{\Omega_i} (f_i + v_i)\varphi_i dx + \sum_{j\in J_i^-} \int_{\sigma_{ij}} \omega_{ij}\varphi_i d\sigma_{ij} \\ &- \sum_{j\in J_i^+} \int_{\sigma_{ij}} \omega_{ij}\varphi_i d\sigma_{ij}, \quad \forall \varphi_i \in V_i, \end{aligned} \tag{3.28}$$

ce qui n'est autre que la formulation variationnelle du système direct local (3.17) sur le sous-domaine Ω_i.

– **<u>équation en p_i :</u>**

$$\int_{\Omega_i} \nabla y_i \nabla \varphi_i dx = \int_{\Omega_i} (y_i - y_{d,i})\varphi_i dx + \sum_{j\in J_i} \left((\lambda_{ij}, \varphi_i)_{ij} + \frac{r}{2}(y_i - y_j, \varphi_i)_{ij} \right) \quad \forall \varphi_i \in V_i,$$

obtenue en différentiant $\mathcal{L}_r$ par rapport à y_i. Or, par la formule de Green, nous avons

$$\int_{\Omega_i} \nabla p_i \nabla \varphi_i dx = \int_{\Omega_i} (-\Delta p_i)\varphi_i dx + \sum_{j\in J_i} \int_{\sigma_{ij}} \frac{\partial p_i}{\partial \eta_{ij}} \varphi_i d\sigma_{ij}, \quad \forall \varphi_i \in V_i. \tag{3.29}$$

Par conséquent, nous avons

$$(Adj)_i \qquad \begin{cases} -\Delta p_i &= y_i - y_{d,i} & \text{dans} \quad \Omega_i, \\ p_i &= 0 & \text{sur} \quad , i, \\ \dfrac{\partial p_i}{\partial \eta_{ij}} &= \lambda_{ij} + r(y_i - y_j) & \text{sur} \quad \sigma_{ij},\ j \in J_i. \end{cases} \tag{3.30}$$

D'autre part, soit j l'indice du domaine voisin au $i$ème sous-domaine, la différentiation de $\mathcal{L}_r$ par rapport à y_j implique

$$\frac{\partial p_j}{\partial \eta_{ji}} = -\lambda_{ij} - r(y_i - y_j) \;=\; -\frac{\partial p_i}{\partial \eta_{ij}} \quad \text{sur} \quad \sigma_{ij}.$$

Donc, on obtient la continuité du flux des p_i sur les interfaces entre les sous-domaines.

– **équation en u_i:**

$$\int_{\Omega_i} (p_i + \nu u_i)(v_i - u_i)dx \geq 0 \quad \forall v_i \in \mathcal{U}_{ad}. \tag{3.31}$$

Soit alors, $v \in \mathcal{U}_{ad}$, on pose $v_i = v\,|_{\Omega_i} \in \mathcal{U}_{ad}^i$. Considérons $\tilde{u} = (u_i)_{i\in I}$, alors $\tilde{u} \in \mathcal{U}_{ad}$ et nous avons

$$\int_{\Omega} (p + \nu\tilde{u})(v - \tilde{u})dx = \sum_{i\in I} \int_{\Omega_i} (p_i + \nu u_i)(v_i - u_i)dx \geq 0. \tag{3.32}$$

Or, un tel $\tilde{u}$, quand il existe, est unique, donc, nécessairement

$$\tilde{u} = u.$$

– **équations en ω_{ij} et λ_{ij}:**
Elles sont données respectivement par

$$p_i = p_j \quad \text{sur} \quad \sigma_{ij}, \tag{3.33}$$
$$y_i = y_j \quad \text{sur} \quad \sigma_{ij}. \tag{3.34}$$

Nous avons montré que les u_i, y_i et p_i sont exactement les restrictions des u, y et p qui sont les solutions du système d'optimalité du problème $(\mathcal{P})$ sur le domaine global Ω. (*cf.* Chapitre 2. section 2.2.2).

A l'issue de cette preuve, nous donnons les systèmes qui résument la caractérisation d'un point-selle $(u_I, y_I; p_I, \omega_J, \lambda_J)$ de $\mathcal{L}_r$:

– Etat direct local :

$$\begin{cases} -\Delta y_i = f_i + u_i & \text{dans} \quad \Omega_i, \\ y_i = 0 & \text{sur} \quad ,_i, \\ \dfrac{\partial y_i}{\partial \eta} = \omega_{ij} & \text{sur} \quad \sigma_{ij}\ (j \in J_i). \end{cases} \tag{3.35}$$

– Etat adjoint local :

$$\begin{cases} -\Delta p_i = y_i - y_{d,i} & \text{dans} \quad \Omega_i, \\ p_i = 0 & \text{sur} \quad ,_i, \\ \dfrac{\partial p_i}{\partial \eta_{ij}} = \lambda_{ij} + r(y_i - y_j) & \text{sur} \quad \sigma_{ij}\ \ (j \in J_i). \end{cases} \tag{3.36}$$

– Condition d'optimalité locale :

$$\int_{\Omega_i} (p_i + \nu u_i)(v_i - u_i)dx \geq 0 \qquad \forall v_i \in \mathcal{U}_{ad}^i. \tag{3.37}$$

Théorème 3.5.1 *Si $(u, y; p)$ est la solution du problème de contrôle optimal sur le domaine global, alors, en posant $u_i = u|_{\Omega_i}$, $y_i = y|_{\Omega_i}, p_i = p|_{\Omega_i}$, il existe des multiplicateurs ω_{ij} et λ_{ij} tels que $(u_I, y_I; p_I, \omega_J, \lambda_J)$ soit un point-selle de $\mathcal{L}_r$.*

Preuve : D'une part, y et p sont dans $H^1(\Omega)$ et d'après les propriétés des traces, sur toute interface σ_{ij} entre chaque sous-domaine Ω_i et son voisin Ω_j, nous avons la continuité des états direct et adjoint, c'est à dire,

$$\begin{array}{rcll} y_i & = & y_j & \text{sur} \quad \sigma_{ij}, \\ \dfrac{\partial y_i}{\partial \eta_{ij}} & = & \dfrac{\partial y_j}{\partial \eta_{ij}} & \text{sur} \quad \sigma_{ij}, \\ p_i & = & p_j & \text{sur} \quad \sigma_{ij}, \\ \dfrac{\partial p_i}{\partial \eta_{ij}} & = & \dfrac{\partial p_j}{\partial \eta_{ij}} & \text{sur} \quad \sigma_{ij}. \end{array} \tag{3.38}$$

De plus, les restrictions des y_i et p_i sont dans $H^{\frac{1}{2}}(\sigma_{ij})$ tandis que les $\dfrac{\partial p_i}{\partial \eta_{ij}}$ et $\dfrac{\partial y_i}{\partial \eta_{ij}}$ sont dans $H^{-\frac{1}{2}}(\sigma_{ij})$ (cela dépend de la géométrie du domaine).
Donc, il suffit de prendre

$$\begin{array}{rcl} \omega_{ij} & = & \dfrac{\partial y_i}{\partial \eta_{ij}}, \\ \lambda_{ij} & = & \dfrac{\partial p_i}{\partial \eta_{ij}} - r(y_i - y_j). \end{array} \tag{3.39}$$

D'autre part, considérons l'application caractéristique χ_i de Ω_i, nous avons alors,

$$p_i + \nu u_i = (p + \nu u)\chi_i = 0, \quad \text{pout tout } i = 1, ..., m. \tag{3.40}$$

Dans le cas avec contraintes, nous avons l'inéquation d'Euler sur le domaine global :

$$\int_{\Omega} (p + \nu u)(v - u)dx \geq 0, \quad \forall v \in \mathcal{U}_{ad} \tag{3.41}$$

Soit $v_i \in \mathcal{U}_{ad}^i$, il suffit de prendre dans (3.41)

$$v = v_i \ \text{ sur } \Omega_i \ \text{ et } v = u_j \ \text{ sur } \Omega_j, \ j \neq i,$$

et nous avons la condition d'optimalité locale.

Remarque 3.5.1 *En posant,*

$$\begin{aligned}\tilde{J}^i_{dec}(v_i, y_i) &= \frac{1}{2}\Big(\int_{\Omega_i} |y_i - y_{d,i}|^2 dx + \nu \int_{\Omega_i} |v_i|^2 dx\Big) \\ &+ \sum_{j\in J_i}\left((\lambda_{ij}, y_i - y_j)_{ij} + \frac{r}{2}|y_i - y_j|^2_{ij}\right)\end{aligned}$$

les systèmes (3.35), (3.36) *et* (3.37) *s'interprétent donc comme un système d'optimalité du problème de contrôle optimal local défini par :*

$$\begin{array}{|c|}\hline \\ \textit{Minimiser } \tilde{J}^i_{dec}\ (v_i, y_i) \\ (v_i, y_i) \in \mathcal{U}^i_{ad} \times V_i, \\ v_i,\ y_i \textit{ liés par } (Dir)_i. \\ \\ \hline\end{array}$$

Les variables λ_{ij} et ω_{ij}, étant fixées pour ces problèmes de contrôle optimal locaux, jouent donc le rôle de **recollement** *des solutions locales.*

Une façon de confirmer cette interprétation est de considérer une formulation Lagrangienne de ce sous-problème et on retrouve exactement les systèmes ci-dessus.

En effet, on définit le Lagrangien local associé au problème donné sur Ω_i, c'est à dire, un Lagrangien local $\mathcal{L}^i_r$ tel que :
pour $(v_i, y_i, p_i, (\omega_{ij})_j, (\lambda_{ij})_j)$ dans $\mathcal{U}^i_{ad} \times V_i^2 \times (\Pi_{j\in J_i^-} W_{ij})^2$, l'on ait :

$$\begin{aligned}\mathcal{L}^i_r(v_i, y_i, p_i, (\omega_{ij})_j, (\lambda_{ij})_j) &= J^i_{dec}(v_i, y_i) + \sum_j ((\lambda_{ij}, y_i - y_j)_{ij} + \frac{r}{2}|y_i - y_j|^2_{ij}) \\ &- \int_{\Omega_i} \nabla y_i \nabla p_i dx - \int_{\Omega_i} (f_i + v_i) p_i dx \\ &+ \sum_{j\in J_i^-} \int_{\sigma_{ij}} \omega_{ij} p_i d\sigma_{ij} - \sum_{j\in J_i^+} \int_{\sigma_{ij}} \omega_{ij} p_i d\sigma_{ij}.\end{aligned}$$

Enonçons, maintenant, une proposition donnant une relation entre la dérivée partielle de $\mathcal{L}^i_r$ par rapport à v_i et le gradient de $\tilde{J}^i_{dec}$:

Proposition 3.5.2 *Si $(u_i, y_i, p_i, (\omega_{ij})_j, (\lambda_{ij})_j)$ est un point-selle de $\mathcal{L}^i_r$ noté $(p.s)_i$, alors,*

$$\frac{\partial \mathcal{L}^i_r}{\partial v_i}((p.s)_i) = p_i + \nu u_i = \tilde{J}^i_{dec}{}'(u_i). \tag{3.42}$$

Preuve : . Tout d'abord, montrons que y_i, la solution du système (3.35), est affine continue en u_i en considérant fixés les ω_{ij}. On a :

$$y(u_i) = \mathcal{A}_i u_i + \mathcal{B}_i$$

où $\mathcal{A}_i$ et $\mathcal{B}_i$ sont solutions respectives de

$$\left\{ \begin{array}{rcll} -\Delta \mathcal{B}_i & = & f & \text{dans } \Omega_i, \\ \mathcal{B}_i & = & 0 & \text{sur } , i, \\ \dfrac{\partial \mathcal{B}_i}{\partial \eta} & = & \omega_{ij} & \text{sur } \sigma_{ij}, \end{array} \right. \tag{3.43}$$

et

$$\left\{ \begin{array}{rcll} -\Delta \mathcal{A}_i & = & u_i & \text{dans } \Omega_i, \\ \mathcal{A}_i & = & 0 & \text{sur } , i, \\ \dfrac{\partial \mathcal{A}_i}{\partial \eta} & = & 0 & \text{sur } \sigma_{ij}. \end{array} \right. \tag{3.44}$$

Il vient donc que

$$\begin{aligned} (\frac{\partial \tilde{J}^i_{dec}}{\partial v_i}, v_i - u_i) &= \int_{\Omega_i} (y_i(v_i) - y_i(u_i))(y_i(u_i) - y_{d,i}) dx + \nu \int_{\Omega_i} u_i(v_i - u_i) dx \\ &+ \sum_{j \in J_i} [(\lambda_{ij}, y_i(v_i) - y_i(u_i))_{ij} + r(y_i(u_i) - y_j(u_j), y_i(v_i) - y_i(u_i))_{ij}] \end{aligned}$$

Considérons maintenant les formulations variationnelles respectives de (3.35) et (3.36). Pour tout φ_i et ϕ_i dans V_i, on a :

$$\int_{\Omega_i} \nabla y_i \nabla \varphi_i dx = \int_{\Omega_i} (f_i + u_i)\varphi_i dx + \sum_{j \in J_i^-} \omega_{ij} \varphi_i d\sigma_{ij}, \tag{3.45}$$

$$\int_{\Omega_i} \nabla y_i \nabla \phi_i dx = \int_{\Omega_i} (y_i(u_i) - y_{d,i})\phi_i dx + \sum_{j \in J_i^-} \int_{\sigma_{ij}} (\lambda_{ij} + r(y_i(u_i) - y_j(u_j)))\phi_i d\sigma_{ij}. \tag{3.46}$$

Ainsi, dans un premier temps, il suffit de prendre $\varphi_i = p_i$ dans (3.45) , d'une part pour le système en $y_i(u_i)$ et d'autre part en $y_i(v_i)$ et de faire une soustraction, ce qui donne :

$$\int_{\Omega_i} \nabla (y_i(v_i) - y_i(u_i)) p_i dx = \int_{\Omega_i} (p_i + \nu u_i)(v_i - u_i) dx$$

et dans un deuxième temps, $\phi_i = y_i(v_i) - y_i(u_i)$ dans (3.46) il résulte

$$\begin{aligned}\int_{\Omega_i} \nabla(y_i(v_i) - y_i(u_i))p_i dx &= \int_{\Omega_i} (y_i(v_i) - y_i(u_i))(y_i(u_i) - y_{d,i})dx + \sum_{j\in J_i}((\lambda_{ij}, y_i(v_i) - y_i(u_i))_{ij} \\ &+ r(y_i(u_i) - y_j(u_j, y_i(v_i) - y_i(u_i))_{ij}).\end{aligned}$$

Donc, en identifiant les deux dernières équations et en rappelant que le produit scalaire $(.,.)_{ij}$ est celui de $L^2(\sigma_{ij})$, nous retrouvons le résultat de la proposition.

■

Intéressons-nous maintenant à un algorithme de résolution pour la nouvelle formulation du problème $(\mathcal{P})_{dec}$.

3.6 Un algorithme de résolution

A ce stade, en utilisant la décomposition du domaine, on a réduit le problème original à une recherche de point-selle du Lagrangien augmenté que nous lui avons associé. Nous décrivons dans cette section une méthode itérative pour résoudre notre problème de recherche de point-selle.
Il existe par ailleurs, dans la littérature plusieurs algorithmes liés à ces méthodes Lagrangiennes, voir par exemple, [Fortin et Glowinski, 1982] et [Glowinski et Le Tallec, 1989] pour une étude plus détaillée des algorithmes de Lagrangien augmenté ainsi que leurs applications. Nous en avons présenté quelques rappels au chapitre 1.
Quant à notre cas de problème, nous allons considérer une modification de l'algorithme `ALG1` qui est un algorithme de type `Uzawa` tiré de ces mêmes références :

Algorithme: (`Lag-ddm.1`)

Etape 0. Initialisation :
Pour tout $i \in I,\ j \in J_i^-,\ \omega_{ij}^1,\ \lambda_{ij}^1$ sont pris dans W_{ij}.

Etape 1. Résolution de problèmes de contrôle locaux :
avec ω_{ij}^n, λ_{ij}^n connus, pour chaque $i \in I$, on calcule u_i^n, y_i^n et p_i^n par :

$$\begin{cases} -\Delta y_i^n = f_i + u_i^n & \text{dans } \Omega_i, \\ y_i^n = 0 & \text{sur } \gamma_i, \\ \dfrac{\partial y_i^n}{\partial \eta} = \omega_{ij}^n & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.47}$$

$$\begin{cases} -\Delta p_i^n = y_i^n - y_{d,i} & \text{dans } \Omega_i, \\ p_i^n = 0 & \text{sur } \gamma_i, \\ \dfrac{\partial p_i^n}{\partial \eta_{ij}} = \lambda_{ij}^n + r(y_i^n - y_j^n) & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.48}$$

$$\int_{\Omega_i} (p_i^n + \nu u_i^n)(v_i - u_i^n)dx \geq 0 \qquad \forall v_i \in \mathcal{U}_{ad}^i. \tag{3.49}$$

Etape 2. Mise à jour des multiplicateurs :
avec ρ_ω et ρ_λ deux réels strictement positifs, ω_{ij}^{n+1} et λ_{ij}^{n+1} sont donnés par :

$$\begin{array}{rcll} \omega_{ij}^{n+1} & = & \omega_{ij}^n + \rho_\omega(p_i^n - p_j^n) & \text{sur } \sigma_{ij}, \\ \lambda_{ij}^{n+1} & = & \lambda_{ij}^n + \rho_\lambda(y_i^n - y_j^n) & \text{sur } \sigma_{ij}. \end{array} \tag{3.50}$$

Faire $n \longleftarrow n+1$ et aller à Etape 1.

Cet algorithme amène quelques remarques :

Proposition 3.6.1 *L'algorithme "*(`Lag-ddm.1`)*" est bien défini.*

Preuve : En effet, à l'étape 1., une fois que les multiplicateurs de Lagrange ω_{ij}^n et λ_{ij}^n sont connus et donc fixés, d'après la remarque 3.5.1, les systèmes (3.47), (3.48) et (3.49) sont équivalents au sous-problème de contrôle optimal local dont l'équation d'état est (3.47) et la fonction coût "*locale*" est donnée par :

$$\tilde{J}^i_{dec}(v^n_i, y^n_i) = J^i_{dec}(v^n_i, y^n_i) + \sum_{j\in J_i}[(\lambda^n_{ij}, y^n_i - y^n_j)_{ij} + \frac{r}{2}|y^n_i - y^n_j|^2_{ij}]. \tag{3.51}$$

Remarque 3.6.1 *Il est simple de vérifier que la fonctionnelle $\tilde{J}^i_{dec}$ admet bien sur $\mathcal{U}^i_{ad}$ un minimum qui est le contrôle optimal u^n_i. De plus, dans l'algorithme, l'étape 1. représente une boucle (ou itération) interne et pour la résoudre, nous pourrons utiliser un algorithme de gradient conjugué compte tenu du caractère quadratique que préserve la fonction coût locale.*

3.6.1 Interprétation de (Lag-ddm.1)

L'étape 1. de (Lag-ddm.1) correspond à un problème de contôle optimal local tandis que les mises à jour dans l'étape 2. ne sont autres que la condition nécessaire de minimum par rapport aux variables q^n_{ij} et l'étape de descente d'Uzawa relativement aux multiplicateurs de Lagrange ω^n_{ij} et λ^n_{ij}. Sous forme d'équations ou inéquations, cela s'interpréte par :

- Etape 1. :

$$\mathcal{L}_r(u^n_I, y^n_I; \phi^n_I, \omega^n_J, \lambda^n_J) \leq \mathcal{L}_r(u^n_I, y^n_I; p^n_I, \omega^n_J, \lambda^n_J) \leq \mathcal{L}_r(v^n_I, z^n_I; p^n_I, \omega^n_J, \lambda^n_J)$$

$$\forall v_I \in \mathcal{U}_{ad}, \quad \forall z_i, \phi_i \in V_i.$$

- Etape 2. :

$$\omega^{n+1}_{ij} = \omega^n_{ij} + \rho_\omega \frac{\partial \mathcal{L}_r}{\partial \omega_{ij}}(u^n_I, y^n_I; p^n_I, \omega^n_J, \lambda^n_J),$$

$$\lambda^{n+1}_{ij} = \lambda^n_{ij} + \rho_\lambda \frac{\partial \mathcal{L}_r}{\partial \lambda_{ij}}(u^n_I, y^n_I; p^n_I, \omega^n_J, \lambda^n_J).$$

Remarque 3.6.2 *Une autre remarque importante est que l'algorithme proposé se prête bien au calcul parallèle. En effet, les systèmes* (3.47) *peuvent être résolus indépendamment et simultanément. Ensuite, les valeurs des y^n_i sur les interfaces sont transmises mutuellement entre chaque domaine et ses voisins, puis, elles sont utilisées pour résoudre encore une fois indépendamment et simultanément les systèmes* (3.48)*. Par conséquent, une implantation de notre algorithme sur une machine parallèle est bien adaptée.*

Remarque 3.6.3 *Divers choix de produits scalaires sur l'espace de trace sont possibles. Néanmoins, il faut tenir compte de leur influence sur la convergence des algorithmes de Lagrangien augmenté que nous proposons.*
Premièrement, nous pouvons choisir tout simplement le produit scalaire $L^2(\sigma_{ij})$: il est très facile à implanter mais il n'est pas équivalent au produit scalaire de $H^{\frac{1}{2}}(\sigma_{ij})$. En découle

alors une convergence dépendante du pas de discrétisation et du nombre de sous-domaines. Un deuxième choix est celui du produit scalaire induit par l'opérateur de Laplace-Beltrami sur l'interface σ_{ij} *:*$(-\Delta_{\sigma_{ij}})$ *et défini par:*

$$(\mu, \mu')_{ij} =< (-\Delta_{\sigma_{ij}})^{\frac{1}{2}}\mu, \mu' >, \ \forall \mu, \mu' \in W_{ij}. \tag{3.52}$$

Ce produit scalaire (3.52) *est bien équivalent à celui de* $H^{\frac{1}{2}}(\sigma_{ij})$*. Il est bien adapté au problème d'élasticité non-linéaire et ces choix ont été mentionnés dans un travail détaillé de Sassi [Sassi, 1993] ainsi que par Le Tallec et Sassi dans [Le Tallec et Sassi, 1995].*

■

Nous exposons maintenant une nouvelle formulation de la méthode de décomposition de domaine proposée ci-dessus en considérant différemment les contraintes de continuité aux interfaces.

3.7 Une nouvelle version de la méthode de décomposition

Dans le paragraphe précédent, au niveau des interfaces, nous avons seulement découplé la contrainte de continuité sur le flux. Nous pourrions aussi découpler la continuité des y sur les σ_{ij} à condition d'introduire de nouvelles variables supplémentaires (*cf.* [Bounaïm, 1998]). Nous nous référons à une interprétation de R. Glowinski et P. Le Tallec dans les proceedings de la 3ème conférence sur les méthodes de décomposition de domaine [Glowinski et Le Tallec, 1990] : dans leur étude et dans le cas d'un problème elliptique, ils ont, par un découplage des contraintes aux interfaces et par une approche Lagrangienne, retrouvé les conditions de transmission de P.-L. Lions [Lions, 1988], [Lions, 1989] et [Lions, 1990] (*cf.* Chapitre 1.).
La différence entre cette interprétation et la nôtre est que dans [Glowinski et Le Tallec, 1990], l'EDP elliptique est transformée en un problème de minimisation d'une fonction quadratique provenant de la formulation variationnelle du problème initial alors que dans notre étude, l'EDP représente une contrainte pour le problème de contrôle optimal - qui est une minimisation d'une fonction coût.

Considérons les contraintes de continuité sur l'ensemble des interfaces communes entre les sous-domaines : Pour $i \in I$, $j \in J_i$, elles sont données par :

$$y_i = y_j \quad \text{sur} \quad \sigma_{ij}. \tag{3.53}$$

En introduisant de nouvelles variables $q_{ij} \in W_{ij}$ telles que $q_{ij} = q_{ji}$ et

$$y_i = q_{ij} \quad \text{sur} \quad \sigma_{ij}, \tag{3.54}$$

$$y_j = q_{ij} \quad \text{sur} \quad \sigma_{ij}, \tag{3.55}$$

chacune des contraintes ci-dessus est alors propre au sous-domaine auquel elle est associée. Avec ce changement, notre problème devient :

$$(\mathcal{P}_{new}) \qquad \boxed{\begin{array}{l} \text{Minimiser } J_{dec}((v_i)_{i\in I},(y_i)_{i\in I}) \\ \quad \text{pour } i \in I \ : \\ \quad v_i \in \mathcal{U}_{ad}^i, \\ \quad v_i \text{ et } y_i \text{ liés par (3.17)}, \\ \quad y_i \ = \ q_{ij} \text{ sur } \sigma_{ij}, \quad j \in J_i. \end{array}}$$

3.7.1 Formulation du nouveau problème $(\mathcal{P}_{new})$

Comme dans la section précédente, nous donnons le Lagrangien augmenté associé au problème de minimisation $(\mathcal{P}_{new})$ que l'on notera : $\mathcal{L}_r^{new}$ défini sur $\mathcal{U} \times V^2 \times (W^-)^2 \times W$ et à valeurs dans $\mathbb{R}$ par :

$$\mathcal{L}_r^{new}(v_I, y_I, p_I, q_J, \omega_J, \lambda_{IJ}) = \sum_{i\in I} J_{dec}^i(v_i, y_i) - \sum_{i\in I}\left(\int_{\Omega_i} \nabla y_i \nabla p_i dx - \int_{\Omega_i}(f_i + v_i)p_i dx\right)$$
$$+ \sum_{i\in I}\sum_{j\in J_i^-}\int_{\sigma_{ij}} \omega_{ij}(p_i - p_j)d\sigma_{ij} + \sum_{i\in I}\sum_{j\in J_i}\left((\lambda_{ij}, y_i - q_{ij})_{ij} + \frac{r}{2}|y_i - q_{ij}|_{ij}^2\right).$$

En adoptant les mêmes notations que précédemment, c'est à dire que pour une variable x, nous posons :

$$x_I = (x_i)_{i\in I}, \quad x_{IJ} = (x_{ij})_{[i,\ j,\ i\neq j \text{ et } \sigma_{ij}\neq\emptyset\ ]}, \quad x_{IJ^-} = (x_{ij})_{[i\in I \text{ et } j\in J_i^-]}.$$

Nous énonçons également une proposition liant l'existence d'un éventuel point-selle de $\mathcal{L}_r^{new}$ avec la solution du problème sur le domaine global :

Proposition 3.7.1 *Si $(u_I, \omega_J, y_I, q_J; p_I, \lambda_{IJ})$ est un point-selle de $\mathcal{L}_r^{new}$, quand il existe, alors, pour $i \in I$,*

$$u_i = u|_{\ \Omega_i}, \quad y_i = y|_{\ \Omega_i}, \quad p_i = p|_{\ \Omega_i},$$

où (u, y, p) est la solution du problème $(\mathcal{P})$ (cf. chapitre 2.)

Preuve : Comme pour la démonstration de la proposition 3.5.1, nous donnons la caractérisation d'un point-selle de $\mathcal{L}_r^{new}$ qu'on note

$$(p.s)_{new} = (u_I, y_I, p_I, \omega_{IJ^-}, q_{IJ^-}, \lambda_{IJ})$$

et lorsque l'on différentie $\mathcal{L}_r^{new}$ en chacune de ses variables, nous avons les équations suivantes :

– **équation en y_i :** est la même que (3.17).

– **équation en p_i :**

$$\begin{cases} -\Delta p_i &= y_i - y_{d,i} & \text{dans} \quad \Omega_i, \\ p_i &= 0 & \text{sur} \quad , i, \\ \dfrac{\partial p_i}{\partial \eta_{ij}} &= \lambda_{ij} + r(y_i - q_{ij}) & \text{sur} \quad \sigma_{ij},\ j \in J_i. \end{cases} \tag{3.56}$$

– **équation en q_{ij} :**
Pour $i \in I$ et j tel que $\sigma_{ij} \neq \emptyset$, nous avons

$$(-(\lambda_{ij} + \lambda_{ji}) - r(y_i - q_{ij}) - r(y_j - q_{ij}), d\mu)_{ij} = 0 \quad \forall d\mu \in W_{ij}, \tag{3.57}$$

et il vient que

$$\lambda_{ij} + r(y_i - q_{ij}) = -\lambda_{ji} - r(y_j - q_{ij}) \quad \text{sur} \quad \sigma_{ij}, \tag{3.58}$$

ce qui revient à écrire $\dfrac{\partial p_i}{\partial \eta_{ij}} = -\dfrac{\partial p_j}{\partial \eta_{ji}}$.

– **équations en λ_{ij} et ω_{ij} :**

$$y_i = q_{ij} \quad \text{sur} \quad \sigma_{ij}, \tag{3.59}$$

$$p_i = p_j \quad \text{sur} \quad \sigma_{ij}. \tag{3.60}$$

– **équation en u_i :** est la même que (3.31).

On en déduit donc la continuité des états directs et adjoints et de leurs flux aux interfaces. D'où le résultat de la proposition.

■

Quant au choix d'un algorithme de résolution, nous nous référons une fois encore aux variantes de l'algorithme d'Uzawa et nous proposons deux variantes des algorithmes issus de `ALG2` et `ALG3` dans [Fortin et Glowinski, 1982] et [Glowinski et Le Tallec, 1989]:

Algorithme: (`Lag-ddm.2`)

Etape 0. Initialisation :

Pour tout $i \in I$, ω_{ij}^1, λ_{ij}^1 et q_{ij}^0 sont pris arbitrairement dans W_{ij}.

Etape 1. Résolution de problèmes de contrôle locaux :

avec ω_{ij}^n, λ_{ij}^n et q_{ij}^{n-1} connus, pour chaque $i \in I$, on calcule u_i^n, y_i^n et p_i^n par :

$$\begin{cases} -\Delta y_i^n &= f_i + u_i^n \quad \text{dans} \quad \Omega_i, \\ y_i^n &= 0 \qquad\quad \text{sur} \quad , _i, \\ \dfrac{\partial y_i^n}{\partial \eta} &= \omega_{ij}^n \qquad \text{sur} \quad \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.61}$$

$$\begin{cases} -\Delta p_i^n &= y_i^n - y_{d,i} \qquad\quad \text{dans} \quad \Omega_i, \\ p_i^n &= 0 \qquad\quad \text{sur} \quad , _i, \\ \dfrac{\partial p_i^n}{\partial \eta_{ij}} &= \lambda_{ij}^n + r(y_i^n - q_{ij}^{n-1}) \quad \text{sur} \quad \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.62}$$

$$\int_{\Omega_i} (p_i^n + \nu u_i^n)(v_i - u_i^n)dx \geq 0 \qquad \forall v_i \in \mathcal{U}_{ad}^i. \tag{3.63}$$

Etape 2. Mise à jour des multiplicateurs :

avec ρ_ω et ρ_λ deux réels strictement positifs, ω_{ij}^{n+1}, q_{ij}^n et λ_{ij}^{n+1} sont donnés par :

$$\begin{array}{rcll} \omega_{ij}^{n+1} &=& \omega_{ij}^n + \rho_\omega(p_i^n - p_j^n) & \text{sur} \quad \sigma_{ij}, \\ 2rq_{ij}^n &=& (\lambda_{ij}^n + \lambda_{ji}^n) + r(y_i^n + y_j^n) & \text{sur} \quad \sigma_{ij}, \\ \lambda_{ij}^{n+1} &=& \lambda_{ij}^n + \rho_\lambda(y_i^n - q_{ij}^n) & \text{sur} \quad \sigma_{ij}. \end{array} \tag{3.64}$$

Faire $n \longleftarrow n+1$ et aller à Etape 1.

Algorithme: (`Lag-ddm.3`)

Etape 0. Initialisation :

Pour tout $i \in I$, ω_{ij}^1, λ_{ij}^1 et q_{ij}^0 sont pris arbitrairement dans W_{ij}.

Etape 1. Résolution de problèmes de contrôle locaux :

avec ω_{ij}^n, λ_{ij}^n et q_{ij}^{n-1} connus, pour chaque $i \in I$, on calcule u_i^n, y_i^n et p_i^n par :

$$\begin{cases} -\Delta y_i^n = f_i + u_i^n & \text{dans } \Omega_i, \\ y_i^n = 0 & \text{sur } \gamma_i, \\ \dfrac{\partial y_i^n}{\partial \eta} = \omega_{ij}^n & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.65}$$

$$\begin{cases} -\Delta p_i^n = y_i^n - y_{d,i} & \text{dans } \Omega_i, \\ p_i^n = 0 & \text{sur } \gamma_i, \\ \dfrac{\partial p_i^n}{\partial \eta_{ij}} = \lambda_{ij}^n + r(y_i^n - q_{ij}^{n-1}) & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.66}$$

$$\int_{\Omega_i} (p_i^n + \nu u_i^n)(v_i - u_i^n)dx \geq 0 \qquad \forall v_i \in \mathcal{U}_{ad}^i. \tag{3.67}$$

Etape 2. Mise à jour des multiplicateurs :

avec ρ_ω et ρ_λ deux réels strictement positifs, ω_{ij}^{n+1}, $\lambda_{ij}^{n+\frac{1}{2}}$, q_{ij}^n et λ_{ij}^{n+1} sont donnés par :

$$\begin{aligned} \omega_{ij}^{n+1} &= \omega_{ij}^n + \rho_\omega(p_i^n - p_j^n) && \text{sur } \sigma_{ij}, \\ \lambda_{ij}^{n+\frac{1}{2}} &= \lambda_{ij}^n + \rho_\lambda(y_i^n - q_{ij}^{n-1}) && \text{sur } \sigma_{ij}. \\ 2rq_{ij}^n &= (\lambda_{ij}^{n+\frac{1}{2}} + \lambda_{ji}^{n+\frac{1}{2}}) + r(y_i^n + y_j^n) && \text{sur } \sigma_{ij}, \\ \lambda_{ij}^{n+1} &= \lambda_{ij}^{n+\frac{1}{2}} + \rho_\lambda(y_i^n - q_{ij}^n) && \text{sur } \sigma_{ij}. \end{aligned} \tag{3.68}$$

Faire $n \longleftarrow n+1$ et aller à Etape 1.

Proposition 3.7.2 *Les deux algorithmes "(*`Lag-ddm.2`*)" et "(*`Lag-ddm.3`*)" sont bien définis.*

En effet, il suffit de remarquer qu'ils forment chacun, une suite de sous-problèmes de contrôle optimal locaux dont l'équation d'état, qui est la même pour les deux algorithmes, est (3.65) et la fonction coût est définie par

$$\hat{J}^i_{dec}(v^n_i, y^n_i) = J^i_{dec}(v^n_i, y^n_i) + \sum_{j\in J_i}[(\lambda^n_{ij}, y^n_i - q^{n-1}_{ij})_{ij} + \frac{r}{2}|y^n_i - q^{n-1}_{ij}|^2_{ij}]. \tag{3.69}$$

Cette série de problèmes locaux est suivie d'une mise à jour pour "(`Lag-ddm.2`)" (resp. 2 mises à jour pour "(`Lag-ddm.3`)") des multiplicateurs de Lagrange et des variables supplémentaires q_{ij} qui permettent tous ensemble de raccorder les données du problème aux zones frontières.

3.7.2 Interprétation des algorithmes (`Lag-ddm.2`) et (`Lag-ddm.3`)

En analogie avec les algorithmes présentés dans les rappels sur le Lagrangien augmenté (*cf.* section 1.3.2 du chapitre 1.), l'étape 1. représente ls condition min-max relativement à la contrainte donnée par l'équation d'état local (sur Ω_i), cela s'écrit également

$$\mathcal{L}_r(u^n_I, y^n_I, q^{n-1}_J; \phi^n_I, \omega^n_J, \lambda^n_J) \le \mathcal{L}_r(u^n_I, y^n_I, q^{n-1}_J; p^n_I, \omega^n_J, \lambda^n_J) \le \mathcal{L}_r(v^n_I, z^n_I, q^{n-1}_J; p^n_I, \omega^n_J, \lambda^n_J)$$

$$\forall v_I \in \mathcal{U}_{ad}, \quad \forall z_i, \phi_i \in V_i,$$

que l'on peut considérer comme un problème de contrôle optimal local (voir proposition ci-dessus ainsi que l'interprétation de (`Lag-ddm.1`)). Quant à l'étape 2., elle représente une descente d'Uzawa par rapport aux multiplicateurs ω^n_{ij} et λ^n_{ij} et la condition nécessaire de minimisation relativement aux variables supplémentaires q^n_{ij}. L'algorithme (`Lag-ddm.3`) comprend deux mises à jour des λ^n_{ij}.

Remarque 3.7.1 *Dans l'algorithme "(`Lag-ddm.3`)", en supprimant les variables λ^n_{ij}, ω^n_{ij} et q^n_{ij} dans le cas où l'on a "$\rho_\omega = \rho_\lambda = r$", de nouvelles conditions aux limites sur les interfaces sont obtenues :*
Pour l'état adjoint :

$$-\frac{\partial p^{n+1}_i}{\partial \eta_{ij}} + r y^{n+1}_i = \frac{\partial p^n_j}{\partial \eta_{ji}} + r y^n_j, \tag{3.70}$$

qui sont exactement les conditions aux limites pour les états adjoints p_i de l'algorithme `Alg2` *de [Benamou, 1994, Benamou, 1996].*
Pour l'état direct :

$$-\frac{\partial y^{n+1}_i}{\partial \eta_{ij}} + r p^n_i = \frac{\partial y^n_j}{\partial \eta_{ji}} + r p^n_j, \tag{3.71}$$

et dans ce cas, ces conditions pour les y_i sont proches de celles données dans ce même algorithme `Alg2`.

Preuve : Tout d'abord, nous rappelons que les algorithmes proposés par Benamou reposent sur l'utilisation des conditions de transmission introduites et étudiées par Lions. Comme le système d'optimalité auquel se réduit le problème de contrôle optimal est composé de systèmes d'équations aux dérivées partielles et d'une inéquation variationnelle, l'application desdites conditions d'interface (appelées aussi de type Robin) se déduit sans difficultés, et particulièrement, dans le cas de problèmes elliptiques. Nous présentons, ici, les conditions de transmission l'algorithme **Alg2** qui est le plus proche de notre algorithme :

$$-\frac{\partial y_i^{n+1}}{\partial \eta_{ij}} + r p_i^{n+1} = \frac{\partial y_j^n}{\partial \eta_{ji}} + r p_j^n, \tag{3.72}$$

$$-\frac{\partial p_i^{n+1}}{\partial \eta_{ij}} + r y_i^{n+1} = \frac{\partial p_j^n}{\partial \eta_{ji}} + r y_j^n. \tag{3.73}$$

Les systèmes direct et adjoint locaux, dans ce cas, sont couplés et les conditions aux limites sur les interfaces à l'itération $(n+1)$ utilisent celles sur les domaines voisins et à l'itération précédente. C'est exactement le principe des algorithmes de Schwarz réécrits et interprétés par Lions.

Revenons à la preuve des transformations obtenues ci-dessus. En partant de la condition aux limites de p_i^{n+1}, nous avons

$$\begin{aligned}
-\frac{\partial p_i^{n+1}}{\partial \eta_{ij}} + r y_i^{n+1} &= -\lambda_{ij}^{n+1} + r q_{ij}^n, \\
&= -\lambda_{ij}^{n+\frac{1}{2}} - r y_i^n + 2 r q_{ij}^n, && (\text{ expression de } \lambda_{ij}^{n+1}) \\
&= -\lambda_{ij}^{n+\frac{1}{2}} - r y_i^n + \lambda_{ij}^{n+\frac{1}{2}} + \lambda_{ji}^{n+\frac{1}{2}} + r(y_i^n + y_j^n), && (\text{ expression de } 2 r q_{ij}^n) \\
&= \lambda_{ji}^{n+\frac{1}{2}} + r y_j^n, \\
&= \frac{\partial p_j^n}{\partial \eta_{ji}} + r y_j^n.
\end{aligned}$$

Idem, pour les y_i^n, nous partons de l'expression de la condition aux limites Neumann de y_i^{n+1} sur σ_{ij}, c'est à dire,

$$\begin{aligned}
-\frac{\partial y_i^{n+1}}{\partial \eta_{ij}} &= -\omega_{ij}^{n+1}, \\
&= -\omega_{ij}^n - r(p_i^n - p_j^n),
\end{aligned}$$

Donc,

$$-\frac{\partial y_i^{n+1}}{\partial \eta_{ij}} + r p_i^n = -\omega_{ij}^n + r p_j^n = \frac{\partial y_j^n}{\partial \eta_{ji}} + r p_j^n.$$

En conclusion, nous remarquons bien que les systèmes à résoudre dans les différents algorithmes proposés ne sont pas couplés, contrairement aux systèmes de Benamou.

3.7.3 Autre formulation de $(\mathcal{P})_{dec}$

Nous avons remarqué que dans l'expression de $\mathcal{L}_r$, les ω_{ij} jouent le rôle de multiplicateurs implicites (puisqu'ils proviennent de façon "naturelle" de la formulation variationnelle des équations en y_i). Nous remarquons de plus que si $(u_I, y_I, q_J; p_I, \omega_J, \lambda_{IJ})$ de $\mathcal{L}_r^{new}$, il l'est également pour $\mathcal{L}_r^{bis}$ qu'on définit par

$$\mathcal{L}_r^{bis}(u_I, y_I, q_J; p_I, \omega_J, \lambda_{IJ}) = \mathcal{L}_r^{new}(u_I, y_I, q_J; p_I, \omega_J, \lambda_{IJ}) + \frac{r}{2}\sum_{i=1}^{m}\sum_{j\in J_i}\int_{\sigma_{ij}} |p_i - p_j|^2 d\sigma_{ij} \quad (3.74)$$

Une manière équivalente d'écrire $\mathcal{L}_r^{bis}$ est d'introduire des variables supplémentaires q_{ij}^p et noter les q_{ij} précédemment introduites par des q_{ij}^y, nous avons alors

$$\begin{aligned}\mathcal{L}_r^{bis}(v_I, y_I, q_J^y, q_J^p; p_I, \omega_J, \lambda_{IJ}) = \sum_{i\in I} J_{dec}^i(v_i, y_i) - \sum_{i\in I}\left(\int_{\Omega_i}\nabla y_i \nabla p_i dx - \int_{\Omega_i}(f_i + v_i)p_i dx\right)\\ + \sum_{i\in I}\sum_{j\in J_i}\left[\int_{\sigma_{ij}}\omega_{ij}(p_i - q_{ij}^p)d\sigma_{ij} + \int_{\sigma_{ij}}\lambda_{ij}(y_i - q_{ij}^y)\sigma_{ij} + \frac{r}{2}\int_{\sigma_{ij}}|y_i - q_{ij}|^2 d\sigma_{ij}\right].\end{aligned}$$

Le même développement que dans les paragraphes précédents nous amène à donner la caractérisation d'un point-selle de $\mathcal{L}_r^{bis}$ (quand il existe) qui diffère de celui de $\mathcal{L}_r^{new}$ par les seules conditions aux limites sur les interfaces des systèmes de y_i : (3.17), c'est à dire, nous avons le système suivant :

$$\left\{\begin{array}{rcll} -\Delta y_i &=& f_i + u_i & \text{dans } \Omega_i, \\ y_i &=& 0 & \text{sur } , i, \\ \dfrac{\partial y_i}{\partial \eta} &=& \omega_{ij} + r(p_i - q_{ij}^p) & \text{sur } \sigma_{ij}\ (j \in J_i). \end{array}\right. \quad (3.75)$$

Pour la nouvelle formulation du problème décomposé, nous nous contentons de donner uniquement une nouvelle version de l'algorithme (`Lag-ddm.3`) pour le calcul d'un point-selle de $\mathcal{L}_r^{bis}$. L'étape 1. est la même que celle de cet algorithme avec une différence sur les conditions aux limites sur les interfaces σ_{ij}. Quant aux doubles mises à jour de l'étape 2., elles portent sur les multiplicateurs ω_{ij}^n et λ_{ij}^n.

Algorithme: (`Lag-ddm.3bis`)

Etape 0. Initialisation :

Pour tout $i \in I$, ω_{ij}^1, λ_{ij}^1, $q_{ij}^{y,0}$ et $q_{ij}^{p,0}$ sont pris arbitrairement dans W_{ij}.

Etape 1. Résolution de problèmes de contrôle locaux :

avec ω_{ij}^n, λ_{ij}^n, $q_{ij}^{y,n-1}$ et $q_{ij}^{p,n-1}$ connus, pour chaque $i \in I$, on calcule u_i^n, y_i^n et p_i^n par :

$$\begin{cases} -\Delta y_i^n = f_i + u_i^n & \text{dans } \Omega_i, \\ y_i^n = 0 & \text{sur } \gamma_i, \\ \dfrac{\partial y_i^n}{\partial \eta} = \omega_{ij}^n + r(p_i^n - q_{ij}^{p,n-1}) & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.76}$$

$$\begin{cases} -\Delta p_i^n = y_i^n - y_{d,i} & \text{dans } \Omega_i, \\ p_i^n = 0 & \text{sur } \gamma_i, \\ \dfrac{\partial p_i^n}{\partial \eta_{ij}} = \lambda_{ij}^n + r(y_i^n - q_{ij}^{n-1}) & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{cases} \tag{3.77}$$

$$\int_{\Omega_i} (p_i^n + \nu u_i^n)(v_i - u_i^n)dx \geq 0 \qquad \forall v_i \in \mathcal{U}_{ad}^i. \tag{3.78}$$

Etape 2. Mise à jour des multiplicateurs :

avec ρ_ω et ρ_λ deux réels strictement positifs, $\omega_{ij}^{n+\frac{1}{2}}$, $\lambda_{ij}^{n+\frac{1}{2}}$, $q_{ij}^{y,n}$, $q_{ij}^{p,n}$, ω_{ij}^{n+1} et λ_{ij}^{n+1} sont donnés par :

$$\begin{array}{lcll} \omega_{ij}^{n+\frac{1}{2}} & = & \omega_{ij}^n + \rho_\omega(p_i^n - q_{ij}^{p,n-1}) & \text{sur } \sigma_{ij}, \\ \lambda_{ij}^{n+\frac{1}{2}} & = & \lambda_{ij}^n + \rho_\lambda(y_i^n - q_{ij}^{y,n-1}) & \text{sur } \sigma_{ij}. \\ q_{ij}^{p,n} & = & (\omega_{ij}^{n+\frac{1}{2}} + \omega_{ji}^{n+\frac{1}{2}})/2r + (p_i^n + p_j^n)/2 & \text{sur } \sigma_{ij}, \\ q_{ij}^{y,n} & = & (\lambda_{ij}^{n+\frac{1}{2}} + \lambda_{ji}^{n+\frac{1}{2}})/2r + (y_i^n + y_j^n)/2 & \text{sur } \sigma_{ij}, \\ \omega_{ij}^{n+1} & = & \omega_{ij}^{n+\frac{1}{2}} + \rho_\omega(p_i^n - q_{ij}^{p,n}) & \text{sur } \sigma_{ij}, \\ \lambda_{ij}^{n+1} & = & \lambda_{ij}^{n+\frac{1}{2}} + \rho_\lambda(y_i^n - q_{ij}^{y,n}) & \text{sur } \sigma_{ij}. \end{array} \tag{3.79}$$

Faire $n \longleftarrow n+1$ et aller à Etape 1.

3.7.4 Réécriture des conditions de transmission sur les interfaces

Par analogie à la remarque 3.7.1, nous pouvons également, dans la nouvelle version de (`Lag-ddm.3`), éliminer les variables $q_{ij}^{p,n}$, ω_{ij}^n, $q_{ij}^{y,n}$ et λ_{ij}^n de l'étape 2. de (`Lag-ddm.3bis`). Nous retrouvons les mêmes conditions de transmission (3.70) et une légère différence par rapport à celles de p_i^n, i.e., (3.71). En effet, dans les mises à jour des ω_{ij}^n, on prend $\rho_\omega = r$ et nous avons

$$\begin{aligned}
-\frac{\partial y_i^{n+1}}{\partial \eta_{ij}} + r p_i^{n+1} &= -\omega_{ij}^{n+1} + r q_{ij}^{p,n}, \\
&= -\omega_{ij}^{n+\frac{1}{2}} - r p_i^n + 2 r q_{ij}^{p,n}, \qquad (\text{ expression de } \omega_{ij}^{n+1}) \\
&= -\omega_{ij}^{n+\frac{1}{2}} - r p_i^n + \omega_{ij}^{n+\frac{1}{2}} + \omega_{ji}^{n+\frac{1}{2}} + r(p_i^n + p_j^n), \qquad (\text{ expression de } 2 r q_{ij}^{p,n}) \\
&= \omega_{ji}^{n+\frac{1}{2}} + r p_j^n, \\
&= \frac{\partial y_j^n}{\partial \eta_{ji}} + r p_j^n.
\end{aligned}$$

Ces dernières conditions sont exactement celles de l'algorithme `Alg2` obtenu par Benamou dans [Benamou, 1994, Benamou, 1996].

En résumé, en supprimant les multiplicateurs de Lagrange et les variables supplémentaires dans l'algorithme (`Lag-ddm.3bis`), nous retrouvons exactement l'un des algorithmes de [Benamou, 1994] où on montre sa convergence grâce à des considérations de pseudo-énergie.

3.7.5 Autre considération des conditions limites aux interfaces

Si nous remplaçons, la contrainte de continuité du flux sur les bords entre les sous-domaines

$$\frac{\partial y_i}{\partial \eta} = \frac{\partial y_j}{\partial \eta} = \omega_{ij} \quad \text{sur } \sigma_{ij}, \tag{3.80}$$

par la combinaison

$$\frac{\partial y_i}{\partial \eta} + k y_i = \frac{\partial y_j}{\partial \eta} + k y_j = \omega_{ij} \quad \text{sur } \sigma_{ij}, \tag{3.81}$$

il résulte que dans l'algorithme (`Lag_ddm.3`), les conditions d'interfaces des y_i^n et des p_i^n sont remplacées respectivement par

$$\frac{\partial y_i^n}{\partial \eta} + k y_i^n = \omega_{ij}^n \quad \text{sur } \sigma_{ij}, \tag{3.82}$$

$$\frac{\partial p_i^n}{\partial \eta} + k p_i^n = \lambda_{ij}^n + \rho_\lambda (y_i^n - q_{ij}^{n-1}) \quad \text{sur } \sigma_{ij}. \tag{3.83}$$

Remarque 3.7.2 *Si nous supprimons les λ_{ij}^n, ω_{ij}^n et les q_{ij}^n dans les conditions aux limites ci-dessus, il vient*

$$-\frac{\partial y_i^{n+1}}{\partial \eta_{ij}} + k y_i^{n+1} + \rho_\omega p_i^n = -\frac{\partial y_j^n}{\partial \eta_{ij}} + k y_j^n + \rho_\omega p_j^n \quad sur\ \sigma_{ij}, \tag{3.84}$$

$$-\frac{\partial p_i^{n+1}}{\partial \eta_{ij}} + k p_i^{n+1} + \rho_\omega y_i^n = -\frac{\partial p_j^n}{\partial \eta_{ij}} + k p_j^n + \rho_\omega y_j^n \quad sur\ \sigma_{ij}. \tag{3.85}$$

Ces conditions apparaissent également sur l'un des algorithmes proposés dans [Benamou, 1994].

■

Avant de présenter une application de la méthode de décomposition de domaine que nous avons développée tout au long de ce chapitre, nous allons consacrer le chapitre suivant à une introduction et des généralités sur le parallélisme aussi bien que certains outils essentiels pour l'implantation des "`(Lag_ddm.i)`," i=1,2,3.

Chapitre **4 :**

Parallélisme et programmation par passage de messages.

Le but de ce chapitre est d'exposer des généralités sur le calcul parallèle. Nous allons passer en revue le fonctionnement de certaines machines parallèles tout en s'attardant sur celles à mémoire distribuée qui sont utilisées pour l'implantation des résultats numériques de cette thèse. Nous parlons aussi des performances d'un programme parallèle ainsi que des caractéristiques de la programmation par passage de messages et plus particulièrement de la librairie **MPI**.

4.1 Introduction

Tout le monde sait que le travail avance plus rapidement si on le fait à plusieurs que si l'on est seul. Néanmoins, il n'y a pas très longtemps que ce principe a commencé à faire ses pas dans le monde de l'informatique. En effet, cela est dû à deux raisons essentielles : d'une part, les coûts deviennent plus élevés pour la production et le développement des codes et d'autre part, il y a eu d'importants progrès dans la technologie des microprocesseurs durant les deux dernières décennies. Sans oublier pour autant une autre raison, économique cette fois, qui a conduit à de significatives évolutions dans **le parallélisme** et elle concerne la fabrication des calculateurs. Il est à préciser qu'un calculateur conçu à partir de processeurs de grande série coûte moins cher qu'un superordinateur très rapide.

4.2 Les différents types de parallélisme

Parmi les premiers types de machines parallèles, nous distinguons les machines **vectorielles** telles que le `CRAY`: leur fonctionnement est basé sur le travail dit "*à la chaîne*". Viennent ensuite, les premières machines parallèles un peu plus performantes que les machines vectorielles et qui ont été constituées de plusieurs processeurs vectoriels très rapides.

Les machines parallèles sont divisées en deux catégories :

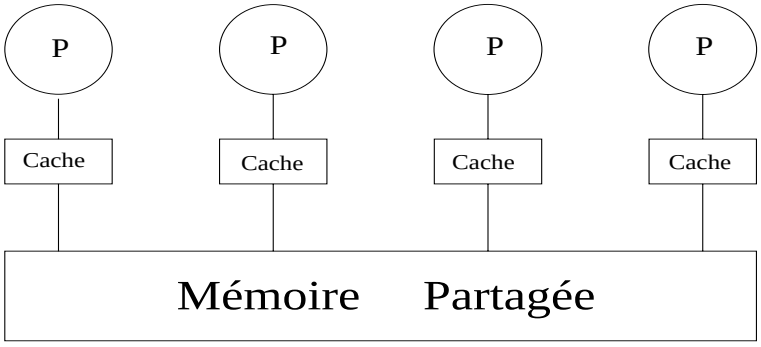

FIG. 4.1 – *Architecture parallèle à mémoire partagée.*

4.2.1 Machines parallèles à mémoire partagée

Comme leur nom l'indique, ces machines sont composées d'un seul banc mémoire de taille importante auquel accèdent de façon uniforme plusieurs processeurs (voir figure 4.1). Cependant, ces architectures connaîssent de sérieuses limitations pour accroître sensiblement la puissance de calcul disponible, notamment,

- Les coûts de construction pour ce type de machines sont très élevés.

- D'un point de vue conception, l'évolution matérielle de ce type d'architecture est limitée puisque le nombre de processeurs que l'on peut relier de façon performante à une telle mémoire ne dépasse pas 8 ou 16. De plus, le banc mémoire n'est pas extensible à l'infini : au delà d'une certaine limite, les temps d'accès à des zones différentes distantes de la mémoire ne peuvent être considérés comme constants car la vitesse de circulation de l'information est bornée par la vitesse des électrons dans la matière.

Une façon d'apporter une amélioration à ces architectures à mémoire partagée consiste à adopter une architecture de type "*distribuée*". Et c'est cette catégorie de machines à laquelle nous nous intéressons dans la suite.

4.2.2 Machines parallèles à mémoire distribuée

Dans ce genre de machines proposées par les constructeurs, des nœuds constitués d'un processeur et de sa mémoire locale, à laquelle il est le seul à accéder directement, sont reliés entre eux grâce à un réseau de communication performant (voir figure 4.2). Ce dernier, outre la connexion entre les processeurs, permet l'échange de l'information nécessaire pour coordonner leur travail.
Un avantage de cette architecture est son coût relativement moindre comparée à une

machine conçue autour d'une mémoire partagée et comportant le même nombre de processeurs. En effet, pour les nœuds d'une telle machine, les constructeurs utilisent généralement le dernier représentant d'une famille de processeurs ayant déjà fait ses preuves dans le monde des stations de travail et lui ajoutent de la mémoire standard "bon marché". Du même coup, ils réduisent aussi leur investissement logiciel puisque des compilateurs performants sont disponibles pour ce type de processeurs.

Reste bien sûr à leur charge la conception d'un réseau de communication assez rapide pour relier au mieux les nœuds de calcul. Et s'il est possible de réduire le nombre de communications ou se contenter d'une efficacité parallèle moyenne, des stations de travail, reliées entre elles par des liaisons Ethernet classiques, peuvent construire un réseau de stations homogènes ou hétérogènes, ce qui constitue une sorte de "`machine parallèle du pauvre`".

D'autre part, outre leur coût compétitif, ces "clusters" de processeurs présentent aussi l'intérêt de pouvoir évoluer facilement et sans surcoût en fonction des besoins de l'utilisateur. Dans le cas des architectures modernes de type SP2 ou CRAY T3D/E par exemple, on peut aisément ajouter des nœuds supplémentaires sans dégrader les preformances de la machine globale. Mais l'inconvénient majeur de ce type de machine vient de la forte hiérarchisation de la mémoire : par l'intermédiaire d'un réseau d'interconnexion, l'accès à une information "distante", c'est à dire contenue dans la mémoire d'un autre processeur, prendra typiquement un temps 1000 fois plus long que pour une information "locale". Pour obtenir des performances soutenues, il est donc essentiel pour le développeur de limiter au minimum le nombre de ces accès distants. C'est le but de la recherche en algorithmique parallèle.

4.3 Généralités sur les machines SPMD

Si le choix d'une architecture à mémoire distribuée définit les caratéristiques intrinsèques d'une machine, la qualité de l'environnement logiciel mis à disposition du programmeur joue un rôle considérable dans la facilité de conception d'un programme parallèle. La "vision" qu'a l'utilisateur de la machine est importante. Des compilateurs comme HPF (High Performance Fortran) peuvent, par exemple, lui épargner toute gestion des communications en lui présentant la mémoire comme un seul bloc accessible par tous les processeurs simultanément. Cependant, HPF présente pour l'instant des performances toujours inférieures au modèle SPMD (Single Process Multiple Data) combiné avec l'utilisation d'une bibliothèque de passage de messages. C'est ce dernier mode de programmation que nous adoptons pour la présente étude.

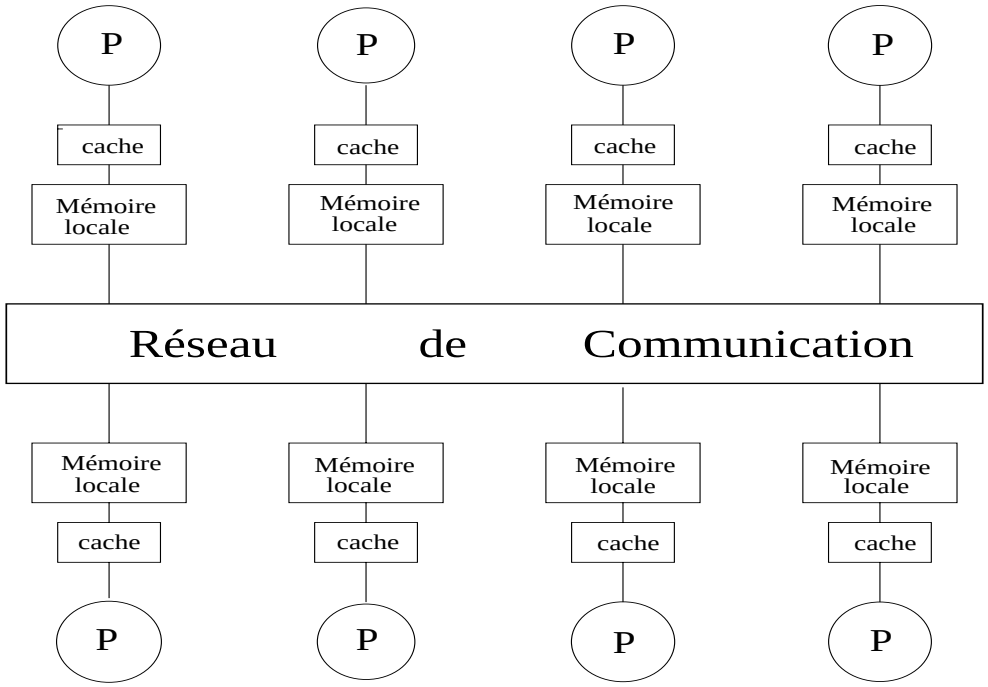

FIG. 4.2 – *Architecture parallèle à mémoire distribuée.*

4.3.1 Comment se fait la programmation par "processus communicants"?

L'environnement logiciel de programmation par passage de messages ou par "*processus communicants*" est une manière plus proche de ce qui se passe réellement (voir figure 4.3). En effet, chaque processeur exécute une copie du problème et n'a accès qu'aux données contenues dans sa propre mémoire. L'accès aux informations distantes doit être géré explicitement par le programmeur grâce à des instructions pour envoyer ou recevoir des messages qui circulent d'un nœud à l'autre sur le réseau de communication.
Concrètement, chaque processeur exécute une tâche séquentielle quand ces tâches sont indépendantes et à chaque fois qu'il a besoin de données de chez ses voisins ou éventuellement leur en transmettre, intervient donc l'outil du réseau pour permettre aux processeurs de communiquer entre eux ensemble ou en partie.
En résumé, chaque processeur accède donc directement aux données qui sont "*propres à lui*" mais ne "voit" le monde extérieur que par l'intermédiaire de messages qu'il envoie ou reçoit et ce sont ces messages qui assurent la coopération entre les processeurs pour la constitution d'un programme structuré.

Cette façon de procéder est fournie par de nombreuses bibliothèques dont nous allons, plus loin, exposer un exemple "**MPI**" (Message passing Interface).

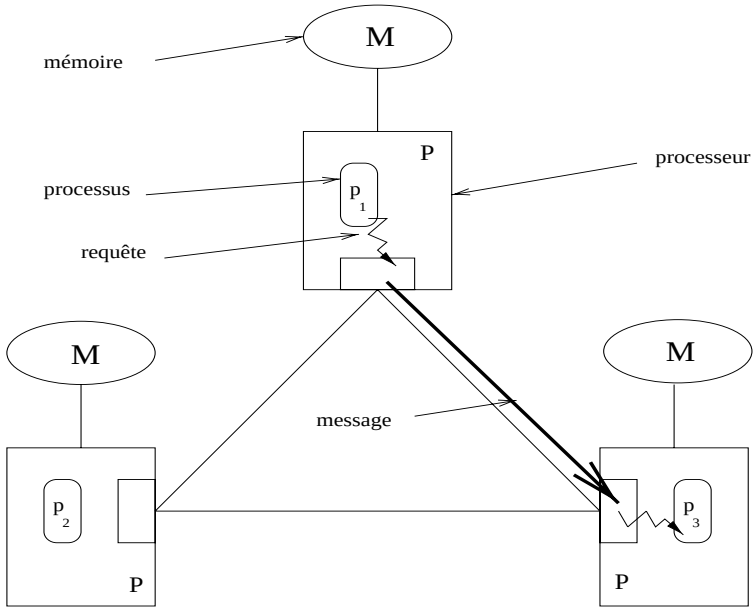

FIG. 4.3 – *Exemple de programmation par passage de messages.*

4.4 Comment évaluer les performances d'un programme parallèle SPMD ?

Pour finir ces généralités sur le modèle SPMD, nous allons voir comment évaluer les performances d'un programme parallèle et encore faut-il préciser que les notions que nous allons présenter rentrent dans le cas où on fait une parallélisation d'un code dont la version séquentielle est déjà existante, ce qui est un peu différent du cas où l'on peut exhiber des algorithmes adaptés aux machines parallèles tel qu'il s'est présenté pour notre algorithme de décomposition de domaine du chapitre précédent.

En général, il existe plusieurs façons de mesurer les performances d'un ordinateur, à savoir, la vitesse de calcul sur des variables réelles. Pour les machines parallèles, plus particulièrement, il faut introduire certaines notions telles que **l'accélération** ou **l'efficacité** d'un programme parallèle et qui sont définies par :

L'accélération : est le rapport du temps d'exécution du programme séquentiel ou sur un seul processeur ($\mathbf{T}_1$) sur le temps d'exécution de la même application en parallèle sur n processeurs ($\mathbf{T}_n$) :

$$A_n = \frac{T_1}{T_n} \tag{4.1}$$

L'efficacité : est l'accélération divisée par le nombre de processeurs. Elle s'exprime en général sous forme de pourcentage :

$$E_n = \frac{A_n}{n} = \frac{T_1}{nT_n} \tag{4.2}$$

Mais pour des gains d'exécution appréciables, il est capital qu'une grande proportion du programme soit parallélisable.
Par exemple, si une fraction du programme p $(0 \leq p \leq 1)$ est parallélisable et si T_1 est le temps CPU sur un processeur, alors T_n, le temps CPU sur n processeurs, est tel que :

$$T_n = \underbrace{\frac{p}{n}T_1}_{\text{Temps en parallèle}} + \underbrace{(1-p)T_1}_{\text{Temps séquentiel}} + \underbrace{T_{comm(n\_proc)}}_{\text{Temps de communication}} \tag{4.3}$$

Et en remplaçant dans (4.2), nous avons l'efficacité théorique de **la loi d'Amdahl** :

$$E_n = \frac{1}{p + (1-p)n + \frac{T_{comm(n\_proc)}}{T_1}} \tag{4.4}$$

Quelques remarques immédiates découlent de la loi d'Amdahl:

D'une part, un code parallélisable à 100%, ne nécessitant aucune communication, a une efficacité de 1 et dans tous les autres cas, nous avons $E_n \leq 1$.
D'autre part, il est logique de prendre " nT_n " pour le temps CPU parallèle puisque l'on consomme simultanément " T_n " sur chaque processeur. Cette façon d'évaluer les coûts de calcul peut surprendre, mais, c'est bien ainsi que les centres de calcul facturent le temps CPU consommé.

Contrairement aux bornes théoriques de la loi d'Amdahl, qui n'espèrent pas un avenir "positif" du calcul parallèle, il existe une autre version légèrement différente : **la loi de Gustafon** [Gustafon, 1988] :

Soient T_n , le temps CPU d'un programme parallèle sur n processeurs et p la fraction du code parallélisable, alors, le temps CPU d'exécution sur un seul processeur est donné par :

$$T_1 = (1-p)T_n + npT_n \tag{4.5}$$

Donc, l'accélération A_n et l'efficacité E_n sont données respectivement par :

$$A_n = (1-p) + np \tag{4.6}$$

$$E_n = \frac{1-p}{n} + p \tag{4.7}$$

A partir de la loi de Gustafon, nous retrouvons une autre notion utilisée chez les spécialistes du calcul parallèle : **extensibilité** (ou le terme anglophone "scalability"). Cette notion traduit le comportement d'un algorithme parallèle en fonction du nombre de processeurs. Nous pouvons, donc, soit augmenter le nombre de processeurs tout en préservant

la même taille du problème, ou bien augmenter la taille du problème proportionnellement au nombre de processeurs. Et c'est ce dernier choix qui est le plus avantageux puisque l'on conserve une charge constante par processeur.

Nous présentons maintenant la bibliothèque de passage de messages "**MPI**" qui nous a servi en grande partie, d'outil de parallélisation pour cette thèse.

4.5 Présentation de la bibliothèque de passage de messages MPI

4.5.1 Introduction et généralités

L'environnement de passage (ou d'échange) de messages est l'outil le plus utilisé pour la programmation des machines parallèles à mémoire distribuée. Cela revient, d'une part, à la simplicité de ce modèle puisque c'est la manière la plus proche de ce qui se passe réellement dans ces machines et d'autre part, à son implantation efficace sur une grande variété d'ordinateurs. En outre, les envois-réceptions de messages se font grâce à des procédures stockées dans une librairie accessible depuis un code Fortran ou C, ce qui reste familier au programmeur.

Au cours des 5 dernières années, la bibliothèque PVM (**P**arallel **V**irtual **M**achine) [Geist et al., 1993] et l'interface normalisée MPI (**M**essage **P**assing **I**nterface) se sont imposées comme des standards. Cependant, dans ce mémoire, nous nous limiterons au MPI.

La librairie de passage de messages MPI est issue d'un consortium créé en avril 92 et réunissant des représentants du monde académique et du monde industriel. Ils ont décidé d'adopter les structures et les méthodes du groupe de travail HPF. Puis, un brouillon de la première version MPI.1 a été présenté à la fin de l'année 93 lors du "*Supercomputing'93*" et ensuite, la première version MPI.1 a été achevée en avril 94 dans le rapport du forum MPI [Dongara et al., 1994]. Cette version de MPI vise surtout la portabilité et la garantie de bonnes performances.

4.5.2 Les caractéristiques de MPI

Une application MPI est un ensemble de processsus autonomes, qui exécutent chacun leur code et communiquent par des appels à des sous-programmes de la bibliothèque MPI. Ces sous-programmes sont classés selon les grandes catégories suivantes :

a - Gestion de l'environnement

Il existe des interfaces en Fortran ou C et les fonctions présentées ci-contre sont celles du Fortran. Au lancement de MPI, les processus sont membres d'un communicateur par

défaut `MPI_COMM_WORLD`.

`MPI_INIT` : initialise toute application MPI.

`MPI_FINALIZE` : désactive l'environnement MPI et une fois lancé, aucun autre sous-programme MPI ne peut être appelé. Le programmeur, en l'occurence, doit s'assurer que les communications impliquées dans un processus ont été effectuées avant `MPI_FINALIZE`.

`MPI_COMM_SIZE` : renvoie la valeur du nombre de processeurs (`nb_procs`) gérés par un communicateur donné.

`MPI_COMM_RANK` : permet d'obtenir le rang d'un processus (ce rang est compris entre 0 et nb_procs-1).

b - Communications point à point

Une communication, dite **point à point**, a lieu entre un processus emetteur et un processus récepteur qui sont identifiés par leur **rang** dans le communicateur. Les communications point à point de base sont :

`MPI_SEND` : envoie un message.

`MPI_RECV` : reçoit un message.

Ces deux sous-programmes comportent chacun une "*enveloppe du message*" qui renseigne sur les rangs des processeurs emetteur et récepteur, une étiquette du message et le nom du communicateur auquel ils appartiennent.

c - Communications collectives

Une série de communications point à point peut se faire en une seule opération grâce à des **communications collectives**. Les plus importantes fonctions sont :

`MPI_BARRIER` : c'est une "*barrière*" qui synchronise globalement toutes les tâches.

`MPI_BCAST` : diffuse des données à tous les processeurs.

`MPI_SCATTER` : distribue des données d'une tâche à toutes les autres: chacune reçoit des données "personnelles".

`MPI_GATHER` : permet la collecte des données de toutes les tâches en une seule.

`MPI_ALLGATHER` : permet la collecte des données de toutes les tâches par toutes les tâches.

`MPI_REDUCE` : effectue des opérations globales (somme, produit, max,...) auxquelles prennent part tous les processeurs. Elle assure, en plus, la synchronisation des tâches.

`MPI_ALLREDUCE` : est une fonction de réduction avec diffusion : en fait, c'est `MPI_REDUCE` suivi de `MPI_BCAST`.

Deux dernières fonctions de MPI, sur les quelles nous n'insisterons pas, sont, notamment, la gestion des groupes et des communicateurs et la définition de topologies.

Pour la première, il s'agit de diviser un ensemble de processus pour créer des sous-ensembles, qui auront chacun son espace de communication ou bien de construire de nouvelles communications dans le communicateur initial. Concernant la deuxième fonction, il est intéressant de pouvoir disposer les processeurs suivant une topologie régulière et cela est très utile pour les méthodes de décomposition de domaine puisque l'on est capable, moyennant la définition de nouvelles topologies, de décomposer le domaine de calcul sur l'ensemble des processeurs sous forme de grille par exemple. Chaque processeur dans ce cas, est identifié par ses coordonnées dans la grille.

4.5.3 Critiques de MPI

Certains buts visés par la création de MPI.1 ont été effectivement atteints. Cependant, restent encore quelques propriétés à ajouter à cette première version, ce qui a conduit à MPI.2 (2ème version de MPI) dont un brouillon a été présenté au "*supercomputing'96*" suite à des travaux entamés au printemps 95. Les principaux objectifs de MPI.2 résident dans : la gestion dynamique des processus, la communication mémoire à mémoire, la gestion des entrées/sorties pour une bonne flexibilité et enfin l'amélioration des communications collectives.

Conclusion

MPI est une librairie conviviale et elle offre une interface facile à l'implantation d'algorithmes parallèles. Cependant, nous attendons un peu plus de son avenir, entre autres, la mise en place de débogueur et analyseurs de performances portables ainsi que l'introduction de la programmation orientée objet avec une interface "C^{++} / Fortran".

Chapitre **5 :**

Problème-test et résultats numériques.

Nous présentons sur un problème-test, les applications des algorithmes développés au chapitre précédent, comparés aux résultats sur le domaine global. Leurs analyses et des résultats de leur implantation sur une machine parallèle à l'aide de la librairie MPI sont exposés.

5.1 Le problème test

5.1.1 Présentation du problème sur le domaine global

Dans ce chapitre, nous reprenons un problème-test donné dans [Bensoussan et al., 1973]. Cela nous permet de faire une comparaison entre les résultats obtenus par l'algorithme proposé dans leur étude et ceux de notre algorithme. Nous signalons également que le même exemple a été repris dans [Benamou, 1994] et dont les conditions de transmission sont proches des nôtres.

Soit le problème de contrôle "*Frontière*" sur un domaine rectangulaire Ω de $\mathbb{R}^2$ (voir figure 5.1) dont l'équation d'état est donnée par :

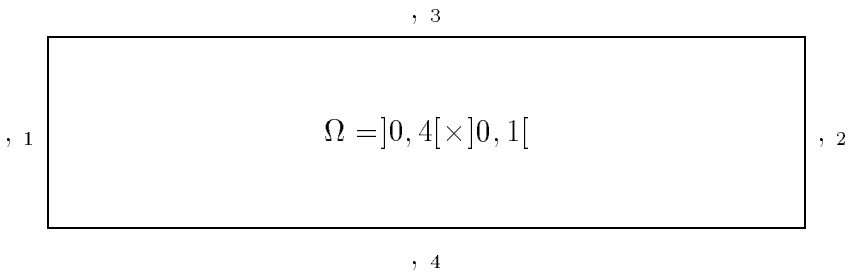

FIG. 5.1 – *Domaine de Calcul* Ω

$$\left\{\begin{array}{lcll} -\Delta y(v) & = & f & \text{dans } \Omega, \\ y & = & 0 & \text{sur } \Gamma_3 \cup \Gamma_4, \\ \dfrac{\partial y}{\partial \eta} & = & v & \text{sur } \Gamma_1 \cup \Gamma_2, \end{array}\right. \tag{5.1}$$

où

$$\begin{array}{lcl} \Omega & = &]0,4[\times]0,1[, \\ \Gamma_1 & = & \{0\}\times]0,1[,\ \Gamma_2 = \{4\}\times]0,1[, \\ \Gamma_3 & = &]0,4[\times\{0\},\ \Gamma_4 =]0,4[\times\{1\}, \\ v & = & (v_1,v_2) \in L^2(\Gamma_1)\times L^2(\Gamma_2), \end{array}$$

et pour tout (x,y) de Ω,

$$\begin{array}{lcl} f(x,y) & = & 2(-x^2-y^2+4x+y), \\ y_d(x,y) & = & (x^2-4x)(y-y^2)-8\nu. \end{array}$$

La fonction coût est définie par :

$$J(v) = \frac{1}{2}\left(\int_\Omega |y(v)-y_d|^2 dx + \nu \int_{\Gamma_1\cup\Gamma_2} |v|^2 d\sigma\right). \tag{5.2}$$

Donc, sur le domaine global, le problème de contrôle optimal est le suivant :

$$\inf_{v\in\mathcal{U}_{ad}} J(v) = J(u) \quad \text{et } u \in \mathcal{U}_{ad}, \tag{5.3}$$

$\mathcal{U}_{ad}$ étant un convexe fermé de $L^2(\Gamma_1)\times L^2(\Gamma_2)$.

Tout d'abord, d'après les résultats du chapitre 2., le système d'équations aux dérivées partielles (5.1) admet une solution unique $y(v)$ dans $H^1(\Omega)$ qui dépend continûment de la variable de contrôle v.
Ensuite, il est évident que J est Gâteaux-dérivable et strictement convexe (car elle est ν-convexe), donc, elle admet un minimum unique dans $\mathcal{U}_{ad}$.

Dans la suite de ce premier paragraphe, nous présentons la méthode utilisée pour résoudre le problème-test sur le domaine global. La partie de discrétisation concernant la minimisation de la fonction coût sera également utilisée et reprise dans le cas de la décomposition du domaine.

5.1.2 Optimisation numérique

La résolution numérique du problème de contrôle optimal utilise les techniques d'une fonction coût qui minimise l'écart quadratique au sens des moindres carrés entre la solution de l'équation d'état et des observations données.

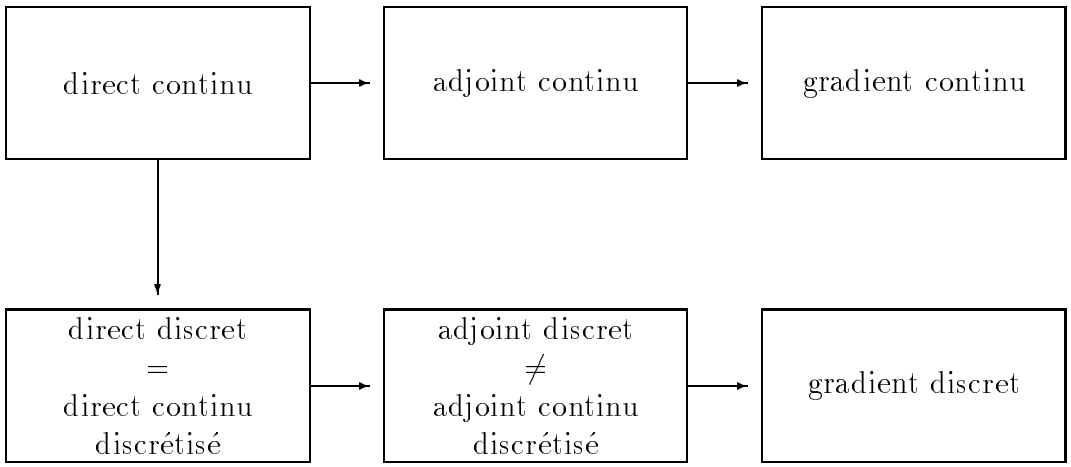

FIG. 5.2 – *Relation modèle continu-modèle discret*

Pour résoudre ce problème de minimisation, nous utilisons des algorithmes qui reposent en général sur des méthodes de descente et plus précisément sur la recherche d'une direction de descente. Explicitement, les étapes de telles méthodes se résument par : on choisit arbitrairement un point de départ dans l'espace de minimisation, ensuite, le but de l'algorithme est de construire une suite de points qui converge vers le point de contrôle optimal en faisant diminuer rapidement la fonctionnelle calculée en ces points. La direction qui relie deux points successifs de cette suite de points est dite "direction de descente". Elle est calculée aux différents points, à partir du gradient et des valeurs de la fonctionnelle. Il faut préciser que plus le calcul de cette direction est exact, plus l'algorithme est efficace et par conséquent la convergence est atteinte en moins d'itérations. Donc, il est indispensable de calculer le plus exactement possible le vecteur gradient discret.
Dans le cas du problème de contrôle optimal gouverné par des équations aux dérivées partielles, nous avons vu que, moyennant une formulation Lagrangienne, le gradient de la fonction coût se calcule de façon très simple à partir de l'état adjoint. Encore faut-il préciser que le gradient que nous calculons est le gradient continu, différent du gradient discret exigé pour l'optimisation numérique. Donc, le **gradient discret** doit être calculé en fonction de l'**état adjoint discret** et non pas de l'**état adjoint continu discrétisé**. Nous résumons cette relation entre modèles continus discrétisés et modèles discrets sur le diagramme 5.2.

Par ailleurs, la programmation de l'état adjoint discret "exige le plus grand soin" de la part du développeur afin d'avoir une bonne transposition du "code direct discret".

Il existe actuellement des logiciels conçus pour calculer automatiquement le code adjoint discret à partir du code direct. Nous citons ici, le logiciel de différentiation automatique "*Odyssée*", développé dans le projet SAFIR, à l'INRIA Sophia-Antipolis. Il a été récemment utilisé dans un modèle météorologique Meso-NH au sein du projet IDOPT.

Comme algorithmes de minimisation, nous avons utilisé les deux méthodes vues au chapitre 2. (paragraphe 2.3.1), à savoir, l'algorithme de gradient conjugué et la méthode de quasi-Newton. Et à chaque itération de l'un de ces algorithmes, il faut calculer les états direct et adjoint discrets. Cela est présenté dans le paragraphe suivant.

5.1.3 Discrétisation

Pour revenir à la résolution numérique de notre problème-test, nous utilisons, pour discrétiser l'état direct, un schéma aux différences finies à cinq points. Les conditions aux limites Neumann sur les bords Γ_1 et Γ_2 et qui correspondent aux contrôles, sont traitées par la méthode dite des "*points fictifs*".

Explicitement, dans le cas d'une discrétisation du rectangle Ω en $Nx+1$ (resp. $Ny+1$) points dans la direction x (resp. y) (voir figure 5.3 dans le cas d'une grille régulière, i.e., $hx = hy$), nous posons :

$hx = \dfrac{1}{Nx}$: est le pas de discrétisation dans la direction x,

$hy = \dfrac{1}{Ny}$: est le pas de discrétisation dans la direction y.

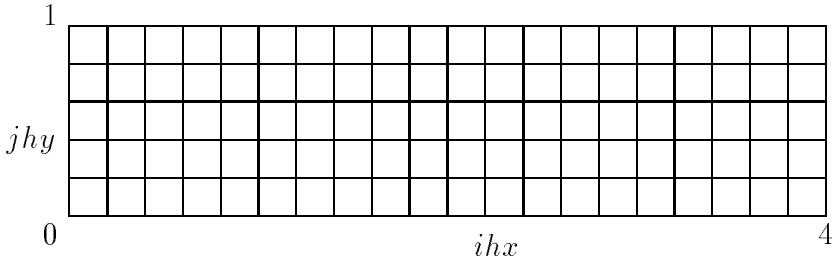

FIG. 5.3 – *Discrétisation de* Ω

Donc, l'état direct discret $y_{ij} = y(ihx, jhy)$ vérifie :

- $\underline{i = 2,\ Nx,\ j = 2,\ Ny}$:

$$-y_{ij} + \frac{hy^2}{2(hx^2+hy^2)}(y_{i-1j} + y_{i+1j}) + \frac{hx^2}{2(hx^2+hy^2)}(y_{ij-1} + y_{ij+1}) = -\frac{hx^2hy^2}{2(hx^2+hy^2)}f_{ij}.$$

- $\underline{i = 1,\ j = 2,\ Ny}$:

$$-y_{ij} + \frac{hy^2}{(hx^2+hy^2)}y_{i+1j} + \frac{hx^2}{2(hx^2+hy^2)}(y_{ij-1} + y_{ij+1}) = -\frac{hx^2hy^2}{2(hx^2+hy^2)}f_{ij} - \frac{hxhy^2}{(hx^2+hy^2)}v_{1j}.$$

- $\underline{i = Nx+1,\ j = 2,\ Ny}$:

$$-y_{ij} + \frac{hy^2}{(hx^2+hy^2)} y_{i-1j} + \frac{hx^2}{2(hx^2+hy^2)}(y_{ij-1} + y_{ij+1}) = -\frac{hx^2hy^2}{2(hx^2+hy^2)} f_{ij} - \frac{hxhy^2}{(hx^2+hy^2)} v_{2j}.$$

- $\underline{i = 1, Nx+1}$:

$$y_{i1} = 0 \quad \text{et} \quad y_{iNy+1} = 0$$

En résumé, l'état direct discret est donné par la résolution du système linéaire

$$AY = F \tag{5.4}$$

où, A est une matrice creuse dont les composantes sont précisées dans la discrétisation ci-dessus et F, le second membre de ce système est fonction des f_{ij}, v_{1j} et v_{2j}.

Puis, afin d'obtenir le système vérifié par l'état adjoint discret, il suffit de faire une transposition du système (5.4) et on obtient un système de la forme

$$A^t P = B \tag{5.5}$$

le second membre B est fonction de $(Y - Y_d)$. Une étude détaillée de cette étape est donnée dans [Luong, 1995].

On obtient enfin à partir du calcul de Y et P les valeurs de la fonction coût discrète et du vecteur gradient discret.

5.1.4 Sur la résolution des systèmes linéaires direct et adjoint discrets

Nous rappelons que pour la résolution du problème de contrôle optimal, nous avons testé deux méthodes différentes : La méthode de gradient conjugué et une méthode de Quasi-Newton (*cf.* Chapitre 2.).
Ainsi, afin de calculer la valeur de la fonction coût et le vecteur gradient (nécessaires aux deux optimiseurs ci-dessus), nous avons à résoudre les deux systèmes linéaires des états direct et adjoint discrets pour chaque simulation du problème. Pour cela, nous avons utilisé une méthode itérative, qui ne demande généralement que des produits matrices-vecteurs. L'algorithme employé est : `Bi-CGSTAB` (Bi-Conjugate Gradient STABilized) dont nous donnons un aperçu dans l'annexe A.

Degrés de liberté	Nombre d'itérations Quasi-Newton	Temps CPU (s)	Mémoire (Mo)
8×32	7	négligeable	0.5
16×64	10	2	0.8
32×128	11	19	2
64×256	11	199	7.2

TAB. 5.1 – *Résultats du domaine global pour différents maillages en utilisant le solveur* `BiCGSTAB` .

5.1.5 Résultats du domaine global

Nous présentons ici les tests faits sur le domaine global en utilisant la méthode quasi-Newton (*cf.* paragraphe 2.3.1). le critère d'arrêt choisi pour les tests de cette section est:

$$\frac{\|g_k\|}{\|g_0\|} \leq \epsilon_{grad} = 10^{-6} \tag{5.6}$$

où g_k est le gradient après k itérations de l'optimiseur et g_0 est le gradient à l'état initial. Sur le tableau 5.1, nous donnons pour plusieurs maillages, les nombres d'itérations ainsi que les temps CPU calculés sur une station SUN. La première remarque à faire est la croissance rapide du temps de calcul en fonction de la dimension de la grille de la discrétisation. Nous observons également que la mémoire nécessaire à ce calcul est une fonction croissante du maillage du domaine. Le nombre d'itérations reste, pour sa part, pratiquement constant.
Donc, à première vue de ces résultats, décomposer le problème est une stratégie efficace afin de prendre des tailles plus grandes.

5.2 Décomposition du domaine

5.2.1 Implantation parallèle des différents algorithmes

Tous les programmes parallèles testés dans ce chapitre sont écrits en Fortran 77 et utilisent la bibliothèque **MPI** pour les échanges de messages. L'avantage de cette solution est la portabilité du code. En outre, comme les compilateurs Fortran sont très efficaces sur toutes les machines, cela permet d'atteindre de bonnes performances.

Nous rappelons que la décomposition du domaine est une décomposition sans recouvrement. Par ailleurs, il existe différentes façons de partager le rectangle Ω, soit en bande (voir figure 5.5), soit en une grille (voir figure 5.6).

Le choix de la librairie MPI est très avantageux dans la mesure où MPI contient des sous-programmes qui permettent d'avoir une disposition des processeurs en grille et

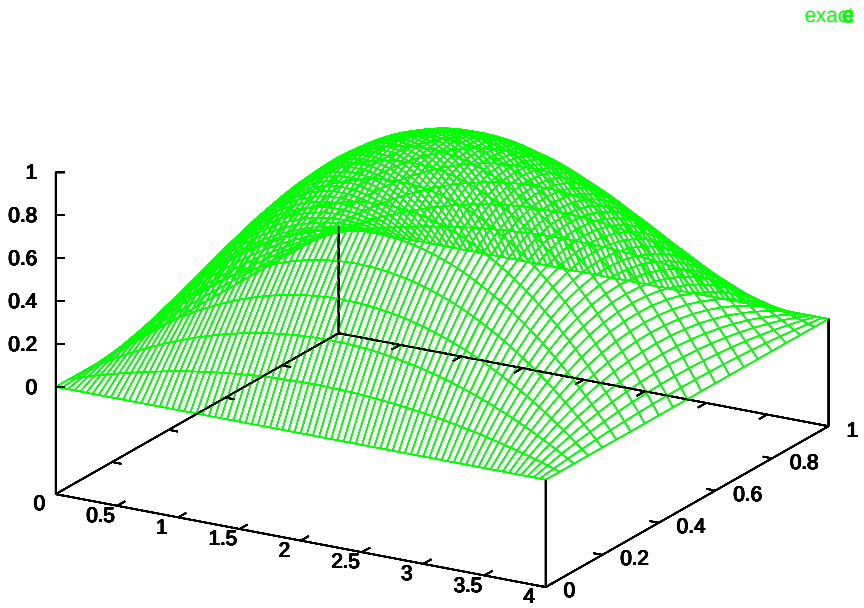

FIG. 5.4 – *Solution exacte sur le domaine global.* $N = 20$

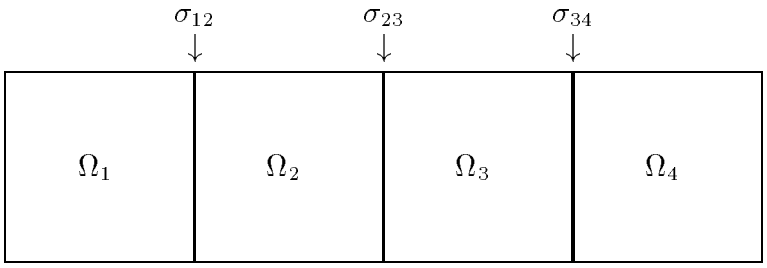

FIG. 5.5 – *Décomposition sans recouvrement de* Ω *en 4 domaines en* `bande`.

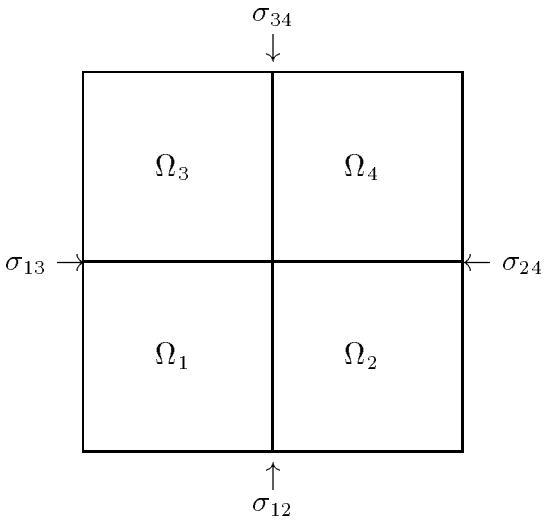

FIG. 5.6 – *Décomposition sans recouvrement de* Ω *en 4 domaines en* `grille`.

chaque processeur est repéré par ses coordonnées dans cette grille (la disposition en bande en est un cas particulier : les coordonnées des processeurs étant de la forme (x_procs, 0)). Le programmeur n'a qu'à préciser le nombre de processeurs dans chaque direction. En plus, il est possible d'avoir un communicateur partiel qui ne fait intervenir que le processeur en question et ses voisins Nord, Sud, Est et Ouest dans la grille.

Dans tous les tests que nous avons effectués, le nombre de processeurs coïncide avec le nombre de sous-domaines. Ainsi, chaque sous-domaine est distribué sur un processeur. Dans le cas de notre problème test, le contrôle est pris en compte sur les deux bords Est et Ouest de Ω, cela représente, pour l'algorithme de minimisation, un tableau de $2(Ny+1)$ alors que les systèmes directs et adjoints sont de la même taille avec $(Nx+1)(Ny+1)$ degrés de liberté chacun. Nous remarquons que la part de minimisation est faible en temps de calcul comparée à celle de la fonction coût et du gradient à partir des états directs et adjoints, ce qui représente une simulation du problème de contrôle optimal. Pour cela, nous avons gardé une minimisation globale de telle sorte que l'on ait un processeur maître qui est chargé de diriger la minimisation et il regroupe toutes les données dont il a besoin de chez les autres processeurs esclaves.

Algorithme de résolution:

Etape 0. Initialisation :
Pour tout $i \in I$, ω_{ij}^1, λ_{ij}^1 et q_{ij}^0 sont pris arbitrairement dans $L^2(\sigma_{ij})$.

Etape 1. Résolution de problèmes de contrôle locaux :

avec ω_{ij}^n, λ_{ij}^n et q_{ij}^{n-1} connus, pour chaque $i \in I$, on calcule u_i^n, y_i^n et p_i^n par :

$$\left\{\begin{array}{lcll} -\Delta y_i^n & = & f_i & \text{dans } \Omega_i, \\ y_i^n & = & 0 & \text{sur } \ , _i, \\ \dfrac{\partial y_i^n}{\partial \eta} & = & u_i^n & \text{sur } (\ , _1 \cup \ , _2) \cap \partial\Omega_i, \\ \dfrac{\partial y_i^n}{\partial \eta} & = & \omega_{ij}^n & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{array}\right. \tag{5.7}$$

$$\left\{\begin{array}{lcll} -\Delta p_i^n & = & y_i^n - y_{d,i} & \text{dans } \Omega_i, \\ p_i^n & = & 0 & \text{sur } \ , _i, \\ \dfrac{\partial p_i^n}{\partial \eta} & = & 0 & \text{sur } (\ , _1 \cup \ , _2) \cap \partial\Omega_i, \\ \dfrac{\partial p_i^n}{\partial \eta_{ij}} & = & \lambda_{ij}^n + r(y_i^n - q_{ij}^{n-1}) & \text{sur } \sigma_{ij} \quad (j \in J_i). \end{array}\right. \tag{5.8}$$

$$\int_{\Omega_i} (p_i^n + \nu u_i^n)(v_i - u_i^n)dx \geq 0 \qquad \forall v_i \in \mathcal{U}_{ad}|_{\Omega_i}. \tag{5.9}$$

Etape 2. Mise à jour des multiplicateurs :

avec ρ_ω et ρ_λ deux réels strictement positifs, ω_{ij}^{n+1}, $\lambda_{ij}^{n+\frac{1}{2}}$, q_{ij}^n et λ_{ij}^{n+1} sont donnés par :

$$\begin{array}{lcll} \omega_{ij}^{n+1} & = & \omega_{ij}^n + \rho_\omega(p_i^n - p_j^n) & \text{sur } \sigma_{ij}, \\ \lambda_{ij}^{n+\frac{1}{2}} & = & \lambda_{ij}^n + \rho_\lambda(y_i^n - q_{ij}^{n-1}) & \text{sur } \sigma_{ij}. \\ 2rq_{ij}^n & = & (\lambda_{ij}^{n+\frac{1}{2}} + \lambda_{ji}^{n+\frac{1}{2}}) + r(y_i^n + y_j^n) & \text{sur } \sigma_{ij}, \\ \lambda_{ij}^{n+1} & = & \lambda_{ij}^{n+\frac{1}{2}} + \rho_\lambda(y_i^n - q_{ij}^n) & \text{sur } \sigma_{ij}. \end{array} \tag{5.10}$$

Faire $n \longleftarrow n+1$ et aller à Etape 1.

Donc, dans le cas d'une décomposition en bande, le processeur maître est représenté par le premier processeur (qui est associé au premier sous-domaine). L'initialisation se fait en parallèle sur tous les processeurs, ensuite, pour résoudre le problème de contrôle optimal, débute alors, la minimisation dont une itération se déroule comme suit :

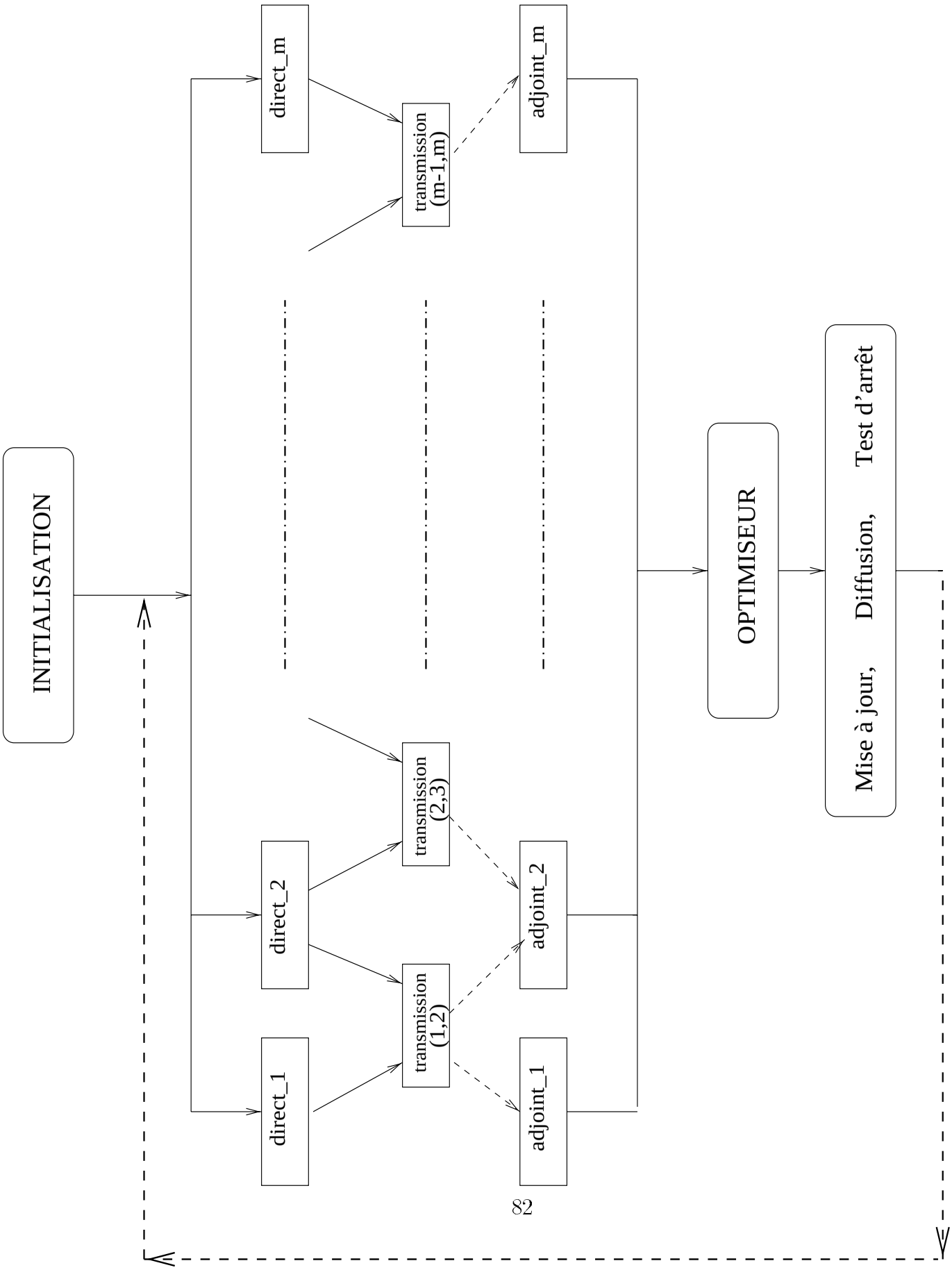

FIG. 5.7 – *Diagramme de l'algorithme de résolution.*

5.2.2 Description de la résolution de l'étape 1.

A la $n$ième itération de l'algorithme ci-dessus, le point de contrôle u^n, étant connu, il est ensuite décomposé en u_1^n qui reste sur le premier processeur (ou processeur maître) et en u_2^n qui est envoyé au dernier processeur. Puis, tous en parallèle, les processeurs

1- effectuent chacun le calcul de l'état direct local y_i^n par la résolution du système,

2- s'échangent les valeurs des y_i^n sur les interfaces communes,

3- calculent la valeur de la fonction coût locale,

4- effectuent le calcul de l'état adjoint local p_i^n.

A ce stade, le processeur maître reçoit du dernier processeur esclave le bloc de p_i^n correspondant à , $_2$ pour calculer le gradient de la fonctionnelle.... Ensuite, une nouvelle direction de descente et un nouveau itéré $u_i^{n+1},\ i = 1, 2$ sont calculés et ainsi de suite.

Encore faut-il préciser que nous avons adopté deux stratégies permettant de rendre compte des influences des hiéarchies dans les calculs des nouveaux contrôles et la mise à jour des multiplicateurs de Lagrange, ce qui est représenté dans les alogorithmes (`Lag_ddm.i`), $i = 1, 2, 3$ (*cf.* chapitre 3.) par la résolution des problèmes de contrôle locaux:

(a) Pour la première stratégie, nous considérons des itérations internes au niveau de la résolution des problèmes de contrôle optimal locaux tandis que l'itération externe se situe au niveau de la mise à jour des multiplicateurs de Lagrange. Explicitement, on se fixe un test d'arrêt dans l'optimiseur et une fois un nouveau vecteur de contrôle est calculé (ce qui est représenté par l'étape 1. de l'algorithme ci-dessus), on effectue une mise à jour des multiplicateurs ω_{ij} et λ_{ij} (étape 2. de l'algorithme) et on réitère le processus. Cela est résumé dans le diagramme de la figure 5.7.

(b) Quant à la deuxième stratégie, nous effectuons une seule itération générale qui englobe la recherche d'une nouvelle descente dans l'algorithme de minimisation et dès que nous avons un nouveau point de contrôle, nous enchaînons par une mise à jour immédiate des multiplicateurs de Lagrange. Dans ce cas, nous n'avons pas d'itération interne ni externe mais une seule itération globale. D'une autre manière, l'étape 2. de l'algorithme de résolution fait partie d'une itération de minimisation de l'étape 1.

5.2.3 Résultats du domaine décomposé

Lorsque nous mettons en œuvre les algorithmes présentés au chapitre 2., pour le cas d'une décomposition en deux sous-domaines, le premier avantage est que nous pouvons maintenant résoudre un problème un peu plus grand.

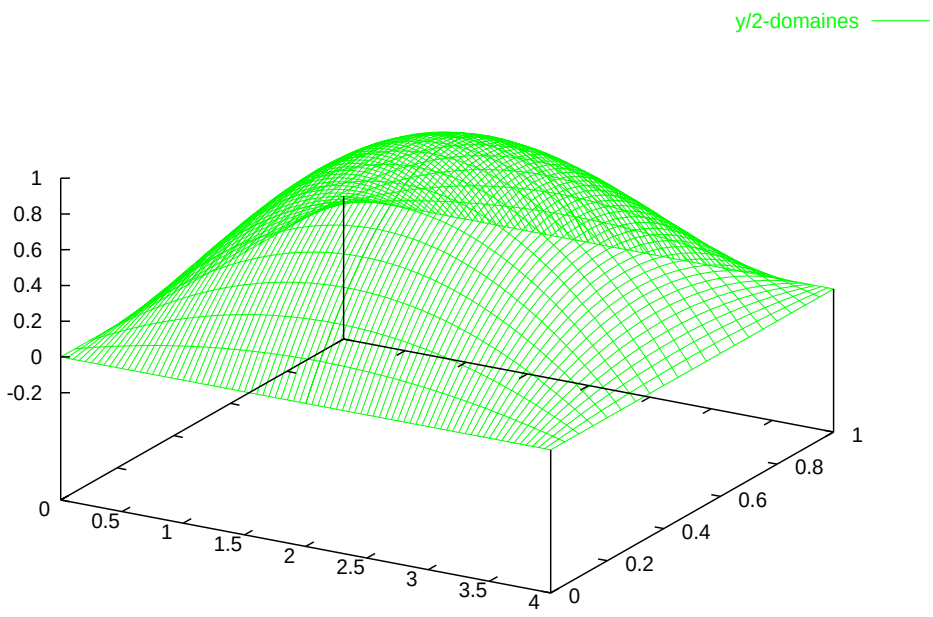

FIG. 5.8 – *Solution sur le domaine décomposé en deux sous-domaines, l'interface étant prise au milieu.* $N = 20$

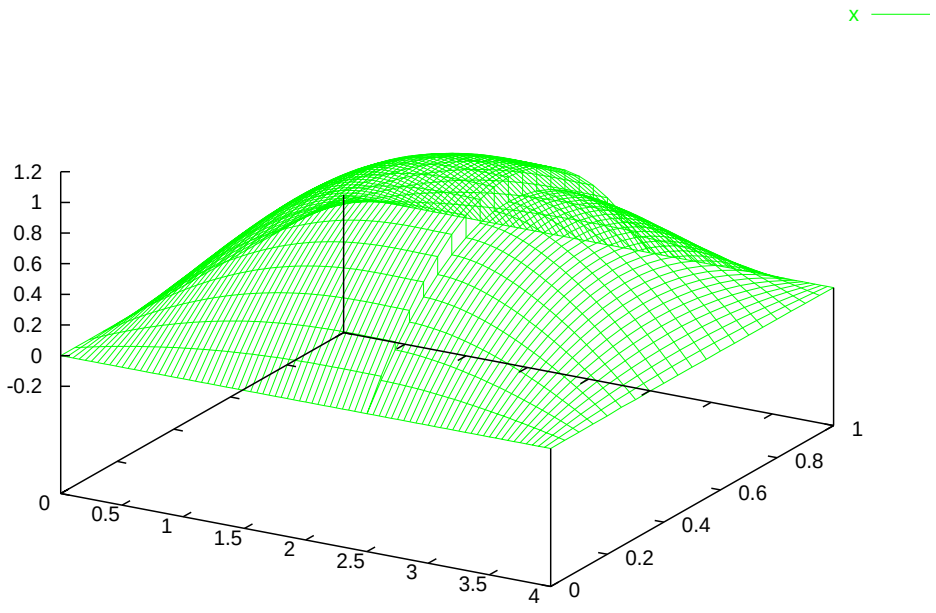

FIG. 5.9 – *Solution du problème décomposé sur 2 domaines avec* `Lag-ddm.3`.

Si nous revenons au cas de notre problème-test, nous remarquons qu'il est symétrique par rapport à l'interface quand elle est prise au milieu du domaine global. Les multiplicateurs de Lagrange sont initialisés à zéro. Dans ce cas, les résultats sont satisfaisants et la solution obtenue sur deux sous-domaines est la même que celle calculée sur le domaine globale. La solution calculée, c'est à dire, l'état direct associé au contrôle optimal, est présentée sur la figure 5.8 et la continuité est bien assurée sur l'interface.

Ensuite, pour éliminer ce cas particulier, l'interface est décalée du milieu du domaine. Les résultats dans ce cas, sont moins encourageants: la première contrainte à imposer est de prendre un test d'arrêt dans l'optimiseur qui ne soit pas "trop sévère" (par rapport au choix dans le cas du domaine global ou du cas où l'interface est au milieu du domaine) $\frac{\|g_k\|}{\|g_0\|} \leq \epsilon_{grad} = 10^{-4}$. Relativement à ce test, la solution final présente un "*saut*" au voisinage de l'interface, voir figure 5.9. Ce résultat est obtenu lorsque nous utilisons `Lag_ddm.3` et nous précisons que les deux algorithmes `Lag_ddm.i`, i=1 ou 2, donnent des résultats très proches. Cela montre que nous n'avons pas pu raccorder au mieux les données sur l'interface. Néanmoins, le seul résultat (favorable!) que nous remarquons est $|\frac{y_i + y_j}{2} - y_{exact}|_{L^2(\sigma_{ij})} \sim o(10^{-4})$ pour $N = 20$; dans ce cas, le maillage du domaine global étant de $4N \times N$. Cette erreur obtenue est proche d'un résultat de convergence lorsque l'on utilise les conditions de transmission de type Robin : il figure dans l'algorithme de Lions, voir le théorème 1.2.1 (*cf.* [Lions, 1990]).

Nous utilisons de la remarque sur la demi-somme des états directs discrets sur l'interface, pour calculer la fonction coût globale.
Cette démarche est appliquée aussi quand le domaine est décomposé en 4 sous-domaines (en bande). L'erreur $\sum_i \sum_j |\frac{y_i + y_j}{2} - y_{exact}|_{L^2(\sigma_{ij})}$, cette fois, est de l'ordre de 10^{-2}: La précision a baissé. Nous en déduisons que le nombre de sous-domaines influence nettement sur les résultats de ces tests.

Nous considérons une autre stratégie: le coefficient "**r**" du Lagrangien augmenté, n'étant pas très grand "de l'ordre de 10^{-2}", nous ajoutons une boucle interne de 2 itérations par rapport à la mise à jour des multiplicateurs de Lagrange: ω_{ij} et les λ_{ij}. Nous obtenons donc une nouvelle version de l'algorithme `Lag-ddm.3` que nous avons utilisée pour obtenir les résultats de la figure 5.10. Une simulation de notre problème test, pour cette nouvelle version, se déroule de la manière suivante dans le cas de 2 sous-domaines :

- $u^n \longrightarrow [u_1^n, u_2^n]$, ω^n, λ_i^n et q^{n-1} connus,

$\star$ Itération interne :
$iter = iter + 1$
$r = \frac{r}{2}$

- calcul de $y_1^n(u_1^n, \omega^n)$ et $y_2^n(u_2^n, \omega^n)$,

- calcul de $J(u_1^n, y_1^n, u_2^n, y_2^n) + \sum_{i=1.2} [(\lambda_i^n, y_i^n - q^{n-1}) + \frac{r}{2}|y_i^n - q^{n-1}|^2]$,

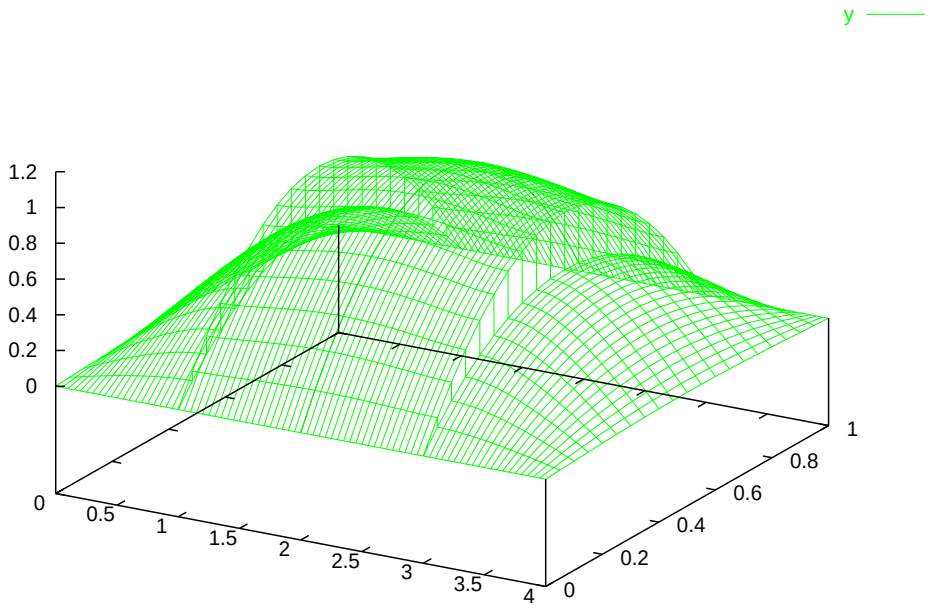

FIG. 5.10 – *Solution du problème décomposé sur 4 domaines avec* `Lag-ddm.3`.

où la valeur des y_i^n sur l'interface est la demi-somme $\dfrac{y_i + y_j}{2}$

- calcul de p_1^n et p_2^n,

- $$\left\| \begin{array}{lcl} q^n & = & (\lambda_1^n + \lambda_2^n)/2/r + (y_1^n + y_2^n)/2 \\ \lambda_1^{n+1} & = & \lambda_1^n + r(y_1^n - q^n) \\ \lambda_2^{n+1} & = & \lambda_2^n + r(y_2^n - q^n) \\ \omega^{n+1} & = & \omega^n + r(p_1^n - p_2^n) \end{array} \right.$$

  Retourner à $\star$.

Remarque 5.2.1 *Résoudre le problème $(\mathcal{P})_{dec}$ revient aussi à minimiser la fonctionnelle $\sum_{i=1}^{m} J_{dec}^i$ sous les contraintes données par les systèmes locaux $(\mathcal{D}ir)_i$ et la continuité aux interfaces (voir chapitre 3.).*
Cette minimisation est également équivalente à la minimisation de la fonctionnelle décomposée à laquelle nous avons ajouté un terme pénalisant, c'est à dire, avec $\alpha > 0$, nous avons

$$Minimiser \quad \sum_{i=1}^{m} J_{dec}^i(v_i, y_i) + \frac{\alpha}{2} \sum_{j, \sigma_{ij} \neq \emptyset} |y_i - y_j|_{ij}^2 \tag{5.11}$$

Cependant, il n'est pas facile, au vu des tests numériques, de choisir le paramètre α de façon optimale. Par exemple, dans le cas d'une décomposition en deux sous-domaines, l'influence de α n'est pas grande, mais, ce coefficient doit rester dans un borné.

5.3 Conclusion

Les résultats obtenus dans ce chapitre sont mitigés. En effet, les algorithmes que nous avons testés dans ce chapitre mettent en jeu plusieurs variables supplémentaires :

- Tout d'abord, nous ne disposons d'aucun moyen de contrôler les ω_{ij} introduites pour raccorder le flux de l'état direct sur les interfaces entre les sous-domaines: d'une part, nous ne pouvons faire un bon choix pour leur initialisation et d'autre part, leur mise à jour à l'aide d'un algorithme de type Uzawa, ne permet pas d'avoir une idée sur la bonne manière de choisir la valeur du pas de descente de cet algorithme pour obtenir une meilleure convergence (Le choix $\rho = r$ étant le meilleur dans le cas de plusieurs problèmes traités dans la littérature).

- La même remarque peut être faite pour les multiplicateurs λ_{ij}.

- Pour tous les tests, le produit scalaire utilisé est celui de $L^2(\sigma_{ij})$. A l'instar de ce qui existe dans la littérature et plus précisément, dans les travaux de Le Tallec et Sassi

[Le Tallec et Sassi, 1995], ce choix s'avère dépendant du nombre de sous-domaines ainsi que du pas de discrétisation. Le Tallec et Sassi montrent dans le cas d'un problème aux limites en élasticité qu'un autre choix de produit scalaire (*cf.* remarque 3.6.3) et propre à leur problème permet de surmonter cette dépendance. Ce choix de produit scalaire sur les interfaces joue le rôle de préconditionneur pour le problème décomposé.

- Finalement, le coefficient r du Lagrangien augmenté a été ajusté suivant l'ordre de grandeur des données de notre problème test. Le choix d'un r optimal reste à étudier.

Chapitre **6 :**

Généralités sur un problème de contrôle optimal régi par une edp parabolique.

Nous étudions dans ce chapitre, un exemple de problème de contrôle optimal gouverné par une équation aux dérivées partielles parabolique. Sont exposés également des généralités et des résultats pour caractériser le contrôle optimal.

6.1 Résultats et généralités.

6.1.1 Préliminaires-notations

Tout d'abord, nous allons donner des notions et des généralités sur le cadre fonctionnel avant de définir notre problème de contrôle.

Soient les espaces de Hilbert V et H tels que :

$$V \subset H \subset V',$$

où V' est le dual de V alors que H est identifié à son dual. Les normes sur V et H seront notées respectivement $\|.\|$ et $|.|$. Et $[0,T]$ est l'intervalle de temps, T étant fini.

Pour chaque pas de temps t, soient les formes bilinéaires continues sur V, $a(t;.,.)$ données par :

$$\begin{array}{rcl} V \times V & \longrightarrow & \mathbb{R} \\ (\varphi, \psi) & \longrightarrow & a(t;\varphi,\psi) \end{array}$$

et qui vérifient les hypothèses suivantes :

(H1) Pour tout $(\varphi,\psi) \in V \times V$, la fonction définie par : $t \mapsto a(t;\varphi,\psi)$ est mesurable sur $]0,T[$ et de plus, il existe une constante $C > 0$ telle que

$$|a(t;\varphi,\psi)| \leq C\|\varphi\|\|\psi\|. \tag{6.1}$$

(H2) Il existe ε et α deux réels positifs tels que (coercivité élargie)

$$a(t;\varphi,\varphi)+\varepsilon|\varphi|^2 \geq \alpha\|\varphi\|^2, \quad \forall\varphi\in V \text{ et } \forall t\in[0,T]. \tag{6.2}$$

On associe à la forme bilinéaire a, un opérateur A dans $\mathcal{L}(V,V')$ tel que

$$a(t;\varphi,\psi) =< A\varphi(t),\psi(t) >_{V',V} . \tag{6.3}$$

Dans le cadre fonctionnel des problèmes paraboliques, nous précisons :

$$\begin{aligned} L^2(0,T;V) &= \{f(t) \text{ tel que } \textstyle\int_0^T \|f(t)\|_V^2 dt < +\infty\}, \\ W(0,T) &= \{f\in L^2(0,T;V) \text{tel que } \tfrac{\partial f}{\partial t}\in L^2(0,T;V')\} \end{aligned}$$

et la norme dans $W(0,T)$ est donnée par

$$\|f\|^2_{W(0,T)} = \|f\|^2_{L^2(0,T;V)} + \|f'\|^2_{L^2(0,T;V')} \tag{6.4}$$

■

Nous énonçons quelques résultats, mais, nous ne les démontrons pas ici. Le lecteur pourra trouver des détails sur les preuves dans [Raviart et Thomas, 1988] et [Lions, 1968]. Nous avons le théorème suivant :

Théorème 6.1.1 *$W(0,T)$ muni de la norme définie par (6.4) est un espace de Hilbert. De plus, toute fonction f de $W(0,T)$ est continue de $[0,T]$ dans H et pour tout φ et ψ de $W(0,T)$, nous avons*

$$\int_0^T < \varphi',\psi >_{V',V} dt = -\int_0^T < \psi',\varphi >_{V',V} dt + (\varphi(T),\psi(T))_H - (\varphi(0),\psi(0))_H$$

Nous passons donc à la définition d'un problème de contrôle optimal gouverné par une équation aux dérivées partielles parabolique donnée par le modèle direct :

6.1.2 Le modèle direct

Soit Ω un ouvert borné de $\mathbb{R}^n$ à frontière assez régulière , . Dans ce paragraphe, nous posons $V = H^1(\Omega)$ et $H = L^2(\Omega)$ et nous nous intéressons au problème évolutif suivant :

$$\begin{cases} \dfrac{\partial y}{\partial t} + Ay &= f \text{ dans } \Omega\times]0,T[, \\ y(0) &= y_0 \text{ dans } \Omega, \\ &+ \text{ Conditions aux Limites.} \end{cases} \tag{6.5}$$

avec f est dans $L^2(0,T;V')$ et y_0 est dans H.

Théorème 6.1.2 *Sous les hypothèses (H1) et (H2), le problème* (6.5) *admet une solution unique y dans $W(0,T)$. Cette solution dépend continûment des données f et y_0.*

6.1.3 Position du problème de contrôle optimal

Soient $\mathcal{U}$ l'espace de Hilbert des contrôles et $\mathcal{B}$ une application linéaire continue de $\mathcal{U}$ dans $L^2(0,T;V')$.
Supposons que les hypothèses (H1) et (H2) sont vérifiées, soit alors $y(v)$ la solution du système :

$$\begin{cases} \dfrac{\partial y}{\partial t} + Ay &= f + \mathcal{B}v \ \text{ dans } \ \Omega\times]0,T[, \\ y(0) &= y_0 \ \text{ dans } \ \Omega, \\ &+ \quad \text{Conditions aux Limites.} \end{cases} \tag{6.6}$$

$y(v)$, l'état du système, dépend de t, x et v, il s'écrit donc : $y(v) = y(t,x,v)$.
Comme conséquence du théorème 6.1.2, nous avons le résultat :

Proposition 6.1.1 *Le système* (6.6) *admet une solution unique, qui est une fonction affine continue en v.*

Toutes les notations avec un indice t veulent dire que l'on considère des problèmes dépendant du temps.

6.1.4 Observations et fonction coût

Dans le cas d'observations réparties en temps, considérons les applications linéaires: C de $L^2(0,T;V)$ dans $L^2(0,T;\mathcal{H})$, où $\mathcal{H}$ est l'espace des observations et N est une application linéaire définie sur $\mathcal{U}$, coercive et autoadjointe. Puis, y_d, représentant les observations (ou mesures), est pris dans $L^2(0,T;\mathcal{H})$. Nous définissons la fonctionnelle coût par :

$$J_t(v) = \frac{1}{2}\Big(\int_0^T \|Cy(v) - y_d\|^2_{\mathcal{H}}\,dt + \int_0^T (Nv,v)_{\mathcal{U}}\,dt\Big) \tag{6.7}$$

Enfin, le problème de contrôle optimal est donné par :

$$(\mathcal{P})_t \qquad J_t(u) = \inf_{v\in\mathcal{U}_{ad}} J_t(v), \qquad u \in \mathcal{U}_{ad}, \tag{6.8}$$

où $\mathcal{U}_{ad}$ et un convexe fermé de $\mathcal{U}$.

6.2 Sur l'existence et la caractérisation de la solution de $(\mathcal{P})_t$

Théorème 6.2.1 *Il existe une solution unique u qui minimise la fonction J_t.*
L'inéquation d'Euler pour caractériser u s'écrit:

$$(J_t'(u), v-u)_{\mathcal{U}} \geq 0, \quad \forall v \in \mathcal{U}_{ad}. \tag{6.9}$$

Dans le cas sans contraintes, nous avons $J_t'(u) = 0$: c'est l'équation d'Euler.

6.2.1 Calcul de $J_t'(u)$

D'une part, nous avons

$$\begin{aligned}(J_t'(u), v-u)_{\mathcal{U}} &= \int_0^T < C^\star \Lambda_{\mathcal{H}}(Cy(u) - y_d), y(v) - y(u) >_{V',V} \, dt \\ &+ \int_0^T (Nu, v-u)_{\mathcal{U}} dt,\end{aligned} \tag{6.10}$$

où $\Lambda_{\mathcal{H}}$ est l'isométrie canonique de $\mathcal{H}$ dans $\mathcal{H}'$.
Nous allons essayer de simplifier le premier terme de (6.10) en introduisant un état adjoint p, l'unique solution dans $W(0,T)$ du système :

$$\left\{\begin{array}{lcll} -\dfrac{\partial p}{\partial t} + A^\star p &=& C^\star \Lambda_{\mathcal{H}}(Cy(u) - y_d) & \text{dans } \Omega\times]0,T[, \\ p(T) &=& 0 & \text{dans } \Omega, \\ &+& \text{Conditions aux Limites.} & \end{array}\right. \tag{6.11}$$

C'est un problème rétrograde en temps.

Tous les opérateurs, introduits ci-dessus, sont bien définis.
En effet, $Cy(u) - y_d$ est dans $L^2(0,T;\mathcal{H})$, donc, $\Lambda_{\mathcal{H}}(Cy(u) - y_d)$ est dans $L^2(0,T;\mathcal{H}')$ et $C^\star$ est une application linéaire de $L^2(0,T;\mathcal{H}')$ dans $L^2(0,T;V')$.

D'autre part, pour revenir au calcul de $J_t'(u)$, nous utilisons la formulation faible de l'état adjoint et le résultat du théorème 6.1.1, il vient que

$$\begin{aligned}(J_t'(u), v-u)_{\mathcal{U}} &= \int_0^T < -\frac{\partial p}{\partial t} + A^\star p, y(v)-y(u) >_{V',V}\, dt + \int_0^T (Nu, v-u)_{\mathcal{U}} dt, \\ &= \int_0^T [< p, \frac{\partial}{\partial t}(y(v)-y(u) >_{V,V'} + < p, A(y(v)-y(u)) >_{V,V'}] dt \\ &+ \int_0^T (Nu, v-u)_{\mathcal{U}} dt, \end{aligned} \tag{6.12}$$

Puis, en soustrayant les formulations variationnelles de $y(v)$ et $y(u)$, le premier terme de (6.12) se réduit à :

$$\int_0^T < p, \mathcal{B}(v-u) >_{V,V'}\, dt.$$

Et comme $\mathcal{B}$ est défini de $L^2(0,T;\mathcal{U})$ dans $L^2(0,T;V')$, donc, $\mathcal{B}^\star$ est de $L^2(0,T;V)$ dans $L^2(0,T;\mathcal{U}')$. Nous considérons ensuite, l'isométrie canonique $\Lambda_{\mathcal{U}}$ de $\mathcal{U}$ dans $\mathcal{U}'$ et il en résulte la nouvelle expression du gradient de J_t en u :

$$J_t'(u) = \Lambda_{\mathcal{U}}^{-1}\mathcal{B}^\star p + Nu. \tag{6.13}$$

D'où, dans un cas simple de conditions aux limites Dirichlet homogènes, c'est à dire, $y(v) = 0$ sur $\Sigma \times]0,T[$, la caractérisation du contrôle optimal est donnée dans la proposition suivante :

Proposition 6.2.1 *La solution du problème $(P)_t$ est caractérisée par :*

$$\left\{\begin{array}{ll}
(i)\ \underline{\textbf{Etat direct}} & \left\{\begin{array}{lll} \frac{\partial y}{\partial t} + Ay = f + \mathcal{B}u & dans & Q, \\ y = 0 & sur & \Sigma, \\ y(0) = y_0 & dans & \Omega. \end{array}\right. \\
(ii)\ \underline{\textbf{Etat adjoint}} & \left\{\begin{array}{lll} -\frac{\partial p}{\partial t} + A^\star p = Cy - y_d & dans & Q, \\ p = 0 & sur & \Sigma, \\ p(T) = 0 & dans & \Omega. \end{array}\right. \\
(iii)\ \underline{\textbf{Condition d'optimalité}} & (\Lambda_{\mathcal{U}}^{-1}\mathcal{B}^\star p + Nu, v-u)_{\mathcal{U}} \geq 0,\ \forall v \in \mathcal{U}_{ad} \\ & ou \quad u = -N^{-1}\Lambda_{\mathcal{U}}^{-1}\mathcal{B}^\star p \quad cas\ sans\ contraintes
\end{array}\right.$$

Remarque 6.2.1 *(u, y, p) est aussi le point-selle (il existe et il est unique) du Lagrangien associé au problème de minimisation $(\mathcal{P})_t$. De la caractérisation de ce point-selle, résulte le même système d'optimalité ci-dessus moyennant une formulation Lagrangienne similaire à ce qui a été fait pour le cas du problème stationnaire dans la première partie.*

Chapitre **7 :**

Une méthode de décomposition de domaine sans recouvrement en espace pour un problème de contrôle optimal évolutif.

Nous abordons maintenant, dans ce chapitre, une méthode de décomposition de domaine sans recouvrement en espace pour un problème-test de contrôle optimal gouverné par l'équation de la chaleur. Une méthode de résolution est développée et des algorithmes sont proposés.

7.1 Introduction

Un plus grand nombre de travaux sur les méthodes de décomposition de domaine a été dédié aux problèmes elliptiques et linéaires. Cependant, le besoin de résoudre des problèmes de grande taille, par exemple, en mécanique des fluides et en général, divers problèmes dépendant du temps, a amené les spécialistes à se pencher sur le développement de méthodes performantes et bien adaptées aux architectures parallèles.

Comme nous l'avons mentionné dans la première partie, pour les problèmes elliptiques, les méthodes de décomposition de domaine peuvent être considérées par des techniques de décomposition d'espace; précisons que c'est de l'espace fonctionnel qu'il s'agit. La même approche peut être utilisée pour résoudre des problèmes paraboliques. En effet, dans [Tai, 1998], la décomposition de l'espace est vue de deux manières différentes :

(1) On peut discrétiser l'équation parabolique en temps et on obtient, à chaque pas de temps, une équation elliptique. Celle-ci peut donc être résolue par une décomposition d'espace : c'est "*l'approche itérative*".
Elle a été utilisée, dans un premier temps, par [Lions, 1988]. Puis, récemment, par Cai [Cai, 1991] qui a utilisé pour cela, les méthodes de Schwarz additive et multiplicative, par Dryja [Dryja, 1991] avec une méthode de sous-structuration et par Le Tallec [Le Tallec, 1994] qui a introduit l'opérateur de Schur local pour résoudre les problèmes elliptiques résultant de la discrétisation en temps.

(2) La deuxième façon est de combiner la décomposition d'espace avec le pas de temps de manière à obtenir un schéma implicite par morceaux en temps : c'est "*l'approche non itérative*", c'est à dire que l'on peut prendre des pas de temps différents sur des sous-domaines distincts.
Il est à signaler que pour une décomposition de domaine avec recouvrement, cette approche a été utilisée par Jäger, Hebecker et Kuznetsov [Jäger et al., 1992]. Dans le cas d'une décomposition sans recouvrement et en particulier, pour l'exemple de l'équation de la chaleur, Dawson, Du et Dupont [Dawson et al., 1991] ont considéré un schéma de différences finies explicite pour calculer les valeurs sur les interfaces entre les sous-domaines et puis, les points intérieurs sont calculés par une différentiation rétrograde en temps et deux des derniers auteurs ont développé, dans un autre travail [Dawson et Dupont, 1991], une méthode de Galerkin implicite-explicite et toujours après discrétisation en temps du problème.

7.2 Rappels bibliographiques

Après avoir cité quelques travaux sur le cas général de l'application des méthodes de décomposition de domaine à des problèmes dépendant du temps, nous regardons ce qui a été fait quant à leur application aux problèmes de contrôle optimal régis par des équations d'évolution. A notre connaissance, les travaux qui lui sont consacrés ne sont pas nombreux; nous en mentionnons les principaux :

Dans le cas d'un problème gouverné par une équation hyperbolique, plus précisément, l'équation des ondes, J.-D. Benamou a proposé dans [Benamou, 1997], une méthode de décomposition de domaine en espace (domaine de calcul), se basant, comme pour le cas de problèmes elliptiques, sur l'utilisation des conditions de transmission de type Robin [Lions, 1990]: à l'itération $(n+1)$ sur le domaine courant, il utilise les données du domaine voisin à l'itération n sur tout l'intervalle de temps. Et pour cela, il introduit une loi de "feed-back" et se ramène à une équation de Riccati.

Nous citons aussi deux autres travaux où il est question d'une décomposition, non pas en espace, mais une décompsition en temps :
Le premier est un récent travail de Ralph [Ralph, 1996] dans lequel il développe un algorithme parallèle pour des problèmes de contrôle optimal sans contraintes discrétisés en temps. Une décomposition par étapes de l'intervalle de temps est répartie sur un nombre donné de processeurs et les instants où la décomposition est faite sont soumis à des conditions spéciales pour préserver la continuité du problème en temps.
Le deuxième travail est de Berggren et Heinkenschloss [Berggren et Heinkenschloss, 1997]: ils ont dérivé une méthode de résolution parallèle pour un problème de contrôle évolutif en introduisant des multiplicateurs de Lagrange pour assurer la continuité entre deux temps successifs.
Le point faible de ces deux dernières méthodes est que la taille des sous-problèmes est la

même que celle du problème global puisque l'on résout les modèles direct et adjoint sur le domaine global en espace.

La méthode que nous proposons repose sur une décomposition du domaine de calcul en espace. A première vue, nous réduisons déjà la taille des états direct et adjoint lors de la résolution et nos systèmes sont découplés.

7.3 Problème modèle - Préliminaires

Nous étudions une extension de la méthode, proposée dans la partie précédente pour des problèmes elliptiques, à un problème d'évolution. Comme nous ne considérons qu'une décomposition en "espace", le développement de la méthode proposée dans ce chapitre est similaire à ce qui a été présenté.

Nous considérons le problème modèle suivant :

$$(\mathcal{D}ir)_t \qquad \left\{ \begin{array}{lcll} \dfrac{\partial y}{\partial t} - \Delta y & = & f & \text{dans} \quad \Omega \times]0,T[, \\ y & = & 0 & \text{sur} \quad ,^d \times]0,T[, \\ \dfrac{\partial y}{\partial \eta} & = & v & \text{sur} \quad ,^c \times]0,T[, \\ y(0) & = & y_0 & \text{dans} \quad \Omega. \quad \text{(condition initiale)} \end{array} \right. \tag{7.1}$$

C'est un problème de *Cauchy* avec des conditions aux limites mêlées Dirichlet-Neumann. Physiquement, le système $(\mathcal{D}ir)_t$ décrit l'évolution au cours du temps de la température d'une barre $\bar{\Omega}$ homogène sous l'action d'une source de chaleur f. Nous considérons que toutes les constantes physiques sont égales à un. Sur une partie de la frontière de la barre $,^d$, la température est fixée à zéro tandis que sur l'autre partie $,^c$, nous contrôlons le flux de la température au cours du temps.

Nous définissons également la fonction coût J_t par :

$$J_t(v) = \frac{1}{2} \int_0^T \int_\Omega |y - y_d|^2 dx dt + \nu \int_0^T \int_{\Gamma^c} |v|^2 d\sigma dt. \tag{7.2}$$

Et le problème de contrôle optimal est donné par

$$(\mathcal{P})_t \qquad \inf_{v \in \mathcal{U}_{ad}} J_t(v) = J(u), \quad u \in \mathcal{U}_{ad},$$

où $\mathcal{U}_{ad}$ est un convexe fermé de $L^2(,^c \times]0,T[)$.

Au vu des résultats du chapitre précédent, ce contrôle optimal est caractérisé par le système suivant :

$$
\left\{
\begin{array}{ll}
(i)\ \underline{\textbf{Etat direct}} & \left\{
\begin{array}{lcll}
\dfrac{\partial y}{\partial t} - \Delta y & = & f & \text{dans } \Omega\times]0,T[, \\
y & = & 0 & \text{sur } ,^{d}\times]0,T[, \\
\dfrac{\partial y}{\partial \eta} & = & v & \text{sur } ,^{c}\times]0,T[, \\
y(0) & = & y_0 & \text{dans } \Omega. \quad \text{(condition initiale)}
\end{array}
\right. \\
(ii)\ \underline{\textbf{Etat adjoint}} & \left\{
\begin{array}{lcll}
-\dfrac{\partial p}{\partial t} - \Delta p & = & y - y_d & \text{dans } \Omega\times]0,T[, \\
p & = & 0 & \text{sur } ,^{d}\times]0,T[, \\
\dfrac{\partial p}{\partial \eta} & = & 0 & \text{sur } ,^{c}\times]0,T[, \\
p(T) & = & 0 & \text{dans } \Omega. \quad \text{(condition finale)}
\end{array}
\right. \\
(iii)\ \underline{\textbf{Condition d'optimalité}} & \int_0^T \int_{\Gamma^c} (p(u)+\nu u)(v-u)d\sigma \geq 0,\ \forall v \in \mathcal{U}_{ad}, \\
& \text{ou} \quad p + \nu u = 0 \ \text{ cas sans contraintes.}
\end{array}
\right. \tag{7.3}
$$

■

Pour continuer, nous avons besoin de certains notations et préliminaires.

7.3.1 Notations

Premièrement, le domaine Ω est décomposé en m sous-domaines $\Omega_i,\ i = 1, ..., m$ et nous posons :

$$Q = \Omega\times]0,T[= \cup_{i=1}^{m} Q_i \cup \Sigma^T,$$

où $\quad Q_i = \Omega_i\times]0,T[$

et $\quad \Sigma^T = \cup_{i=1}^{m} \cup_{j\neq i} \sigma_{ij}\times]0,T[= \cup_{i=1}^{m} \cup_{j\neq i} \Sigma_{ij}^T,$ avec $\sigma_{ij} = \partial\Omega_i \cap \partial\Omega_j,$

$,^{d}\times]0,T[= \cup_{i=1}^{m}, {}_i^{d}\times]0,T[= \cup_{i=1}^{m}, {}_i^{d,T},$ avec $, {}_i^{d} = ,^{d} \cap \partial\Omega_i,$

et $\quad ,^{c}\times]0,T[= \cup_{i=1}^{m}, {}_i^{c}\times]0,T[= \cup_{i=1}^{m}, {}_i^{c,T},$ avec $, {}_i^{c} = ,^{c} \cap \partial\Omega_i.$

avec :

- $, {}_i^{d,T}$ est le bord de Ω_i commun avec le bord du domaine global au cours de l'intervalle de temps $]0,T[$ et qui correspond à la condition limite Dirichlet homogène.

- $,_i^{c,T}$ est le bord de Ω_i commun avec le bord du domaine global au cours de l'intervalle de temps $]0,T[$ et qui correspond à la condition limite Neumann : le contrôle "frontière".

- Σ_{ij}^T est l'interface au cours du temps entre deux sous-domaines voisins en espace.

D'autre part, relativement à chaque sous-domaine Ω_i, nous définissons les espaces fonctionnels suivants :

$$\begin{aligned}
V_i &= \{\varphi \in H^1(\Omega_i) \text{ tel que } \varphi = 0 \text{ sur } ,_i^d\},\\
H_i &= L^2(\Omega_i), \quad \mathcal{U}_{ad}^i = \mathcal{U}_{ad}|_{\Omega_i},\\
W_i(0,T) &= \{y \in L^2(0,T;V_i) \text{ tel que } \frac{\partial y}{\partial t} \in L^2(0,T;V_i)\},\\
W_{ij} &= TrV_i|_{\sigma_{ij}}, \quad W_{ij}^T = W_{ij}\times]0,T[, \quad W^T = \Pi_{i\in I}\,\Pi_{j\in J_i}\, W_{ij}^T.
\end{aligned}$$

7.4 Décomposition du problème de contrôle optimal évolutif

Une première conséquence de cette décomposition est la réécriture de la fonctionnelle coût en fonction des contrôles et des états directs locaux. C'est à dire, nous avons

$$J_{dec,t}(v) = \sum_{i=1}^{m} J_{dec,t}^i(v_i, y_i), \tag{7.4}$$

où

$$J_{dec,t}^i(v_i, y_i) = \frac{1}{2}\left[\int_0^T \int_{\Omega_i} |y_i - y_{d,i}|^2 dxdt + \nu \int_0^T \int_{\Gamma_i^c} |v_i|^2 d\sigma_i dt\right] \tag{7.5}$$

A ce stade, il faut ajouter des conditions supplémentaires pour que la contrainte donnée par l'edp parabolique soit la même après la décomposition du domaine Ω. Ce qui revient à introduire les contraintes de continuité des y_i et de leur flux sur les frontières entre les sous-domaines à chaque pas de temps.
Ainsi, nous montrons facilement la relation suivante entre le problème global et le problème décomposé :

Proposition 7.4.1 *La résolution du problème* $(\mathcal{P})_t$ *est équivalente à*

$$\left\{\begin{array}{l} \textit{Minimiser } \displaystyle\sum_{i=1}^{m} J^i_{dec,t}(v_i, y_i) \\ \\ \left\|\begin{array}{l} v_i \textit{ et } y_i \textit{ liés par la restriction de } (\mathcal{D}ir)_t \textit{ sur } Q_i, \\ (a)\ y_i(t) = y_j(t) \textit{ sur } \Sigma^T_{ij},\ j \neq i \\ (b)\ \dfrac{\partial y_i}{\partial \eta_{ij}}(t) = \dfrac{\partial y_i}{\partial \eta_{ij}}(t) \textit{ sur } \Sigma^T_{ij}\ j \neq i. \end{array}\right. \quad \underline{i = 1, ..., m} \end{array}\right. \tag{7.6}$$

Une façon de commenter cette proposition est de considérer la discrétisation du système $(\mathcal{D}ir)_t$ par un schéma explicite en temps, par exemple, par le schéma d'Euler, nous obtenons des systèmes elliptiques à chaque pas de temps :

$$(I - \Delta t A) y^{k+1} = \Delta t f^{k+1} + y^k \tag{7.7}$$

k et Δt sont respectivement l'indice de discrétisation et le pas de temps.
Nous reconnaissons alors ce qui a été développé dans la première partie pour le cas de problèmes elliptiques.

Ainsi, à chaque pas de temps, nous introduisons les contraintes de la proposition 7.4.1. Celles-ci peuvent se décomposer en deux parties :

- une partie **implicite** que nous formulons plus loin et
- une partie **explicite**: on introduit des variables d'intefaces $\omega_{ij}(t)$ de telle sorte que l'on ait les systèmes suivants sur chaque sous-domaine :

$$(\mathcal{D}ir)_{t,i} \quad \left\{\begin{array}{lcll} \dfrac{\partial y_i}{\partial t} - \Delta y_i & = & f_i & \text{dans } Q_i, \\ \\ y_i & = & 0 & \text{sur } {}^{d,T}_{i}, \\ \\ \dfrac{\partial y_i}{\partial \eta_i} & = & v_i & \text{sur } {}^{c,T}_{i}, \\ \\ \dfrac{\partial y_i}{\partial \eta}(t) & = & \omega_{ij}(t) & \text{sur } \Sigma^T_{ij},\ j \neq i, \\ \\ y_i(0) & = & y_{0,i} & \text{dans } \Omega_i. \end{array}\right. \tag{7.8}$$

Nous adoptons les mêmes notations et les conventions sur les indices que dans le chapitre 2.3.1, c'est à dire, $\eta = \eta_{min(i,j),max(i,j)}$.

Proposition 7.4.2 *Le système $(Dir)_{t,i}$ est bien posé et admet une solution unique y_i dans $W_i(0,T)$ qui dépend continûment de v_i et des ω_{ij}, $(j;\sigma_{ij}\neq\emptyset)$.*

Preuve : la démonstration de cette proposition repose sur les résultats des théorèmes du chapitre précédent.

7.4.1 Formulation faible de $(\mathcal{D}ir)_{t,i}$

Donnons la formulation faible de $(\mathcal{D}ir)_{t,i}$ que nous utilisons pour réécrire le nouveau problème. Soit alors une fonction test φ_i de V_i, nous avons (*cf.* chapitre 7. de [Raviart et Thomas, 1988])

$$\begin{aligned} &-\int_0^T < y_i, \frac{\partial \varphi_i}{\partial t} >_{V_i,V_i'} dt + (\varphi_i(T), y_i(T))_{H_i} - (\varphi_i(0), y_{0,i})_{H_i} + \int_0^T < \nabla y_i, \nabla \varphi_i >_{V_i,V_i'} dt = \\ &\int_0^T (f,\varphi_i)_{H_i} dt + \int_0^T (v_i,\varphi_i)_{H_i} dt + \sum_{j\in J_i^-} \int_0^T (\omega_{ij},\varphi_i)_{L^2(\sigma_{ij})} dt - \sum_{j\in J_i^+} \int_0^T (\omega_{ij},\varphi_i)_{L^2(\sigma_{ij})} dt, \end{aligned} \tag{7.9}$$

où V_i' est l'espace dual de V_i

7.5 Le nouveau problème

Comme pour le cas elliptique, notre idée est basée sur la formulation d'un nouveau problème sur le domaine décomposé. Ainsi, après avoir explicitement posé la contrainte de continuité de la dérivée normale aux interfaces dans les systèmes directs locaux en introduisant des variables $\omega_{ij}(t)$, donc, sur tout l'intervalle de temps, le problème $(\mathcal{P})_t$ devient :

$$(\mathcal{P})_{t,dec} \qquad \begin{cases} \text{Minimiser } \displaystyle\sum_{i=1}^{m} J^i_{dec,t}(v_i,y_i) \\ \\ \left\| \begin{array}{l} v_i \in \mathcal{U}^i_{ad}, \\ v_i \text{ et } y_i \text{ liés par (7.8)}, \\ y_i(t) \;=\; y_j(t) \text{ sur } \Sigma^T_{ij}. \end{array} \right. \end{cases} \tag{7.10}$$

7.6 Reformulation de $(\mathcal{P})_{t,dec}$

Nous avons un nouveau problème d'optimisation à plusieurs variables qui est soumis à un ensemble de contraintes. Une façon de le traiter est de définir le Lagrangien augmenté lui correspondant. Pour chaque temps de l'intervalle $]0,T[$, nous introduisons les

multiplicateurs de Lagrange relatifs aux contraintes :

$$y_i(t) = y_j(t) \text{ sur } \Sigma_{ij}^T. \tag{7.11}$$

D'autre part, le système $(\mathcal{D}ir)_{t,i}$ représente également une contrainte pour le problème de contrôle, nous définissons alors les états adjoints p_i, qui ne sont autres que les multiplicateurs de Lagrange relatifs aux contraintes données par les modèles directs locaux sur chaque sous-domaine Ω_i.

7.6.1 Définition du Lagrangien augmenté

Le Lagrangien augmenté associé à notre nouveau problème, noté $\mathcal{L}_r^T$, est défini sur $\Pi_{i=1}^m \mathcal{U}_{ad}^i \times (\Pi_{i=1}^m W_i'(0,T))^2 \times (\Pi_{i=1}^m \Pi_{j\in J_i^-} W_{ij}^T)^2 \times \Pi_{i=1}^m \Pi_{j\in J_i} W_{ij}^T$ à valeurs dans $\mathbb{R}$ tel que :

$$\begin{aligned}
\mathcal{L}_r^T(v_i, y_i, p_i, \omega_{ij}, \lambda_{ij}, q_{ij}) &= \sum_{i=1}^m J_{dec,t}^i(v_i, y_i) - \sum_{i=1}^m \int_0^T [< \frac{\partial y_i}{\partial t}, >_{V_i', V_i} + < \nabla y_i, \nabla p_i >_{V_i', V_i} \\
&- < f_i, p_i >_{V_i', V_i} - (v_i, p_i)_{L^2(\Gamma_i^c)}]dt + \sum_{i=1}^m \sum_{j\in J_i^-} \int_0^T \int_{\Omega_i} \omega_{ij}(t)(p_i(t) - p_j(t)))d\sigma_{ij}dt \\
&+ \sum_{i=1}^m \sum_{j\in J_i} \int_0^T [< y_i - q_{ij}), \lambda_{ij} >_{ij} + \frac{r}{2}|y_i - q_{ij}|_{ij}^2]dt.
\end{aligned}$$

7.6.2 Premières remarques

- Après avoir explicité la formulation variationnelle des y_i, nous remarquons que les variables ω_{ij} sont associées au saut des fonctions p_i sur les frontières entre le sous-domaine Ω_i et ses voisins Ω_j. Ces variables constituent donc des multiplicateurs "*implicites*" pour la continuité des p_i sur les interfaces σ_{ij} à chaque pas de temps.

- Les q_{ij} sont des variables supplémentaires telles que

$$y_i(t) = y_j(t) \text{ sur } \Sigma_{ij}^T \iff \left\{ \begin{array}{rcl} y_i(t) & = & q_{ij}(t) \text{ sur } \Sigma_{ij}^T, \\ y_j(t) & = & q_{ij}(t) \text{ sur } \Sigma_{ij}^T. \end{array} \right. \tag{7.12}$$

- r est un réel positif, coefficient de pénalisation de $\mathcal{L}_r^T$.

- Les deux Lagrangiens $\mathcal{L}_r^T$ et $\mathcal{L}_0^T$ ont les mêmes point-selles [Fortin et Glowinski, 1982].

- $\mathcal{L}_r^T$ est convexe en v_i et y_i, $i = 1, ..., m$.

- $< ., . >_{ij}$ sera pris celui de $L^2(\sigma_{ij})$.

7.6.3 Caractérisation d'un point-selle du Lagrangien $\mathcal{L}_r^T$

Soit $(u_I^\star, y_I^\star, p_I^\star, \omega_{J^-}^\star, q_{J^-}^\star, \lambda_J^\star)$ un point-selle de $\mathcal{L}_r^T$, quand il existe et que nous notons : $(p.s)_T$. Comme $\mathcal{L}_r^T$ est différentiable en chacune de ses variables, pour tout $i \in I$ et pour tout $t \in]0,T[$, sont vérifiés les systèmes suivants :

$$
\begin{aligned}
(\frac{\partial \mathcal{L}_r^T}{\partial y_i}(p.s)_T(t), \varphi_i) &= 0, \ \forall \varphi_i \in V_i, && (7.13)\\
(\frac{\partial \mathcal{L}_r^T}{\partial p_i}(p.s)_T(t), \varphi_i) &= 0, \ \forall \varphi_i \in V_i, && (7.14)\\
(\frac{\partial \mathcal{L}_r^T}{\partial v_i}(p.s)_T(t), v_i - u_i) &\geq 0, \ \forall v_i \in \mathcal{U}_{ad}^i, && (7.15)\\
(\frac{\partial \mathcal{L}_r^T}{\partial \omega_{ij}}(p.s)_T(t), d\omega) &= 0, \ \forall d\omega \in W_{ij}, \ j \in J_i^-, && (7.16)\\
(\frac{\partial \mathcal{L}_r^T}{\partial \lambda_{ij}}(p.s)_T(t), d\lambda) &= 0, \ \forall d\lambda \in W_{ij}, \ j \in J_i, && (7.17)\\
(\frac{\partial \mathcal{L}_r^T}{\partial q_{ij}}(p.s)_T(t), dq) &= 0, \ \forall dq \in W_{ij}, \ j \in J_i^-. && (7.18)
\end{aligned}
$$

$*$ Pour le premier système en p_i, il suffit d'utiliser le théorème 6.1.1 et la formule de Green pour obtenir la formulation variationnelle vérifiée par p_i, soit,

$$
\begin{aligned}
&< (y_i^\star - y_{d,i})(t), \varphi_i >_{V_i',V_i} + < \frac{\partial p_i^\star}{\partial t}(t), \varphi_i >_{V_i',V_i} - (p_i^\star(T), \varphi_i(T))_{H_i} + (p_i^\star(0), \varphi_i(0))_{H_i}\\
&+ < \Delta p_i^\star(t), \varphi_i >_{V_i',V_i} - \sum_{j \in J_i} \int_{\sigma_{ij}} \frac{\partial p_i^\star}{\partial \eta_{ij}}(t) \varphi_i d\sigma_{ij} \qquad (7.19)\\
&+ \sum_{j \in J_i} \int_{\sigma_{ij}} (\lambda_{ij}^\star(t) + r(y_i^\star(t) - q_{ij}^\star(t)) \varphi_i d\sigma_{ij} = 0,
\end{aligned}
$$

pour tout t de l'intervalle $]0,T[$ et pour toute fonction φ_i de V_i.
Ce qui est équivalent à l'équation de l'état adjoint local :

$$
(\mathcal{A}dj)_{t,i} \left\{
\begin{array}{lcll}
-\dfrac{\partial p_i^\star}{\partial t} - \Delta p_i^\star &=& y_i^\star - y_{d,i}^\star & \text{dans} \quad Q_i,\\
p_i^\star &=& 0 & \text{sur} \quad \Gamma_i^{d,T},\\
\dfrac{\partial p_i^\star}{\partial \eta_i} &=& 0 & \text{sur} \quad \Gamma_i^{c,T}, \qquad (7.20)\\
\dfrac{\partial p_i^\star}{\partial \eta}(t) &=& \lambda_{ij}^\star(t) + r(y_i^\star - q_{ij}^\star)(t) & \text{sur} \quad \Sigma_{ij}^T, \ \ j \neq i.\\
p_i^\star(T) &=& 0 & \text{dans} \quad \Omega_i.
\end{array}
\right.
$$

* Puis, la différentiation par rapport aux états p_i donne tout simplement l'équation d'état local : $(\mathcal{D}ir)_{t,i}$.

* Les conditions d'optimalité locales sont obtenues par l'inéquation d'Euler, c'est à dire, pour $i \in I$,

$$\int_{\Gamma_i^{c,T}} (p_i^\star + \nu u_i^\star)(v_i - u_i^\star) d\sigma dt \geq 0, \quad \forall v_i \in \mathcal{U}_{ad}^i. \tag{7.21}$$

* Les équations en $\omega_{ij}^\star$ et $\lambda_{ij}^\star$ sont équivalentes à :

$$p_i^\star = p_j^\star \quad \text{sur } \Sigma_{ij}^T, \tag{7.22}$$
$$y_i^\star = q_{ij}^\star \quad \text{sur } \Sigma_{ij}^T. \tag{7.23}$$

* Enfin, l'équation en $q_{ij}^\star$ donne

$$2rq_{ij}^\star = \lambda_{ij}^\star + \lambda_{ji}^\star + r(y_i^\star + y_j^\star) \quad \text{sur } \Sigma_{ij}^T. \tag{7.24}$$

■

Cette caractérisation nous sert à démontrer la proposition suivante :

Proposition 7.6.1 *Si $(u_I^\star, y_I^\star, p_I^\star, \omega_{J^-}^\star, q_{J^-}^\star, \lambda_J^\star)$ est un point-selle de $\mathcal{L}_r^T$, quand il existe, alors, les $u_i^\star$, $y_i^\star$ et $p_i^\star$ ne sont autres que les restrictions sur Q_i, respectivement du contrôle optimal u, de l'état direct y et de l'état adjoint p, qui sont les solutions du système d'optimalité (7.3) .*

Preuve : Nous utilisons la caractérisation du point-selle ci-dessus. Comme il y a unicité de la solution du système d'optimalité global (7.3), le résultat de la proposition en découle.

Proposition 7.6.2 *Dans le cas sans contraintes, la réciproque de la proposition 7.6.1 est vraie. De plus, il y a existence d'un point-selle de $\mathcal{L}_r^T$.*

Preuve : En effet, soit (u, y, p) la solution du problème global $(\mathcal{P})_T$ et posons

$$y_i = y|_{Q_i}, \quad p_i = p|_{Q_i}, \quad u_i = u|_{\mathcal{U}_{ad}^i}. \tag{7.25}$$

Tout d'abord, pour tout $t \in]0,T[$, $y(t)$ et $p(t)$ sont dans $H^1(\Omega)$, donc grâce aux propriétés des traces dans $H^1(\Omega)$, nous avons

$$y_i(t) = y_j(t) \text{ et } p_i(t) = p_j(t) \text{ sur } \sigma_{ij} \text{ et pour tout } t \in]0,T[. \tag{7.26}$$

Par ailleurs, d'après [Lions et Magenes, 1968], pour tout $t \in]0,T[$, nous obtenons

$$\frac{\partial y_i}{\partial \eta_{ij}}(t) \in H^{-\frac{1}{2}}(\sigma_{ij}) \text{ et } \frac{\partial p_i}{\partial \eta_{ij}}(t) \in H^{-\frac{1}{2}}(\sigma_{ij}). \tag{7.27}$$

Nous posons donc,

$$\omega_{ij}(t) = \frac{\partial y_i}{\partial \eta_{ij}}(t) = -\frac{\partial y_j}{\partial \eta_{ji}}(t) = \omega_{ji}(t) \in H^{-\frac{1}{2}}(\sigma_{ij}), \tag{7.28}$$

$$\lambda_{ij}(t) = \frac{\partial p_i}{\partial \eta_{ij}}(t) = -\frac{\partial p_j}{\partial \eta_{ji}}(t) = -\lambda_{ji}(t) \in H^{-\frac{1}{2}}(\sigma_{ij}). \tag{7.29}$$

Pour la condition d'optimalité locale, si nous considérons l'application caractéristique χ_i de Ω_i, alors, il vient que

$$p_i + \nu u_i = (p + \nu u)\chi_i = 0.$$

Nous avons donc montré au passage, l'existence d'un point-selle et plus particulièrement, l'existence des multiplicateurs de $\mathcal{L}_r^T$.

7.7 Un algorithme de recherche de point-selle de $\mathcal{L}_r^T$

Nous avons ramené le problème, après une décomposition spatiale, à une recherche d'un point-selle du Lagrangien augmenté $\mathcal{L}_r^T$. L'algorithme que nous proposons est similaire à (`Lag_ddm.3`), nous le notons (`Lag_ddm_T.3`)

Algorithme: (`Lag_ddm_T.3`)

Etape 0. Initialisation :

Pour tout $i \in I$, ω_{ij}^1 $(j \in J_i^-)$, λ_{ij}^1 $(j \in J_i)$ et q_{ij}^0 $(j \in J_i^-)$ choisis arbitrairement

Etape 1. Résolution de problèmes de contrôle locaux au cours du temps:

avec ω_{ij}^n, λ_{ij}^n et q_{ij}^{n-1} connus, pour chaque $i \in I$, on calcule u_i^n, y_i^n et p_i^n par :

$$(\mathcal{D}ir)_{t,i,n} \left\{ \begin{array}{lcll} \dfrac{\partial y_i^n}{\partial t} - \Delta y_i^n &=& f_i & \text{dans } Q_i, \\ y_i^n &=& 0 & \text{sur } \Gamma_i^{d,T}, \\ \dfrac{\partial y_i^n}{\partial \eta_i} &=& u_i^n & \text{sur } \Gamma_i^{c,T}, \\ \dfrac{\partial y_i^n}{\partial \eta}(t) &=& \omega_{ij}^n(t) & \text{sur } \Sigma_{ij}^T, \quad j \in J_i, \\ y_i^n(0) &=& y_{0,i} & \text{dans } \Omega_i. \end{array} \right. \tag{7.30}$$

$$(\mathcal{A}dj)_{T,i,n} \left\{ \begin{array}{lcll} -\dfrac{\partial p_i^n}{\partial t} - \Delta p_i^n &=& y_i^n - y_{d,i}^n & \text{dans } Q_i, \\ p_i^n &=& 0 & \text{sur } \Gamma_i^{d,T}, \\ \dfrac{\partial p_i^n}{\partial \eta_i} &=& 0 & \text{sur } \Gamma_i^{c,T}, \\ \dfrac{\partial p_i^n}{\partial \eta_{ij}}(t) &=& \lambda_{ij}^n(t) + r(y_i^n - q_{ij}^{n-1}) & \text{sur } \Sigma_{ij}^T, \quad j \neq i, \\ p_i^n(T) &=& 0 & \text{dans } \Omega_i. \end{array} \right. \tag{7.31}$$

$$\int_0^T \int_{\Gamma_i^c} (p_i^n + \nu u_i^n)(v_i - u_i^n) d\sigma dt \geq 0 \qquad \forall v_i \in \mathcal{U}_{ad,T}^i. \tag{7.32}$$

Etape 2. Mise à jour des multiplicateurs :

avec ρ_ω et ρ_λ deux réels strictement positifs, $\omega_{ij}^{n+1}(t)$, $\lambda_{ij}^{n+\frac{1}{2}}(t)$, $q_{ij}^n(t)$ et $\lambda_{ij}^{n+1}(t)$ sont calculés par :

$$\left\{ \begin{array}{lcll} \omega_{ij}^{n+1}(t) &=& \omega_{ij}^n(t) + \rho_\omega(p_i^n(t) - p_j^n(t)) & \text{sur } \Sigma_{ij}^T, \\ \lambda_{ij}^{n+\frac{1}{2}}(t) &=& \lambda_{ij}^n(t) + \rho_\lambda(y_i^n(t) - q_{ij}^{n-1}(t)) & \text{sur } \Sigma_{ij}^T. \\ 2r q_{ij}^n(t) &=& (\lambda_{ij}^{n+\frac{1}{2}} + \lambda_{ji}^{n+\frac{1}{2}})(t) + r(y_i^n(t) + y_j^n(t)) & \text{sur } \Sigma_{ij}^T, \\ \lambda_{ij}^{n+1}(t) &=& \lambda_{ij}^{n+\frac{1}{2}}(t) + \rho_\lambda(y_i^n(t) - q_{ij}^n(t)) & \text{sur } \Sigma_{ij}^T. \end{array} \right. \tag{7.33}$$

Faire $n \longleftarrow n+1$ et aller à **Etape 1**.

Proposition 7.7.1 *L'algorithme (*`Lag_ddm_T.3`*) est bien défini.*

En effet, à chaque itération, l'existence et l'unicité des solutions des états direct et adjoint sur chaque sous-domaine sont assurées dès que l'on choisit les contrôles u_i^n et les multiplicateurs ω_{ij}^n, λ_{ij}^n et q_{ij}^{n-1} dans des espaces appropriés. Ce qui est déjà fait.

Remarque 7.7.1 *D'une manière analogue aux problèmes elliptiques, l'***Etape 1.** *de l'algorithme ci-dessus, représente la résolution d'un problème de contrôle optimal local suivie des mises à jour des multiplicateurs de Lagrange qui servent à maintenir les contraintes de continuité sur les frontières entre les sous-domaines.*

Concrètement, pour ω_{ij}^n, λ_{ij}^n et q_{ij}^{n-1} fixés, nous aboutissons au sous-problème de contrôle suivant défini par sa fonctionnelle coût :

$$\begin{aligned}\hat{J}_{dec,t}^i(u_i^n, y_i^n) &= \frac{1}{2}\int_0^T \left[\int_{\Omega_i} |y_i^n - y_{d,i}|^2 dx + \nu \int_{\Gamma_i^c} |u_i^n|^2 d\sigma\right] dt \\ &+ \sum_{j,\sigma_{ij}\neq\emptyset} \left[\int_{\Sigma_{ij}^T} \lambda_{ij}^n (y_i^n - q_{ij}^{n-1}) d\sigma_{ij} dt + \frac{r}{2}\int_{\Sigma_{ij}^T} |y_i^n - q_{ij}^{n-1}|^2 d\sigma_{ij} dt\right]\end{aligned}$$

et son équation d'état : $(\mathcal{D}ir)_{t,i,n}$.
Ce qui revient à

$$(\mathcal{P})_{t,i,n} \quad \begin{cases} \textit{Minimiser} \quad \hat{J}_{dec,t}^i(u_i^n, y_i^n) \\ \\ \left\| \begin{array}{l} u_i^n \in \mathcal{U}_{ad,i} \; y_i^n \in V_i \\ u_i^n \textit{ et } y_i^n \textit{ liés par } (\mathcal{D}ir)_{t,i,n}. \end{array} \right. \end{cases}$$

Il suffit de remarquer que la fonctionnelle $\hat{J}_{dec,t}^i$ conserve bien la convexité en u_i^n et y_i^n.

7.8 Réduction des conditions de transmission aux frontières entre les sous-domaines

Si, dans les conditions limites aux interfaces des systèmes $(\mathcal{D}ir)_{t,i,n}$ et $(\mathcal{A}dj)_{t,i,n}$, nous supprimons les multiplicateurs de Lagrange ω_{ij}^n, λ_{ij}^n et les variables supplémentaires q_{ij}^n et avec le choix $\rho_\omega = \rho_\lambda = r$, nous retrouvons les mêmes conditions de transmission du chapitre 3. Autrement dit, à la $(n+1)$ème itération, nous écrivons la condition limite Neumann vérifiée par y_i^{n+1} sur Σ_{ij}^T :

$$-\frac{\partial y_i^{n+1}}{\partial \eta_{ij}} + r p_i^n = -\frac{\partial y_j^n}{\partial \eta_{ij}} + r p_j^n \text{ sur } \Sigma_{ij}^T \tag{7.34}$$

Et de la même manière que dans 3.7.1, nous avons aux nouvelles conditions sur les frontières entre les sous-domaines, faisant intervenir une combinaison des états directs et adjoints :

$$-\frac{\partial p_i^{n+1}}{\partial \eta_{ij}} + r y_i^{n+1} = -\frac{\partial p_j^n}{\partial \eta_{ij}} + r y_j^n \text{ sur } \Sigma_{ij}^T. \tag{7.35}$$

Nous remarquons également ici, que les systèmes des états direct et adjoint sur chaque sous-domaine (en espace) sont découplés.

La seule référence à notre connaissance, qui ait traité de ces conditions est [Benamou, 1997] et nous retrouvons, par notre approche via le Lagrangien augmenté, des conditions très similaires à celles introduites par l'intermédiaire des conditions de transmission de Lions.

7.9 Application à un problème-test

7.9.1 Le problème global

Considérons un problème-test similaire à ce que nous avons présenté dans les paragraphes précédents, c'est à dire, un problème de contrôle optimal "frontière" du flux de la température à travers un bord d'une barrière carrée $\bar{\Omega} = [0,1]^2$

L'équation d'état du problème est donnée par

$$\left\{ \begin{array}{lcll} \dfrac{\partial y}{\partial t} - \Delta y & = & f & \text{dans } \Omega \times]0,T[, \\ y & = & 0 & \text{sur } (\,,_N \cup \,,_S \cup \,,_W) \times]0,T[, \\ \dfrac{\partial y}{\partial \eta} & = & v & \text{sur } \,,^E \times]0,T[, \\ y(0) & = & y_0 & \text{dans } \Omega. \quad \text{(condition initiale)} \end{array} \right. \tag{7.36}$$

et on veut minimiser sur $\mathcal{U}_{ad}$, un convexe fermé de $L^2(\,,_E) \times]0,T[$, la fonctionnelle :

$$J(v) = \frac{1}{2}\left[\int_\Omega |y(v) - y_d|^2 dx + \nu \int_{\Gamma_E} |v|^2 d\sigma \right] dt$$

avec les données :

- Source F : $F(x,y,t) = -x(y^2 - y + 2(2T - t))$,
- Observations y_d : $y_d(x,y,t) = (y^2 - y)(\nu + (2T - t)x) - 2\nu(2T - t)$,
- Condition initiale y_0 : $y_0(x,y) = 2Tx(y^2 - y)$.

7.9.2 Discrétisation

Afin de résoudre numériquement notre problème de contrôle d'évolution, nous utilisons une méthode de descente. Pour cela, nous avons besoin de connaître en un point donné, la valeur de la fonction J ainsi que son gradient. Par suite, nous avons à calculer les états direct et adjoint discrets associés au point en question. Nous nous intéressons donc à une discrétisation de (7.36) :

Pour la discrétisation temporelle, nous utilisons un schéma d'Euler explicite, c'est à dire,

$$\frac{\partial y}{\partial t} = \frac{y^k - y^{k-1}}{\delta t}, \tag{7.37}$$

où δt est le pas de discrétisation en temps, l'indice k étant le numéro du pas de temps.

En espace, nous discrétisons par un schéma de différences finies sur une grille régulière, ainsi, le Laplacien discrétisé est donné par le schéma classique à cinq points.

Donc, le système d'edp parabolique (7.36) discrétisé se résume en :

- $\underline{i = 2,\ Nx,\ j = 2,\ Ny,\ k = 1,\ NT}$: (à l'intérieur du domaine global)

$$-\frac{\delta t}{hx^2}(y^k_{i-1j} + y^k_{i+1j}) - \frac{\delta t}{hy^2}(y^k_{ij-1} + y^k_{ij+1}) + (1 + \frac{2\delta t}{hx^2} + \frac{2\delta t}{hy^2})y^k_{ij} = \delta t F^k_{ij} + y^{k-1}_{ij},$$

- $\underline{i = Nx+1,\ j = 2,\ Ny,\ k = 1,\ NT}$:

$$-\frac{\delta t}{hx^2}y^k_{Nxj} - \frac{\delta t}{hy^2}(y^k_{Nx+1j-1} + y^k_{Nx+1j+1}) + (1 + \frac{2\delta t}{hx^2} + \frac{2\delta t}{hy^2})y^k_{Nx+1j} = \delta t F^k_{Nx+1j} + y^{k-1}_{Nx+1j},$$

Et il faut ajouter à ces équations, celles découlant des conditions aux limites ainsi que la condition initiale.

Nous avons donc, à chaque pas de temps, la résolution d'un système linéaire dont la matrice est creuse et non symétrique.

Quant à l'état adjoint discret, il se déduit de la transposition du système direct discret ci-dessus. Quelques détails sur le calcul et l'implantation de l'état adjoint discret sont présentés en annexe.

7.10 Conclusions

Nous rencontrons les mêmes difficultés que pour le problème stationnaire. En effet, dans le cas du problème dépendant du temps, nous assurons à chaque pas de temps les contraintes de continuité sur les interfaces entre les sous-domaines. Par conséquent un

nombre plus grand de variables supplémentaires ($\omega_{ij}(t)$ et $\lambda_{ij}(t)$) qu'il n'est pas facile à contrôler. Par suite, comme les raccordements ne sont pas bien assurés au premier pas de temps, ces "erreurs" sont propagées aux autres pas de temps. le choix de r est également un problème à résoudre.
Nous avons essayé de remédier à ce problème en ajoutant un recouvrement entre les sous-domaines tout en gardant les mêmes conditions de raccord. Ensuite, nous avons reformulé la nouvelle version du problème par un Lagrangien augmenté. Cependant, les résultats ne se sont pas très bien améliorés.

Nous avons vu qu'avec une décomposition du domaine sans recouvrement, l'algorithme de "type Uzawa" et plus spécialement, la mise à jour des multiplicateurs qui n'est pas bien contrôlée, n'étaient pas suffisants pour avoir une solution globale à partir des solutions locales, qui soit continue sur les interfaces.
C'est pourquoi, nous allons introduire dans la partie suivante de nouveaux algorithmes basés cette fois sur une décomposition du domaine avec recouvrement.

Troisième partie

Sur des méthodes de décomposition de domaine avec recouvrement pour la résolution de problèmes de contrôle optimal.

Chapitre **8 :**

Algorithmes de Schwarz et problèmes de contrôle optimal.

Dans ce chapitre, essentiellement numérique, nous nous intéressons à l'implantation de divers algorithmes parallèles pour la résolution de systèmes direct et adjoint dans les deux cas de problèmes de contrôle optimal : elliptique et parabolique. Nous étudions ainsi les conséquences induites par la combinaison de ces algorithmes sur le comportement des algorithmes d'optimisation pour résoudre le problème de contrôle optimal.

Introduction

Jusqu'à présent, nous nous sommes intéressés à une décomposition du domaine sans recouvrement pour résoudre un problème de contrôle optimal. La prise en compte des raccordements entre les sous-domaines par des techniques de Lagrangien augmenté nous a amené à considérer des algorithmes qui intégrent les contraintes du problème de contrôle par l'intermédiaire de multiplicateurs.
Dans ce chapitre, nous considérons le problème de contrôle optimal pris sur le domaine global en entier. L'utilisation des méthodes de décomposition du domaine intervient uniquement au niveau de la résolution des systèmes linéaires issus des états direct et adjoint discrets.
Nous utilisons des méthodes de décomposition de domaine de type Schwarz additif ou multiplicatif avec recouvrement, soit en tant que solveur ou préconditionneur des systèmes linéaires des états direct et adjoint discrets.

Avant de présenter les résultats des différents algorithmes de ce chapitre, nous donnons dans le paragraphe qui suit des rappels sur les deux méthodes essentielles que nous employons, notamment, la méthode de minimisation et celle de résolution des systèmes linéaires.

8.1 Rappels

La plupart des résultats de ce chapitre sont obtenus en utilisant un algorithme de type quasi-Newton avec la formule de mise à jour B.F.G.S. à mémoire limitée. Mais nous en présentons également une comparaison dans certains cas avec l'algorithme de gradient conjugué (ce dernier est détaillé au chapitre 2. paragraphe 2.3.1). Nous avons plus précisément utilisé l'algorithme "`M1QN3`" (ou sa version double précision `N1QN3`) programmé par J.-C. Gilbert et C. Lemaréchal de l'INRIA [Gilbert et Lemaréchal, 1989].

Rappelons que l'algorithme de Newton pour la recherche d'un zéro d'une fonction à plusieurs variables consiste à linéariser la fonction au voisinage du point courant et à résoudre successivement ces problèmes linéarisés. Cet algorithme est donné par :

Soit $G : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ et soit à résoudre $G(X) = 0$. Pour X_0 donné dans $\mathbb{R}^n$. On calcule la suite X_n qui, sous certaines conditions, converge vers un zéro de G, par:

$$X_{k+1} = X_k - (JacG(X_k))^{-1}.G(X_k).$$

Dans un problème de contrôle optimal, on cherche à minimiser le critère J. Donc, la fonction G dont on cherche le zéro est la suivante :

$$\begin{aligned} G : \mathbb{R}^n &\longrightarrow \mathbb{R}^n && (8.1)\\ & && (8.2)\\ X &\mapsto \overrightarrow{\nabla} J(X). && (8.3) \end{aligned}$$

Le Jacobien de G au point X_k est, dans ce cas, le Hessien de la fonction coût J calculé en ce même point. L'algorithme de Newton pour un problème de minimisation s'écrit donc :

$$X_{k+1} = X_k - (HessJ(X_k))^{-1}.\overrightarrow{\nabla} J(X_k).$$

La méthode de Newton utilise la formule dite de B.F.G.S. (resp. B.F.G.S. inverse), dûe à Broydon, Fletcher, Goldfarb et Shanno, qui fournit une façon d'évaluer récursivement une approximation du Hessien (resp. de l'inverse du Hessien) au point X_k. Cette formule utilise uniquement les vecteurs gradients G_k et les points X_k auxquels les G_k ont été évalués. La méthode de quasi-Newton est en réalité une méthode de type gradient car elle n'utilise que les différentielles jusqu'à l'ordre un de la fonctionnelle à minimiser J. Elle s'écrit donc :

Méthode de quasi-Newton de type B.F.G.S :"M1QN3"

$$
\begin{array}{lcll}
X_0 & \in & \mathbb{R}^n; & \\
G_0 & \in & \mathbb{R}^n; G_0 = \overrightarrow{\nabla} J(X_0); & \\
B_0 & \in & \mathcal{M}(\mathbb{R}^n); B_0 = HessJ(X_0); & \\
H_0 & \in & \mathcal{M}(\mathbb{R}^n); H_0 \approx B_0^{-1}; &
\end{array}
$$

$$
\left\|
\begin{array}{lcll}
d_k & = & -H_k.G_k; & \text{(direction de descente)} \\
X_{k+1} & = & X_k + \rho_k d_k; & \text{(recherche linéaire)} \\
G_{k+1} & = & \overrightarrow{\nabla} J(X_{k+1}); & \text{(calcul de gradient)} \\
s_k & = & X_{k+1} - X_k; & \text{(différence entre 2 points)} \\
y_k & = & G_{k+1} - G_k; & \text{(différence entre 2 gradients)} \\
B_{k+1} & = & B_k + \dfrac{y_k \otimes y_k}{\langle y_k, s_k \rangle} - \dfrac{[(B_k.s_k) \otimes (B_k.s_k)]}{\langle s_k, B_k.s_k \rangle}; & \text{(B.F.G.S.)} \\
H_{k+1} & = & H_k + \dfrac{[(s_k - H_k.y_k) \otimes s_k] + [s_k \otimes (s_k - H_k.y_k)]}{\langle y_k, s_k \rangle^2} - \dfrac{\langle s_k - H_k.y_k, y_k \rangle}{\langle y_k, y_k \rangle^2}[s_k \otimes s_k]; & \\
 & & & \text{(B.F.G.S inverse)}
\end{array}
\right.
$$

où $\langle ., . \rangle$ est un produit scalaire sur $\mathbb{R}^n$; $[. \otimes .]$ est le produit tensoriel associé à ce produit scalaire, i.e.

$$
\begin{array}{rcl}
\otimes : \mathbb{R}^n \times \mathbb{R}^n & \longrightarrow & \mathcal{L}(\mathbb{R}^n) \\
(U, V) & \mapsto & U \otimes V
\end{array}
$$

avec $(U \otimes V)X = \langle V, X \rangle U \quad \forall X \in \mathbb{R}^n$.

Nous avons quelques remarques sur cette méthode :

- Quand le produit scalaire utilisé est le produit scalaire euclidien, alors, la matrice dans la base canonique $(e_i)_{1 \leq i \leq n}$ du produit tensoriel est donnée par : $U.V^t$.
- Les formules de BFGS et BFGS inverse conservent la positivité des matrices B_{k+1} et H_{k+1} calculées si et seulement si $\langle y_k, s_k \rangle$ l'est. La recherche du pas de descente ρ_k

se fait par une interpolation cubique de la fonctionnelle J dans la direction de descente d_k. Pour préserver la condition de positivité pendant cette étape de recherche linéaire, nous exigeons que le pas ρ_k satisfait deux conditions supplémentaires:

$$\begin{cases} J(X_k + \rho_k d_k) \leq J(X_k) + \alpha_1 \rho_k \langle G_k, d_k \rangle \\ \langle \overrightarrow{\nabla} J(X_k + \rho_k d_k), d_k \rangle \leq \alpha_2 \langle G_k, d_k \rangle \end{cases} \tag{8.4}$$

avec $0 < \alpha_1 < \frac{1}{2}$ et $\alpha_1 < \alpha_2 < 1$. Cette étape de recherche linéaire représente une itération interne dans la méthode de quasi-Newton jusqu'à obtenir le "bon" pas.

- Pour un problème de grande taille, nous n'avons pas besoin de calculer les coefficients de H_k et de les stocker en mémoire. En effet, la matrice H_k ne dépend que de k couples de vecteurs $(y_i, s_i)_{1 \leq i \leq k}$. Ainsi, en stockant les $2k$ vecteurs $\{(y_i, s_i, 1 \leq i \leq k\}$, nous pouvons calculer la direction de descente $d_k = -H_k.G_k$ à la $k$ème étape de la minimisation. Mais ce stockage représente toujours un problème et plus spécialement quand le nombre d'itérations est de l'ordre du nombre de paramètres. Ce problème peut être résolu en introduisant la méthode quasi-Newton à mémoire limitée: elle consiste à calculer H_k en ne faisant intervenir qu'un nombre limité de couples de vecteurs (y_i, s_i), à savoir les plus récents.

Les mesures de temps CPU faites dans le cas de la résolution séquentielle (ou sur un seul processeur) nous ont montré que le code d'optimisation, proprement dit, consomme un temps négligeable devant le temps consacré au calcul de la fonction coût et du gradient, ce qui revient à la résolution des systèmes direct et adjoint discrets. Nous le montrons sur notre problème test (défini au chapitre 5.). Pour $N = 64$, les résultats de ce test sont effectués sur une station SUN et les pourcentages des différentes parties du code sur le domaine global sont représentés sur la figure 8.1. La méthode numérique de minimisation employée est la méthode que nous avons décrite ci-dessus: `N1QN3` . Les systèmes direct et adjoint discrets sont calculés par une méthode itérative de Krylov: `BiCGSTAB` . Nous remarquons alors que 99% du code est consacré au solveur des systèmes direct et adjoint discrets et ses sous-programmes.
Ces constatations nous ont donc permis d'opter pour une implantation efficace d'algorithmes parallèles sur des architectures distribuées en répartissant uniquement la charge des solveurs. Nous allons utiliser soit une parallélisation directe ou bien trouver des algorithmes se prêtant bien au calcul parallèle. Ce dernier cas repose sur les méthodes de Schwarz (Schwarz additive ou Schwarz multiplicative exposées au premier chapitre de ce mémoire).
Afin de résoudre les systèmes linéaires direct et adjoint discrets, nous avons utilisé une méthode itérative basée sur les sous-espaces de Krylov dont un aperçu est présenté dans l'annexe A. Et plus précisément, l'algorithme que nous avons utilisé est `BiCGSTAB` (Bi-Conjugate Gradient STABilized) basé sur des techniques de gradient bi-conjugué qui sont encore valables pour des systèmes non symétriques (en effet, les systèmes auxquels nous aboutissons à la suite de la discrétisation des états direct et adjoint de notre problème-test ne sont pas symétriques).
Cet algorithme `BiCGSTAB` est décrit explicitement ci-après :

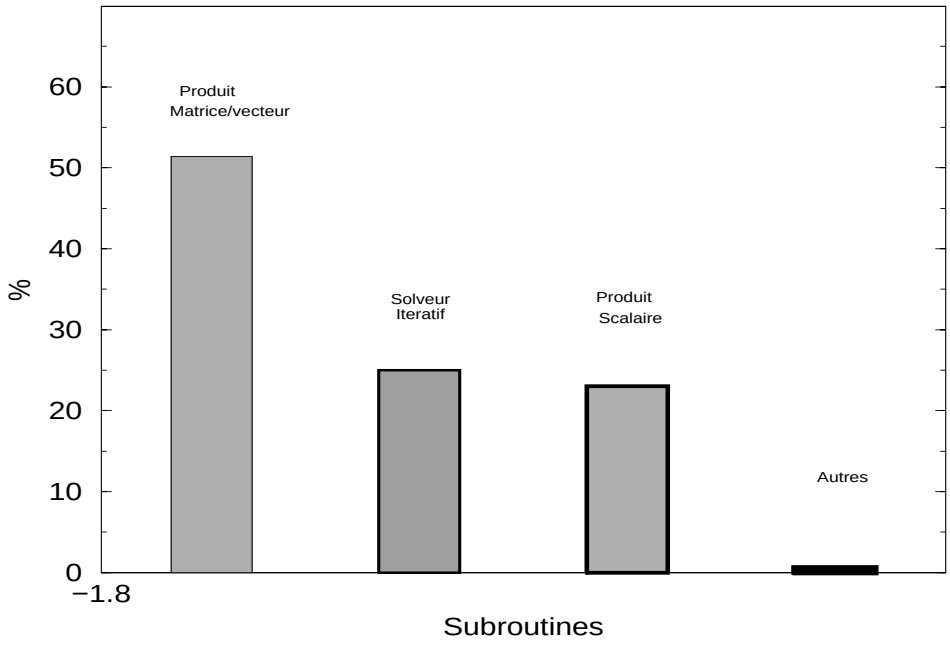

FIG. 8.1 – *Parts des sous-programmes du code séquentiel, $N = 64$. Poids relatif des subroutines. (profiling d'un calcul complet)*

Algorithme BiCGSTAB

Soient $\mathtt{x}_0$, ϵ, ϵ_{stop} donnés
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v} = \mathbf{p} = 0$
$\rho_0 = \alpha = \omega = 1$
Tant que $\frac{\|\mathtt{r}\|_2}{\|\mathtt{r}_0\|_2} > \epsilon$ et $\|\mathtt{r}\|_2 > \epsilon_{stop}$ faire
$\rho = (\mathtt{r}_0{}^T \mathtt{r})$
$\beta = \alpha\rho/\rho_0\omega$, $\rho_0 = \rho$
$\mathtt{p} = \mathtt{r} + \beta(\mathtt{p} - \omega\mathtt{v})$
Résoudre $\mathbf{C}\widehat{\mathtt{p}} = \mathtt{p}$
$\mathtt{v} = \mathbf{A}\widehat{\mathtt{p}}$
$\alpha = \rho_1/(\mathtt{r}_0{}^T \mathtt{v})$
$\mathtt{s} = \mathtt{r} - \alpha\mathtt{v}$
Résoudre $\mathbf{C}\mathtt{z} = \mathtt{s}$
ÉCHANGE_BORDS (z)
$\mathtt{t} = \mathbf{A}\mathtt{z}$
$\omega = \dfrac{(\mathtt{t}^T \mathtt{s})}{(\mathtt{t}^T \mathtt{t})}$
$\mathtt{x} = \mathtt{x} + \alpha\widehat{\mathtt{p}} + \omega\mathtt{z}$
$\mathtt{r} = \mathtt{s} - \omega\mathtt{t}$
Fin

La méthode ci-dessus permet, d'une part, de traiter des problèmes de taille plus importante (cependant, nous nous heurtons à un problème de conditionnement des gros systèmes linéaires) et d'autre part, elle permet de tirer partie des méthodes de décomposition de domaine en tant que préconditionneurs.

Ainsi, pour une simulation du problème test (nécessaire pour une itération de l'optimiseur), ces méthodes de décomposition de domaine peuvent être résumées par le schéma 8.2.

Dans ce qui suit, nous présentons donc trois algorithmes de résolution de notre problème de contrôle optimal qui diffèrent par leur méthode de résolution des systèmes direct et adjoint. Pour chacun de ces trois solveurs itératifs, nous donnons les résultats numériques associés ainsi qu'une comparaison entre les divers cas-tests effectués. Cela est développé dans trois parties :

1. BiCGSTAB réparti : c'est la parallélisation directe du solveur itératif BiCGSTAB .
2. Méthode de Schwarz : c'est un algorithme parallèle qui consiste à utiliser une méthode de Schwarz (avec recouvrement) pour résoudre le système direct et en déduire une transposition pour calculer l'état adjoint discret.
3. BiCGSTAB préconditionné : c'est BiCGSTAB préconditionné par une méthode de décomposition de domaine, à savoir, une méthode de Schwarz additive.

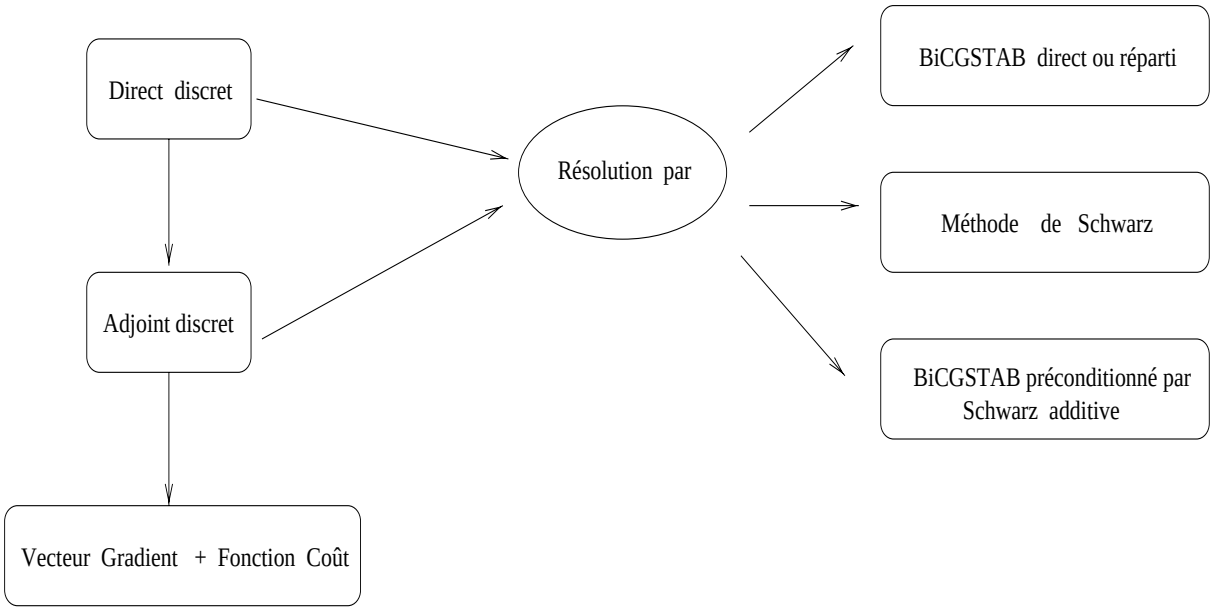

FIG. 8.2 – *Une simulation du problème test: les différentes façons de résoudre (direct, adjoint).*

8.2 Parallélisation

Dans ce paragraphe, les états direct et adjoint sont résolus par `BiCGSTAB` distribué. En effet, ce solveur itératif a la particularité de se paralléliser naturellement puisqu'il présente l'avantage de faire appel à des calculs de produits scalaires et des produits matrice-vecteur, qui se distribuent facilement sans menacer l'équilibrage de charge. Il suffit en effet, d'ajouter judicieusement à leur implantation séquentielle, des appels à des outils de communication et des opérations globales de réduction.

Comment se déroule cette parallélisation pour résoudre un problème de contrôle optimal stationnaire ?

Comme nous l'avons mentionné plus haut, l'optimisation en elle même consomme peu de temps de calcul et par conséquent, la parallélisation se fait essentiellement au niveau de la résolution des états direct et adjoint discrets.

Si nous revenons au solveur de Krylov, comment se fait sa distribution sur plusieurs processeurs ?

Premièrement, nous profitons largement des moyens de communication et des opérations de réduction fournis par la librairie de passage de message MPI. Ainsi, chaque sous-domaine local de Ω (i.e, la restriction du domaine global sur chaque processeur) est élargi par des tableaux qui contiennent des valeurs communiquées par ses voisins les plus

$\mathtt{x}_0$, ϵ, ϵ_{stop}
ÉCHANGE_BORDS($\mathtt{x}_0$)
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v} = \mathbf{p} = 0$
$\rho_0 = \alpha = \omega = 1$
TANT QUE $\dfrac{\texttt{GLOBAL\_}\|\mathtt{r}\|_2}{\texttt{GLOBAL\_}\|\mathtt{r}_0\|_2} > \epsilon$
ET GLOBAL_$\|\mathtt{r}\|_2 > \epsilon_{stop}$ FAIRE
$\rho = \texttt{GLOBAL\_}({\mathtt{r}_0}^T\mathtt{r})$
$\beta = \alpha\rho/\rho_0\omega$, $\rho_0 = \rho$
$\mathtt{p} = \mathtt{r} + \beta(\mathtt{p} - \omega\mathtt{v})$
RÉSOUDRE $\mathbf{M}\widehat{\mathtt{p}} = \mathtt{p}$
ÉCHANGE_BORDS($\widehat{\mathtt{p}}$)
$\mathtt{v} = \mathbf{A}\widehat{\mathtt{p}}$
$\alpha = \rho_1/\texttt{GLOBAL\_}({\mathtt{r}_0}^T\mathtt{v})$
$\mathtt{s} = \mathtt{r} - \alpha\mathtt{v}$
RÉSOUDRE $\mathbf{M}\mathtt{z} = \mathtt{s}$
ÉCHANGE_BORDS(z)
$\mathtt{t} = \mathbf{A}\mathtt{z}$
$\omega = \dfrac{\texttt{GLOBAL\_}(\mathtt{t}^T\mathtt{s})}{\texttt{GLOBAL\_}(\mathtt{t}^T\mathtt{t})}$
$\mathtt{x} = \mathtt{x} + \alpha\widehat{\mathtt{p}} + \omega\mathtt{z}$
$\mathtt{r} = \mathtt{s} - \omega\mathtt{t}$
FIN

FIG. 8.3 – BiCGSTAB *distribué*

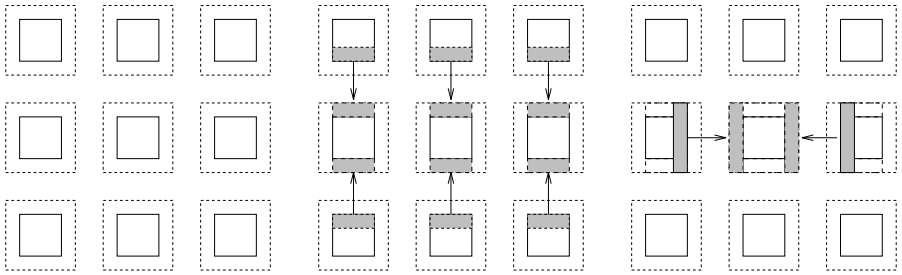

FIG. 8.4 – *Extension d'un domaine local avec des valeurs de ses voisins.*

proches ou par des valeurs provenant de la discrétisation des conditions aux limites sur le bord de Ω (commun au sous-domaine en question). En effet, dans `BiCGSTAB` distribué, l'appel à "`ÉCHANGE_BORDS`" fournit les valeurs communiquées par les voisins dans les directions (Nord, Sud, Est et Ouest), voir figure 8.4. Ensuite, toutes les opérations précédées par "`GLOBAL`"sont effectuées sur chaque nœud avant d'appeler une opération de réduction globale qui retourne à tous les autres nœuds le résultat global. Une description complète de cette partie ainsi qu'un exemple d'implantation sont présentés dans [Kortas et Angot, 1996] et [Keyes et al., 1995]. Il est à noter qu'une parallélisation directe est faite en posant " $\mathbf{M} = \mathbf{I}$" dans "$\mathbf{M}\hat{\mathbf{p}} = \mathbf{p}$" et "$\mathbf{Mz} = \mathbf{s}$" de l'algorithme `BiCGSTAB` distribué présenté sur la figure 8.3.

■

Dans cette partie, les tests numériques sont effectués sur les machines Cray-T3E et l'IBM-SP1. Nous utilisons la librairie de passage de message MPI et tous les programmes sont écrits en fortran.
Dans cette première méthode qu'est la parallélisation, nous considérons un maillage du domaine global de 512×128. Nous précisons au fur et à mesure les comparaisons entre les résultats obtenus sur les machines parallèles Cray-T3E et l'IBM-SP1.

8.2.1 Parallélisation et algorithmes d'optimisation

Pour résoudre le problème de contrôle optimal, deux méthodes d'optimisation (déjà décrites dans le chapitre 2.) sont testées :

1. **Méthode de Gradient conjugué "`GradCjg `":** Comme la fonctionnelle coût est quadratique, cette méthode s'avère efficace. Cependant, il faut imposer une précision très grande, c'est à dire, ϵ et ϵ_{stop} de `BiCGSTAB` (voir figure 8.3) doivent être très petits. (une description de `GradCjg` est présentée au chapitre 2. paragraphe 2.3.1.)

2. **Méthode de Quasi-Newton :** En utilisant `N1QN3` (une version en double précision de `M1QN3` [Gilbert et Lemaréchal, 1989]), nous remarquons que l'on est pas contraint de prendre des tolérances très petites dans `BiCGSTAB` : Pour $\epsilon = \epsilon_{stop} = 1.e-10$ ou $\epsilon = \epsilon_{stop} = 1.e-13$, nous obtenons le même nombre d'itérations de `N1QN3` , et plus particulièrement, le déroulement des appels à chaque itération se fait de la même façon pour les deux tests d'arrêt.

Les tableaux 8.2 et 8.3 fournissent respectivement le nombre d'itérations ainsi que les temps CPU obtenus avec les deux optimiseurs `GradCjg` et `N1QN3` lorsque nous varions le nombre de processeurs et lorsque nous changeons la manière de décomposer le domaine:

En premier lieu, pour un test d'arrêt dans `N1QN3` : $\epsilon_{grad} = \dfrac{\|g_k\|}{\|g_0\|} = 1.e-6$, nous avons obtenu le même nombre d'itérations dans chacun des deux optimiseurs quelque soit le

nb_procs	grille locale	temps CPU (s)	nb_iter_GradCjg
$1 = 1 \times 1$	512×128	338.30	6
$2 = 2 \times 1$	256×128	155.42	6
$4 = 4 \times 1$	128×128	84.42	6
$4 = 2 \times 2$	256×64	114.46	6
$8 = 8 \times 1$	64×128	53.85	6
$8 = 4 \times 2$	128×64	62.53	6
$16 = 16 \times 1$	32×128	48.25	6
$16 = 8 \times 2$	64×64	40.46	6

TAB. 8.2 – $N = 128$, *optimiseur* = `GradCjg` , $\epsilon = \epsilon_{stop} = 1.e-13$. *Résultats obtenus pour différents nombres de processeurs et pour les deux manières de la décomposition: bande ou grille. Calcul effectué sur Cray-T3E.*

nb_procs	grille locale	temps CPU (s)	nb_iter_N1QN3
$1 = 1 \times 1$	512×128	281.84	9
$2 = 2 \times 1$	256×128	147.15	9
$4 = 4 \times 1$	128×128	80.12	9
$4 = 2 \times 2$	256×64	102.51	9
$8 = 8 \times 1$	64×128	51.15	9
$8 = 4 \times 2$	128×64	56.22	9
$16 = 16 \times 1$	32×128	45.71	9
$16 = 8 \times 2$	64×64	37.09	9

TAB. 8.3 – $N = 128$, *optimiseur* = `N1QN3` , $\epsilon = \epsilon_{stop} = 1.e-10$. *Résultats obtenus pour différents nombre de processeurs et deux manières de la décomposition.Calcul effectué sur Cray-T3E.*

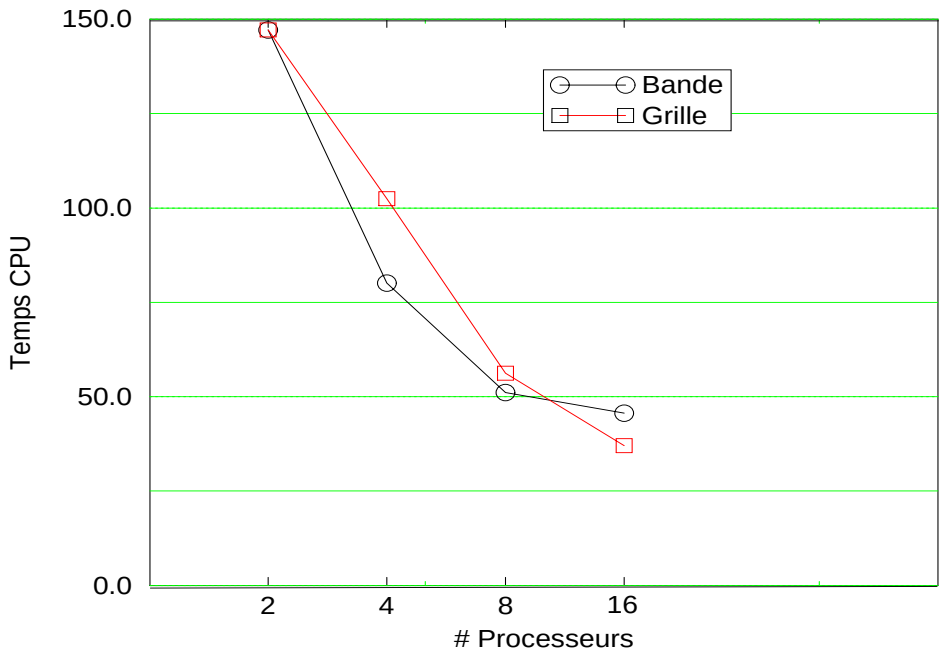

FIG. 8.5 – *Temps CPU (s) en fonction du nombre de processeurs.* $N = 128$, *optimiseur* = `N1QN3` . *Influence de la nature de la décomposition sur le temps CPU. (Calcul complet sur Cray T3E)*

nombre de processeurs. Donc, nous gardons la même précision de résolution des états direct et adjoint au cours de chaque simulation de notre problème test. Ces résultats sont exactement ceux obtenus pour le calcul sur le domaine global (ou sur un seul processeur).

D'autre part, nous avons procédé à deux manières de décomposer le domaine de calcul, le nombre de processeurs étant égal au nombre de sous-domaines de telle sorte que chaque sous-domaine soit distribué sur un processeur:

1. Décomposition en **bande**: Dans ce cas, les communications se font seulement au niveau des bords Est et Ouest de chaque sous-domaine et ses voisins directs.

2. Décomposition en **grille**: Les communications entre les sous-domaines voisins se font dans les deux sens Est-Ouest et Nord-Sud. Nous réduisons aussi la dimension des données transférées, néanmoins, nous avons plus de communications et cela coûte cher en temps calcul.

En nous référant aux deux tableaux 8.2 et 8.3, les temps CPU sont bien influencés par la nature de la décomposition pour les deux algorithmes.

Nous avons représenté cette influence sur la figure 8.5 dans le cas de l'utilisation de `N1QN3` . Les temps CPU avec la décomposition en bande sont nettement meilleurs jusqu'à 8 processeurs et ensuite, les deux courbes se croisent et avec 16 processeurs, le temps CPU de la décomposition en grille est inférieur à celui en bande. Nous pouvons conclure qu'avec un maillage plus grand et sur plusieurs processeurs disposés en grille, les calculs sont moins coûteux.
Nous remarquons également que le nombre d'itérations de `GradCjg` est inférieur à celui de `N1QN3` . Cependant, nous consommons un temps calcul plus important avec `GradCjg` qu'avec `N1QN3` : cela est dû en effet, au test d'arrêt sévère exigé pour `BiCGSTAB` lorsque nous utilisons l'optimiseur `GradCjg` . Ce qui induit un nombre d'itérations dans `BiCGSTAB` plus élevé. Les tests du tableau 8.2 sont obtenus avec $\epsilon = \epsilon_{stop} = 1.e-13$. Mais, quand ces deux valeurs sont égales à $1.e-10$, l'optimisation est bloquée sur la 5ème itération sauf si l'on diminue la précision sur le test du gradient dans l'optimiseur `GradCjg` .

Conclusion 1.

On peut donc conclure que `N1QN3` **est plus robuste que** `GradCjg` **et qu'il est peu sensible aux erreurs d'arrondis.**

Remarque 8.2.1 *Le nombre d'itérations moyen de* `BiCGSTAB` *pour calculer les états direct et adjoint sont respectivement de* 520 *et* 600 *pour* `GradCjg` *et de* 401 *et* 398 *pour* `N1QN3` .

Conclusion 2.

Le nombre d'itérations dans les deux optimiseurs ne dépend pas de la nature de la décomposition. Par contre, le coût CPU est plus élevé avec une décomposition en grille mais, il s'améliore et devient plus compétitif dès que le nombre de processeurs devient grand.

8.2.2 Performances parallèles

Nous rappelons que le code de résolution de notre problème de contrôle optimal n'est pas totalement parallélisé mais que la part qui l'est, en représente un grand pourcentage. Plus précisément, les presque 2% seulement du programme ne sont pas parallélisables. Par suite, d'après la loi d'Amdhal (*cf.* Chapitre 4. paragraphe 4.4), nous espérons avoir une efficacité parallèle satisfaisante. Nous définissons cette efficacité par :

$$E_n = \frac{T_1}{nT_n} \qquad \text{en } \% \tag{8.5}$$

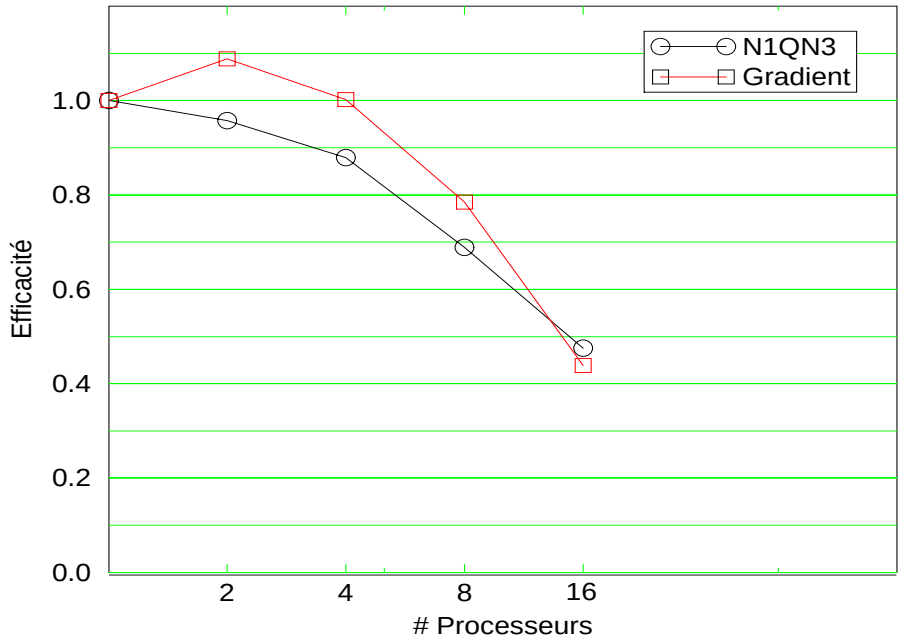

FIG. 8.6 – *Efficacités parallèles des deux optimiseurs quasi-Newton et gradient conjugué avec* `BiCGSTAB` *distribué, en fonction du nombre de processeurs.* $N = 128$. *(Calcul complet sur Cray-T3E).*

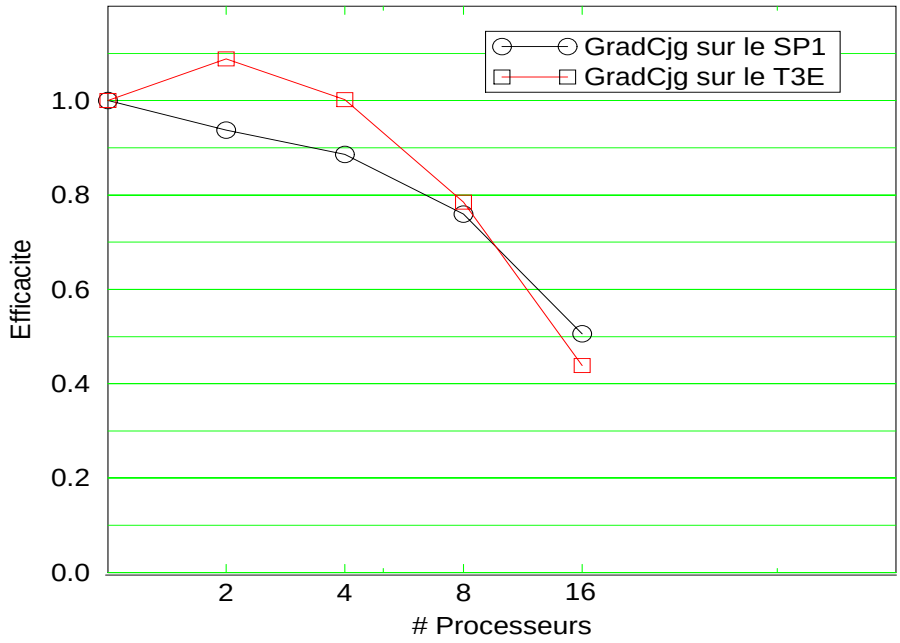

FIG. 8.7 – *Comparaison des efficacités de* `GradCjg` *avec* `BiCGSTAB` *distribué, obtenues sur l'IBM-SP1 et sur le Cray-T3E.* $N = 128$.

où T_1 et T_n sont les temps CPU effectués respectivement sur un seul processeur et n processeurs de la machine parallèle utilisée.

D'après la figure 8.6, les efficacités des deux optimiseurs se dégradent dès que le nombre de processeurs est élevé (qui est classique). D'autre part, nous observons que l'efficacité obtenue avec l'optimiseur `GradCjg` est supérieure à 100% avec 2 et 4 processeurs, ou superlinéaire. Ces résultats reflétent en fait, une meilleure gestion de la mémoire cache lorsque le domaine est distribué sur 2 ou 4 processeurs sur le Cray T3E. Et pour montrer que la nature de la machine parallèle utilisée joue un rôle dans le calcul de l'efficacité parallèle d'une application, nous donnons une comparaison des efficacités obtenues sur le SP1 et sur le T3E pour `GradCjg` sur la figure 8.7. Nous relevons que l'efficacité de `GradCjg` avec 16 processeurs sur le SP1 est légèrement supérieure à celle sur le T3E.
Néanmoins, avec `N1QN3` , l'efficacité s'améliore avec un nombre de processeurs élevé (ici ≥ 16 processeurs).

D'autres tests d'efficacité pour `N1QN3` montrent que l'on gagne plus en efficacité parallèle quand la taille du problème s'accroît. Sur la figure 8.8, nous observons que lorsque N est multiplié par 4, l'efficacité avec 16 processeurs est multipliée par presque 8.
Ces résultats confirment que ces algorithmes distribués sont bien adaptés aux architectures

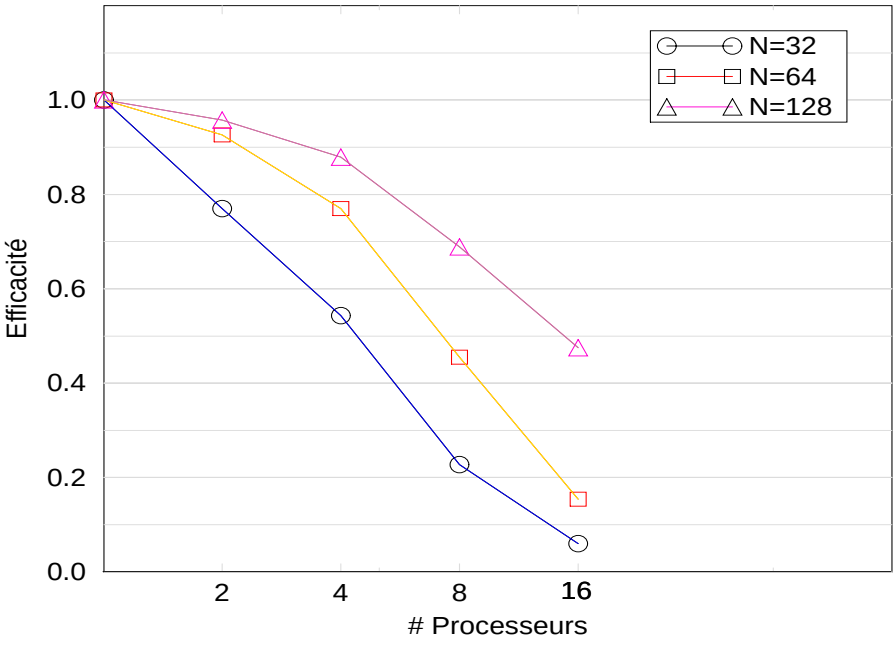

FIG. 8.8 – *Efficacité parallèle de* `N1QN3` *avec* `BiCGSTAB` *parallélisé en fonction de* N. *Calcul complet sur Cray-T3E.*

parallèles et leur efficacité est meilleure quand le problème est de grande taille.

8.3 Schwarz multiplicative pour le problème de contrôle optimal

Dans ce paragraphe, il ne s'agit plus d'une parallélisation directe des codes direct et adjoint. Mais nous considérons plutôt une méthode de décomposition de domaine avec recouvrement et l'optimisation reste toujours globale dans le cas de ce problème test. Nous résolvons les états directs locaux par la méthode de Schwarz multiplicative. Or, pour résoudre le problème de contrôle optimal, nous rappelons que nous avons besoin de connaître le gradient discret et la valeur de la fonction coût en un point donné. Pour cela, nous avons parlé précédemment de la transposition du système direct discret afin de calculer avec une bonne précision l'adjoint discret. Dans notre cas de décomposition de domaine, il faut transposer les systèmes discrets locaux. En pratique, ceci ne pose pas de problème puisque, de la méthode de Schwarz multiplicative avec des conditions aux limites Dirichlet, résultent des systèmes linéaires que nous écrivons par exemple sous la forme simplifiée :

$$A_i Y_i^{n+1} = F_i(f_i, u_i^{n+1}, y_j^n|_{\partial\Omega_i \cup \Omega_j}) \tag{8.6}$$

où sur chaque sous-domaine Ω_i, les u_i^{n+1} sont les vecteurs de contrôle à la $(n+1)$ème itération, les f_i sont les sources ou les seconds membres des systèmes directs restreints à Ω_i et les y_j^n sont les états directs des sous-domaines voisins à Ω_j.
Ensuite, l'état adjoint discret local p_i^{n+1} à la $(n+1)$ème itération se calcule en transposant le système local obtenu par (8.6). Quant aux bords du recouvrement commun à Ω_i et Ω_j, les valeurs de p_i^{n+1} sont celles du sous-domaine voisin de telle sorte que l'on puisse écrire le système que vérifie p_i^{n+1} par l'équation formelle :

$$A_i^t P_i^{n+1} = G_i(y_i^{n+1} - y_{d,i}, p_j^n|_{\partial\Omega_i \cup \Omega_j}) \tag{8.7}$$

■

Nous avons effectué des tests pour un maillage de 256×64 et pour différentes valeurs du recouvrement entre les sous-domaines. Le problème de contrôle optimal est le même que précédemment. Nous faisons varier le nombre de processeurs pour chaque recouvrement et nous regardons ensuite l'influence de chacun de ces paramètres sur le comportement de l'optimiseur, ce qui revient à priori à étudier la précision obtenue dans chaque simulation de notre problème-test pour calculer le gradient et la fonction coût.

La méthode de minimisation employée dans cette partie est `N1QN3` et les systèmes (8.6) et (8.7) sont résolus à l'aide du même solveur de Krylov dont il a été question précédemment, mais cette fois, il est utilisé localement (sur chaque sous-domaine étendu). Dans ce cas, nous avons choisi une précision suffisante dans le solveur de Krylov pour que

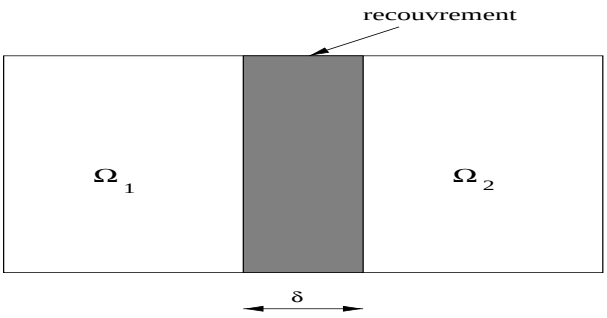

FIG. 8.9 – *Recouvrement de longueur δ entre deux sous-domaines.*

le déroulement de l'optimisation ne pose pas de problème. La tolérance sur le gradient dans l'optimiseur est de $\epsilon_{grad} = 1.e-6$. Tous les tests de cette partie ont été effectués sur le Cray T3E.

Avant de présenter les différentes propriétés de cette deuxième méthode, nous précisons la définition du recouvrement que nous allons utiliser dans toute la suite. Le recouvrement entre deux sous-domaines est, par définiton, le δ schématisé sur la figure 8.9. Quant au recouvrement qui figure sur les tableuax de résultats est donné par δ_i tel que $\delta = 2.\delta_i.h$, où h est le pas de discrétisation du maillage. (en fait, δ_i est la moitié du nombre de points constituant le recouvrement). Par la suite, nous utilisons le terme de recouvrement pour δ_i au lieu du vrai recouvrement δ.

8.3.1 Coût des calculs avec Schwarz multiplicatif et influence du recouvrement

A partir du tableau 8.4, la première caractéristique que nous remarquons en utilisant Schwarz multiplicatif est que les temps CPU sont prohibitifs. Mais, nous observons une nette amélioration quand le nombre de processeurs est élevé et quand nous augmentons le recouvrement δ_i. Sur la figure 8.10, les temps sont meilleurs avec $\delta_i = 8$ qu'avec $\delta_i = 4$ et à 16 processeurs, ces temps ont fortement diminué comparés à ce qui est calculé pour 2 ou 4 processeurs. Cependant, nous restons loin des performances obtenues avec une parallélisation directe de la première méthode.

8.3.2 Influence de la décomposition

Quant à l'influence de la décomposition, il est évident que l'on communique moins avec une décomposition en bande qu'avec une décomposition en grille. En effet, bien que les données transférées entre les sous-domaines voisins soient de dimension plus petite, nous en communiquons beaucoup et c'est bien cela qui coûte cher en temps CPU sur une machine parallèle. Il est à préciser que pour l'implantation de cette deuxième méthode avec une décomposition du domaine en grille, nous avons colorié selon 4 couleurs les domaines affectés aux processeurs et nous avons ainsi levé la dépendance entre les

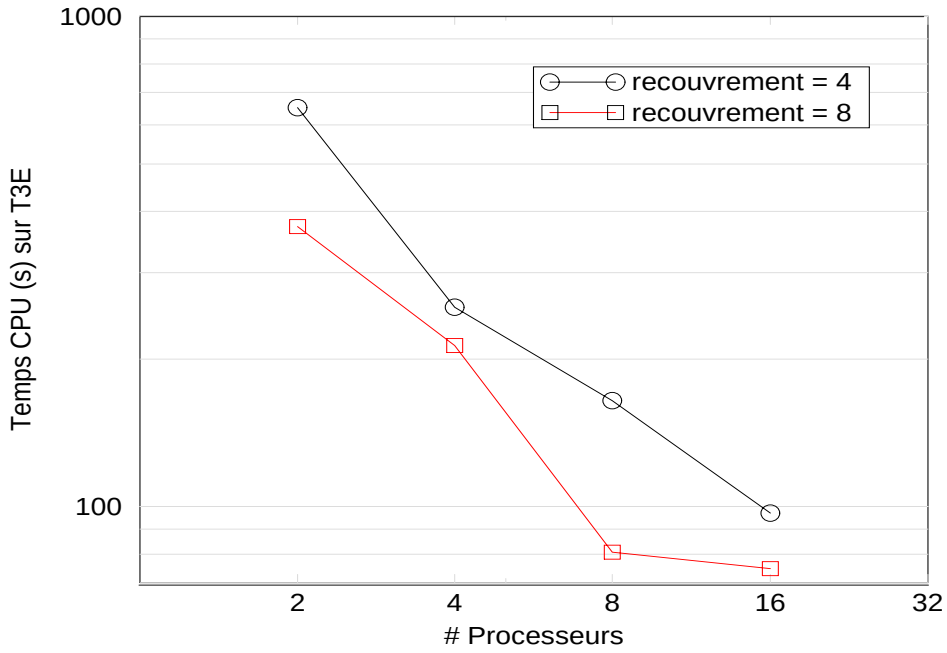

FIG. 8.10 – *Temps CPU (s) en fonction du nombre de processeurs. Influence du recouvrement sur le comportement de* `N1QN3` *avec Schwarz multiplicatif.* $N = 64$.

recouvrement δ_i	nb_procs	temps CPU (s)	nb_iter_`N1QN3`
4	$2 = 2 \times 1$	650.74	11
	$4 = 2 \times 2$	255.15	15
	$4 = 4 \times 1$	374.88	18
	$8 = 4 \times 2$	713.73	12
	$8 = 8 \times 1$	164.32	15
	$16 = 16 \times 1$	97.03	12
8	$2 = 2 \times 1$	372.85	12
	$4 = 4 \times 1$	213.14	18
	$8 = 4 \times 2$	526.80	15
	$8 = 8 \times 1$	80.73	11
	$16 = 16 \times 1$	74.71	10

TAB. 8.4 – $N = 64$, *optimiseur =* `N1QN3` , *Méthode =* `N1QN3` *avec Schwarz multiplicative. Comportement de* `N1QN3` *en fonction du recouvrement et du nombre de processeurs. (Calcul effectué sur Cray-T3E).*

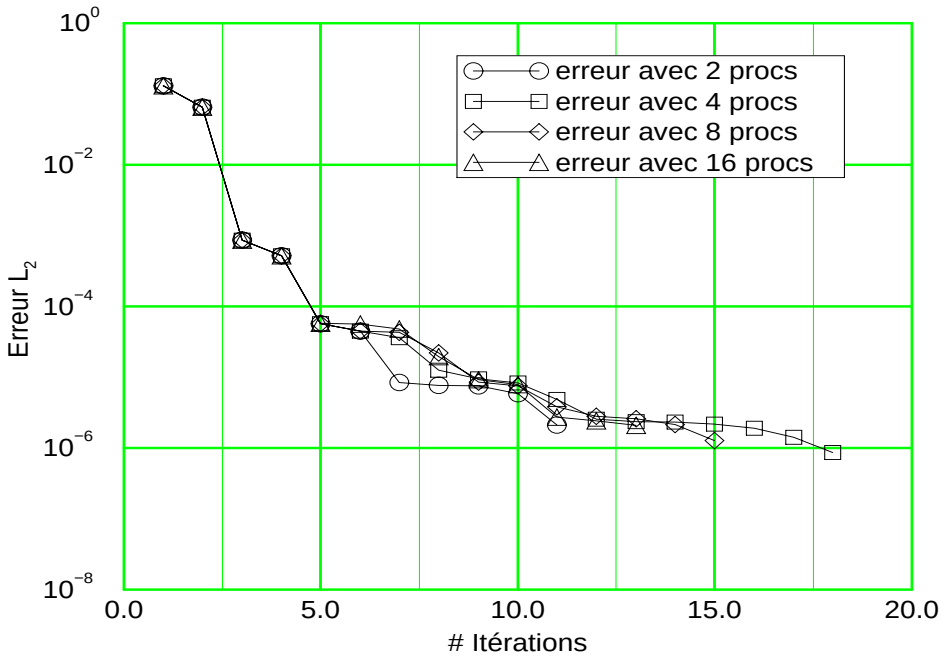

FIG. 8.11 – *Erreur L^2 en fonction du nombre d'itérations. Résultats de* `N1QN3` *obtenus avec Schwarz multiplicatif pour différents nombres de processeurs, $N = 64$, $\delta_i = 4$.*

sous-domaines qui peuvent dès lors effectuer leur calcul en parallèle même dans la phase Schwarz multiplicative. Seules les décompositions en grille en 4 et en 8 sous-domaines ont été présentées, *cf.* tableau 8.4, les calculs devenant très coûteux pour un nombre de processeurs supérieur à cause de l'augmentation du nombre d'itérations.

■

Nous passons maintenant à l'influence de l'algorithme Schwarz multiplicatif sur le déroulement de l'optimisation avec `N1QN3` :

8.3.3 Comportement de `N1QN3` avec Schwarz multiplicative

Sur le tableau 8.4, nous relevons aussi le nombre d'itérations de `N1QN3` : il reste pratiquement constant quand le nombre de sous-domaines est plus grand et dans ce cas il est très proche de celui obtenu avec un seul domaine et qu'il ne dépend pas du recouvrement entre les sous-domaines voisins. Cela signifie que l'on a une bonne précision de calcul dans le simulateur de notre problème de contrôle optimal.
Pour avoir une idée sur ce qui se passe dans l'optimiseur et expliquer la variation du nombre d'itérations, nous représentons sur la figure 8.11, les erreurs L^2 entre la solution

calculée par Schwarz multiplicative et la solution exacte pour différents nombres de processeurs dans le cas du recouvrement $\delta_i = 4$. Comme nous le remarquons, ce n'est qu'à partir de la 5ème itération que les différences entre les nombres de processeurs sont observées: avec 2 processeurs, l'erreur diminue rapidement alors qu'avec un nombre plus élevé de processeurs, elle décrît plus lentement.

Conclusion 3.

On peut donc conclure que la méthode de Schwarz multiplicative utilisée dans la résolution du problème de contrôle optimal est robuste mais elle est coûteuse.

■

Après avoir présenté une méthode de Schwarz en tant que solveur, nous revenons à la parallélisation des systèmes direct et adjoint. Mais, nous utilisons cette fois, une méthode de Schwarz comme préconditionneur du solveur de Krylov `BiCGSTAB` distribué.

8.4 Utilisation d'un solveur de Krylov préconditionné par une méthode de Schwarz pour la résolution des états direct et adjoint

Utilisées comme préconditionneurs des solveurs de Krylov, les méthodes de décomposition de domaine avec recouvrement telles que les méthodes alternées de Schwarz (additive ou multiplicative), permettent d'améliorer le taux de convergence de ces algorithmes et de limiter le nombre de communications nécessaires à leur implantation sur machine parallèle.

Remarque 8.4.1 *Nous trouverons dans [Smith et al., 1996], les performances comparées sur un Laplacien à l'ordre 2 pour un solveur GMRES (un solveur itératif de systèmes linéaires) préconditionné par un algorithme de Schwarz additif et multiplicatif ainsi qu'une étude théorique du conditionnement des solveurs de Krylov préconditionnés par les méthodes de décomposition de domaine.*

Remarque 8.4.2 *L'algorithme de Schwarz additif a l'avantage d'avoir un taux de parallélisme élevé. Ainsi, en l'utilisant comme préconditionneur d'un solveur de Krylov, il présente de bonnes propriétés de convergence.*

■

Dans ce paragraphe, nous nous limitons au cas de l'optimisation avec `N1QN3` , car avec le gradient conjugué, nous nous heurtons au choix judicieux de la tolérance d'arrêt qui

diffère chaque fois que nous changeons ou bien le nombre de processeurs ou bien la nature de la décomposition (en bande ou en grille).

8.4.1 Comment est défini le préconditionnement ?

Pour rester dans le même cadre de comparaison, nous considérons dans cette partie également, un maillage du domaine global de 256×64. Si nous revenons à la figure 8.3, le préconditionneur est donné par la matrice $\mathbf{M}$. Nous préconditionnons donc, le solveur de Krylov "`BiCGSTAB` " par une méthode de décomposition de domaine : "**Schwarz additif**" pour résoudre à chaque itération $\mathbf{M}\widehat{\mathbf{p}} = \mathbf{p}$ ou $\mathbf{M}\mathbf{z} = \mathbf{s}$. Dans cette étape de préconditionnement, nous étendons, tout d'abord, la contribution locale de $\mathbf{s}$ ou $\mathbf{p}$ au sous-domaine étendu par le recouvrement. Sur chaque sous-domaine, nous résolvons exactement un problème local, cette fois avec des conditions aux limites Dirichlet homogène [Smith et al., 1996]. Et enfin, la solution globale $\widehat{\mathbf{p}}$ ou $\mathbf{z}$ est déduite à partir des projections de la solution de chaque problème local.

Pour cette méthode également, les ϵ et ϵ_{stop} de l'algorithme, donnés sur la figure 8.3, sont pris suffisamment petits pour avoir une précision suffisante dans une simulation du problème de contrôle optimal. Tous les tests de cette partie sont obtenus sur le T3E.

8.4.2 Quelques résultats de `BiCGSTAB` préconditionné par Schwarz additif

- Tout d'abord, les temps CPU sont meilleurs comparés à ceux obtenus par la méthode Schwarz multiplicative. Mais ils restent relativement supérieurs à ceux de la parallélisation directe. Par contre, nous gagnons bien au niveau parallélisme et ce résultat est illustré sur le tableau 8.5: Pour $\delta_i = 8$, le temps CPU est divisé par deux quand le nombre de processeurs passe de 4 à 8 pour une décomposition en bande. Et nous avons le même nombre d'itérations dans `N1QN3` .

- Une deuxième remarque concerne le nombre d'itérations de `N1QN3` : il est relativement grand avec 4 processeurs mais il diminue progressivement quand le nombre de processeurs augmente pour rester constant avec 16 et 32 processeurs.

- Quant à l'influence du recouvrement, nous observons sur la figure 8.12 que les temps sont peu élevés avec $\delta_i = 4$ et mieux encore quand le calcul est distribué sur 16 processeurs.

- Le dernier point à relever des tests de cette partie (tableau 8.5), est l'influence très importante de la décomposition en grille sur les temps CPU ainsi que sur le comportement de l'optimiseur : Le nombre d'itérations de `N1QN3` avec une décomposition en bande est inférieur à celui avec une décomposition en grille *cf.* figure 8.13.
  Nous montrons également cette différence en traçant sur la figure 8.14, les erreurs

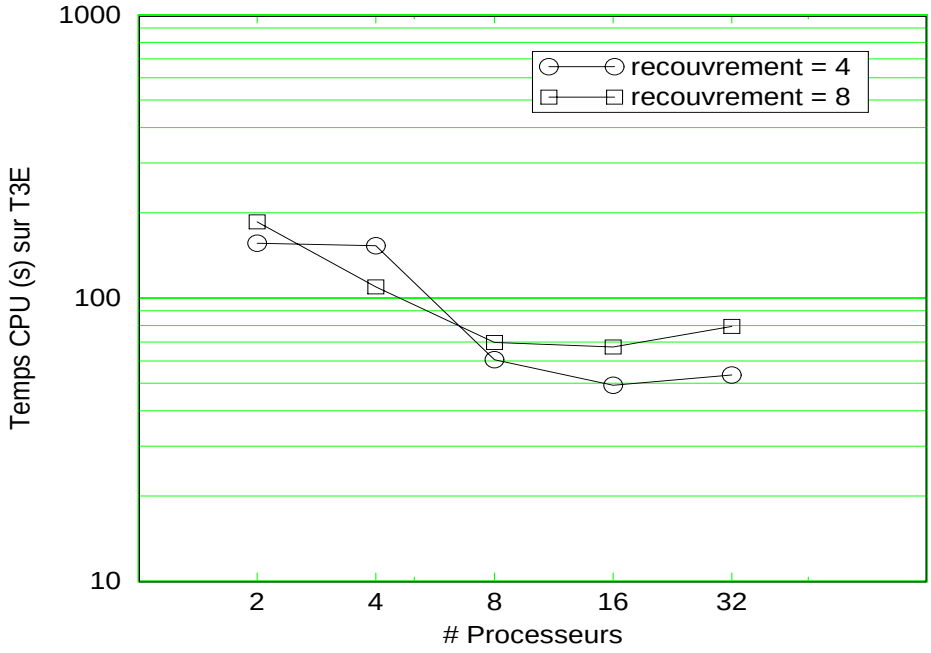

FIG. 8.12 – *Temps CPU (s) sur Cray-T3E en fonction du nombre de processeurs. Influence du recouvrement sur le comportement de* `N1QN3` *avec* `BiCGSTAB` *préconditionné par Schwarz additif.* $N = 64$.

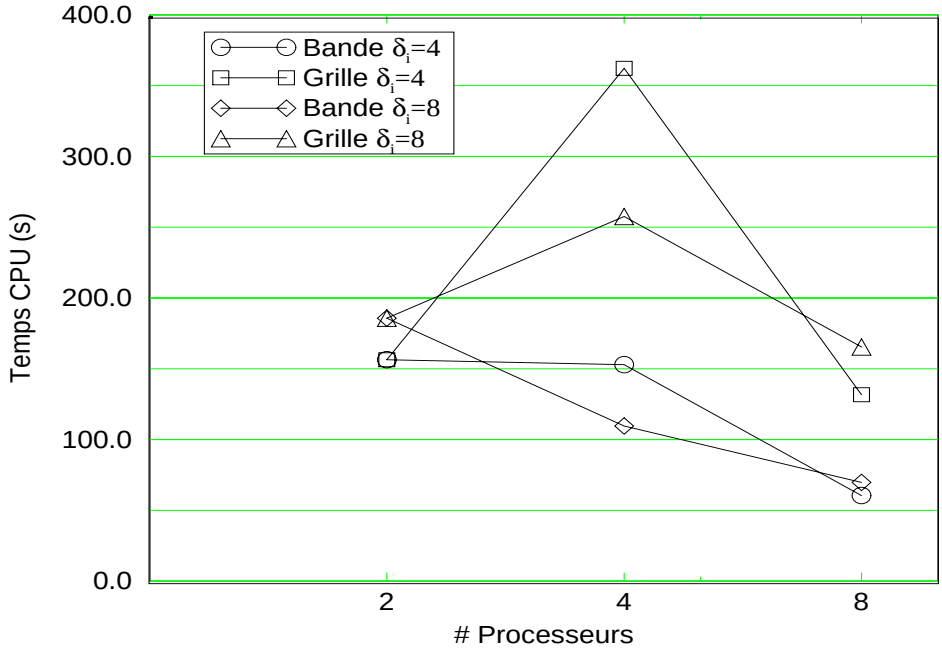

FIG. 8.13 – *Temps CPU (s) sur T3E en fonction du nombre de processeurs. Influence du type de la décomposition (grille ou bande) sur les temps CPU (s) de l'optimiseur* `N1QN3` *avec* `BiCGSTAB` *préconditionné par Schwarz addtive.* $N = 64$. $\delta_i = 4, 8$.

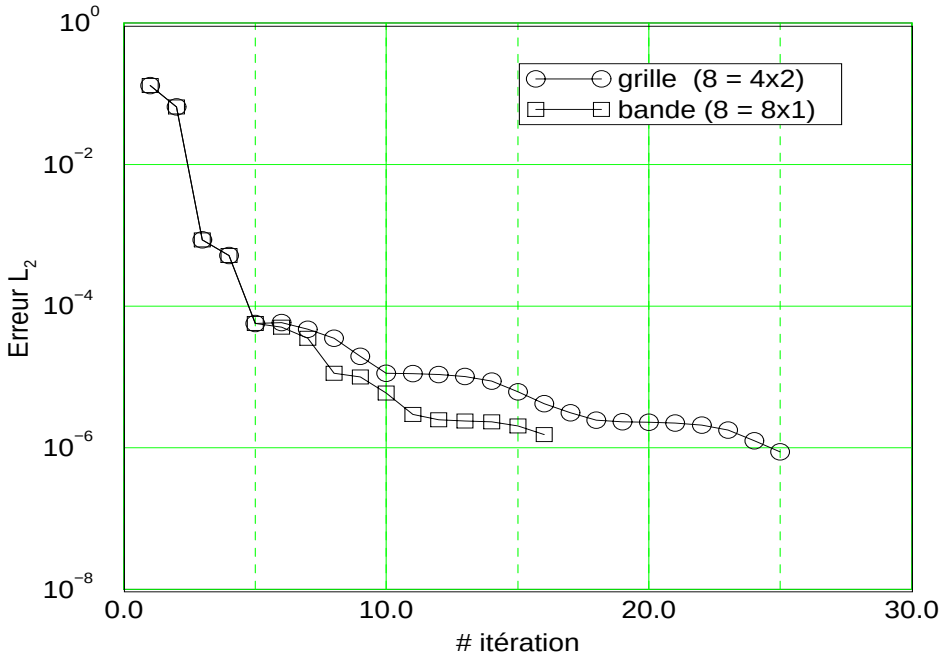

FIG. 8.14 – *Erreur L^2 de l'état direct (associé au contrôle optimal) avec la solution exacte, en fonction du nombre d'itérations de* `N1QN3` *. Influence du type de la décomposition pour 8 sous-domaines sur* `N1QN3` *avec* `BiCGSTAB` *préconditionné. $N = 64$. (Calcul effectué sur Cray-T3E).*

recouvrement δ_i	nb_procs	temps CPU (s)	nb_iter_N1QN3
4	$2 = 2 \times 1$	156.26	10
	$4 = 2 \times 2$	362.22	18
	$4 = 4 \times 1$	152.91	20
	$8 = 4 \times 2$	131.67	13
	$8 = 8 \times 1$	60.52	12
	$16 = 16 \times 1$	49.22	12
	$32 = 32 \times 1$	53.58	12
8	$2 = 2 \times 1$	185.73	12
	$4 = 2 \times 2$	257.46	15
	$4 = 4 \times 1$	109.59	15
	$8 = 4 \times 2$	165.43	25
	$8 = 8 \times 1$	69.64	16
	$16 = 16 \times 1$	67.16	20
	$32 = 32 \times 1$	79.31	12

TAB. 8.5 – $N = 64$, *optimiseur =* `N1QN3` *, Comportement de* `N1QN3` *en fonction du recouvrement et du nombre de processeurs avec* `BiCGSTAB` *préconditionné par Schwarz additive. (Calcul complet sur Cray-T3E).*

L_2 entre la solution calculée et la solution exacte au cours des itérations de `N1QN3` pour chaque cas, lorsque l'on considère une décomposition en 8 sous-domaines. Nous rappelons que cela s'explique par le coût des communications avec une grille et plus particulièrement, le préconditionneur "*va chercher les données dont il a besoin dans un entourage plus large quand le recouvrement est plus grand*".

8.4.3 Influence de `BiCGSTAB` préconditionné sur le comportement de l'optimiseur `N1QN3`

Le nombre d'itérations de `N1QN3` est presque constant quand $\delta_i = 4$ sauf pour la décomposition en 4 sous-domaines.
Nous présentons sur la figure 8.15, le comportement de `N1QN3` avec 8, 16 et 32 processeurs: le test d'arrêt de l'optimiseur est pris ($\epsilon_{grad} = 1.e-6$) pour tous ces calculs. Bien que nous ayons le même nombre d'itérations pour ces 3 cas tests, nous remarquons une légère amélioration quand le nombre de processeurs est élevé et une meilleure convergence. Nous observons également sur cette figure que jusqu'à la 7ème itération, les résultats de l'optimiseur sont les mêmes dans le cas des trois tests. Mais, à partir de la 8ème itération, il y a un net avantage quand le nombre de processeurs est maximum (pour ce groupe de tests).
Nous expliquons ces différences éventuellement par une accumulation d'erreurs d'arrondi.

A la fin de ces tests, nous pouvons conclure que :

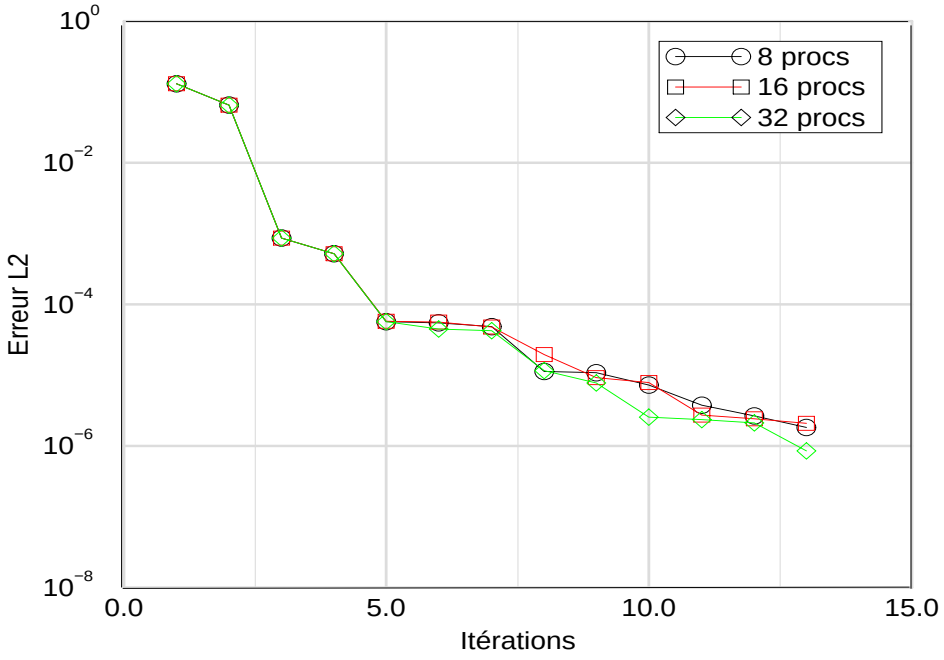

FIG. 8.15 – *Erreur L^2 ($\|y - y_{ex}\|_{L^2}$) en fonction du nombre d'itérations. Comportement de* `N1QN3` *avec* `BiCGSTAB` *préconditionné en variant le nombre de processeurs. $N = 64$, $\delta_i = 4$. (Calcul effectué sur Cray-T3E).*

Conclusion 4.

Les calculs avec `BiCGSTAB` préconditionné par Schwarz additif sont moins coûteux qu'avec Schwarz multiplicatif utilisé en tant que solveur. Le comportement de l'optimiseur `N1QN3` est légèrement influencé par l'usage de `BiCGSTAB` préconditionné.

8.5 Conclusions

Les méthodes parallèles, décrites et comparées dans ce chapitre pour résoudre des problèmes de contrôle optimal, combinent à la fois l'usage des méthodes de minimisation (de type gradient conjugué ou quasi-Newton) et des algorithmes itératifs de Schwarz utilisés soit

- comme solveurs: nous en avons utilisé la version multiplicative qui fait l'objet de la méthode du paragraphe 8.3,

- ou bien comme préconditionneurs: c'est la version additive que nous avons employée puisqu'elle présente de bonnes propriétés de convergence pour préconditionner le solveur de Krylov `BiCGSTAB` . (paragraphe 8.4).

Nous avons comparé ces deux méthodes avec une parallélisation directe de `BiCGSTAB` retenu pour résoudre les systèmes direct et adjoint. Nous tirons alors, les conclusions suivantes :

1. En utilisant les 3 algorithmes testés, nous arrivons à calculer la solution de notre problème-test de contrôle optimal pour une précision donnée.

2. D'après nos tests, `N1QN3` est plus robuste que `GradCjg` et il est peu sensible aux erreurs d'arrondi.

3. Pour un problème de grande taille, nous gagnons en efficacité parallèle dans le cas de `BiCGSTAB` distribué et plus encore si nous disposons les processeurs en grille à partir d'un nombre élevé (typiquement ≥ 16 processeurs).

4. En utilisant la méthode de Schwarz multiplicative comme solveur, le meilleur résultat est obtenu pour un recouvrement relativement grand. Le comportement de `N1QN3` est fortement influencé par la nature de la décomposition du domaine. A la suite des résultats présentés, nous concluons que `N1QN3` avec Schwarz multiplicatif constitue un algorithme du problème de contrôle optimal, robuste mais coûteux.

5. Dans le cas où une méthode de Schwarz, ici additive, est utilisée comme préconditionneur, le calcul est moins coûteux que celui obtenu par Schwarz utilisé en tant que solveur. Les meilleurs résultats sont obtenus avec un recouvrement relatif de

25%. Pour des problèmes de "petite taille", la décomposition en grille induit des communications plus nombreuses et donc coûteuses).

6. Les temps CPU obtenus par `BiCGSTAB` préconditionné, pour $N = 64$ (sur le tableau 8.4) et à partir de 8 processeurs, sont proches de ceux obtenus avec `BiCGSTAB` distribué pour $N = 128$, voir tableau 8.3. Et au vu de l'allure de la courbe 8.12, nous pouvons avancer que `BiCGSTAB` préconditionné par Schwarz additive sera plus compétitif à grand nombre de degrés de liberté d'autant plus qu'on peut accélérer la résolution des problèmes dans la phase de préconditionnement par des solveurs de type multigrille bien adaptés à la résolution du Laplacien.

7. L'application à un problème parabolique est immédiate parce que l'on considère une décomposition du domaine de calcul en espace. Nous ne présentons pas de résultats numériques de l'implantation parallèle des méthodes de ce chapitre pour de tels problèmes.
   Nous retenons par contre que toutes les propriétés déduites pour le Laplacien (équation d'état de notre problème stationnaire) restent valables pour le cas du problème parabolique (une suite de problèmes elliptiques à chaque pas de temps).

Conclusion.

Dans cette étude, nous avons proposé deux approches différentes permettant de développer des méthodes numériques basées sur les méthodes de décomposition de domaine pour résoudre des problèmes de contrôle optimal gouvernés par des équations aux dérivées partielles.

Dans un premier temps, nous nous sommes intéressés à une méthode de décomposition du domaine sans recouvrement: les raccordements entre les sous-domaines ayant été traités par des techniques de multiplicateurs de Lagrange. Nous avons montré que sur le domaine décomposé, le problème global se ramène à une recherche de point-selle du Lagrangien augmenté associé au nouveau problème d'optimisation. Nous avons alors proposé un algorithme de résolution basé sur l'algorithme d'Uzawa. Mais, les résultats, à l'exception de quelques cas particuliers, n'ont pas été à la hauteur de nos attentes. En effet, d'après les tests faits, nous observons toujours un "petit saut" au voisinage de l'interface dû à la difficulté de contrôler les nombreuses variables supplémentaires (les multiplicateurs de Lagrange λ_{ij}, les ω_{ij} et le coefficient r). On pourrait tenter de préconditionner le problème de l'interface par le choix (ou la construction) d'un produit scalaire qui prendrait mieux en compte les raccordements entre les sous-domaines.

Puis dans un deuxième temps, nous avons introduit des méthodes de décomposition de domaine avec recouvrement basées sur les algorithmes itératifs de Schwarz utilisés comme solveurs ou préconditionneurs pour résoudre les systèmes linéaires direct et adjoint du problème de contrôle optimal étudié. Dans ce cas, nous avons obtenu de bonnes propriétés de convergence pour 3 algorithmes utilisant soit

- un solveur de Krylov distribué: `BiCGSTAB` ,
- un algorithme de Schwarz multiplicative en tant que solveur,
- un solveur de Krylov préconditionné par une méthode de Schwarz.

Les résultats de cette partie sont très satisfaisants. D'une part, nous avons construit un algorithme parallèle en utilisant la méthode itérative de Schwarz multiplicative en tant que solveur. Ceci permet de construire naturellement l'état adjoint par transposition des systèmes locaux. L'algorithme global défini par la méthode de minimisation de type quasi-Newton et ce solveur de Schwarz constitue une méthode robuste de résolution du

problème de contrôle optimal, mais coûteuse. On pourrait chercher un recouvrement relatif optimal de telle manière que le coût soit lui aussi optimal tout en donnant de bons résultats de convergence.
D'autre part, et plus particulièrement, pour des problèmes de grande taille, l'algorithme de type quasi-Newton combiné avec le solveur de Krylov `BiCGSTAB` préconditionné par une méthode de Schwarz additive est plus compétitif dans la mesure où l'on obtient de bonnes preformances parallèles.

Une extension de ce travail serait d'appliquer les techniques de la partie avec recouvrement à un problème réel, par exemple, le modèle quasi-géostrophique, ou en général des problèmes dont la discrétisation des ses états direct et adjoint se ramène à la résolution de systèmes linéaires. L'implantation, dans ce cas, serait bien adaptée à partir de 16 processeurs.
En continuant de tirer partie des bonnes propriétés de convergence des méthodes avec recouvrement, on pourrait paralléliser efficacement la part de l'optimiseur et ceci même si elle reste proportionnellement négligeable. On pourrait également optimiser les communications, accélérer la résolution des problèmes de coût dans la phase de préconditionnement soit par les méthodes directes par blocs ou par des solveurs multigrilles. D'autre part, pour des problèmes évolutifs, on pourra tirer profit de ces méthodes si l'on a une version "bien optimisée" du code séquentiel de telle sorte que la part de l'optimiseur soit assez négligeable.

Annexe **A :**

Solveurs de Krylov.

Soit à résoudre le système linéaire

$$Ax = b, \tag{A.1}$$

où A est une matrice non nécessairement symétrique ou définie positive, des propriétés pour lesquelles existent différentes méthodes directes ou itératives qui en prennent compte. Nous allons nous intéresser à des méthodes itératives pour résoudre le système $(A.1)$ qui génèrent des itérés successifs $x^0, x^1, x^2, ..., x^k, ...$ en partant d'un vecteur initial x^0 donné. Avec les notions suivantes :

- $x^\star$: est la solution exacte de $(A.1)$,
- $r^k = b - Ax^k$: est le résidu obtenu à la k-ième itération,
- $e^k = x^\star b - x^k$: est l'erreur (ou correction) qu'il faudrait ajouter à x^k pour obtenir la solution exacte $x^\star$,

Ainsi, résoudre le système ci-dessus revient à inverser le système :

$$Ae^k = r^k. \tag{A.2}$$

Le plus simple des algorithmes itératifs est la méthode de "*Richardson*" ou méthode de gradient à pas fixe ρ préconditionnée par l'opérateur C, c'est à dre,

$$x^{k+1} = x^k + \rho C^{-1} r^k \tag{A.3}$$

avec $C^{-1}r^k$ approche au mieux $A^{-1}r^k$ de façon à ce que le produit $C^{-1}r^k$ soit peu coûteux à calculer. Ce préconditionneur C (ou C^{-1}) n'est pas calculé explicitement dans la plupart des temps et classiquement, calculer $z^k = C^{-1}r^k$ se ramène par exemple à faire quelques itérations de Jacobi ou de Gauss-Seidel sur le système $Az^k = r^k$, ou encore appliquer une itération d'une méthode de décomposition de domaine de type Schwarz avec recouvrement. Un choix judicieux pour ρ peut accélérer significativement la convergence de l'algorithme. Le problème majeur de la méthode de Richardson consiste non seulement à trouver un bon préconditionneur C^{-1} de telle sorte que $\kappa(C^{-1}A)$ soit proche de 1 mais, aussi, à déterminer un ρ optimal, à conaître une bonne estimation des valeurs propres maximales et minimales de $C^{-1}A$ qui ne sont généralement pas connues.

Le principe des méthodes de Krylov est de pallier ce manque d'information sur le spectre de $C^{-1}A$ en raisonnant sur les vecteurs intervenant dans plusieurs itérations de Richardson successives et en écrivant une équation de "correction" telle que :

$$u^{k+1} = u^k + p^k, \tag{A.4}$$

où la direction de descente p^k est un vecteur du "**sous-espace de krylov**":

$$Vect\{C^{-1}r^k, (C^{-1}A)C^{-1}r^k, (C^{-1}A)^2C^{-1}r^k, ..., (C^{-1}A)^{k+p-1}C^{-1}r^k\}$$

choisi de façon à minimiser $b - Ax^{k+1}$.

Parmi les solveurs de Krylov les plus utilisés, il y a les solveurs GMRES [Saad, 1985], [Saad et Schultz, 1986] et Bi-CGSTAB [Sonnelved, 1989]:

Dans le premier algorithme "GMRES", (Generalized Minimal RESidual algorithm) mis au point par Youcef Saad, la direction de descente p^k se calcule en prenant en compte toutes les directions de descente précédentes pour obtenir un espace de Krylov de dimension maximale k. Une description complète de cet algorithme est donnée dans [Saad et Schultz, 1986]. GMRES nécessite le stockage coûteux de k vecteurs de travail. Lorsque la convergence demande plusieurs itérations $n \geq k$, on utilise le "*restarted GMRES*", qui consiste à stocker au plus k vecteurs de travail et à redémarrer l'algorithme à partir du nouveua résidu lorsque l'on a épuisé ces vecteurs de travail. (c'est en quelque sorte un GMRES à mémoire limitée).
Les performances de GMRES pour une application à des problèmes de mécanique de fluides sont détaillées dans [Van der Vorst et Vuik, 1994].

Quant au deuxième algorithme `BiCGSTAB` , nous en présentons un apreçu ci-après:

Algorithme de Gradient Bi-Conjugué

`BiCGSTAB` est un algorithme de la famille des méthodes de **gradient bi-conjugué**, c'est un algorithme basé sur des sous-espaces de Krylov (*cf.* Annexe A.).
A la fin des années 80, Sonnelved et Van der Vorst développent des techniques de gradient bi-conjugué basées comme dans le cas de gradient conjugué sur une récurrence d'ordre 2 mais restent encore valables pour des systèmes linéaires non-symétriques. Sonnelved met au point `CGS` (Conjugate Gradient Squared) [Sonnelved, 1989] et Van der Vorst en propose peu après, une amélioration : `Bi-CGSTAB` qui stabilise la convergence [Van der Vorst, 1992] (la description de l'algorithme est donnée dans l'annexe 1.). Une combinaison de k `BiCGSTAB` donne `BiCGSTAB` (k) et ces deux algorithmes sont comparés dans [Sleijpen et Fokkema, 1993].
Nous précisons que bien que la convergence de Bi-CGSTAB soit stabilisée par rapport à celle de CGS, elle n'en est pas pour autant monotone.
Nous y reviendrons plus loin au chapitre 8. où nous allons parler de sa version parallèle ainsi que de sa version préconditionnée.

Soient $\mathtt{x}_0$, ϵ, ϵ_{stop} donnés
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{v} = \mathbf{p} = 0$
$\rho_0 = \alpha = \omega = 1$
Tant que $\frac{\|\mathtt{r}\|_2}{\|\mathtt{r}_0\|_2} > \epsilon$ et $\|\mathtt{r}\|_2 > \epsilon_{stop}$ faire
$\rho = ({\mathtt{r}_0}^T \mathtt{r})$
$\beta = \alpha\rho/\rho_0\omega$, $\rho_0 = \rho$
$\mathtt{p} = \mathtt{r} + \beta(\mathtt{p} - \omega\mathtt{v})$
Résoudre $\mathbf{C}\widehat{\mathtt{p}} = \mathtt{p}$
$\mathtt{v} = \mathbf{A}\widehat{\mathtt{p}}$
$\alpha = \rho_1/({\mathtt{r}_0}^T \mathtt{v})$
$\mathtt{s} = \mathtt{r} - \alpha\mathtt{v}$
Résoudre $\mathbf{C}\mathtt{z} = \mathtt{s}$
ÉCHANGE_BORDS (z)
$\mathtt{t} = \mathbf{A}\mathtt{z}$
$\omega = \dfrac{(\mathtt{t}^T\mathtt{s})}{(\mathtt{t}^T\mathtt{t})}$
$\mathtt{x} = \mathtt{x} + \alpha\widehat{\mathtt{p}} + \omega\mathtt{z}$
$\mathtt{r} = \mathtt{s} - \omega\mathtt{t}$
Fin

FIG. A.16 – *Algorithme BiCGSTAB*

Annexe **B :**

Calcul et implantation du modèle adjoint pour un problème de contrôle optimal régi par une edp parabolique.

Après discrétisation du modèle direct (voir Chapitre 7.), nous symbolisons le schéma temporel par

$$(I + h\Delta)Y^k = Y^{k-1} + hBF^k + hCV^k, \qquad k = 1, ..., NT, \tag{B.1}$$

où

- Y^k : est le vecteur d'état à l'instant k,
- h : est le pas de discrétisation en temps,
- NT : est le nombre des pas de temps,
- B : est l'opérateur : $F \mapsto$ source F ou 1er second membre discrétisé,
- C : est l'opérateur : $V \mapsto$ contrôle V ou 2ème second membre discrétisé.

En posant, $A = I + h\Delta$, le programme direct est organisé comme suit :

A chaque pas de temps, nous avons

$$\begin{aligned} RHS1 &:= Y^{k-1} \\ RHS2 &:= B * F^k \\ RHS3 &:= C * V^k \\ RHS &:= RHS1 + h * (RHS2 + RHS3) \\ Y^k &:= A^{-1} * RHS \end{aligned}$$

$$\left(\begin{array}{ccccc|c|c|c} -I & A & & & & -hB & & -hC \\ & -I & A & & & & -hB & & -hC \\ & \ddots & \ddots & \ddots & & & \ddots & & \ddots \\ & & a_{M-1} & -I & A & & & -hB & & -hC \\ & & & & -I & A & & & -hB & -hC \end{array}\right)$$

A chaque ligne de cette "grande" matrice, je fais correspondre un vecteur ajoint $(P^1, \cdots, P^{NT})$. J'ai donc le "schéma" suivant:

$$\cdot \left(\begin{array}{ccccccccccccccccc} Y^0 & Y^1 & Y^2 & \cdots & Y^{NT-1} & Y^{NT} & & F^1 & F^2 & \cdots & F^{NT-1} & F^{NT} & & V^1 & V^2 & \cdots & V^{NT-1} & V^{NT} \end{array} \right)^t = 1$$

Y^0	Y^1	Y^2	$\cdot$	Y^{NT-1}	Y^{NT}	F^1	F^2	$\cdot$	F^{NT-1}	F^{NT}	V^1	V^2	$\cdot$	V^{NT-1}	V^{NT}	
$-I$	A					$-hB$					$-hC$					P^1
	$-I$	A					$-hB$					$-hC$				P^2
	$\ddots$	$\ddots$	$\ddots$					$\ddots$					$\ddots$			$\vdots$
		a_{M-1}	$-I$	A					$-hB$					$-hC$		P^{NT-1}
				$-I$	A					$-hB$					$-hC$	P^{NT}

Supposons que la fonction coût s'écrive:

$$J = \frac{1}{2} \sum_{k=1}^{NT} \|Y^k - Y^k_{obs}\|^2$$

alors, le schéma adjoint rétrograde après transposition est:

$$\begin{aligned} P^{NT} &= A^{-t}\left(Y^{NT} - Y^{NT}_{obs}\right) \\ \frac{\partial J}{\partial V^{NT}} &= hC^t \cdot P^{NT} \end{aligned}$$

$$\begin{aligned} \text{Pour } k := (NT-1) \quad &\text{à} \quad 1 \text{ faire :} \\ \text{Résoudre } A^t P^k &= \left(Y^k - Y^k_{obs}\right) + P^{k-1} \\ \text{Puis } \frac{\partial J}{\partial V^k} &= hC^t \cdot P^k. \end{aligned}$$

Annexe **C :**

Un peu d'histoire sur la méthode alternée de Schwarz.

Cette annexe constitue une version française d'une partie de l'introduction faite par O. Widlund pour les proceedings de la 3ème conférence internationale sur les méthodes décomposition de domaine [Chan et al., 1990]. Elle décrit plus ou moins l'histoire de la plus ancienne des méthodes de décomposition de domaine: **méthode de Schwarz** ainsi que ses récentes extensions.

Algorithmes "diviser pour régner"

Les méthodes de décomposition de domaine semblent être les plus prometteuses pour la résolution parallèle des très souvent important systèmes d'équations linéaires ou non-linéaires issus de problèmes élliptique d'élasticité, de mécanique des fluides et d'autres importants domaines d'application qui sont discrétisés au moyen d'éléments finis ou de différences finies. Ces méthodes peuvent être vues comme des algorithmes de type "diviser pour règner" consistant en autant de sous-problèmes correspondant à des sous-domaines

d'un modèle qui peuvent être résolus au moyen de techniques connues avec une interaction de ces sous-domaines assurée par une itération qui "transmet" l'information adéquate de part et d'autre des interfaces divisant le domaine global. Lorsque les sous-problèmes sont nombreux, l'utilisation d'un modèle "grossier" peut grandement améliorer le taux de convergence des itérations en calculant une solution approchée des interactions à grande échelle du modèle original. De ce point de vue, les méthodes de décomposition de domaine sont similaires aux méthodes multigrilles.

De nombreux algorithmes de décomposition de domaine ont tout d'abord été développés et étudiés pour des problèmes modèles linéaires de second ordre, élliptiques définis positif et auto-adjoints, et ce sujet fait encore l'objet de travaux. Cependant, alors que ce domaine parvient à maturité, l'accent fut mis sur des problèmes bien plus importants faisant intervenir des sous-domaines nombreux.

L'algorithme de Schwarz et quelques extensions récentes

La plus ancienne des méthodes de décomposition de domaine est communément attribué à Herman Amandus Schwarz qui découvrit la "méthode alternée" en 1869 (voir les travaux collectés de Schwarz, Gesammelete Mathemmatische Ablandlungen, Springer, Berlin 1890). Schwarz, à qui l'on doit des contributions nombreuses dans le domaine de l'analyse complexe, utilisa cette méthode pour établir l'existence de fonctions harmoniques sur des domaines à frontières non régulières. Ces domaines étaient construits comme une réunion successive de sous-domaines, en commençant par ceux dont l'existence de telles fonctions pouvait être démontrée par d'autres moyens.

L'algorithme de Schwarz donne une suite géométrique convergente de fonctions dont la limite est la fonction harmonique qui satisfait aux conditions aux limites données. La figure 1, tirée d'un des premiers articles de Schwarz, illustre cette méthode. Lors du premier pas d'une itération à pas fractionnaires qui en comporte deux, la valeur approchée de la solution sur T_1 calculée précédemment est remplacée par les valeurs de la fonction harmonique imposées comme conditions aux limites de type Dirichlet sur δT_1, frontière de T_2 calculées à l'itération précedente. (Au premier pas, on part d'une solution arbitraire). Le second pas fractionnaire, au cours duquel de nouvelles valeurs sont obtenues pour T_2 est mené de la même façon. Il est alors possible, en utilisant seulement des solveurs pour un domaine circulaire ou rectangulaire, de calculer une approximation arbitrairement précise de la fonction harmonique cherchée sur cette région. Des méthodes classiques de séparation des variables peuvent être utilisées pour ces sous-problèmes successifs. Bien que des algorithmes de décomposition de domaine à convergence rapide aient été développés pour des domaines de formes assez générales et différentes sortes d'équations différentielles, on peut utiliser, pour considérablement réduire le coût d'une itération, des algorithmes rapides de résolution, comme des solveurs de Poisson rapide et les méthodes spectrales qui sont disponibles sur des domaines de géométrie simple. Le principal problème relatif à la méthode de Schwarz - ainsi qu'aux autres méthodes de décomposition de domaine - est

de montrer que le taux de convergence par itération est satisfaisant, ou croît suffisament lentement lorsque la discrétisation des modèles est raffinée ou que l'on augmente le nombre de sous-domaines.

Il est intéressant de remarquer qu'il est possible de dissimuler l'utilisation de sous-domaines recouvrant dans l'algorithme de Schwarz. Une itération peut être vue en terme de recherche d'un point fixe pour la projection d'une famille de fonctions définies sur la courbe L_2. L'algorithme de Schwarz se ramène alors à une méthode de sous-structuration. La projection et son rayon spectral de convergence dépendent de l'étendue du recouvrement : une augmentation de celui-ci entrîne une augmentation des coûts de calcul mais aussi une amélioration du taux de convergence ce qui nécessite un compromis.

Au cours de sa présentation, Schwarz suggéra une analogie physique à son algorithme. Les interfaces L_1 et L_2 peuvent être assimilées à des valves. Lors du premier pas fractionnaire, on pompe l'air de T_1 avec L_1 ouverte et L_2 fermée. En inversant le rôle des sous-domaines et des valves, on répète ce même procédé pour compléter l'itération. L'analogie proposée par Schwarz peut être modifiée pour démontrer qu lors de chaque pas fractionnaire, la norme énergie de l'erreur ne s'accroît pas et qu'au moins une certaine fraction fixe de cette énergie est évacuée à chaque itération. Ce résultat peut être établi, le plus aisément dans le cadre d'espaces de Hilbert, où la méthode à pas fractionnaire est formulée en termes de projection sur des sous-espaces naturellement liés aux problèmes définis sur les sous-domaines.

Dans sa démonstration, le principal outil qu'utilisait Schwarz était le principe du maximum. Il y a plus de 50 ans, Sobolev établissait la formulation variationelle de l'algorithme de Schwarz (The Schwarz Algorithm in the Theory of Elasticity, Dokl. Acad. Nauk. USSR, Vol. IV, 1936). Le résultat de Sobolev offre un point de départ pour utiliser l'algorithme sur des problèmes ne vérifiant pas le principe du maximum, et rend possible l'extension de son analyse à de nombreux modèles d'éléments finis et à d'autres méthodes obtenues par un calcul de variations.

Pierre-Louis Lions, un intervenant des trois conférences lors des récents symposiums sur les méthodes de décomposition de domaine, a de nouveau examiné et étendu significativement quelques unes de ces idées classiques dans un certain nombre de directions. L'extension de la méthode de Schwarz au cas de nombreux sous-domaines est important car pour deux sous-domaines uniquement, l'algorithme est intrinsèquement séquentiel. En particulier, il est important de résoudre de nombreux sous-problèmes en parallèle de façon à garder petit le nombre de pas fractionnaires. C'est un des problèmes qui fut récemment examiné par Lions et d'autres.

Bibliographie

[Arrow et al., 1958] Arrow, K. J., Hurwicz, L., et Uzawa, H. (1958). Studies in nonlinear programming. In *Stanford University press*. Axelsson O.

[Benamou, 1994] Benamou, J.-D. (1994). Domain decomposition methods with coupled transmission conditions for the control of systems governed by elliptic partial differential equations. Technical Report 2246, INRIA.

[Benamou, 1996] Benamou, J.-D. (1996). A domain decomposition method with coupled transmission conditions for the control of systems governed by elliptic partial differential equations. *SIAM J. Numer. Anal.*, 33(6):2401–2416.

[Benamou, 1997] Benamou, J.-D. (1997). Résolution d'un cas test de contrôle optimal pour un système gouverné par l'équation des ondes à l'aide d'une méthode de décomposition de domaine. Technical Report 3095, INRIA.

[Benamou et Desprès, 1997] Benamou, J.-D. et Desprès, B. (1997). A domain decomposition method for the Helmholtz equation and related optimal control problems. *J. Comp. Physics*, (136):68–82.

[Bensoussan et al., 1973] Bensoussan, A., Glowinski, R., et Lions, J.-L. (1973). Méthode de décomposition appliquée au contrôle optimal de systèmes distribués. *Lecture Notes in Computer Science*, 5. In the fifth IFIP Conference on Optimization Techniques.

[Berggren et Heinkenschloss, 1997] Berggren, M. et Heinkenschloss, M. (1997). Parallel solution of optimal-control problems by time-domain decomposition. In *Computational Science for the 21st century*. John Wiley & Sons.

[Bjørstad, 1987] Bjørstad, P. (1987). A large scale, sparse, secondary storage, direct linear equationsolver for structural analysis and its implementation on vector parallel architectures. *J. Parallel Computing*, 5.

[Bjørstad et al., 1998] Bjørstad, P. E., Espedal, M., et Keyes, D., editors (1998). *Proceedings of the Ninth International Conference on Domain Decomposition Methods*. John Wiley & Sons. held in Bergen, Norway, June 1996.

[Bounaïm, 1998] Bounaïm, A. (1998). A Lagrangian approach to a ddm for an optimal control problem. In Bjørstad, P. E., Espedal, M., et Keyes, D., editors, *Domain Decomposition Methods in Sciences and Engineering*. John Wiley & Sons. Proceedings from the Ninth International Conference, held in Bergen, Norway, June 1996.

[Bramble et al., 1990] Bramble, J. H., Pasciak, J. E., et Xu, J. (1990). Parallel multilevel preconditioners. In SIAM, editor, *Proc. of Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*.

[Céa, 1971] Céa, J. (1971). *Optimisation: Théorie et Algorithmes*. Dunod, Paris.

[Cai, 1991] Cai, X.-C. (1991). Additive schwarz algorithm for parabolic convection diffusion equations. *Numer. Math.*, 60:41–62.

[Chan et al., 1989] Chan, T. F., Glowinski, R., Périaux, J., et Widlund, O. B., editors (1989). *Proceedings of the Second International Conference on Domain Decomposition Methods*, Philadelphia. SIAM.

[Chan et al., 1990] Chan, T. F., Glowinski, R., Périaux, J., et Widlund, O. B., editors (1990). *Proceedings of the Third International Conference on Domain Decomposition Methods*, Philadelphia. SIAM.

[Dawson et al., 1991] Dawson, C. N., Du, Q., et Dupont, T. F. (1991). A finite difference domain decomposition algorithm for the numerical solution of heat equation. *Math. Comput.*, 57:63–71.

[Dawson et Dupont, 1991] Dawson, C. N. et Dupont, T. F. (1991). Explicit/implicit conservative Galerkin domain decomposition method of a model heat equation. *Math. Comput.*, 58:63–71.

[Després, 1991] Després, B. (1991). Domain decomposition method and the Helmholtz problem. In Cohen, G., Joly, P., et Halpern, L., editors, *Mathematical and Numerical Aspects of Wave Propagation Phenoma*, pages 44–52.

[Després, 1993] Després, B. (1993). Domain decomposition method and the Helmholtz problem (part ii). In Kleimann, R., Angell, T., Colton, D., Santosa, F., et Stackgold, I., editors, *Second international conference on mathmatical and numerical aspects of wave propagation phenoma*, pages 197–206. SIAM.

[Dongara et al., 1994] Dongara, J. et al. (1994). Mpi : A message-passing interface standard. Technical report, Message Passing Interface Forum.

[Dryja, 1991] Dryja, M. (1991). Substructure methods for parabolic problems. In Glowinski, R., Kuznetsov, Y. A., Meurant, G. A., Périaux, J., et Widlund, O. B., editors, *Proceedings of the Foorth International Conference on Domain Decomposition Methods*, pages 264–271, Philadelphia. SIAM.

[Dryja et Widlund, 1989] Dryja, M. et Widlund, O. (1989). Some domain decomposition algorithms for elliptic problems. In *Proc. of Second International Symposium on Domain Decomposition Methods for Partial Differential Equations*.

[Ekeland et Temam, 1974] Ekeland, I. et Temam, R. (1974). *Analyse convexe et problèmes variationnels*. Dunod-Gauthier Villars, Paris.

[Espedal et Tai, 1997] Espedal, M. et Tai, X.-C. (1997). A space decomposition method for minimization problems. In Bjørstad, P. E., Espedal, M., et Keyes, D., editors, *Domain Decomposition Methods in Sciences and Engineering*. John Wiley & Sons. Proceedings from the Ninth International Conference, June 1996, Bergen, Norway.

[Faurre, 1986] Faurre, P. (1986). Note d'optimisation. Technical report, Ecole Polytechnique, Département de mathématiques Appliquées.

[Fortin et Glowinski, 1982] Fortin, M. et Glowinski, R. (1982). *Méthodes de Lagrangien augmenté, Application aux problèmes aux limites*. Bordas.

[Geist et al., 1993] Geist, A. et al. (1993). PVM 3 user's guide and reference manual. Technical report, Oak Ridge National Laboratory. available from netlib.

[Gilbert et Lemaréchal, 1989] Gilbert, J.-C. et Lemaréchal, C. (1989). Some numerical experiments with variable storage quasi-Newton algorithms. *Mathematical Programming*, 45:407–435.

[Glowinski et al., 1988] Glowinski, R., Golub, G. H., Meurant, G. A., et Périaux, J., editors (1988). *Proceedings of the First International Conference on Domain Decomposition Methods*, Philadelphia. SIAM.

[Glowinski et al., 1991] Glowinski, R., Kuznetsov, Y. A., Meurant, G. A., Périaux, J., et Widlund, O. B., editors (1991). *Proceedings of the Forth International Conference on Domain Decomposition Methods*, Philadelphia. SIAM.

[Glowinski et Le Tallec, 1989] Glowinski, R. et Le Tallec, P. (1989). *Augmented Lagrangian and Operator Splitting in Nonlinear Mechanics*. SIAM.

[Glowinski et Le Tallec, 1990] Glowinski, R. et Le Tallec, P. (1990). Augmented Lagrangian interpretation of the nonoverlapping Schwarz alternating method. In Chan, T. F., Glowinski, R., Périaux, J., et Widlund, O. B., editors, *Proc. 3rd Conference on Domain Decomposition Methods.*, pages 224–231.

[Glowinski et al., 1976] Glowinski, R., Lions, J.-L., et Trémolières, R. (1976). *Analyse numérique des inéquations variationnelles*. Dunod-Bordas, Paris.

[Glowinski et al., 1996] Glowinski, R., Périaux, J., Shi, Z.-C., et Widlund, O. B., editors (1996). *Proceedings of the Eighth International Conference on Domain Decomposition Methods*, Chichester. Wiley and Sons.

[Gustafon, 1988] Gustafon, J.-L. (1988). Reevaluating amdahl's law. In *Communications of the ACM*, volume 31, pages 532–533.

[Jäger et al., 1992] Jäger, J., Hubecker, F. K., et Kuznetsov, Y. A. (1992). An overlapping domain decomposition superposition method for a model heat equation. Technical report, IBM Germany, Heidelberg scientific center.

[Keyes et al., 1992] Keyes, D. E., Chan, T. F., Meurant, G. A., Scroggs, J. S., et Voigt, R. G., editors (1992). *Proceedings of the Fifth International Conference on Domain Decomposition Methods*, Philadelphia. SIAM.

[Keyes et al., 1995] Keyes, D. E., Saad, Y., et Truhlar, D. G., editors (1995). *Domain-Based Parallelism and Problem Decomposition Methods in Computational Sciences and Engineering*. SIAM.

[Keyes et Xu, 1995] Keyes, D. E. et Xu, J., editors (1995). *Proceedings of the Seventh International Conference on Domain Decomposition Methods*, number 180 in Contemporary Mathematics, Providence. AMS.

[Kortas et Angot, 1996] Kortas, S. et Angot, P. (1996). A practical and portable model of programming for iterative solvers on distributed memory machines. *Parallel computing*, 4(22):487–512.

[Kunisch et Tai, 1997] Kunisch, K. et Tai, X.-C. (1997). Nonoverlapping domain decomposition methods for inverse problems. In Bjørstad, P. E., Espedal, M., et Keyes, D., editors, *Domain Decomposition Methods in Sciences and Engineering*. John Wiley & Sons. Proceedings from the Ninth International Conference, June 1996, Bergen, Norway.

[Lagnese et Leugering, 1998] Lagnese, J. et Leugering, G. (1998). Dynamic domain decomposition in approximate and exact boundary control in transmission problems. Preprint.

[Le Tallec, 1994] Le Tallec, P. (1994). Domain decomposition in computational mechanics. *Computational Mechanics Advances*, 1. North-Holland, Amsterdam.

[Le Tallec et Sassi, 1995] Le Tallec, P. et Sassi, T. (1995). Domain decomposition with nonmatching grids: Augmented lagrangian approach. *Mathematics of Computation*, 64(212):1367–1396.

[Leugering, 1997a] Leugering, G. (1997a). Domain decomposition of optimal control problems for dynamic networks of elastic strings. In Hager, W. W. et Pardalos, P. M., editors, *Optimal Control: Theory, Algorithms and Applications*. Kluwer Academic Publishers B. V., Holland.

[Leugering, 1997b] Leugering, G. (1997b). Dynamic domain decomposition of optimal control problems for networks of euler-bernouilli beams. Preprint.

[Lions, 1968] Lions, J.-L. (1968). *Contrôle optimal de systèmes gouvernés par des équations aux dérivées partielles*. Dunod, Paris.

[Lions, 1998a] Lions, J.-L. (1998a). Parallel stabilization. In *Chinese Annals of Mathematics*.

[Lions, 1998b] Lions, J.-L. (1998b). Parallel stabilization of hyperbolic and petrovsky systems. In *Fourth world congress on Computational Mechanics*.

[Lions, 1998c] Lions, J.-L. (1998c). Parallel stabilization of navier stokes equations. In *10th International Conference on Parallel CFD*, Hsinchu, Taiwan.

[Lions et Magenes, 1968] Lions, J.-L. et Magenes, E. (1968). *Problèmes aux limites non homogènes*. Dunod.

[Lions et Pironneau, 1998a] Lions, J.-L. et Pironneau, O. (1998a). Algorithmes parallèles pour la solution de problèmes aux limites. In *C. R. Acad. Sci. Paris*.

[Lions et Pironneau, 1998b] Lions, J.-L. et Pironneau, O. (1998b). Sur le contrôle parallèle des systèmes distribués. In *C. R. Acad. Sci. Paris*.

[Lions, 1988] Lions, P.-L. (1988). On the Schwarz alternating method I:
A variant for nonoverlapping subdomains. In Glowinski, R., Golub, G. H., Meurant, G. A., et Périaux, J., editors, *Proc. 1rst Conference on Domain Decomposition Methods*, pages 1–42, Philadelphia. SIAM.

[Lions, 1989] Lions, P.-L. (1989). On the Schwarz alternating method II:
A variant for nonoverlapping subdomains. In Chan, T. F., Glowinski, R., Périaux, J., et Widlund, O. B., editors, *Proc. of Second International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia. SIAM.

[Lions, 1990] Lions, P.-L. (1990). On the Schwarz alternating method III:
A variant for nonoverlapping subdomains. In Chan, T. F., Glowinski, R., Périaux, J., et Widlund, O. B., editors, *Proc. 3rd Conference on Domain Decomposition Methods*, pages 202–223, Philadelphia. SIAM.

[Luong, 1995] Luong, B. (1995). *Techniques de contrôle optimal pour un modèle quasi-géostrophique de circulation océanique. Application à l'assimilation de données altimétriques satellitaires*. PhD thesis, Université Joseph Fourier.

[Quarteroni et al., 1994] Quarteroni, A., Périaux, J., Kuznetsov, Y. A., et Widlund, O. B., editors (1994). *Proceedings of the Sixth International Conference on Domain Decomposition Methods*, number 157 in Contemporary Mathematics, Providence. AMS.

[Ralph, 1996] Ralph, D. (1996). A parallel method for unconstrained discrete-time optimal control problems. *Journal on Optimization*, 6:488–512.

[Raviart et Thomas, 1988] Raviart, P. A. et Thomas, J. M. (1988). Introduction à l'analyse numérique des équations aux dérivées partielles. Masson.

[Roux, 1989] Roux, F.-X. (1989). *Domain Decomposition Methods with Coupled Transmission conditions for the Control of Systems governed by Elliptic Partial Differential Equations*. PhD thesis, ONERA.

[Saad, 1985] Saad, Y. (1985). Practical use of some krylov subspace methods for solving indefinite and nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 6:865–881.

[Saad et Schultz, 1986] Saad, Y. et Schultz, M.-H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869.

[Sassi, 1993] Sassi, T. (1993). *Méthodes de décomposition de domaine pour la résolution de problème d'élasticité non-linéaire avec maillages incompatibles*. PhD thesis, Université Paris IX.

[Schwarz, 1890] Schwarz, H. A. (1890). *Gesammelte Mathematische Abhandlungen*, volume 2. Springer, Berlin. First published in Vierteljahrsschrift der Naturforsch. Gesellschaft in Zürich, volume 15, 1870, pp. 272-286, 1870.

[Sleijpen et Fokkema, 1993] Sleijpen, G. L. G. et Fokkema, D. R. (1993). Bicgstab(1) forb linear equations involving unsymmetric matrices with complex pectrum. *Electronic Transaction on Numerical Analysis*, 1:11–31.

[Smith et al., 1996] Smith, B., Bjørstad, P., et Gropp, W. (1996). *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press.

[Sobolev, 1936] Sobolev, S. L. (1936). L'algorithme de Schwarz dans la théorie de l'élasticité. In *Comptes rendus de l'académie des sciences de l'URSS*, volume IV((XIII)6), pages 243–246.

[Sonnelved, 1989] Sonnelved, P. (1989). CGS: a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52.

[Tai, 1995a] Tai, X.-C. (1995a). Parallel function decomposition and space decomposition methods 1. function decomposition. *Beijing Mathematics*, 1:104–134.

[Tai, 1995b] Tai, X.-C. (1995b). Parallel function decomposition and space decomposition methods 2. space decomposition. *Beijing Mathematics*, 1:135–152.

[Tai, 1998] Tai, X.-C. (1998). A space decomposition method for parabolic equations. *Num. Meth. for Part. Diff. Equat.*, 14(1).

[Van der Vorst, 1992] Van der Vorst, H. (1992). Bi-CGSTAB: a fast smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644.

[Van der Vorst et Vuik, 1994] Van der Vorst, H. et Vuik, C. (1994). GMRESR: a family of nested GMRES methods. *Num. Alg. with Appl.*, 1:369–386.