Développement du parallélisme des méthodes numériques adaptatives pour un code industriel de simulation en mécanique des fluides

Eli Laucoin

UNIVERSITÉ JOSEPH FOURIER – GRENOBLE 1

Thèse présentée pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER

Spécialité : « **Mathématiques appliquées** »

préparée au Laboratoire de Modélisation et Développements Logiciels (CEA Grenoble)
en collaboration avec le Laboratoire Jean KUNZTMANN (Université Joseph FOURIER)

dans le cadre de l'École Doctorale :
« Mathématiques, Sciences et Technologies de l'Information, Informatique »

par

M. Éli LAUCOIN

Développement du parallélisme des méthodes numériques adaptatives pour un code industriel de simulation en mécanique des fluides

---

Directrice de thèse : M^lle^ Isabelle CHARPENTIER

Thèse soutenue le 24 octobre 2008 devant la commission d'examen :

M. Christophe PRUD'HOMME	Université Joseph FOURIER – Grenoble I	*Président du jury*
M^lle^ Isabelle CHARPENTIER	Université Joseph FOURIER – Grenoble I	*Directrice de thèse*
M. Frédéric HECHT	Université Pierre et Marie CURIE – Paris VI	*Rapporteur*
M. Zakaria BELHACHMI	Université de Metz	*Rapporteur*
M. Philippe ÉMONOT	CEA	*Examinateur*
M. Christophe CALVIN	CEA	*Encadrant*

Remerciements

Je tiens tout d'abord à remercier Christophe CALVIN pour m'avoir proposé le sujet de ce travail de thèse, et Isabelle CHARPENTIER pour en avoir accepté la direction. Je remercie également Frédéric HECHT et Zakaria BELHACHMI pour l'honneur qu'ils m'ont fait en acceptant de rapporter sur ce manuscript. Enfin, je remercie également Philippe ÉMONOT pour sa participation au jury de cette thèse, et Christophe PRUD'HOMME pour en avoir accepté la présidence.

Je tiens ensuite à remercier Frédéric DUCROS, chef du Laboratoire de Modélisation et de Développements Logiciels, pour m'avoir accueilli au sein de son unité. Mes remerciements s'adressent aussi à Philippe MONTARNAL, chef du Laboratoire de Modélisation des Transferts dans les Milieux Solides, pour m'avoir proposé de le rejoindre, et pour m'avoir permis de mener à terme ce travail de thèse au cours des deux dernières années.

Je remercie également tous les membres de l'équipe du LMDL, et tout particulièrement Philippe ÉMONOT, Gauthier FAUCHET, Benoît MATHIEU et Fabien PERDU, pour leur patience, leur écoute attentive, leur aide précieuse et leurs judicieux conseils, tant en matière d'analyse numérique qu'en ce qui concerne le développement au sein de TrioU. Merci aussi à toute la joyeuse bande des thésards du LMDL, Thomas, Marion (ne t'inquiète pas pour le SAV Ubuntu post-thèse), Benoît, Adrien, Guillaume, Younès, Alex, et tous les autres, pour leur bonne humeur, leurs sourires, et la tonne de bons moments passés ensemble. J'adresse également une pensée particulière à Aparicio, dont l'humour et les bons mots nous manqueront.

Enfin, je tiens à remercier toute ma famille, en commençant par mes parents qui m'ont toujours soutenu et qui m'ont tant apporté, ainsi que mon frère et ma sœur, toujours présents malgé les distances.

Je ne saurai clore ce paragraphe un peu trop formel à mon goût sans remercier celle qui a su me soutenir, m'encourager, me supporter, et finalement me donner la motivation d'achever ce travail. Merci à toi, ma chère Fanny, pour avoir toujours été là, pour avoir toujours cru en moi.

Table des matières

Liste des illustrations

Introduction

Présentation des travaux

Avec l'avènement de l'ère informatique, les techniques de simulation numérique deviennent des outils très répandus dans le monde industriel. Ces méthodes permettent en effet de modéliser des phénomènes toujours plus complexes et d'en étudier les caractéristiques en minimisant le recours à l'expérience. À ce titre, les efforts engagés dans le développement de méthodes numériques fiables et efficaces, capables de réaliser des simulations à grande échelle sont grandissants.

Cependant, en raison de ce contexte d'expansion, les méthodes numériques utilisées aujourd'hui dans l'industrie conduisent le plus souvent à des simulations extrêmement coûteuses. Ce coût se mesure à l'aune de l'espace mémoire et du temps de calcul nécessaires à la réalisation d'un calcul. En effet, il n'est pas rare aujourd'hui de réaliser des simulations numériques présentant plusieurs dizaines voire centaines de millions d'inconnues, pour des calculs pouvant durer entre quelques heures et plusieurs semaines. Pour ce faire, le recours au calcul massivement parallèle est devenu une nécessité. Cependant, la conception et l'implémentation d'un outil de simulation permettant d'exploiter la puissance d'une machine parallèle s'avère une tâche très difficile. En effet, la perspective du parallélisme massif et celle des gros volumes de données soulèvent des questions inédites dans le monde industriel.

Bien que le calcul parallèle permette de lever la barrière technologique imposée par les capacités des machines conventionnelles, il reste toujours pertinent de chercher à minimiser le coût des simulations numériques. Les méthodes dites « adaptatives » constituent précisément une famille de techniques permettant d'optimiser ce coût tout en assurant la qualité des solutions calculées. Le principe de ces méthodes consiste à modifier dynamiquement la discrétisation du problème de façon à accentuer l'effort de précision sur les régions où l'écart entre la solution calculée et la solution exacte du problème a été estimé comme étant important. De cette façon, on cherche à mettre en place une simulation présentant le nombre minimal de degrés de liberté qui permette de calculer une bonne solution approchée.

En dépit de leur efficacité, les méthodes adaptatives restent cependant assez peu utilisées dans le monde industriel. Ce fait est essentiellement lié à la complexité de leur mise en œuvre, en particulier dans un logiciel développé pour être exécuté dans un contexte parallèle. En effet, le caractère dynamique de la discrétisation est très lourd de conséquences en termes de génie logiciel. Par exemple, modifier les maillages impose de ré–assembler les problèmes discrets après chaque phase d'adaptation. De la même façon, dans le cas de l'étude d'un problème instationnaire, ces modifications peuvent nuire à

la stabilité des schémas numériques qui contrôlent les itérations en temps. Et dans un contexte d'exécution parallèle, l'utilisation d'une méthode adaptative pose nécessairement la question difficile de l'équilibrage de la charge de calcul entre les processeurs.

Dans ce travail de thèse, nous nous intéressons à la problématique encore ouverte aujourd'hui de la mise en œuvre de méthodes numériques adaptatives dans un contexte d'exécution parallèle. Ce travail s'inscrit dans une phase de développement amont de la plate–forme logicielle Trio-U, développée depuis 1993 au Laboratoire de Modélisation et de Développements Logiciels (LMDL), à Grenoble. Ce laboratoire appartient au Service de Simulation en Thermohydraulique (SSTH) du Commissariat à l'Énergie Atomique (CEA), et, à ce titre, s'intéresse principalement à la simulation numérique des écoulements complexes qui se développent au sein des réacteurs nucléaires civils.

Nos travaux ont donc pour principal objectif de concevoir et implémenter une extension de la plate–forme Trio-U permettant la simulation de problèmes tridimensionnels complexes issus du domaine de la mécanique des fluides, en utilisant une démarche adaptative déséquilibrante dans un contexte d'exécution parallèle. Ce travail s'articule autour de contributions :

- dans le domaine de la géométrie, où nous avons réalisé une analyse critique approfondie de différentes méthodes d'adaptation de maillage, afin de déterminer la démarche la plus adaptée à la plate–forme Trio-U ;
- en analyse numérique, où nous avons établi un formalisme reposant sur la méthode des éléments joints, adapté à l'élément fini de Crouzeix–Raviart, qui constitue la base de la méthode de Volumes–Éléments finis de Trio-U ;
- en algorithmique, où nous avons développé des algorithmes parallèles permettant la mise en œuvre notre méthode adaptative ;
- en génie logiciel, où nous avons conçu une architecture à la fois souple et extensible, pour implémenter notre méthode au sein de la plate–forme Trio-U.

Organisation du mémoire

Ce manuscrit est structuré de la façon suivante. Dans le premier chapitre, nous présentons en détail la problématique étudiée. Nous insistons sur l'intérêt que présentent les méthodes adaptatives pour une plate–forme telle que Trio-U, nous présentons une analyse critique des différentes approches rencontrées dans la littérature, et nous soulevons les différentes questions liées à ce sujet.

Dans le deuxième chapitre, nous rappelons les concepts fondamentaux des méthodes de décomposition de domaine, formalisme mathématique qui servira de socle à notre méthode numérique. Nous détaillons également les différents modèles et techniques utilisées aujourd'hui pour appréhender le calcul massivement parallèle.

Le troisième chapitre est dédié à une analyse critique de différentes méthodes d'adaptation de maillage. Nous étudions dans le détail les caractéristiques de ces méthodes, tant du point de vue de la qualité des maillages obtenus que de celui de l'efficacité algorithmique, en particulier dans un contexte d'exécution parallèle. L'objectif de cette étude est de justifier le choix de l'une ou l'autre de ces techniques en regard des contraintes imposées par l'implémentation dans la plate–forme Trio-U.

Dans le quatrième chapitre, nous présentons les travaux entrepris pour développer un formalisme d'éléments joints adapté à la méthode VEF de Trio-U. Ce formalisme

nous permettra, comme cela est suggéré dans les articles de C. BERNARDI, Y. MADAY et F. HECHT [21, 22], de prendre en compte convenablement les non–conformités générées par notre algorithme d'adaptation de maillage. Nous présentons également les concepts nécessaires à l'interpolation des champs sur les maillages adaptés, ainsi que ceux liés à l'estimation de l'erreur numérique.

Dans le cinquième chapitre, nous abordons les questions relatives à l'implémentation parallèle de notre méthode. En effet, cette dernière présente, à l'instar de toute approche adaptative, un caractère déséquilibrant, qu'il est nécessaire de prendre en compte pour assurer son efficacité. Nous présentons donc les idées que nous avons retenues pour résoudre au moins partiellement cette difficulté, notamment par l'introduction d'heuristiques de partitionnement de graphe.

Le sixième chapitre est consacré à une présentation détaillée de l'architecture logicielle retenue pour l'implémentation de notre méthode parallèle au sein de la plate–forme Trio-U.

Enfin, le septième chapitre présente les résultats obtenus. Nous distinguerons essentiellement deux types de résultats :

- les résultats numériques, qui illustrent les capacités qualitatives de notre méthode pour résoudre un certain nombre de problèmes numériques ;
- les résultats algorithmiques, qui illustrent en termes quantitatifs les performances de notre méthode, concernant entre autres la qualité des maillages générés et le coût du calcul.

À l'attention du lecteur, soulignons que notre démarche dans la rédaction de ces chapitres pourra parfois s'avérer quelque peu déroutante. En effet, nous verrons que la plupart de nos résultats constituent en général une extension de résultats existants, de façon à les adapter à notre contexte. Afin de clarifier la présentation, nous adopterons généralement la démarche suivante :

- description détaillée du résultat pré–existant ;
- analyse critique de ce résultat au regard de notre contexte ;
- présentation et justification d'une extension ou d'une réinterprétation appropriée.

Toujours dans l'optique de rendre plus aisée la lecture de ce mémoire, nous introduisons chaque chapitre par un paragraphe qui en rappelle la problématique et la replace dans le cadre de la globalité de notre travail. De même, chaque chapitre se termine par un résumé succinct des principaux résultats qui y sont utilisés ou établis.

CHAPITRE 1

Implémentation parallèle d'une méthode adaptative

Ce chapitre a pour vocation d'introduire la problématique de notre travail dans sa plus grande généralité. Nous y exposons l'intérêt que présentent les méthodes numériques adaptatives pour assurer l'efficacité des logiciels de simulation, ainsi que différentes solutions proposées jusqu'ici pour leur mise en œuvre. Nous analysons les avantages et les inconvénients de ces approches, avant de présenter les idées fondatrices de la méthode que nous proposons.

1.1 Enjeux et Motivations

1.1.1 La démarche générale du calcul scientifique

Une simulation numérique repose sur une modélisation du système physique étudié, généralement représentée de façon concrète par un système d'équations aux dérivées partielles dont la solution caractérise le phénomène considéré. L'équation de la chaleur :

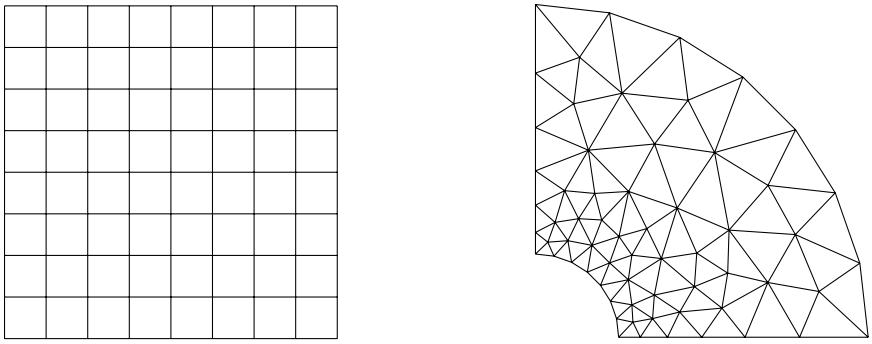
$$\partial_t T - \nabla \cdot (a(x) \nabla T) = f(t,x)$$

permet par exemple de décrire l'évolution de la température dans un domaine Ω donné. Dans cette équation, T désigne le champ de température, $a(x)$ est un champ scalaire décrivant les propriétés de diffusivité du milieu, et $f(t,x)$ correspond à une source volumique de chaleur. Assortie de conditions aux limites adéquates (température imposée ou flux de chaleur imposé sur le bord du domaine) et de conditions initiales pertinentes (c.-à-d. la valeur de la température au temps initial), et moyennant certaines hypothèses de régularité sur a et f, cette équation admet une unique solution $T^\star$, qui décrit complètement le phénomène étudié.

Malheureusement, les systèmes d'équations que l'on obtient lors de la modélisation des phénomènes physiques sont souvent très complexes, et il est généralement difficile, voire impossible, d'envisager de les résoudre analytiquement. L'idée du calcul scientifique est de construire un problème qui soit « proche » de celui décrivant le phénomène, mais qui soit également beaucoup plus simple à résoudre. Cette notion d'*approximation* est fondamentale, car elle permet d'assurer la pertinence de la démarche, et de mesurer la qualité de la solution numérique. Il existe en effet de nombreuses méthodes permettant

de construire un problème approché, qui se distinguent essentiellement par la notion d'approximation à laquelle chacune se réfère. Nous nous concentrons ici sur la méthode des éléments finis, dont le cadre est aujourd'hui parfaitement établi [40, 48, 54, 92, 113].

Une des principales difficultés de la résolution analytique des systèmes d'équations aux dérivées partielles vient du fait que la solution recherchée appartient à un espace de dimension infinie. C'est pourquoi toute méthode numérique introduit une notion de *discrétisation*, qui a pour but de chercher une solution approchée dans un espace de dimension finie. De cette façon, la résolution du problème discret consiste en l'inversion d'un système matriciel, et l'on peut alors réutiliser tous les théorèmes et algorithmes de l'algèbre linéaire. En pratique, cette notion de discrétisation repose le plus souvent sur l'introduction d'un maillage du domaine de calcul. Bien entendu, ce maillage doit correspondre aux besoins de la méthode numérique envisagée. Par exemple, la méthode des différences finies nécessite que le maillage soit une grille structurée, alors que la méthode des éléments finis permet d'utiliser des maillages plus généraux (voir la figure 1.1).



(a) Grille cartésienne (b) Maillage non–structuré

FIG. 1.1 : Exemples de maillages classiquement utilisés

De façon générale, la *fiabilité* et l'*efficacité* [53] sont les deux préoccupations essentielles du numéricien. Une méthode numérique est considérée comme fiable dès l'instant que l'erreur numérique (c.-à-d. l'écart entre la solution approchée et la solution exacte) peut être contrôlée. Cette erreur est inévitable de part la nature même du calcul scientifique. Elle peut provenir de différentes sources : erreurs de modélisation, erreur de troncature, erreur d'approximation, erreur de consistance, etc. Pour assurer la qualité des solutions approchées, une méthode numérique doit donc fournir les moyens de calculer, ou au moins d'estimer, ces erreurs, et d'assurer que leur contribution reste faible au cours du calcul.

La notion d'efficacité se rapporte au coût de la simulation, à savoir la durée du calcul et la place mémoire nécessaires à l'obtention de la solution numérique. Bien évidemment, le numéricien cherche à minimiser ce coût, car le temps de calcul est une ressource onéreuse, et bien que la quantité de mémoire disponible sur les machines actuelles soit gigantesque par rapport à celle qui équipait les premiers ordinateurs, elle n'en reste pas moins une ressource limitée. Les caractéristiques des machines de calcul déterminent par conséquent la faisabilité d'un calcul, et le critère d'efficacité permet de s'assurer de cette dernière. Bien entendu, fiabilité et efficacité sont des objectifs par essence contradictoires, car assurer une bonne précision impose généralement une augmentation du

coût du calcul, soit en introduisant de nouveaux degrés de liberté, soit en rallongeant la durée du calcul.

1.1.2 Les enjeux actuels

Aujourd'hui, nous pouvons observer entre autres deux grandes tendances dans l'évolution de la simulation numérique. D'une part, on constate que les champs d'application du calcul scientifique se diversifient énormément. Dans le monde industriel, ce dernier devient un outil incontournable d'aide à la décision, en permettant de modéliser des phénomènes toujours plus variés. À titre d'exemple, le recours à la simulation numérique est aujourd'hui devenu classique dans des domaines aussi divers que la météorologie, l'industrie nucléaire, les sciences des matériaux, la chimie, la finance, etc. Une conséquence de ce constat est que les phénomènes étudiés se complexifient d'autant, et avec eux les modèles mathématiques qui les décrivent. D'autre part, ces phénomènes font intervenir des échelles physiques différentes. Par exemple, on souhaite pouvoir calculer l'écoulement du fluide caloporteur dans un cœur de réacteur nucléaire complet en prenant en compte des phénomènes se produisant à des échelles microscopiques (tourbillons, ébullition, couches limites, etc).

En conséquence, l'utilisation du calcul scientifique pour la résolution de problèmes industriels conduit à des besoins toujours croissants en termes de ressources informatiques. Afin de prendre en compte tant la complexité des problèmes que la diversité des échelles mises en jeu, il est nécessaire, si l'on se limite à l'utilisation de méthodes classiques et éprouvées, de discrétiser les domaines de calculs de façon très fine. De telles discrétisations conduisent à l'élaboration de systèmes linéaires gigantesques, dont le stockage et la résolution nécessitent des ressources très importantes. Ce phénomène est d'ailleurs amplifié par la complexité croissante des géométries de calculs considérées. Ainsi, qu'il s'agisse de temps de calcul ou d'espace mémoire, le coût des simulations numériques tend à croître inexorablement.

Par ailleurs, les phénomènes physiques qui se produisent à des échelles différentes sont généralement modélisés par des problèmes de nature différente. Or chaque méthode numérique présente en quelque sorte un « type de problème de prédilection ». Ainsi, les méthodes de type volumes finis sont parfaitement adaptées à la résolution d'équations correspondant à des lois de conservation, alors que les méthodes de type éléments finis s'avèrent extrêmement efficaces lors de l'étude de problèmes de nature elliptique. Il semble donc naturel que les modèles mathématiques, ainsi que les méthodes numériques, puissent différer d'une région à l'autre du domaine de calcul, selon la nature du phénomène physique prépondérant dans cette région. Considérons par exemple un écoulement diphasique d'un liquide contenant des inclusions de vapeur. La physique décrivant l'interface liquide–vapeur est très complexe, mais elle peut être simulée numériquement à l'aide de modèles de type interfaces diffuses. Il semble cependant parfaitement inutile de devoir prendre en compte la physique du changement de phase au sein même d'une phase : ce dernier a lieu uniquement à l'interface entre les deux fluides. Il paraît donc naturel de vouloir recourir à un modèle monophasique à l'intérieur des phases, et à un modèle diphasique au voisinage des interfaces entre fluides. Cette idée de spécialisation du modèle et de la méthode en fonction des phénomènes étudiés introduit un degré de complexité supplémentaire dans la résolution des problèmes concrets rencontrés dans le domaine industriel.

1.1.3 L'approche usuelle : pertinence et inconvénients

La démarche classique consiste à s'assurer en premier lieu de la précision du calcul. Ceci paraît somme toute assez naturel, car, rappelons–le, nous cherchons avant tout à déterminer une bonne approximation de la solution du problème étudié. Pour ce faire, le numéricien effectue une analyse *a priori* du problème à résoudre, en étudiant les propriétés globales de régularité de la solution que l'on peut déduire des données du problème. Une fois ces propriétés établies, on peut utiliser les propriétés d'approximation de la méthode numérique pour déterminer un pas de maillage maximal permettant d'atteindre un degré de précision donné. Une fois que l'on connait ce paramètre, la construction d'un maillage du domaine de calcul qui respecte ce critère assure la qualité de l'approximation de la solution numérique.

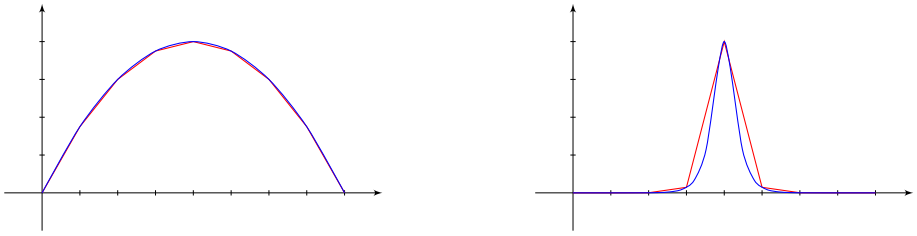
Cette approche très naturelle présente de nombreux intérêts. Tout d'abord, son principe reste très proche des idées fondatrices du calcul scientifique : le travail théorique réalisé avant le lancement du calcul permet d'assurer que les solutions résultant de ce dernier seront pertinentes. Ayant établi à l'avance les propriétés de régularité de la solution recherchée, on assure, avant même d'avoir lancé le calcul, qu'elle sera une bonne approximation de la solution exacte du problème étudié. Ainsi, cette approche permet de satisfaire pleinement le critère de fiabilité, et ce de façon tout–à–fait naturelle. Ensuite, elle permet d'utiliser des méthodes numériques simples, faciles à implémenter, et surtout bien éprouvées. Cet argument est particulièrement important dans un contexte industriel, car la robustesse des techniques numériques constitue un point crucial pour assurer la stabilité et la pérennité des logiciels développés. Soulignons enfin que cette approche reste relativement simple à mettre en œuvre. En effet, la majorité des outils de simulation numérique reposent essentiellement sur trois ingrédients :

– un logiciel de maillage ;
– un logiciel de discrétisation ;
– un logiciel de résolution.

Le premier a pour objectif de fournir un maillage du domaine de calcul. Le deuxième permet de construire, à partir du maillage, un système linéaire correspondant à la discrétisation du problème étudié. Le dernier fournit les algorithmes nécessaires à la résolution de ce système pour obtenir la solution numérique. Comme on peut le voir, la seule contrainte imposée par la démarche présentée ci-dessus consiste à pouvoir choisir un pas de maillage maximal lors de l'appel au logiciel de maillage, ce qui ne présente pas de réelle difficulté.

Cependant, cette démarche présente également de sérieux inconvénients. En premier lieu, lorsque l'on étudie des problèmes aussi complexes que ceux qui apparaissent dans un contexte industriel, il n'est pas rare de constater que l'analyse *a priori* atteint ses limites. En effet, établir les propriétés de la solution recherchée peut s'avérer très difficile, en particulier quand les problèmes rencontrés sont raides, instationnaires, ou non–linéaires. Dans ce contexte, il se peut que cette démarche ne soit pas applicable. Une autre difficulté apparaît lorsque l'on étudie la faisabilité de la mise en œuvre de cette approche. En effet, étant donnés les enjeux actuels du calcul scientifique, cette démarche conduit généralement à des simulations extrêmement coûteuses, qu'il n'est pas possible de réaliser à l'aide d'un ordinateur conventionnel. Pour remédier à ce problème tout en conservant cette approche, l'idée la plus communément retenue consiste à développer des codes de calcul pouvant exploiter la puissance d'une machine parallèle. En répar-

tissant le volume de données et la charge de travail sur plusieurs unités de calcul, on lève en quelque sorte la « barrière technologique » imposée par les limites des capacités des machines traditionnelles. Soulignons cependant que cette technique possède elle-même ses propres limitations, et qu'il est possible d'envisager une situation où la dimension du problème à résoudre soit telle que le parallélisme ne soit d'aucun secours. Enfin, se posent des questions relatives à l'efficacité de cette démarche. Premièrement, fixer le pas du maillage revient à en fixer le nombre d'éléments, et donc le nombre de degrés de liberté du problème approché. Or, l'efficacité du calcul dépend essentiellement de la complexité algorithmique des solveurs utilisés, qui dépend elle–même directement du nombre d'inconnues à calculer. Ce dernier étant fixé, l'efficacité de la méthode ne peut être améliorée qu'en travaillant sur les algorithmes de résolution, en cherchant à minimiser leur complexité, ce qui limite sérieusement les pistes de recherche. Deuxièmement, et c'est en réalité là la grande faiblesse de cette approche, *rien n'indique qu'il soit nécessaire d'utiliser une discrétisation fine sur l'intégralité du domaine de calcul.* D'ailleurs, l'intuition indiquerait même exactement le contraire. Comme cela est illustré sur la figure 1.2, plus une fonction complexe est régulière, plus il est facile de l'approcher convenablement à l'aide de fonctions simples (p. ex. affines par morceaux). Ce résultat qualitatif est



(a) Cas d'une fonction régulière (b) Cas d'une fonction singulière

FIG. 1.2 : Approximation de fonctions

bien entendu soutenu par des arguments théoriques qui relèvent de la théorie de l'approximation. Nous renvoyons à [54] pour plus de détails. Par définition, les singularités d'une fonction sont localisées dans l'espace. On conçoit donc aisément que l'effort de précision devrait être concentré sur ces singularités, et que la finesse de la discrétisation devrait être adaptée à la régularité locale de la solution. L'analyse *a priori* ne prend pas en compte la localisation des singularités, car elle ne fournit que des propriétés globales de la solution. Ainsi, cette approche conduit à la construction d'un maillage uniforme, suffisamment fin pour atteindre la précision souhaitée dans les régions singulières, mais raffiné de façon inutile dans les régions régulières. Elle introduit par conséquent un surcoût à la résolution en augmentant la précision dans des régions où cela n'est pas nécessaire. En ce sens, l'approche usuelle est fiable, car la précision souhaitée est obtenue, mais elle est très loin d'être efficace, car l'effort de précision n'est pas focalisé sur les régions singulières.

1.1.4 De l'intérêt des méthodes adaptatives

Nous nous proposons ici d'utiliser une autre approche, en cherchant à préserver le critère d'efficacité sans toutefois renoncer à la recherche de la précision. Pour ce faire, on procède exactement comme cela a été suggéré dans le paragraphe précédent, ainsi que

dans [53]. L'idée est de chercher à raffiner sélectivement la discrétisation dans les régions où cela est nécessaire pour obtenir la précision souhaitée. Dans les régions où la solution est régulière, on cherche à diminuer le coût de la simulation en minimisant le nombre de degrés de liberté nécessaires à la satisfaction du critère de fiabilité. Une telle approche est dite *adaptative*, car elle ajuste la discrétisation « à la volée », en fonction des propriétés de la solution qu'elle est en train de calculer. La figure 1.3 en présente la démarche générale. Il s'agit d'introduire une rétroaction entre le processus de résolution et celui de

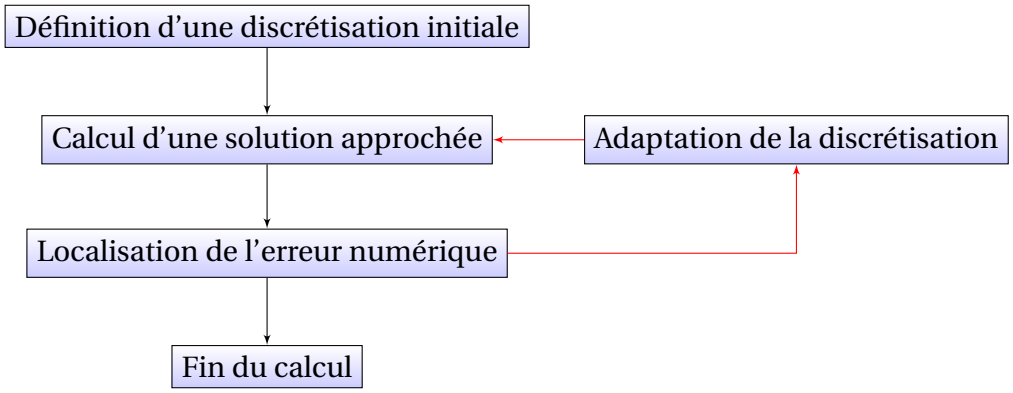


FIG. 1.3 : Schéma classique du déroulement d'un calcul adaptatif

discrétisation (en rouge sur le schéma). Cet algorithme itératif permet d'optimiser l'utilisation des ressources, en n'introduisant une discrétisation fine que dans les régions où cela est réellement nécessaire. Bien évidemment, la mise en œuvre de cette approche est plus complexe que celle de l'approche classique, mais elle permet d'assurer la fiabilité et l'efficacité du calcul de façon beaucoup plus satisfaisante.

Pour mettre en application les idées que nous venons de présenter, nous devons faire appel à plusieurs ingrédients :

- un « détecteur de singularités », qui nous permet de choisir les zones où la discrétisation doit être raffinée ;
- un processus de modification de la discrétisation, pour raffiner les zones singulières et déraffiner les régions moins critiques ;
- un critère d'arrêt portant sur le degré de précision souhaité.

Nous verrons au paragraphe 4.4 que le « détecteur de singularités » peut être construit sur la base des estimations d'erreur *a posteriori*, introduites par I. BABUŠKA et W. RHEINBOLT en 1978 [5, 6]. En effet, ces estimations peuvent être localisées, ce qui permet de détecter les zones sur lesquelles il faut focaliser l'adaptation du maillage. Notre deuxième ingrédient, le processus de modification de la discrétisation, peut être appréhendé de plusieurs façons. Nous verrons aux chapitres 3 et 4 qu'il doit impérativement garantir la non–dégénérescence de la qualité géométrique des éléments du maillage. Diverses approches permettent de satisfaire cette contrainte, selon la complexité des algorithmes géométriques utilisés, et les capacités de la méthode numérique envisagée. Enfin, notre troisième outil, à savoir le critère d'arrêt, peut être spécifié de différentes façons : il peut être de nature mathématique (lié à l'estimation d'erreur ou à la régularité de la solution, par exemple), physique (lié au modèle utilisé), ou algorithmique (limitation du nombre d'étapes de raffinement, limitation du nombre total de degrés de liberté). Un choix judicieux de ce critère permet de garantir que le calcul n'entrera pas dans une boucle infinie.

1.2 Notre point de départ : la plate–forme Trio-U

Avant de formuler plus précisément la problématique de notre travail, nous présentons dans les paragraphes qui suivent le contexte dans lequel il s'inscrit. Nous décrivons donc la plate–forme logicielle Trio-U, qui rentre dans la catégorie des logiciels à caractère industriel reposant sur la démarche usuelle présentée ci–dessus.

1.2.1 Présentation

Trio-U [33, 34] est une plate-forme de simulation numérique en thermohydraulique développée depuis 1993 au Laboratoire de Modélisation et de Développements Logiciels du CEA–Grenoble. Le principal champ d'application des simulations réalisées à l'aide de cet outil est l'industrie nucléaire, et plus particulièrement la simulation des écoulements complexes qui se développent dans le cœur des centrales nucléaires civiles. À cette fin, Trio-U présente un certain nombre d'outils physico–numériques permettant de prendre en compte de façon précise la physique de ces écoulements : modèles de turbulence, suivi d'interface pour la simulation d'écoulements polyphasiques, etc. Étant donné un tel champ d'application, il va sans dire que Trio-U est soumis à un impératif de fiabilité extrêmement exigeant.

Face à cet objectif, Trio-U adopte une position tout–à–fait classique dans le monde industriel. Le principe retenu consiste en effet à recourir à des discrétisations globalement fines, faisant intervenir plusieurs centaines de millions d'éléments. Pour pouvoir supporter de telles situations, Trio-U a été conçu dès l'origine dans le but d'exploiter la puissance d'une machine massivement parallèle à mémoire distribuée. Ainsi, la plate–forme propose un ensemble d'outils permettant de faciliter le développement du code dans un contexte parallèle. L'exemple le plus couramment retenu est celui des *vecteurs distribués*, qui prend en charge, via une interface de haut niveau, les communications inter–processeurs nécessaires aux solveurs parallèles d'algèbre linéaire. On peut également citer un service d'entrées–sorties parallèle, ou un service de gestion des maillages distribués. Nous présentons ces concepts plus en détail dans le chapitre 2.

1.2.2 Les méthodes numériques utilisées

Trio-U propose essentiellement deux méthodes numériques :
- la méthode des Volumes–Différences Finis (VDF) [68] ;
- la méthode des Volumes–Éléments Finis (VEF) [42, 52, 69, 60].

Toutes deux appartiennent à la familles des méthodes de volumes finis, mais elles se distinguent par la notion d'approximation sous–jacente. En effet, la méthode VDF repose sur le formalisme des différences finies, où l'on remplaces les dérivées partielles par des différences finies en utilisant des formules de Taylor, alors que la méthode VEF repose sur le formalisme des éléments finis, à savoir une approximation variationnelle du problème. La conséquence la plus notable de cette différence est que la méthode VDF suppose, à l'instar des différences finies, que le maillage soit une grille cartésienne, alors que la méthode VEF repose sur un maillage non–structuré de triangles en dimension 2 ou de tétraèdres en dimension 3. Nous nous concentrons ici sur la méthode VEF car nous aurons besoin par la suite des résultats de convergence de la méthode des éléments finis.

Les bases de la méthode VEF de Trio-U sont présentées en détail dans les travaux de thèse de P. ÉMONOT [52]. Il s'agit en quelque sorte d'une réinterprétation d'une discrétisation par éléments finis dans un cadre de volumes finis. L'idée est d'associer à chaque degré de liberté un « volume de contrôle », et de réinterpréter les intégrations par parties de la méthode des éléments finis comme l'écriture des lois de conservation sur ces volumes de contrôle. Les intérêts de coupler une approche volumes finis et une approche éléments finis sont multiples :

- l'approche volumes finis est particulièrement bien adaptée aux lois de conservations utilisées en thermohydraulique (conservation de la masse, de la quantité de mouvement et de l'énergie) ;
- l'approche volumes finis garantit le caractère conservatif de la méthode ;
- l'approche éléments finis fournit un cadre mathématique cohérent et parfaitement établi.

En quelque sorte, la méthode VEF de Trio-U permet de rassembler les avantages des deux approches.

1.2.3 Architecture de la plate–forme

Trio-U est développé en C++ selon un modèle orienté objet. L'architecture globale de Trio-U est présentée sur la figure 1.4. Comme on peut le voir, il s'agit d'une structure modulaire, articulée autour d'un noyau numérique abstrait. On trouve dans ce noyau les définitions des classes de base abstraites, ainsi que celles des classes génériques correspondantes. Les modules périphériques rassemblent quant à eux les spécialisations de

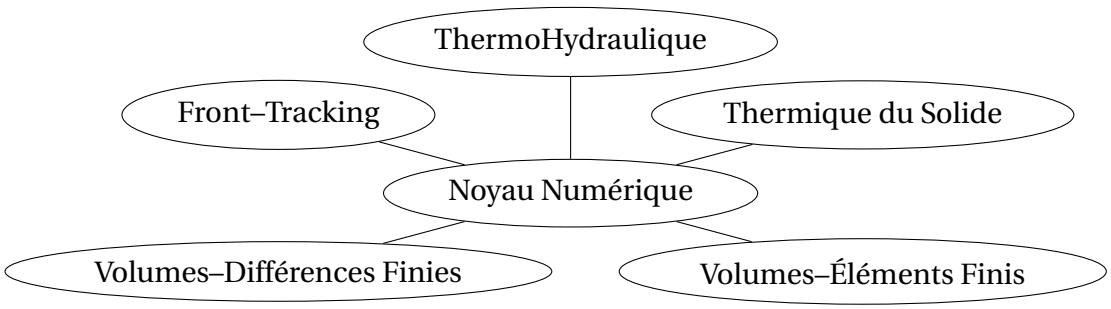


FIG. 1.4 : L'architecture globale de Trio-U

ces classes de base, regroupées selon la nature de la discrétisation (modules Volumes–Éléments Finis et Volumes–Différences Finies), ou la nature du problème étudié (modules Thermique du Solide, Thermohydraulique et Front–Tracking, ce dernier étant dédié à la simulation d'écoulements polyphasiques). L'intérêt de cette structure est qu'elle assure l'extensibilité du logiciel. En effet, le noyau numérique permet de décrire de façon générique n'importe quel problème mathématique, et il suffit de développer un module périphérique pour introduire une nouvelle discrétisation ou un nouveau champ d'application. Cette approche présente toutefois un inconvénient : elle conduit nécessairement au développement d'une hiérarchie de classes assez conséquente, ce qui peut nuire à la lisibilité du code. Nous renvoyons au paragraphe 6.1 pour plus de détails quant aux principes de programmation dans Trio-U.

1.3 Identification du problème

1.3.1 Choix d'une méthode adaptative

Sur les bases du paragraphe 1.1.4, nous considérons qu'une méthode adaptative permet d'assurer tant la fiabilité que l'efficacité d'une simulation numérique. Ainsi, partant des constats établis ci–dessus, notre problématique peut s'énoncer comme suit : « comment intégrer une méthode numérique adaptative et massivement parallèle au sein d'une plate–forme logicielle à vocation industrielle telle que Trio-U ? ». Bien entendu, il existe de nombreuses façons de définir une telle méthode. Une première possibilité consiste à adapter dynamiquement le maillage en raffinant ou en déraffinant les éléments qui le composent. Il s'agit des méthodes dites AMR, pour Adaptive Mesh Refinement. Un autre choix consiste à empiler des niveaux de discrétisation de plus en plus fins. Il s'agit d'approches dites « multi–niveaux », ou encore « multi–grilles ». Il est également possible de modifier le degré des fonctions d'approximations (généralement des polynômes) utilisées lors de la discrétisation, comme cela est suggéré dans [64]. Ces différentes approches conduisent à des techniques d'implémentation différentes présentant chacune leurs avantages et leurs inconvénients. Nous ferons une revue de détail de ces méthodes au chapitre 3, et nous présenterons alors les choix que nous avons retenus.

Trio-U fournit des discrétisations parfaitement adaptées à la simulation numérique en mécanique des fluides. Nous choisissons donc de développer notre méthode adaptative sur la base de ces dernières. Plus exactement, nous choisissons de travailler sur les bases de la méthode VEF. En effet, l'approche éléments finis fournit naturellement les concepts d'estimation d'erreur, qui nous seront extrêmement utiles pour localiser les zones de singularité de la solution. Ce choix aura des conséquences fondamentales sur la spécification de notre méthode adaptative. Par exemple, un résultat classique indique que le conditionnement d'un problème discret construit par une méthode d'éléments finis est directement lié à la qualité géométrique des éléments du maillage. Nous étudions en détail ce point au chapitre 3, mais nous pouvons d'ores et déjà considérer que la géométrie du raffinement de maillage doit vérifier un certain nombre de contraintes afin de ne pas nuire au conditionnement du problème discret.

1.3.2 Trio-U et la notion de maillage dynamique

Dans Trio-U, la géométrie du problème, c'est–à–dire le maillage du domaine de calcul, est considérée comme une donnée statique, qui ne varie pas au cours du calcul. Cette conception est radicalement opposée à celle des techniques d'adaptation de discrétisation. C'est pourquoi nous devons implémenter un module permettant d'intégrer dans Trio-U la notion de maillage dynamique, en définissant les opérations permettant de modifier cette géométrie. Étant donné que nous avons choisi de définir notre méthode adaptative sur la base de la méthode VEF de Trio-U, nous spécifions ces concepts dans le cas de maillages de triangles (en dimension 2) ou de tétraèdres (en dimension 3). Cependant, les idées restent tout–à–fait générales, et transposables à d'autres types de maillages (maillages de quadrangles ou d'hexaèdres par exemple).

Par ailleurs, Trio-U considérant la géométrie de façon statique, cette dernière est largement utilisée pour faciliter l'implémentation parallèle du code. En effet, le principe retenu est d'utiliser un maillage fixe, d'éventuellement le distribuer sur plusieurs pro-

cesseurs, et de lancer le calcul. Si la précision du calcul est insuffisante, on génère un nouveau maillage, et on relance le calcul sur ce maillage. Cette approche est classiquement retenue dans la plupart des codes industriels. Le parallélisme se trouve facilité pour les raisons suivantes :

- l'équilibrage de charge est assuré à l'initialisation du calcul par l'utilisation d'un partitionnement adéquat ;
- la minimisation du volume de données à communiquer est également assurée par le partitionnement du domaine ;
- les schémas de communications inter–processeurs sont statiques, et ils peuvent être déterminés et optimisés à l'initialisation du calcul.

Cette vision du parallélisme est implémentée dans Trio-U au travers de ses vecteurs distribués (voir le paragraphe 2.5.3). Bien que cette approche fournisse une interface de haut niveau pour l'implémentation de solveurs parallèles, elle suppose l'utilisation d'une géométrie statique. Comme nous choisissons de développer une méthode adaptative, il est nécessaire d'introduire une autre perception du parallélisme dans Trio-U.

1.3.3 La question de l'équilibrage de charge

Comme nous venons de le voir, le parallélisme dans Trio-U est perçu essentiellement de façon statique, et il est étroitement lié à la gestion de la géométrie. Ainsi, la performance de l'exécution parallèle du code est assurée par un découpage intelligent du domaine de calcul. On se ramène ainsi à une question de partitionnement de graphe, qui est résolue à l'initialisation du calcul. Actuellement, Trio-U peut réaliser ce partitionnement de plusieurs façons, selon le type de maillage considéré. Si l'on utilise un maillage structuré de quadrangles ou d'hexaèdres (méthode VDF), on peut rapidement partitionner le domaine selon les axes directeurs du maillage. Si on utilise un maillage non–structuré de triangles ou de tétraèdres, Trio-U encapsule l'appel à une librairie externe appelée METIS [80, 81]. Cette librairie fournit plusieurs heuristiques de partitionnement de graphe particulièrement efficaces, qui permettent d'assurer la qualité du découpage du maillage.

Cette approche fonctionne parfaitement tant que la méthode envisagée ne présente pas de caractère adaptatif. Cependant, dès l'instant que l'on introduit une notion de discrétisation dynamique, elle ne peut plus être utilisée. En effet, adapter localement la discrétisation conduit à l'introduction et à la suppression de degrés de liberté dans des régions précises du domaine de calcul. Ainsi, la charge de calcul affectée à un processeur devient variable au cours de la simulation. De même, si la zone concernée comprend des degrés de liberté partagés par plusieurs processeurs, cela introduit une modification des volumes de données à échanger, et donc des schémas de communication à utiliser. Il apparaît donc nécessaire de développer une autre approche du parallélisme dans Trio-U pour pouvoir intégrer une méthode adaptative. Nous développons dans le chapitre 5 les concepts mis en œuvre dans notre méthode en vue d'assurer l'équilibrage de la charge, et, de façon plus générale, l'efficacité de l'implémentation parallèle de notre méthode.

1.4 Solutions pré-existantes

Nous présentons ici différentes approches rencontrées dans la littérature. Elles proposent chacune une réponse (totale ou partielle) à la question posée. L'analyse de ces

solutions permet d'identifier les avantages et les inconvénients de chacune d'entre elles, ainsi que les difficultés que nous rencontrerons par la suite.

1.4.1 Regénération totale du maillage

Cette approche repose sur les concepts et algorithmes pour la génération de maillages exposés dans [64]. Ces techniques permettent de générer des maillages dont les dimensions des mailles sont contrôlées. Ces algorithmes fonctionnent grâce à la spécification d'une « carte de tailles de mailles », qui définit l'anisotropie du maillage en indiquant, pour chaque point du domaine, quelle devrait être la taille de la maille le contenant. Après exécution de cet algorithme, on obtient un maillage qui respecte au mieux ces spécifications. Elle a été mise en œuvre au sein du projet GAMMA de l'INRIA[1] [1].

Le principe de la méthode adaptative associée est alors tout–à–fait naturel : considérant un maillage initial, on calcule une première solution. L'estimation de la qualité de cette solution fournit l'information nécéssaire à la génération d'une nouvelle carte de tailles de mailles. On peut alors générer un nouveau maillage sur les bases de cette carte, et relancer le processus de résolution. Cette approche a ceci de séduisant qu'elle semble indépendante de la méthode numérique utilisée pour discrétiser le problème. Elle permet en outre de voir le processus de résolution comme l'exécution d'un solveur classique à géométrie fixée. Ainsi, elle permet de réutiliser pleinement un logiciel conçu pour des méthodes non–adaptatives. Enfin, les algorithmes de génération de maillages utilisés permettent d'assurer la bonne qualité géométrique des éléments générés.

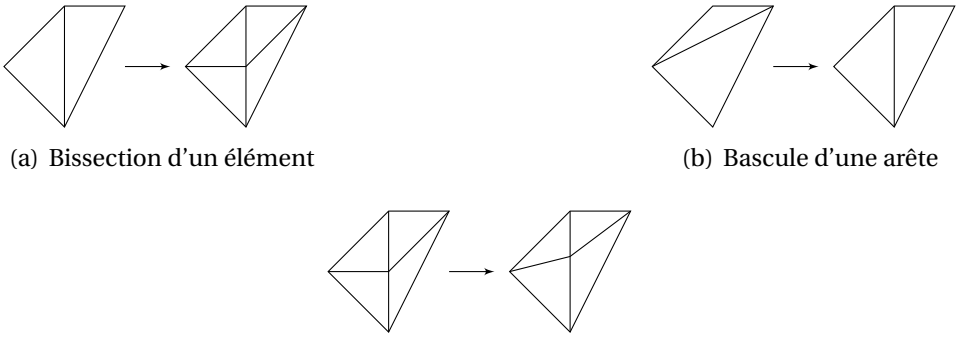
Cette approche présente toutefois quelques inconvénients qu'il ne faut pas sous–estimer. Du point de vue numérique, il est nécessaire de définir un processus d'interpolation des champs entre deux maillages qui n'ont en commun que la description géométrique de leur frontière extérieure. Une telle interpolation peut s'avérer à la fois coûteuse et peu précise. La grande faiblesse de cette approche reste cependant qu'il s'agit d'une démarche essentiellement séquentielle, où alternent les phases de résolution et celles de transformation du maillage. Or, si la phase de résolution peut être facilement parallélisée, il n'en est rien de la phase de regénération du maillage. Généralement, ce processus est réalisé de façon séquentielle sur un unique processeur. On voit donc que la mise à jour de la géométrie constitue un goulot d'étranglement dans le processus global, et ce tant en termes de temps calcul qu'en termes d'espace mémoire. En effet, si un unique processeur est sollicité pour effectuer la mise à jour du maillage, les autres processeurs se voient obligés d'attendre qu'il ait terminé, ce qui entraîne une sous–exploitation de la machine. Par ailleurs, le maillage généré doit pouvoir tenir dans la mémoire du processeur en question, ce qui impose une limite très restrictive sur la dimension du maillage. On voit donc que cette approche, bien que numériquement séduisante, présente des inconvénients majeurs en vue d'une utilisation dans un contexte d'exécution parallèle.

1.4.2 Optimisation locale du maillage

Les logiciels AOMD [114–116] et PARED [35–38] proposent une méthode relativement différente. Toujours fondés sur une approche géométrique du problème, ces logiciels présentent cette fois un caractère parallèle beaucoup plus marqué. Le principe de la

[1] Institut National de Recherche en Informatique et en Automatique.

méthode sur laquelle ils reposent consiste à limiter la mise à jour du maillage à une succession de procédures élémentaires, telles que la bissection d'un élément, la bascule d'une arête, le barycentrage de sommets, etc. Ces modifications élémentaires, illustrées sur la figure 1.5, ont la particularité d'être locales. Autrement dit, chacune de ces modi-



(a) Bissection d'un élément

(b) Bascule d'une arête

(c) Barycentrage d'un sommet

FIG. 1.5 : Opérations élémentaires de modification du maillage

fications ne perturbe qu'un nombre fini et petit d'éléments adjacents les uns aux autres. Par conséquent, elles peuvent être exécutées en parallèle par chacun des processeurs sur leur sous–domaine propre. Il semble donc que cette méthode soit plus efficace que la précédente, car elle ne présente pas de goulot d'étranglement séquentiel.

Cependant, elle n'est pas exempte d'inconvénients pour autant. Tout d'abord, si dans le cas bidimensionnel, les modifications autorisées s'avèrent relativement simples, celles utilisables sur un cas tridimensionnel s'avèrent beaucoup plus complexes (voir les remarques de P.-L. GEORGE et H. BOROUCHAKI dans [67]). Ensuite, ces modifications, pour élémentaires qu'elles soient, ne garantissent pas la qualité géométrique des éléments du maillage. Il s'agit uniquement d'une heuristique, car aucun résultat théorique n'a été établi dans ce sens. Enfin, une fois le maillage mis à jour, il est fort probable qu'il soit déséquilibré. Pour résoudre ce problème, les logiciels AOMD et PARED adoptent une démarche similaire. Elle consiste à migrer un certain nombre d'éléments d'un processeur à un autre, de façon à améliorer la distribution des degrés de liberté sur les processeurs. La sélection de ces éléments à migrer peut se faire de plusieurs façons, mais il reste cependant difficile d'assurer tant l'équilibrage de la charge que la minimisation des volumes de données à communiquer.

1.4.3 Solutions multi–échelles

Une autre façon d'aborder l'adaptation de maillage a été mise en œuvre dans des codes CHOMBO [43], AGRIF [49, 50], PLTMG [11] et PHAML [101–105]. Le point commun de ces programmes est qu'ils reposent sur une description hiérarchique de la géométrie, de façon à proposer un algorithme de résolution reposant sur une méthode multi–grilles. Pour ce faire, ces logiciels utilisent un raffinement simple des éléments du maillage, reposant généralement sur une méthode de subdivision (voir le chapitre 3). La prise en compte des non–conformités introduites par ces technique peut être réalisée via en utilisant des raffinements particuliers qui permettent de rétablir la conformité globale du maillage.

L'intérêt majeur de cette approche réside dans le fait que les méthodes multi–grilles fournissent généralement de bons préconditionneurs. C'est pourquoi on espère ainsi pouvoir disposer de solveurs particulièrement efficaces en exploitant la hiérarchie de maillages. Malheureusement, la prise en compte des non–conformités au niveau géométrique conduit à une explosion combinatoire des cas à prendre en compte lorsque l'on augmente la dimension du problème. Si le cas de la dimension 2 reste relativement simple, car un algorithme de fermeture transitive permet de résoudre la difficulté, le cas tridimensionnel s'avère beaucoup plus difficile à mettre en œuvre car le nombre de configurations est nettement plus important. Par ailleurs, l'implémentation parallèle proposée par exemple dans les codes PHAML et PLTMG reste limitée à l'étude de problèmes bidimensionnels, car elle repose sur un parcours du maillage par voisinage dont les propriétés établies pour la dimension 2 ne sont plus valables en dimension 3.

1.5 L'approche par décomposition de domaine

1.5.1 Idée générale

Nous étudions ici la possibilité d'utiliser une approche par décomposition de domaine pour mettre en œuvre notre méthode adaptative. L'idée originale a été proposée par C. BERNARDI, Y. MADAY et F. HECHT dans [21, 22], et consiste à raisonner en termes de niveaux de raffinement. Considérons par exemple un maillage initial, sur lequel on calcule une première solution. On peut alors estimer l'erreur numérique et localiser les éléments où cette erreur est la plus importante. En choisissant de raffiner ces éléments, on espère voir décroître l'erreur globale. On introduit donc deux sous–domaines, cor-

(a) Maillage initial et localisation de l'erreur

(b) Domaine découpé selon les niveaux de raffinement

FIG. 1.6 : Construction des sous–domaines selon le raffinement des éléments

respondant chacun à un niveau de raffinement. Le premier rassemble les éléments du maillage initial qui ont été conservés, alors que le second contient les éléments issus du raffinement des éléments ayant une erreur importante (voir figure 1.6). On isole ainsi les éléments selon leur niveau de raffinement.

Du point de vue géométrique, l'adaptation de maillage se trouve facilitée par le fait qu'un sous–domaine entier doit être raffiné. Il est d'ailleurs possible d'utiliser plusieurs formes de raffinement de maillage. Nous étudions cette question au chapitre 3, et nous verrons qu'il est possible d'utiliser une stratégie de raffinement simple, qui assure la non–dégénérescence des éléments du maillage.

Du point de vue numérique, cette approche nécessite de coupler correctement des problèmes posés sur des maillages de finesse différentes. Comme on peut le voir sur la figure 1.6, la phase de raffinement introduit une non–conformité dans le maillage. Plus exactement, le maillage apparaît localement conforme dans chaque sous–domaine, mais globalement non–conforme, au niveau des interfaces entre sous–domaines. Nous présentons plusieurs approches permettant de traiter cette difficulté au chapitre 4. Nous verrons que le formalisme des éléments joints nous permet de résoudre ce problème élégamment, sans introduire de nouveaux concepts. Il faut cependant développer ce formalisme dans le contexte de la méthode VEF de Trio-U.

Enfin, du point de vue du parallélisme, l'approche par décomposition de domaine reste très proche des modèles théoriques utilisés en conception logicielle. Ceci permet d'appréhender simplement la problématique de l'implémentation parallèle de la méthode. Le travail de thèse de A. SCHWERTNER CHARÃO [132] nous indique qu'il est possible de mettre en œuvre l'approche par décomposition de domaine de façon efficace en utilisant la notion de processus légers. Bien que Trio-U ne permette pas à l'heure actuelle d'utiliser le mécanisme de processus légers, nous développons au chapitre 5 les concepts nécessaires à la mise en œuvre d'un équilibrage de charge pertinent.

1.5.2 Intérêts dans le cadre de Trio-U

Comme nous l'avons indiqué au paragraphe 1.2, la conception de Trio-U ne repose pas sur les techniques de décomposition de domaine. Certes, l'implémentation parallèle de Trio-U nécessite de découper le maillage pour le distribuer sur les processeurs, mais les méthodes de résolution utilisées ne font pas intervenir de notion de sous–problèmes couplés. On est donc en droit de se demander quel est l'intérêt d'aborder la question de l'intégration d'une méthode adaptative dans Trio-U par le biais des méthodes de décomposition de domaine. En réalité, les intérêts de cette approche sont multiples.

Tout d'abord, le formalisme des méthodes de décomposition de domaine permet d'assurer la cohérence mathématique de la démarche. Ce point est essentiel étant donnés les objectifs de fiabilité de la plate–forme.

Ensuite, l'approche par décomposition de domaine permet de percevoir le parallélisme d'une façon très claire : il s'agit de distribuer n problèmes sur p processeurs, sous la contrainte de l'équilibrage de la charge, et de la minimisation des volumes de communication. Nous l'avons déjà souligné, cette vision est extrêmement flexible, et très proche des modèles théoriques utilisés pour concevoir des applications parallèles.

Enfin, l'idée d'isoler les niveaux de raffinement dans des sous–domaines distincts permet de réutiliser pleinement les outils présents dans Trio-U. En effet, les éléments présentant le même niveau de raffinement constituent un sous–domaine dont le maillage est conforme. Autrement dit, un tel sous–domaine remplit les conditions nécessaires pour pouvoir servir à la construction d'un problème numérique avec Trio-U. Ainsi, il est possible de discrétiser et d'assembler séparément les sous–problèmes définis sur les différents niveaux de raffinement à l'aide des classes déjà implémentées dans Trio-U.

1.5.3 Les difficultés qui se posent

Nous pouvons d'ores–et–déjà isoler trois types de difficultés posées par la méthode que nous envisageons :

– les difficultés d'ordre géométrique ;
– les difficultés d'ordre numérique ;
– les difficultés d'ordre informatique.

Les difficultés d'ordre géométrique apparaissent dès que l'on cherche à imposer des contraintes sur la forme prise par l'adaptation du maillage. Par exemple, on veut s'assurer que le raffinement n'introduit pas d'élément de qualité médiocre, susceptible d'induire un problème discret mal conditionné. De même, la prise en compte du déraffinement complique singulièrement les algorithmes géométriques. Elle est cependant nécessaire à l'efficacité de la méthode, en particulier lors de l'étude de problèmes instationnaires.

Du point de vue numérique, nous devons spécifier une méthode mathématiquement cohérente. Nous sommes aidés en cela par le formalisme des méthodes de décomposition de domaine, mais nous verrons qu'il est nécessaire de l'adapter aux discrétisations de Trio-U. Nous devons également assurer que les estimations d'erreur permettent effectivement de localiser les singularités de la solution de façon précise. Ceci afin de démontrer que les régions raffinées sont pertinentes en regard du problème étudié.

Enfin, les difficultés d'ordre informatique portent essentiellement sur la conception logicielle d'un module à intégrer à Trio-U permettant de mettre en œuvre notre méthode. Nous verrons que de nombreux détails peuvent s'avérer problématiques. Par exemple, il est nécessaire d'empêcher les communications inter–processeurs lors de la phase de construction des sous–problèmes, alors que la démarche usuelle consiste à les imposer. Il est également nécessaire de développer une nouvelle perception du parallélisme dans Trio-U. En effet, comme nous l'avons souligné plus tôt, le caractère statique des vecteurs distribués de Trio-U s'oppose à leur utilisation dans un contexte de maillage dynamique. Il faut donc proposer un autre moyen d'automatiser les communications, tout en développant les outils permettant d'équilibrer la charge dynamiquement au cours du calcul.

Résumé

Ce premier chapitre nous a permis de présenter dans le détail la problématique et les enjeux de nos travaux. Nous justifions notre démarche par les intérêts que représentent les méthodes adaptatives pour assurer tant la fiabilité que l'efficacité d'un logiciel de simulation numérique. L'analyse des travaux précédemment réalisés dans ce domaine nous a permis d'orienter les premiers choix pour notre méthode adaptative, et de soulever les questions auxquelles nous allons devoir apporter des réponses dans la suite de ce mémoire.

Nous choisissons donc de mettre en œuvre une méthode adaptative reposant sur le modèle des méthodes de décomposition de domaine, tel que cela a été suggéré par C. BERNARDI, Y. MADAY *et* F. HECHT *dans les articles [21, 22]. Nous accorderons une attention particulière aux questions suivantes :*

– *« quelle forme choisir pour l'adaptation géométrique du maillage ? » ;*
– *« quelles sont les concepts à développer pour concevoir une méthode adaptative numériquement cohérente avec les principes de la méthode VEF de Trio-U ? » ;*
– *« comment envisager l'implémentation de cette méthode pour garantir son efficacité dans un contexte d'exécution parallèle ? » ;*
– *« quelle architecture logicielle faut-il retenir pour faciliter l'intégration de cette méthode au sein de la plate–forme Trio-U ? ».*

Apporter des réponses pertinentes à ces questions sera l'objet des chapitres qui suivent.

CHAPITRE 2

Rappels sur le parallélisme et les méthodes de décomposition de domaine

Partant de la brève étude bibliographique exposée dans le paragraphe précédent, nous avons choisi de développer une approche adaptative reposant sur le formalisme des méthodes de décomposition de domaine. Le principe retenu consiste à partitionner le domaine de calcul de sorte que les éléments d'un même sous–domaine présentent le même degré de raffinement. La résolution du problème couplé qui en découle est alors réalisée à l'aide d'une méthode d'éléments joints. Nous nous proposons donc dans ce chapitre de revenir sur les concepts clefs des méthodes de décomposition de domaine, et de détailler en particulier les idées fondatrices de la méthode des éléments joints. Nous présentons également un certain nombre de concepts liés au parallélisme, ainsi que diverses techniques de programmation des machines massivement parallèles. Ce sera l'occasion de présenter plus en détail les aspects techniques de l'implémentation parallèle de la plate–forme Trio-U.

2.1 Principe des méthodes de décomposition de domaine

2.1.1 Diviser pour régner

L'idée des méthodes de décomposition de domaine est relativement simple et naturel. L'objectif consiste à ramener la résolution d'un problème numérique complexe défini sur un domaine donné à celle d'une collection de sous–problèmes (que l'on espère plus simples) définis sur des sous–domaines recouvrant le domaine initial.

Historiquement, la première méthode de décomposition de domaine a été introduite par H.-A. SCHWARZ en 1870 [131], afin de démontrer un théorème d'existence de fonctions harmoniques ayant des valeurs imposées sur la frontière d'un domaine de géométrie complexe. Le point crucial du travail de H.-A. SCHWARZ a été de démontrer la convergence en norme infinie d'une résolution de ce problème par une méthode de décomposition de domaine. Depuis lors, de nombreux travaux sont venus étoffer cette théorie, dont notamment ceux de S.-L. SOBOLEV [137], de I. BABUŠKA [8], et de P.-L. LIONS [88–90]. Nous disposons donc aujourd'hui d'un formalisme mathématique bien établi autour de cette approche. Nous renvoyons à [142] pour une présentation détaillée de ce formalisme.

2.1.2 Avantages et Inconvénients

Le cadre théorique offert par les méthodes de décomposition de domaine constitue indéniablement leur principal avantage. Par ailleurs, ce cadre se transpose aisément en termes d'implémentation, ce qui facilite d'autant leur utilisation. Les motivations qui conduisent au choix d'une méthode de décomposition de domaine peuvent être très variées. Par exemple, si on étudie un problème à l'aide d'une méthode de différences finies sur un domaine complexe, il peut être pratique de pouvoir se ramener à une collection de géométrie simples, sur lesquels la définition d'une grille cartésienne est presque triviale. Cette idée est illustrée sur la figure 2.1. Clairement, définir un maillage structuré

(a) Domaine initial (b) Domaine découpé

FIG. 2.1 : Décomposition de domaine pour la méthode des différences finies

sur un domaine comme celui de la figure 2.1(a) nécessite de décrire convenablement ses frontières. À l'inverse, définir une grille structurée sur des sous–domaines rectangulaires comme sur la figure 2.1(b) devient trivial : il suffit pour chaque sous–domaine de connaitre les coordonnées des points extrémaux et le pas du maillage dans chaque direction. Ainsi, alors que la première approche nécessite de définir une géométrie complète, la seconde permet de générer la géométrie à partir de sous–domaines simples. On économise donc beaucoup en termes d'espace mémoire, ainsi qu'en complexité des algorithmes d'assemblages.

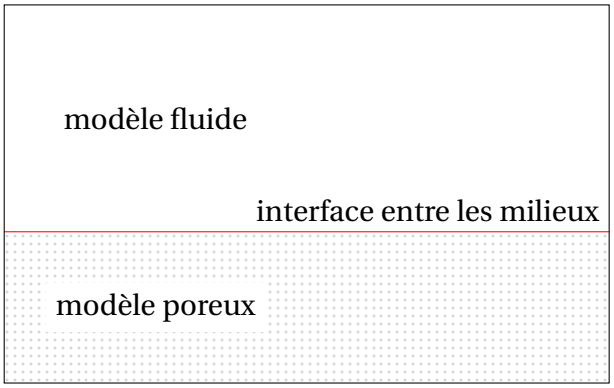


FIG. 2.2 : Décomposition de domaine pour l'isolement de modèles physiques

Une autre motivation peut être d'isoler des modèles physiques différents dans des sous–domaines séparés. La figure 2.2 présente une configuration où l'on étudie l'écoule-

ment d'un fluide dans un canal, ce dernier comportant une partie poreuse (en bas) et une partie libre (en haut). Dans la région libre, on retiendra par exemple un modèle de mécanique des fluides classique, reposant sur les équations de NAVIER–STOKES. Dans la partie poreuse, le modèle sera plus probablement fondé sur les équations de DARCY. Une méthode de décomposition de domaine permet alors de résoudre chacun de ces problèmes séparément, tout en assurant la continuité et la cohérence de la solution au voisinage de l'interface entre les régions.

Enfin, les méthodes de décomposition de domaine présentent un caractère intrinsèquement parallèle. En effet, on peut voir la résolution de chaque sous–problème comme une tâche à réaliser. Ces sous–domaines étant adjacents, la méthode de décomposition de domaine impose un certain nombre de contraintes sur ces interfaces. Ainsi, les tâches correspondant aux sous–problèmes sont en réalité inter–dépendantes, et l'on peut créer un graphe de dépendance entre ces tâches. L'objectif est alors de déterminer un ordonnancement de ces tâches sur les processeurs d'une machine parallèle qui satisfasse le graphe de dépendance. Cette approche est celle présentée dans les travaux de thèse de A. SCHWERTNER CHARÃO.

Les méthodes de décomposition de domaine présentent toutefois un certain nombre d'inconvénients. Tout d'abord, ces méthodes restent jusqu'ici relativement peu utilisées dans un cadre industriel. En effet, la plupart des codes industriels parallèles, à l'instar de Trio-U, ne reposent pas sur ce formalisme. La démarche usuelle consiste plutôt à utiliser des solveurs itératifs parallèles sur un problème discret distribué sur les processeurs. La vision du parallélisme dans ce contexte est certes simplifiée, quoique beaucoup moins souple que celle adoptée par les méthodes de décomposition de domaine. Un autre inconvénient des méthodes de décomposition de domaine est qu'elles restent relativement difficiles à transposer en dimension 3. Il s'agit là plutôt d'une barrière technique que d'un problème théorique. En effet, la difficulté qui se présente est de définir des maillages d'interfaces qui permettent de raccorder efficacement les solutions. Enfin, il reste que les méthodes de décomposition de domaine reposent sur des algorithmes essentiellement itératifs, qui sont assez difficiles à préconditionner. La littérature est certes abondante dans ce domaine, mais elle reste cantonnée à l'étude de problèmes modèles, beaucoup plus simples que les problèmes relevant du monde industriel.

2.1.3 Notations et définitions

Nous introduisons maintenant les notations qui seront utilisées dans la suite de ce chapitre. On considère donc un domaine Ω de $\mathbb{R}^d$, pour $d = 2$ ou 3, dont la frontière est notée $\partial\Omega$. On cherche à résoudre un problème pouvant s'écrire sous la forme générique suivante :

$$\begin{cases} \mathscr{L}u = f & \text{dans } \Omega \\ \mathscr{B}u = g & \text{sur } \partial\Omega \end{cases}$$

Dans ce problème, $\mathscr{L}$ et $\mathscr{B}$ désignent des opérateurs aux dérivées partielles quelconques, et f et g sont des fonctions données. Ainsi, si $\mathscr{B}$ désigne l'identité, alors on cherche à imposer des conditions aux limites de DIRICHLET, alors que si $\mathscr{B} = \partial_n$, on a affaire à des conditions aux limites de NEUMANN. De même, suivant la nature du problème étudié, $\mathscr{L}$ peut désigner un opérateur laplacien, un opérateur de diffusion–convection, etc. Nous

supposerons toutefois que ce problème est bien posé, à savoir qu'il admet une unique solution dépendant continûment des données f et g.

Nous considèrerons une famille $\{\Omega_i\}_{1\leq i\leq n}$ de sous–domaines de Ω, formant un recouvrement de ce dernier :

$$\bigcup_{i=1}^{n}\overline{\Omega_i}=\overline{\Omega}$$

Pour chaque sous–domaine Ω_i, on notera :

- $\partial\Omega_i$ sa frontière ;
- $\partial\Omega_i^e$ sa frontière extérieure : $\partial\Omega_i^e=\partial\Omega_i\cap\partial\Omega$;
- Γ_{ij} son interface avec un autre sous–domaine : $\Gamma_{ij}=\partial\Omega_i\cap\Omega_j$.

Essentiellement, deux cas de figure s'offrent à nous, selon que les sous–domaines se recouvrent ou non, comme cela est illustré sur la figure 2.3. Dans le cas où les sous–

(a) Décomposition avec recouvrement

(b) Décomposition sans recouvrement

FIG. 2.3 : Exemples de décomposition de domaines

domaines ne se recouvrent pas, c'est–à–dire si pour tout i différent de j, on a $\mathring{\Omega}_i\cap\mathring{\Omega}_j=\emptyset$, alors la propriété suivante est valide : $\Gamma_{ij}=\Gamma_{ji}=\partial\Omega_i\cap\partial\Omega_j$. À l'inverse, si les sous–domaines se recouvrent, alors en général on a $\Gamma_{ij}\neq\Gamma_{ji}$. Nous verrons que les différentes méthodes de décomposition de domaine n'ont pas les mêmes besoins en termes de recouvrement des sous–domaines : certaines imposeront un recouvrement nul, alors que d'autres ne pourront converger que si ce dernier n'est pas nul.

2.2 Éléments de classification

2.2.1 Approche de type « EDP »

Une première famille de méthodes de décomposition de domaines rassemble les méthodes dites « de SCHWARZ ». Ces méthodes s'inspirent de la méthode originelle proposée par ce dernier en 1870. Nous employons le terme d'approche de type « EDP », car ces méthodes reposent sur la résolution de problèmes locaux qui se présentent sous la forme d'équations aux dérivées partielles. Nous illustrons ici les deux versions les plus classiques de cette approche, dans le cas d'un domaine découpé en deux sous–domaines.

La première méthode de SCHWARZ repose sur l'algorithme alterné présenté dans l'encadré 2.4. Dans cet algorithme, les opérateurs $\mathscr{C}_1$ et $\mathscr{C}_2$ représentent des opérateurs aux dérivées partielles correspondant aux conditions aux limites imposées respectivement sur les bords Γ_{12} et Γ_{21}. On remarque que si ces opérateurs sont égaux à l'identité, alors le recouvrement de sous–domaines est nécessaire. En effet, s'il n'y a pas de recouvrement,

Initialisation : u_2^0 donné sur Γ_{12}
Itérer jusqu'à convergence {

résoudre :

$$\begin{aligned} &\mathscr{L} u_1^n = f && \text{sur } \Omega_1 \\ &\mathscr{B} u_1^n = g && \text{sur } \partial\Omega_1^e \\ &\mathscr{C}_1 u_1^n = \mathscr{C}_1 u_2^{n-1} && \text{sur } \Gamma_{12} \end{aligned}$$

puis résoudre :

$$\begin{aligned} &\mathscr{L} u_2^n = f && \text{sur } \Omega_2 \\ &\mathscr{B} u_2^n = g && \text{sur } \partial\Omega_2^e \\ &\mathscr{C}_2 u_2^n = \mathscr{C}_2 u_1^n && \text{sur } \Gamma_{21} \end{aligned}$$

n = n+1
}

FIG. 2.4 : Méthode de SCHWARZ multiplicative

la valeur à l'interface n'évoluera jamais. Cet algorithme est qualifié de « multiplicatif » car il est équivalent à une résolution par une méthode de GAUSS–SEIDEL par blocs. Remarquons enfin que cette méthode ne présente pas de caractère parallèle explicite : les deux résolutions doivent être faites consécutivement. Cependant, si le domaine est découpé en plus de deux sous–domaines, alors on peut utiliser une technique de coloriage pour extraire le parallélisme de cet algorithme. L'idée consiste à affecter une couleur à chaque sous–domaine, de sorte que deux sous–domaines voisins ne possèdent jamais la même couleur, tout en cherchant à minimiser le nombre de couleurs utilisées. La figure 2.5 présente un tel coloriage en dimension 2. Le principe consiste alors à itérer sur

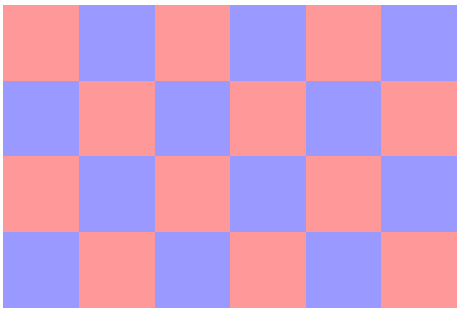


FIG. 2.5 : Exemple de coloriage des sous–domaines permettant une exécution parallèle de la méthode de SCHWARZ multiplicative

l'ensemble des couleurs, et à résoudre en parallèle tous les sous–problèmes associés à des sous–domaines de même couleur.

La seconde méthode de SCHWARZ est qualifiée d'« additive ». Elle est fondée sur l'algorithme 2.6. On a utilisé les mêmes notations pour l'application des conditions aux limites sur les frontières Γ_{12} et Γ_{21}. De même que pour l'algorithme multiplicatif, si ces conditions sont des conditions de DIRICHLET, le recouvrement des sous–domaines est nécessaire. Cependant, l'algorithme additif est équivalent à une résolution par blocs utilisant une méthode de JACOBI, et elle présente donc un caractère intrinsèquement paral-

Initialisation : u_1^0 donné sur Γ_{21}, u_2^0 donné sur Γ_{12}
Itérer jusqu'à convergence {

résoudre :

$$\begin{aligned} &\mathscr{L} u_1^n = f && \text{sur } \Omega_1 \\ &\mathscr{B} u_1^n = g && \text{sur } \partial\Omega_1^e \\ &\mathscr{C}_1 u_1^n = \mathscr{C}_1 u_2^{n-1} && \text{sur } \Gamma_{12} \end{aligned}$$

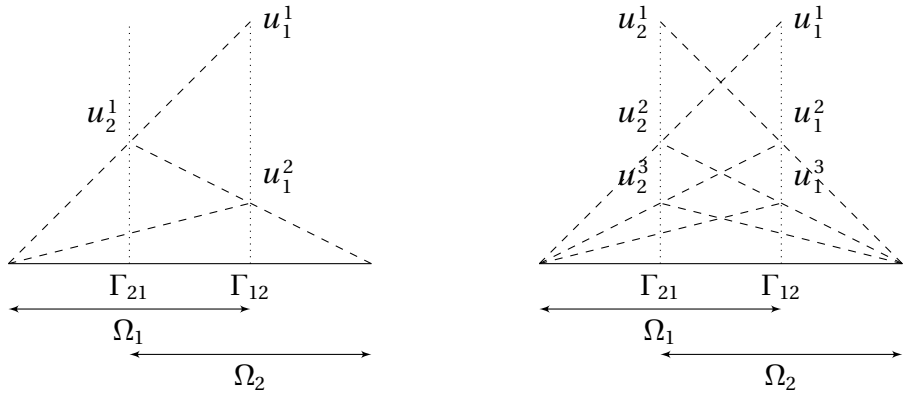
ainsi que :

$$\begin{aligned} &\mathscr{L} u_2^n = f && \text{sur } \Omega_2 \\ &\mathscr{B} u_2^n = g && \text{sur } \partial\Omega_2^e \\ &\mathscr{C}_2 u_2^n = \mathscr{C}_2 u_1^{n-1} && \text{sur } \Gamma_{21} \end{aligned}$$

n = n+1
}

FIG. 2.6 : Méthode de SCHWARZ additive

lèle. En effet, la résolution d'un des sous–problèmes ne dépend que de la solution précédemment calculée, et non des valeurs en cours de calcul sur un sous–domaine voisin. En termes de convergence, cette propriété a un prix. En effet, on montre assez simplement que l'algorithme multiplicatif converge sensiblement plus rapidement que l'algorithme additif. La figure 2.7 illustre cette idée dans le cas d'un problème en dimension 1. D'ailleurs, B. SMITH, P. BJORSTAD et W. GROPP indiquent dans [136] qu'en pratique, la mé-



(a) Cas de l'algorithme multiplicatif (b) Cas de l'algorithme additif

FIG. 2.7 : Convergence des méthodes de SCHWARZ

thode additive nécessite en moyenne environ deux fois plus d'itérations que la méthode multiplicative.

2.2.2 Approche algébrique

Une approche algébrique des méthodes de décomposition de domaine consiste à travailler sur la version discrétisée du problème, plutôt que sur le système d'équations aux dérivées partielles. La méthode de référence est la méthode de SCHUR dite « primale »,

que nous allons maintenant présenter. Soulignons que ces méthodes fonctionnent essentiellement pour des décompositions en sous–domaines qui ne se recouvrent pas.

Une fois discrétisé sur l'ensemble du domaine, le problème modèle se résume à la résolution d'un système linéaire de la forme $Ax = b$. La décomposition du domaine Ω en sous–domaines revient à ordonner les inconnues du vecteur x selon leur appartenance à l'un ou l'autre des sous–domaines. Il s'agit ni plus ni moins que d'une renumérotation des inconnues. On représente donc le vecteur x sous la forme suivante :

$$x = \begin{bmatrix} x_I^1 \\ x_I^2 \\ \vdots \\ x_I^n \\ x_\Gamma \end{bmatrix}$$

où x_I^j et x_Γ rassemblent respectivement les inconnues définies à l'intérieur du j^{e} sous–domaine, et celles définies sur l'union des interfaces Γ_{kl}. Le système linéaire à résoudre prend alors la forme suivante :

$$\begin{bmatrix} A_{II}^1 & & & & A_{I\Gamma}^1 \\ & A_{II}^2 & & & A_{I\Gamma}^2 \\ & & \ddots & & \vdots \\ & & & A_{II}^n & A_{I\Gamma}^n \\ A_{\Gamma I}^1 & A_{\Gamma I}^2 & \dots & A_{\Gamma I}^n & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} x_I^1 \\ x_I^2 \\ \vdots \\ x_I^n \\ x_\Gamma \end{bmatrix} = \begin{bmatrix} b_I^1 \\ b_I^2 \\ \vdots \\ b_I^n \\ b_\Gamma \end{bmatrix}$$

On peut alors utiliser formellement une élimination de GAUSS pour ne conserver que les variables x_Γ. On obtient alors le système suivant :

$$Sx_\Gamma = d$$

où l'on a posé :

$$S = A_{\Gamma\Gamma} - \left(\sum_{i=1}^{n} A_{\Gamma I}^i \left(A_{II}^i \right)^{-1} A_{I\Gamma}^i \right) \qquad d = b_\Gamma - \left(\sum_{i=1}^{n} A_{\Gamma I}^i \left(A_{II}^i \right)^{-1} b_I^i \right)$$

La matrice S est appelée « complément de SCHUR » du problème étudié. Une fois ce système résolu, on dispose des valeurs de la solution sur les interfaces entre sous–domaines. On pourra alors résoudre en parallèle chacun des sous–problèmes de façon indépendante.

La résolution du problème portant sur le complément de SCHUR ne peut être envisagée de façon directe. Il semble en effet illusoire de pouvoir assembler la matrice S de façon explicite. Cependant, les méthodes itératives, telles que les méthodes de KRYLOV [121], ne nécessitent pas de connaître S explicitement, mais simplement de savoir calculer le résultat du produit de S sur un vecteur donné :

$$y = Sv = A_{\Gamma\Gamma} v - \left(\sum_{i=1}^{n} A_{\Gamma I}^i \left(A_{II}^i \right)^{-1} A_{I\Gamma}^i v \right)$$

Le calcul de $A_{I\Gamma}^i v$ est un simple produit matrice–vecteur local. Le calcul de $\left(A_{II}^i \right)^{-1} A_{I\Gamma}^i v$ est réalisé localement en résolvant le système $A_{II}^i z = A_{I\Gamma}^i v$. Enfin, la multiplication par

$A^i_{\Gamma I}$ est également un produit matrice–vecteur local. Il ne reste donc qu'à calculer $A_{\Gamma\Gamma}v$. Or, $A_{\Gamma\Gamma}$ peut être décomposé en une somme de contributions provenant de chacun des sous–domaines, que l'on note $A^i_{\Gamma\Gamma}$. Il suffit alors de calculer localement $A^i_{\Gamma\Gamma}v$ et d'assembler les résultats. En résumé, on peut écrire :

$$S=\sum_{i=1}^{n}S^i=\sum_{i=1}^{n}\left(A^i_{\Gamma\Gamma}-A^i_{\Gamma I}\left(A^i_{II}\right)^{-1}A^i_{I\Gamma}\right)$$

et l'application de S à un vecteur v peut être calculée en appliquant les S^i en parallèle et en sommant les résultats.

2.2.3 Approche variationnelle

Une troisième approche des méthodes de décomposition de domaine consiste à travailler à partir de la formulation variationnelle associée au problème. Cette idée est exploitée par exemple dans la méthode de SCHUR dite « duale ». Nous verrons dans un instant la raison de cette appellation. La formulation variationnelle associée au problème modèle peut être présentée sous la forme suivante :

Trouver $u\in H$ tel que :

$$\forall v\in H,\qquad a(u,v)=l(v)$$

où H, $a(\cdot,\cdot)$ et $l(\cdot)$ désignent respectivement un espace de Hilbert, une forme bilinéaire continue et coercive sur $H\times H$, et une forme linéaire continue sur V. Le théorème de LAX–MILGRAM [54] assure le caractère bien posé de ce problème. La résolution de ce problème est équivalente à la résolution du problème de minimisation suivant :

Trouver $u\in H$ réalisant :

$$J(u)=\min_{v\in H}J(v)\qquad où\qquad J(v)=\frac{1}{2}a(v,v)-l(v)$$

Le découpage du domaine Ω en sous–domaines revient alors à décomposer l'espace H en un produit d'espaces H_i sur lequel s'applique une contrainte. En dualisant cette contrainte, on introduit un multiplicateur de LAGRANGE, et on définit un nouveau problème, comme étant la recherche d'un point selle du lagrangien associé. La discrétisation de ce problème conduit à l'élaboration d'un problème matriciel portant à la fois sur les inconnues primales u et le multiplicateur de LAGRANGE ou inconnue duale.

Considérons par exemple un problème de LAPLACE, avec des conditions de DIRICHLET homogènes :

$$\begin{aligned}-\Delta u&=f&&\text{sur }\Omega\\ u&=0&&\text{sur }\partial\Omega\end{aligned}$$

Le problème de minimisation associé est défini par :

Trouver $u\in H^1_0(\Omega)$ réalisant :

$$J(u)=\min_{v\in H^1_0(\Omega)}J(v)\qquad où\qquad J(v)=\frac{1}{2}\int_\Omega|\nabla v|^2-\int_\Omega fv$$

Si Ω est décomposé en deux sous–domaines, ce problème devient :

Trouver $(u_1, u_2) \in H$ *réalisant :*

$$J_1(u_1) + J_2(u_2) = \min_{(v_1,v_2)\in H} J_1(v_1) + J_2(v_2)$$

où on a défini :

$$H = \left\{(v_1, v_2) \in H^1(\Omega_1) \times H^1(\Omega_2),\ v_i = 0 \text{ sur } \partial\Omega_i^e, v_1 = v_2 \text{ sur } \Gamma = \Gamma_{12} = \Gamma_{21}\right\}$$
$$J_i(v_i) = \frac{1}{2}\int_{\Omega_i} |\nabla v_i|^2 - \int_{\Omega_i} f v_i$$

On peut dualiser la contrainte de continuité sur Γ, en introduisant un multiplicateur de LAGRANGE. Pour cela, on cherchera un point–selle du lagrangien :

$$\mathscr{L}(v_1, v_2, \mu) = J_1(v_1) + J_2(v_2) + \int_\Gamma \mu(v_1 - v_2)$$

En dérivant ce lagrangien par rapport à v_1, v_2 et μ au voisinage de son point–selle, que l'on notera (u_1, u_2, λ), on peut caractériser ce dernier par les équations suivantes :

$$\begin{aligned}
&\int_{\Omega_1} \nabla u_1 \nabla v_1 + \int_\Gamma v_1 \lambda = \int_{\Omega_1} f v_1 && \forall v_1 \in H^1(\Omega_1), v_1 = 0 \text{ sur } \partial\Omega_1^e \\
&\int_{\Omega_2} \nabla u_2 \nabla v_2 - \int_\Gamma v_2 \lambda = \int_{\Omega_2} f v_2 && \forall v_2 \in H^1(\Omega_2), v_2 = 0 \text{ sur } \partial\Omega_2^e \\
&\int_\Gamma \mu(v_1 - v_2) = 0 && \forall \mu \in H_0^{\frac{1}{2}}(\Gamma)
\end{aligned}$$

La discrétisation de ce problème conduit à l'élaboration du système matriciel suivant :

$$\begin{bmatrix} A_1 & 0 & B_1^T \\ 0 & A_2 & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ \Lambda \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ 0 \end{bmatrix}$$

Comme dans la méthode de SCHUR primale, on peut éliminer les inconnues intérieures aux sous–domaines, pour construire un problème ne portant que sur les inconnues interfaciales :

$$\left(B_1 (A_1)^{-1} B_1^T + B_2 (A_2)^{-1} B_2^T\right) \Lambda = B_1 (A_1)^{-1} F_1 + B_2 (A_2)^{-1} F_2$$

Nous parlons ici d'une méthode de SCHUR duale, par opposition à la méthode de SCHUR primale, en raison de la façon dont on impose les contraintes à l'interface. Il est en effet possible d'aborder la méthode de SCHUR primale par une approche variationnelle. Cependant, là où la méthode duale introduit un multiplicateur de Lagrange pour appliquer la contrainte de continuité, la méthode primale consiste à chercher des solutions qui vérifient par définition la condition de continuité. Autrement dit, la méthode de SCHUR primale consiste à imposer intrinsèquement la contrainte de continuité dans les espaces fonctionnels utilisés, alors que la méthode de SCHUR duale consiste à n'imposer qu'une contrainte de continuité affaiblie.

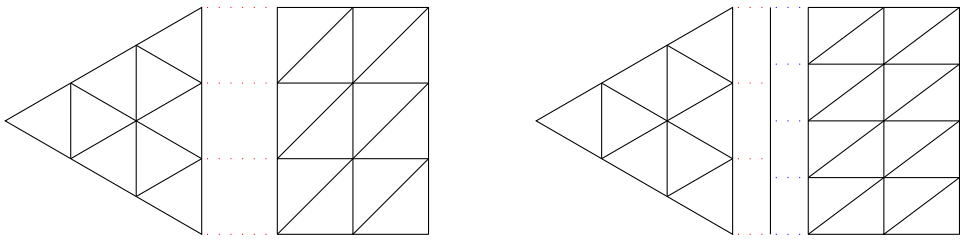
2.3 La méthode des éléments joints

2.3.1 Principe général

La méthode des éléments joints, connue également sous le nom de Mortar Element Method, constitue en quelque sorte une généralisation des méthodes de SCHUR. En effet, ces dernières reposent sur les deux hypothèses suivantes :

– la discrétisation utilisée est la même sur l'intégralité du domaine ;
– le maillage est conforme sur les interfaces entre les sous–domaines.

Cette notion de conformité du maillage aux interfaces est illustrée sur la figure 2.8. La



(a) Maillage conforme (b) Maillage non–conforme

FIG. 2.8 : La notion de conformité d'un maillage

méthode des éléments joints permet de lever ces deux contraintes. Elle a été proposée initialement par C. BERNARDI,Y. MADAY et A. PATERA dans [23], et elle permet de coupler de façon générique les résolutions des sous–problèmes, en offrant la possibilité d'utiliser des maillages non–conformes et des discrétisations différentes sur les sous–domaines.

Illustrons tout d'abord le principe de la méthode des éléments joints sur un exemple. On considère un problème de LAPLACE posé sur domaine Ω, muni de conditions aux limites de DIRICHLET homogènes :

$$\begin{aligned} -\Delta u &= f \qquad \text{sur } \Omega \\ u &= 0 \qquad \text{sur } \partial\Omega \end{aligned}$$

Soient Ω_1 et Ω_2 deux sous–domaines non vides formant une partition de Ω :

$$\overline{\Omega_1} \cup \overline{\Omega_2} = \overline{\Omega} \qquad \mathring{\Omega}_1 \cap \mathring{\Omega}_2 = \emptyset$$

On note Γ l'interface entre Ω_1 et Ω_2. Introduisons les espaces suivants :

$$\begin{aligned} X^1 &= \left\{ u \in H^1(\Omega_1),\ u|_{\partial\Omega_1 \cap \partial\Omega} = 0 \right\} \\ X^2 &= \left\{ u \in H^1(\Omega_2),\ u|_{\partial\Omega_2 \cap \partial\Omega} = 0 \right\} \\ X &= \left\{ (u_1, u_2) \in X^1 \times X^2,\ \gamma_1(u_1) = \gamma_2(u_2) \right\} \end{aligned}$$

où γ_1 (resp. γ_2) désigne l'opérateur de trace sur Γ définit sur X_1 (resp. X_2). Une formulation variationnelle de notre problème adaptée à ce découpage est donnée par :

Problème 2.1 : *Trouver* $(u_1, u_2) \in X$ *tel que :*

$$\forall (v_1, v_2) \in X, \qquad \int_{\Omega_1} \nabla u_1 \cdot \nabla v_1 + \int_{\Omega_2} \nabla u_2 \cdot \nabla v_2 = \int_{\Omega_1} f v_1 + \int_{\Omega_2} f v_2$$

À l'aide d'une méthode numérique quelconque, nous discrétisons les espaces X^1 et X^2, ce qui nous conduit à la construction de deux espaces fonctionnels discrets que nous noterons respectivement X_h^1 et X_h^2. Pour discrétiser complètement notre problème, il nous faut encore définir l'équivalent discret de la contrainte de continuité. Nous introduisons donc les espaces suivants :

$$W_h^1 = \left\{ u \in L^2(\Gamma),\ \exists u_1 \in X_h^1,\ u = \gamma_{1,h}(u_1) \right\}$$
$$W_h^2 = \left\{ u \in L^2(\Gamma),\ \exists u_2 \in X_h^2,\ u = \gamma_{2,h}(u_2) \right\}$$

où $\gamma_{i,h}$ désigne l'opérateur de trace discrète sur Γ défini sur X_h^i. Ces deux espaces sont en réalité des sous–espaces de $L^2(\Gamma)$. Nous introduisons donc l'opérateur $\pi_{1,h}$ de projection au sens de L^2 sur W_h^1 :

$$\forall (v, w) \in L^2(\Gamma) \times W_h^1, \qquad \int_\Gamma \pi_{1,h}(v) w = \int_\Gamma v\, w$$

Nous pouvons alors définir l'espace discret X_h par :

$$X_h = \left\{ (u_1, u_2) \in X_h^1 \times X_h^2,\ \gamma_{1,h}(u_1) = \pi_{1,h}\left(\gamma_{2,h}(u_2)\right) \right\}$$

Et la version discrète de notre problème s'écrit :

Problème 2.2 : *Trouver* $(u_1, u_2) \in X_h$ *tel que l'on ait :*

$$\forall (v_1, v_2) \in X_h, \qquad \int_{\Omega_1} \nabla u_1 \cdot \nabla v_1 + \int_{\Omega_2} \nabla u_2 \cdot \nabla v_2 = \int_{\Omega_1} f v_1 + \int_{\Omega_2} f v_2$$

Notons que nous aurions pu utiliser l'opérateur $\pi_{2,h}$ de projection au sens de $L^2(\Gamma)$ sur W_h^2 en lieu et place de l'opérateur $\pi_{1,h}$. Ce choix reste arbitraire et ne pose aucune difficulté théorique. Nous aurions alors simplement obtenu un autre problème discret.

Cet exemple permet de voir que l'idée fondamentale de la méthode des éléments joints consiste à imposer la contrainte de raccordement des solutions en un sens faible. Cela est réalisé en introduisant une discrétisation des interfaces, et en projetant chacune des traces de la solution de part et d'autre de l'interface sur l'espace discret interfacial. En pratique, les discrétisations des sous–domaines induisent, pour chaque interface, deux discrétisations possibles. On choisit donc l'une de ces discrétisations induites pour servir de discrétisation « maître », et on projette alors la trace de l'autre solution sur l'espace associé. Cette seconde discrétisation est alors qualifiée d'« esclave ».

2.3.2 Le formalisme proposé par F. Ben Belgacem

L'approche présentée dans le paragraphe précédent s'apparente à une méthode de Schur primale : la contrainte de raccordement apparaît explicitement dans la définition des espaces X et X_h. Il est possible de développer un formalisme se rapprochant d'une méthode de Schur duale. C'est ce que propose F. Ben Belgacem dans [18]. L'idée est, encore une fois, de dualiser la contrainte de raccordement, et d'obtenir un problème de type point–selle par l'introduction de multiplicateurs de Lagrange.

À titre d'exemple, nous considérons le problème suivant :

Problème 2.3 : *Trouver* $u \in H_0^1(\Omega)$ *vérifiant :*

$$\begin{aligned} -\Delta u &= f \qquad \textit{dans}\,\Omega \\ u &= 0 \qquad \textit{sur}\,\partial\Omega \end{aligned}$$

Comme dans le cadre de l'approche variationnelle, nous pouvons définir un problème de minimisation équivalent :

Problème 2.4 : *Trouver* $u \in H_0^1(\Omega)$ *réalisant :*

$$\min_{v \in H_0^1(\Omega)} J(v) \qquad \textit{où} \qquad J(v) = \frac{1}{2}\int_\Omega \nabla v \cdot \nabla v - \int_\Omega f v$$

Soient maintenant Ω_1 et Ω_2 deux sous–domaines non vides formant une partition de Ω. On définit également :

$$\Gamma = \partial\Omega_1 \cap \partial\Omega_2 \qquad \Gamma_1 = \partial\Omega_1 \cap \Gamma \qquad \Gamma_2 = \partial\Omega_2 \cap \Gamma$$

et on notera $\boldsymbol{n}_i$ la normale extérieure à Ω_i sur la frontière Γ_i. Introduisons les espaces suivants :

$$\begin{aligned} X^1 &= \left\{ u \in H^1(\Omega_1),\ u|_{\partial\Omega_1 \cap \partial\Omega} = 0 \right\} \\ X^2 &= \left\{ u \in H^1(\Omega_2),\ u|_{\partial\Omega_2 \cap \partial\Omega} = 0 \right\} \\ X &= \left\{ (u_1, u_2) \in X^1 \times X^2 \right\} \end{aligned}$$

Remarquons que la définition de X ne fait pas intervenir de contrainte de raccordement et que $H_0^1(\Omega) \subset X$. On introduit la forme bilinéaire :

$$\begin{aligned} a\ &: X \times X \longrightarrow \mathbb{R} \\ &(u, v) \longmapsto a(u, v) = \int_{\Omega_1} \nabla u_1 \cdot \nabla v_1 + \int_{\Omega_2} \nabla u_2 \cdot \nabla v_2 \end{aligned}$$

ainsi que la forme linéaire :

$$\begin{aligned} l\ &: X \longrightarrow \mathbb{R} \\ &u \longmapsto l(u) = \int_{\Omega_1} f u_1 + \int_{\Omega_2} f u_2 \end{aligned}$$

Le problème considéré est donc équivalent au problème de minimisation suivant :

Problème 2.5 : *Trouver* $u \in H_0^1(\Omega)$ *réalisant :*

$$\min_{v \in H_0^1(\Omega)} J(v) \qquad \textit{où} \qquad J(v) = \frac{1}{2} a(v, v) - l(v)$$

Pour définir l'espace fonctionnel dans lequel nous choisirons nos multiplicateurs de Lagrange, nous introduisons :

$$Z = \left\{ \boldsymbol{q} \in \left(L^2(\Omega) \right)^d,\ \nabla \cdot \boldsymbol{q} \in L^2(\Omega),\ \boldsymbol{q} \cdot \boldsymbol{n}|_{\partial\Omega} = 0 \right\}$$

Muni de la norme :

$$\|\boldsymbol{q}\|_Z = \left(\|\boldsymbol{q}\|^2_{(L^2(\Omega))^2} + \|\nabla\cdot\boldsymbol{q}\|^2_{L^2(\Omega)} \right)^{\frac{1}{2}}$$

il s'agit d'un espace de BANACH. Pour tout $\boldsymbol{q} \in Z$, on peut définir sa trace normale sur Γ_i, notée $\boldsymbol{q}\cdot\boldsymbol{n}_i$ et qui appartient à l'espace $H^{-\frac{1}{2}}(\Gamma_i)$, grâce à la formule de GREEN suivante :

$$\forall(\boldsymbol{q}, v_i) \in Z \times X^i, \qquad \int_{\Omega_i} \nabla v_i \cdot \boldsymbol{q} + v_i \nabla\cdot\boldsymbol{q} = \langle \boldsymbol{q}\cdot\boldsymbol{n}_i, \gamma_i(v_i)\rangle_{H^{-1/2}(\Gamma_i),H^{1/2}(\Gamma_i)}$$

Enfin, nous définissons l'espace des multiplicateurs de LAGRANGE par :

$$M = \left\{ (\psi_1,\psi_2) \in H^{-\frac{1}{2}}(\Gamma_1)\times H^{-\frac{1}{2}}(\Gamma_2),\ \exists \boldsymbol{q}\in Z, \psi_i = \boldsymbol{q}\cdot\boldsymbol{n}_i, i=1,2 \right\}$$

Muni de la norme induite, il s'agit également d'un espace de BANACH. Introduisons maintenant la forme bilinéaire suivante :

$$\begin{aligned} b\ &: X\times M \longrightarrow \mathbb{R} \\ &(v,\psi) \longmapsto b(v,\psi) = \sum_{i=1}^{2} \langle \psi_i, \gamma_i(v_i)\rangle_{H^{-1/2}(\Gamma_i),H^{1/2}(\Gamma_i)} \end{aligned}$$

En utilisant le théorème de HAHN–BANACH [54, 112], on montre que l'on peut caractériser l'espace $H^1_0(\Omega)$ par :

$$H^1_0(\Omega) = \{ u\in X,\ b(u,\psi)=0,\ \forall\psi\in M \}$$

Cette caractérisation de $H^1_0(\Omega)$ nous permet de voir le problème de minimisation de J comme un problème de minimisation sous contraintes :

Problème 2.6 : *Trouver $u \in X$ réalisant*

$$\min_{\substack{v\in X \\ b(v,\psi)=0,\ \forall\psi\in M}} J(v) \qquad \text{où} \qquad J(v) = \frac{1}{2}a(v,v) - l(v)$$

On introduit alors le lagrangien de ce problème :

$$\begin{aligned} \mathscr{L}\ &: X\times M \longrightarrow \mathbb{R} \\ &(u,\lambda) \longmapsto \mathscr{L}(u,\lambda) = \frac{1}{2}a(u,u) - l(u) + b(u,\lambda) \end{aligned}$$

La recherche du point–selle de ce lagrangien s'écrit alors :

Problème 2.7 : *Trouver $(u,\lambda)\in X\times M$ tel que :*

$$\begin{aligned} a(u,v) + b(v,\lambda) &= l(v) \qquad \forall v\in X \\ b(u,\mu) &= 0 \qquad \forall \mu\in M \end{aligned}$$

Tout ce que nous venons de voir est également valide pour la méthode de SCHUR duale. En effet, la différence entre les méthodes de SCHUR et la méthode des éléments joints apparaît lorsque l'on travaille sur la version discrète du problème. Cependant, nous avons développé ces résultats pour comprendre d'où proviennent les multiplicateurs de LAGRANGE. Le paragraphe précédent nous indique qu'ils appartiennent à l'espace dual de l'espace des traces des fonctions définies sur les sous–domaines.

Introduisons donc les espaces X_h^1 et X_h^2, définis comme précédemment par la discrétisation des espaces X^1 et X^2. Encore une fois, les discrétisations utilisées sur Ω_1 et Ω_2 ne sont pas nécessairement les mêmes, et les maillages de Ω_1 et de Ω_2 ne sont pas supposés conformes à l'interface Γ. L'espace discret X_h est cette fois défini par :

$$X_h = \left\{(u_1, u_2) \in X_h^1 \times X_h^2\right\}$$

Encore une fois, remarquons que nous ne faisons pas intervenir de condition de raccordement dans la définition de X_h. Nous définissons la forme bilinéaire discrète associée à notre problème par :

$$\begin{aligned} a_h \ : X_h \times X_h \ &\longrightarrow \mathbb{R} \\ (u, v) \quad &\longmapsto a_h(u, v) = \sum_{i=1}^{2} \int_{\Omega_i} \nabla u_i \cdot \nabla v_i \end{aligned}$$

La forme linéaire discrète s'écrit quant à elle :

$$\begin{aligned} l_h \ : X_h \ &\longrightarrow \mathbb{R} \\ v \quad &\longmapsto l_h(v) = \sum_{i=1}^{2} \int_{\Omega_i} f v_i \end{aligned}$$

Il reste à définir un équivalent discret de la forme bilinéaire b, qui sert dans le cas continu à imposer la contrainte de raccordement. Pour ce faire, nous introduisons, comme pour l'approche « primale », les espaces de traces discrètes :

$$\begin{aligned} W_h^1 &= \left\{u \in L^2(\Gamma),\ \exists u_1 \in X_h^1,\ u = \gamma_{1,h}(u_1)\right\} \\ W_h^2 &= \left\{u \in L^2(\Gamma),\ \exists u_2 \in X_h^2,\ u = \gamma_{2,h}(u_2)\right\} \end{aligned}$$

ainsi que l'opérateur $\pi_{1,h}$ de projection au sens de L^2 sur W_h^1. En choisissant nos multiplicateurs de LAGRANGE dans l'espace dual de W_h^1, que nous noterons $W_h^{1'}$, on définit la forme bilinéaire discrète :

$$\begin{aligned} b_h \ : X_h \times W_h^{1'} \ &\longrightarrow \mathbb{R} \\ (v, \psi) \quad &\longmapsto b_h(v, \psi) = \left\langle \psi, \gamma_{1,h}(v_1) - \pi_{1,h}\left(\gamma_{2,h}(v_2)\right)\right\rangle_{W_h^{1'}, W_h^1} \end{aligned}$$

Notre problème discret s'écrit alors :

Problème 2.8 : *Trouver* $(u, \lambda) \in X_h \times W_h^{1'}$, *tel que :*

$$\begin{aligned} a_h(u, v) + b_h(v, \lambda) &= l_h(v) \qquad \forall v \in X_h \\ b_h(u, \mu) &= 0 \qquad \forall \mu \in W_h^{1'} \end{aligned}$$

F. BEN BELGACEM démontre dans [18] que ce problème est bien posé, et il explicite la forme des espaces X_h et $W_h^{1'}$ dans le cas où l'on utilise une méthode d'éléments finis pour discrétiser les sous–problèmes.

2.3.3 Les difficultés principales

Nous venons de le voir, la méthode des éléments joints est extrêmement générale. Elle permet de coupler des sous–problèmes discrétisés à l'aide de méthodes numériques différentes, sur des maillages qui n'ont pas besoin d'être conformes au niveau des interfaces. Il reste cependant que l'utilisation de cette méthode dans le cas de problèmes tridimensionnels pose quelques difficultés techniques. En effet, dans un tel cas, les interfaces entre problèmes se trouvent être des surfaces, et les opérations de projection des traces peuvent s'avérer à la fois coûteuses et peu précises.

Cependant, imposer des contraintes de raccordement reste l'opération la plus difficile de cette méthode. En effet, si l'on utilise l'approche primale, il est nécessaire de construire une base de l'espace X_h qui satisfasse ces conditions de raccord. Or construire une telle base peut s'avérer très difficile, en particulier s'il existe des nœuds partagés par plus de deux sous–domaines, situation très fréquente dès que l'on fait apparaître plus de deux sous–domaines. Si l'on utilise l'approche duale proposée par F. Ben Belgacem, ces difficultés liées aux nœuds multiples disparaissent. Cependant, dans le cas d'un problème de Laplace, cette formulation conduit à la résolution de sous–problèmes avec des conditions aux limites de Neumann sur les interfaces. S'il existe un sous–domaine Ω_i tel que $\partial\Omega_i \cap \partial\Omega = \emptyset$, alors ce sous–problème, pris isolément, est mal posé, car il admet une infinité de solutions. Plus exactement, le formalisme de F. Ben Belgacem conduit à l'élaboration d'un problème global bien posé, bien que les sous–problèmes locaux puissent être séparément mal posés. Cette difficulté peut s'avérer importante lorsque l'on cherche à préconditionner la résolution du problème. Nous reviendrons sur ce point au chapitre 4, et nous proposerons une modification de la formulation variationnelle du problème de façon à obtenir à la fois un problème global et des problèmes locaux n'admettant qu'une unique solution dépendant continûment des données.

2.4 Architectures pour le calcul parallèle

Nous abordons maintenant un tout autre sujet. Après avoir détaillé un outil mathématique qui nous servira à concevoir notre méthode numérique, nous développons à présent les concepts et outils liés à son contexte d'exécution. Nous souhaitons en effet mettre en œuvre une méthode capable d'exploiter les ressources d'une machine massivement parallèle à mémoire distribuée, de façon à pouvoir réaliser des simulations à caractère industriel. Nous présentons donc dans les paragraphes qui suivent les machines disponibles pour réaliser ce genre de calcul, ainsi que les modèles et techniques d'implémentation qui permettent de les utiliser.

Commençons par étudier la question de l'architecture matérielle des machines qui nous intéressent. Nous adoptons ici les idées présentées par M.-J. Flynn dans [57, 58] afin de différencier les diverses familles de machines de calcul. Ce dernier utilise deux critères pour caractériser le comportement d'une machine :

- le nombre de flots de données, noté n_{fd} ;
- le nombre de flots d'instructions, noté n_{fi}.

La dénomination utilisée par M.-J. Flynn pour identifier différentes familles de machines est la suivante :

- SISD : Single Instruction, Single Data $\left(n_{fd} = n_{fi} = 1\right)$;

– SIMD : Single Instruction, Multiple Data $\left(n_{fi}=1,\ n_{fd}>1\right)$;
– MISD : Multiple Instruction, Single Data $\left(n_{fi}>1,\ n_{fd}=1\right)$;
– MIMD : Multiple Instruction, Multiple Data $\left(n_{fi}>1,\ n_{fd}>1\right)$.

La catégorie SISD rassemble les machines ne présentant qu'une unité de traitement capable d'exécuter une instruction sur une donnée à la fois. Il s'agit des ordinateurs monoprocesseurs classiques, connus du grand public depuis de nombreuses années (bien que le marché actuel tend à populariser les machines reposant sur plusieurs unités de traitement). Soulignons que les machines SISD présentent toutefois une certaine forme de parallélisme, ou plus exactement de concurrence, sous la forme de pipelines de traitement câblés dans les processeurs.

La famille MISD fait référence aux machines présentant de multiples unités de traitement enchainant leurs effets sur un même flot de données. Il s'agit en quelque sorte d'un gigantesque pipeline de processeurs, chacun envoyant son résultat à l'entrée du suivant. Les machines de cette catégorie présentent une architecture dite systolique [111].

La catégorie SIMD regroupe les machines présentant plusieurs unités de traitement, mais une unique unité de contrôle du flot d'instruction. Ainsi, toutes les unités de traitement exécutent la même instruction, mais chacune utilise son propre flot de données. On trouve par exemple dans cette famille de machines les processeurs vectoriels tel que le fameux Cray–1™, construit en 1976 par Seymour CRAY.

Enfin, la famille MIMD rassemble les machines possédant de multiples unités de traitement, chacune exécutant son propre flot d'instructions en utilisant son propre flot de données. Nous pouvons distinguer trois sous–familles importantes dans la famille MIMD :

– les multiprocesseurs ;
– les multicalculateurs ;
– les machines hybrides.

Ces sous–familles se distinguent par la nature des relations entre unités de calcul (processeurs) et unités de stockage (mémoire). Les multiprocesseurs sont des machines à mémoire partagée : ils sont composés de multiples processeurs ayant accès à une unique mémoire commune. Cette architecture est illustrée sur la figure 2.9.

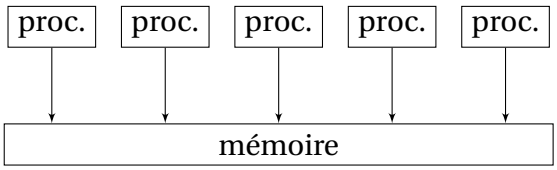


FIG. 2.9 : Architecture d'un multiprocesseur

Les multicalculateurs, également appelés « machines à mémoire distribuée », sont composés de plusieurs processeurs reliés par un réseau, chaque processeur disposant de son propre espace mémoire. Cette architecture est présentée sur la figure 2.10.

Enfin, les machines hybrides présentent une architecture à plusieurs niveaux. Elles sont en effet composées de plusieurs nœuds interconnectés par un réseau, chaque nœud étant lui–même constitué de plusieurs unités de calcul partageant une mémoire commune. La figure 2.11 présente cette architecture qui tend aujourd'hui à devenir la plus répandue.

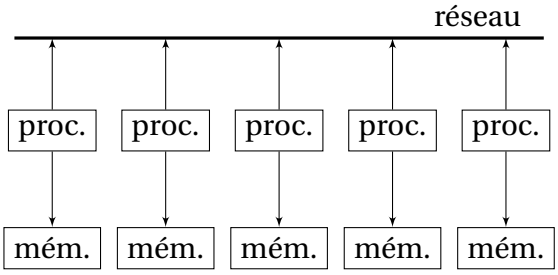


FIG. 2.10 : Architecture d'un multicalculateur

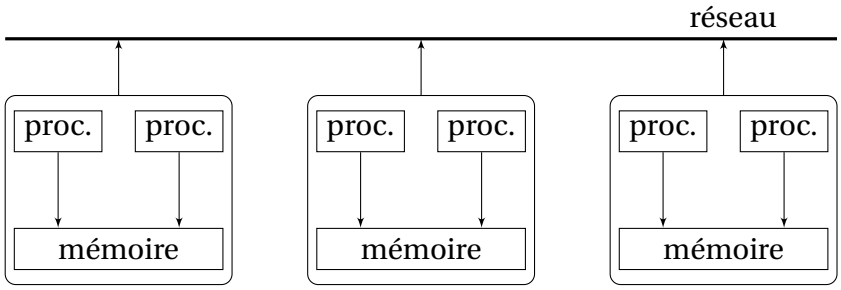


FIG. 2.11 : Architecture d'une machine hybride

Dans le cadre de notre travail, nous nous intéresserons principalement aux machines à mémoire distribuées et aux machines hybrides. Les raisons de ce choix sont les suivantes. D'une part, il s'agit à l'heure actuelle des architectures les plus utilisées pour réaliser des calculs de type industriel. Et d'autre part, Trio-U a précisément été développé pour pouvoir être exécuté sur ce type de machines. Présentons donc maintenant les méthodes et outils qui permettent d'exploiter de telles machines.

2.5 Programmation des machines parallèles

2.5.1 Modèles de programmation parallèle

La conception et la programmation d'algorithmes parallèles peuvent être abordées de différentes manières. En effet, il existe différents modèles de programmation, chacun présentant ses avantages et inconvénients au regard des programmes à implémenter et des machines ciblées. Ces modèles se distinguent essentiellement par la structure des algorithmes exécutés par les différents processeurs au cours de l'exécution.

Par exemple, dans le modèle Single Program, Multiple Data (SPMD), tous les processeurs exécutent la même séquence d'instructions sur les données qui leurs ont été attribuées (voir [47]). On retrouve ainsi le comportement d'une machine SIMD sur une machine MIMD, et on parle alors de parallélisme de données. Ce modèle repose uniquement sur des communications globales et synchrones entre processeurs. Son principal intérêt est de simplifier la programmation des algorithmes, car il n'y a qu'une seule séquence d'instruction à programmer. Il est souvent utilisé dans les bibliothèques de solveurs d'algèbre linéaire, où les instructions à exécuter sont généralement uniformes.

On peut également utiliser un modèle de programmation dit Multiple Program, Multiple Data (MPMD), dans lequel chaque processeur exécute sa propre séquence d'instructions sur son propre flot de données (voir [61, 110]). On parle alors de parallélisme de tâches. Il s'agit typiquement du modèle de programmation retenu lorsque l'on utilise une méthode de décomposition de domaine. En effet, avec ce genre de méthode, chaque processeur se voit confier la résolution d'un problème local qui est indépendant (au moins lors de la résolution) des autres sous–problèmes. Ce mode de programmation est sensiblement plus compliqué à mettre en œuvre que le modèle SPMD, mais il offre en contrepartie beaucoup plus de flexibilité. Il est en effet possible de considérer des schémas de communication plus fins, ne faisant intervenir que quelques processeurs. Il faut donc prêter une attention particulière à la bonne synchronisation des processeurs pour éviter tout blocage à l'exécution.

Un dernier exemple de modèle de programmation d'algorithmes parallèles est le modèle BSP, ce qui signifie Bulk Synchronous Parallelism (voir [26, 135]). Il s'agit en quelque sorte d'une approche hybride entre les modèles SPMD et MPMD. En effet, le modèle BSP consiste à utiliser une alternance de phases de calcul et de phases de communications, les communications étant synchronisées sur l'ensemble des processeurs. Ainsi, on utilise un modèle MPMD lors de la phase de calcul, car les processeurs peuvent exécuter des instructions différentes sur des données différentes. À l'inverse, les communications sont nécessairement globales et synchrones, comme dans le cas du paradigme SPMD. Ce modèle est relativement simple à utiliser, grâce notamment à la synchronisation des communications. Par ailleurs, à l'instar du modèle SPMD, mais contrairement au modèle MPMD, il permet d'évaluer facilement la complexité algorithmique du programme en utilisant un modèle simple pour calculer le coût des communications.

2.5.2 Techniques d'implémentation d'algorithmes parallèles

La programmation effective d'un algorithme sur une machine parallèle peut être réalisée à l'aide de différents outils. Comme nous allons le voir, certains de ces outils sont particulièrement adaptés à un type de machine donné, alors que d'autre sont plus généraux et permettent d'utiliser plusieurs modèles de programmation sur différents types de machines.

Ainsi, il existe essentiellement deux outils permettant d'implémenter un programme parallèle destiné à être exécuté sur un multiprocesseur. Le premier repose sur la notion de processus légers, aussi appelés threads selon la terminologie anglo–saxonne. Le standard POSIX [31, 75] définit une spécification précise de l'interface de programmation de ces processus légers, l'implémentation étant généralement réalisée par le concepteur du système d'exploitation de la machine. Dans ce modèle, l'accès à la mémoire est partagé, et les communications sont réalisées par l'utilisation explicite de variables communes dont l'accès est règlementé par l'introduction de verrous. Le second de ces outils est fourni par le standard OpenMP [39, 46, 122], qui définit un ensemble de procédures et de directives de compilation facilitant la parallélisation d'un code séquentiel sur ce genre de machines. Une différence essentielle entre ces deux outils se situe au niveau du mode de compilation du programme. En effet, l'approche par processus légers est généralement mise en œuvre par liaison avec une librairie binaire, alors que l'utilisation du standard OpenMP repose sur l'appel à un compilateur spécifique capable d'interpréter les directives de parallélisation. Ces directives permettent de spécifier la distribution des

données sur les processeurs et de paralléliser l'exécution des itérations sur les données, chaque processeur n'itérant que sur les données qui lui ont été attribuées.

Pour la programmation sur multicalculateur, l'approche usuelle consiste à développer le programme à l'aide d'un modèle de passage de messages tel que celui proposé par le standard MPI [70, 71, 99, 100]. Selon ce modèle, chaque unité de calcul est perçue comme un processeur pouvant envoyer (resp. recevoir) des messages à destination (resp. en provenance) des autres processeurs. Pour cela, le standard MPI fournit de nombreuses fonctions de communication permettant l'élaboration de schémas de communication très variés : communications point–à–point ou globales, communications synchrones ou asynchrones, communications bloquantes ou non–bloquantes, etc. Nous renvoyons à [110] ainsi qu'à la spécification du standard MPI pour le détail des possibilités offertes. La flexibilité de ce système est à l'origine de sa portabilité et de sa popularité. Il est en effet possible de l'utiliser sur la quasi–totalité des machines parallèles, et les constructeurs fournissent généralement une implémentation optimisée de ce standard pour chacune de leurs machines. Notons que bien que la sémantique des fonctions définies dans le standard MPI soit particulièrement adaptée au modèle d'exécution des multicalculateurs, il est tout–à–fait possible de développer un programme parallèle utilisant MPI pour l'exécuter ensuite sur un multiprocesseur. L'efficacité du programme dépendra alors de la qualité de l'implémentation du standard sur cette machine.

Enfin, si l'on désire utiliser le modèle de programmation BSP évoqué plus haut, il existe essentiellement deux librairies fournissant une interface de programmation adaptée à sa mise en œuvre. Il s'agit de la BSPlib [73, 74] et de la Paderborn University BSP Library [28, 29]. Ces librairies proposent chacune une petite collection de fonctions de communication et de synchronisation de haut niveau reflétant les fonctionnalités requises par le modèle BSP. L'implémentation de ces fonctions reposent sur le standard MPI, ce qui assure leur portabilité.

2.5.3 Le parallélisme dans Trio-U

Nous présentons maintenant la perception et la mise en œuvre du parallélisme dans la plate–forme Trio-U. Il s'agit d'une conception relativement simple et classique, reposant sur un modèle SPMD et à destination d'une machine de type MIMD ou hybride. Présentons tout d'abord comment est réalisée la distribution des données sur les processeurs. Partant d'un maillage du domaine de calcul, celui–ci est dans un premier temps partitionné en n sous–domaines, n désignant le nombre de processeurs utilisés pour le calcul. Ce partitionnement est effectué sur la base de deux critères :

- le nombre de degrés de liberté doit être équilibré entre les sous–domaines ;
- le nombre de degrés de liberté partagés par plusieurs sous–domaines doit être minimal.

Si l'on suppose que la machine utilisée est homogène, c.-à-d. les processeurs sont identiques, le premier critère assure que la charge de calcul et le volume de données sont distribués équitablement sur ces derniers. Le second critère permet de minimiser le volume de communications inter–processeurs lors de la résolution du problème. Une fois le domaine découpé, il est distribué sur les processeurs de la machine. Chaque processeur construit alors la partie du système linéaire qui correspond au sous–domaine qui lui a été affecté. À chaque étape de la résolution, un processeur aura à mettre à jour les valeurs des degrés de libertés qui lui ont été affectés. Comme il existe des degrés de liberté partagés

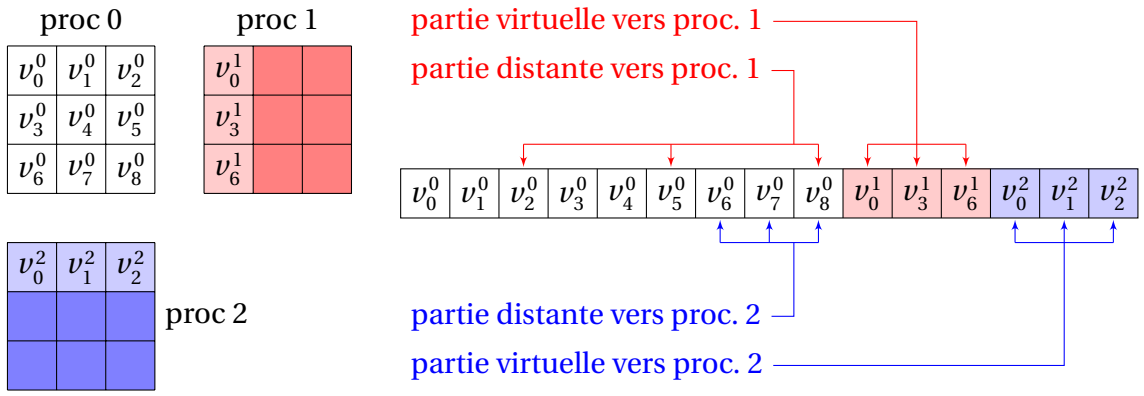


FIG. 2.12 : Illustration de la notion de vecteur distribué de Trio-U

entre les processeurs, ces derniers nécessiteront un traitement particulier. Plus exactement, la mise à jour de ces inconnues se fera à l'aide d'une phase de communication inter–processeurs. Afin de simplifier le traitement de ces degrés de liberté particuliers, Trio-U introduit la notion de *vecteur distribué*.

D'un point de vue global, un vecteur distribué représente un tableau de valeurs distribué sur les processeurs. Du point de vue d'un processeur, il s'agit d'un tableau présentant plusieurs parties :

- une partie dite « réelle » ;
- une liste de parties dites « virtuelles » ;
- une liste de parties dites « distantes ».

La figure 2.12 illustre cette structure. La partie réelle d'un vecteur distribué correspond aux données du tableau que le processeur possède en propre : c'est lui qui aura la charge de calculer les valeurs contenues dans cette partie du tableau au cours de la résolution. Une partie virtuelle correspond à des données qui sont situées sur un autre processeur, mais dont le processeur a besoin pour effectuer ses calculs. Bien entendu, un processeur peut avoir à échanger des données avec plusieurs autres processeurs, d'où la nécessité d'avoir une liste de parties virtuelles. Une partie distante est la transposée d'une partie virtuelle : il s'agit des données que possède le processeur en question, mais dont un autre processeur a besoin pour pouvoir effectuer ses calculs. La structure de ces vecteurs est déterminée une fois pour toutes à l'initialisation du calcul, et dépend essentiellement du découpage du maillage. Une fois cette structure établie, chacun des processeurs est à même de connaître les processeurs avec lesquels il devra communiquer. Partant de cette information, Trio-U optimise l'ordonnancement des communications de façon à éviter de saturer la bande passante du réseau. En pratique, l'implémentation de ces concepts repose sur le standard MPI évoqué dans le paragraphe ci–dessus. Ce choix permet d'assurer la portabilité de Trio-U sur différentes architectures, car ce standard est aujourd'hui bien répandu et généralement implémenté sur la plupart des machines parallèles.

Résumé

Nous avons présenté dans ce chapitre deux notions essentielles à nos travaux, à savoir les méthodes de décomposition de domaine et le parallélisme. La première constitue le socle

mathématique de la méthode adaptative que nous nous proposons d'élaborer. Nous avons à cette occasion insisté sur les propriétés de la méthode des éléments joints, que nous chercherons à adapter à nos besoins. Nous développerons ces concepts au chapitre 4.

La seconde notion abordée présente le contexte d'exécution dans lequel nous souhaitons mettre en œuvre notre méthode. À cet effet, nous avons décrit les machines ciblées, ainsi que les modalités de programmation pour de telles machines. Nous avons également mis en exergue les points essentiels qui permettent d'assurer les bonnes performances d'un code parallèle, à savoir l'équilibrage de la charge de calcul sur les processeurs de la machine, et la minimisation du coût des communications inter–processeurs. Nous présenterons au chapitre 5 les idées que nous retenons pour satisfaire ces deux critères.

CHAPITRE 3

Choix d'une méthode adaptative : aspects géométriques

Nous abordons maintenant la question de l'adaptation de maillage proprement dite. Notre objectif dans ce chapitre est d'analyser le comportement de différentes techniques d'adaptation de maillage, en vue d'établir un choix pour la méthode que nous envisageons. Pour cela, nous allons procéder en trois étapes. Dans un premier temps, nous présentons différentes familles de méthodes adaptatives, et nous justifions notre choix parmi ces dernières. Dans un second temps, nous étudions différentes méthodes appartenant à cette famille, et nous en détaillons le comportement. À la lumière de cette analyse, et en prenant en compte la localisation du raffinement, nous arrêtons notre choix en matière d'adaptation géométrique. Pour compléter cette étude, nous proposons en fin de chapitre un algorithme adapté à ce choix qui permet de prendre en compte convenablement tant le raffinement que le déraffinement des éléments du maillage.

3.1 Techniques d'adaptation de discrétisation

Les méthodes adaptatives ayant connu un essort assez important ces dernières années dans le monde de la recherche académique, la littérature concernant les techniques d'adaptation de maillage est extrêmement riche. Nous nous proposons ici d'effectuer une revue de ces techniques, en les regroupant selon une classification désormais devenue classique. Afin de motiver notre choix pour l'une ou l'autre de ces méthodes, nous retenons les critères suivants :

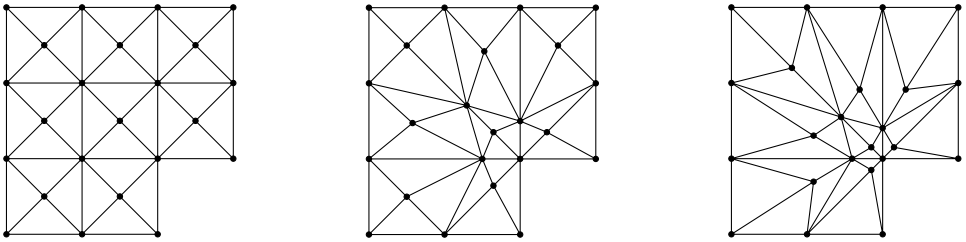
- qualité des maillages générés ;
- complexité algorithmique ;
- simplicité d'implémentation.

Nous détaillerons donc les intérêts et les inconvénients de chacune de ces méthodes au regard de ces critères.

3.1.1 Les r–méthodes

Une *r-méthode* est une méthode adaptative de déformation de maillage. Le principe retenu consiste à déplacer les nœuds du maillage tout en conservant la connectivité de ce dernier. De cette façon, on cherche à augmenter ou diminuer la densité des nœuds

dans certaines régions du domaine, afin de les concentrer dans les zones d'intérêt et de les espacer dans les régions moins-critiques. La figure 3.1 représente l'évolution typique d'un maillage selon ce type de méthode. En conservant la connectivité du maillage, on



(a) Maillage initial (b) Première adaptation (c) Maillage adapté

FIG. 3.1 : Évolution d'un maillage lors de l'utilisation d'une r–méthode

espère ne pas augmenter le coût de la simulation, car la dimension du problème discrétisé reste constante. Toutefois, comme on peut le voir sur la figure 3.1, le processus d'adaptation du maillage fait généralement apparaître des éléments très étirés. Or ce type d'éléments nuit singulièrement au conditionnement du système, paramètre prépondérant pour la vitesse de convergence des algorithmes itératifs d'algèbre linéaire. On observe donc en général une baisse sensible des performances quand la déformation du maillage est trop importante. De plus, les r–méthodes font nécessairement appel à un certain nombre d'interpolations, qui peuvent elles aussi nuire à la qualité de la solution numérique. Des travaux récents [86, 87, 140, 141, 151] présentent toutefois des développements tout–à–fait intéressants de cette méthode, en illustrant comment exploiter les propriétés de régularisation des problèmes étudiés pour générer des maillages adaptés de bonne qualité.

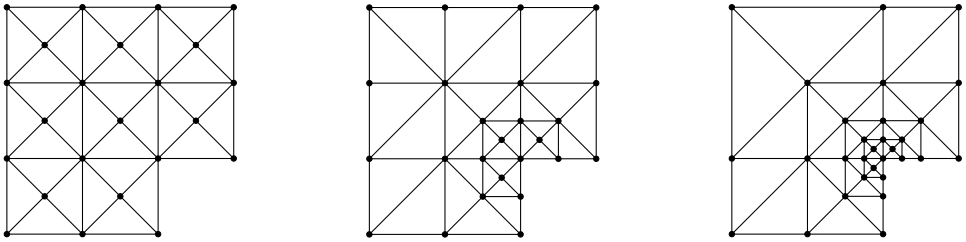
Du point de vue du parallélisme, les r–méthodes constituent un cas très particulier. En effet, comme la connectivité du maillage est conservée au cours de la résolution, les r–méthodes n'introduisent *a priori* pas de déséquilibre de charge, et ne modifient pas les schémas de communications. Cependant, les perturbations du conditionnement du système dues aux modifications géométriques des éléments du maillages introduisent un déséquilibre de charge, qui se révèle plus particulièrement lors du préconditionnement du problème. Ce déséquilibre restant toutefois marginal, il suffit généralement que le partitionnement initial soit pertinent pour garantir l'efficacité de la simulation dans un contexte parallèle.

Toutefois, l'hypothèse fondatrice des r–méthodes s'avère très restrictive. En effet, assurer la conservation du nombre de degrés de liberté du problème discret limite sérieusement les capacités de ces méthodes en termes d'amélioration de l'approximation. Cette faiblesse intrinsèque nous conduit à ne pas choisir ce type d'approche pour notre méthode adaptative.

3.1.2 Les h–méthodes

Les *h–méthodes* envisagent l'adaptation de maillage selon un angle différent. Là où les r–méthodes cherchent à optimiser la position des nœuds du maillage en conservant sa connectivité, le principe des h–méthodes consiste à modifier localement le maillage

en le raffinant ou en le déraffinant selon les besoins. Comme on peut le voir sur la figure 3.2, cette adaptation est réalisée soit par l'introduction de nouveaux sommets et de nouveaux éléments dans le maillage, soit par la suppression de sommets et d'éléments existants. Bien entendu, un tel procédé modifie la connectivité du maillage, ainsi que le



(a) Maillage initial (b) Première adaptation (c) Maillage adapté

FIG. 3.2 : Évolution d'un maillage lors de l'utilisation d'une h–méthode

nombre de degrés de liberté du problème discret. L'intérêt principal de ce choix est qu'il permet de modifier le maillage en contrôlant la qualité des éléments générés. Ainsi, cette démarche permet d'assurer que l'adaptation du maillage ne nuit pas au conditionnement du problème discret.

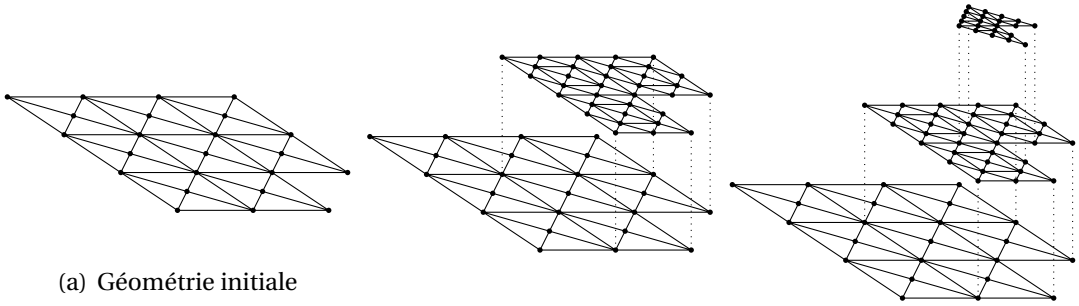
L'implémentation parallèle d'une h–méthode soulève deux questions primordiales. La première concerne le caractère local des opérations de modifications du maillage. On dira qu'une telle modification est locale si elle n'affecte qu'un nombre fini et petit d'éléments voisins. L'utilisation de modifications locales est particulièrement important pour l'implémentation parallèle d'une h–méthode, afin que l'algorithme d'adaptation du maillage n'engendre pas de communications exhorbitantes. La seconde question se rapporte au déséquilibre de charge qu'une h–méthode génère de façon presque inévitable. En effet, on conçoit aisément que, suite à un déraffinement, un processeur puisse voir décroître le nombre de degrés de liberté qui lui a été affecté, alors qu'un autre processeur voie le sien augmenter parce qu'il effectue un raffinement. Ainsi, il est nécessaire de mettre en place un algorithme de rééquilibrage de la charge, qui compense ce phénomène en redistribuant les éléments du maillage après chaque modification.

3.1.3 Les méthodes multi–niveaux

Les *méthodes multi–niveaux* constituent une troisième famille de méthodes numériques adaptatives. Alors que les méthodes précédentes n'utilisent qu'un seul maillage qu'elles modifient au cours de la simulation, les méthodes multi–niveaux définissent une hiérarchie de maillages, engendrée par les raffinements successifs des zones de d'intérêt (voir figure 3.3), selon un patron de raffinement donné [117–119]. Ces méthodes sont également appelées *méthodes de zoom* ou *méthodes multi–grilles*. Cette dernière appellation est cependant ambiguë, car elle est utilisée à la fois dans le contexte des méthodes adaptatives[1], et dans celui des solveurs linéaires[2]. Les méthodes multi–niveaux ont été explorées dans les années 1980 [19, 20, 76] et ont été mises en œuvre dans les codes CHOMBO [43] et AGRIF [49, 50] pour les discrétisations structurées, et dans les codes

[1] On parle alors d'une méthode multi–grille géométrique.

[2] On parle dans ce cas d'une approche multi–grille algébrique.



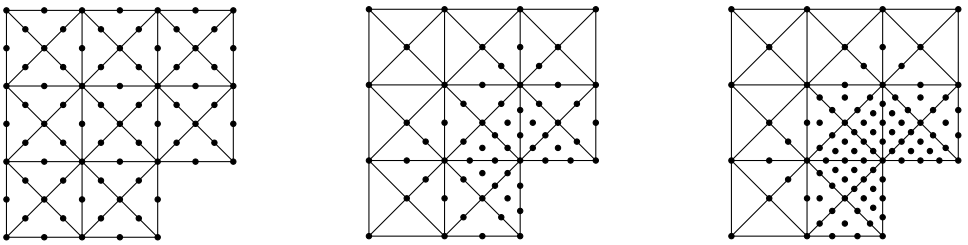
(a) Géométrie initiale (b) Géométrie intermédiaire (c) Géométrie adaptée

FIG. 3.3 : Évolution de la discrétisation lors de l'utilisation d'une méthode multi–niveaux

PLTMG [11], PHAML [101–105] et UG [13, 14] pour les discrétisations non–structurées. Des extensions ont été proposées pour traiter le cas des équations de Navier–Stokes incompressibles, comme par exemple la méthode *Fast Adaptive Composite* [98] ou la méthode *Flux Interface Correction* [44, 85, 146]. Le couplage entre les problèmes posés sur les différents niveaux de grilles peut être réalisé de plusieurs façons. Ces problèmes peuvent être de nature différentes, ce qui offre une grande liberté. Du point de vue du parallélisme, les méthodes multi-niveaux présentent un comportement proche de celui des h–méthodes, en génèrant des déséquilibres de charge pouvant s'avérer problématiques. Cependant, alors que les h–méthodes reposent généralement sur un parallélisme de données, la nature intrinsèquement hiérarchique des méthodes multi-niveaux conduit naturellement à l'utilisation d'un parallélisme de tâches. Cette perception du parallélisme peut être exploitée lors de l'implémentation d'algorithmes d'équilibrage dynamique de la charge.

3.1.4 Les p–méthodes

Alors que les méthodes précédemment citées modifient la discrétisation géométrique du domaine de calcul (c.-à-d. le maillage), les *p–méthodes* opèrent au niveau de la méthode numérique elle–même. Développées à l'origine pour des méthodes de type éléments finis, les p–méthodes consistent à adapter le degré des fonctions d'interpolation locales selon la régularité de la solution. La figure 3.4 présente ce principe, dans le cadre d'éléments finis $\mathbb{P}^k$ de LAGRANGE (voir [54]). L'intérêt majeur de cette approche est que



(a) Discrétisation initiale (b) Première adaptation (c) Discrétisation adaptée

FIG. 3.4 : Évolution de la discrétisation lors de l'utilisation d'une p–méthode

le maillage est conservé lors de l'adaptation de la discrétisation. On réalise ainsi, en regard des h–méthodes, un gain non–négligeable en terme d'espace mémoire, car seul l'algorithme d'assemblage du système linéaire doit être modifié. Toutefois, cette approche présente l'inconvénient majeur de ne pas être générique. Elle reste en effet difficilement transposable à d'autres types de méthodes numériques comme les volumes finis, par exemple. De la même façon, il faut prendre quelques précautions si l'on veut utiliser cette méthode avec des éléments finis particuliers, comme par exemple l'élément fini de CROUZEIX–RAVIART. En effet, en dimension 2, les éléments finis de CROUZEIX–RAVIART de degré pair ne présentent pas de caractère unisolvant [54], et il est nécessaire de modifier leur définition pour pouvoir les utiliser (par exemple en introduisant les fonctions bulles de FORTIN–SOULIÉ [59]). Notons qu'il existe une approche hybride, dite *hp–méthode*, qui repose à la fois sur l'adaptation de maillage des h–méthodes et sur la modification des espaces d'approximation des p–méthodes (voir par exemple [7, 130, 139]). Cette approche reste cependant très complexe à mettre en œuvre, sans toutefois lever la difficulté présentée ci–dessus.

3.1.5 Premiers éléments de choix

Soulignons tout d'abord que si les p–méthodes sont tout–à–fait attrayantes de part leurs fondations mathématiques, elles n'en restent pas moins très difficiles d'accès dans la perspective d'un code industriel. Ceci est particulièrement vrai dans le cas de Trio-U, car la méthode VEF de ce dernier repose sur l'élément fini de CROUZEIX–RAVIART, pour lequel les p–méthodes s'avèrent difficilement exploitables. Par ailleurs, l'intérêt des r-méthodes semble relativement limité. En effet, bien que leur implémentation parallèle ne présente pas de réelle difficulté, les problèmes qui se posent alors étant essentiellement identiques à ceux soulevés par l'implémentation parallèle d'un code non–adaptatif, les r–méthodes s'avèrent quelque peu limitantes en termes de précision, étant donné la contrainte de conservation de la connectivité du maillage. Restent les h–méthodes et les méthodes multi–niveaux. Ces méthodes présentent de nombreux points communs, notamment pour ce qui est de la définition du raffinement d'un ensemble d'éléments présents dans le maillage. Nous allons maintenant développer ces idées afin de déterminer complètement notre approche adaptative.

3.2 Techniques de raffinement de maillage

Qu'il s'agisse d'une h–méthode ou d'une approche multi–niveaux, il est nécessaire de définir une procédure de modification des éléments du maillage. Dans cette section, nous présentons les techniques les plus souvent retenues pour définir le raffinement des éléments d'un maillage. Afin de simplifier l'exposé, nous supposons pour l'instant que nous souhaitons raffiner l'intégralité de ce maillage. Cette hypothèse sera levée par la suite dans la section 3.3. Nous nous restreignons aux cas de maillages de triangles en dimension deux et de tétraèdres en dimension trois, car il s'agit des éléments utilisés comme support de la méthode VEF de Trio-U.

3.2.1 Définitions

Nous commençons par définir ici quelques notions de base, essentielles à la présentation des méthodes d'adaptation de maillage.

Définition 3.1 : *Un sous–ensemble fermé K de $\mathbb{R}^n$ est un simplexe d'ordre k, si $0 \leq k \leq n$, et s'il s'agit de l'enveloppe convexe de $(k+1)$ points notés $(x_i)_{0\leq i\leq k}$ de $\mathbb{R}^n$:*

$$K = [x_i]_{0\leq i\leq k} \equiv \left\{ x \in \mathbb{R}^n, x = \sum_{i=0}^{k} \lambda_i x_i \ ; \ \forall i \in [\![0,k]\!], \lambda_i \in [0,1], \sum_{i=0}^{k} \lambda_i = 1 \right\}$$

Si $k = n$, on parlera plus simplement de simplexes de $\mathbb{R}^n$. Ainsi, les simplexes de $\mathbb{R}^2$ (resp. $\mathbb{R}^3$) sont des triangles (resp. tétraèdres).

Définition 3.2 : *On dira que deux simplexes $K = [x_i]_{0\leq i\leq k}$ et $L = [y_i]_{0\leq i\leq k}$ d'ordre k sont égaux, ce que l'on note $K = L$, si l'on a :*

$$\forall i \in [\![0,k]\!], \qquad x_i = y_i$$

Dans cette relation, l'ordre d'apparence des sommets dans la définition de K et de L est de première importance. À l'inverse, la définition suivante est indépendante de l'ordre d'apparence des sommets dans la définition de K et de L.

Définition 3.3 : *On dira que deux simplexes $K = [x_i]_{0\leq i\leq k}$ et $L = [y_i]_{0\leq i\leq k}$ d'ordre k sont égaux au sens des ensembles , ce que l'on note $K \simeq L$, s'ils désignent chacun le même sous–ensemble de $\mathbb{R}^n$.*

Définition 3.4 : *Un simplexe $L = [y_i]_{0\leq i\leq l}$ d'ordre l est un sous–simplexe d'ordre l d'un simplexe $K = [x_i]_{0\leq i\leq k}$ d'ordre k si l vérifie $0 \leq l < k \leq n$, et s'il existe une suite croissante d'indices $\left(i_j\right)_{0\leq j\leq l}$ telle que $\forall j \in [\![0,l]\!]$, $y_j = x_{i_j}$.*

Les sous–simplexes d'ordre 0 et 1 d'un simplexe d'ordre k de $\mathbb{R}^n$ sont respectivement ses sommets et ses arêtes. Un simplexe K d'ordre k possède exactement $N_{k,l} = \binom{k+1}{l+1}$ sous–simplexes d'ordre l, que l'on note $(\partial_l K_i)_{0\leq i\leq N_{k,l}}$.

Définition 3.5 : *Un simplexe K d'ordre k est dit dégénéré si ses sommets appartiennent tous à un hyperplan de dimension strictement inférieure à k, ou de façon équivalente, si les k vecteurs $(\boldsymbol{u}_i)_{1\leq i\leq k}$ définis par :*

$$\forall i \in [\![1,k]\!], \qquad \boldsymbol{u}_i = x_i - x_0$$

sont linéairement dépendants.

Définition 3.6 : *La dégénérescence d'un simplexe K d'ordre k, notée $\delta(K)$, est définie par :*

$$\delta(K) = \frac{h(K)}{\rho(K)} = \frac{h(K)}{2n\nu_k(K)} \left(\sum_{j=0}^{N_{k,k-1}} \nu_{k-1}\left(\partial_{k-1} K_j\right) \right)$$

où $h(K)$ désigne la longueur de la plus grande arête de K, $\rho(K)$ le diamètre de la plus grande boule de dimension k inscrite dans K, et $\nu_l(L)$ désigne le volume de dimension l du simplexe L d'ordre l. Il existe d'autres définitions possibles pour la mesure de la dégénérescence d'un simplexe, mais celle que nous retenons est particulièrement pertinente au regard des analyses de convergence de la méthodes des éléments finis.

Propriété 3.1 : *Soit K un simplexe d'ordre k de $\mathbb{R}^n$. Les trois propositions suivantes sont équivalentes :*

(i) *K est dégénéré ;*

(ii) $v_k(K)=0$;

(iii) $\delta(K)=\infty$.

Ainsi, dans le contexte de la discrétisation d'équations aux dérivées partielles, on cherchera à éviter les éléments présentant une grande dégénérescence, car ce sont ceux qui sont susceptible de nuire au conditionnement du système discret.

Définition 3.7 : *Deux simplexes K et L d'ordre k de $\mathbb{R}^n$ sont dits congruents, ce que l'on note $K \cong L$, s'il existe une similitude[3] $\mathscr{S}$ de $\mathbb{R}^n$ dans lui–même telle que $K=\mathscr{S}(L)$. Dans ce cas, on dit que K et L appartiennent à la même classe de congruence.*

Propriété 3.2 : *Soient K et L deux simplexes d'ordre k de $\mathbb{R}^n$. On a alors :*

$$K \cong L \quad \Longrightarrow \quad \delta(K)=\delta(L)$$

Démonstration. Si $K \cong L$, il existe une similitude $\mathscr{S}$ telle que $K=\mathscr{S}(L)$. Par définition, il existe donc une homothétie f et une isométrie g telles que $K=f\circ g(L)$. Posons $M=g(L)$. Comme g est une isométrie, il vient immédiatement : $\delta(M)=\delta(L)$. Par ailleurs, on a $K=f(M)$. En notant α le rapport d'homothétie de f, on a :

$$\begin{gathered} h(K)=\alpha h(M) \\ v_k(K)=\alpha^k v_k(M) \\ \sum_{j=0}^{N_{k,k-1}} v_{k-1}\left(\partial_{k-1}K_j\right)=\alpha^{k-1}\left(\sum_{j=0}^{N_{k,k-1}} v_{k-1}\left(\partial_{k-1}M_j\right)\right) \end{gathered}$$

Et il vient donc :

$$\delta(K)=\delta(M)=\delta(L)$$

□

Définition 3.8 : *Soit Ω un domaine polygonal borné de $\mathbb{R}^n$. Une triangulation, ou maillage simplicial, de Ω, notée $\mathscr{T}$, consiste en la donnée d'une famille de simplexes de $\mathbb{R}^n$ non–dégénérés et disjoints formant un recouvrement de Ω.*

Définition 3.9 : *La dégénérescence d'un maillage simplicial $\mathscr{T}$ est définie par :*

$$\delta(\mathscr{T})=\max_{K\in\mathscr{T}}\delta(K)$$

Définition 3.10 : *Une triangulation $\mathscr{T}$ est dite conforme si pour tout couple $(K,L)\in\mathscr{T}^2$, on a :*

$$\overline{K}\cap\overline{L}=\emptyset \quad \textit{ou bien} \quad \exists(l,i,j)\in[\![0,n]\!]\times[\![0,N_{k,l}]\!]\times[\![0,N_{k,l}]\!]\,, \quad \overline{K}\cap\overline{L}\simeq\partial_l K_i\simeq\partial_l L_j$$

Autrement dit, un maillage sera dit conforme si l'intersection de deux de ses simplexes est soit vide, soit égale au sens des ensembles à un sous–simplexe commun à ces deux simplexes.

[3] On rappelle qu'une similitude est la composée d'une homothétie et d'une isométrie.

Définition 3.11 : *Le raffinement d'un simplexe K d'ordre k consiste en la donnée d'une triangulation $\mathscr{R}(K)$ de K contenant au moins deux éléments.*

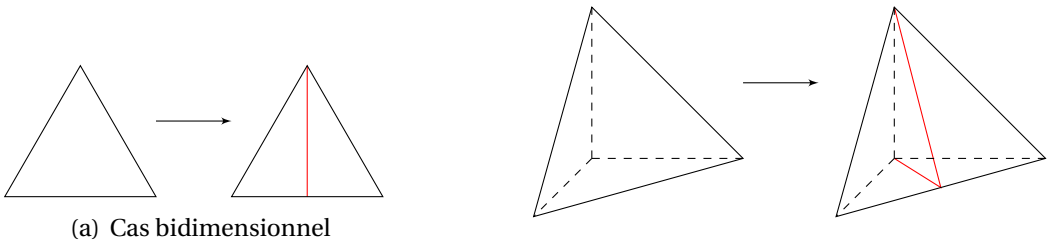
Définition 3.12 : *Soient $\mathscr{T}$ et $\mathscr{T}'$ deux triangulations d'un même domaine Ω. On dira que $\mathscr{T}'$ est un raffinement de $\mathscr{T}$ si, pour tout simplexe K de $\mathscr{T}$, ou bien K appartient aussi à $\mathscr{T}'$, ou bien il existe un raffinement $\mathscr{R}(K)$ de K tel que $\mathscr{R}(K) \subset \mathscr{T}'$.*

Définition 3.13 : *Une hiérarchie de triangulations $(\mathscr{T}_i)_i$ est une famille de triangulations telle que, pour tout i, $\mathscr{T}_{i+1}$ soit un raffinement de $\mathscr{T}_i$. Une telle famille de triangulations sera qualifiée de stable si la suite $(\delta(\mathscr{T}_i))_i$ est uniformément bornée.*

Nous introduisons maintenant les techniques les plus couramment utilisées pour définir une hiérarchie de triangulations à partir d'une triangulation initiale $\mathscr{T}_0$ donnée. Le principe de ces techniques consiste à définir un raffinement élémentaire, applicable à tout simplexe, et à définir la hiérarchie de façon itérative, chaque triangulation $\mathscr{T}_i$ étant générée à partir de la triangulation précédente $\mathscr{T}_{i-1}$. Nous verrons que ces techniques se fixent des priorités différentes, et qu'elles présentent donc chacune leurs avantages et leurs inconvénients.

3.2.2 Méthodes de bissection récursive

Le raffinement élémentaire des méthodes de bissection récursive consiste à découper un simplexe en deux simplexes, comme cela est illustré sur la figure 3.5. Différentes



(a) Cas bidimensionnel

(b) Cas tridimensionnel

FIG. 3.5 : Découpage élémentaire de la bissection récursive

méthodes relèvent de la bissection récursive. Elles se distinguent essentiellement au niveau du choix de l'arête de l'élément qui doit être découpée. La première méthode a été proposée par E.-G. SEWELL en 1979 [134] dans le cas bidimensionnel. Elle repose sur l'hypothèse que l'arête découpée doit toujours être l'arête opposée au dernier sommet créé dans l'élément. Une amélioration de cette approche a été présentée par W.-F. MITCHELL en 1991 [101], et elle a été étendue au cas tridimensionnel par E. BÄNSCH [32], J.-M.-L. MAUBACH [96, 97], C.-T. TRAXLER [143], A. LIU et B. JOE [91] et enfin D.-N. ARNOLD, A. MUKHERJEE et L. POULY [4]. Enfin, une autre approche a été proposée par M.-C. RIVARA en 1984 [117–119], qui consiste a toujours choisir la plus grande arête de l'élément. Une version tridimensionnelle a été exposée dans [120].

Supposons que la triangulation initiale $\mathscr{T}_0$ soit conforme. Le découpage de simplexes par bissection ne permet pas de conserver cette propriété. C'est pourquoi la construction de $\mathscr{T}_{i+1}$ à partir de $\mathscr{T}_i$ est en réalité plus complexe que la simple application du découpage élémentaire à chacun des éléments du maillage. En pratique, on utilise l'algorithme global illustré dans l'encadré 3.6. C'est le caractère récursif de cet algorithme qui donne son

```
Soit S_0 l'ensemble des éléments à raffiner
i = 0
while (S_i ≠ ∅) {
  découper chaque élément contenu dans S_i
  placer dans S_{i+1} les éléments présentant une non-conformité
}
```

FIG. 3.6 : Algorithme de bissection récursive

nom à cette famille de méthodes. En termes algorithmiques, il s'agit essentiellement d'un algorithme de fermeture transitive, qui permet avant toute chose d'assurer la conformité globale du maillage adapté. La figure 3.7 en présente le fonctionnement. Dans cette fi-

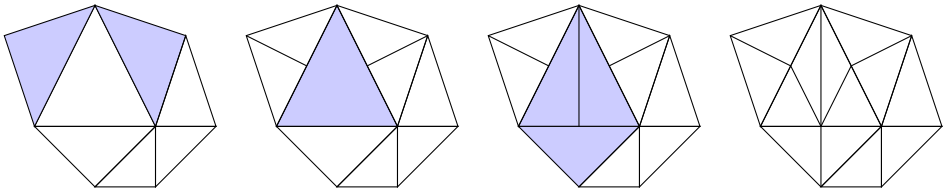


FIG. 3.7 : Application de l'algorithme de bissection récursive

gure, on a colorié les éléments appartenant aux ensembles $\mathscr{S}_i$ au début de chaque itération. En dépit de son caractère récursif, M.-C. RIVARA a pu montrer que l'algorithme de bissection récursive se termine en un nombre fini d'itérations.

On voit donc que l'objectif principal de la méthode de la bissection récursive est de construire un maillage adapté qui ne présente pas de non–conformité. Il reste cependant à vérifier que cette approche ne génère pas d'élément susceptible de nuire au conditionnement du système linéaire, autrement dit, que la hiérarchie de triangulations générée par plusieurs applications successives de cet algorithme est stable. En dimension deux, la propriété suivante, établie par M.-C. RIVARA, permet d'assurer la stabilité de la hiérarchie.

Propriété 3.3 : *Considérons Ω un domaine polygonal borné de $\mathbb{R}^2$, et $\mathscr{T}_0$ une triangulation conforme de Ω. L'application itérative de l'algorithme de bissection récursive fournit une hiérarchie de triangulations $(\mathscr{T}_i)_i$ vérifiant :*

(i) quelque soit i, $\mathscr{T}_i$ est une triangulation conforme de Ω ;

(ii) le nombre de classes de congruence des simplexes appartenant à l'une quelconque des triangulations $\mathscr{T}_i$ est borné indépendamment de i.

Corollaire 3.1 : *La hiérarchie de triangulations $(\mathscr{T}_i)_i$ d'un domaine Ω de $\mathbb{R}^2$ généré par une méthode de bissection récursive est stable.*

À notre connaissance, on ne dispose malheureusement pas de résultats aussi forts en dimension trois. En particulier, la stabilité de la version tridimensionnelle de la méthode de M.-C. RIVARA n'a à ce jour fait l'objet que de conjectures. Par contre, les méthodes de J.-M.-L. MAUBACH et C.-T. TRAXLER, qui s'avèrent être équivalentes, vérifient la propriété suivante :

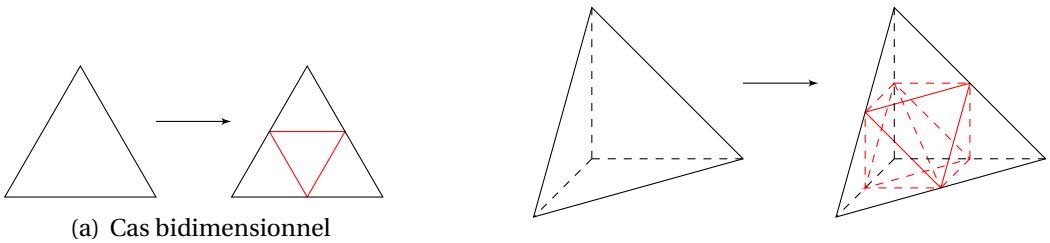
Propriété 3.4 : *Soit Ω un domaine polygonal borné de $\mathbb{R}^3$. Soit $\mathscr{T}_0$ une triangulation de Ω. On suppose que le nombre d'éléments adjacents à une même arête de la triangulation $\mathscr{T}_0$*

est toujours pair, quelle que soit l'arête considérée. Dans ce cas, la hiérarchie de triangulations générée par l'application itérative de la bissection récursive fondée sur l'approche de J.-M.-L. MAUBACH *ou de* C.-T. TRAXLER *est stable.*

La démonstration de cette propriété est assez technique, et nous renvoyons à [97] et [143] pour plus de détails. Nous voyons que les méthodes de bissection récursive s'avèrent relativement difficiles d'accès dans le cas tridimensionnel. En effet, les hypothèses nécessaires à l'établissement de la stabilité des hiérarchies générées semblent assez restrictives. Par ailleurs, bien qu'il soit relativement simple de vérifier si une triangulation satisfait ces hypothèses, il reste difficile de spécifier les corrections à apporter dans le cas où ces dernières ne sont pas satisfaites.

3.2.3 Méthodes de subdivision

La méthode de subdivision la plus connue est probablement celle introduite par R.-E. BANK, A.-H. SHERMAN et A. WEISER dans [12] dans le cas bidimensionnel. Les méthodes de subdivision reposent sur les découpages élémentaires de simplexes illustrés sur la figure 3.8. Ainsi, en dimension deux, un triangle est découpé en quatre sous–triangles en



(a) Cas bidimensionnel

(b) Cas tridimensionnel

FIG. 3.8 : Découpage élémentaire du raffinement régulier

joignant le milieu de ses arêtes. Il est trivial de vérifier que les sous–triangles ainsi générés appartiennent à la même classe de congruence que le triangle initial. En dimension trois, joindre le milieu des arêtes d'un tétraèdre définit quatre sous–tétraèdres et un octaèdre. De la même façon, ces quatre sous–tétraèdres sont trivialement congrus au tétraèdre initial. L'octaèdre interne est à son tour divisé en quatre tétraèdres, en le découpant selon l'une de ses trois diagonales (voir figure 3.9). En général, ces quatre sous–tétraèdres n'ap-

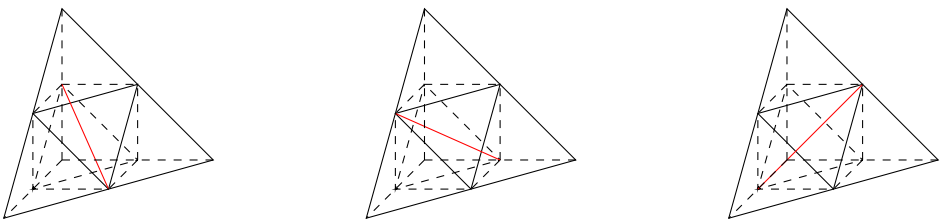


FIG. 3.9 : Découpages possibles de l'octaèdre interne

partiennent pas à la même classe de congruence que le tétraèdre initial. Le choix de la diagonale de l'octaèdre interne constitue donc un élément clef pour assurer la stabilité

de la hiérarchie de triangulations engendrée par cette méthode. En effet, S. ZHANG a pu montrer que si la diagonale choisie est toujours la plus longue, alors la hiérarchie générée ne peut être stable [149, 150]. Pour remédier à ce problème, S. ZHANG propose essentiellement deux solutions qui permettent de choisir correctement cette diagonale. La première solution consiste procéder de la façon suivante. Lors du premier découpage, la diagonale est choisie de façon arbitraire. Lors des découpages suivants, la diagonale choisie est celle qui s'appuie sur l'arête correspondant à la diagonale choisie au découpage précédent. La seconde solution, plus simple en apparence, consiste à toujours choisir la diagonale la plus courte. Bien que ces solutions semblent radicalement différentes, on dispose du résultat suivant, établi par M.-E. ONG [107] :

Proposition 3.1 : *Si, dans la première approche, la diagonale choisie lors du découpage initial correspond à la plus courte diagonale du tétraèdre de départ, alors les éléments engendrés par ces deux stratégies de découpage sont strictement identiques.*

Ainsi, ces deux solutions sont en réalité équivalentes. De part sa simplicité conceptuelle, la seconde approche semble être la plus simple à mettre en œuvre.

Cependant, M.-E. ONG [107] et J. BEY [24, 25] remarquent tous deux que la première approche de S. ZHANG constitue en réalité une spécialisation à la dimension trois d'un algorithme établi par H. FREUDENTHAL en 1942 [63] pour la résolution d'un problème d'analyse combinatoire. Cet algorithme correspond en quelque sorte à la généralisation canonique en dimension quelconque de la méthode de R.-E. BANK, A.-H. SHERMAN et A. WEISER. Bien que les problèmes étudiés ici soient toujours définis sur des domaines de $\mathbb{R}^2$ ou de $\mathbb{R}^3$, l'étude de l'algorithme de H. FREUDENTHAL s'avère extrêmement fructueuse. Nous en rappelons ici les principaux résultats. L'algorithme de H. FREUDENTHAL, dans le cas bidimensionnel, est équivalent au découpage introduit précédemment. Dans le cas tridimensionnel, il peut s'écrire sous la forme suivante :

Proposition 3.2 : *Soit $K = [x_0, x_1, x_2, x_3]$ un tétraèdre de $\mathbb{R}^3$. On note x_{ij} le milieu du segment $[x_i, x_j]$. Alors les tétraèdres suivants définissent un raffinement de K :*

$$\begin{aligned} K_1 &= [x_0, x_{01}, x_{02}, x_{03}] & K_5 &= [x_{02}, x_{12}, x_2, x_{23}] \\ K_2 &= [x_{01}, x_1, x_{12}, x_{13}] & K_6 &= [x_{02}, x_{12}, x_{13}, x_{23}] \\ K_3 &= [x_{01}, x_{02}, x_{03}, x_{13}] & K_7 &= [x_{02}, x_{03}, x_{13}, x_{23}] \\ K_4 &= [x_{01}, x_{02}, x_{12}, x_{13}] & K_8 &= [x_{03}, x_{13}, x_{23}, x_3] \end{aligned}$$

La formulation explicite de ce raffinement est particulièrement utile, car son implémentation en devient triviale. Ce raffinement vérifie par ailleurs la propriété suivante :

Propriété 3.5 : *Soit Ω un domaine polygonal borné de $\mathbb{R}^3$, et $\mathcal{T}_0$ une triangulation conforme de Ω. L'application itérative de l'algorithme de* H. FREUDENTHAL *fournit une hiérarchie de triangulations $(\mathcal{T}_i)_i$ vérifiant :*

(i) quelque soit i, $\mathcal{T}_i$ est une triangulation conforme de Ω ;

(ii) le nombre de classes de congruence des simplexes appartenant à l'une quelconque des triangulations $\mathcal{T}_i$ est borné indépendamment de i.

La démonstration de cette propriété est présentée dans l'article [25]. Notons cependant que la définition de ce raffinement ne tient absolument pas compte des propriétés géométriques des éléments du maillage. En particulier, l'orientation des tétraèdres et de leurs faces n'est pas conservée lors de son application. Ceci provient du fait que cet algorithme a été établi à partir de considérations purement algébriques. Pour des raisons

de simplicité d'implémentation, il serait souhaitable de disposer d'un algorithme de raffinement qui conserve les propriétés géométriques des éléments.

Remarquons que ce raffinement est identique à celui défini par la première méthode de S. ZHANG, sous la condition que la diagonale choisie soit toujours celle qui relie le milieu de l'arête $[x_0, x_2]$ et le milieu de l'arête $[x_1, x_3]$. Cette propriété peut être établie en observant que la définition des éléments K_3, K_4, K_6 et K_7 place systématiquement les sommets x_{02} et x_{13} respectivement en première et troisième position ou en deuxième et quatrième position. Cette propriété caractérise la méthode de S. ZHANG. Or nous pouvons spécifier un algorithme qui vérifie cette propriété, mais qui conserve également les propriétés d'orientation des tétraèdres :

Proposition 3.3 : *Soit $K = [x_0, x_1, x_2, x_3]$ un tétraèdre de $\mathbb{R}^3$. En adoptant les mêmes notations que précédemment, les tétraèdres suivants définissent un raffinement de K :*

$$\begin{aligned} K_1 &= [x_0, x_{01}, x_{02}, x_{03}] & K_5 &= [x_{02}, x_{23}, x_{13}, x_{12}] \\ K_2 &= [x_{01}, x_1, x_{12}, x_{13}] & K_6 &= [x_{23}, x_{13}, x_{03}, x_{02}] \\ K_3 &= [x_{02}, x_{12}, x_2, x_{23}] & K_7 &= [x_{13}, x_{03}, x_{02}, x_{01}] \\ K_4 &= [x_{03}, x_{13}, x_{23}, x_3] & K_8 &= [x_{12}, x_{02}, x_{01}, x_{13}] \end{aligned}$$

On vérifie aisément que ce raffinement permet de conserver l'orientation des tétraèdres, et nous verrons au chapitre 6 qu'il est également cohérent avec la définition et l'orientation de leurs faces. Ainsi, nous venons de définir une subdivision de K qui est également équivalente à celle de S. ZHANG, et par conséquent à celle de H. FREUDENTHAL, et qui présente le comportement souhaité par rapport aux propriétés géométriques des tétraèdres.

Nous disposons donc d'une méthode de subdivision applicable en dimensions deux et trois, qui présente toutes les qualités requises :

- elle assure la stabilité de la hiérarchie de triangulations qu'elle engendre ;
- elle assure la conformité de la hiérarchie de triangulations qu'elle engendre ;
- elle est simple à implémenter, grâce à la donnée explicite du raffinement d'un élément ;

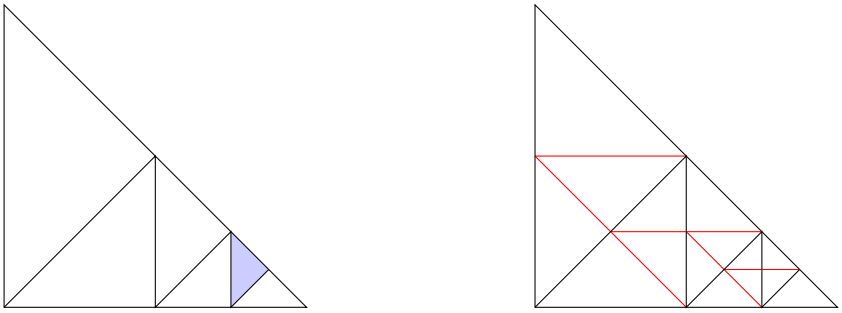
En outre, nous remarquons que la méthode de subdivision ne présente pas de caractère récursif, et que sa complexité est linéaire, et par là même optimale.

3.3 Localisation du raffinement

Jusqu'à présent, nous n'avons considéré que des procédures de raffinement s'appliquant à la totalité d'une triangulation. Or, selon le principe même des méthodes adaptatives, l'objectif n'est pas de raffiner l'intégralité du maillage, mais de n'en raffiner qu'une partie, désignée par le calcul de l'estimation de l'erreur. Nous étudions donc ici le comportement des techniques précédemment présentées lorsque l'on cherche à ne raffiner qu'une partie d'une triangulation.

3.3.1 Méthodes de bissection récursive

Les méthodes de bissection récursive s'adaptent relativement bien à la définition d'un raffinement local. En effet, il suffit, dans la première étape de l'algorithme 3.6, de ne placer que les éléments que l'on veut raffiner dans l'ensemble $\mathscr{S}_0$. On est alors assuré que les



(a) Maillage initial (b) Maillage adapté

FIG. 3.10 : Propagation de la zone de raffinement par une méthode de bissection

éléments sélectionnés seront raffinés dans la triangulation générée. Notons cependant que le caractère récursif de l'algorithme de bissection récursive peut induire le raffinement d'autres éléments, afin de maintenir la conformité du maillage (voir figure 3.7).

Ce phénomène de propagation du raffinement peut toutefois poser quelques difficultés. En effet, la zone de propagation peut être très importante, comme cela est illustré sur la figure 3.10. Dans le cas des méthodes de M.-C. RIVARA, l'importance de la zone de propagation dépend essentiellement des propriétés géométriques du maillage. À l'inverse, dans le cas des méthodes de W.-F. MITCHELL, de J.-M.-L. MAUBACH ou de C.-T. TRAXLER, la propagation du raffinement dépend plutôt de la connectivité du maillage. On voit donc que bien que la définition d'un raffinement sélectif à l'aide d'une méthode de bissection récursive soit particulièrement simple, la localité stricte du raffinement reste difficile à obtenir.

En particulier, l'extension de la zone de raffinement peut poser quelques difficultés lors de l'implémentation parallèle de l'algorithme de raffinement. En effet, la propagation s'effectue de proche en proche, d'un élément vers ses éléments voisins. Cependant, dans un contexte parallèle, ces éléments peuvent être distribués sur des processeurs différents. Ainsi, afin de propager correctement le raffinement, l'algorithme de bissection récursive induit des communications inter–processeurs à chacune de ses itérations. Comme pour tous les algorithmes de fermeture transitive, le volume et la nature de ces communications sont très difficiles à prévoir et à mettre en œuvre. Ainsi, une implémentation parallèle efficace d'un algorithme de bissection récursive reste donc une question ouverte.

3.3.2 Méthodes de subdivision

Supposons maintenant que l'on adopte une méthode de subdivision, et que l'on ne souhaite raffiner qu'une partie des éléments de la triangulation. Comme cela est illustré sur la figure 3.11, cette approche génère des non–conformités. Deux solutions peuvent être envisagées pour prendre en compte ces non–conformités. L'approche la plus communément retenue consiste à découper les éléments présentant une non–conformité selon un patron de découpage différent, de façon à rétablir la conformité du maillage. Cette idée remonte à R.-E. BANK, A.-H. SHERMAN et A. WEISER pour la dimension deux [12], et elle a été étendue à la dimension trois par J. BEY [24, 25]. La figure 3.12 présente différents patrons de raffinement envisageables. Ces raffinements sont qualifiés de *non–réguliers*,

(a) Triangulation initiale (b) Triangulation raffinée

FIG. 3.11 : Application locale d'une méthode de subdivision

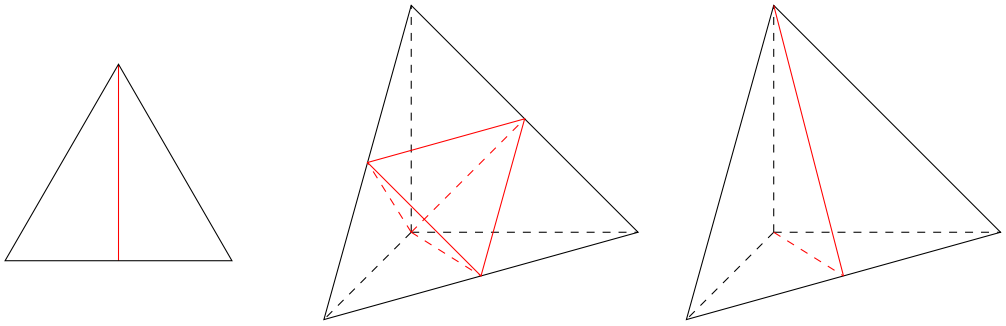


FIG. 3.12 : Exemples de raffinements non–régulier pour les méthodes de subdivision

par opposition à celui utilisé dans la méthode de subdivision classique. L'intérêt majeur de cette approche est qu'elle reste purement géométrique, et qu'elle permet d'obtenir une triangulation conforme lors d'un raffinement local reposant sur une méthode de subdivision. Cependant, la mise en œuvre de cette technique repose, comme les méthodes de bissection récursive, sur un algorithme de fermeture transitive. On voit donc nécessairement apparaître une extension de la zone de raffinement lors de son application, et l'implémentation parallèle de cette approche s'avère aussi complexe que celle des méthodes de bissection récursive. Par ailleurs, l'introduction de raffinements non–réguliers n'est pas sans conséquence sur la qualité des maillages générés. En particulier, la propriété 3.5 n'est plus valide. Enfin, en dimension trois, la prise en compte des différents raffinements non–réguliers introduit de nombreux cas particuliers, ce qui complexifie d'autant sa mise en œuvre.

La seconde solution consiste à conserver le maillage tel quel après raffinement des éléments sélectionnés, et à prendre en compte convenablement les non–conformités au niveau de la méthode numérique. On assure de cette façon que les propriétés des méthodes de subdivision sont conservées, et que la zone de raffinement ne peut s'étendre au delà de la région sélectionnée par les estimateurs d'erreur. De même, cette approche n'introduit pas de cas particulier dans le traitement du maillage, ce qui facilite grandement sa mise en œuvre. Autre avantage, l'absence d'algorithme de fermeture, qui permet d'une part de conserver une complexité linéaire, et d'autre part de s'affranchir des difficultés rencontrées lors de l'implémentation parallèle de l'algorithme de raffinement. La prise en compte des non–conformités au niveau de la méthode numérique peut se faire, par exemple, au niveau de la procédure d'assemblage du problème discret. C'est la solution qui a été retenue dans le code CONCEPTS [62]. Une autre possibilité a été proposée par C. BERNARDI, F. HECHT et Y. MADAY [21, 22]. Elle consiste à adopter une approche par

décomposition de domaine, où ce dernier est découpé selon les différents niveaux de raffinement des éléments.

3.3.3 Choix de notre méthode adaptative

Étant donnés les éléments présentés ci–dessus, nous sommes maintenant en mesure de choisir l'une ou l'autre de ces techniques de raffinement. Rappelons que les critères que nous avons retenus sont les suivants :

- qualité des maillages générés ;
- complexité algorithmique ;
- simplicité d'implémentation.

En regard de la qualité des maillages générés, les méthodes de subdivision apparaissent clairement comme le choix pertinent. En effet, les résultats mathématiques dont disposent ces méthodes sont clairement supérieurs en dimension trois à ceux établis pour les méthodes de bissection. Pour ce qui est de la complexité algorithmique comme de la simplicité d'implémentation, le point le plus important semble être de devoir éliminer toute forme d'algorithme de fermeture transitive. En effet, ce genre d'algorithme présente une complexité en $\mathcal{O}(n \log n)$, clairement supérieure à la complexité optimale qui est en $\mathcal{O}(n)$, et leur mise en œuvre dans un contexte parallèle reste très difficile. Ainsi, notre choix en matière de méthode de raffinement de maillage se porte sur la méthode de subdivision reposant sur le découpage présenté dans la proposition 3.3, qui satisfait tous les critères énumérés ci–dessus. Lors du traitement d'un raffinement localisé, nous optons pour une approche numérique du traitement des non–conformités, ce qui permet d'assurer que l'on n'introduit pas d'algorithme de fermeture. Nous présentons dans le chapitre suivant les concepts nécessaires à la mise en œuvre de cette approche pour la discrétisation VEF de Trio-U.

3.4 Prise en compte du déraffinement

Jusqu'à présent, nous n'avons parlé que de raffinement de maillage. Or, dans le cadre d'une méthode adaptative, la prise en compte du déraffinement est tout aussi importante, en particulier lors de l'étude de problèmes instationnaires, pour lesquels les singularités (et donc les zones qui nécessitent d'être raffinées) peuvent se déplacer au cours de la simulation. Nous présentons ici une technique permettant de gérer le déraffinement local du domaine, qui repose sur une approche proposée par J. Bey [24].

3.4.1 Différence entre h–méthodes et méthodes multi–niveaux

Dans le cadre d'une h–méthode, la discrétisation du domaine repose sur l'utilisation d'un maillage unique. Pour ce faire, lors de l'étape de raffinement, les éléments sélectionnés sont supprimés, puis remplacés par les éléments engendrés par leur découpage. Cette approche présente l'avantage de minimiser l'empreinte mémoire, car on ne conserve alors que les éléments du maillage qui présentent un intérêt pour le calcul. Cependant, cette technique ne permet pas de réaliser un déraffinement de façon simple. En effet, un déraffinement correspond alors au remplacement d'un ensemble d'éléments (quatre en dimension deux et huit en dimension trois) par un unique élément dont ils

forment une partition. Or si l'on ne dispose que de la géométrie des éléments à déraffiner, il s'avère difficile de vérifier s'ils forment une partition d'un élément plus grossier. Il est souhaitable de conserver une information supplémentaire lors du raffinement, qui permette d'identifier les ensembles d'éléments qui sont issus du raffinement d'un même élément initial. Il s'agit précisément de l'information conservée lors de l'utilisation d'une méthode multi–niveaux. Par définition, une méthode multi–niveaux repose sur l'intéraction de solutions calculées sur des niveaux de raffinement différents. Pour ce faire, il est nécessaire de conserver la totalité de la hiérarchie de triangulations générée par l'algorithme de raffinement. Ainsi, on dispose de l'information indiquant que tel élément de tel niveau est issu du raffinement de tel autre élément appartenant au niveau inférieur, ce qui correspond en quelque sorte à « l'historique du raffinement du maillage ». On peut alors exploiter cette information pour implémenter de façon simple une stratégie de déraffinement.

3.4.2 Algorithme général d'adaptation de maillage

Nous proposons ici un algorithme global d'adaptation de maillage qui permet de prendre en compte tant le raffinement que le déraffinement. Cet algorithme repose sur l'information disponible lorsque l'on conserve l'intégralité de la hiérarchie de discrétisations. Contrairement aux approches classiques, on ne cherche pas à définir un maillage à partir d'un autre maillage, mais à construire une hiérarchie de triangulations à partir d'une autre.

Considérons donc une hiérarchie de triangulations $(\mathscr{T}_i)_{0\leq i\leq n}$. Chaque niveau $\mathscr{T}_i$ de la hiérachie rassemble deux types d'éléments :

– des éléments raffinés ;
– des éléments non–raffinés.

Les éléments fils d'un élément raffiné d'un niveau donné sont ceux du niveau suivant générés par son découpage. Ainsi, le niveau $\mathscr{T}_{i+1}$ rassemble tous les éléments fils des éléments raffinés du niveau $\mathscr{T}_i$. Bien entendu, le dernier niveau $\mathscr{T}_n$ ne contient que des éléments non–raffinés. À l'issue du calcul de l'estimation d'erreur, chaque élément non–raffiné de chaque niveau de la hiérarchie est marqué d'un drapeau indiquant les modifications à lui apporter. Ce drapeau peut prendre les valeurs 1, 0, ou -1, selon que l'on souhaite le raffiner, le laisser tel quel, ou le déraffiner, respectivement. Partant de cette information, l'algorithme 3.13 permet de mettre à jour la hiérarchie de triangulations de façon à obtenir une nouvelle hiérarchie conforme à ces indications.

Il est parfaitement possible d'obtenir une hiérarchie présentant un niveau de raffinement inférieur à celle donnée en entrée : ce cas correspond à un déraffinement pur. Notons cependant que le déraffinement ne pourra être réalisé au delà de la triangulation de départ $\mathscr{T}_0$, qui constitue le maillage le plus grossier atteignable. Cette contrainte reste somme toute assez naturelle, car on suppose que la triangulation initiale correspond au maillage le plus grossier qui permette de décrire les aspérités de la frontière du domaine, et de prendre en compte les singularités des termes sources ou les discontinuités des coefficients caractéristiques du milieu. Par ailleurs, cet algorithme laisse toute liberté quant au choix entre une h–méthode ou une méthode multi–niveaux. En effet, la différence entre ces méthodes se situe alors au niveau de l'utilisation de la hiérarchie de triangulations qu'il génère. Si l'on utilise la totalité de la hiérarchie pour discrétiser les champs, et que la méthode retenue repose sur l'intéraction entre les solutions calculées sur chaque

```
for (i = 0 to n) {
  for all (K ∈ T_i) {
    if (K est raffiné) {
      drapeau(K) = 1
    }
  }
}

for (i = n to 0) {
  for all (K ∈ T_i) {
    if (K est raffiné) {
      if (drapeau(K_j) = −1, ∀K_j ∈ T_{i+1} fils de K) {
        drapeau(K) = 0
        for all (K_j ∈ T_{i+1} fils de K) {
          supprimer K_j de T_{i+1}
        }
      }
      else {
        for all (K_j ∈ T_{i+1} fils de K, drapeau(K_j) = −1) {
          drapeau(K_j) = 0
        }
      }
    }
  }
}

for (i = 0 to n) {
  for all (K ∈ T_i) {
    if (K n'est pas raffiné et drapeau(K) = 1) {
      for all (K_j fils de K) {
        insérer K_j dans T_{i+1}
        drapeau(K_j) = 0
      }
    }
  }
}
```

FIG. 3.13 : Algorithme global de mise à jour d'une hiérarchie de maillages

niveau, alors on a affaire à une méthode multi–niveaux. Si, à l'inverse, on choisit de n'utiliser que les éléments non–raffinés pour réaliser le calcul, alors il s'agit d'une h–méthode. Dans le cadre de nos travaux, nous avons choisi de nous tourner principalement vers la seconde approche. Nous parlons cependant d'une *h–méthode multi–niveaux*, de façon à expliciter le fait que notre approche repose sur la construction de la hiérarchie de triangulations.

Résumé

Ce chapitre a été consacré à une étude approfondie de diverses techniques d'adaptation de maillage. À l'issue de ce travail, nous sommes en mesure d'arrêter notre choix quant à l'adaptation géométrique que nous utiliserons dans notre méthode. Ces choix ont été établis sur la base des critères suivants :

- *qualité et non–dégénérescence des éléments des maillages générés ;*
- *optimalité de la complexité algorithmique ;*
- *simplicité de l'implémentation.*

Nous optons donc pour une approche hybride entre une h–méthode et une méthode multi–niveaux. L'adaptation du maillage repose sur une technique de subdivision inspirée de l'algorithme de H. FREUDENTHAL. *Nous décidons de traiter les non–conformités du maillage au niveau de la méthode numérique, ce qui sera présenté dans le chapitre suivant. Nous avons également spécifié un algorithme de modification du maillage qui permet de prendre en compte convenablement tant le raffinement local que le déraffinement local des élements de la triangulation.*

CHAPITRE 4

Choix d'une méthode adaptative : aspects numériques

Les idées développées jusqu'à présent ne portent que sur les aspects géométriques de notre méthode adaptative. Cependant, les choix que nous avons faits présentent des répercussions sur la forme de la méthode numérique utilisée pour discrétiser le problème. Nous présentons dans ce chapitre les compléments qu'il est nécessaire d'apporter à la méthode de Volumes–Éléments Finis de Trio-U pour mettre en œuvre notre approche adaptative au sein de cette plate–forme logicielle. Ainsi, après avoir rappelé le principe de cette méthode, nous décrivons les outils numériques nécessaires à notre démarche, à savoir la définition d'éléments joints pour l'élément fini de CROUZEIX–RAVIART*, les opérateurs d'interpolation hiérarchique des champs, et la notion d'estimateur d'erreur a posteriori.*

4.1 La méthode des Volumes–Éléments Finis

4.1.1 Principe de la méthode VEF de Trio-U

La méthode de Volumes–Éléments Finis implémentée dans Trio-U a été initialement proposée par P. ÉMONOT [52], et développée par la suite par P. CLÉMENT [42], S. HEIB [69], et T. FORTIN [60] dans leurs travaux de thèse respectifs. Cette méthode correspond à une approche hybride entre une méthode de Volumes Finis et une méthode d'Éléments Finis. L'idée principale consiste à associer un volume de contrôle à chaque degré de liberté d'une discrétisation par Éléments Finis, et à interpréter les intégrations par parties de cette méthode comme l'écriture de lois de conservation sur ces volumes. La figure 4.1 présente une construction possible de tels volumes associés aux degrés de liberté d'une discrétisation par éléments finis $\mathbb{P}^1$ de LAGRANGE. Grâce à cette réinterprétation, on montre aisément le caractère conservatif de la méthode, ce qui est essentiel pour la réalisation de simulations en thermohydraulique. Par ailleurs, le choix des éléments finis utilisés dans la méthode VEF de Trio-U présente une propriété remarquable. On montre en effet que la réinterprétation en termes de volumes finis de ces éléments n'a d'influence que sur le calcul de la matrice de masse et la prise en compte des conditions aux limites. Cette propriété facilite grandement l'assemblage du système linéaire associé au problème discrétisé, et elle permet de réutiliser les résultats de convergence de la méthode Éléments Finis sous–jacente.

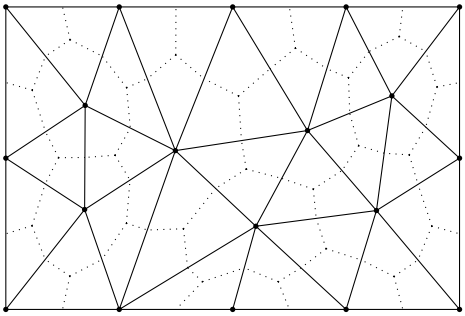


FIG. 4.1 : Exemples de volumes de contrôles associés au degrés de liberté $\mathbb{P}^1$ de LAGRANGE

Présentons donc plus en détail les éléments finis choisis comme fondements de la méthode VEF de Trio-U. Rappelons tout d'abord que le principal champ d'application de Trio-U reste la thermohydraulique d'écoulements de fluides incompressibles. À ce titre, les problèmes les plus couramment étudiés sont généralement des variantes de l'un des problèmes de DARCY, STOKES ou NAVIER–STOKES. Le dénominateur commun de cette famille de problème est qu'ils se présentent sous la forme d'une recherche de point selle :

Problème 4.1 : *Soient X et M deux espaces de* HILBERT. *Soit $f \in \mathscr{L}(X,\mathbb{R})$ une forme linéaire définie et continue sur X. On considère également $a \in \mathscr{L}(X\times X,\mathbb{R})$ et $b \in \mathscr{L}(X\times M,\mathbb{R})$ deux formes bilinéaires définies et continues respectivement sur $X\times X$ et $X\times M$. On recherche un couple $(u,p)\in X\times M$ tel que :*

$$\begin{aligned} a(u,v)+b(v,p) &= f(v) \qquad \forall v \in X \\ b(u,q) &= 0 \qquad \forall q \in M \end{aligned}$$

Un résultat classique [54] affirme que le caractère bien posé de ce genre de problème relève de l'existence d'une condition dite « inf–sup » ou de « BABUŠKA–BREZZI ». Plus exactement, nous disposons du théorème suivant :

Théorème 4.1 : *Le problème ci–dessus admet une unique solution dépendant continûment des données si et seulement si les deux hypothèses suivantes sont vérifiées :*

i. la forme bilinéaire $a(\cdot,\cdot)$ est définie positive sur le sous–espace K de X défini par :

$$K = \{u \in X,\ \forall q \in M, b(u,q) = 0\}$$

ii. il existe une constante β strictement positive telle que

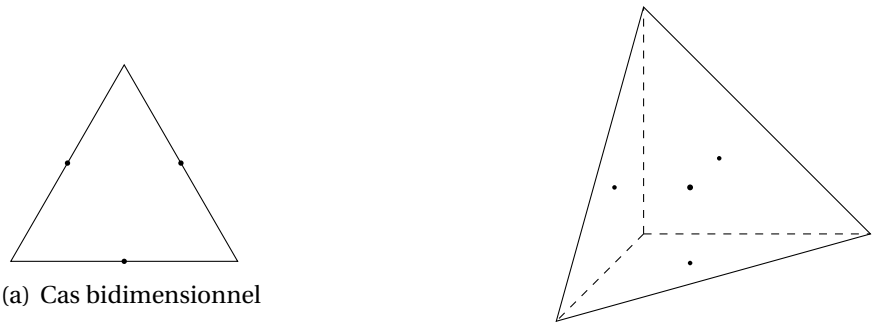
$$\inf_{q\in M}\sup_{v\in X}\frac{b(v,q)}{\|q\|_M \|v\|_X} \geq \beta$$

Dans ce cas, il existe deux constantes c_1 et c_2 telles que :

$$\begin{aligned} \|u\|_X &\leq c_1 \|f\|_{\mathscr{L}(X,\mathbb{R})} \\ \|p\|_M &\leq c_2 \|f\|_{\mathscr{L}(X,\mathbb{R})} \end{aligned}$$

La version discrète de ces problèmes, quelle que soit la discrétisation choisie, reste également sous la forme d'une recherche de point–selle, et, malheureusement, le caractère bien posé du problème continu n'implique en général pas celui du problème discret. Il faut donc choisir convenablement les espaces de discrétisation et l'expression des formes bilinéaires discrètes de façon à assurer qu'une condition de BABUŠKA–BREZZI, dite condition « inf–sup discrète », soit vérifiée. Dans le cas d'une méthode d'Éléments Finis, cela revient à choisir convenablement les éléments finis utilisés pour discrétiser X et M.

Une première possibilité, lors de l'étude de problèmes de DARCY, STOKES, ou NAVIER–STOKES, consiste à utiliser l'élément fini de TAYLOR–HOOD. Ce dernier revient à discrétiser l'espace des vitesses X à l'aide de l'élément fini $\mathbb{P}^2$ de LAGRANGE, et l'espace des pressions M à l'aide de l'élément fini $\mathbb{P}^1$ de LAGRANGE. Nous renvoyons à [54] pour le détail des propriétés de cet élément fini. Une autre possibilité, qui est le choix retenu dans la méthode VEF de Trio-U, consiste à discrétiser l'espace des vitesses à l'aide de l'élément fini de CROUZEIX–RAVIART, noté $\mathbb{P}^1_{NC}$ (voir [45]), et l'espace des pressions à l'aide de l'élément fini $\mathbb{P}^0$ de LAGRANGE. Une variante de cette approche, également implémentée dans Trio-U, consiste à enrichir l'espace de pression discret, en utilisant pour discrétiser M le couple d'éléments finis $\mathbb{P}^1+\mathbb{P}^0$ de LAGRANGE. Cette variante permet essentiellement d'améliorer les propriétés d'approximation de la méthode (voir par exemple [60] ou [69] pour une justification de ce choix).



(a) Cas bidimensionnel

(b) Cas tridimensionnel

FIG. 4.2 : Localisation des degrés de liberté de l'élément fini de CROUZEIX–RAVIART

Nous rappelons maintenant les propriétés de l'élément fini de CROUZEIX–RAVIART. Conformément à la définition générique d'un élément fini introduite par CIARLET [40], on peut le définir par le triplet $\mathbb{P}^1_{NC}=\{K,\mathbb{P},\Sigma\}$, où

- K désigne un simplexe de $\mathbb{R}^n$,
- $\mathbb{P}=\mathbb{P}^1(K)$ désigne l'ensemble des polynômes à coefficients réels, définis sur K et à valeurs dans $\mathbb{R}$, de degré total inférieur ou égal à 1,
- $\Sigma=(\sigma_i)_{0\leq i\leq n}$ désigne l'ensemble des formes linéaires σ_i définies sur $\mathbb{P}$ par :

$$\forall p\in\mathbb{P},\ \sigma_i(p)=\frac{1}{|\partial_{n-1}K_i|}\int_{\partial_{n-1}K_i}p$$

Autrement dit, la forme linéaire σ_i associe à tout polynôme $p\in\mathbb{P}$ sa valeur moyenne sur la i^{e} face de K. Comme les polynômes de $\mathbb{P}$ sont au plus de degré 1, leur valeur moyenne sur une face de K est égale à leur valeur ponctuelle au barycentre de cette face. C'est pourquoi l'on représente fréquemment les degrés de liberté de l'élément fini $\mathbb{P}^1_{NC}$ comme cela est illustré sur la figure 4.2.

4.1.2 Application au problème de LAPLACE

Le problème de LAPLACE, muni de conditions aux limites de DIRICHLET homogènes, s'écrit sous la forme suivante :

Problème 4.2 : *Soit Ω un ouvert borné de $\mathbb{R}^n$, et f une fonction de $L^2(\Omega)$. On cherche une fonction u telle que :*

$$\begin{aligned} -\Delta u &= f \qquad dans\, \Omega \\ u &= 0 \qquad sur\, \partial\Omega \end{aligned}$$

Une formulation variationnelle de ce problème est donnée par :

Problème 4.3 : *Trouver $u \in H_0^1(\Omega)$ telle que $\forall v \in H_0^1(\Omega)$, l'on ait :*

$$a(u,v) = l(v)$$

où on a posé :

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \qquad l(v) = \int_\Omega f v$$

Ce problème est bien posé au sens de J. HADAMARD. En effet, l'inégalité de POINCARÉ permet d'établir la coercivité de la forme bilinéaire $a(\cdot,\cdot)$ sur $H_0^1(\Omega)$, et le théorème de LAX–MILGRAM permet alors de conclure à l'existence d'une unique solution dépendant continûment de f. Comme on peut le voir, il ne s'agit pas là d'un problème de recherche de point–selle. Cependant, nous chercherons à le résoudre en utilisant la même discrétisation que celle utilisée par la méthode VEF pour ses inconnues primales, à savoir l'élément fini $\mathbb{P}^1_{NC}$. Considérons $\mathscr{T}_h$, une triangulation conforme de Ω. Notons $\mathscr{N}^I(\mathscr{T}_h)$ et $\mathscr{N}^\partial(\mathscr{T}_h)$ l'ensemble des barycentres des faces des éléments de $\mathscr{T}_h$ respectivement intérieures à Ω et situées sur le bord de Ω. On introduit alors l'espace discret :

$$\begin{aligned} X_h = \big\{ & u \in L^2(\Omega)\,;\ \forall K \in \mathscr{T}_h\,,\ u|_K \in \mathbb{P}^1(K)\,; \\ & \forall x \in \mathscr{N}^I(\mathscr{T}_h)\,,\ u \text{ continue en } x\,;\ \forall x \in \mathscr{N}^\partial(\mathscr{T}_h)\,,\ u(x) = 0 \big\} \end{aligned}$$

ainsi que la forme bilinéaire discrète :

$$\begin{aligned} a_h\ : X_h \times X_h &\longrightarrow \mathbb{R} \\ (u_h, v_h) &\longmapsto a_h(u_h, v_h) = \sum_{K \in \mathscr{T}_h} \int_K \nabla u_h \cdot \nabla v_h \end{aligned}$$

et la forme linéaire discrète :

$$\begin{aligned} l_h : X_h &\longrightarrow \mathbb{R} \\ v_h &\longmapsto l_h(v_h) = \sum_{K \in \mathscr{T}_h} \int_K f v_h \end{aligned}$$

Notre problème discret s'écrit alors :

Problème 4.4 : *Trouver $u_h \in X_h$ telle que $\forall v_h \in X_h$, l'on ait :*

$$a_h(u_h, v_h) = l_h(v_h)$$

Bien que ce problème discret constitue en réalité une approximation non–conforme du problème continu 4.3, car $X_h \not\subset H_0^1(\Omega)$, il reste un problème bien posé au sens de J. HADAMARD, dont la solution est une approximation de celle du problème continu (voir [54] pour l'étude exhaustive des propriétés de ce problème). De plus, on dispose de l'estimation d'erreur suivante :

Lemme 4.1 : *Supposons que la solution u du problème 4.3 soit dans $H^2(\Omega)$. Il existe une constante c telle que :*

$$\forall h, \|u - u_h\|_{V(h)} \leq c\, h\, |u|_{H^2(\Omega)}$$

où h désigne le pas du maillage $\mathcal{T}_h$, $V(h)$ désigne l'espace $X_h + H_0^1(\Omega)$ muni de la norme :

$$\forall v \in V(h),\ \|u\|_{V(h)} = \left(\int_\Omega u^2 \right)^{\frac{1}{2}} + \left(\sum_{K \in \mathcal{T}_h} \int_K \nabla v \cdot \nabla v \right)^{\frac{1}{2}}$$

et où $|\cdot|_{H^2(\Omega)}$ désigne la semi–norme associée à l'espace $H^2(\Omega)$.

À titre d'exemple, considérons le problème suivant :

Problème 4.5 : *On cherche une fonction u telle que :*

$$\begin{aligned} -\Delta u &= 2\sin(x)\sin(y) && \text{dans } \Omega = [0,\pi]^2 \\ u &= 0 && \text{sur } \partial\Omega \end{aligned}$$

La solution analytique de ce problème est donnée par :

$$\begin{aligned} u : [0,\pi]^2 &\longrightarrow \mathbb{R} \\ (x,y) &\longmapsto u(x,y) = \sin(x)\sin(y) \end{aligned}$$

La figure 4.3 présente la solution approchée obtenue par la résolution du problème issu de la discrétisation de ce problème par l'élément fini de CROUZEIX–RAVIART $\mathbb{P}_{NC}^1$. Sur cette

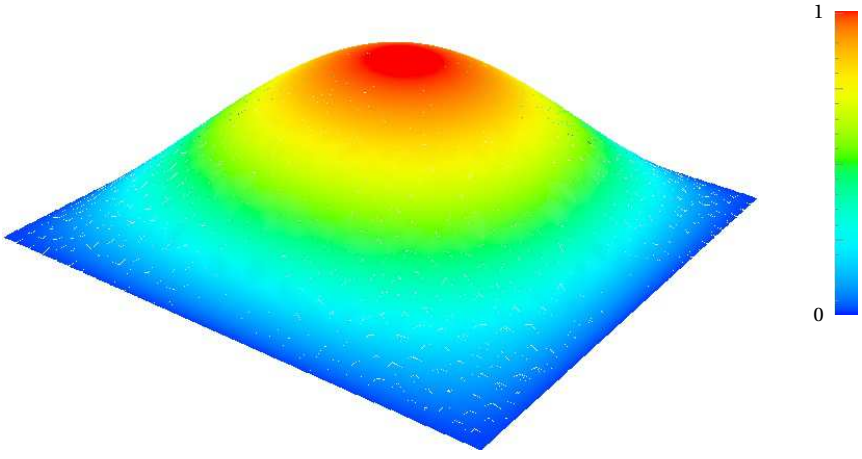


FIG. 4.3 : Solution du problème 4.5 par discrétisation par l'élément fini $\mathbb{P}_{NC}^1$

figure, on remarque quelques irrégularités sur la solution : elles proviennent essentiellement du fait que nous visualisons le champ tel qu'il a été calculé, en conservant ses non-conformités.

4.1.3 Application au problème de STOKES

Le problème de STOKES, muni de conditions aux limites de DIRICHLET homogènes, s'écrit sous la forme suivante :

Problème 4.6 : *Soit Ω un ouvert borné de $\mathbb{R}^n$, et $\boldsymbol{f}$ une fonction de $L^2(\Omega)^n$. On cherche un couple de fonctions $(\boldsymbol{u}, p)$, à valeurs dans $\mathbb{R}^n$ et $\mathbb{R}$ respectivement, tel que :*

$$\begin{aligned} -\Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \qquad \text{dans } \Omega \\ \nabla \cdot \boldsymbol{u} &= 0 \qquad \text{dans } \Omega \\ \boldsymbol{u} &= 0 \qquad \text{sur } \partial\Omega \end{aligned}$$

Une formulation variationnelle de ce problème est donnée par :

Problème 4.7 : *Trouver $(\boldsymbol{u}, p) \in H_0^1(\Omega)^n \times L_0^2(\Omega)$ tel que $\forall (\boldsymbol{v}, q) \in H_0^1(\Omega)^n \times L_0^2(\Omega)$, l'on ait :*

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{v}) - b(\boldsymbol{v}, p) &= l(\boldsymbol{v}) \\ b(\boldsymbol{u}, q) &= 0 \end{aligned}$$

où $L_0^2(\Omega)$ désigne l'ensemble des fonctions de $L^2(\Omega)$, à valeur moyenne nulle sur Ω, et où l'on a posé :

$$a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \qquad b(\boldsymbol{v}, q) = \int_\Omega q \nabla \cdot \boldsymbol{u} \qquad l(\boldsymbol{v}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}$$

Le caractère bien posé de ce problème peut être établi grâce à deux résultats fondamentaux :

- la coercivité de la forme bilinéaire $a(\cdot, \cdot)$;
- la surjectivité de l'opérateur $\nabla \cdot : H_0^1(\Omega)^n \longrightarrow L_0^2(\Omega)$.

Ce second résultat a été établi par BOGOVSKIĬ dans [27] et permet d'établir l'existence d'une constante réelle β strictement positive telle que :

$$\inf_{q \in L_0^2(\Omega)} \sup_{\boldsymbol{v} \in H_0^1(\Omega)^n} \frac{\int_\Omega q \nabla \cdot \boldsymbol{v}}{\|v\|_{H_0^1(\Omega)^n} \left\|q\right\|_{L_0^2(\Omega)}} \geq \beta$$

Voyons maintenant l'expression du problème approché que l'on obtient en discrétisant ce problème à l'aide de la méthode VEF de Trio-U. Soit donc $\mathcal{T}_h$ une triangulation conforme de Ω, et notons comme précédemment $\mathcal{N}^I(\mathcal{T}_h)$ et $\mathcal{N}^\partial(\mathcal{T}_h)$ l'ensemble des barycentres des faces des éléments de $\mathcal{T}_h$ respectivement intérieures à Ω et situées sur le bord de Ω. On introduit alors l'espace des vitesses discrètes :

$$\begin{aligned} \boldsymbol{X}_h = \big\{ \boldsymbol{u} \in L^2(\Omega)^n \ ; \ & \forall K \in \mathcal{T}_h \ , \ \boldsymbol{u}|_K \in \mathbb{P}^1(K)^n \ ; \\ & \forall x \in \mathcal{N}^I(\mathcal{T}_h) \ , \ \boldsymbol{u} \text{ continue en } x \ ; \ \forall x \in \mathcal{N}^\partial(\mathcal{T}_h) \ , \ \boldsymbol{u}(x) = 0 \big\} \end{aligned}$$

Cet espace n'étant pas inclus dans $H_0^1(\Omega)^n$, nous ne pouvons pas utiliser un produit scalaire induit. Nous introduisons donc le produit scalaire et sa norme induite, définis sur $\boldsymbol{X}_h$ par :

$$(\boldsymbol{u}_h, \boldsymbol{v}_h)_{\boldsymbol{X}_h} = (\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) = \sum_{K \in \mathcal{T}_h} \int_K \nabla \boldsymbol{u}_h : \nabla \boldsymbol{v}_h$$

$$\|\boldsymbol{u}_h\|_{\boldsymbol{X}_h} = \left(\sum_{K \in \mathcal{T}_h} \int_K \nabla \boldsymbol{u}_h : \nabla \boldsymbol{u}_h \right)^{\frac{1}{2}}$$

Nous définissons alors la forme bilinéaire discrète :

$$\begin{aligned} a_h &: \boldsymbol{X}_h \times \boldsymbol{X}_h \longrightarrow \mathbb{R} \\ &(\boldsymbol{u}_h, \boldsymbol{v}_h) \longmapsto a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) = \sum_{K \in \mathscr{T}_h} \int_K \nabla \boldsymbol{u}_h : \nabla \boldsymbol{v}_h = (\boldsymbol{u}_h, \boldsymbol{v}_h)_{\boldsymbol{X}_h} \end{aligned}$$

Sur l'ensemble $\left(\boldsymbol{X}_h \cap H_0^1(\Omega)^n\right) \times \left(\boldsymbol{X}_h \cap H_0^1(\Omega)^n\right)$, cette forme bilinéaire est consistante avec la forme continue $a(\cdot,\cdot)$. Elle est également uniformément continue sur $\boldsymbol{X}_h \times \boldsymbol{X}_h$ (voir [69]). Nous définissons également la forme linéaire suivante :

$$\begin{aligned} l_h &: \boldsymbol{X}_h \longrightarrow \mathbb{R} \\ &\boldsymbol{v}_h \longmapsto l_h(\boldsymbol{v}_h) = \sum_{K \in \mathscr{T}_h} \int_K \boldsymbol{f} \cdot \boldsymbol{v}_h \end{aligned}$$

Pour définir l'espace des pressions discrètes, nous introduisons les espaces auxiliaires suivants :

$$\begin{aligned} W_h^0 &= \{q \in L^2(\Omega)\,;\ \forall K \in \mathscr{T}_h,\ q|_K \in \mathbb{P}^0(K)\} \\ W_h^1 &= \{q \in \mathscr{C}^0(\Omega)\,;\ \forall K \in \mathscr{T}_h,\ q|_K \in \mathbb{P}^1(K)\} \end{aligned}$$

On pose alors :

$$\begin{aligned} M_h^0 &= W_h^0 \cap L_0^2(\Omega) \\ M_h^1 &= (W_h^0 + W_h^1) \cap L_0^2(\Omega) = \left(W_h^0 \cap L_0^2(\Omega)\right) + \left(W_h^1 \cap L_0^2(\Omega)\right) \end{aligned}$$

Remarquons que toute fonction q_h appartenant à M_h^1 peut s'écrire de la façon suivante :

$$q_h = q_h^0 + q_h^1 \quad \text{avec} \quad q_h^0 \in W_h^0 \cap L_0^2(\Omega)\left(= M_h^0\right) \quad \text{et} \quad q_h^1 \in W_h^1 \cap L_0^2(\Omega)$$

La version standard de la méthode VEF repose sur la forme bilinéaire suivante :

$$\begin{aligned} b_h^0 &: \boldsymbol{X}_h \times M_h^0 \longrightarrow \mathbb{R} \\ &(\boldsymbol{u}_h, q_h) \longmapsto b_h^0(\boldsymbol{v}_h, q_h) = \sum_{K \in \mathscr{T}_h} \int_K q_h \nabla \cdot \boldsymbol{v}_h \end{aligned}$$

La version étendue de la méthode VEF utilise quant à elle la forme bilinéaire :

$$\begin{aligned} b_h^1 &: \boldsymbol{X}_h \times M_h^1 \longrightarrow \mathbb{R} \\ &(\boldsymbol{u}_h, q_h) \longmapsto b_h^1(\boldsymbol{v}_h, q_h) = \sum_{K \in \mathscr{T}_h} \int_K q_h^0 \nabla \cdot \boldsymbol{v}_h + \sum_{F \in \mathscr{T}_h} \int_F [\![\boldsymbol{v}_h \cdot \boldsymbol{n}_F]\!]\, q_h^1 d\sigma \end{aligned}$$

où F, $\boldsymbol{n}_F$, et $[\![\cdot]\!]$ désignent respectivement une face du maillage associé à $\mathscr{T}_h$, la normale unitaire à cette face, et le saut d'une fonction à travers cette face (avec le signe approprié). Ces formes bilinéaires sont également consistantes avec la forme continue $b(\cdot,\cdot)$, respectivement sur $\left(\boldsymbol{X}_h \cap H_0^1(\Omega)^n\right) \times M_h^0$ et sur $\left(\boldsymbol{X}_h \cap H_0^1(\Omega)^n\right) \times M_h^1$. S. HEIB a pu montrer qu'elles vérifient chacune une condition de BABUŠKA–BREZZI discrète (voir pour cela [69]). Par conséquent, les problèmes discrets 4.8 et 4.9 sont bien posés au sens de J. HADAMARD, et leurs solutions constituent chacune une approximation de la solution du problème continu 4.7.

Problème 4.8 : *Trouver* $(\boldsymbol{u}_h, p_h) \in \boldsymbol{X}_h \times M_h^0$ *tel que* $\forall (\boldsymbol{v}_h, q_h) \in \boldsymbol{X}_h \times M_h^0$, *l'on ait :*

$$\begin{aligned} a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) - b_h^0(\boldsymbol{v}_h, p_h) &= l(\boldsymbol{v}_h) \\ b_h^0(\boldsymbol{u}_h, q_h) &= 0 \end{aligned}$$

Problème 4.9 : *Trouver* $(\boldsymbol{u}_h, p_h) \in \boldsymbol{X}_h \times M_h^1$ *tel que* $\forall (\boldsymbol{v}_h, q_h) \in \boldsymbol{X}_h \times M_h^1$, *l'on ait :*

$$\begin{aligned} a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) - b_h^1(\boldsymbol{v}_h, p_h) &= l(\boldsymbol{v}_h) \\ b_h^1(\boldsymbol{u}_h, q_h) &= 0 \end{aligned}$$

Afin d'illustrer l'intérêt de la version étendue de la méthode VEF, nous considérons le problème suivant :

Problème 4.10 : *Trouver* $(\boldsymbol{u}, p)$ *vérifiant :*

$$\begin{aligned} -\Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \qquad dans\, \Omega = [0,1] \\ \nabla \cdot \boldsymbol{u} &= 0 \qquad dans\, \Omega \\ \boldsymbol{u} &= 0 \qquad sur\, \partial\Omega \end{aligned}$$

où nous choisirons une fonction $\boldsymbol{f}$ particulière : $\boldsymbol{f} = \nabla\Phi$. Ainsi, la solution analytique de ce problème est donnée par :

$$\boldsymbol{u} = 0 \qquad p = \Phi + p_0$$

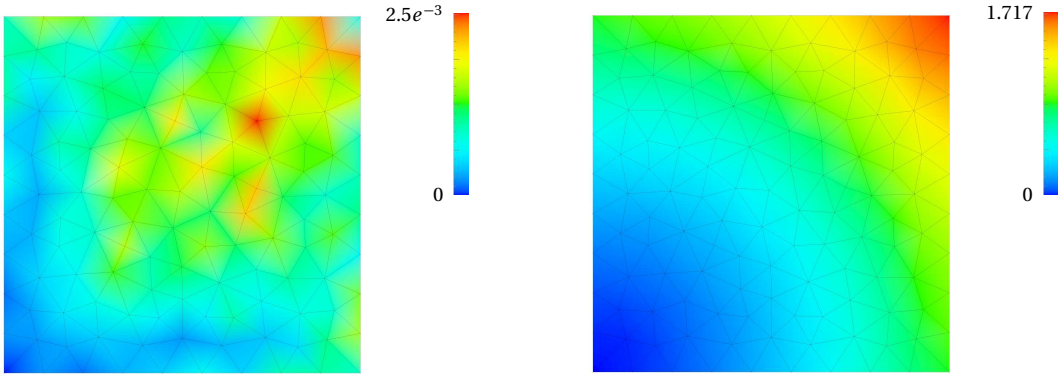
où p_0 est une constante, que nous déterminons en imposant :

$$\int_\Omega p = 0 \quad \text{ou bien} \qquad \int_\Omega p = \int_\Omega \Phi$$

En choisissant la seconde option, nous obtenons $p = \Phi$. Si l'on choisit pour Φ une fonction régulière (en pratique un polynôme de degré supérieur ou égal à deux suffit), nous observons que les méthodes VEF n'ont pas le même comportement. Ceci est illustré sur les figure 4.4 et 4.5, pour le choix de Φ suivant :

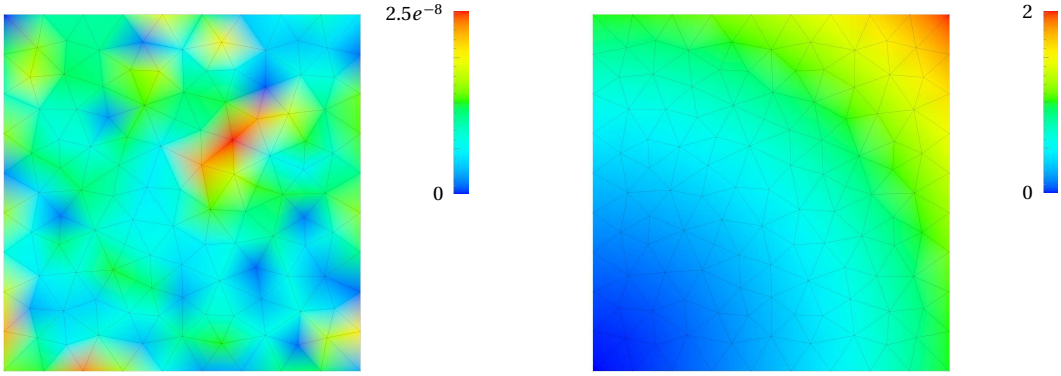
$$\Phi = x^2 + y^2$$

Dans le cas de la méthode standard, on constate que le champ de vitesse calculé n'est pas nul. On peut comprendre ce phénomène en analysant le comportement de la pression. En effet, dans la méthode standard, les polynômes utilisés pour discrétiser la pression sont de degré faible. Par conséquent, il est difficile d'obtenir une pression discrète dont le gradient soit proche de celui de Φ : $\nabla p_h \neq \nabla\Phi$. Ainsi, la pression ne peut à elle seule absorber l'intégralité du terme source, ce qui conduit à l'apparition de courants parasites. On comprend maintenant pourquoi la norme du champ de vitesse reste supérieure à 10^{-3} dans la figure 4.4(a). À l'inverse, dans la méthode étendue, on utilise des polynômes de degré 1 pour discrétiser la pression, ce qui permet d'approcher le terme source de façon beaucoup plus satisfaisante. Il est alors possible d'obtenir un champ de vitesse dont la norme est beaucoup plus faible. En pratique, dans le cas présenté sur la figure 4.5(a), la norme de la vitesse est contrôlée par le seuil de convergence imposé au solveur. Nous renvoyons à [60] pour plus de détails concernant les propriétés d'approximation des deux méthodes VEF implémentées dans Trio-U.



(a) Norme du champ de vitesse (b) Champ de pression

FIG. 4.4 : Résolution par la méthode VEF standard du problème 4.10 avec $\Phi = x^2 + y^2$



(a) Norme du champ de vitesse (b) Champ de pression

FIG. 4.5 : Résolution par la méthode VEF étendue du problème 4.10 avec $\Phi = x^2 + y^2$

4.2 Éléments joints pour l'élément de CROUZEIX–RAVIART

4.2.1 Motivations

L'étude des diverses techniques d'adaptation de maillage nous a amenés à choisir une approche nécessitant une méthode numérique capable de prendre en compte un maillage non–conforme issu du raffinement localisé par une méthode de subdivision. Or, à l'instar de la majorité des méthodes classiques d'Éléments Finis, la méthode VEF de Trio-U suppose l'utilisation d'un maillage conforme. Il est nécessaire de développer les outils numériques qui permettront d'exploiter cette méthode sur les maillages générés par nos algorithmes d'adaptation géométrique.

Nous choisissons pour cela de mettre en place une approche similaire à celle proposée par C. BERNARDI, Y. MADAY et F. HECHT dans [21, 22]. Dans ces articles, les auteurs proposent une approche originale, qui consiste à adopter une méthode de décomposition de domaine fondée sur la notion de niveau de raffinement. Le principe de cette technique revient à isoler dans un même sous–domaine les éléments du maillage adapté qui résultent du même nombre d'étapes de raffinement des éléments d'un maillage initial conforme. Ainsi, chacun des sous–domaines correspond à un maillage conforme, et

les non–conformités se trouvent cantonnées aux interfaces entre les sous–domaines. La technique numérique retenue par C. BERNARDI, Y. MADAY et F. HECHT pour traiter les non–conformités est une méthode d'éléments joints, développée pour des éléments finis $\mathbb{P}^k$ de LAGRANGE, avec $k \geq 2$.

En partant du même principe, nous nous proposons de développer un formalisme d'éléments joints pour l'élément fini de CROUZEIX–RAVIART, afin d'utiliser une approche similaire reposant sur la méthode VEF de Trio-U. L. MARCINKOWSKI propose un tel formalisme dans les articles [93, 94]. Les paragraphes qui suivent ont pour objectif de présenter en détail ce formalisme, et d'en proposer une extension facilitant sa mise en œuvre. Par ailleurs, nous verrons que les non–conformités qui apparaissent en raison de l'adaptation du maillage présentent des caractéristiques tout–à–fait remarquables, ce qui permet de simplifier considérablement les algorithmes de couplage entre les sous–domaines (voir les paragraphes 4.2.4 et 4.2.5).

4.2.2 Le formalisme de L. MARCINKOWSKI

Nous présentons ici un résumé des travaux de L. MARCINKOWSKI concernant l'élaboration d'une méthode d'éléments joints pour l'élément fini de CROUZEIX–RAVIART. Nous considérons donc un problème de LAPLACE muni de conditions aux limites de DIRICHLET homogènes :

Problème 4.11 : *Soit Ω un ouvert borné de $\mathbb{R}^n$ et f une fonction de $L^2(\Omega)$. Trouver $u \in H_0^1(\Omega)$ tel que $\forall v \in H_0^1(\Omega)$, l'on ait :*

$$a(u,v) = l(v)$$

où l'on a posé :

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \qquad l(v) = \int_\Omega f v$$

Par souci de simplicité, mais sans aucune restriction théorique, nous supposons que Ω est un domaine polygonal. Considérons une famille $(\Omega_i)_{1\leq i\leq N}$ de N sous–domaines polygonaux de Ω formant une partition de ce dernier :

$$\bigcup_{i=1}^{N} \overline{\Omega}_i = \overline{\Omega} \qquad (i \neq j) \Longrightarrow \left(\mathring{\Omega}_i \cap \mathring{\Omega}_j = \emptyset\right)$$

Supposons maintenant que chacun des sous–domaines Ω_i soit muni d'une triangulation conforme, $\mathscr{T}_h(\Omega_i)$. Remarquons que nous disposons alors d'un maillage de Ω qui n'est que localement conforme, mais qui peut être globalement non–conforme si les triangulations de deux sous–domaines voisins ne coïncident pas sur l'interface qui les sépare. Notons $\mathscr{N}^I(\mathscr{T}_h(\Omega_i))$ et $\mathscr{N}^\partial(\mathscr{T}_h(\Omega_i))$ l'ensemble des barycentres des faces des éléments de $\mathscr{T}_h(\Omega_i)$ respectivement intérieures à Ω_i et situées sur le bord extérieur de Ω_i (c'est–à–dire sur $\partial\Omega_i \cap \partial\Omega$). Introduisons, pour chaque sous–domaine Ω_i l'espace de discrétisation associé à l'élément fini de CROUZEIX–RAVIART :

$$\begin{aligned} W_h(\Omega_i) = \big\{ & u_h^i \in L^2(\Omega_i)\,;\ \forall K \in \mathscr{T}_h(\Omega_i)\,,\ u_h^i|_K \in \mathbb{P}^1(K)\,; \\ & \forall x \in \mathscr{N}^I(\mathscr{T}_h(\Omega_i))\,,\ u_h^i \text{ continue en } x\,;\ \forall x \in \mathscr{N}^\partial(\mathscr{T}_h(\Omega_i))\,,\ u_h^i(x) = 0 \big\} \end{aligned}$$

ainsi que les normes et semi–normes brisées suivantes :

$$\left\|u_h^i\right\|_{W_h(\Omega_i)} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left\|u_h^i\right\|^2_{H^1(K)}\right)^{\frac{1}{2}} \qquad \left|u_h^i\right|_{W_h(\Omega_i)} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left|u_h^i\right|^2_{H^1(K)}\right)^{\frac{1}{2}}$$

Nous pouvons maintenant définir un espace discret global par :

$$W_h = \prod_{i=1}^{N} W_h(\Omega_i) = \left\{u_h = \left(u_h^i\right)_{1\leq i\leq n},\ u_h^i \in W_h(\Omega_i)\right\}$$

muni de la norme et de la semi–norme suivante :

$$\|u_h\|_{W_h} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left\|u_h^i\right\|^2_{W_h(\Omega_i)}\right)^{\frac{1}{2}} \qquad |u_h|_{W_h} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left|u_h^i\right|^2_{W_h(\Omega_i)}\right)^{\frac{1}{2}}$$

Nous pouvons percevoir W_h comme l'ensemble des fonctions de $L^2(\Omega)$ dont la restriction à chacun des Ω_i appartient à l'espace $W_h(\Omega_i)$.

Considérons maintenant les interfaces Γ_{ij} entre les sous–domaines :

$$\Gamma_{ij} = \partial\overline{\Omega}_i \cap \partial\overline{\Omega}_j$$

Les triangulations des sous–domaines induisent sur chacune de ces interfaces deux triangulations indépendantes, que nous noterons $\mathscr{T}_h^i(\Gamma_{ij})$ et $\mathscr{T}_h^j(\Gamma_{ij})$. Sur une interface donnée, nous pouvons choisir arbitrairement l'une de ces deux triangulations comme étant la triangulation maître, et l'autre comme étant la triangulation esclave. En posant :

$$\eta = \min(i,j) \qquad \text{et} \qquad \xi = \max(i,j) \quad ,$$

nous choisirons systématiquement comme discrétisation maître (resp. esclave) de Γ_{ij} celle induite par la triangulation de Ω_η (resp. Ω_ξ). Grâce à ce choix, nous pouvons introduire un espace de fonctions tests défini sur Γ_{ij} grâce à la discrétisation maître :

$$M_h(\Gamma_{ij}) = \left\{u \in L^2(\Gamma_{ij}), \forall K \in \mathscr{T}_h^\eta(\Gamma_{ij}), u \in \mathbb{P}^0(K)\right\}$$

et nous introduisons également l'opérateur de projection au sens de $L^2(\Gamma_{ij})$ sur $M_h(\Gamma_{ij})$ défini par :

$$\begin{aligned} \pi_h^{ij} \ : L^2(\Gamma_{ij}) &\longrightarrow M_h(\Gamma_{ij}) \\ u \qquad &\longmapsto \pi_h^{ij}(u) \end{aligned}$$

et qui vérifie :

$$\forall (u,v) \in L^2(\Gamma_{ij}) \times M_h(\Gamma_{ij}), \quad \left(\pi_h^{ij}(u), v\right)_{L^2(\Gamma_{ij})} = (u,v)_{L^2(\Gamma_{ij})}$$

Nous noterons γ_{ij}^i et γ_{ij}^j les opérateurs de trace sur Γ_{ij} définis respectivement sur $W_h(\Omega_i)$ et sur $W_h(\Omega_j)$. Les images respectives de ces opérateurs étant clairement incluses dans $L^2(\Gamma_{ij})$, nous pouvons définir notre espace discret global de la façon suivante :

$$X_h = \left\{u_h \in W_h,\ u_h = \left(u_h^i\right)_{1\leq i\leq N},\ \pi_h^{ij}\left(\gamma_{ij}^i\left(u_h^i\right)\right) = \pi_h^{ij}\left(\gamma_{ij}^j\left(u_h^j\right)\right)\ \forall (i,j)\right\}$$

en le munissant du produit scalaire et de la norme de W_h. Cet espace correspond à l'ensemble des fonctions de W_h qui vérifient une condition de continuité faible sur chacune des interfaces Γ_{ij}.

Nous pouvons maintenant définir la forme discrète de notre problème de la façon suivante :

Problème 4.12 : *Trouver une fonction $u_h \in X_h$ telle que $\forall v_h \in X_h$, l'on ait :*

$$a_h(u_h, v_h) = l_h(v_h)$$

où l'on a posé :

$$\begin{aligned} a_h &: X_h \times X_h \longrightarrow \mathbb{R} \\ &(u_h, v_h) \longmapsto a_h(u_h, v_h) = \sum_{i=1}^{N} a_h^i(u_h^i, v_h^i) = \sum_{i=1}^{N} \left(\sum_{K \in \mathcal{T}_h(\Omega_i)} \int_K \nabla u_h^i \cdot \nabla v_h^i \right) \end{aligned}$$

et

$$\begin{aligned} l_h &: X_h \longrightarrow \mathbb{R} \\ &v_h \longmapsto l_h(v_h) = \sum_{i=1}^{N} l_h^i(v_h^i) = \sum_{i=1}^{N} \left(\sum_{K \in \mathcal{T}_h(\Omega_i)} \int_K f v_h^i \right) \end{aligned}$$

Ce problème vérifie le lemme suivant :

Lemme 4.2 : *Le problème 4.12 admet une unique solution qui constitue une approximation consistante de la solution du problème 4.11.*

Démonstration. La consistance du formalisme provient de la consistance de la forme bilinéaire discrète $a_h(\cdot,\cdot)$ avec la forme bilinéaire continue $a(\cdot,\cdot)$ sur $X_h \cap H_0^1(\Omega)$. Par ailleurs, la forme bilinéaire $a_h(\cdot,\cdot)$ est clairement positive sur X_h. Pour établir l'existence et l'unicité de la solution du problème 4.12, il nous suffit donc de montrer qu'elle est également définie. Ainsi, supposons qu'il existe $v_h = \left(v_h^i\right)_{1 \leq i \leq N} \in X_h$ telle que l'on ait $a_h(v_h, v_h) = 0$. La définition de X_h et de $a_h(\cdot,\cdot)$ imposent alors que, pour tout i, v_h^i soit constante sur chacun des éléments de $\mathcal{T}_h(\Omega_i)$. La continuité de v_h^i aux barycentres des faces du maillage permet de voir que, pour tout i, v_h^i est constante sur Ω_i. Enfin, les conditions de raccord qui définissent X_h ainsi que les conditions aux limites discrètes conduisent à la conclusion que $v_h = 0$.

Ainsi, $a_h(\cdot,\cdot)$ est une forme bilinéaire définie positive sur X_h qui est consistante avec la forme continue $a(\cdot,\cdot)$ sur $X_h \cap H_0^1(\Omega)$: le problème 4.12 n'admet donc qu'une unique solution qui est une approximation consistante de celle du problème 4.11. □

Remarquons au passage que cette formulation correspond à une approche primale d'une méthode d'éléments joints. En effet, les conditions de raccord imposées sur les interfaces entre les sous–domaines apparaissent explicitement dans la définition de l'espace X_h. Nous allons maintenant présenter une forme duale de cette méthode, en faisant apparaître des multiplicateurs de LAGRANGE, selon le même principe que celui présenté par F. BEN BELGACEM dans ses travaux [18].

4.2.3 Extension du formalisme

Dans l'écriture du problème 4.12, les contraintes de raccordement apparaissent explicitement dans la définition de l'espace X_h. En ce sens, il s'agit d'une approche primale d'une méthode d'éléments joints. Une telle approche peut s'avérer difficile à mettre en œuvre, car la construction de X_h suppose d'exhiber une base qui vérifie toutes les contraintes de raccordement. Nous développons donc ici une approche duale où ces contraintes n'apparaissent plus dans la définition de l'espace. Le problème 4.12 est équivalent au problème de minimisation suivant :

Problème 4.13 : *Trouver $u_h \in X_h$ réalisant :*

$$\min_{v_h \in X_h} J(v) = \frac{1}{2} a_h(v_h, v_h) - l_h(v_h)$$

Introduisons l'espace suivant :

$$M_h = \prod_{i<j} M_h(\Gamma_{ij}) = \left\{ \phi_h = \left(\phi_h^{ij}\right)_{1 \le i < j \le N},\ \forall i<j, \phi_h^{ij} \in M_h(\Gamma_{ij}) \right\}$$

ainsi que la forme bilinéaire suivante :

$$\begin{aligned} b_h\ &: W_h \times M_h \longrightarrow \mathbb{R} \\ &(v_h, \phi_h) \longmapsto b_h(v_h, \phi_h) = \sum_{1 \le i < j \le N} \left(\phi_h^{ij}\ ,\ \pi_h^{ij}\left(\gamma_{ij}^i\left(v_h^i\right)\right) - \pi_h^{ij}\left(\gamma_{ij}^j\left(v_h^j\right)\right)\right)_{L^2(\Gamma_{ij})} \end{aligned}$$

où $(\cdot\ ,\ \cdot)_{L^2(\Gamma_{ij})}$ désigne le produit scalaire usuel de $L^2(\Gamma_{ij})$. Nous pouvons caractériser X_h comme sous–espace de W_h de la façon suivante :

$$X_h = \left\{ u_h \in W_h,\ \forall \phi_h \in M_h,\ b(u_h, \phi_h) = 0 \right\}$$

Ainsi, le problème 4.13 peut être écrit comme un problème de recherche d'un minimum sous contraintes :

Problème 4.14 : *Trouver $u_h \in W_h$ réalisant :*

$$\min_{\substack{v_h \in W_h \\ \forall \phi_h \in M_h,\ b_h(v_h, \phi_h) = 0}} J(v) = \frac{1}{2} a_h(v_h, v_h) - l_h(v_h)$$

En introduisant des multiplicateurs de LAGRANGE, et en cherchant un point–selle du lagrangien de ce problème contraint, nous établissons la formulation 4.15 dans laquelle les contraintes de raccord aux interfaces n'apparaissent plus dans la définition des espaces.

Problème 4.15 : *Trouver $(u_h, \lambda_h) \in W_h \times M_h$ tel que $\forall (v_h, \mu_h) \in W_h \times M_h$, l'on ait :*

$$\begin{aligned} a(u_h, v_h) + b(v_h, \lambda_h) &= l_h(v_h) \\ b(u_h, \mu_h) &= 0 \end{aligned}$$

Une fois assemblé, le système linéaire associé à ce problème prend la forme suivante :

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} U \\ \Lambda \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}$$

Il peut être inversé à l'aide, par exemple, d'un algorithme de type UZAWA, à l'instar de ce qui est fait dans les schémas de projection utilisés en mécanique des fluides incompressibles. Cependant, une telle approche n'exploite pas le caractère intrinsèquement parallèle de la méthode de décomposition de domaine sous–jacente. Il serait judicieux, en particulier à des fins de préconditionnement, de pouvoir percevoir notre problème comme la résolution couplée de N sous–problèmes. Or il se trouve que la formulation 4.15 n'est pas réellement adaptée à une telle démarche. La raison en est la suivante. Bien que le problème 4.15 soit globalement bien posé, rien n'assure que les problèmes locaux, issus de la décomposition des formes bilinéaires sur chacun des sous–domaines, soient eux aussi bien posés. En particulier, si la partition du domaine fait apparaître un sous–domaine « flottant », c'est–à–dire tel que

$$\partial\Omega_i \cap \partial\Omega = \emptyset$$

alors nous pouvons être sûrs que le problème local associé sera mal posé. En effet, dans la perspective d'un problème local, les multiplicateurs de LAGRANGE se comportent comme des conditions aux limites de NEUMANN. Ainsi, un sous–domaine flottant conduit à la construction d'un problème local de LAPLACE n'ayant que des conditions aux limites de NEUMANN. Or, un tel problème admet une infinité de solutions, définies à une constante près. Par conséquent, la formulation 4.15 conduit à la construction d'un problème global bien posé, bien que les problèmes locaux sous–jacents puissent eux être mal posés. Afin de résoudre cette difficulté, nous nous proposons de transformer notre problème de façon à ce que les problèmes locaux perçoivent sur les interfaces des conditions de type DIRICHLET, au lieu de conditions de type NEUMANN.

Pour ce faire, nous nous proposons de redéfinir les conditions de raccord imposées sur les interfaces séparant les sous–domaines. Considérons donc une telle interface Γ_{ij} séparant les sous–domaines Ω_i et Ω_j. Dans la formulation 4.15, la continuité d'une fonction u_h de W_h sur Γ_{ij} est imposée en un sens faible, à savoir qu'on impose au saut des projections sur $M_h(\Gamma_{ij})$ des traces de u_h^i et u_h^j sur Γ_{ij} d'être orthogonal à toute fonction de $M_h(\Gamma_{ij})$. Une autre façon d'imposer ce genre de contrainte consiste à procéder comme suit.

Définissons tout d'abord les espaces de fonctions suivants :

$$\begin{aligned}
M_h^i(\Gamma_{ij}) &= \left\{ u \in L^2(\Gamma_{ij}),\ \forall K \in \mathscr{T}_h^i(\Gamma_{ij}),\ u \in \mathbb{P}^0(K) \right\} \\
M_h^j(\Gamma_{ij}) &= \left\{ u \in L^2(\Gamma_{ij}),\ \forall K \in \mathscr{T}_h^j(\Gamma_{ij}),\ u \in \mathbb{P}^0(K) \right\} \\
M_h(\Gamma_{ij}) &= M_h^\eta(\Gamma_{ij})
\end{aligned}$$

où, comme précédemment, on a posé arbitrairement $\eta = \min(i, j)$. Introduisons également les opérateurs de projection au sens de $L^2(\Gamma_{ij})$ sur $M_h^i(\Gamma_{ij})$ et $M_h^j(\Gamma_{ij})$ définis par :

$$\begin{array}{rcl@{\qquad}rcl}
\pi_{ij}^i : L^2(\Gamma_{ij}) & \longrightarrow & M_h^i(\Gamma_{ij}) & \text{et} & \pi_{ij}^j : L^2(\Gamma_{ij}) & \longrightarrow & M_h^j(\Gamma_{ij}) \\
u & \longmapsto & \pi_{ij}^i(u) & & u & \longmapsto & \pi_{ij}^j(u)
\end{array}$$

qui vérifient, pour k valant i ou j :

$$\forall (u, v) \in L^2(\Gamma_{ij}) \times M_h^k(\Gamma_{ij}), \quad \left(\pi_{ij}^k(u), v \right)_{L^2(\Gamma_{ij})} = (u, v)_{L^2(\Gamma_{ij})}$$

La condition de continuité au sens faible sur Γ_{ij} d'une fonction u_h de W_h peut alors se traduire par l'existence d'une fonction $u_{\Gamma_{ij}} \in M_h(\Gamma_{ij})$ telle que les deux conditions suivantes soient vérifiées :

– le saut entre la projection sur $M_h^i(\Gamma_{ij})$ de la trace de u_h^i sur Γ_{ij} et la projection sur $M_h^i(\Gamma_{ij})$ de $u_{\Gamma_{ij}}$ est orthogonal à toute fonction de $M_h^i(\Gamma_{ij})$;
– le saut entre la projection sur $M_h^j(\Gamma_{ij})$ de la trace de u_h^j sur Γ_{ij} et la projection sur $M_h^j(\Gamma_{ij})$ de $u_{\Gamma_{ij}}$ est orthogonal à toute fonction de $M_h^j(\Gamma_{ij})$.

Pour généraliser cette idée à chacune des interfaces Γ_{ij}, nous introduisons les espaces suivants :

$$L_h^i = \prod_{j\neq i} M_h^i(\Gamma_{ij}) = \left\{ \mu_h^i = \left(\mu_h^{i,j}\right)_{j\neq i},\ \forall j,\ \mu_h^{i,j} \in M_h^i(\Gamma_{ij}) \right\}$$

$$L_h = \prod_{i=1}^{N} L_h^i = \left\{ \mu_h = \left(\mu_h^i\right)_{1\leq i\leq N},\ \forall i,\ \mu_h^i \in L_h^i \right\}$$

$$M_h = \prod_{i=1}^{N} \left(\prod_{j<i} M_h^i(\Gamma_{ij}) \right) = \left\{ u_\Gamma = \left(u_{\Gamma_{ij}}\right)_{1\leq i<j\leq N},\ \forall i<j,\ u_{\Gamma_{ij}} \in M_h^i(\Gamma_{ij}) \right\}$$

Définissons également les formes bilinéaires suivantes :

$$b_h : L_h \times W_h \longrightarrow \mathbb{R}$$

$$(\mu_h, v_h) \longmapsto b_h(\mu_h, v_h) = \sum_{i=1}^{N} b_h^i(\mu_h^i, v_h^i) = \sum_{i=1}^{n} \left(\sum_{j\neq i} \left(\mu_h^{i,j}\ ,\ \pi_{ij}^i \left(\gamma_{ij}^i \left(v_h^i \right) \right) \right)_{L^2(\Gamma_{ij})} \right)$$

et

$$c_h : L_h \times M_h \longrightarrow \mathbb{R}$$

$$\begin{aligned}(\mu_h, v_\Gamma) \longmapsto c_h(\mu_h, v_\Gamma) &= \sum_{i=1}^{N} c_h^i(\mu_h^i, v_\Gamma) = \sum_{i=1}^{N} \left(\sum_{j\neq i} \left(\mu_h^{i,j}\ ,\ \pi_{ij}^i \left(v_{\Gamma_{ij}} \right) \right)_{L^2(\Gamma_{ij})} \right) \\ &= \sum_{1\leq i<j\leq N} \left[\left(\mu_h^{i,j}\ ,\ \pi_{ij}^i \left(v_{\Gamma_{ij}} \right) \right)_{L^2(\Gamma_{ij})} + \left(\mu_h^{j,i}\ ,\ \pi_{ij}^j \left(v_{\Gamma_{ij}} \right) \right)_{L^2(\Gamma_{ij})} \right]\end{aligned}$$

En utilisant ces définitions ainsi que celles données plus haut, nous pouvons donner une formulation variationnelle de notre problème qui correspond à nos attentes :

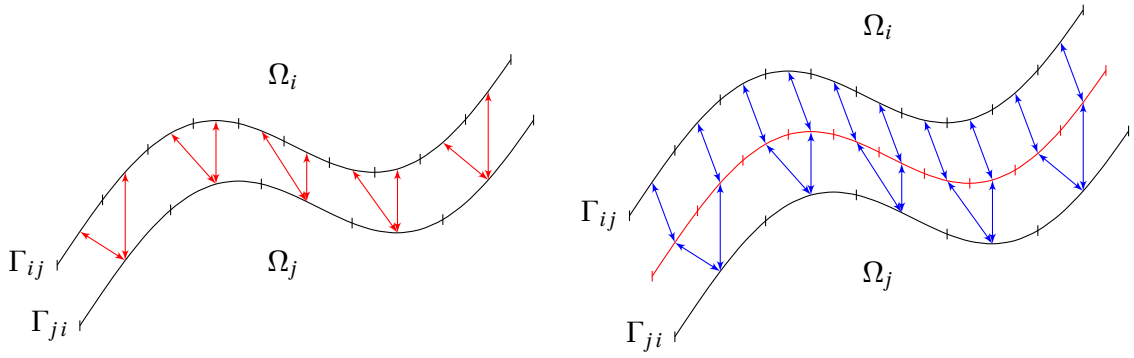
Problème 4.16 : *Trouver* $(u_h, \lambda_h, u_\Gamma) \in W_h \times L_h \times M_h$ *tel que* $\forall \left(v_h, \mu_h, v_\Gamma\right) \in W_h \times L_h \times M_h$, *l'on ait :*

$$\begin{aligned} a_h(u_h, v_h) + b_h(\lambda_h, v_h) &= l_h(v_h) \\ b_h(\mu_h, u_h) - c_h(\mu_h, u_\Gamma) &= 0 \\ -c_h(\lambda_h, v_\Gamma) &= 0 \end{aligned}$$

Une fois assemblé, le système linéaire associé à ce problème discret prend la forme suivante :

$$\begin{bmatrix} A & B^T & 0 \\ B & 0 & -C^T \\ 0 & -C & 0 \end{bmatrix} \begin{bmatrix} U \\ \Lambda \\ U_\Gamma \end{bmatrix} = \begin{bmatrix} F \\ 0 \\ 0 \end{bmatrix}$$

La figure 4.6 illustre les différences entre l'approche duale classique issue du formalisme de F. BEN BELGACEM et l'approche que nous proposons. Sur la figure 4.6(a), on a repré-



(a) Approche duale « standard »

(b) Approche duale modifiée

FIG. 4.6 : Différences entre les deux approches duales de la méthode d'éléments joints pour l'élément fini de CROUZEIX–RAVIART

senté en rouge le couplage entre les degrés de libertés du sous–domaine Ω_i situés sur Γ_{ij} et ceux du sous–domaine Ω_j situés sur Γ_{ji}. Ce couplage est réalisé par l'introduction des multiplicateurs de LAGRANGE. Sur la figure 4.6(b), on a représenté en rouge la discrétisation de l'interface, et en bleu les couplages entre les degrés de libertés du sous–domaine Ω_i (resp. Ω_j), situés sur Γ_{ij} (resp. Γ_{ji}) et ceux situés sur l'interface.

4.2.4 Application aux maillages issus du raffinement par subdivision

Dans les paragraphes qui suivent, nous nous proposons d'expliciter et de simplifier la forme matricielle de notre problème discret. Pour ce faire, nous tiendrons compte des propriétés de l'élément fini de CROUZEIX–RAVIART, ainsi que de la forme particulière des non–conformités générées par notre algorithme d'adaptation de maillage. Mais donnons tout d'abord une expression plus détaillée de notre système linéaire :

$$\left[\begin{array}{cc|ccc|c|ccc|c}
A_{II}^1 & A_{I\Gamma}^{1T} & & & & & 0 & & & \\
A_{I\Gamma}^1 & A_{\Gamma\Gamma}^1 & & & & & B_\Gamma^{1T} & & & \\
\hline
 & & \ddots & & & & & \ddots & & \\
\hline
 & & & A_{II}^N & A_{I\Gamma}^{NT} & & & 0 & & \\
 & & & A_{I\Gamma}^N & A_{\Gamma\Gamma}^N & & & B_\Gamma^{NT} & & \\
\hline
0 & B_\Gamma^1 & & & & & & & -C_\Gamma^{1T} & \\
\hline
 & & \ddots & & & & & & \vdots & \\
\hline
 & & & 0 & B_\Gamma^N & & & & -C_\Gamma^{NT} & \\
\hline
 & & & & & -C_\Gamma^1 & \dots & -C_\Gamma^N & &
\end{array}\right]
\left[\begin{array}{c} U_I^1 \\ U_\Gamma^1 \\ \vdots \\ U_I^N \\ U_\Gamma^N \\ \Lambda^1 \\ \vdots \\ \Lambda^N \\ U_\Gamma \end{array}\right]
=
\left[\begin{array}{c} F_I^1 \\ F_\Gamma^1 \\ \vdots \\ F_I^N \\ F_\Gamma^N \\ 0 \\ \vdots \\ 0 \\ 0 \end{array}\right]$$

On a ici noté :

- U_I^k la valeur de l'inconnue sur les degrés de libertés internes au sous–domaine Ω_k ;
- U_Γ^k la valeur de l'inconnue sur les degrés de libertés du sous–domaine Ω_k situés sur une interface ;
- Λ^k les multiplicateurs de LAGRANGE associés aux interfaces du sous–domaine Ω_k.

Nous pouvons d'ores–et–déjà simplifier cette expression en tenant compte de la définition de l'élément fini de CROUZEIX–RAVIART ainsi que de celle de la forme bilinéaire

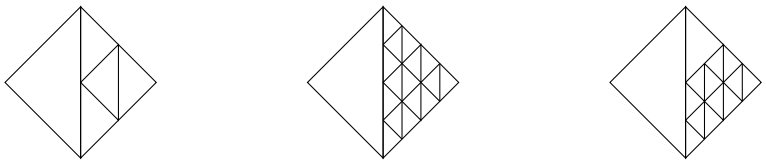


FIG. 4.7 : Quelques situations générables par notre algorithme adaptatif en dimension 2

$b_h(\cdot,\cdot)$. Nous constatons en effet de façon immédiate que les matrices B_Γ^k sont égales à l'identité, car elles associent à l'inconnue U_Γ^k sa valeur moyenne sur les faces des interfaces, ce qui est la définition du degré de liberté de l'élément fini de CROUZEIX–RAVIART sur ces faces. Nous pouvons alors poursuivre notre simplification, en éliminant les inconnues U_Γ^k ainsi que les multiplicateurs de LAGRANGE Λ^k par une élimination de GAUSS. Après manipulation, il vient :

$$\begin{bmatrix} A_{II}^1 & & & A_{I\Gamma}^{1T} C_\Gamma^{1T} \\ & \ddots & & \vdots \\ & & A_{II}^N & A_{I\Gamma}^{NT} C_\Gamma^{NT} \\ C_\Gamma^1 A_{I\Gamma}^1 & \dots & C_\Gamma^N A_{I\Gamma}^N & E_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} U_I^1 \\ \vdots \\ U_I^N \\ U_\Gamma \end{bmatrix} = \begin{bmatrix} F_I^1 \\ \vdots \\ F_I^N \\ F_\Gamma \end{bmatrix}$$

où l'on a posé :

$$E_{\Gamma\Gamma} = \sum_{k=1}^{N} C_\Gamma^k A_{\Gamma\Gamma}^k C_\Gamma^{kT}$$

$$F_\Gamma = \sum_{k=1}^{N} C_\Gamma^k F_\Gamma^k$$

Notons que la structure de cette matrice est extrêmement proche de celle que l'on obtient lorsque l'on utilise une méthode de SCHUR (voir paragraphe 2.2.2).

Pour continuer à simplifier notre problème, nous tenons compte de la forme particulière des non–conformités générées par notre algorithme d'adaptation de maillage. Cette démarche nous permettra de construire explicitement les matrices de couplages C_Γ^k. Rappelons que nous avons opté pour une méthode de subdivision. L'effet d'une telle méthode sur les faces du maillage est très clairement identifiable : en dimension 2 (resp. 3), les faces du maillage se trouvent découpées en 2 (resp. 4). La figure 4.7 présente diverses situations potentiellement générées par notre algorithme adaptatif en dimension 2. On constate que les non–conformités générées par notre algorithme présenteront toujours d'un côté une face dite « grossière », et de l'autre, un ensemble de faces dites « fines», formant un recouvrement de la face grossière. Les qualificatifs « grossières » et « fines » se rapportent à la surface des faces de part et d'autre de la non–conformité. Comme on peut le voir sur le troisième cas de la figure 4.7, les différentes faces fines d'une non–conformité ne présentent pas nécessairement le même niveau de raffinement. Une propriété fondamentale des non–conformités générées par notre algorithme est la suivante :

Propriété 4.1 : *Considérons une non–conformité faisant intervenir une face grossière F et n faces fines $(f_i)_{1\leq i\leq n}$. Notons r_F et r_{f_i} les niveaux de raffinement des sous–domaines*

auxquels appartiennent respectivement F et chacune des f_i. Par définition, on a $r_F < r_{f_i}$, pour tout i. Notons S_F et S_{f_i} les surfaces respectives de la face F et de chacune des faces f_i. Enfin, posons $\alpha = 2$ en dimension 2, et $\alpha = 4$ en dimension 3. Avec ces notations, on a toujours :

$$\frac{S_{f_i}}{S_F} = \alpha^{r_F - r_{f_i}} \ (<1)$$

De plus, comme les faces fines forment un recouvrement de la face grossière, on a toujours :

$$\sum_{i=1}^{n} \alpha^{r_F - r_{f_i}} = 1$$

Cette propriété va nous permettre d'expliciter les coefficients des matrices C_Γ^k.

Par souci de simplicité, nous choisissons de trier les sous–domaines par ordre décroissant de niveau de raffinement. Ainsi, pour chaque interface, nous choisissons implicitement la discrétisation la plus fine comme discrétisation maître. Rappelons que C_Γ^k provient de la forme bilinéaire :

$$\begin{aligned} c_h^k \ : L_h^k \times M_h &\longrightarrow \mathbb{R} \\ (\mu_h^k, v_\Gamma) &\longmapsto c_h^k(\mu_h^k, v_\Gamma) = \sum_{i \neq k} \left(\mu_h^{k,i} \ , \ \pi_{ki}^k \left(v_{\Gamma_{ki}} \right) \right)_{L^2(\Gamma_{ki})} \end{aligned}$$

En notant $(\theta_i)_i$ et $\left(\psi_j\right)_j$ les fonctions de base respectives des espaces L_h^k et M_h, nous avons :

$$\left(C_\Gamma^k\right)_{ij} = c_h^k \left(\theta_i, \psi_j\right)$$

Notons F_i et F_j les support respectifs des fonctions θ_i et ψ_j. Il vient alors :

$$\left(C_\Gamma^k\right)_{ij} = \frac{1}{|F_i|} \int_{F_i \cap F_j} 1 = \frac{\left| \left(F_i \cap F_j \right) \right|}{|F_i|}$$

Ainsi, nous pouvons nous trouver dans 3 cas de figures :

i. $F_i \cap F_j = \emptyset$. Dans ce cas, $\left(C_\Gamma^k\right)_{ij} = 0$.

ii. $F_i \cap F_j = F_i$. Nous nous trouvons dans ce cas quand la face du sous–domaine Ω_k est une face fine du point de vue de la non–conformité. Nous avons alors :

$$\left(C_\Gamma^k\right)_{ij} = 1$$

iii. $F_i \cap F_j = F_j \subsetneq F_i$. Nous nous trouvons dans ce cas quand la face du sous–domaine Ω_k est une face grossière du point de vue de la non–conformité. En conservant les mêmes notations que précédemment, il vient :

$$\left(C_\Gamma^k\right)_{ij} = \alpha^{r_{F_i} - r_{F_j}}$$

En résumé, la formulation variationnelle 4.16 fournit une version discrète du problème 4.11 qui présente les caractéristiques suivantes :

– elle supporte l'utilisation d'un maillage généré par notre algorithme d'adaptation géométrique ;

– elle peut être perçue comme la résolution couplée de N problèmes de LAPLACE munis de conditions de DIRICHLET sur les interfaces séparant les sous–domaines ;
– l'expression des matrices de couplage C_Γ^k est donnée explicitement par la géométrie des interfaces.

La seconde propriété permet d'assurer, contrairement à la formulation 4.15, que les problèmes locaux seront toujours bien posés au sens de J. HADAMARD. Enfin, la troisième propriété permet de concevoir une implémentation efficace de ce problème discret car elle permet de s'affranchir d'un assemblage complet des termes de couplage.

4.2.5 Application à la méthode VEF sur un problème de STOKES

Dans les paragraphes qui suivent, nous utilisons les arguments établis ci–dessus pour généraliser la résolution d'un problème de STOKES à l'aide de la méthode VEF sur un maillage généré par notre algorithme d'adaptation. Nous étudions donc le problème suivant :

Problème 4.17 : *Trouver* $(\boldsymbol{u},p)\in H_0^1(\Omega)^n\times L_0^2(\Omega)$ *tel que* $\forall(\boldsymbol{v},q)\in H_0^1(\Omega)^n\times L_0^2(\Omega)$, *l'on ait :*

$$\begin{aligned} a(\boldsymbol{u},\boldsymbol{v})-b(\boldsymbol{v},p)&=l(\boldsymbol{v})\\ b(\boldsymbol{u},q)&=0 \end{aligned}$$

où $L_0^2(\Omega)$ *désigne l'ensemble des fonctions de* $L^2(\Omega)$, *à valeur moyenne nulle sur* Ω, *et où l'on a posé :*

$$a(\boldsymbol{u},\boldsymbol{v})=\int_\Omega \nabla\boldsymbol{u}:\nabla\boldsymbol{v} \qquad b(\boldsymbol{v},q)=\int_\Omega q\nabla\cdot\boldsymbol{u} \qquad l(\boldsymbol{v})=\int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}$$

Supposons que le domaine Ω soit recouvert par un maillage généré par notre algorithme d'adaptation géométrique. Autrement dit, supposons qu'il existe une famille de sous–domaines $(\Omega_i)_{1\leq i\leq N}$ de Ω formant une partition de ce dernier :

$$\bigcup_{i=1}^{N}\overline{\Omega}_i=\overline{\Omega} \qquad (i\neq j)\Longrightarrow\left(\mathring{\Omega}_i\cap\mathring{\Omega}_j=\emptyset\right)$$

Supposons également que chacun de ces sous–domaines soit muni d'une triangulation conforme $\mathscr{T}_h(\Omega_i)$, et que les non–conformités entre sous–domaines présentent les caractéristiques énumérées précédemment. Notre objectif consiste à établir une formulation variationnelle discrète du problème 4.17 qui présente les mêmes caractéristiques que celles établies pour le cas du problème de LAPLACE.

Remarquons que la méthode VEF de Trio-U ne suppose pas la continuité du champ des pressions discrètes p_h. Ainsi, nous ne chercherons à imposer de contrainte de continuité aux interfaces que sur le champ des vitesses discrètes $\boldsymbol{u}_h$, et nous traiterons le champ des pressions discrètes de façon indépendante sur chacun des sous–domaines. Définissons donc les espaces suivants :

$$\begin{aligned}\boldsymbol{W}_h(\Omega_i)=\big\{&\boldsymbol{u}_h^i\in L^2(\Omega_i)^n\ ;\ \forall K\in\mathscr{T}_h(\Omega_i)\ ,\ \boldsymbol{u}_h^i|_K\in\left(\mathbb{P}^1(K)\right)^n\ ;\\ &\forall x\in\mathscr{N}^I(\mathscr{T}_h(\Omega_i))\ ,\ \boldsymbol{u}_h^i \text{ continue en } x\ ;\ \forall x\in\mathscr{N}^\partial(\mathscr{T}_h(\Omega_i))\ ,\ \boldsymbol{u}_h^i(x)=0\big\}\end{aligned}$$

ainsi que les normes et semi–normes brisées suivantes :

$$\left\|\boldsymbol{u}_h^i\right\|_{\boldsymbol{W}_h(\Omega_i)} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left\|\boldsymbol{u}_h^i\right\|^2_{H^1(K)^n} \right)^{\frac{1}{2}} \qquad \left|\boldsymbol{u}_h^i\right|_{\boldsymbol{W}_h(\Omega_i)} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left|\boldsymbol{u}_h^i\right|^2_{H^1(K)^n} \right)^{\frac{1}{2}}$$

Nous pouvons maintenant définir un espace global pour les vitesses discrètes :

$$\boldsymbol{W}_h = \prod_{i=1}^{N} \boldsymbol{W}_h(\Omega_i) = \left\{ \boldsymbol{u}_h = \left(\boldsymbol{u}_h^i\right)_{1\le i\le n},\ \boldsymbol{u}_h^i \in \boldsymbol{W}_h(\Omega_i) \right\}$$

muni de la norme et de la semi–norme suivante :

$$\|\boldsymbol{u}_h\|_{\boldsymbol{W}_h} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left\|\boldsymbol{u}_h^i\right\|^2_{\boldsymbol{W}_h(\Omega_i)} \right)^{\frac{1}{2}} \qquad |\boldsymbol{u}_h|_{W_h} = \left(\sum_{K\in\mathscr{T}_h(\Omega_i)} \left|\boldsymbol{u}_h^i\right|^2_{\boldsymbol{W}_h(\Omega_i)} \right)^{\frac{1}{2}}$$

Nous pouvons percevoir $\boldsymbol{W}_h$ comme l'ensemble des fonctions de $L^2(\Omega)^n$ dont la restriction à chacun des Ω_i appartient à l'espace $\boldsymbol{W}_h(\Omega_i)$. Introduisons maintenant les espaces de discrétisation liés aux interfaces entre sous–domaines. Pour chaque interface Γ_{ij}, nous posons :

$$\begin{aligned}
\boldsymbol{M}_h^i(\Gamma_{ij}) &= \left\{ \boldsymbol{u} \in L^2(\Gamma_{ij})^n,\ \forall K \in \mathscr{T}_h^i(\Gamma_{ij}),\ \boldsymbol{u} \in \left(\mathbb{P}^0(K)\right)^n \right\} \\
\boldsymbol{M}_h^j(\Gamma_{ij}) &= \left\{ \boldsymbol{u} \in L^2(\Gamma_{ij})^n,\ \forall K \in \mathscr{T}_h^j(\Gamma_{ij}),\ \boldsymbol{u} \in \left(\mathbb{P}^0(K)\right)^n \right\} \\
\boldsymbol{M}_h(\Gamma_{ij}) &= \boldsymbol{M}_h^\eta(\Gamma_{ij})
\end{aligned}$$

et nous définissons :

$$\begin{aligned}
\boldsymbol{L}_h^i &= \prod_{j\neq i} \boldsymbol{M}_h^i(\Gamma_{ij}) = \left\{ \boldsymbol{\mu}_h^i = \left(\boldsymbol{\mu}_h^{i,j}\right)_{j\neq i},\ \forall j,\ \boldsymbol{\mu}_h^{i,j} \in \boldsymbol{M}_h^i(\Gamma_{ij}) \right\} \\
\boldsymbol{L}_h &= \prod_{i=1}^{N} \boldsymbol{L}_h^i = \left\{ \boldsymbol{\mu}_h = \left(\boldsymbol{\mu}_h^i\right)_{1\le i\le N},\ \forall i,\ \boldsymbol{\mu}_h^i \in \boldsymbol{L}_h^i \right\} \\
\boldsymbol{M}_h &= \prod_{i=1}^{N} \left(\prod_{j<i} \boldsymbol{M}_h^i(\Gamma_{ij}) \right) = \left\{ \boldsymbol{u}_\Gamma = \left(\boldsymbol{u}_{\Gamma_{ij}}\right)_{1\le i<j\le N},\ \forall i<j,\ \boldsymbol{u}_{\Gamma_{ij}} \in \boldsymbol{M}_h^i(\Gamma_{ij}) \right\}
\end{aligned}$$

Les espaces $\boldsymbol{W}_h$, $\boldsymbol{L}_h$, et $\boldsymbol{M}_h$ constituent les versions vectorielles des espaces W_h, L_h, et M_h construits pour le problème de LAPLACE. Pour définir l'espace des pressions discrètes, nous introduisons pour chaque sous–domaine Ω_i les espaces auxiliaires suivants :

$$\begin{aligned}
Z_h^0(\Omega_i) &= \left\{ q \in L^2(\Omega_i)\ ;\ \forall K \in \mathscr{T}_h(\Omega_i),\ q|_K \in \mathbb{P}^0(K) \right\} \\
Z_h^1(\Omega_i) &= \left\{ q \in \mathscr{C}^0(\Omega_i)\ ;\ \forall K \in \mathscr{T}_h(\Omega_i),\ q|_K \in \mathbb{P}^1(K) \right\} \\
X_h^0(\Omega_i) &= Z_h^0(\Omega_i) \cap L_0^2(\Omega_i) \\
X_h^1(\Omega_i) &= (Z_h^0(\Omega_i) + Z_h^1(\Omega_i)) \cap L_0^2(\Omega_i) = \left(Z_h^0(\Omega_i) \cap L_0^2(\Omega_i)\right) + \left(Z_h^1(\Omega_i) \cap L_0^2(\Omega_i)\right)
\end{aligned}$$

Nous posons alors :

$$X_h^0 = \prod_{i=1}^{N} X_h^0(\Omega_i) = \left\{ q_h^0 = \left(q_{h,i}^0\right)_{1\le i\le N},\ q_{h,i}^0 \in X_h^0(\Omega_i) \right\}$$
$$X_h^1 = \prod_{i=1}^{N} X_h^1(\Omega_i) = \left\{ q_h^1 = \left(q_{h,i}^1\right)_{1\le i\le N},\ q_{h,i}^1 \in X_h^1(\Omega_i) \right\}$$

Remarquons que toute fonction q_h appartenant à X_h^1 peut s'écrire de la façon suivante :

$$q_h = q_h^0 + q_h^1 \quad \text{avec} \quad q_h^0 \in X_h^0 \quad \text{et} \quad q_h^1 \in \prod_{i=1}^{N} \left(Z_h^1(\Omega_i) \cap L_0^2(\Omega_i) \right)$$

Définissons maintenant les formes bilinéaires et les formes linéaires nécessaires à l'établissement de notre formulation variationnelle. Pour cela, nous noterons γ_{ij}^i et γ_{ij}^j les opérateurs de trace sur Γ_{ij} définis respectivement sur $\boldsymbol{W}_h(\Omega_i)$ et $\boldsymbol{W}_h(\Omega_j)$, et tous deux à valeurs dans $L^2(\Gamma_{ij})^n$. Nous introduisons également π_{ij}^i et π_{ij}^j, les opérateurs de projection au sens de $L^2(\Gamma_{ij})^n$ respectivement sur $\boldsymbol{M}_h^i(\Gamma_{ij})$ et $\boldsymbol{M}_h^j(\Gamma_{ij})$:

$$\begin{array}{cccc} \pi_{ij}^i : L^2(\Gamma_{ij})^n \longrightarrow \boldsymbol{M}_h^i(\Gamma_{ij}) & \text{et} & \pi_{ij}^j : L^2(\Gamma_{ij})^n \longrightarrow \boldsymbol{M}_h^j(\Gamma_{ij}) \\ \boldsymbol{u} \longmapsto \pi_{ij}^i(\boldsymbol{u}) & & \boldsymbol{u} \longmapsto \pi_{ij}^j(\boldsymbol{u}) \end{array}$$

qui vérifient, pour k valant i ou j :

$$\forall (\boldsymbol{u},\boldsymbol{v}) \in L^2(\Gamma_{ij})^n \times \boldsymbol{M}_h^k(\Gamma_{ij}), \quad \left(\pi_{ij}^k(\boldsymbol{u}),\boldsymbol{v}\right)_{L^2(\Gamma_{ij})^n} = (\boldsymbol{u},\boldsymbol{v})_{L^2(\Gamma_{ij})^n}$$

Nous posons maintenant :

$$a_h : \boldsymbol{W}_h \times \boldsymbol{W}_h \longrightarrow \mathbb{R}$$
$$(\boldsymbol{u}_h,\boldsymbol{v}_h) \longmapsto a_h(\boldsymbol{u}_h,\boldsymbol{v}_h) = \sum_{i=1}^{N} a_h^i(\boldsymbol{u}_h^i,\boldsymbol{v}_h^i) = \sum_{i=1}^{n} \left(\sum_{K\in\mathcal{T}_h(\Omega_i)} \int_K \nabla \boldsymbol{u}_h : \nabla \boldsymbol{v}_h \right)$$

$$b_h^0 : \boldsymbol{W}_h \times X_h^0 \longrightarrow \mathbb{R}$$
$$(\boldsymbol{u}_h,q_h) \longmapsto b_h^0(\boldsymbol{v}_h,q_h) = \sum_{i=1}^{N} b_{h,i}^0(\boldsymbol{v}_h^i,q_{h,i}^0) = \sum_{i=1}^{N} \left(\sum_{K\in\mathcal{T}_h(\Omega_i)} \int_K q_{h,i}^0 \nabla\cdot \boldsymbol{u}_h^i \right)$$

$$b_h^1 : \boldsymbol{W}_h \times X_h^1 \longrightarrow \mathbb{R}$$
$$(\boldsymbol{u}_h,q_h) \longmapsto b_h^1(\boldsymbol{v}_h,q_h) = \sum_{i=1}^{N} b_{h,i}^1(\boldsymbol{v}_h^i, q_{h,i}^0 + q_{h,i}^1)$$
$$= \sum_{i=1}^{N} \left(\sum_{K\in\mathcal{T}_h(\Omega_i)} \int_K q_{h,i}^0 \nabla\cdot \boldsymbol{u}_h^i + \sum_{F\in\mathcal{T}_h(\Omega_i)} \int_F \left[\!\left[\boldsymbol{v}_h^i \cdot \boldsymbol{n}_F \right]\!\right] q_{h,i}^1 d\sigma \right)$$

$$c_h : \boldsymbol{L}_h \times \boldsymbol{W}_h \longrightarrow \mathbb{R}$$

$$(\boldsymbol{\mu}_h, \boldsymbol{v}_h) \longmapsto c_h(\boldsymbol{\mu}_h, \boldsymbol{v}_h) = \sum_{i=1}^{N} c_h^i(\boldsymbol{\mu}_h^i, \boldsymbol{v}_h^i) = \sum_{i=1}^{n} \left(\sum_{j \neq i} \left(\boldsymbol{\mu}_h^{i,j} \, , \, \pi_{ij}^i \left(\gamma_{ij}^i \left(\boldsymbol{v}_h^i \right) \right) \right)_{L^2(\Gamma_{ij})^n} \right)$$

$$d_h : \boldsymbol{L}_h \times \boldsymbol{M}_h \longrightarrow \mathbb{R}$$

$$\begin{aligned}(\boldsymbol{\mu}_h, \boldsymbol{v}_\Gamma) \longmapsto d_h(\boldsymbol{\mu}_h, \boldsymbol{v}_\Gamma) &= \sum_{i=1}^{N} d_h^i(\boldsymbol{\mu}_h^i, \boldsymbol{v}_\Gamma) = \sum_{i=1}^{N} \left(\sum_{j \neq i} \left(\boldsymbol{\mu}_h^{i,j} \, , \, \pi_{ij}^i \left(\boldsymbol{v}_{\Gamma_{ij}} \right) \right)_{L^2(\Gamma_{ij})^n} \right) \\ &= \sum_{1 \leq i < j \leq N} \left[\left(\boldsymbol{\mu}_h^{i,j} \, , \, \pi_{ij}^i \left(\boldsymbol{v}_{\Gamma_{ij}} \right) \right)_{L^2(\Gamma_{ij})^n} + \left(\boldsymbol{\mu}_h^{j,i} \, , \, \pi_{ij}^j \left(\boldsymbol{v}_{\Gamma_{ij}} \right) \right)_{L^2(\Gamma_{ij})^n} \right]\end{aligned}$$

et enfin :

$$l_h : \boldsymbol{W}_h \longrightarrow \mathbb{R}$$

$$\boldsymbol{v}_h \longmapsto l_h(\boldsymbol{v}_h) = \sum_{i=1}^{N} l_h^i(\boldsymbol{v}_h^i) = \sum_{i=1}^{n} \left(\sum_{K \in \mathcal{T}_h(\Omega_i)} \int_K \boldsymbol{f} \cdot \boldsymbol{v}_h^i \right)$$

À l'aide de ces définitions, nous pouvons établir les formulations variationnelles associées aux généralisations des méthodes VEF standard et étendue de Trio-U lors de l'utilisation d'un maillage provenant de notre algorithme d'adaptation. La généralisation de la méthode VEF standard s'écrit :

Problème 4.18 : *Trouver* $(\boldsymbol{u}_h, p_h, \boldsymbol{\lambda}_h, \boldsymbol{u}_\Gamma) \in \boldsymbol{W}_h \times X_h^0 \times \boldsymbol{L}_h \times \boldsymbol{M}_h$ *tel que* $\forall (\boldsymbol{v}_h, q_h, \boldsymbol{\mu}_h, \boldsymbol{v}_\Gamma) \in \boldsymbol{W}_h \times X_h^0 \times \boldsymbol{L}_h \times \boldsymbol{M}_h$*, l'on ait :*

$$\begin{aligned} a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) - b_h^0(\boldsymbol{v}_h, p_h) + c_h(\boldsymbol{\lambda}_h, \boldsymbol{v}_h) &= l_h(\boldsymbol{v}_h) \\ b_h^0(\boldsymbol{u}_h, q_h) &= 0 \\ c_h(\boldsymbol{\mu}_h, \boldsymbol{u}_h) - d_h(\boldsymbol{\mu}_h, \boldsymbol{u}_\Gamma) &= 0 \\ -d_h(\boldsymbol{\lambda}_h, \boldsymbol{v}_\Gamma) &= 0 \end{aligned}$$

De la même façon, la généralisation de la méthode VEF étendue s'écrit :

Problème 4.19 : *Trouver* $(\boldsymbol{u}_h, p_h, \boldsymbol{\lambda}_h, \boldsymbol{u}_\Gamma) \in \boldsymbol{W}_h \times X_h^1 \times \boldsymbol{L}_h \times \boldsymbol{M}_h$ *tel que* $\forall (\boldsymbol{v}_h, q_h, \boldsymbol{\mu}_h, \boldsymbol{v}_\Gamma) \in \boldsymbol{W}_h \times X_h^1 \times \boldsymbol{L}_h \times \boldsymbol{M}_h$*, l'on ait :*

$$\begin{aligned} a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) - b_h^1(\boldsymbol{v}_h, p_h) + c_h(\boldsymbol{\lambda}_h, \boldsymbol{v}_h) &= l_h(\boldsymbol{v}_h) \\ b_h^1(\boldsymbol{u}_h, q_h) &= 0 \\ c_h(\boldsymbol{\mu}_h, \boldsymbol{u}_h) - d_h(\boldsymbol{\mu}_h, \boldsymbol{u}_\Gamma) &= 0 \\ -d_h(\boldsymbol{\lambda}_h, \boldsymbol{v}_\Gamma) &= 0 \end{aligned}$$

Les remarques effectuées pour le problème de LAPLACE concernant la simplification du problème restent valables pour le problème de STOKES. En effet, les termes de couplages n'interviennent que sur le champ de vitesse, qui présentent les mêmes caractéristiques que le champ inconnu du problème de LAPLACE. Après simplification, donc, le système

linéaire associé au problème de STOKES peut prendre l'une des deux formes suivantes, qui sont en réalité équivalentes :

$$\left[\begin{array}{cc|c|cc|c} A_{II}^1 & B_I^{1T} & & & & A_{I\Gamma}^{1T}C_\Gamma^{1T} \\ B_I^1 & 0 & & & & B_\Gamma^1 C_\Gamma^{1T} \\ \hline & & \ddots & & & \vdots \\ \hline & & & A_{II}^N & B_I^{NT} & A_{I\Gamma}^{NT}C_\Gamma^{NT} \\ & & & B_I^N & 0 & B_\Gamma^N C_\Gamma^{NT} \\ \hline C_\Gamma^1 A_{I\Gamma}^1 & C_\Gamma^1 B_\Gamma^{1T} & \dots & C_\Gamma^N A_{I\Gamma}^N & C_\Gamma^N B_\Gamma^{NT} & E_{\Gamma\Gamma} \end{array}\right] \left[\begin{array}{c} U_I^1 \\ P^1 \\ \vdots \\ U_I^N \\ P^N \\ U_\Gamma \end{array}\right] = \left[\begin{array}{c} F_I^1 \\ 0 \\ \vdots \\ F_I^N \\ 0 \\ F_\Gamma \end{array}\right]$$

ou

$$\left[\begin{array}{ccc|ccc} A_{II}^1 & & & A_{I\Gamma}^{1T}C_\Gamma^{1T} & B_I^{1T} & & \\ & \ddots & & \vdots & & \ddots & \\ & & A_{II}^N & A_{I\Gamma}^{NT}C_\Gamma^{NT} & & & B_I^{NT} \\ C_\Gamma^1 A_{I\Gamma}^1 & \dots & C_\Gamma^N A_{I\Gamma}^N & E_{\Gamma\Gamma} & C_\Gamma^1 B_\Gamma^{1T} & \dots & C_\Gamma^N B_\Gamma^{NT} \\ \hline B_I^1 & & & B_\Gamma^1 C_\Gamma^{1T} & 0 & & \\ & \ddots & & & & \ddots & \\ & & B_I^N & B_\Gamma^N C_\Gamma^{NT} & & & 0 \end{array}\right] \left[\begin{array}{c} U_I^1 \\ \vdots \\ U_I^N \\ U_\Gamma \\ P^1 \\ \vdots \\ P^N \end{array}\right] = \left[\begin{array}{c} F_I^1 \\ \vdots \\ F_I^N \\ F_\Gamma \\ 0 \\ \vdots \\ 0 \end{array}\right]$$

où l'on a posé :

$$E_{\Gamma\Gamma} = \sum_{k=1}^{N} C_\Gamma^k A_{\Gamma\Gamma}^k C_\Gamma^{kT}$$

$$F_\Gamma = \sum_{k=1}^{N} C_\Gamma^k F_\Gamma^k$$

Par ailleurs, l'expression donnée dans le paragraphe 4.2.4 pour les coefficients des matrices de couplage C_Γ^k reste valable ici, car on impose exactement les mêmes conditions pour le raccord de chacune des composantes de la vitesse. Remarquons que la première formulation nous permet de percevoir notre problème comme la résolution couplée de N problèmes de STOKES, alors que le second système correspond à un problème global de type recherche de point–selle.

4.3 Opérateurs d'interpolation hiérarchique des champs

4.3.1 Motivations

Il est bien connu que les méthodes itératives d'algèbre linéaire convergent d'autant plus rapidement que la solution initiale est proche de la solution recherchée. Or, dans le cadre d'une méthode adaptative, nous disposons d'une telle approximation : il s'agit de la solution calculée sur le maillage précédent. Il serait donc souhaitable de pouvoir initialiser la résolution du système linéaire en partant de la solution calculée à l'étape précédente. Or, notre algorithme d'adaptation de maillage présente une caractéristique particulière par rapport aux algorithmes usuels utilisés dans ce domaine. En effet, classiquement, on cherche à construire un maillage adapté du domaine de calcul à partir de la

donnée d'un maillage initial. Ainsi, un opérateur d'interpolation classique comme celui de P. CLÉMENT [41] ou celui de L.-R. SCOTT et S. ZHANG [133] peut permettre de réaliser une interpolation satisfaisante de la solution initiale. Dans le cas de notre algorithme, l'objectif consiste plutôt à définir une hiérarchie de triangulations à partir d'une hiérarchie donnée. Nous avons fait ce choix de façon à pouvoir prendre en compte tant le raffinement que le déraffinement du maillage (voir le paragraphe 3.4). La conséquence principale de ce fait est que les sous–domaines obtenus après application de l'algorithme peuvent être radicalement différents de ceux présents dans la hiérarchie initiale. Ainsi, dans notre contexte, les opérateurs d'interpolation usuels ne peuvent être utilisés pour définir une solution initiale sur le maillage adapté à partir la solution précédente. Nous proposons, dans les paragraphes qui suivent, une définition des opérateurs d'interpolation adaptée à notre situation.

4.3.2 Opérateur d'interpolation de SCHIEWECK

Dans [124], F. SCHIEWECK présente un opérateur d'interpolation général, qui permet, étant donné deux espaces d'éléments finis V_h^1 et V_h^2, de construire un champ interpolé défini sur V_h^2 à partir d'un champ donné défini sur V_h^1. Dans cet article, l'application principale de cet opérateur reste cependant le post–traitement des champs. En effet, F. SCHIEWECK souligne qu'il est plus simple de résoudre par exemple un problème de STOKES à l'aide d'un champ de vitesse non–conforme (par exemple, discrétisé à l'aide de l'élément de CROUZEIX–RAVIART), mais que la visualisation de la solution est facilitée par l'obtention d'une solution conforme (plus exactement continue sur son domaine de définition). Ainsi, F. SCHIEWECK motive la construction de son opérateur d'interpolation par la volonté de construire un champ interpolé conforme à partir d'un champ donné non–conforme. Notons cependant que différents cas d'applications sont énumérés à la fin de cet article, et notamment la généralisation de l'opérateur d'interpolation à un opérateur de prolongement pour une méthode multi–niveaux reposant sur une discrétisation par éléments finis non–conforme. Il s'agit précisément du cadre dans lequel nous allons utiliser cet opérateur.

Mais avant tout, commençons par présenter les idées fondamentales de la définition de l'opérateur d'interpolation de F. SCHIEWECK. Considérons donc un domaine Ω de $\mathbb{R}^n$, muni d'une triangulation conforme $\mathscr{T}_h$. Considérons deux triplets $\{K, \mathbb{P}_1, \Sigma_1\}$ et $\{K, \mathbb{P}_2, \Sigma_2\}$ où :

- K désigne un simplexe de $\mathbb{R}^n$;
- $\mathbb{P}_1$ et $\mathbb{P}_2$ désignent deux ensembles de polynômes définis sur K et à valeurs dans $\mathbb{R}$;
- Σ_1 et Σ_2 désignent deux ensembles de formes linéaires définies respectivement sur $\mathbb{P}_1$ et $\mathbb{P}_2$.

Autrement dit, nous considérons deux éléments finis définis sur l'élément de référence de la triangulation $\mathscr{T}_h$. Introduisons l'espace global suivant :

$$W_h = \left\{ u_h \in L^2(\Omega),\ \forall K \in \mathscr{T}_h,\ u_h|_K \in \mathscr{C}^\infty(K) \right\}$$

Nous supposerons que les espaces de discrétisation associés aux éléments finis définis ci–dessus, que nous noterons X_h^1 et X_h^2, vérifient :

$$X_H^1 \subset W_h \quad X_H^2 \subset W_h \qquad X_h^1 + X_h^2 \subset W_h$$

Pour définir l'opérateur d'interpolation, nous devons introduire les fonctions de base de l'espace X_h^2. Notons donc $\left(\phi_i^2\right)_{1\leq i\leq N_2}$ les fonctions de base de l'espace X_h^2. Pour chaque élément K de $\mathscr{T}_h$, nous définissons :

$$\mathscr{N}^2(K)=\left\{i\in[\![1,N_2]\!],\ \text{supp}\left(\phi_i^2\right)\cap\mathring{K}\neq\emptyset\right\}$$

Nous pouvons alors, pour chaque élément K, définir un ensemble de fonctions de base locales $\phi_i^2|_K$ et un ensemble des formes linéaires $\sigma_{i,K}\in(\mathscr{C}^\infty(K))'$ qui vérifient la relation de dualité suivante :

$$\sigma_{i,K}\left(\phi_i^2|_K\right)=\delta_{ij},\quad\forall(i,j)\in\mathscr{N}^2(K)^2$$

où δ_{ij} désigne le symbole de KRONECKER. Pour tout i de $\mathscr{N}^2(K)$, nous posons :

$$\mathscr{T}_h^i=\left\{K\in\mathscr{T}_h,\ i\in\mathscr{N}^2(K)\right\}$$

Définissons alors N_2 formes linéaires sur l'espace W_h en posant, pour tout i dans $\mathscr{N}^2(K)$:

$$\begin{aligned}\sigma_i : W_h &\longrightarrow \mathbb{R}\\ u &\longmapsto \sigma_i(u)=\frac{1}{\omega_i}\sum_{K\in\mathscr{T}_h^i}\omega_{i,K}\sigma_{i,K}\end{aligned}$$

avec

$$\omega_i=\sum_{K\in\mathscr{T}_h^i}\omega_{i,K}$$

et où les poids $\omega_{i,K}$ sont supposés positifs, mais choisis de façon arbitraire. À l'aide de ces formes linéaires, nous pouvons définir un opérateur de projection sur X_h^2 défini sur W_h par :

$$\begin{aligned}\pi_h : W_h &\longrightarrow X_h^2\\ u &\longmapsto \pi_h(u)=\sum_{i=1}^{N_2}\sigma_i(u)\phi_i^2\end{aligned}$$

Comme X_h^1 est inclus dans W_h, nous pouvons alors définir notre opérateur d'interpolation $\mathscr{R}_h^{1,2}$ comme étant la restriction du projecteur π_h à l'espace X_h^1 :

$$\begin{aligned}\mathscr{R}_h^{1,2} : X_h^1 &\longrightarrow X_h^2\\ u_h^1 &\longmapsto \mathscr{R}_h^{1,2}\left(u_h^1\right)=\pi_h\left(u_h^1\right)=\sum_{i=1}^{N_2}\sigma_i\left(u_h^1\right)\phi_i^2\end{aligned}$$

Définir un opérateur de prolongement pour une méthode multi–niveaux à l'aide de ces définitions est alors relativement simple. Considérons maintenant $\mathscr{T}_h^k$ et $\mathscr{T}_h^{k+1}$ deux triangulations de Ω telles que $\mathscr{T}_h^{k+1}$ soit un raffinement de $\mathscr{T}_h^k$, dans le sens défini au paragraphe 3.2.1. Ainsi, on est assuré que :

$$\forall K\in\mathscr{T}_h^{k+1},\ \exists!\ K'\in\mathscr{T}_h^k,\ K\subset K'$$

Notons V_h^k et V_h^{k+1} les espaces discrets définis respectivement à l'aide des triangulations $\mathscr{T}_h^k$ et $\mathscr{T}_h^{k+1}$. La propriété d'inclusion rappelée ci–dessus a alors pour conséquence :

$$V_h^k\subset W_h=\left\{u\in L^2(\Omega),\ \forall K\in\mathscr{T}_h^{k+1},\ u|_K\in\mathscr{C}^\infty(K)\right\}$$

En définissant l'opérateur de projection π_h comme précédemment :

$$\begin{aligned}\pi_h : W_h &\longrightarrow V_h^{k+1} \\ u &\longmapsto \pi_h(u) = \sum_{i=1}^{N_{k+1}} \sigma_i(u)\phi_i^{k+1}\end{aligned}$$

nous pouvons définir notre opérateur de prolongement en prenant sa restriction à l'espace V_k :

$$\begin{aligned}\mathscr{R}_h^{k,k+1} : V_h^k &\longrightarrow V_h^{k+1} \\ u_h^k &\longmapsto \mathscr{R}_h^{k,k+1}\left(u_h^k\right) = \pi_h\left(u_h^k\right) = \sum_{i=1}^{N_{k+1}} \sigma_i\left(u_h^k\right)\phi_i^{k+1}\end{aligned}$$

Nous renvoyons à [123–126] pour l'étude approfondie des propriétés de cet opérateur, notamment pour les résultats d'approximation et de stabilité.

4.3.3 Application à notre méthode adaptative

Voyons maintenant comment utiliser cet opérateur d'interpolation dans le cadre de notre méthode. Considérons par exemple un problème de LAPLACE muni de conditions aux limites de DIRICHLET homogènes sur un domaine borné Ω de $\mathbb{R}^n$. Supposons que nous disposons d'une hiérarchie de triangulations, que nous noterons $(\mathscr{T}_i)_{1\leq i\leq N}$, de notre domaine de calcul. Pour tout $i \in [\![0, N]\!]$, nous notons Ω_i la réunion des éléments non–raffinés de $\mathscr{T}_i$. Les sous–domaines Ω_i forment par définition une partition de Ω, et présentent chacun un niveau de raffinement différent. Nous pouvons donc résoudre numériquement notre problème sur cette géométrie à l'aide de la méthode numérique présentée plus haut. Cette résolution nous fournit une solution appartenant à l'espace suivant :

$$W_h = \prod_{i=1}^{N} W_h(\Omega_i) = \left\{ u_h = \left(u_h^i\right)_{1\leq i\leq n},\ u_h^i \in W_h(\Omega_i) \right\}$$

où $W_h(\Omega_i)$ correspond à l'espace discret associé à l'élément de CROUZEIX–RAVIART sur le sous–domaine Ω_i :

$$\begin{aligned}W_h(\Omega_i) = \big\{ & u_h^i \in L^2(\Omega_i)\,;\ \forall K \in \mathscr{T}_h(\Omega_i)\,,\ u_h^i|_K \in \mathbb{P}^1(K)\,; \\ & \forall x \in \mathscr{N}^I(\mathscr{T}_h(\Omega_i))\,,\ u_h^i \text{ continue en } x\,;\ \forall x \in \mathscr{N}^\partial(\mathscr{T}_h(\Omega_i))\,,\ u_h^i(x) = 0 \big\}\end{aligned}$$

Après avoir estimé l'erreur numérique (voir le paragraphe 4.4), nous construisons un indicateur de raffinement qui associe à chaque élément de chacun des sous–domaines Ω_i un entier valant -1, 0, ou 1, selon que l'on désire respectivement le déraffiner, le conserver ou bien le raffiner. Nous disposons maintenant des données nécessaires pour appliquer notre algorithme d'adaptation de maillage.

Afin de pouvoir définir notre solution sur le maillage généré par notre algorithme, nous allons procéder comme suit. Dans un premier temps, nous allons « lisser » notre solution sur les éléments raffinés de la triangulation actuelle. De cette façon, si un ensemble d'éléments se trouve être supprimé de la hiérarchie par déraffinement, la solution initiale sur l'élément père les contenant sera d'ores–et–déjà connue. Il nous reste à

définir les solutions initiales sur les nouveaux éléments générés par raffinement. Pour cela, nous utilisons l'opérateur d'interpolation défini ci–dessus, en utilisant comme espaces de départ et d'arrivée les espaces de discrétisation par l'élément fini de CROUZEIX–RAVIART. De cette façon, nous procédons à l'interpolation des champs à l'instant même où nous définissons la nouvelle hiérarchie de triangulations, et nous disposons, une fois l'adaptation terminée, d'une solution initiale définie sur la nouvelle hiérarchie de triangulations. Nous voyons clairement que nos champs sont alors perçus comme étant définis sur l'intégralité de la hiérarchie de triangulations, et c'est pour cela que nous parlons d'*interpolation hiérarchique des champs*.

4.4 Estimation de l'erreur numérique

Le dernier outil numérique dont nous aurons besoin pour mettre en œuvre notre méthode est une technique permettant de localiser et de quantifier l'erreur numérique. Le paragraphe qui suit a pour objectif de fournir les définitions nécessaires à l'établissement d'une estimation de cette erreur qui soit utilisable dans le contexte de notre méthode adaptative. Nous présentons ensuite divers résultats établis pour le cas des problèmes de LAPLACE et de STOKES.

4.4.1 Estimation d'erreur *a priori* et *a posteriori*

Comme nous l'avons souligné plus haut, l'un des objectifs de notre méthode consiste à déterminer une hiérarchie de triangulations qui permette de minimiser l'erreur numérique, à savoir l'écart entre la solution exacte et la solution numérique de notre problème, tout en assurant l'efficacité du calcul. Pour cela, nous avons besoin de déterminer cette erreur, et plus particulièrement de la localiser, afin de pouvoir par la suite adapter le maillage en conséquence. Deux approches sont classiquement employées pour estimer l'erreur, à savoir l'analyse d'erreur *a priori* et l'analyse d'erreur *a posteriori*. La première approche cherche à établir une majoration de l'erreur numérique à partir de la régularité de la solution exacte du problème. À l'inverse, la seconde approche cherche à majorer l'erreur numérique à partir de la régularité de la solution numérique approchée. Les deux concepts fondamentaux sur lesquels reposent ces deux techniques sont la *précision* et la *stabilité*. Dans le cas de l'analyse *a priori*, la notion de précision est reliée aux propriétés d'approximation de la méthode numérique utilisée, et la notion de stabilité se rapporte au problème approché. L'idée sous–jacente peut s'exprimer de la façon suivante :

$$\text{faible erreur d'interpolation} + \text{stabilité du problème discret} \Longrightarrow \text{faible erreur numérique}$$

Dans le cas de l'analyse *a posteriori*, la notion de stabilité s'applique cette fois au problème continu, et la notion de précision se rapporte au résidu, c'est–à–dire à la précision avec laquelle la solution discrète vérifie les équations du problème continu. Ainsi, la démarche de l'analyse *a posteriori* peut se résumer par :

$$\text{faible résidu} + \text{stabilité du problème continu} \Longrightarrow \text{faible erreur numérique}$$

Nous avons souligné au paragraphe 1.1.3 que l'analyse *a priori* ne permet pas de réaliser une estimation localisée de l'erreur. Ceci est essentiellement dû au fait que la précision est évaluée à l'aide des propriétés d'approximation de la méthode numérique utilisée. Nous nous tournons donc vers le formalisme des estimations d'erreurs *a posteriori*,

développé initialement par I. BABUŠKA et W. RHEINBOLT en 1978 [5, 6]. Nous pouvons regrouper ce type d'estimateurs en plusieurs familles :

- les estimations d'erreur *a posteriori* fondés sur le résidu ;
- les estimations d'erreur *a posteriori* hiérarchiques ;
- les estimations d'erreur *a posteriori* reposant sur une méthode de dualité.

Les premiers ont été introduits par I. BABUŠKA et W. RHEINBOLT en 1978 [5, 6], puis étendus par R. VERFÜRTH [144, 145]. Les seconds ont été introduits par R. BANK et A. WEISER en 1985 [10], puis étendus par R. BANK et R. SMITH [9]. Enfin, les derniers ont été introduits par C. JOHNSON en 1987 [78], puis développés par R. BECKER et R. RANNACHER [15].

Nous introduisons ici les définitions utilisées dans les paragraphes qui suivent. Pour cela nous considérons le problème modèle suivant :

Problème 4.20 : *Soient W et V deux espaces de* HILBERT, *$a(\cdot,\cdot)$ une forme bilinéaire sur $W \times V$ et $l(\cdot)$ une forme linéaire sur V. On cherche une fonction $u \in W$ telle que $\forall v \in V$, l'on ait :*

$$a(u,v) = l(v)$$

Nous supposerons que ce problème est bien posé au sens de J. HADAMARD. Introduisons maintenant une version discrète de ce problème :

Problème 4.21 : *Trouver une fonction $u_h \in W_h$ telle que $\forall v_h \in V_h$, l'on ait :*

$$a_h(u_h, v_h) = l_h(v_h)$$

où W_h, V_h, $a_h(\cdot,\cdot)$ et $l_h(\cdot)$ désignent respectivement un espace de dimension finie approchant W, un espace de dimension finie approchant V, une approximation discrète de $a(\cdot,\cdot)$, et une approximation discrète de $l(\cdot)$. Nous supposerons que ces approximations sont construites sur la base d'une triangulation du domaine de calcul notée $\mathcal{T}_h$. Là encore, nous ferons l'hypothèse que ce problème discret est bien posé au sens de J. HADAMARD.

Définition 4.1 : *Nous dirons qu'une fonction $e(h, u_h, l)$ est un estimateur d'erreur a posteriori si :*

$$\|u - u_h\|_{W+W_h} \leq e(h, u_h, l)$$

Cette définition est quelque peu trop générale, car nous avons besoin de pouvoir localiser l'erreur numérique. Pour cela, nous posons :

Définition 4.2 : *Nous dirons qu'une fonction $e(h, u_h, l)$ est un estimateur d'erreur a posteriori localisé si :*

$$\|u - u_h\|_{W+W_h} \leq e(h, u_h, l)$$

et si

$$e(h, u_h, l) = \left(\sum_{K \in \mathcal{T}_h} e_K(u_h, l)^2 \right)^{\frac{1}{2}}$$

Dans ce cas, les quantités $e_K(u_h, l)$ sont qualifiées d'estimateurs d'erreur a posteriori locaux.

4.4.2 Estimation d'erreur *a posteriori* pour le problème de LAPLACE

Considérons comme problème modèle un problème de LAPLACE, posé sur un domaine Ω borné dans $\mathbb{R}^n$, et muni de conditions aux limites de DIRICHLET homogènes :

Problème 4.22 : *Soit f une fonction de $L^2(\Omega)$. Nous cherchons une fonction $u \in H_0^1(\Omega)$ telle que $\forall v \in H_0^1(\Omega)$, l'on ait :*

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v = l(v)$$

L'objectif des paragraphes qui suivent est de définir un estimateur d'erreur *a posteriori* fondé sur le résidu de la solution obtenue lors de la résolution de ce problème à l'aide de la méthode présentée au paragraphe 4.2. Pour cela, nous partons de la définition d'un estimateur d'erreur classique pour ce problème lorsque l'on utilise une approximation conforme par éléments finis $\mathbb{P}^k$ de LAGRANGE. Nous présentons ensuite deux extensions de cet estimateur pour le cas non–conforme, la première se rapportant à l'élément fini de CROUZEIX–RAVIART, la seconde correspondant à une résolution de ce problème par une méthode d'éléments joints dont les sous–problèmes sont discrétisés à l'aide d'éléments finis $\mathbb{P}^k$ de LAGRANGE. La formulation d'un estimateur d'erreur adapté à notre problème découle des fortes similitudes que l'on observe entre ces deux estimateurs.

Considérons donc $\mathscr{T}_h$, une triangulation du domaine de calcul , ainsi que V_h, l'espace discret associé à l'élément fini $\mathbb{P}^k$ de LAGRANGE. Comme il s'agit d'un cadre d'approximation conforme, à savoir $V_h \subset H_0^1(\Omega)$, on dispose d'une propriété essentielle, appelée orthogonalité de GALERKINE :

Propriété 4.2 :

$$\forall v_h \in V_h,\ a(u-u_h, v_h) = 0$$

Cette propriété permet de construire un estimateur d'erreur *a posteriori* localisé bien connu pour le problème de LAPLACE :

Propriété 4.3 : *En supposant que la dégénérescence de la triangulation $\mathscr{T}_h$ est bornée, il existe une constante c indépendante des h_K telle que :*

$$\|u-u_h\|_{H^1(\Omega)} \leq c \left(\sum_{K \in \mathscr{T}_h} \eta_K^2 + h_K^2 \left\| f - f_h \right\|_{L^2(K)}^2 \right)^{\frac{1}{2}}$$

avec

$$\forall K \in \mathscr{T}_h,\ \eta_K = h_K \left\| f_h + \Delta u_h \right\|_{L^2(K)} + \frac{1}{2} \sum_{F \in \mathscr{F}_K} h_F^{\frac{1}{2}} \left\| [\![\partial_n u_h]\!]_F \right\|_{L^2(F)}$$

où h_K désigne la longueur de la plus longue arête de K, f_h une approximation discrète de f, $\mathscr{F}_K$ l'ensemble des faces de K, h_F la longueur de la plus grande arête de la face F, et $[\![\partial_n u_h]\!]_F$ le saut de la dérivée normale de u_h à travers la face F. Nous renvoyons à [54] pour la démonstration de la construction de cet estimateur, ainsi que le détail de ses propriétés. Remarquons que dans le cas de l'élément fini $\mathbb{P}^1$ de LAGRANGE, on a :

$$\forall K \in \mathscr{T}_h,\ \Delta u_h|_K = 0 \qquad \text{et} \qquad \forall F \in \mathscr{T}_h,\ [\![\partial_n u_h]\!]_F = \text{cste}$$

Ceci nous permet de simplifier l'expression de cet estimateur sous la forme suivante :

$$\forall K \in \mathscr{T}_h,\ \eta_K = h_K \left\| f_h \right\|_{L^2(K)} + \frac{1}{2} \sum_{F \in \mathscr{F}_K} h_F^{\frac{1}{2}} |F|^{\frac{1}{2}} [\![\partial_n u_h]\!]_F$$

Plusieurs estimateurs d'erreur *a posteriori* adaptés à la discrétisation par l'élément fini de CROUZEIX–RAVIART de divers problèmes ont été proposés dans [3, 51, 127]. Comme précédemment, nous nous limiterons ici au cas d'un problème de LAPLACE muni de conditions aux limites de DIRICHLET homogènes. Considérons donc une triangulation du domaine de calcul, notée $\mathscr{T}_h$, et introduisons l'espace discret associé à l'élément fini de CROUZEIX–RAVIART :

$$W_h = \{ u_h \in L^2(\Omega) \,;\, \forall K \in \mathscr{T}_h \,,\, u_h|_K \in \mathbb{P}^1(K) \,; \\ \forall x \in \mathscr{N}^I(\mathscr{T}_h) \,,\, u_h \text{ continue en } x \,;\, \forall x \in \mathscr{N}^\partial(\mathscr{T}_h) \,,\, u_h(x) = 0 \}$$

où $\mathscr{N}^I(\mathscr{T}_h)$ et $\mathscr{N}^\partial(\mathscr{T}_h)$ désignent respectivement les milieux des faces intérieures de $\mathscr{T}_h$ et les milieux des faces de $\mathscr{T}_h$ situées sur le bord du domaine. Notons également X_h l'espace discret associé à l'élément fini $\mathbb{P}^1$ de LAGRANGE :

$$X_h = \left\{ u_h \in \mathscr{C}^0(\Omega) \,;\, \forall K \in \mathscr{T}_h, u_h|_K \in \mathbb{P}^1(K) \right\}$$

Notre problème discret est défini à partir de la forme bilinéaire suivante :

$$a_h : W_h \times W_h \longrightarrow \mathbb{R} \\ (u_h, v_h) \longmapsto a_h(u_h, v_h) = \sum_{K \in \mathscr{T}_h} \int_K \nabla u_h \cdot \nabla v_h$$

Nous chercherons à établir une estimation d'erreur *a posteriori* en utilisant la norme brisée suivante :

$$\|u\| = a_h(u,u)$$

Cette norme contrôle uniformément la norme L^2 sur $H^1_0(\Omega) + W_h$ et on dispose de la propriété suivante (voir par exemple [54]) :

Propriété 4.4 : *En supposant que la dégénérescence de la triangulation $\mathscr{T}_h$ est bornée, il existe une constante c, indépendante des h_K, telle que :*

$$\|u - u_h\| \leq c \left(\sum_{K \in \mathscr{T}_h} \eta_K^2 + h_K^2 \left\| f - f_h \right\|^2_{L^2(K)} \right)^{\frac{1}{2}} + \inf_{v_h \in X_h} \|u_h - v_h\|$$

On remarque que cette formulation fait intervenir l'estimation d'erreur η_K associé à l'élément fini conforme $\mathbb{P}^1$ de LAGRANGE, auquel est ajouté un terme résultant de l'estimation de la non–conformité de la solution. Introduisons l'opérateur d'interpolation de OSWALD [108], $\mathscr{O}_h : W_h \longrightarrow X_h$, qui vérifie :

$$\forall x \in \mathscr{N}(\mathscr{T}_h),\ \forall w_h \in W_h,\ \mathscr{O}_h(w_h(x)) = \frac{1}{n_x} \sum_{K \in \Omega_x} w_h|_K(x)$$

où $\mathscr{N}(\mathscr{T}_h)$ désigne l'ensemble des sommets intérieurs de $\mathscr{T}_h$, Ω_x l'ensemble des éléments de $\mathscr{T}_h$ admettant x comme sommet, et n_x le cardinal de cet ensemble. Pour les sommets situés sur le bord, nous posons :

$$\mathscr{O}_h(w_h(x)) = 0$$

En notant $[\![w_h]\!]_F$ le saut de $w_h \in W_h$ à travers la face F de la triangulation, on dispose de la propriété suivante :

Propriété 4.5 : *Si la dégénérescence de la triangulation $\mathscr{T}_h$ est bornée, alors il existe une constante c, indépendante des h_K, telle que :*

$$\forall w_h \in W_h,\ \forall K \in \mathscr{T}_h,\ |w_h - \mathscr{O}_h(w_h)|_{H^1(K)} \leq c \sum_{F \in \mathscr{F}_K} h_F^{-\frac{1}{2}} \left\| [\![w_h]\!]_F \right\|_{L^2(F)}$$

où $\mathscr{F}_K$ désigne l'ensemble des faces de l'élément K. On en déduit alors l'estimation d'erreur suivante :

Corollaire 4.1 : *Si la dégénérescence de la triangulation $\mathscr{T}_h$ est bornée, alors il existe une constante c, indépendante des h_K, telle que :*

$$\|u - u_h\| \leq c \left(\sum_{K \in \mathscr{T}_h} \eta_K^2 + \sum_{F \in \mathscr{F}_h} \eta_F^2 + \sum_{K \in \mathscr{T}_h} h_K^2 \left\| f - f_h \right\|_{L^2(K)}^2 \right)^{\frac{1}{2}}$$

où $\mathscr{F}_h$ désigne l'ensemble des faces des éléments de la triangulation $\mathscr{T}_h$, et où on a posé :

$$\eta_K = h_K \left\| f_h + \Delta u_h \right\|_{L^2(K)} + \frac{1}{2} \sum_{F \in \mathscr{F}_K} h_F^{\frac{1}{2}} \left\| [\![\partial_n u_h]\!]_F \right\|_{L^2(F)}$$

$$\eta_F = h_F^{-\frac{1}{2}} \left\| [\![u_h]\!]_F \right\|_{L^2(F)}$$

Remarquons que la simplification de η_K énoncée au paragraphe ci–dessus reste valide.

Nous présentons maintenant un estimateur d'erreur adapté à la résolution de notre problème modèle par une méthode d'éléments joints reposant sur une discrétisation par éléments finis $\mathbb{P}^k$ de LAGRANGE. Cet estimateur a été établi par C. BERNARDI, F. HECHT et Y. MADAY dans les articles [21, 22] dans le cas d'éléments finis $\mathbb{P}^k$ de LAGRANGE, avec $k \geq 2$, et il a été étendu à la discrétisation par éléments finis $\mathbb{P}^1$ de LAGRANGE par Z. BELHACHMI dans [16]. Il s'écrit sous la forme suivante :

$$\|u - u_h\| \leq c \left(\sum_{K \in \mathscr{T}_h} \eta_K^2 + \sum_{F \in \Gamma_h} \eta_F^2 + \sum_{K \in \mathscr{T}_h} h_K^2 \left\| f - f_h \right\|_{L^2(K)}^2 \right)^{\frac{1}{2}}$$

où Γ_h désigne les faces composant les interfaces entre sous–domaines, où η_K désigne l'estimateur d'erreur étudié plus haut dans le cas des éléments finis conformes $\mathbb{P}^k$ de LAGRANGE, et où η_F s'écrit :

$$\eta_F = h_F^{-\frac{1}{2}} \left\| [\![u_h]\!]_F \right\|_{L^2(F)}$$

Ainsi, nous constatons que la forme de cet estimateur d'erreur est pratiquement identique à celle de l'estimateur utilisé dans le cas de l'élément fini de CROUZEIX–RAVIART. Il s'agit en effet de décomposer l'erreur numérique en deux contributions, l'une associée à l'approximation conforme et l'autre aux non–conformités du champ, et les expressions retenues pour ces deux contributions sont strictement identiques. La seule différence se situe au niveau du nombre de faces présentant une non–conformité : alors que toutes les faces de la triangulation sont concernées dans le cas de l'élément fini de CROUZEIX–RAVIART, on ne considère que les faces constituant les interfaces entre sous–domaines dans le cadre de la méthode de C. BERNARDI, F. HECHT et Y. MADAY. Nous pouvons donc généraliser cet estimateur d'erreur à notre méthode numérique, en prenant en compte de la même façon les non–conformités liées à l'élément fini de CROUZEIX–RAVIART et celles liées aux éléments joints.

4.4.3 Estimation d'erreur *a posteriori* pour le problème de STOKES

Nous nous proposons maintenant d'étudier la construction d'un estimateur d'erreur pour la résolution d'un problème de STOKES utilisant la méthode présentée au paragraphe 4.2. Comme pour le problème de LAPLACE, nous partons d'un estimateur d'erreur établi dans le cadre d'une approximation conforme, pour ensuite en étudier deux extensions au cas non–conforme, l'une adaptée à la discrétisation par la méthode VEF, l'autre adaptée à la résolution par une méthode d'éléments joints. La formulation de l'estimateur envisagé découle alors des similitudes que l'on observe entre ces deux estimateurs.

Rappelons tout d'abord l'estimateur d'erreur présenté par A. ERN et J.-L. GUERMOND dans [54] dans le cadre d'une approximation conforme. On considère le problème de STOKES suivant :

Problème 4.23 : *Soit $\boldsymbol{f}$ une fonction de $L^2(\Omega)^n$. Nous cherchons un couple de fonctions $(\boldsymbol{u},p)\in H^1_0(\Omega)^n\times L^2_0(\Omega)$ tel que $\forall(\boldsymbol{v},q)\in H^1_0(\Omega)^n\times L^2_0(\Omega)$, l'on ait :*

$$\begin{aligned} a(\boldsymbol{u},\boldsymbol{v})-b(\boldsymbol{v},p)&=l(\boldsymbol{v})\\ b(\boldsymbol{u},q)&=0 \end{aligned}$$

où $L^2_0(\Omega)$ désigne l'ensemble des fonctions de $L^2(\Omega)$, à valeur moyenne nulle sur Ω, et où l'on a posé :

$$a(\boldsymbol{u},\boldsymbol{v})=\int_\Omega \nabla\boldsymbol{u}:\nabla\boldsymbol{v} \qquad b(\boldsymbol{v},q)=\int_\Omega q\nabla\cdot\boldsymbol{u} \qquad l(\boldsymbol{v})=\int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}$$

Pour une face F de la triangulation correspondant à l'intersection de deux éléments K_i et K_j, nous notons :

$$[\![\partial_{\boldsymbol{n}}\boldsymbol{u}_h-p_h\boldsymbol{n}]\!]_F=\left(\nabla\boldsymbol{u}_h|_{K_i}\cdot\boldsymbol{n}_i-p_h|_{K_i}\boldsymbol{n}_i\right)+\left(\nabla\boldsymbol{u}_h|_{K_j}\cdot\boldsymbol{n}_j-p_h|_{K_j}\boldsymbol{n}_j\right)$$

où la normale $\boldsymbol{n}_i$ (resp. $\boldsymbol{n}_j$) désigne la normale à F sortant de K_i (resp. K_j). Dans le cas où l'on utilise une approximation conforme, on dispose de l'estimateur d'erreur suivant :

$$\|\boldsymbol{u}-\boldsymbol{u}_h\|^2_{H^1_0(\Omega)^n}+\left\|p-p_h\right\|^2_{L^2(\Omega)}\leq c\sum_{K\in\mathscr{T}_h}\left(\eta_K^2+h_K^2\left\|\boldsymbol{f}-\boldsymbol{f}_h\right\|^2_{L^2(K)}\right)$$

où $\boldsymbol{f}_h$ désigne une approximation discrète de $\boldsymbol{f}$, et où l'on a posé :

$$\eta_K=h_K\left\|\boldsymbol{f}_h+\Delta\boldsymbol{u}_h-\nabla p_h\right\|_{L^2(K)^n}+\|\nabla\cdot\boldsymbol{u}_h\|_{L^2(K)}+\frac{1}{2}\sum_{F\in\mathscr{F}_K}h_F^{\frac{1}{2}}\left\|[\![\partial_{\boldsymbol{n}}\boldsymbol{u}_h-p_h\boldsymbol{n}]\!]_F\right\|_{L^2(F)^n}$$

Nous remarquons qu'il s'agit d'une extension naturelle de l'estimateur établi pour le problème de LAPLACE dans le cas d'une approximation conforme. Nous renvoyons à [54] pour le détails des propriétés de cet estimateur.

Une première généralisation de cet estimateur pour un cadre d'approximation non–conforme a été proposée par V. JOHN en 1998 dans l'article [77]. Cette extension se rapporte à la discrétisation d'un problème de STOKES par le couple d'éléments finis $\mathbb{P}^1_{NC}$–$\mathbb{P}^0$, ce qui correspond exactement à la méthode VEF standard de Trio-U. L'estimateur correspondant s'écrit sous la forme suivante :

Propriété 4.6 : *Si la dégénérescence de la triangulation $\mathscr{T}_h$ est bornée, alors il existe une constante c telle que :*

$$\|\boldsymbol{u}-\boldsymbol{u}_h\|^2+\left\|p-p_h\right\|^2_{L^2(\Omega)}\leq c\left(\sum_{K\in\mathscr{T}_h}\eta_K^2+\sum_{F\in\mathscr{F}_h}\eta_F^2+\sum_{K\in\mathscr{T}_h}h_K^2\left\|\boldsymbol{f}-\boldsymbol{f}_h\right\|^2_{L^2(K)^n}\right)$$

où l'on a posé :

$$\eta_K=h_K\left\|\boldsymbol{f}_h+\Delta\boldsymbol{u}_h-\nabla p_h\right\|_{L^2(K)^n}+\|\nabla\cdot\boldsymbol{u}_h\|_{L^2(K)}+\frac{1}{2}\sum_{F\in\mathscr{F}_K}h_F^{\frac{1}{2}}\left\|\left[\!\left[\partial_{\boldsymbol{n}}\boldsymbol{u}_h-p_h\boldsymbol{n}\right]\!\right]_F\right\|_{L^2(F)^n}$$

$$\eta_F=h_F^{-\frac{1}{2}}\left\|\left[\!\left[\boldsymbol{u}_h\right]\!\right]_F\right\|_{L^2(F)^n}$$

et où $\|\cdot\|$ désigne la norme brisée suivante :

$$\|\boldsymbol{u}\|^2=a_h(\boldsymbol{u},\boldsymbol{u})=\sum_{K\in\mathscr{T}_h}\int_K\nabla\boldsymbol{u}:\nabla\boldsymbol{u}$$

Par ailleurs, dans le cadre de la méthode VEF standard, on a :

$$\forall K\in\mathscr{T}_h,\ \ \Delta\boldsymbol{u}_h|_K=0\qquad\text{et}\qquad\nabla p_h|_K=0$$

Ceci nous permet de simplifier l'expression de cet estimateur d'erreur de la façon suivante :

$$\eta_K=h_K\left\|\boldsymbol{f}_h\right\|_{L^2(K)^n}+\|\nabla\cdot\boldsymbol{u}_h\|_{L^2(K)}+\frac{1}{2}\sum_{F\in\mathscr{F}_K}h_F^{\frac{1}{2}}\left\|\left[\!\left[\partial_{\boldsymbol{n}}\boldsymbol{u}_h-p_h\boldsymbol{n}\right]\!\right]_F\right\|_{L^2(F)^n}$$

Par contre, dans le cadre de la méthode VEF étendue, on a seulement :

$$\forall K\in\mathscr{T}_h,\ \ \Delta\boldsymbol{u}_h|_K=0\qquad\text{et}\qquad\nabla p_h|_K=\text{cste}$$

ce qui donne :

$$\eta_K=h_K\left\|\boldsymbol{f}_h-\nabla p_h\right\|_{L^2(K)^n}+\|\nabla\cdot\boldsymbol{u}_h\|_{L^2(K)}+\frac{1}{2}\sum_{F\in\mathscr{F}_K}h_F^{\frac{1}{2}}\left\|\left[\!\left[\partial_{\boldsymbol{n}}\boldsymbol{u}_h-p_h\boldsymbol{n}\right]\!\right]_F\right\|_{L^2(F)^n}$$

Une autre généralisation a été proposée par Z. BELHACHMI dans l'article [17] pour la résolution d'un problème de STOKES à l'aide d'une méthode d'éléments joints, dont les sous–problèmes sont discrétisés dans un cadre d'approximation conforme (élément fini $\mathbb{P}^1+$ *bulle* pour la vitesse, élément fini $\mathbb{P}^1$ pour la pression). Dans ce cadre, l'estimateur d'erreur établi par Z. BELHACHMI s'écrit :

$$\|\boldsymbol{u}-\boldsymbol{u}_h\|^2+\left\|p-p_h\right\|^2_{L^2(\Omega)}\leq c\left(\sum_{K\in\mathscr{T}_h}\eta_K^2+\sum_{F\in\Gamma_h}\eta_F^2+\sum_{K\in\mathscr{T}_h}h_K^2\left\|\boldsymbol{f}-\boldsymbol{f}_h\right\|^2_{L^2(K)^n}\right)$$

où Γ_h désigne l'ensemble des faces composant les interfaces entre sous–domaines, et où l'on a posé :

$$\eta_K=h_K\left\|\boldsymbol{f}_h+\Delta\boldsymbol{u}_h-\nabla p_h\right\|_{L^2(K)^n}+\|\nabla\cdot\boldsymbol{u}_h\|_{L^2(K)}+\frac{1}{2}\sum_{F\in\mathscr{F}_K}h_F^{\frac{1}{2}}\left\|\left[\!\left[\partial_{\boldsymbol{n}}\boldsymbol{u}_h-p_h\boldsymbol{n}\right]\!\right]_F\right\|_{L^2(F)^n}$$

$$\eta_F=h_F^{-\frac{1}{2}}\left\|\left[\!\left[\boldsymbol{u}_h\right]\!\right]_F\right\|_{L^2(F)^n}$$

On remarque là encore que l'estimateur d'erreur proposé pour la méthode VEF et celui établi pour la méthode d'éléments joints présentent des expressions pratiquement similaires. L'erreur est décomposée en deux contributions, l'une relevant de l'approximation conforme, l'autre associée aux non–conformités, et nous remarquons que les expressions de ces contributions sont identiques dans les deux situations. Là aussi, la seule différence se situe au niveau du nombre de faces présentant une non–conformité : alors que toutes les faces sont concernées dans le cas de la méthode VEF, on ne considère que les faces constituant les interfaces entre sous–domaines dans le cas de la méthode d'éléments joints. Nous pouvons donc naturellement considérer la généralisation de cet estimateur d'erreur au cas de notre méthode numérique, en prenant en compte de la même façon les non–conformités liées à la méthode VEF et celles liées aux éléments joints.

Résumé

Nous avons présenté dans ce chapitre différents concepts mathématiques sur lesquels repose notre méthode adaptative. Nous avons dans un premier temps développé un formalisme d'éléments joints adapté à la méthode de Volumes-Éléments Finis de Trio-U. Cet outil permet de prendre en compte numériquement les non–conformités du maillage générées par l'algorithme d'adaptation choisi au chapitre précédent. Nous avons ensuite présenté un opérateur d'interpolation hiérarchique des champs adapté à notre méthode. Cet opérateur permet de lisser et d'interpoler les champs sur les différents niveaux de raffinement présents dans le maillage. Enfin, nous avons développé la notion d'estimation d'erreur a posteriori, en présentant plus en détail les estimateurs d'erreur les plus classiques pour les problèmes de Laplace *et de* Stokes. *Ces derniers outils sont essentiels à la mise en œuvre de notre méthode car ils permettent de localiser l'erreur numérique afin de guider l'adaptation du maillage.*

Nous disposons donc maintenant de tous les outils théoriques nécessaires à la mise en œuvre de notre méthode adaptative, à savoir :

- *une technique d'adaptation géométrique reposant sur une méthode de subdivision explicite ;*
- *une méthode numérique prenant en compte les non–conformités du maillage (méthode VEF avec formalisme d'éléments joints) ;*
- *des opérateurs d'interpolation hiérarchique des champs adaptés à notre géométrie ;*
- *un outil de localisation de l'erreur numérique.*

Nous allons donc nous tourner dans le chapitre qui suit vers les questions relatives à l'implémentation parallèle de cette méthode.

CHAPITRE 5

Parallélisme adaptatif et équilibrage dynamique de la charge

Ce chapitre est consacré à l'implémentation parallèle de notre méthode adaptative. Nous commençons par présenter dans le détail une méthode générale de conception d'algorithmes parallèles. Nous cherchons ensuite à appliquer cette méthode à la parallélisation des différentes phases de notre méthode adaptative. Nous verrons que la phase d'adaptation du maillage et la phase de résolution du problème ne peuvent être appréhendées en utilisant les mêmes techniques. Nous étudions en détail la problématique de l'équilibrage de la charge de la phase de résolution, grâce à des heuristiques de partitionnement de graphe. Ce sera l'occasion de présenter quelques–unes de ces heuristiques. Au sortir de ce travail, nous sommes en mesure d'énumérer les développements qu'il sera nécessaire d'apporter à la plate–forme Trio-U pour implémenter notre méthode adaptative.

5.1 Une démarche générale pour la conception d'algorithmes parallèles

Nous décrivons ici une démarche générale de conception d'algorithmes parallèles proposée initialement par I. FOSTER [61] et détaillée par M.-J. QUINN dans [110]. Cette démarche utilise un modèle de tâches communicantes : le programme est perçu comme un ensemble de tâches s'exécutant en parallèle et pouvant échanger des données par envoi et réception de messages. Dans ce modèle, l'envoi de données est considéré comme non-bloquant : une fois le message envoyé, le processeur peut continuer à exécuter le code associé à la tâche courante. À l'inverse, la réception de message est bloquante : une tâche reste en attente d'un message en provenance d'une autre tâche tant que cette dernière ne l'a pas envoyé. La démarche proposée par I. FOSTER pour concevoir un algorithme parallèle consiste à rechercher le maximum de scalabilité en traitant le plus tard possible les questions relatives à la machine utilisée. Elle se décompose en quatre étapes : le partitionnement, où le problème est découpé en tâches élémentaires, la structuration des communications, où l'on spécifie les échanges d'informations entre tâches élémentaires nécessaires à l'exécution de l'algorithme, l'agglomération, qui consiste à regrouper les tâches élémentaires en macro–tâches de façon à minimiser les communications, et enfin le placement, qui affecte chaque macro–tâche à une unité de calcul de façon à

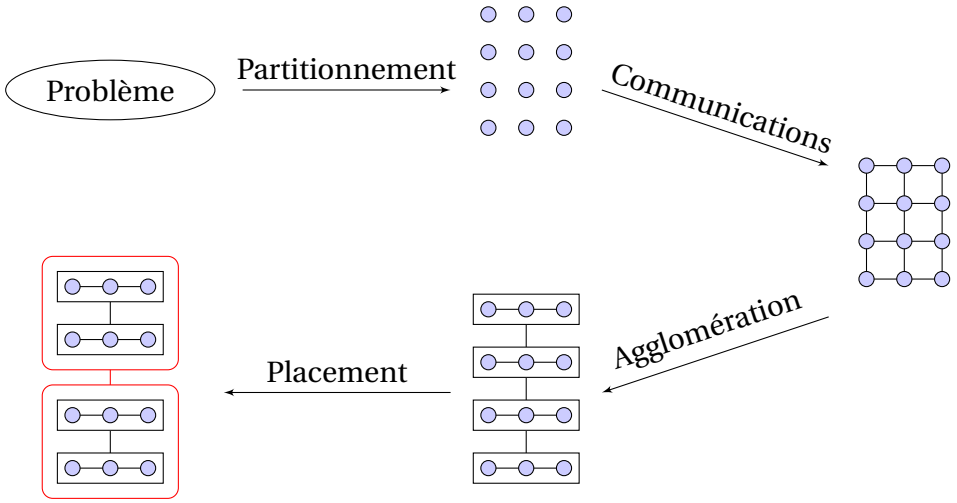


FIG. 5.1 : La méthode de conception d'algorithmes parallèles de I. FOSTER

équilibrer la charge. La figure 5.1 représente schématiquement le rôle de chacune de ces étapes dans la construction d'un algorithme parallèle. Nous allons maintenant détailler les principes de chacune de ces étapes.

5.1.1 Partitionnement du problème

L'objectif principal de la phase de partitionnement consiste à identifier le maximum de parallélisme possible en décomposant le problème en tâches élémentaires. Cette décomposition porte à la fois sur les données du problème et sur les instructions agissant sur ces données. Ainsi, cette phase peut être abordée de deux façons, selon que l'on cherche avant tout à partitionner les données du problèmes (on parle alors de parallélisme de données, ou de parallélisme structurel), ou bien les intructions du programme (on parle alors de parallélisme de tâches, ou de parallélisme fonctionnel). Quelle que soit la méthode considérée, l'objectif est de maximiser le nombre de tâches élémentaires, car il s'agit d'une borne supérieure sur les capacités d'exécution en parallèle de l'algorithme. Un bon partitionnement devrait vérifier les propriétés suivantes :

- le nombre de tâches élémentaires doit être supérieur d'au moins un ordre de grandeur au nombre de processeurs de la machine cible (ceci afin de faciliter les choix d'implémentation ultérieurs) ;
- la redondance des instructions et des données au sein des tâches élémentaires doit être minimale (dans le cas contraire, l'algorithme obtenu pourrait ne pas pouvoir supporter l'augmentation de la taille du problème) ;
- les tâches élémentaires doivent représenter des charges de travail sensiblement équivalentes (si tel n'est pas le cas, il sera difficile par la suite d'équilibrer la charge entre les processeurs) ;
- le nombre de tâches élémentaires doit être une fonction croissante de la taille du problème (sinon, il pourrait s'avérer impossible d'utiliser plus de processeurs pour résoudre un problème plus grand).

5.1.2 Définition des schémas de communication

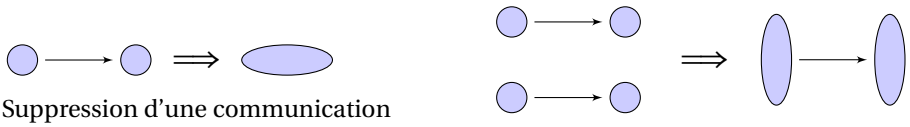
Une fois les tâches élémentaires identifiées, il s'agit de déterminer le schéma de communication qu'il est nécessaire de mettre en place pour implémenter l'algorithme. Nous pouvons distinguer essentiellement deux types de communications : les communications locales, qui n'impliquent qu'un nombre restreint de tâches, et les communications globales, faisant intervenir la plupart, si ce ne sont toutes les tâches élémentaires. Bien qu'il soit pertinent de noter la présence de communications globales, ces dernières ne sont pas réellement importantes à ce stade de la conception de l'algorithme. Leur effet principal est d'introduire un point de synchronisation dans l'algorithme, mais comme une large majorité des tâches sont impliquées dans une communication globale, elles n'introduisent pas de réel déséquilibre, et la seule façon de les optimiser réside dans le câblage des connexions physiques du réseau reliant les unités de calcul. À l'inverse, les communications locales peuvent introduire un réel déséquilibre et un surcoût important à l'exécution si elles ne sont pas optimisées. Nous devons donc les prendre en compte, et pour cela, nous construisons un graphe dans lequel chaque nœud représente une tâche, et chaque arête reliant une tâche à une autre correspond à l'introduction d'une communication locale entre ces deux tâches. Les communications inter–processeurs représentent une part non–négligeable du surcoût introduit par le parallélisme, en ce sens qu'un algorithme séquentiel n'en génère pas. Il est donc très important d'évaluer la structure du schéma de communication pour se faire une idée des propriétés de l'algorithme. Ainsi, un schéma de communication sera considéré efficace si :

- les communications sont réparties équitablement entre les tâches ;
- les communications peuvent être réalisées de façon concurrente ;
- les temps de communication peuvent être recouverts par des phases de calcul.

5.1.3 Agglomération des tâches élémentaires

L'objectif des deux premières étapes est de fournir une description de l'algorithme à implémenter en identifiant le maximum de sources de parallélisme possible. Cependant, si le nombre de tâches élémentaires est très supérieur au nombre de processeurs disponibles sur la machine cible, une telle granularité peut singulièrement nuire à l'efficacité de l'algorithme. Le simple fait de générer la totalité de ces tâches élémentaires peut s'avérer coûteux, et le schéma de communication devient également prohibitif. C'est pourquoi nous introduisons une phase d'agglomération, qui consiste à regrouper les tâches élémentaires en tâches plus importantes, en prenant en compte les caractéristiques de la machine cible. L'un des objectifs de cette phase d'agglomération est de préserver la localité de l'algorithme parallèle : l'idée consiste à regrouper ensemble des tâches élémentaires qui communiquent entre elles. De cette façon, on minimise les communications entre les tâches agglomérées. Une autre idée retenue pour cette phase d'agglomération consiste à regrouper d'un côté les tâches qui envoient des messages, et de l'autre celles qui les reçoivent. De cette façon, les messages élémentaires peuvent être regroupés en un unique message, ce qui permet de minimiser la latence des communications. Ces deux idées sont illustrées sur la figure 5.2. Une façon simple de réaliser cette phase consiste à utiliser un algorithme de partitionnement de graphe (voir par exemple [80, 81, 83, 128]). La qualité de l'agglomération des tâches peut être évaluée à partir des critères suivants :

- l'agglomération a augmenté la localité de l'algorithme ;



(a) Suppression d'une communication

(b) Réduction du nombre de message

FIG. 5.2 : Exemples de techniques d'agglomération de tâches élémentaires

– la quantité de données dupliquées reste faible ;
– les coûts d'exécution et de communication des tâches sont comparables ;
– le nombre de tâches est une fonction croissante de la taille du problème.

5.1.4 Placement des tâches

Le placement des tâches consiste à distribuer les tâches agglomérées sur les processeurs disponibles. Bien évidemment, si le nombre de tâches est fixe, et que la phase d'agglomération a permis de construire autant de tâches que de processeurs, cette phase devient triviale. Il n'est cependant pas rare de rencontrer des algorithmes dont le nombre de tâches varie au cours de l'exécution, et dans ce cas, la phase de placement prend tout son sens. L'objectif consiste généralement à trouver un équilibre entre une utilisation maximale des capacités de la machine, et une minimisation du volume de communications. Il existe pour cela différentes stratégies de placement, comme par exemple l'utilisation d'algorithmes d'ordonnancement ou d'équilibrage. La figure 5.3 présente un arbre de décision permettant de choisir l'une ou l'autre de ces méthodes. En général, le placement des tâches sera considéré comme pertinent si les conditions suivantes sont vérifiées :
– dans le cas d'un placement statique, le nombre de tâches par processeurs devrait être de l'ordre de la dizaine au plus ;
– dans le cas d'un placement dynamique, le processus de distribution des tâches ne doit pas constituer un goulot d'étranglement lors de l'exécution.

5.2 Modèles de parallélisme pour notre méthode adaptive

Avant de chercher à appliquer la méthode générale présentée ci–dessus, nous présentons plus en détail la structure de notre méthode adaptative. Nous cherchons à mettre en évidence les différentes formes de parallélisme présentes dans les algorithmes que nous avons présentés précédemment. Nous allons voir que la phase d'adaptation de la géométrie et la phase de résolution reposent sur des principes radicalement différents, et qu'il sera pertinent de chercher à les paralléliser selon des schémas différents.

5.2.1 Structure globale de notre méthode adaptative

Rappelons tout d'abord le principe de notre méthode adaptative. Considérons un problème modèle (de LAPLACE ou de STOKES), posé sur un domaine borné Ω de $\mathbb{R}^n$. Nous disposons d'une hiérarchie de triangulations $(\mathcal{T}_i)_{1\leq i\leq N}$ de ce domaine. Pour tout i appartenant à $[\![0,N]\!]$, nous notons Ω_i la réunion des éléments non–raffinés de $\mathcal{T}_i$, et $\mathcal{T}_h(\Omega_i)$ la triangulation induite sur Ω_i. Ainsi, les sous–domaines Ω_i présentent les caractéristiques suivantes :

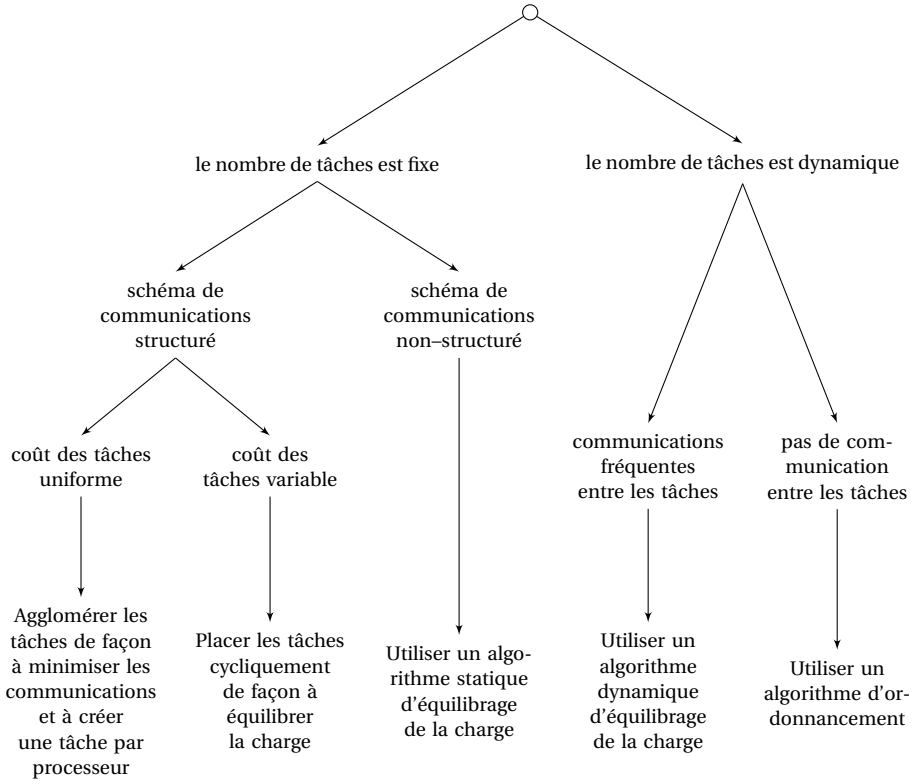


FIG. 5.3 : Arbre de décision permettant de choisir une stratégie de placement

– ils forment une partition de Ω ;
– ils présentent chacun un niveau de raffinement différent ;
– prises séparément, les triangulations $\mathscr{T}_h(\Omega_i)$ sont des triangulations conformes ;
– la réunion des triangulations $\mathscr{T}_h(\Omega_i)$ forme une triangulation non–conforme de Ω, dont les non–conformités vérifient les propriétés énumérées au paragraphe 4.2.4.

En utilisant la méthode numérique présentée dans le chapitre précédent, nous pouvons résoudre numériquement ce problème. Rappelons qu'il s'agit d'une méthode de décomposition de domaine de type éléments joints, définie pour les deux versions de la méthode VEF de Trio-U. Dans ce cadre, notre domaine se trouve naturellement découpé selon la notion de niveau de raffinement. Une fois la solution obtenue, nous utilisons un estimateur adapté au problème étudié pour localiser l'erreur numérique. Si cela s'avère nécessaire, nous pouvons alors construire un indicateur de raffinement et appliquer l'algorithme 3.13 pour construire une nouvelle hiérarchie de triangulations, tout en prenant soin d'interpoler convenablement les champs au cours de ce processus (voir le paragraphe 4.3). En itérant ce procédé, nous obtenons une méthode adaptative susceptible de s'intégrer à la plate–forme Trio-U.

Nous résumons cette démarche dans la figure 5.4. Dans ce schéma, les tâches regroupées dans l'ensemble rouge contribuent à la phase de résolution du problème sur une géométrie hiérarchique donnée, alors que les tâches regroupées dans l'ensemble vert correspondent à la phase d'adaptation de la géométrie une fois la résolution du problème terminée. Ces deux ensembles de tâches se distinguent essentiellement par leurs visions respectives de la géométrie de calcul. En effet, comme nous l'avons souligné au cha-

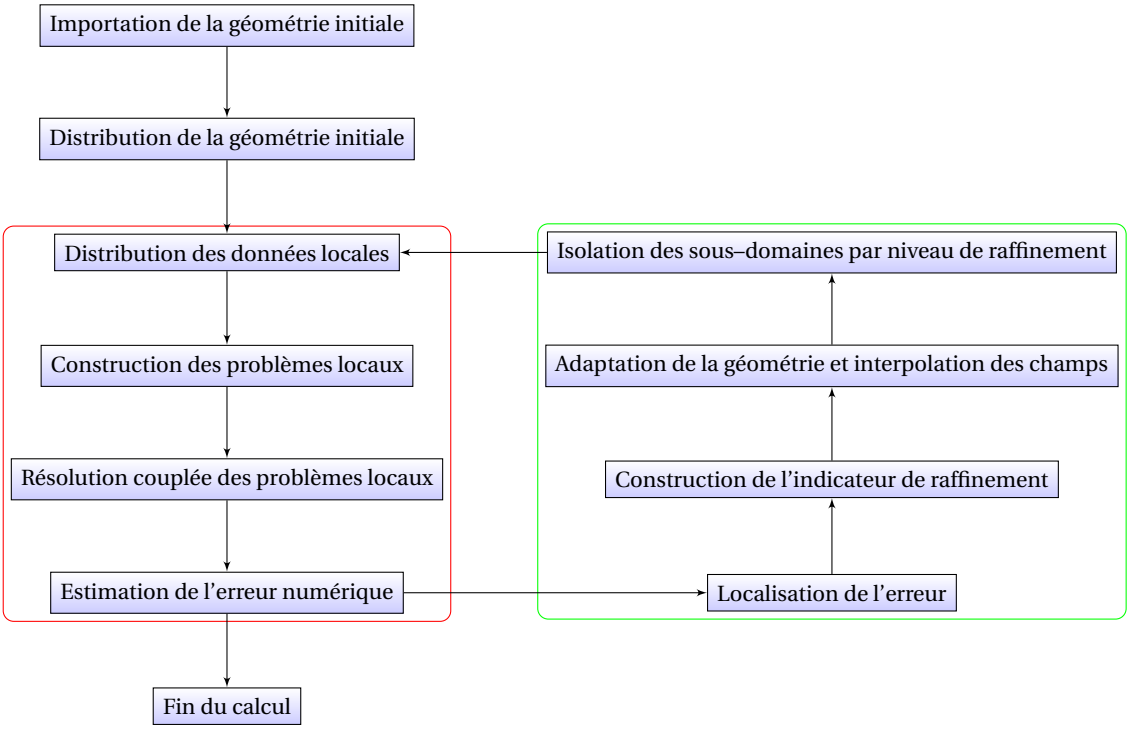
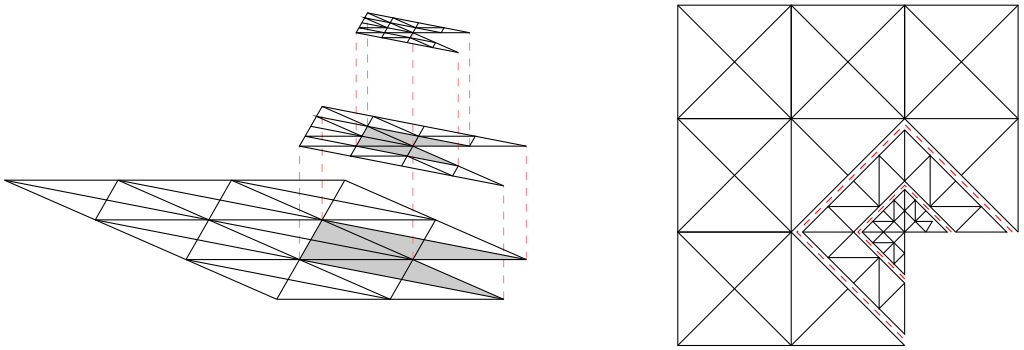


FIG. 5.4 : Algorithme général de notre méthode adaptative

pitre 3, les algorithmes géométriques travaillent sur une hiérarchie de triangulations, en utilisant pleinement les informations de raffinement décrivant l'historique du maillage. À l'inverse, comme nous l'avons indiqué dans le chapitre 4, les algorithmes numériques ne travaillent que sur les parties non–raffinées de ces triangulations. Nous appelons *géométrie hiérarchique* la perception de la géométrie de calcul utilisée par les algorithmes d'adaptation, et *géométrie aplatie* la vision de la géométrie utilisée par les algorithmes numériques. Ces deux visions de la géométrie sont illustrées sur la figure 5.5. Étant donné



(a) Géométrie hiérarchique

(b) Géométrie aplatie

FIG. 5.5 : Les deux visions de la géométrie utilisées dans notre méthode adaptative

que les données nécessaires à ces deux familles d'algorithmes sont radicalement différentes, il semble naturel que leur parallélisation repose sur des concepts différents. Cette question fait l'objet des deux sections qui suivent.

5.2.2 Parallélisation de la phase d'adaptation du maillage

L'objectif des paragraphes qui suivent est de fournir une analyse du parallélisme de l'algorithme 3.13, et d'en proposer une implémentation parallèle. Nous pouvons partitionner notre problème en tâches élémentaires à l'aide d'une approche structurelle : chaque tâche élémentaire correspond au travail à effectuer sur un des éléments de la hiérarchie de triangulation en cours. Ainsi, les communications introduites par l'algorithme 3.13 se résument aux communications reliant un élément d'un niveau de raffinement donné à ses éléments fils dans le niveau suivant, ainsi qu'à son élément père dans le niveau précédent. Nous ne voyons pas apparaître de communication entre les éléments d'un même niveau de raffinement. L'agglomération des tâches élémentaires peut alors permettre de supprimer l'intégralité des communications, en commençant par regrouper les tâches élémentaires selon la filiation des éléments. Nous obtenons alors autant de tâches qu'il y a d'éléments dans le maillage initial. Notons cependant que ces tâches ne présentent pas toutes le même coût, car il peut exister de fortes disparités entre les niveaux de raffinement des éléments. De même, soulignons que ces coûts varient au cours de la simulation. C'est pourquoi la stratégie de placement idéale pour l'algorithme d'adaptation consiste à choisir un algorithme d'équilibrage dynamique, dont l'objectif consiste à distribuer équitablement les éléments du maillage initial ainsi que leurs éléments fils entre les processeurs.

Dans l'analyse qui vient d'être faite, nous avons cependant ommis une phase très importante de l'algorithme d'adaptation : il s'agit de l'identification des sous–domaines qui seront utilisés pour lancer un nouveau calcul. Nous devons en effet être en mesure de regrouper les éléments selon leur niveau de raffinement pour pouvoir construire la géométrie aplatie, nécessaire à la phase de résolution du problème. Or, pour réaliser ce travail, nous devons également identifier les interfaces qui définissent ces sous–domaines, et pour cela, nous avons besoin d'une information cruciale : la relation de voisinage entre les éléments du maillage. Cette relation nous indique, pour un élément donné, quels sont ses éléments voisins, et quel est leur niveau de raffinement. Or le parallélisme de données que nous avons exhibé plus haut ne prend pas en compte les communications nécessaires pour maintenir cette structure de données lors de l'adaptation du maillage. Nous devons donc proposer une démarche qui permette de conserver cette information lors de l'exécution en parallèle de notre algorithme d'adaptation.

Pour ce faire, deux approches sont envisageables :

- trouver un moyen de reconstruire de la connectivité à partir d'une hiérarchie de triangulations adaptée ;
- modifier l'algorithme 3.13 pour qu'il maintienne cette connectivité et proposer une autre approche pour le paralléliser.

La première approche a ceci de satisfaisant que le travail réalisé jusqu'ici peut être exploité pleinement. Cependant, la seule technique dont nous disposons alors pour identifier les relations de voisinage entre éléments repose sur des prédicats géométriques (comparaison des coordonnées des sommets, tests d'appartenance d'un sommet à une face, etc). Or ces tests géométriques sont nécessairement entâchés d'une erreur due à la comparaison de nombre réels, et la stabilité de l'algorithme semble relativement compromise. Nous nous tournons donc vers la seconde approche, et nous considérons que la connectivité des éléments est une donnée qui doit être mise–à–jour en même temps que la hiérarchie de triangulations et les champs calculés. Précisons tout d'abord la notion de

connectivité qui nous intéresse en donnant quelques définitions.

Définition 5.1 : *En dimension 2 (resp. 3), nous dirons que deux éléments sont adjacents s'ils présentent le même niveau de raffinement et si leur intersection est un segment (resp. polygone) non dégénéré.*

Définition 5.2 : *Nous dirons qu'un élément K_1 est incident à un autre élément K_2 si ces deux éléments sont adjacents, ou bien s'il existe un élément K_3 tel que K_2 soit issu du raffinement de K_3 et que K_1 et K_3 soient adjacents.*

Définition 5.3 : *L'incidence d'un élément est la donnée de l'ensemble des éléments qui lui sont incidents.*

L'information pertinente pour notre démarche est la donnée de l'incidence de chaque élément de chaque niveau de la hiérarchie de triangulations. Les raisons de ce choix sont les suivantes :

- cette notion de connectivité permet de détecter simplement les interfaces entre les niveaux de raffinement.
- les informations supprimées lors la phase de déraffinement de l'algorithme 3.13 ne sont pas essentielles, car l'information d'incidence reste valable au niveau sous–jacent ;

La première propriété peut être établie en remarquant qu'une non–conformité apparait dès qu'un élément non–raffiné n'appartient pas à l'incidence de l'un des éléments non–raffinés qui lui sont incidents. La seconde propriété permet quant à elle de faciliter la maintenance de la relation de connectivité lors de l'application de l'algorithme 3.13. Il faut cependant définir correctement la relation d'incidence des éléments générés lors de la phase de raffinement, ce qui modifie singulièrement le schéma de communication entre les tâches élémentaires identifiées plus haut. En effet, nous voyons maintenant apparaître des communications entre des tâches élémentaires correspondant à des éléments issus du raffinement d'éléments initiaux différents. Ainsi, en conservant la méthode d'agglomération des tâches élémentaires présentée ci–dessus, nous pouvons toujours utiliser comme stratégie de placement des tâches un algorithme d'équilibrage dynamique, mais ce dernier doit, en plus de chercher à équilibrer la charge, minimiser les communications générées par la maintenance de la relation d'incidence. Pour des raisons de simplicité d'implémentation, nous avons décidé de relaxer la contrainte d'équilibrage pour les algorithmes d'adaptation de la géométrie. Nous avons choisi de distribuer de façon équilibrée les éléments du maillage initial à l'aide d'un algorithme d'équilibrage statique, et de ne pas introduire de phase de redistribution des tâches qui leur sont associées. Ainsi, chaque processeur se voit confier la gestion du raffinement des éléments qui lui ont été attribués à l'initialisation du calcul, et les schémas de communication nécessaires à la maintenance de la relation d'incidence restent déterministes.

5.2.3 Parallélisation de la phase de résolution

Nous étudions maintenant les caractéristiques d'une implémentation parallèle de la phase de résolution. La méthode numérique que nous proposons pour résoudre un problème posé sur une géométrie aplatie repose sur une approche similaire à celle présentée par C. Bernardi, F. Hecht et Y. Maday dans les articles [21, 22]. Il s'agit essentiellement d'une méthode de décomposition de domaine, qui, comme nous l'avons souligné dans le chapitre 2, présente un caractère intrinsèquement parallèle. Le cadre naturel

pour l'implémentation parallèle de ce genre de méthode est un parallélisme de tâches, chaque tâche correspondant à la résolution de l'un des problèmes locaux. Nous avons par ailleurs renforcé cette idée dans le paragraphe 4.2.3, en introduisant une extension au formalisme des éléments joints de façon à assurer le caractère bien posé de chacun des problèmes locaux. Tout ceci nous conduit à choisir un modèle de parallélisme fonctionnel pour le partitionnement de la phase de résolution. Les communications sont essentiellement matérialisées par les interfaces séparant les sous-domaines. Ainsi, notre graphe de tâches présente un nœud pour chaque sous–domaine, et une arête reliant deux nœuds s'ils s'agit de sous–domaines voisins. Pour réaliser les étapes d'agglomération et de placement, nous optons pour l'utilisation d'un algorithme de partitionnement de graphe, qui découpera le graphe décrit ci–dessus en autant de composantes que nous avons de processeurs disponibles sur la machine cible, tout en minimisant le nombre d'arêtes coupées. Afin d'assurer l'équilibrage de la charge, nous pouvons pondérer ce graphe, en associant un poids à chaque nœud et à chaque arête. Le poids associé à un nœud du graphe modélise le coût de la résolution du problème local associé et sera par exemple égal au nombre de degrés de liberté présents dans ce problème. Le poids associé à une arête modélise le coût d'une communication engendrée par le couplage défini sur l'interface associée, et sera par exemple égal au nombre de degrés de liberté utilisés pour définir la valeur interfaciale. Nous disposons là d'une première approche permettant de réaliser un équilibrage dynamique de la charge, en exécutant l'algorithme de partitionnement de graphe dès l'instant que la géométrie a été modifiée.

L'approche que nous venons de présenter reste cependant relativement grossière. En effet, considèrons par exemple un domaine de calcul bidimensionnel contenant 250 éléments au départ, sur lequel est défini un problème dont la solution présente une singularité très forte et très localisée. Supposons que notre algorithme adaptatif modifie notre domaine de calcul en raffinant 4 fois l'élément contenant cette singularité. Nous obtenons un maillage qui contient 249 éléments grossiers, et $4^4 = 256$ éléments fins. Nous venons de doubler le nombre d'éléments dans le maillage, mais si nous utilisons la technique décrite ci–dessus pour résoudre ce problème sur 3 processeurs, nous ne pourrons pas faire mieux que distribuer les 256 éléments fins sur l'un des processeurs, tandis que les deux processeurs restants se partagent les 249 éléments grossiers restant. Nous obtenons là un algorithme fort peu équilibré. Ceci provient du fait que la granularité des tâches élémentaires n'est pas suffisamment fine.

Pour remédier à ce problème, nous proposons de recourir à la solution suivante. Le partitionnement du problème se fait maintenant selon un parallélisme structurel, en définissant une tâche élémentaire pour chaque élément de la géométrie aplatie. Naturellement, les communications correspondent alors aux relations de voisinage entre éléments dans le maillage, en prenant en compte les non–conformités. Nous pouvons donc construire un autre graphe, qui possède un nœud par élément de la géométrie, et une arête entre deux nœuds s'ils correspondent à des éléments adjacents (au sens large). Nous pouvons encore une fois appliquer un algorithme de partitionnement de graphe pour découper ce graphe en autant de parts qu'il y a de processeurs disponibles, tout en minimisant le nombre d'arêtes coupées. Cependant, il est alors possible que les éléments attribués à un processeur ne présentent pas tous le même niveau de raffinement. Qu'à cela ne tienne, il suffit alors, pour chaque processeur, de regrouper les éléments qui lui ont été affectés selon leur niveau de raffinement. De cette façon, chaque processeur se voir affecter plusieurs sous–domaines, mais le nombre total d'éléments affec-

tés à chaque processeur est alors bien équilibré, et ce indépendamment de la forme du maillage adapté. Dans ce cas présenté ci–dessus, nous attribuerons environ 168 éléments à chaque processeur, indépendemment de leur niveau de raffinement.

5.3 Heuristiques de partitionnement de graphe

Nous avons fait le choix, pour équilibrer dynamiquement la phase de résolution, de recourir à un algorithme de partitionnement de graphe. Nous pouvons définir une version abstraite et générique du problème à résoudre :

Problème 5.1 : *Considérons un graphe constitué de N_s sommets $(S_i)_{1\leq i\leq N_s}$ pondérés par les poids $(\alpha_i)_{1\leq i\leq N_s}$ et de N_a arêtes $(A_i)_{1\leq i\leq N_a}$ pondérées par les poids $(\beta_i)_{1\leq i\leq N_a}$. Nous cherchons à partitionner ce graphe en p composantes de façon à équilibrer la somme des poids des sommets affectés à chaque composante, tout en minimisant la somme des poids des arêtes coupées par ce partitionnement.*

Ce problème appartient à la classe de problèmes dits « NP–complets » [66]. Autrement dit, il n'existe pas d'algorithme permettant de résoudre systématiquement ce problème qui présente une complexité algorithmique polynômiale. Nous devons donc nous tourner vers des heuristiques qui fourniront un partitionnement honorable en un temps acceptable. Il existe de nombreuses heuristiques permettant de partitionner un graphe. Elles se caractérisent par la qualité des partitionnement obtenus, ainsi que par leur complexité algorithmique. Nous pouvons citer par exemple l'algorithme de B. KERNIGHAN et S. LIN et ses variantes, les méthodes reposant sur la bissection récursive (bissection récursive spectrale, bissection récursive d'adjacence), et les approches multi–niveaux. Nous présentons succintement trois de ces approches dans les paragraphes qui suivent.

5.3.1 Algorithme de B. KERNIGHAN et S. LIN

L'algorithme de B. KERNIGHAN et S. LIN a été proposé en 1970 dans l'article [84]. Plutôt que de traiter directement la construction du partitionnement du graphe, cet algorithme fournit le moyen d'optimiser un partitionnement existant. C'est pourquoi il est nécessaire de spécifier un partitionnement initial, qui peut être établi en distribuant les nœuds au hasard, ou bien à l'aide d'un autre algorithme de partitionnement. Partant de là, l'algorithme de B. KERNIGHAN et S. LIN parcourt les nœuds du graphe, en décidant pour chaque nœud s'il est nécessaire ou souhaitable de l'attribuer à une autre partition. La décision du changement d'affectation d'un nœud repose sur l'analyse du voisinage du nœud en question. Il s'agit de compter, pour chaque partition, le nombre de nœuds voisins au nœud considéré qu'elle contient. On décide alors d'affecter le nœud à la partition qui rassemble le plus de nœuds voisins. Ainsi, la complexité de cet algorithme est de $\mathcal{O}(N_s N_v)$, où N_s désigne le nombre de sommets du graphe, et N_v le nombre moyen de voisins pour un sommet. Cependant, l'affectation d'un nœud est susceptible d'évoluer en fonction de l'affectation de ses voisins, et il semble nécessaire d'appliquer plusieurs fois ce processus pour converger vers un partitionnement de qualité acceptable.

5.3.2 Méthode de bissection récursive spectrale

La méthode de bissection récursive spectrale [72, 109] repose sur les propriétés des éléments propres du Laplacien du graphe. Considérant un graphe non pondéré, son Laplacien est une matrice L, appartenant à $\mathscr{M}^{N_s}(\mathbb{R})$, présentant le même graphe d'adjacence, et définie comme suit :

$$L_{ij} = \begin{cases} -1 & \text{si } i \neq j \text{ et si } (S_i, S_j) \text{ est une arête du graphe} \\ \deg(i) & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$$

où $\deg(S_i)$ désigne le degré du sommet S_i, à savoir son nombre de voisins. Si le graphe considéré n'est pas orienté, la matrice L est symétrique. On peut également montrer qu'elle est semi–définie négative. Ainsi, elle admet 0 comme plus petite valeur propre. Le deuxième vecteur propre de L, associé à la deuxième plus petite valeur propre de L, est appelé vecteur de FIEDLER [56]. Ce vecteur présente une propriété remarquable : le signe de ses composantes distribue les nœuds du graphe en deux sous–ensembles équilibrés. Il peut être déterminé en utilisant une méthode de LANCZOS [121]. Il suffit d'appliquer cet algorithme récursivement sur les sous–graphes pour découper le graphe initial en 2^n composantes. Cette approche permet de construire un partitionnement de qualité, mais elle reste cependant limitée, car sa généralisation au cas d'un graphe pondéré n'est pas évidente.

5.3.3 Algorithme de partitionnement multi–niveaux

Le principe de l'algorithme multi–niveaux proposé par G. KARYPIS et V. KUMAR est détaillé dans les articles [80, 81]. Il consiste à procéder en trois phases :

- agglomération des nœuds du graphe ;
- partitionnement du graphe grossier ;
- raffinement du graphe partitionné.

L'agglomération des nœuds du graphe consiste à définir un graphe plus grossier en fusionnant les nœuds reliés par les arêtes de poids important. On obtient alors un graphe grossier contenant moins de nœuds que le graphe initial, que l'on peut partitionner en utilisant par exemple, une variante de l'algorithme de B. KERNIGHAN et S. LIN, comme celle de C.-M. FIDUCCIA et R.-M. MATTHEYSES [55]. Lors de la phase de raffinement, on remplace les nœuds agglomérés par les nœuds réels, en cherchant à optimiser le placement des nœuds situés sur les frontières. Cette approche permet une grande liberté quant au choix des algorithmes utilisés à chaque étape.

Les codes METIS et PARMETIS [80–83, 129] constituent deux bibliothèques logicielles regroupant plusieurs heuristiques de partitionnement de graphe reposant sur l'approche multi–niveaux. METIS est destinée à une utilisation séquentielle, alors que PARMETIS repose sur le standard MPI [70, 71, 99, 100, 110] pour pouvoir être utilisée dans un contexte parallèle. De nombreuses options permettent d'affiner le choix des algorithmes d'agglomération, de partitionnement élémentaire, et de raffinement du graphe. Ainsi, ces bibliothèques permettent de réaliser une campagne de tests en vue d'optimiser l'algorithme global de partitionnement. Nous renvoyons à la documentation fournie avec ces logiciels pour le détail des options disponibles. La possibilité offerte par le logiciel PARMETIS

de réaliser le partitionnement d'un graphe distribué sur plusieurs processeurs représente pour le cas de notre méthode une opportunité tout–à–fait intéressante. Cette technologie nous permet en effet d'envisager une implémentation parallèle de la deuxième approche proposée pour paralléliser la phase de résolution de notre méthode adaptative.

5.4 Développements à apporter à la plate–forme Trio-U

Nous présentons maintenant une liste non–exhaustive des développements qu'il est nécessaire d'intégrer dans la plate–forme Trio-U pour envisager la mise en œuvre de notre méthode adaptative. Nous présentons, pour chaque notion abordée, les différences essentielles entre les besoins relatifs à notre démarche, et les outils disponibles dans la plate–forme.

5.4.1 Notion de géométrie hiérarchique distribuée et dynamique

Notre méthode nécessite impérieusement une description hiérarchique de la géométrie afin de prendre en compte convenablement les aspects géométriques de l'adaptation de maillage. À l'heure actuelle, Trio-U ne suporte qu'une géométrie aplatie conforme, distribuée statiquement sur les processeurs à l'initialisation du calcul. Nous devons donc mettre en place les structures de données nécessaires à la description :

- d'une hiérarchie de triangulations ;
- de la relation d'incidence des éléments de ces triangulations ;
- de la géométrie aplatie non–conforme provenant du regroupement des éléments non–raffinés selon leur niveau de raffinement ;
- des champs définis sur la hiérarchie de triangulations.

En plus de ces structures de données, nous devons développer les algorithmes permettant de les mettre à jour lors de l'application de l'algorithme d'adaptation. Il s'agit là d'un développement entièrement nouveau, car la plate–forme n'offre à l'heure actuelle aucun support pour ce genre de démarche.

5.4.2 Algorithmes d'équilibrage dynamique de la charge

Étant donné que Trio-U utilise une géométrie statique, la problématique de l'équilibrage dynamique de la charge ne présentait pas jusqu'ici d'intérêt majeur pour la plate–forme. Le partitionnement initial du maillage permet en effet d'équilibrer statiquement la charge, ce qui est tout–à–fait suffisant. Notons que ce partitionnement peut être réalisé via une interface vers la bibliothèque METIS.

Rappelons que pour des raisons de simplification d'implémentation, nous choisissons un équilibrage statique des algorithmes d'adaptation de maillage. Cet équilibrage est réalisé à l'initialisation du calcul, en partitionnant le maillage initial en autant de sous–domaines qu'il y a de processeurs disponibles sur la machine, comme cela est fait pour un calcul standard avec Trio-U. Chaque processeur doit alors prendre en charge la gestion du raffinement des éléments qui lui ont été affectés initialement.

En ce qui concerne la phase de résolution, nous avons choisi de recourir à un algorithme d'équilibrage dynamique de la charge. Cet algorithme consiste à construire un graphe pondéré décrivant la géométrie aplatie de façon plus ou moins précise (selon que

les nœuds du graphe correspondent aux sous–domaines de calcul ou aux éléments de la géométrie aplatie), et à partitionner ce graphe en autant de sous–domaines qu'il y a de processeurs disponibles. Cet algorithme cherche à équilibrer la charge sur les processeurs en minimisant les communications lors de la résolution. Notons qu'il est possible de prendre en compte la distribution des données hiérarchiques comme distribution initiale lors de l'appel à l'algorithme de partitionnement. Nous pouvons ainsi chercher à minimiser les communications correspondant à la distribution des problèmes locaux sur les processeurs lors de leur création. Afin de mettre en œuvre efficacement cet algorithme de partitionnement, en particulier lorsque le graphe construit correspond aux relations de voisinage des éléments (voir le paragraphe 5.2.3), nous envisageons de fournir une interface vers la bibliothèque PARMETIS. Nous disposerons ainsi d'une version parallèle de l'algorithme de partitionnement, ce qui permet d'éviter qu'il ne devienne un goulot d'étranglement séquentiel dans la démarche adaptative globale.

5.4.3 Mécanisme de génération de problèmes locaux

Classiquement, dans Trio-U, la création d'un problème est réalisée par l'interprétation d'un jeu de données contenu dans un fichier. Lors d'une exécution en parallèle du programme, seul le processeur maître lit les informations contenues dans ce fichier. Il les diffuse ensuite aux autres processeurs, et enfin chaque processeur interprète les données reçues séparément. Bien entendu, étant donné le modèle de parallélisme de Trio-U, l'interprétation des données peut donner lieu à des communications entre processeurs, comme, par exemple, lors de la phase de construction des vecteurs distribués.

Dans notre méthode adaptative, nous construisons les problèmes locaux de façon indépendante. Plus précisément, la résolution du problème global posé sur la géométrie aplatie est réalisée via un algorithme général qui considère chacun des problèmes locaux comme étant indépendant des autres. Ainsi, à chaque problème local correspond un jeu de données indépendant de celui des autres problèmes locaux. La distribution des problèmes locaux est réalisée par l'envoi de ce jeu de données du processeur qui possède les données hiérarchiques associées vers le processeur qui doit prendre en charge sa résolution. Cependant, à l'inverse de ce qui est réalisé classiquement dans Trio-U, pour assurer l'indépendance des problèmes locaux, nous devons assurer que l'interprétation des jeux de données associés ne génèrera pas de communications entre les processeurs. Pour réaliser cela, nous utilisons la notion de groupe de processeurs présente dans la plate–forme. Lorsqu'un processeur doit interpréter un jeu de données associé à un problème local, nous le plaçons dans un groupe dont il est le seul membre. De cette façon, ce processeur aura alors l'impression d'être en train d'exécuter un code séquentiel, et ne génèrera par conséquent pas de communications lors de l'interprétation du jeu de données.

5.4.4 Algorithmes de résolution

La résolution des systèmes linéaires provenant de la discrétisation du problème posé sur la géométrie aplatie peut être réalisée de différentes façons. Une première manière de procéder consiste à chercher à résoudre le système de SCHUR obtenu en appliquant une élimination de GAUSS pour ne conserver que les inconnues définies sur les interfaces.

Chaque étape de la résolution de ce système nécessite alors la résolution de chacun des problèmes locaux pris isolément.

Nous pouvons également considérer la résolution du système linéaire global sans procéder à une élimination de GAUSS. Remarquons que ce système peut être construit en assemblant les contributions de chaque sous–problème en prenant en compte les matrices de couplage issues de la méthode d'éléments joints. Ainsi, multiplier un vecteur par la matrice globale peut être réalisé de la façon suivante :

- conversion du vecteur global en vecteurs locaux, en appliquant les transposées des matrices de couplage ;
- multiplication des vecteurs locaux par les matrices locales ;
- assemblage des résultats locaux pour générer le résultat global en appliquant les matrices de couplage.

Dans le cas d'un problème de LAPLACE découpé en deux sous–problèmes, cette démarche revient à utiliser l'équations suivante, avec $E_{\Gamma\Gamma} = C_\Gamma^1 A_{\Gamma\Gamma}^1 C_\Gamma^{1T} + C_\Gamma^2 A_{\Gamma\Gamma}^2 C_\Gamma^{2T}$:

$$\begin{bmatrix} A_{II}^1 & 0 & A_{I\Gamma}^{1T} C_\Gamma^{1T} \\ 0 & A_{II}^2 & A_{I\Gamma}^{2T} C_\Gamma^{2T} \\ C_\Gamma^1 A_I^1 \Gamma & C_\Gamma^2 A_{I\Gamma}^2 & E_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_\Gamma \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 0 \\ 0 & C_\Gamma^1 \end{bmatrix} \begin{bmatrix} A_{II}^1 & A_{I\Gamma}^{1T} \\ A_{I\Gamma}^1 & A_{\Gamma\Gamma}^1 \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & 0 & C_\Gamma^{1T} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_\Gamma \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & 0 \\ I & 0 \\ 0 & C_\Gamma^2 \end{bmatrix} \begin{bmatrix} A_{II}^2 & A_{I\Gamma}^{2T} \\ A_{I\Gamma}^2 & A_{\Gamma\Gamma}^2 \end{bmatrix} \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & C_\Gamma^{2T} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_\Gamma \end{bmatrix}$$

Or, le produit matrice–vecteur est l'opération élémentaire des méthodes itératives de résolution de systèmes linéaires, telles que la méthode du Gradient Conjugué, la méthode GMRES ou l'algorithme d'UZAWA [121]. Nous pouvons donc envisager de résoudre notre problème de LAPLACE à l'aide d'un algorithme global de type Gradient Conjugué, car la matrice associée est symétrique définie positive. Dans le cas d'un problème de STOKES, nous pouvons utiliser un algorithme global de type UZAWA, qui permet de résoudre les problèmes s'écrivant sous la forme d'une recherche de point–selle. Afin d'accélérer la convergence de nos algorithmes, il est envisageable de les préconditionner à l'aide par exemple de la résolution locale de problèmes de DIRICHLET en imposant la valeur calculée sur les interfaces. Nous étudions cette question au paragraphe 6.5. Nous voyons donc que les algorithmes de résolution adaptés à notre méthode numérique reposent sur des concepts radicalement différents de ceux utilisés dans l'approche standard de Trio-U. Nous devons donc les implémenter et considérer qu'il s'agit d'une contribution nouvelle à la plate–forme.

5.4.5 Schémas de communication dynamiques

L'implémentation parallèle de Trio-U repose essentiellement sur la notion de vecteur distribué, présentée au chapitre 2. Étant donné que Trio-U utilise une distribution statique des données, la structure de ces vecteurs distribués ainsi que les échanges qu'ils encapsulent sont déterminés une fois pour toutes à l'initialisation du calcul. Nous ne pourrons malheureusement pas utiliser ces structures de données et les schémas de communications associés, car nous considérons une géométrie dynamique.

Dans notre méthode de résolution, les communications entre processeurs correspondent à l'échange de champs définis sur les interfaces des sous–domaines. La structure de données permettant de décrire ces schémas de communications est très voisine

de celle des vecteurs distribués de Trio-U, mais son mode de construction est sensiblement différent. Alors que les vecteurs distribués statiques de Trio-U sont initialisés à l'aide des relations de voisinage des éléments d'une triangulation conforme, les vecteurs distribués dynamiques que nous devons introduire sont créés à partir de la relation d'incidence des éléments de la géométrie hiérarchique, et leur structure évolue dès que la géométrie est mise à jour. Par ailleurs, les schémas de communications associés aux vecteurs distribués de notre méthode doivent pouvoir évoluer en fonction de la distribution des sous–problèmes sur les processeurs.

Remarquons enfin que si nous adoptons une méthode de résolution itérative portant sur le système globlal comme cela a été présenté ci–dessus, les communications liées à notre algorithme de résolution se présentent sous une forme très particulière. En considérant par exemple un algorithme de type Gradient Conjugué, nous voyons apparaître deux types de communications :

– des communications globales, intervenant dans les produits scalaires ;
– des communications locales, intervenant dans les produits matrice–vecteur.

Les communications globales sont réalisées de façon standard à l'aide d'opérations de réduction, ce qui ne présente pas de difficulté particulière. Les communications locales correspondent quant à elles à la conversion du vecteur global en vecteurs locaux, et à l'assemblage des résultats locaux en un résultat global. Ainsi, si le schéma de communication prend la responsabilité d'appliquer les matrices de couplages, nous pouvons simplifier considérablement l'implémentation de nos algorithmes de couplage. Or, cette configuration des vecteurs distribués dynamiques est tout–à–fait envisageable. En effet, les vecteurs distribués de notre méthode sont construits à partir de la relation d'incidence des éléments, qui fournit toutes les informations nécessaires au calcul des coefficients des matrices de couplage (voir paragraphe 4.2.4).

Ainsi, nous devons introduire de nouveaux vecteurs distribués dans Trio-U, implémentant les couplages liés à la méthode d'éléments joints, et permettant la définition de schémas de communications dynamiques. Il s'agit encore une fois d'un développement nouveau à introduire dans la plate–forme, car cette dernière ne propose pas d'outils de ce genre.

Résumé

Ce chapitre nous a permis de présenter et de justifier les choix que nous avons retenus pour l'implémentation parallèle de notre méthode. Cette réflexion repose sur la méthode générale de conception d'algorithmes parallèles proposée par I. FOSTER *dans [61]. Nous retenons de cette étude l'idée qu'il est nécessaire de séparer la parallélisation des algorithmes géométriques de celle des algorithmes de résolution.*

Pour des raisons de simplicité d'implémentation, nous optons pour un équilibrage statique de la phase d'adaptation géométrique, et pour un équilibrage dynamique reposant sur une heuristique de partitionnement de graphe pour la parallélisation de la phase de résolution numérique. Nous avons également énuméré en fin de chapitre les différentes contributions qu'il est nécessaire d'apporter à la plate–forme Trio-U pour mettre en œuvre ces idées.

CHAPITRE 6

Architecture logicielle et implémentation

Nous présentons à présent l'architecture logicielle que nous avons conçue pour implémenter notre méthode au sein de la plate–forme Trio-U. Avant toute chose, nous décrivons le principe des classes génériques de Trio-U, car nous l'utilisons fréquemment dans notre implémentation. Nous décrivons ensuite le rôle des différentes classes que nous avons développées, en les regroupant selon les concepts qu'elles représentent. Ce chapitre quelque peu technique se veut avant tout une description claire et concise de l'architecture logicielle que nous avons mise en place dans Trio-U pour pouvoir valider notre méthode, et, à ce titre, nous avons omis les détails de programmation les moins significatifs.

6.1 Vue d'ensemble et principe de fonctionnement

6.1.1 Notions d'architecture logicielle dans Trio-U

Trio-U est développé en C++ selon un modèle orienté objet. Pratiquement, cela signifie que chaque concept mathématique nécessaire à la description et à la résolution d'un problème numérique est implémenté dans une classe dédiée. Ceci permet de clarifier la conception du logiciel selon le principe de séparation des intérêts. Ainsi, il existe dans Trio-U des classes permettant de représenter un problème, une équation, une condition aux limites, mais également une discrétisation, une matrice ou un solveur d'algèbre linéaire. Plus exactement, ces concepts sont implémentés dans des hiérarchies de classes séparées. Au sein d'une même hiérarchie, les classes sont liées entre elles par la relation d'héritage propre à la programmation orientée objet.

De plus, Trio-U propose un mécanisme d'abstraction qui permet de traiter de façon générique et uniforme les classes représentant des concepts similaires. Ce mécanisme repose sur le modèle de conception de Pont [65] et sur le principe des méthodes virtuelles du C++ [138]. La figure 6.1 représente la structure de ce mécanisme sous forme de diagramme de classes UML[1].

Dans ce schéma, la classe de base spécifie l'interface commune à toutes les classes de la hiérarchie. Par exemple, dans Trio-U, la classe `Probleme_base` définit l'ensemble des méthodes communes à tous les problèmes que l'on peut vouloir étudier. Parmi ces

[1] Unified Modeling Language. Nous renvoyons à [2] et [30] pour une présentation détaillée de cet outil.

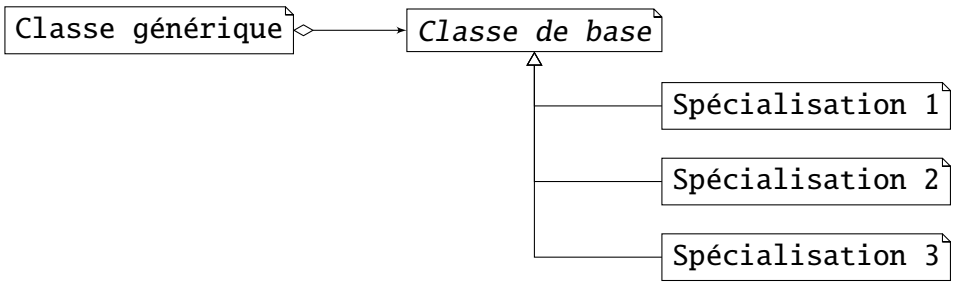


FIG. 6.1 : Mécanisme de généricité dans Trio-U

méthodes, on note l'accès à la description du domaine de calcul, à la discrétisation utilisée, au nombre d'équations qui composent ce problème, ou encore l'accès individuel à chacune de ces équations. Les méthodes déclarées dans la classe de base sont nécessairement virtuelles, nous verrons dans un instant pourquoi. Les classes qui héritent de la classe de base implémentent ces méthodes virtuelles en les spécialisant en fonction de leur nature propre. Par exemple, la classe `Probleme_Conduction` de Trio-U, qui représente un problème de conduction de chaleur dans un milieu, ne permet d'accéder qu'à une équation, l'équation d'évolution de la température. À l'inverse, la classe `Probleme_Thermohydraulique` possède deux équations, à savoir une équation de type NAVIER–STOKES, et une équation de conservation de l'énergie.

La classe générique permet quant à elle d'accéder de façon uniforme aux méthodes des classes dérivées de la classe de base. Pour ce faire, elle possède un pointeur sur une instance d'une des classes spécialisées, et elle délègue ses appels de méthodes à l'instance désignée par ce pointeur. Ainsi, il existe dans Trio-U une classe `Probleme`, qui possède un pointeur vers une instance d'une classe dérivée de *`Probleme_base`*, et qui permet de représenter uniformément tous les types de problèmes que Trio-U comprend. Le bon fonctionnement de ce mécanisme est assuré par le fait que les méthodes déclarées dans la classe de base sont virtuelles. De cette façon, si une classe dérivée redéfinit l'une des méthodes de la classe de base, et que le pointeur de la classe générique pointe en réalité vers une instance de cette classe dérivée, la méthode appelée sera en réalité la version implémentée dans la classe dérivée. Autrement dit, on assure par ce biais que les méthodes qui seront exécutées seront celles correspondant à la classe réelle de l'objet désigné par le pointeur de la classe générique.

6.1.2 Architecture générale du module `AMR`

L'implémentation du module `AMR` (Adaptive Mesh Refinement) que nous avons réalisée se compose de cinq sous–modules aux responsabilités bien spécifiques :

- le sous–module `Framework` ;
- le sous–module `Geometry` ;
- le sous–module `Mortar` ;
- le sous–module `Laplace` ;
- le sous–module `Stokes`.

Le sous–module `Framework` rassemble les classes de bases qui permettent de décrire et de résoudre un problème par notre méthode adaptative. Il s'agit donc d'un module

structurant, qui définit les relations entre les différentes hiérarchies de classes du module AMR. Le sous–module Geometry, comme son nom l'indique, regroupe les classes qui implémentent les structures de données et les algorithmes ayant trait à la gestion du maillage. Nous retrouverons donc naturellement dans ce module l'implémentation des concepts détaillés au chapitre 3. Nous avons conçu ce module comme un « module outil », présentant peu de classes d'interfaces, mais à la fonctionnalité bien précisée. Le sous–module Mortar définit les classes relatives à la résolution d'un problème posé sur une géométrie fixée pouvant présenter des non–conformités. Il s'agit essentiellement de l'implémentation de la méthode d'éléments joints présentée au chapitre 4. De la même façon que pour le module Geometry, le module Mortar a été développé de façon à être le plus indépendant possible, afin de fournir un outil simple d'utilisation. Enfin, les sous–modules Laplace et Stokes implémentent la spécialisation des classes de base du module Framework pour la résolution, respectivement, d'un problème de LAPLACE ou de STOKES. Il s'agit donc de modules applicatifs, dont les classes sont généralement simples, et dont les objectifs principaux sont de fournir une interface utilisateur claire pour la spécification du problème à résoudre et de spécialiser le comportement de certaines parties des algorithmes implémentés dans le module Framework. L'exemple le plus simple d'une telle spécialisation est le choix de la nature de la solution calculée : selon que l'on étudie un problème de LAPLACE ou de STOKES, cette inconnue sera scalaire ou vectorielle.

Nous allons maintenant présenter plus en détail la structure de chacun de ces modules. Nous commencerons par le module Framework, en présentant les détails de l'algorithme de résolution de notre méthode adaptative. Nous ferons également apparaître les classes spécialisées des modules Laplace et Stokes qui appartiennent à ces hiérarchies, car cela permettra de mettre en évidence ce qu'il est nécessaire de développer pour étendre les capacités de notre module à la résolution d'un nouveau type de problème. Nous détaillerons ensuite la conception du module Geometry, en insistant sur deux aspects importants, à savoir la mise en œuvre de l'adaptation du maillage ainsi que le traitement de la question de l'équilibrage de la charge. Enfin, nous présenterons l'implémentation du module Mortar, qui introduit une nouvelle vision du parallélisme dans Trio-U.

6.2 Le sous–module Framework

Le sous–module Framework s'articule autour de 5 hiérarchies de classes :
- la hiérarchie des AMR_Problems ;
- la hiérarchie des AMR_Timesteppers ;
- la hiérarchie des AMR_Fields ;
- la hiérarchie des AMR_Error_Estimators ;
- la hiérarchie des Local_Problems ;

Nous allons présenter chacune de ces hiérarchies dans les paragraphes qui suivent, et nous décrirons ensuite leurs intéractions dans l'algorithme global de résolution.

6.2.1 La hiérarchie des AMR_Problems

La hiérarchie des AMR_Problems est probablement la plus importante de tout le module AMR. Elle est représentée sur la figure 6.2. La classe de base de cette hiérarchie, à

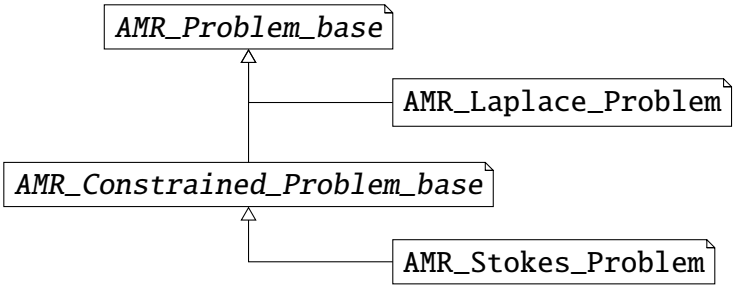


FIG. 6.2 : La hiérarchie des AMR_Problems

savoir la classe *AMR_Problem_base*, représente l'abstraction d'un problème à résoudre en utilisant notre approche adaptative. La classe *AMR_Constrained_Problem_base* qui en hérite correspond à la spécialisation de la classe *AMR_Problem_base* lorsque le problème à résoudre correspond à une recherche de point–selle, comme c'est le cas pour le problème de STOKES. Notons que la figure 6.2 fait également apparaître les classes AMR_Laplace_Problem et AMR_Stokes_Problem, respectivement implémentées dans les modules Laplace et Stokes. Introduire une telle hiérarchie permet de factoriser l'algorithme global de résolution adaptative du problème dans les classes de bases. Ainsi, le développement d'un module permettant de résoudre un nouveau type de problème (comme par exemple un problème de type transport passif ou un problème de type NAVIER–STOKES) s'en trouve grandement facilité.

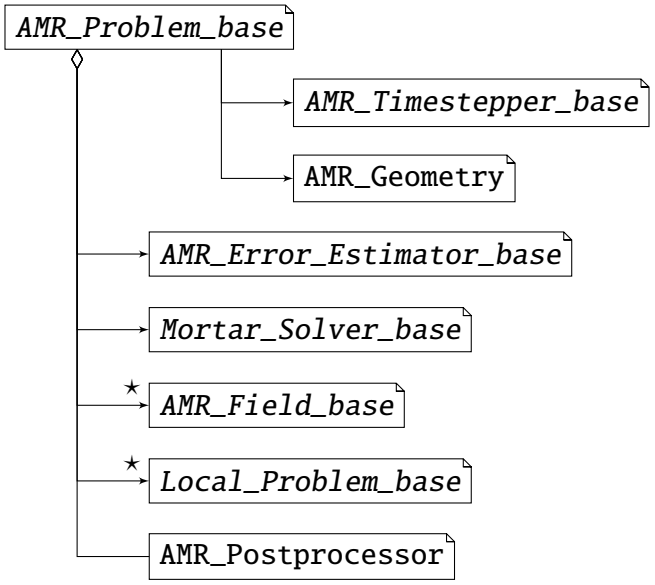


FIG. 6.3 : Principaux membres de la classe *AMR_Problem_base*

La figure 6.3 représente les principaux membres de la classe *AMR_Problem_base*. Nous voyons donc que cette classe est associée à un *AMR_Timestepper_base*, ainsi qu'à une AMR_Geometry, qui est la classe principale du module Geometry. De même, la classe *AMR_Problem_base* possède une référence vers un *AMR_Error_Estimator_base*, une autre vers un *Mortar_Solver_base* (qui est la classe de base des solveurs implémentés dans le module Mortar), ainsi qu'une liste de références vers des *AMR_Field_base* et

une autre liste de références vers des *Local_Problem_base*. Enfin, nous voyons qu'elle possède également un AMR_Postprocessor dont le rôle est de prendre en charge le post-raitement des résultats du calcul. Les paragraphes qui suivent présenteront plus en détail chacune des classes citées ici.

6.2.2 La hiérarchie des AMR_Timesteppers

La hiérarchie des AMR_Timesteppers est représentée sur la figure 6.4. Les classes de

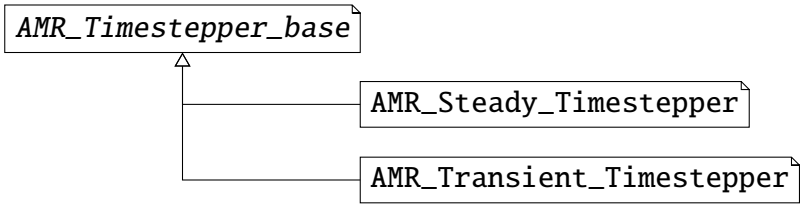


FIG. 6.4 : La hiérarchie des AMR_Timesteppers

cette hiérarchie ont pour responsabilité la gestion des itérations temporelles de la résolution du problème. Nous avons spécialisé la classe *AMR_Timestepper_base* pour pouvoir réaliser des simulations stationnaires avec l'AMR_Steady_Timestepper ainsi que des simulations instationnaires, en utilisant l'AMR_Transient_Timestepper. Ces deux classes diffèrent essentiellement par le fait que l'AMR_Steady_Timestepper ne génère qu'une seule itération en temps, ainsi que par des spécifications d'assemblage différentes (absence de prise en compte de la matrice de masse dans le cas stationnaire).

6.2.3 La hiérarchie des AMR_Fields

La hiérarchie des AMR_Fields est aussi simple que celles des AMR_Timesteppers, comme on peut le voir sur la figure 6.5. Ces classes correspondent à l'implémentation des

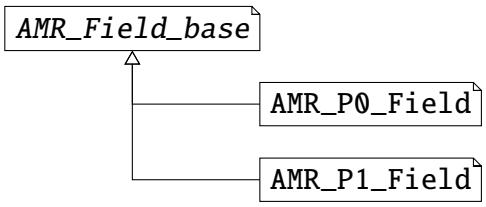


FIG. 6.5 : La hiérarchie des AMR_Fields

champs hiérarchiques, définis sur l'intégralité de la géométrie, et munis de l'opérateur d'interpolation présenté au paragraphe 4.3. Nous avons spécialisé l'implémentation de ces champs pour le cas de champs constants par morceaux (AMR_P0_Field), et pour le cas de champs affines par morceaux (AMR_P1_Field). Cette dernière classe permet d'interpoler autant les champs affines conformes (éléments finis $\mathbb{P}^1$ de LAGRANGE), que non–conformes (éléments finis $\mathbb{P}^1_{NC}$ de CROUZEIX–RAVIART). Notons qu'un *AMR_Field_base* est associé à un objet de type AMR_Refinement, décrit dans la section 6.3, qui fournit une description de l'état de raffinement des éléments du maillage, ainsi que leurs relations de filiation.

6.2.4 La hiérarchie des AMR_Error_Estimators

La hiérarchie des AMR_Error_Estimators est représentée sur la figure 6.6. Le rôle

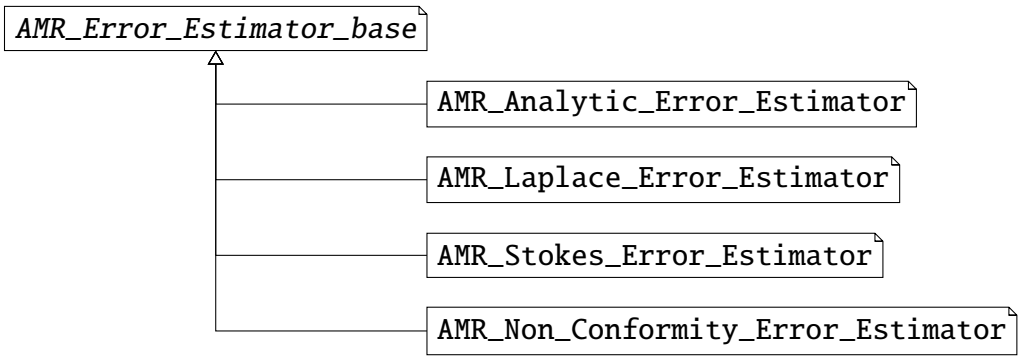


FIG. 6.6 : La hiérarchie des AMR_Error_Estimators

des classes de cette hiérarchie consiste à évaluer l'erreur numérique après la résolution du problème sur une géométrie donnée. Si l'on dispose de la solution analytique du problème étudié, la classe AMR_Analytic_Error_Estimator permet de calculer exactement l'erreur numérique en norme L^2. Dans le cas contraire, nous avons implémenté les estimateurs présentés au paragraphe 4.4 pour le problème de LAPLACE et le problème de STOKES. Enfin, la classe AMR_Non_Conformity_Error_Estimator fournit une estimation de l'erreur numérique relativement générique. Cette estimation repose sur l'idée suivante : l'espace discret engendré par l'élément fini $\mathbb{P}^1_{NC}$ de CROUZEIX–RAVIART contient l'espace discret engendré par l'élément fini $\mathbb{P}^1$ de LAGRANGE. Une différence essentielle entre ces deux espaces est que les fonctions du second sont continues, alors que les fonctions du premier ne le sont qu'en un sens faible. Ainsi, l'écart entre la solution calculée et sa projection sur l'espace des fonctions engendrées par l'élément fini $\mathbb{P}^1$ de LAGRANGE fournit une estimation de la « non–continuité » de la solution calculée. Si l'on sait que la solution du problème est continue, alors on conjecture que cette quantité permet d'estimer, au moins grossièrement, l'erreur numérique entre cette dernière et la solution calculée.

6.2.5 La hiérarchie des Local_Problems

Comme on peut le voir sur la figure 6.7, la hiérarchie des Local_Problems est strictement calquée sur celle des AMR_Problems. Un *Local_Probleme_base* représente l'abstraction d'un problème à résoudre sur un sous–domaine de la géométrie de calcul (voir le paragraphe 6.3). Ces classes permettent d'interfacer le module Framework avec le module Mortar, dont le rôle est de réaliser la résolution conjointe d'une liste d'instances de classes dérivant de *Local_Probleme_base* (voir le paragraphe 6.5). Bien entendu, un AMR_Laplace_Problem (resp. AMR_Stokes_Problem instanciera des problèmes locaux de type Local_Laplace_Problem (resp. Local_Stokes_Problem).

La figure 6.8 représente les principaux membres des classes *Local_Problem_base* et *Local_Constrained_Problem_base*. Nous voyons que la classe *Local_Problem_base* agrège un membre de type AMR_Subdomain, qui fournit la description de la géométrie associée à ce problème local. La classe *Local_Problem_base* possède également un

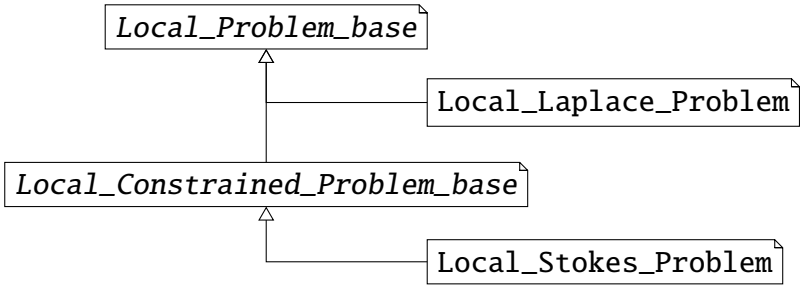


FIG. 6.7 : La hiérarchie des Local_Problems

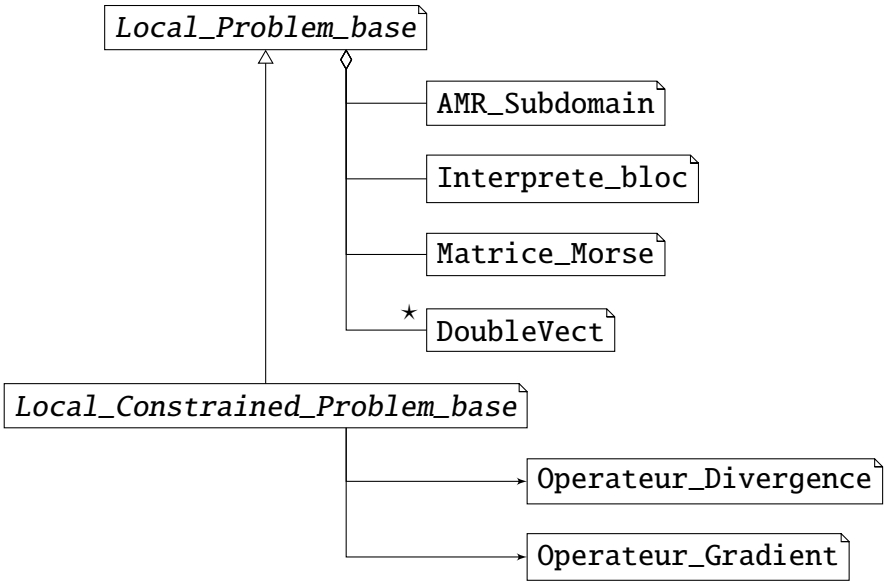


FIG. 6.8 : Principaux membres des classes de bases de la hiérarchie des Local_Problems

Interprete_bloc. Cet objet est particulièrement important, car il permet de construire une instance locale d'un problème en interprétant un jeu de données *ad hoc*. Une fois ce problème construit, il est possible de récupérer dans les autres membres de la classe la matrice (objet de type Matrice_Morse), ainsi que la solution et le second membre (objets de type DoubleVect) du problème local.

Enfin, un *Local_Constrained_Problem_base* est associée à deux objets particuliers, à savoir un Operateur_Divergence et un Operateur_Gradient. Ces classes génériques, présentes dans Trio-U, permettent de référencer les opérateurs discrets correspondants, générés par la discrétisation utilisée lors de l'interprétation du jeu de données. Ces références correspondent aux contraintes imposées au problème de type point–selle (p. ex. nullité de la divergence de la vitesse), ainsi qu'à leurs transposées (p. ex. calcul du gradient de pression).

6.2.6 Algorithme global de résolution

La classe *AMR_Problem_base* possède trois méthodes essentielles, appelées successivement lors de la résolution du problème. La première, initialize(), correspond à

l'initialisation du calcul (allocation mémoire, construction des structures de données géométriques, etc). La deuxième, `solve()`, lance la résolution du problème. Enfin, la troisième méthode, `finalize()`, libère les ressources éventuellement allouées au cours du calcul. Nous allons présenter maintenant l'algorithme implémenté lors de l'appel à la méthode `solve()` de cette classe.

Cet algorithme est illustré sur la figure 6.9. Ainsi, notre algorithme consiste à itérer

```
while ( timestepper has not finished ) {
  initialize time iteration
  while ( geometry has not converged ) {
    initialize space iteration
    iterate
    finalize space iteration
  }
  finalize time iteration
}
```

FIG. 6.9 : Structure globale de l'algorithme de résolution de notre méthode adaptative

sur les pas de temps nécessaires à la résolution du problème, spécifiés par l'objet héritant de la classe *`AMR_Timestepper_base`*. Comme nous l'avons souligné, c'est la classe réelle de l'objet désigné par le pointeur présent dans la classe *`AMR_Problem_base`* qui déterminera la nature de ces itérations en temps.

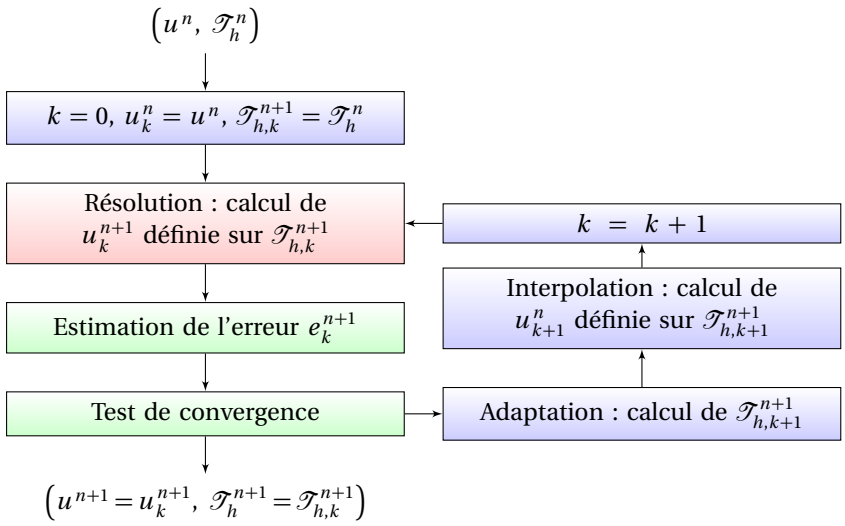


FIG. 6.10 : Intéractions des itérations en temps et de la boucle adaptative

La figure 6.10 présente en détail le travail effectué au cours d'une itération en temps. Dans cette figure, u_j^i et $\mathcal{T}_{h,j}^i$ désignent respectivement le j^{e} itéré de la solution et du maillage, lors de la résolution du i^{e} pas de temps. Nous avons colorié en bleu, rouge et vert les étapes réalisées dans la méthode `initialize_space_iteration`, `iterate` et `finalize_space_iteration`, respectivement. Le principe retenu consiste à effectuer

plusieurs itérations sur le maillage, de façon à déterminer une solution et un maillage au temps t^{n+1} qui vérifient ensemble un critère de précision. Ce critère est implémenté dans le test de convergence sur la figure 6.10, et assure que l'erreur numérique de la solution au temps t^{n+1} calculée sur le maillage au temps t^{n+1} est inférieure à un seuil donné. La phase de résolution est pilotée par l'objet *Mortar_Problem_base*, nous verrons comment dans le paragraphe 6.5. De même, la phase d'estimation de l'erreur est déléguée à l'objet *AMR_Error_Estimator_base*, tandis que la phase d'adaptation est effectuée par la classe AMR_Geometry et que la phase d'interpolation est réalisée par les *AMR_Field_base* eux–mêmes.

Quelque soit le choix retenu pour le schéma en temps, les itérations font systématiquement intervenir une phase d'adaptation géométrique et une phase de résolution. Cette dernière, afin de s'avérer efficace dans un contexte parallèle, peut également être divisée en deux phases : une phase de distribution des sous–problèmes, et une phase de résolution numérique proprement dite. Nous présentons dans les paragraphes qui suivent l'implémentation que nous proposons pour chacune de ces étapes du calcul.

6.3 Le sous–module Geometry : adaptation géométrique

6.3.1 La classe AMR_Geometry

Comme nous l'avons souligné auparavant, la classe AMR_Geometry constitue le point central du module Geometry. La figure 6.11 en présente les différents membres. Nous

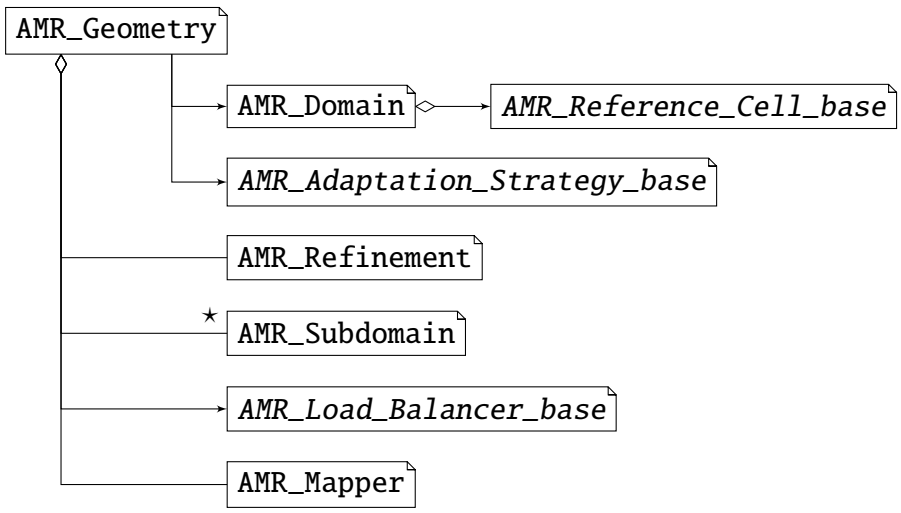


FIG. 6.11 : Principaux membres de la classe AMR_Geometry

voyons que cette classe est associée à un objet de type AMR_Domain. Cet objet correspond à la description de la partie du maillage initial affectée au processeur courant, et il comprend un élément de référence, une liste de sommets, une liste d'éléments, ainsi qu'une liste de face de bords. L'élément de référence correspond au membre indiqué dans la figure 6.11, à savoir la référence à une *AMR_Reference_Cell_base*. Nous décrirons cette classe et sa hiérarchie dans le paragraphe 6.3.2. Les sommets sont décrits par la donnée de leurs coordonnées. Les éléments sont décrits par l'énumération des indices

de leurs sommets. Les faces de bords sont décrites par l'indice de l'élément du sous–domaine qui leur est adjacent, et l'indice local de la face dans cet élément à laquelle elles correspondent (voir le paragraphe 6.3.2 pour la description de la numérotation locale des faces des éléments). Par convention, les faces de bords sont triées selon le bord auquel elles appartiennent. Notons l'existence de bords particuliers, correspondant aux interfaces entre sous–domaines. Les faces de ces bords présentent une propriété supplémentaire, à savoir la donnée de l'indice du sous–domaine voisin, ainsi que celle de la face correspondante dans ce sous–domaine. Les objets de type `AMR_Domain` présentent ceci de particulier qu'une fois créés, ils ne peuvent plus être modifiés.

La classe `AMR_Geometry` est aussi associée à une *`AMR_Adaptation_Strategy_base`*. Cet objet joue un rôle essentiel dans le déroulement de l'algorithme adaptatif, car il a la responsabilité de déterminer, à partir de l'estimation de l'erreur, quels sont éléments du maillage qui nécessitent d'être raffinés, ainsi que ceux qui pourraient être déraffinés. Nous détaillerons la hiérarchie associée à cette classe dans le paragraphe 6.3.3.

Le membre de type `AMR_Refinement` décrit l'état de raffinement de la géometrie. Les membres de type `AMR_Subdomain` portent la représentation des sous–domaines qui seront utilisés lors de la phase de calcul. Nous détaillerons le fonctionnement de ces classes dans le paragraphe 6.3.4. La référence à un *`AMR_Load_Balancer_base`* sert essentiellement à déterminer une bonne distribution des sous–domaines de calcul sur les processeurs de façon à équilibrer la charge lors de la résolution. Le membre de type `AMR_Mapper` facilite la distribution effective des sous–domaine en implémentant le schéma de communications associé. Ces deux dernières classes seront présentées plus en détail dans le paragraphe 6.4.

6.3.2 La hiérarchie des `AMR_Reference_Cells`

Pour l'heure, notre méthode adaptative n'a été implémentée que pour la méthode VEF de Trio-U. Ainsi, nos maillages seront toujours formés de simplexes (triangles en dimension 2, tétraèdres en dimension 3). Cependant, il est tout–à–fait envisageable de tenter de l'étendre au cas de maillage formés de quadrangles en dimension 2 ou d'hexaèdres en dimension 3. Pour cette raison, à la lecture du domaine de calcul dans le jeu de données, et non à la lecture de la dimension du problème, l'`AMR_Domain` crée un objet statique dont le type est déterminé par l'élément géométrique de référence du maillage. Cet objet appartient à la hiérarchie de classes présentée sur la figure 6.12.

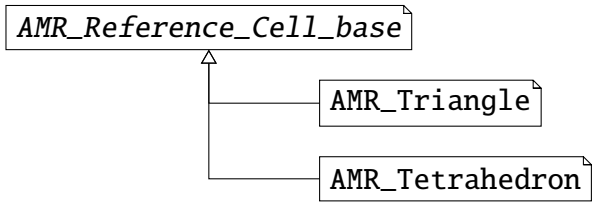
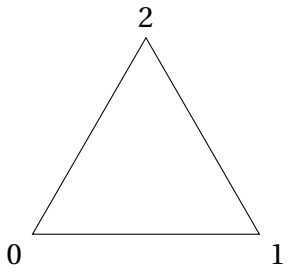


FIG. 6.12 : La hiérarchie des `AMR_Reference_Cells`

Le rôle des objets de cette hiérarchie est particulièrement important. Tout d'abord, ce sont ces objets qui fournissent les informations essentielles de numérotation locale des sommets, arêtes, et faces des éléments du maillage dont nous avons vu plus haut la

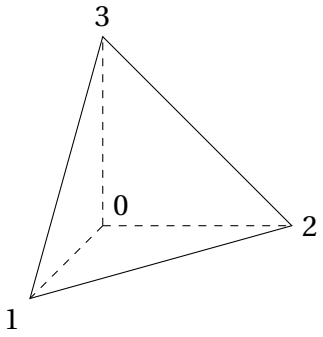
nécessité pour décrire les frontières du domaine. Ces conventions sont présentées sur la figure 6.13. Ensuite, ils implémentent les méthodes de raffinement du maillage. Plus



sommets des arêtes		
1re arête	0	1
2e arête	0	2
3e arête	1	2

sommets des faces		
1re face	1	2
2e face	2	0
3e face	0	1

(a) Élément triangulaire



sommets des arêtes		
1re arête	0	1
2e arête	0	2
3e arête	0	3
4e arête	1	2
5e arête	1	3
6e arête	2	3

sommets des faces			
1re face	1	2	3
2e face	2	0	3
3e face	3	0	1
4e face	0	2	1

(b) Élément tétraèdrique

FIG. 6.13 : Conventions de numérotation locale des éléments géométriques

exactement, ils disposent d'une méthode `refine()`, qui permet de raffiner en ensemble d'éléments donné. Pour cela, elle admet comme arguments un tableau de sommets, un tableau d'éléments et un tableau d'incidence. Le tableau de sommets contient les coordonnées des sommets des éléments à raffiner. Le tableau d'éléments contient l'énumération des indices des sommets de chaque élément. Le tableau d'incidence contient l'énumération des éléments incidents à chaque élément. Au sortir de cette méthode, ces trois tableaux ont été redimensionnés, et ils ont été modifiés de la façon suivante :

- le tableau de sommets s'est vu rajouter les sommets correspondants aux milieux des arêtes du maillage de départ ;
- le tableau d'éléments contient les éléments raffinés générés par la subdivision des éléments initiaux ;
- le tableau d'incidence contient l'incidence des éléments raffinés.

On voit ici l'importance particulière de la remarque qui a été faite au paragraphe 3.2.3 concernant la numérotation des éléments raffinés. En effet, la cohérence des choix retenus pour les conventions de numérotations avec la proposition 3.3 définissant la numérotation des sommets des éléments raffinés permet de faciliter grandement l'implémen-

tation de cet algorithme de raffinement en s'affranchissant d'éventuels cas particuliers, tout en assurant une complexité optimale pour cet algorithme de raffinement.

6.3.3 La hiérarchie des AMR_Adaptation_Strategies

Nous avons vu qu'une AMR_Geometry était associée à un objet héritant de la classe *AMR_Adaptation_Strategy_base*. Comme leur nom l'indique, le rôle des classes de cette hiérarchie est de définir la stratégie de raffinement. Autrement dit, les classes spécialisant la classe de base *AMR_Adaptation_Strategy_base* ont pour responsabilité de choisir les éléments à raffiner et à déraffiner, à partir de la donnée de l'estimation d'erreur. Cette hiérarchie est présentée dans la figure 6.14.

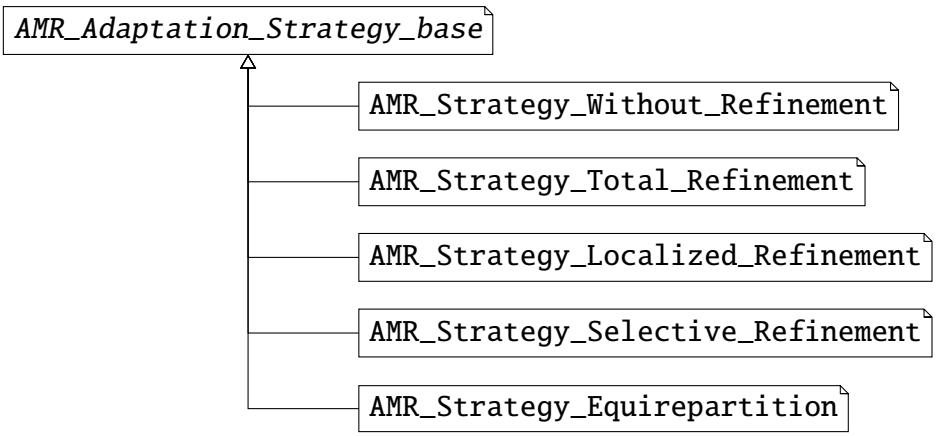


FIG. 6.14 : La hiérarchie des AMR_Adaptation_Strategies

Comme nous pouvons le voir, plusieurs de ces stratégies ont été implémentées. La classe AMR_Strategy_Without_Refinement implémente une approche sans raffinement. Cette classe est particulièrement utile à des fins de déverminage de la phase de résolution, car elle permet de s'affranchir de toute adaptation de maillage.

La classe AMR_Strategy_Total_Refinement implémente une stratégie de raffinement on ne peut plus simple : elle choisit systématiquement de raffiner l'ensemble des éléments du maillage. Encore une fois, cette approche est utile pour valider le comportement du code : elle permet de vérifier le nombre de classe de congruence des éléments raffinés après plusieurs étapes de raffinement.

La classe AMR_Strategy_Localized_Refinement permet de sélectionner certains éléments pour le raffinement, et de conserver les autres éléments. En aucun cas elle ne spécifiera de déraffinement. La classe AMR_Strategy_Selective_Refinement fonctionne de la même façon. Elles se différencient essentiellement au niveaux du choix des éléments à raffiner : la première suppose que ces éléments soient indiqués explicitement par l'utilisateur, alors que la seconde utilise l'estimation d'erreur. Encore une fois, ces classes sont particulièrement utiles pour valider la phase de raffinement des éléments lors de l'adaptation de maillage.

Enfin, la classe AMR_Strategy_Equirepartition permet de définir une véritable démarche adaptative, en autorisant la spécification d'éléments à raffiner, conserver ou déraffiner. L'objectif de cette stratégie consiste à tenter d'uniformiser l'erreur sur les élé-

ments du maillage. Pour cela, elle sélectionne les éléments présentant une erreur faible pour les déraffiner, et ceux présentant une erreur forte pour les raffiner.

6.3.4 Algorithme d'adaptation de la géométrie

Nous présentons maintenant l'implémentation de notre algorithme d'adaptation de maillage. Cet algorithme repose sur l'idée qu'il est possible de déterminer la géométrie complète (comprenant les éléments raffinés et les éléments non–raffinés) à partir de la donnée du maillage initial (fourni par la classe `AMR_Domain`) et de l'état de raffinement (donné par la classe `AMR_Refinement`). Cette idée permet de séparer la mise à jour de la géométrie en deux étapes distinctes :

- dans un premier temps, on procède à la mise à jour de la classe `AMR_Refinement` en partant des spécifications de raffinement fournies par les classes de la hiérarchie des `AMR_Adaptation_Strategies` ;
- dans un second temps, on utilise ce nouvel état de raffinement et les données de la classe `AMR_Domain` pour déterminer la nouvelle géométrie.

Notons au passage que ce nouvel état de raffinement est aussi utilisé pour mettre à jour la structure des objets héritant de *`AMR_Field_base`* présents dans la classe *`AMR_Problem`*.

La classe `AMR_Refinement` encode l'information de l'état de raffinement du maillage en utilisant deux tableaux :

- un tableau nommé `offsets` de taille `n+1`, où `n` est le nombre de niveaux ;
- un tableau nommé `key` de taille `N`, où `N = offsets[n]` est le nombre total d'éléments dans la géométrie (en comptant les éléments raffinés).

Le tableau `offsets` est rempli de sorte que `(offset[i+1] - offset[i])` soit égal au nombre d'éléments dans le $\texttt{i}^{\text{e}}$ niveau. Le tableau `key` est rempli de la façon suivante. Notons `k` la valeur donnée par `key[offsets[i]+j]`, où `i` (resp. `j`) est compris entre `0` et `n` (resp. entre `0` et `offsets[i+1]-offsets[i]`). Si `k` est strictement négatif, alors le $\texttt{j}^{\text{e}}$ élément du $\texttt{i}^{\text{e}}$ niveau n'est pas raffiné. Si `k` est positif ou nul, alors les fils du $\texttt{j}^{\text{e}}$ élément du $\texttt{i}^{\text{e}}$ niveau sont les éléments d'indices `offset[i+1]+p*k+q`, où `p` vaut 4 en dimension 2, et 8 en dimension 3, et où `q` varie entre 0 et `p`. Cette structure nous permet de numéroter de façon globale les éléments de la géométrie hiérarchique, et de connaître les relations de filiation entre les éléments des différents niveaux.

Pour mettre à jour cette structure, nous utilisons l'algorithme détaillé dans l'encadré 3.13. La stratégie de raffinement dont nous venons de parler fournit un drapeau valant `-1`, `0` ou `1` pour chaque élément non–raffiné, indiquant respectivement si l'élément auquel il se réfère doit être déraffiné, conservé ou raffiné. Il s'agit des données d'entrées de notre algorithme d'adaptation. La première chose à faire consiste à étendre la définition de ce drapeau à tous les éléments de la géométrie hiérarchique, en affectant un drapeau 1 à tous les éléments raffinés. L'application de l'algorithme 3.13 nous permet de calculer les tableaux `offsets` et `key` du nouvel `AMR_Refinement`, ce qui est fait mettant à jour dynamiquement les marques de raffinement. En effet, si un élément non–raffiné est marqué pour le raffinement, les marques de raffinement de ses éléments fils doivent être insérées dans les drapeaux du niveau suivant. De même, un élément raffiné marqué pour le déraffinement conduit à la suppression des marques de raffinement de ses éléments fils dans les drapeaux du niveau suivant. Cet algorithme relativement compliqué à aborder est considérablement simplifié par l'utilisation de tableaux de drapeaux temporaires, et l'introduction de la classe `AMR_Cell_Childs_Iterator`, qui permet d'itérer

sur les éléments fils d'un élément raffiné.

Une fois que l'`AMR_Refinement` est à jour, nous calculons la nouvelle géométrie en procédant comme suit. Partant de la géométrie initiale fournie par l'`AMR_Domain`, nous itérons sur les niveaux de raffinement. À chaque niveau, nous sélectionnons les éléments marqués comme étant raffinés dans l'`AMR_Refinement`, et nous appelons la méthode `refine()` de l'*`AMR_Reference_Cell_base`* pour raffiner cette partie du maillage, ce qui nous fournit les éléments du niveau suivant. Nous procédons de même pour raffiner la représentation des bords extérieurs et des interfaces entre domaines affectés à des processeurs différents. Nous obtenons alors une description complète de la géométrie, comprenant à la fois les éléments raffinés et les éléments non–raffinés.

L'étape suivante, qui est également la dernière du processus d'adaptation de la géométrie, consiste alors à regrouper les éléments non–raffinés d'un même niveau en sous–domaines, ce qui est réalisé par l'*`AMR_Load_Balancer_base`*. Nous traiterons en détail cette question au paragraphe suivant, car il s'agit d'une étape déterminante pour assurer l'équilibrage de la charge de calcul lors de la phase de résolution. Cependant, soulignons qu'au sortir de ce travail, nous avons reconstruit la liste des `AMR_Subproblem` que possède l'`AMR_Geometry`. Ces objets seront par la suite utilisés pour construire les problèmes locaux composant le problème à résoudre.

6.4 Le sous–module `Geometry` : équilibrage de la charge

Avant d'aborder la phase de résolution proprement dite, nous présentons d'abord les structures de données et les algorithmes mis en place pour assurer l'équilibrage de la charge. Nous suivons ainsi l'algorithme global, car ce dernier, une fois la géométrie hiérarchique établie, construit et distribue les sous–problèmes avant de lancer la résolution.

6.4.1 La hiérarchie des `AMR_Load_Balancers`

La classe `AMR_Geometry` agrège un *`AMR_Load_Balancer_base`*. Les classes qui héritent de cette dernière sont précisément celles qui permettent de définir une politique d'équilibrage de la charge pour la phase de résolution, en spécifiant l'affectation des problèmes locaux aux différents processeurs. La figure 6.15 représente la hiérarchie de classes associée à ce concept. Encore une fois, plusieurs algorithmes d'équilibrage de la

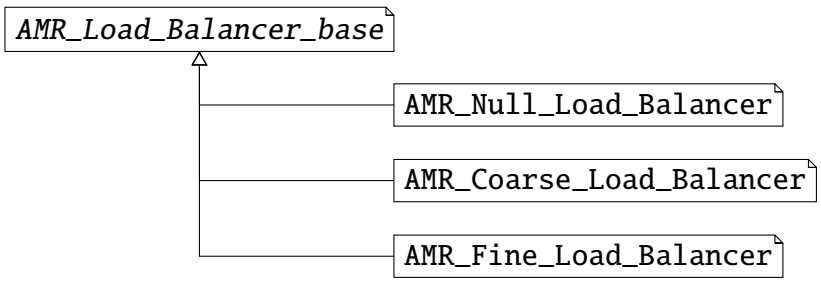


FIG. 6.15 : La hiérarchie des `AMR_Load_Balancers`

charge sont proposés. Ainsi, la classe `AMR_Null_Load_Balancer` implémente une approche sans équilibrage : on ne déplace pas les sous–domaines d'un processeur à l'autre,

et ils restent affectés au processeur dont la partition recouvre le sous–domaine qui leur est associé. Cette classe est particulièrement utile à des fins de déverminage de la phase de résolution, car elle permet de s'affranchir des difficultés liées à la mise en place du schéma de communication associé à la distribution des sous–problèmes.

La classe AMR_Coarse_Load_Balancer implémente une méthode d'équilibrage de la charge relativement grossière. Le principe de cette méthode consiste à construire un graphe contenant un nœud par sous–domaine, et une arête entre deux nœuds si les sous–domaines associés sont adjacents. On peut pondérer ce graphe de façon à modéliser le temps de calcul sur chaque sous–domaine, en associant à chaque nœud le nombre de degrés de libertés contenu par le sous–domaine associé. De la même façon, il est possible de pondérer les arêtes du graphe de façon à prendre en compte les communications inter–processeurs. Le poids associé aux arêtes du graphe sera alors égal au nombre de faces présentes sur l'interface en question. On détermine alors la distribution des sous–problèmes sur les processeurs en utilisant une heuristique de partitionnement de graphe. Pratiquement, le graphe ainsi construit est relativement petit : il contient assez peu de nœuds. Pour cette raison, nous avons choisi d'utiliser un algorithme de partitionnement séquentiel, exécuté par le processeur maître. Cet algorithme est fourni par la librairie METIS.

La classe AMR_Fine_Load_Balancer repose également sur une heuristique de partitionnement de graphe. Cependant, au lieu du graphe des sous–domaines, nous partitionnons cette fois le graphe d'adjacence des éléments non–raffinés du maillage. On assure ainsi la qualité de l'équilibrage, car il s'agit du grain le plus fin atteignable avec notre modèle d'exécution. Comme le graphe associé à la relation d'adjacence est assez important, nous optons pour la version parallèle d'une heuristique de partitionnement, fournie par la librairie PARMETIS. Nous avons dû pour cela encapsuler une interface vers la librairie PARMETIS, car cette dernière ne fait pas partie des dépendances classiques de la plate–forme Trio-U.

6.4.2 Découpage de la géométrie

La manière de découper la géométrie en sous–domaines dépend essentiellement du choix retenu pour la politique d'équilibrage de la charge. Le principe général consiste à attribuer deux entiers à chaque élément non–raffiné de la géométrie : le premier correspond à un numéro global d'identification du sous–domaine aplati auquel il appartient, le second est le numéro du processeur qui va résoudre le sous–problème associé.

L'objectif de l'*AMR_Load_Balancer_base* est de définir cette affectation des éléments sur les processeurs. Nous avons vu que, dans le cas de l'AMR_Null_Load_Balancer, la valeur retournée est le numéro du processeur qui possède la partition contenant l'élément. Dans le cas de l'AMR_Coarse_Load_Balancer, la valeur retournée est quelconque, comprise entre 0 et le nombre de processeurs. Cette valeur est cependant identique pour tous les éléments non–raffinés d'un même niveau de raffinement placés sur le même processeur. Enfin, dans le cas de l'AMR_Fine_Load_Balancer, la valeur retournée est réellement quelconque, entre 0 et le nombre de processeurs.

Ainsi, dans le cas de l'AMR_Null_Load_Balancer, nous n'avons qu'à générer une numérotation globale des sous–domaines, ce qui peut être fait par simple comptage. Il en est de même pour le cas de l'AMR_Coarse_Load_Balancer. Par contre, nous devons continuer le découpage des sous–domaines dans le cas de l'AMR_Fine_Load_Balancer. Il s'agit, pour chaque processeur, de regrouper ensemble les éléments non–raffinés qui

présentent le même niveau de raffinement et qui ont été affectés au même processeur par le partitionnement du graphe. Pour cela, nous parcourons le graphe de connectivité des éléments non–raffinés, et nous les distribuons dans les sous–domaines en fonction de leur raffinement et de leur affectation sur les processeurs. Un algorithme de comptage permet alors de définir un indice global pour chaque sous–domaine. Muni de ces informations, nous pouvons construire les objets `AMR_Subdomain` agrégés par la classe `AMR_Geometry`.

6.4.3 La classe `AMR_Mapper` et le placement des sous–problèmes

La classe `AMR_Mapper` encapsule la notion de répartition des sous–problèmes sur les processeurs, ainsi que les objets nécessaires à l'établissement des communications lors de la phase de distribution. Comme on peut le voir sur le diagramme UML 6.16, cette classe porte deux listes d'objets de type `Local_Problem_Proxy`. Les objets de cette

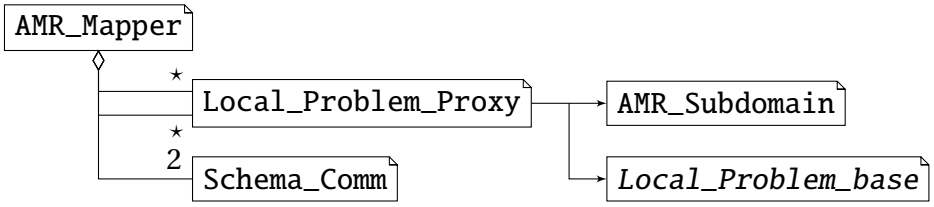


FIG. 6.16 : Principaux membres de la classe `AMR_Mapper`

classe représentent un problème local, défini par un identifiant global, un sous–domaine et les champs associés, un jeu de données, et deux numéros de processeurs. Le premier de ces numéros désigne le processeur possédant la partie de la géométrie à laquelle appartient le sous–domaine. C'est ce processeur qui est capable de construire le `Local_Problem_Proxy` à partir des données géométriques. Le second correspond au processeur qui effectuera la résolution du sous–problème. Nous avons choisi d'implémenter la classe `Local_Problem_Proxy` à partir du patron de conception appelé Mandataire [65], de façon à limiter l'empreinte mémoire de ces objets. De cette façon, un `Local_Problem_Proxy` possède un pointeur vers un objet de type `AMR_Subdomain`, et non un objet de type `AMR_Subdomain` en tant que tel.

La présence des deux listes d'objets de type `Local_Problem_Proxy` dans la classe `AMR_Mapper` est impérative. La première liste est construite à partir des données géométriques, et correspond à la description des sous–problèmes posés sur des sous–domaines inclus dans la partition affectée au processeur considéré. La seconde liste énumère les sous–problèmes que le processeur va devoir traiter dans la phase de résolution. Cette liste est construire à partir de la première, via un échange de message entre processeurs. Cette communication est prise en charge à l'aide des objets de type `Schema_Comm` qu'agrège la classe `AMR_Mapper`. Le premier de ces objets correspond à la communication nécessaire à la phase de distribution des sous–problèmes. Le second correspond au schéma de communication inverse, et il est utilisé lors de la phase de rappatriement des champs à la fin de la résolution. Le principe de la classe `Schema_Comm` est de fournir une interface simplifiée de gestion des communication synchrones, permettant d'assurer la symétrie des échanges.

6.5 Le sous–module Mortar

Le sous–module Mortar implémente la méthode d'éléments joints décrite au chapitre 4 pour résoudre le problème défini sur la géométrie adaptée. Il s'agit essentiellement de résoudre les systèmes linéaires établis aux paragraphes 4.2.4 pour le problème de LAPLACE et 4.2.5 pour le problème de STOKES. Nous rappelons ici la forme de ce système pour le problème de LAPLACE :

$$\begin{bmatrix} A_{II}^1 & & & A_{I\Gamma}^{1T} C_\Gamma^{1T} \\ & \ddots & & \vdots \\ & & A_{II}^N & A_{I\Gamma}^{NT} C_\Gamma^{NT} \\ C_\Gamma^1 A_{I\Gamma}^1 & \dots & C_\Gamma^N A_{I\Gamma}^N & E_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} U_I^1 \\ \vdots \\ U_I^N \\ U_\Gamma \end{bmatrix} = \begin{bmatrix} F_I^1 \\ \vdots \\ F_I^N \\ F_\Gamma \end{bmatrix}$$

où l'on a posé :

$$E_{\Gamma\Gamma} = \sum_{k=1}^{N} C_\Gamma^k A_{\Gamma\Gamma}^k C_\Gamma^{kT}$$

$$F_\Gamma = \sum_{k=1}^{N} C_\Gamma^k F_\Gamma^k$$

Dans ces équations, les éléments de la matrice A_{II}^i définissent le couplage entre deux inconnues internes à un même sous–domaine. Les éléments de la matrice $A_{I\Gamma}^i$ définissent le couplage entre une inconnue interne à un sous–domaine, et une inconnue définie sur la discrétisation d'une interface associée à ce sous–domaine. Remarquons qu'il s'agit de la matrice obtenue lorsque l'on impose sur ce bord des conditions aux limites de NEUMANN homogènes. Quant aux matrices C_Γ^i, elles définissent le couplage entre l'inconnue interfaciale définie sur la discrétisation maître des interface, et les inconnues de bords, définies de part et d'autre de l'interface, à l'aide des discrétisations locales.

L'implémentation du module Mortar repose donc sur trois ensembles de classes :
- les classes réalisant l'abstraction du problème à résoudre ;
- les classes implémentant les algorithmes de résolution ;
- les classes représentant le système linéaire associé au problème.

Chacun de ces ensembles de classes sera présenté dans les paragraphes qui suivent.

6.5.1 La hiérarchie des Mortar_Problems

La hiérarchie des Mortar_Problems est relativement simple, comme on peut le voir sur la figure 6.17. La classe Mortar_Problem représente un problème à résoudre, composé d'une liste de problemes locaux donnés par les *Local_Problem_base* auxquels il est associé. Le Mortar_Constrained_Problem spécialise la classe Mortar_Problem pour pouvoir traiter convenablement le cas d'un problème de recherche de point–selle. Pour cela, il est associé à une collection de *Local_Constrained_Problem_base*. Les problèmes locaux référencés par les Mortar_Problem sont en réalité créés par les classes dérivées de *AMR_Problem_base*, à la suite de leur distribution sur les processeurs par la classe AMR_Mapper.

Rappelons que la classe *Local_Problem_base* agrège une Matrice_Morse et plusieurs DoubleVect pour représenter le système linéaire associé au problème local (voir

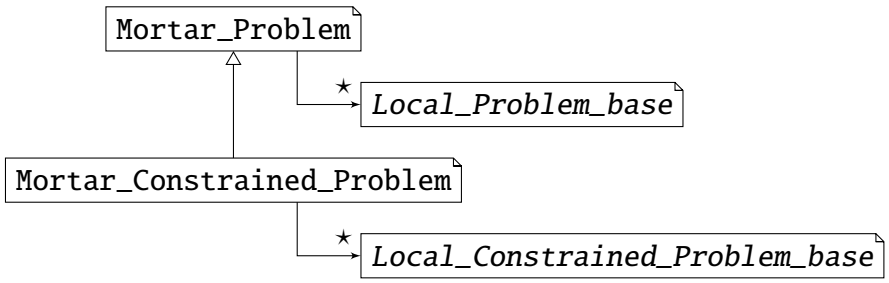


FIG. 6.17 : La hiérarchie des `Mortar_Problems`

la figure 6.8). Cette matrice, une fois assemblée, contient la matrice élémentaire suivante (en adoptant les notations utilisées plus haut) :

$$\begin{bmatrix} A^i_{II} & A^{iT}_{I\Gamma} \\ A^i_{I\Gamma} & A^i_{\Gamma\Gamma} \end{bmatrix}$$

De la même façon, les membres `Operateur_Divergence` et `Operateur_Gradient` de la classe *`Local_Constrained_Problem_base`* désignent les opérateurs divergence et gradient définis sur le sous–domaine associé au problème local considéré. Ainsi, les problèmes locaux représentent une contribution à l'assemblage du système global.

6.5.2 La hiérarchie des `Mortar_Solvers`

La hiérarchie des `Mortar_Solvers` est représentée sur la figure 6.18. Nous avons éga-

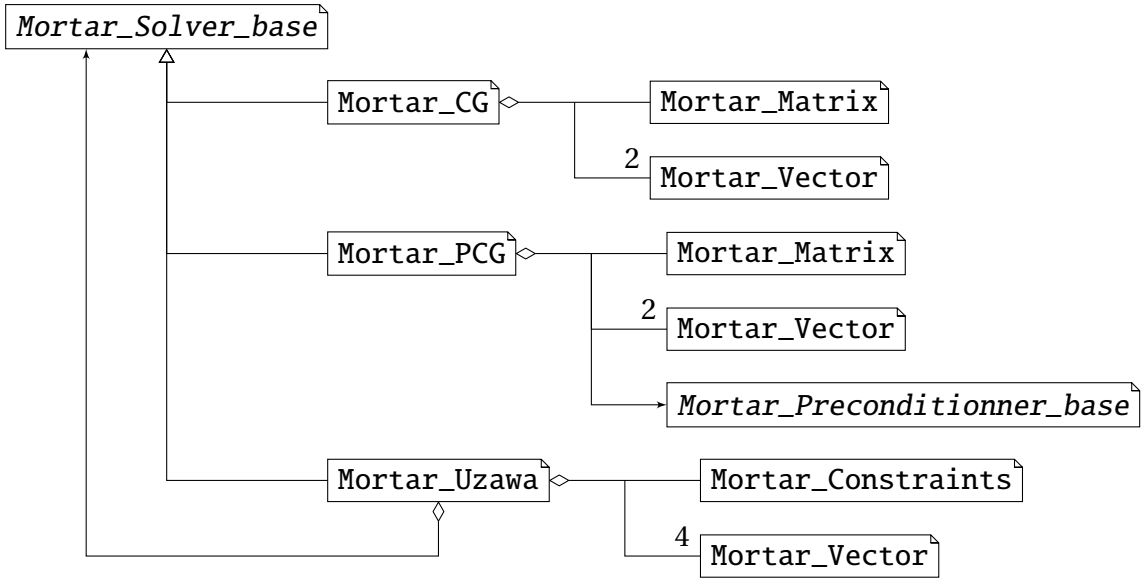


FIG. 6.18 : La hiérarchie des `Mortar_Solvers`

lement fait apparaitre sur cette figure les principaux membres des classes concernées. Les classes `Mortar_CG`, `Mortar_PCG` et `Mortar_Uzawa` implémentent respectivement un algorithme de gradient conjugué, un algorithme de gradient conjugué préconditionné, et

un algorithme de type UZAWA. Nous voyons ici l'intérêt de séparer l'implémentation des problèmes de type minimisation de celle des problèmes de type recherche de point–selle, car les algorithmes de résolution utilisés sont de nature différente. Notons que l'algorithme d'UZAWA implémenté pour résoudre le problème de STOKES est une sorte de généralisation de l'algorithme d'UZAWA classique. En effet, ce dernier peut être perçu comme l'application d'un algorithme de plus profonde descente au système linéaire associé aux inconnues de pression. Afin d'accélérer la convergence, nous avons implémenté notre algorithme d'UZAWA de sorte qu'il corresponde à l'application d'un algorithme de gradient conjugué sur le système en pression.

La figure 6.19 présente la hiérachie des préconditionneurs que nous avons développés pour la classe Mortar_PCG. Le premier est un préconditionneur diagonal classique,

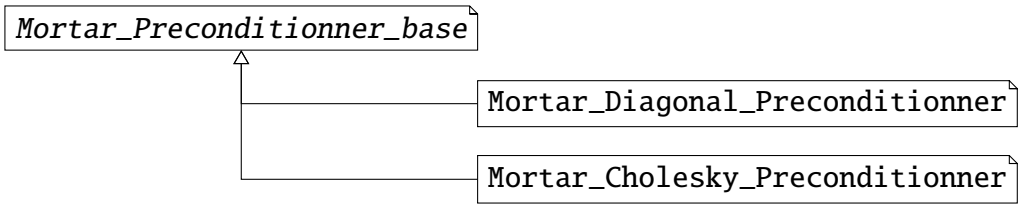


FIG. 6.19 : La hiérarchie des Mortar_Preconditionners

et le second correspond à une résolution directe des problèmes de LAPLACE locaux obtenus en imposant la valeur de l'inconnue interfaciale courante comme condition de DIRICHLET sur les bords correspondant aux interfaces entre sous–domaines. Cette résolution est effectuée à l'aide d'une décomposition de CHOLESKY de la matrice locale associée à ce problème. Ces deux préconditionneurs présentent la propriété non–négligeable d'être parallèles, en ce sens que l'on obtient strictement les mêmes itérés lorsque l'on compare une résolution séquentielle et une résolution parallèle.

6.5.3 Représentation du système linéaire

Nous présentons maintenant les classes Mortar_Matrix, Mortar_Constraints et Mortar_Vector dont les solveurs ont besoin. Ces classes modélisent les systèmes linéaires associés aux problèmes étudiés tels qu'ils ont été présentés aux paragraphes 4.2.4 et 4.2.5.

La classe Mortar_Matrix consiste en réalité en une collection de références vers les objets de type Matrice_Morse agrégés par les *Local_Problem_base*. De la même façon, la classe Mortar_Constraints permet d'accéder aux Operateur_Divergence et Operateur_Gradient de chacun des *Local_Constrained_Problem_base*. Nous verrons par la suite comment ces classes sont utilisées. Auparavant, nous devons détailler un peu plus la classe Mortar_Vector. La figure 6.20 en présente les différents membres. La classe Mortar_Vector peut être perçue comme la concaténation des vecteurs (inconnues et termes sources) associés aux problèmes locaux. Le Mortar_Allocator permet d'ailleurs d'identifier dans le tableau global (attribut de type ArrOfDouble) les indices correspondant aux valeurs associées à chacun des problèmes locaux. Cependant, il faut également assurer que les valeurs locales restent cohérentes au niveau des interfaces entre les sous–domaines. Pour cela, la classe Mortar_Exchanger permet d'échanger entre les sous–domaines les valeurs interfaciales, en fournissant des objets de type

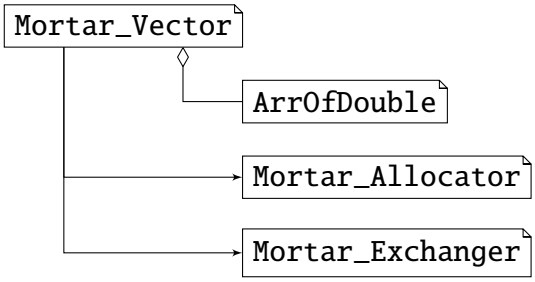


FIG. 6.20 : Principaux membres de la classe `Mortar_Vector`

`Schema_Comm` adaptés. En ce sens, la classe `Mortar_Vector` généralise la notion d'*items communs* [95] des vecteurs distribués statiques déjà présents dans Trio-U. À la différence de ces derniers, les schémas de communication associés aux `Mortar_Vector` sont dynamiques, car ils sont réinitialisés à chaque phase de calcul. De plus, le `Mortar_Exchanger` utilise les informations de connectivité au niveau des interfaces entre sous–domaines pour pouvoir appliquer les matrices de couplage C_Γ^i et C_Γ^{iT}.

Nous allons maintenant illustrer l'intérêt de cette implémentation en explicitant les calculs effectués lors d'un produit matrice–vecteur. Cet exemple est particulièrement important, car les solveurs présentés au paragraphe précédent reposent essentiellement sur ce calcul. Tout d'abord, nous transformons le vecteur global $\left[U_I^1,\ \ldots,\ U_I^N,\ U_\Gamma\right]^T$ en N vecteurs locaux définis par $\left[U_I^i,\ C_\Gamma^{iT}U_\Gamma\right]^T$ pour i variant de 1 à N. Ensuite, nous multiplions ces vecteurs locaux par les matrices locales données par la `Mortar_Matrix`. Nous notons les résultats de ces produits locaux $\left[V_I^i,\ V_\Gamma^i\right]^T$. Enfin, nous assemblons les résultats en un nouveau vecteur global : $\left[V_I^1,\ \ldots,\ V_I^N,\ \sum_{i=1}^N C_\Gamma^i V_\Gamma^i\right]^T$. La première et la dernière étape sont prises en charge en interne par le `Mortar_Exchanger` du `Mortar_Vector`, ce qui permet de clarifier l'implémentation des algorithmes de résolution en masquant ces détails d'implémentation.

Résumé

Ce chapitre quelque peu technique a permis de présenter de façon assez détaillée l'implémentation de notre méthode adaptative au sein de Trio-U. Nous avons tout d'abord décrit les fonctionnalités globales de ce module, en précisant la démarche générale de l'algorithme adaptatif. La description du module `Framework` a été l'occasion de mettre en évidence les principes de développement adoptés, ainsi que la structure des itérations de la résolution. Nous avons ensuite consacré deux sections à la présentation des fonctionnalités du module `Geometry`, à savoir la mise en œuvre de la technique d'adaptation de maillage retenue, et la gestion de l'équilibrage dynamique de la charge. Nous avons enfin terminé ce chapitre par le détail du module `Mortar`, qui implémente les concepts présentés au chapitre 4. Lors de cette présentation, nous avons cherché à insister tant sur les possibilités de réutilisation des fonctionnalités déjà présentes dans la plate–forme, que sur l'intérêt des nouveaux développements apportés. Nous disposons donc maintenant d'une implémentation valide de notre méthode adaptative. Le chapitre suivant sera donc consacré à la présentation de différents résultats obtenus grâce à cette technique.

CHAPITRE 7

Résultats numériques et algorithmiques

Nous consacrons ce dernier chapitre à la présentation de divers résultats obtenus grâce à notre méthode adaptative. Nous cherchons tout d'abord à démontrer les possibilités offertes par notre implémentation, en présentant les résultats obtenus lors de la résolution de quelques problèmes modèles. À ce titre, nous considérons donc divers problèmes de LAPLACE *ou de* STOKES*, stationnaires ou instationnaire, bidimensionnels ou tridimensionnels, afin d'illustrer les différentes fonctionnalités disponibles. Nous chercherons ensuite à démontrer le bon comportement des algorithmes tant en termes de fiabilité qu'en termes de performances. Ainsi, nous vérifions expérimentalement que l'adaptation de maillage choisie ne dégrade pas la qualité des éléments du maillage, et que les outils utilisés pour équilibrer la charge sont effectivement pertinents. Enfin, nous nous proposons de comparer le comportement de notre méthode avec les résultats obtenus par le couplage entre Trio-U et Homard, qui permet également de réaliser des calculs adaptatifs.*

7.1 Problème de LAPLACE stationnaire bidimensionnel

Nous allons présenter dans ce qui suit les résultats de différents calculs réalisés avec le module présenté dans le chapitre précédent. Afin d'illustrer au mieux les capacités de notre méthode adaptative, nous avons choisi de traiter des problèmes dont la solution analytique est connue. Ainsi, nous nous affranchissons des difficultés liées à l'efficacité de tel ou tel estimateur d'erreur *a posteriori*.

Le problème que nous étudions ici est le suivant :

Problème 7.1 : *Soit* Ω *le sous–domaine de* $\mathbb{R}^2$ *défini par* $\Omega = [-1,1] \times [-1,1]$. *On cherche une fonction* u *définie sur* Ω *vérifiant :*

$$\begin{aligned} -\Delta u &= f \quad \text{dans } \Omega \\ u &= g \quad \text{sur } \partial\Omega \end{aligned}$$

où l'on a posé :

$$f = 4\alpha\left(1-\alpha\left(x^2+y^2\right)\right)e^{-\alpha\left(x^2+y^2\right)} \qquad g = e^{-\alpha\left(x^2+y^2\right)} \qquad \alpha = 10^4$$

La solution analytique de ce problème est donnée par :

$$u = e^{-\alpha\left(x^2+y^2\right)}$$

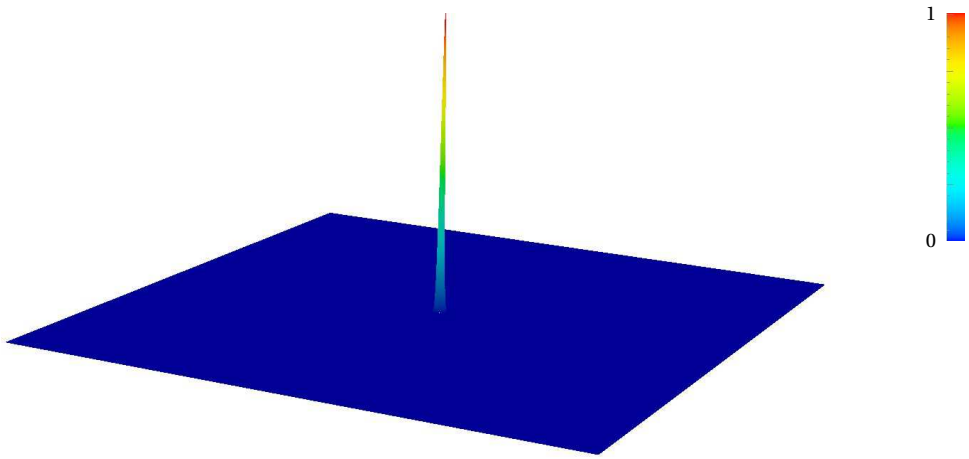


FIG. 7.1 : La solution du problème 7.1

Il s'agit d'une fonction gaussienne centrée à l'origine, dont la valeur maximale est 1, et qui est d'autant plus étroite que le paramètre α est grand. Notre choix de α nous assure que bien que cette fonction soit très régulière, elle présente de fortes variations sur un voisinage restreint de l'origine. Nous l'avons représentée sur la figure 7.1. La figure 7.2 présente le maillage initial utilisé, ainsi que le maillage adapté obtenu en se fixant comme objectif une erreur numérique en norme L^2 inférieure à 10^{-6}. Le tableau 7.3 présente quelques paramètres caractérisant ce calcul.

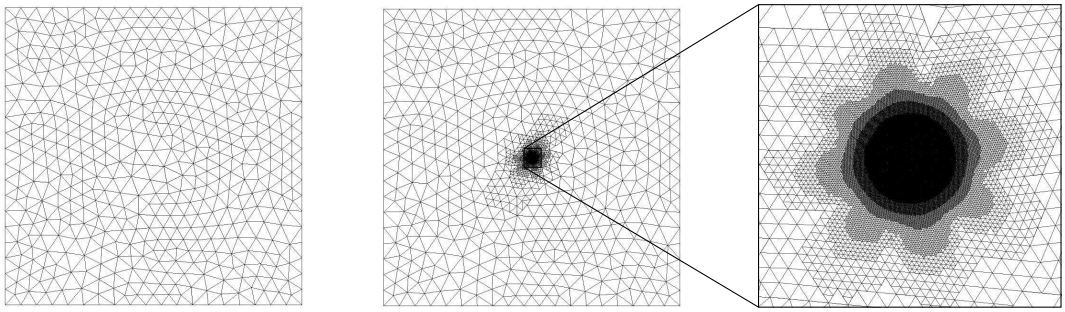


FIG. 7.2 : Maillage initial et maillage adapté pour le problème 7.1

Nombre de mailles initial	924
Nombre de mailles final	218433
Nombre de niveaux de raffinement	11
Nombre de mailles du maillage totalement raffiné équivalent	3875536896

FIG. 7.3 : Paramètres de la résolution du problème 7.1

7.2 Problème de LAPLACE tridimensionnel stationnaire

Nous allons maintenant présenter le résultat de la résolution du problème suivant :

Problème 7.2 : *Soit* Ω *le sous–domaine de* $\mathbb{R}^3$ *défini par :*

$$\Omega = [-1,1]\times[-1,1]\times[-1,1]/[0,1]\times[0,1]\times[0,1]$$

On cherche une fonction u *définie sur* Ω *vérifiant :*

$$\begin{aligned} -\Delta u &= f \quad dans\,\Omega \\ u &= g \quad sur\,\partial\Omega \end{aligned}$$

où l'on a posé :

$$f = 2\alpha\left(3-2\alpha\left(x^2+y^2+z^2\right)\right)e^{-\alpha\left(x^2+y^2+z^2\right)} \qquad g = e^{-\alpha\left(x^2+y^2+z^2\right)} \qquad \alpha = 10^2$$

La solution analytique de ce problème est donnée par :

$$u = e^{-\alpha\left(x^2+y^2+z^2\right)}$$

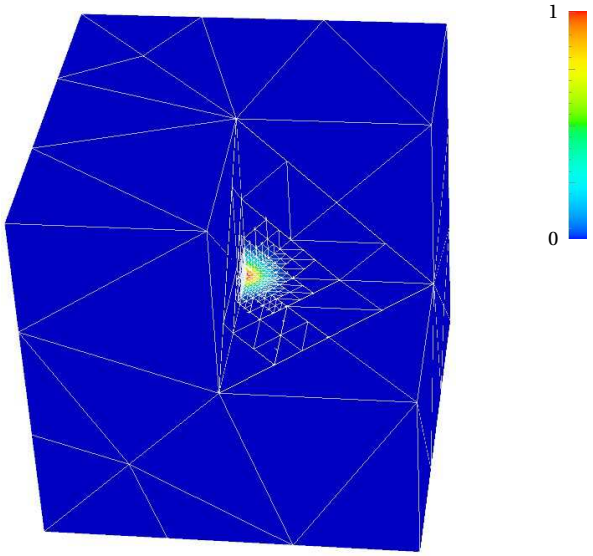


FIG. 7.4 : Solution et maillage adapté obtenus lors de la résolution du problème 7.2

La figure 7.4 représente la solution obtenue en fixant comme critère d'arrêt une erreur en norme L^2 inférieure à 10^{-3}. Le tableau 7.5 donne quelques paramètres caractérisant ce calcul.

7.3 Problème de LAPLACE bidimensionnel instationnaire

Nous nous proposons de résoudre maintenant une variante instationnaire du problème 7.1. Pour cela, nous allons déplacer le centre de la gaussienne le long d'un cercle

Nombre de mailles initial	208
Nombre de mailles final	28831
Nombre de niveaux de raffinement	6
Nombre de mailles du maillage totalement raffiné équivalent	54525952

FIG. 7.5 : Paramètres de la résolution du problème 7.2

centré sur l'origine. Nous vérifierons donc que le schéma en temps développé dans notre module `AMR` permet d'adapter dynamiquement le maillage pour que la zone raffinée suive la « singularité » du terme source. Nous formulons notre problème de la façon suivante :

Problème 7.3 : *Soit* Ω *le sous–domaine de* $\mathbb{R}^2$ *défini par* $\Omega = [-1,1] \times [-1,1]$. *On cherche une fonction* u *définie sur* $[0,2\pi] \times \Omega$ *vérifiant :*

$$\begin{aligned} \partial_t u - \Delta u &= f \quad dans\, \Omega \\ u &= g \quad sur\, \partial\Omega \end{aligned}$$

où l'on a posé :

$$\begin{aligned} f &= 4\alpha \left(1 - \alpha \left((x - x_0(t))^2 + \left(y - y_0(t)\right)^2\right)\right) e^{-\alpha\left((x-x_0(t))^2+\left(y-y_0(t)\right)^2\right)} \\ g &= e^{-\alpha\left((x-x_0(t))^2+\left(y-y_0(t)\right)^2\right)} \end{aligned}$$

et avec :

$$x_0(t) = r\cos(t) \qquad y_0(t) = r\sin(t) \qquad \alpha = 10^4 \qquad r = 0.25$$

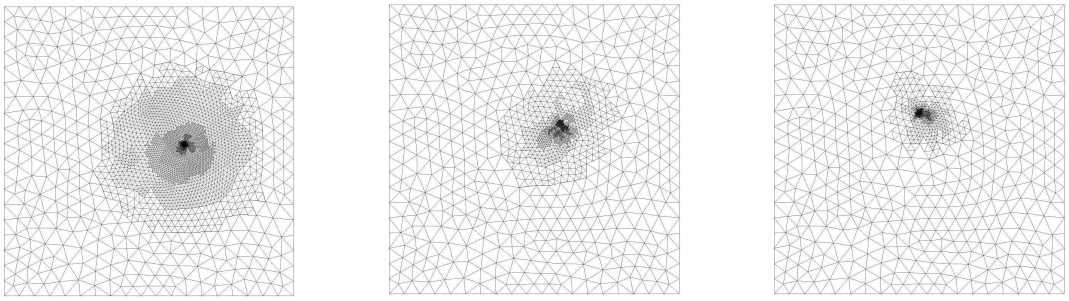
La solution de ce problème est donnée par :

$$u = e^{-\alpha\left((x-x_0(t))^2+\left(y-y_0(t)\right)^2\right)}$$

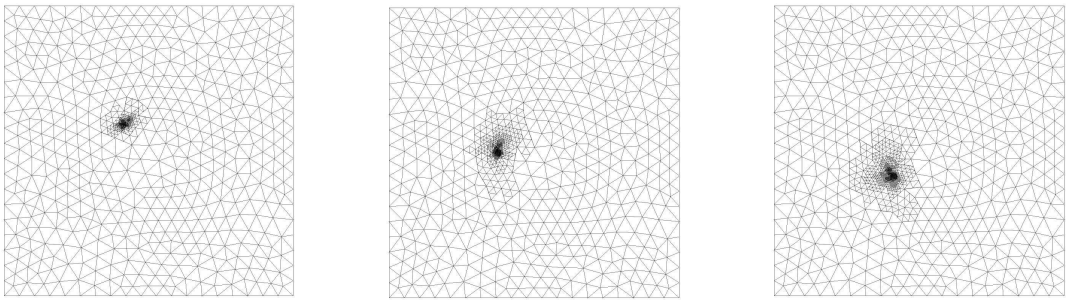
La figure 7.6 présente différents maillages obtenus au cours des itérations temporelles. Nous remarquons sur cette figure que le maillage 7.6(a) semble moins raffiné que les autres. Cette observation se justifie par le fait que la condition initiale du problème est difficile à imposer de façon précise sur le maillage grossier initial. Une conséquence importante de ce phénomène est que l'erreur L^2 utilisée pour déterminer l'adaptation du maillage est bornée inférieurement par l'erreur d'approximation de cette condition initiale sur le maillage grossier. Cette difficulté sera contournée par une correction de l'imposition de la condition initiale dans les futurs développements du module `AMR`.

7.4 Problème de STOKES bidimensionnel stationnaire

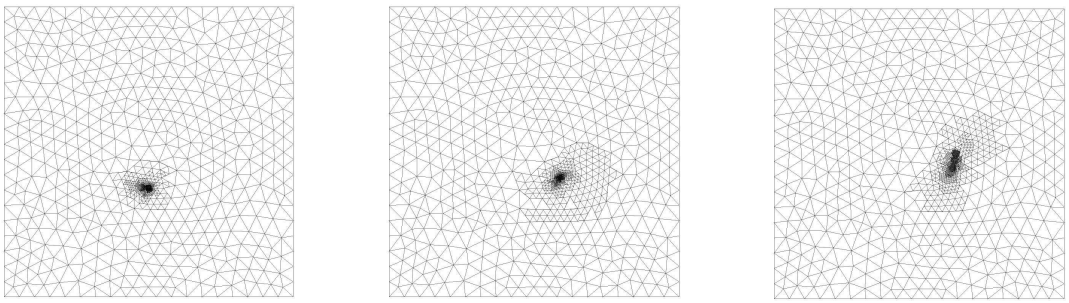
Afin de démontrer les possibilités d'utilisation de notre méthode, nous présentons maintenant le résultat d'un calcul de la cavité de STOKES. Il s'agit d'un problème classique, dont la solution analytique n'est pas connue. Nous ne pouvons donc pas utiliser l'erreur L^2 locale comme critère de raffinement, mais nous pouvons tout de même utiliser les estimations d'erreur *a posteriori* pour déterminer les zones à raffiner. Ce problème peut être décrit comme suit :



(a) Maillage à $t = 0$ (b) Maillage à $t = \frac{\pi}{4}$ (c) Maillage à $t = \frac{\pi}{2}$



(d) Maillage à $t = \frac{3\pi}{4}$ (e) Maillage à $t = \pi$ (f) Maillage à $t = \frac{5\pi}{4}$



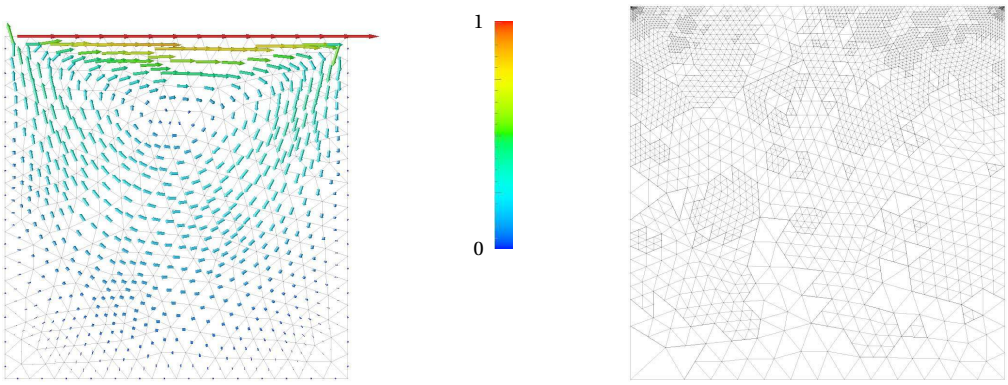
(g) Maillage à $t = \frac{3\pi}{2}$ (h) Maillage à $t = \frac{7\pi}{4}$ (i) Maillage à $t = 2\pi$

FIG. 7.6 : Maillages adaptés obtenus lors de la résolution du problème 7.3

Problème 7.4 : *Soit* Ω *le sous–domaine de* $\mathbb{R}^2$ *défini par* $\Omega = [-1,1] \times [-1,1]$. *On note* Γ_0 *la partie du bord de* Ω *pour laquelle* $y = 1$, *et* Γ_1 *le reste du bord de* Ω. *On cherche un couple de fonctions* $(\boldsymbol{u}, p)$, *définies sur* Ω *vérifiant :*

$$\begin{aligned} -\Delta \boldsymbol{u} + \nabla p &= 0 && \text{dans } \Omega \\ \nabla \cdot \boldsymbol{u} &= 0 && \text{dans } \Omega \\ \boldsymbol{u} &= (1,0)^T && \text{sur } \Gamma_0 \\ \boldsymbol{u} &= (0,0)^T && \text{sur } \Gamma_1 \end{aligned}$$

La figure 7.7 présente le champ de vitesse $\boldsymbol{u}$ calculé à l'aide de notre méthode, en utilisant la version étendue de la méthode VEF de Trio-U. Le tableau 7.8 présente encore une fois les caractéristiques des maillages générés. Nous n'avons pas représenté le champ de pression car ce dernier n'est pas facilement représentable. En effet, il présente essentiellement deux singularités très fortes, localisées sur les coins supérieurs du domaine de calcul. Étant donné la finesse du maillage adapté dans ces régions, la représentation graphique de ces singularité ne s'avère pas significative.



(a) Solution en vitesse du problème 7.4 (b) Maillage adapté

FIG. 7.7 : Résultats de la résolution du problème 7.4

Nombre de mailles initial	118
Nombre de mailles final	4738
Nombre de niveaux de raffinement	9
Nombre de mailles du maillage totalement raffiné équivalent	30932992

FIG. 7.8 : Paramètres de la résolution du problème 7.4

7.5 Conservation de la qualité des éléments du maillage

Une des raisons essentielles des choix retenus en matière de technique d'adaptation de maillage était la conservation de la qualité géométrique des éléments au cours du raffinement. Notre méthode permet de garantir la non–dégénérescence des éléments en fixant une borne supérieure au nombre de familles de congruence d'éléments (voir le chapitre 3). Nous avons vérifié cela expérimentalement en considérant un domaine initial composé d'un unique tétraèdre, auquel nous avons appliqué récursivement notre algorithme de raffinement. La figure 7.9 présente l'évolution du nombre de familles de congruence d'éléments obtenu, en fonction du nombre d'étapes de raffinement.

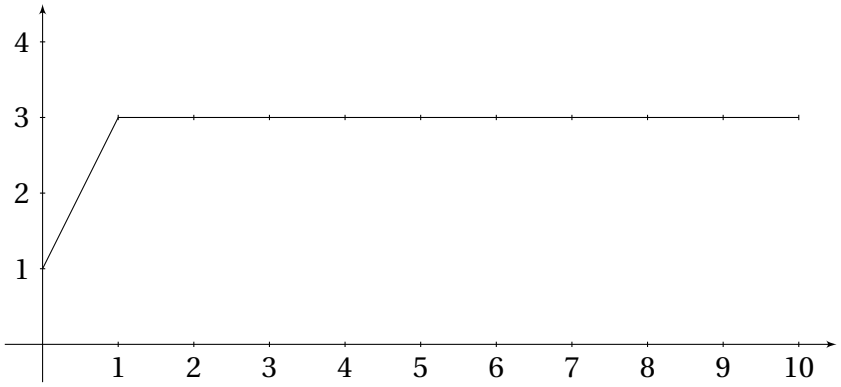


FIG. 7.9 : Évolution du nombre de types d'éléments au cours de raffinements successifs

7.6 Efficacité des politiques d'équilibrage de la charge

L'efficacité des algorithmes d'équilibrage de la charge peut être qualifiée de deux façons. En premier lieu, on est en droit de se demander s'il est pertinent, voire utile, de chercher à équilibrer la charge de calcul. Nous allons voir dans le paragraphe qui suit que cela est en réalité essentiel à l'efficacité du calcul. Ensuite, il s'agit de vérifier que l'algorithme d'adaptation de la charge ne présente pas un coût trop important en regard de celui de la résolution du problème.

La pertinence de l'équilibrage de charge est plus qu'évidente au regard de l'expérience que nous avons réalisée. Nous avons cherché à résoudre un problème de LAPLACE tridimensionnel du type de celui présenté au paragraphe 7.2, en changeant simplement quelques paramètres. Ainsi, nous avons utilisé un maillage initial plus fin (21232 éléments au lieu de 208), et nous avons fixé le paramètre α à 10^4, au lieu de 10^2, afin d'augmenter l'importance de la singularité à l'origine. Nous avons tenté de résoudre ce problème dans trois configurations différentes :

– sans équilibrage de charge ;
– en utilisant l'équilibrage de charge grossier (voir paragraphe 6.4.1) ;
– en utilisant l'équilibrage de charge fin (voir paragraphe 6.4.1).

Nous avons réalisé ces calculs en utilisant 8 processeurs d'une machine massivement parallèle à mémoire distribuée. Il s'agit d'une machine à architecture hybride, comprenant 25 cœurs (AMD) de 8 unités de calculs, reliés entre eux par un réseau rapide dédié (InfiniBand). L'utilisation de cette machine passe par un système de files d'attente[1], qui permet d'allouer au mieux les ressources disponibles. Lors du lancement du calcul, nous avons choisi de limiter à 30 minutes la durée de nos calculs, afin de pouvoir disposer d'une priorité importante dans la file d'attente. Le résultat de cette expérience est sans appel : alors que le calcul non équilibré et le calcul grossièrement équilibré n'ont pas pu se terminer avant la fin des 30 minutes allouées, le calcul finement équilibré a convergé au bout de 10 minutes et 34 secondes. Nous voyons donc clairement l'utilité des algorithmes d'équilibrage de charge à la lumière de cette expérience.

Le deuxième aspect concernant l'efficacité des algorithmes d'adaptation de la charge consiste à comparer leurs temps d'exécution avec celui de la résolution du problème. Là encore, les chiffres parlent d'eux mêmes, comme on peut le voir sur la figure 7.10. Ce tableau rassemble les statistiques d'exécution obtenues lors de la résolution du pro-

Temps consacré à l'adaptation du maillage	3.85s	4.1 %
Temps consacré à la résolution numérique	64.94s	69.2 %
Temps consacré au postraitement des champs	20.23s	21.6 %
Temps consacré à l'équilibrage de la charge	1.42s	1.5 %
Temps de résolution total	93.83s	100 %

FIG. 7.10 : Statistiques de la résolution du problème 7.1

blème 7.1. Nous voyons clairement que le temps pris par l'équilibrage de la charge reste parfaitement négligeable devant le temps nécessaire à la résolution du problème, ce qui confirme l'intérêt porté aux politiques d'équilibrage de la charge pour les calculs adaptatifs à vocation industrielle.

[1] *batch queues* en anglais

7.7 Comparaison avec le couplage Trio-U–Homard

7.7.1 Présentation du couplage Trio-U–Homard

Une autre approche a été proposée [79] pour réaliser des simulations adaptatives avec Trio-U. Cette technique repose sur le couplage entre Trio-U et le logiciel Homard [106], développé à EDF. Il s'agit d'un logiciel dédié à l'adaptation de maillage, qui fonctionne selon le principe suivant. À partir d'un maillage initial et d'un « indicateur de raffinement », qui a pour responsabilité d'identifier les éléments qui nécessitent d'être raffinés, Homard est capable de générer un nouveau maillage adapté qui satisfait au mieux les spécifications données. Du point de vue de Trio-U, ce couplage correspond à une approche par regénération globale (voir le paragraphe 1.4.1), même si les algorithmes d'Homard reposent sur une autre conception.

Le point important de cette approche est que le maillage généré par Homard est toujours conforme. Ainsi, la mise en œuvre de ce couplage ne nécessite que le développement d'opérateurs d'interpolation adaptés et la mise en place des interfaces entre les deux logiciels (format des fichiers d'échange, génération des indicateurs de raffinement, etc). En particulier, il n'est nullement nécessaire de modifier les méthodes numériques ou les solveurs de Trio-U pour utiliser ce couplage. Cependant, l'implémentation actuelle de ce couplage ne repose que sur la version séquentielle d'Homard, avec toutes les limitations que cela comporte en termes de performances et de volumes de données.

7.7.2 Comparaison des résultats

Afin de comparer ces deux approches, nous avons défini quatre calculs de référence, deux en dimension 2, et deux en dimension 3. Voici les spécifications de ces problèmes :

Problème 7.5 : *Soit* $\Omega = [-1,1]\times[-1,1]$. *On cherche une fonction* u *définie sur* Ω *telle que :*

$$\begin{aligned} -\Delta u &= 200000\left(1-1000\left(x^2+y^2\right)\right)e^{\left(-1000\left(x^2+y^2\right)\right)} && \text{dans } \Omega \\ u &= 50e^{\left(-1000\left(x^2+y^2\right)\right)} && \text{sur } \partial\Omega \end{aligned}$$

La solution analytique de ce problème est donnée par :

$$u = 50e^{\left(-1000\left(x^2+y^2\right)\right)}$$

Problème 7.6 : *Soit* $\Omega = [-1,1]\times[-1,1]$. *On cherche une fonction* u *définie sur* Ω *telle que :*

$$\begin{aligned} -\Delta u &= 500000\left(1-2500\left(x^2+y^2\right)\right)e^{\left(-2500\left(x^2+y^2\right)\right)} && \text{dans } \Omega \\ u &= 50e^{\left(-2500\left(x^2+y^2\right)\right)} && \text{sur } \partial\Omega \end{aligned}$$

La solution analytique de ce problème est donnée par :

$$u = 50e^{\left(-2500\left(x^2+y^2\right)\right)}$$

Problème 7.7 : *Soit* $\Omega = [-1,1]\times[-1,1]\times[-1,1]$. *On cherche une fonction* u *définie sur* Ω *telle que :*

$$\begin{aligned} -\Delta u &= 250000\left(3-5000\left(x^2+y^2+z^2\right)\right)e^{\left(-2500\left(x^2+y^2+z^2\right)\right)} && \text{dans } \Omega \\ u &= 50e^{\left(-2500\left(x^2+y^2+z^2\right)\right)} && \text{sur } \partial\Omega \end{aligned}$$

La solution analytique de ce problème est donnée par :

$$u = 50e^{\left(-2500\left(x^2+y^2+z^2\right)\right)}$$

Problème 7.8 : *Soit* $\Omega = [-1,1]\times[-1,1]\times[-1,1]$. *On cherche une fonction* u *définie sur* Ω *telle que :*

$$\begin{aligned} -\Delta u &= 250000\left(3-5000\left(x^2+y^2+z^2\right)\right)e^{\left(-2500\left(x^2+y^2+z^2\right)\right)} && \text{dans } \Omega \\ u &= 50e^{\left(-2500\left(x^2+y^2+z^2\right)\right)} && \text{sur } \partial\Omega \end{aligned}$$

La solution analytique de ce problème est donnée par :

$$u = 50e^{\left(-2500\left(x^2+y^2+z^2\right)\right)}$$

Notons que les problèmes 7.7 et 7.8 sont identiques dans leur formulation, nous verrons par la suite ce qui les différencie.

L'estimateur d'erreur utilisé repose sur la donnée de la solution analytique pour calculer l'erreur locale en norme L^2. La stratégie de raffinement adoptée consiste à spécifier un certain nombre d'étapes de raffinement, et pour chaque étape, un seuil de raffinement $\alpha \in]0,1[$. Notant e_M (resp. e_m) l'erreur locale maximale (resp. minimale), on décidera de raffiner tout élément dont l'erreur locale e vérifie :

$$e_m + \alpha(e_M - e_m) \leq e \leq e_M$$

Les problèmes se distinguent alors selon leur formulation mathématique, le maillage initial et, les valeurs des seuils de raffinements utilisés.

Les tableaux 7.11, 7.12, 7.13 et 7.14 présentent les résultats obtenus pour chacun de ces problèmes. Deux remarques nous semblent importantes. Tout d'abord, il semble que le couplage Trio-U–Homard génère des maillages sensiblement plus gros que ceux générés par le module `AMR` (jusqu'à un facteur 10 dans le cas du problème 7.7). Ensuite, l'erreur en norme L^2 ne semble pas se comporter de la même façon selon la méthode utilisée : il semble, en particulier en dimension 3, que le couplage Trio-U–Homard ne parvienne pas à faire décroître l'erreur en raffinant le maillage, alors qu'on observe l'inverse dans le cas du module `AMR`.

La première observation peut s'expliquer de façon relativement simple. Comme Homard cherche à construire un maillage conforme, il est amené à raffiner des éléments en dehors de la zone spécifiée par les indicateurs d'erreur. De plus, cet élargissement est déterminé par un algorithme de fermeture transitive, ce qui interdit l'établissement d'une borne supérieure au nombre de mailles générées. Ainsi, et cela s'applique tout particulièrement en dimension 3, les maillages générés par le couplage Trio-U–Homard sont nécessairement plus importants que ceux générés par le module `AMR`.

La seconde observation peut aussi s'expliquer par l'introduction d'éléments raffinés pour assurer la conformité. Notoirement, ces éléments présentent une qualité médiocre, ce qui nuit particulièrement à la qualité de la solution sur les degrés de liberté associés. C'est pourquoi l'erreur sera souvent importante sur les éléments introduits pour assurer la conformité du maillage. Cette explication donne par ailleurs un nouvel éclairage sur notre première observation : il est en effet fort probable que ces éléments soient sélectionnés pour être raffinés lors de la phase d'adaptation suivante, ce qui contribue à l'augmentation du nombre de mailles dans les maillages générés par cette approche.

Configuration		Homard		Module AMR	
Niveau	Seuil	Erreur L^2	Nb mailles	Erreur L^2	Nb mailles
0	0,1	1,6698	228	58,8968	228
1	0,4	78,1935	238	8,81287	528
2	0,4	26,1576	400	0,706326	666
3	0,4	6,74028	916	0,291522	696
4	0,5	0,467259	1238	0,187467	759
5	0,5	0,200104	1326	0,107021	873
6	0,5	0,0619134	1384	0,0842151	903
7	0,5	0,0517126	1560	0,0510143	1203
8	0,5	0,0614342	1910	0,0279536	1476
9	0,5	0,0218445	5262	0,0184377	1653
10	0,6	0,0162628	5800	0,0141364	2145
11	0,6	0,00934493	6550	0,00871763	2940
12	0,6	0,00815621	7054	0,00625707	3342
13	0,6	0,00626982	9126	0,004816	3765
14	0,6	0,00633541	9454	0,00418335	4719
15	–	0,00339773	16486	0,00312784	8319

FIG. 7.11 : Statistiques de la résolution du problème 7.5

Configuration		Homard		Module AMR	
Niveau	Seuil	Erreur L^2	Nb mailles	Erreur L^2	Nb mailles
0	0,2	21,6843	824	28,5975	824
1	0,5	66,3244	1484	0,550402	1382
2	0,5	0,764651	1816	0,203406	1400
3	0,6	0,686699	1868	0,111249	1454
4	0,6	0,0880288	2242	0,0829619	1496
5	0,6	0,0541546	2306	0,0613169	1616
6	0,6	0,0338884	2480	0,038268	1715
7	0,6	0,0611611	2606	0,030442	1766
8	0,6	0,0203828	4748	0,028759	1793
9	0,6	0,0173304	5194	0,0229384	2135
10	0,6	0,00786964	5808	0,019725	2309
11	0,6	0,00715076	6276	0,0135961	2564
12	0,6	0,00581539	7396	0,00817953	2834
13	0,6	0,00472266	9122	0,00660115	3035
14	–	0,00390035	13006	0,00557874	3467

FIG. 7.12 : Statistiques de la résolution du problème 7.6

Configuration		Homard		Module AMR	
Niveau	Seuil	Erreur L^2	Nb mailles	Erreur L^2	Nb mailles
0	0,05	0,00177101	682	9,58821	682
1	0,2	1,09095	746	2,51404	1599
2	0,2	1,35351	794	0,681478	1760
3	0,2	3,67299	1106	0,215938	1879
4	0,2	0,78294	1371	0,118979	1984
5	0,2	0,424748	4741	0,0588309	2418
6	0,5	0,422383	11279	0,0242355	4364
7	0,7	0,403016	14326	0,020347	4532
8	0,7	0,403157	20894	0,0181955	4840
9	0,7	0,401348	36298	0,014932	5435
10	–	0,399861	63064	0,0130577	6002

FIG. 7.13 : Statistiques de la résolution du problème 7.7

Configuration		Homard		Module AMR	
Niveau	Seuil	Erreur L^2	Nb mailles	Erreur L^2	Nb mailles
0	0,1	0,801892	7034	1,48188	7034
1	0,4	1,63402	7208	1,10687	7195
2	0,4	0,946192	7359	0,197726	7356
3	0,5	0,377505	7463	0,114061	7419
4	0,5	0,366992	8136	0,10335	7510
5	0,5	0,380251	10965	0,071759	7937
6	0,7	0,386078	22784	0,0412044	8399
7	0,7	0,386231	22867	0,0355085	8497
8	0,7	0,386564	26039	0,0300585	8714
9	0,7	0,388556	39949	0,0270445	8882
10	–	0,388606	40013	0,0231042	9442

FIG. 7.14 : Statistiques de la résolution du problème 7.8

Résumé

Ce dernier chapitre a été l'occasion de présenter différents résultats obtenus avec notre module AMR*. Nous avons illustré les capacités de notre méthode sur différents problèmes de* LAPLACE *ou de* STOKES*. Nous avons également montré la possibilité d'effectuer des simulations stationnaires comme instationnaires, en dimension 2 comme en dimension 3.*

Nous avons également illustré la mise en œuvre des choix retenus pour l'adaptation géométrique du maillage et la gestion de l'équilibrage de la charge. La comparaison de nos résultats avec l'approche utilisée dans le couplage Trio-U–Homard nous conforte d'ailleurs dans notre choix de traiter la non–conformité du maillage au niveau de la méthode numérique, et non au niveau de l'adaptation géométrique.

Conclusions et perspectives

Validation de la démarche

Au cours de ce travail de thèse, nous avons conçu, implémenté et validé une méthode numérique adaptative utilisable dans un environnement d'exécution parallèle pour la réalisation de simulations industrielles en thermohydraulique. Les idées fondatrices de cette méthode sont l'utilisation d'une technique d'adaptation de maillage à la fois simple, robuste et efficace, le traitement des non–conformités du maillage au niveau de la méthode numérique en utilisant les propriétés des méthodes d'éléments joints, et une implémentation parallèle reposant sur les concepts des méthodes de décomposition de domaines.

Nous avons successivement présenté ces idées et les concepts associés dans les chapitres 3, 4 et 5. Pour cela, nous avons spécifié un algorithme d'adaptation de maillage et une technique de raffinement adéquate. Nous avons également développé un formalisme d'éléments joints adapté à l'élément fini de Crouzeix–Raviart et à la forme des non–conformités générées par l'adaptation du maillage. Enfin, nous avons spécifié une démarche théorique générale à nos besoins, afin de concevoir l'implémentation parallèle de notre méthode adaptative.

Nous avons également développé le module décrit au chapitre 6 dans la plate–forme Trio-U pour mettre en œuvre ces idées. Les résultats présentés au chapitre 7 mettent en évidence le bon comportement de ce module lors de la résolution de divers problèmes. La comparaison avec une autre approche des simulations adaptatives déjà présente dans Trio-U nous conforte dans nos choix de conception et d'implémentation.

Perspectives et travaux futurs

Extension à de nouveaux problèmes

De nombreuses perspectives sont envisageables pour poursuivre ce travail. En effet, nous sommes encore loin de pouvoir traiter un problème complet de thermohydraulique. La possibilité de résoudre numériquement un problème de Stokes est un premier pas, mais il reste par exemple à prendre en compte correctement les phénomènes convectifs et turbulents. Or, le formalisme d'éléments joints que nous avons proposé ne tient pas du tout compte de ce genre de phénomènes, et reste très lié à la résolution d'un problème elliptique. Il sera donc probablement nécessaire d'étendre ce formalisme pour pouvoir résoudre de façon adaptative un problème de Navier–Stokes complet.

Extension à de nouvelles discrétisations

La méthode numérique proposée est intrinsèquement liée à la méthode de Volumes–Éléments Finis de Trio-U. Nous avons en effet souvent fait référence à son équivalence avec une formulation par Éléments Finis, ce qui permet de construire plus facilement le formalisme d'éléments joints. Il reste cependant souhaitable de chercher à étendre ces concepts à d'autres méthodes numériques, comme par exemple la méthode de Volumes–Différences Finis également présente dans Trio-U, ou encore d'autres méthodes de volumes finis non encore disponibles dans la plate–forme. L'intérêt d'une telle recherche réside dans le fait que l'adaptation du maillage est à proprement parler indépendante de la méthode numérique utilisée. Ainsi, le sous–module `Geometry` de notre module `AMR` sera parfaitement réutilisable pour la réalisation de ce genre de travaux.

Autres méthodes de résolution

Par ailleurs, nous avons insisté non sans raison au chapitre 3 sur les similitudes entre les h–méthodes et les méthodes multi–niveaux. En effet, les structures de données que nous avons mises en œuvre dans notre module pourraient être aisément adaptées à la résolution du problème par une approche de type multi–niveaux. Nous n'avons pas abordé cette question dans nos travaux car cela nous aurait quelque peu écarté de notre sujet, mais il est probable que la mise en œuvre d'un algorithme de résolution de type multi–niveaux puisse apporter un gain non négligeable en termes de rapidité d'exécution.

Estimations d'erreur *a posteriori*

Enfin, il nous semble important de souligner un point dont nous n'avons finalement que très peu parlé, mais qui s'avère aujourd'hui être un domaine de recherche particulièrement actif, à savoir la notion d'estimateur d'erreur. Dans nos travaux, nous nous sommes généralement cantonnés à résoudre des problèmes dont la solution analytique est connue. Cette démarche nous a été très utile pour valider nos algorithmes adaptatifs, mais elle ne s'applique absolument pas à la réalité des calculs industriels. Nous percevons donc les recherches effectuées en ce moment dans le domaine des estimations d'erreur *a posteriori* (voir par exemple [147, 148]) comme autant de perspectives de développement de notre méthode.

Bibliographie

[1] F. ALAUZET, P.-J. FREY, P.-L. GEORGE et B. MOHAMMADI : Adaptation de maillage non structuré 3D pour les problèmes instationnaires. Application à la mécanique des fluides. Rapport technique 4981, Institut National de Recherche en Informatique et en Automatique, projet GAMMA, Rocquencourt, 2003.

[2] S.-W. AMBLER : *The Elements of UML 2.0 Style*. Cambridge University Press, 2005. ISBN 0521616786.

[3] L. ANGERMANN : A posteriori error estimates for fem with violated galerkin orthogonality. *Numerical Methods for Partial Differential Equations*, 18(2):241–259, 2002.

[4] D.-N. ARNOLD, A. MUKHERJEE et L. POULY : Locally adapted tetrahedral meshes using bissection. *SIAM Journal of Scientific Computing*, 22(2):431–448, 2001.

[5] I. BABUŠKA et W. RHEINBOLT : Error estimates for adaptive finite element method computations. *SIAM Journal of Numerical Analysis*, 15:736–754, 1978.

[6] I. BABUŠKA et W. RHEINBOLT : A posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, 12:1597–1615, 1978.

[7] I. BABUŠKA et T. STROUBOULIS : *Finite Element Method and Its Reliability*. Clarenton Press, Oxford, 2001.

[8] I. BABUŠKA : Über Schwatzsche Algorithmen in partiallen Differntialgleichungen der mathematischen Physik. *ZAMM*, 37(7/8):243–245, 1957.

[9] R. BANK et R. SMITH : A posteriori error estimates base on hierarchical bases. *SIAM Journal of Numerical Analysis*, 30(4):921–935, 1993.

[10] R. BANK et A. WEISER : Some a posteriori error estimators for elliptic partial differential equations. *Mathematics of Computation*, 44:283–301, 1985.

[11] R.-E. BANK : PLTMG : A Software Package for Solving Elliptic Partial Differential Equations. Users' Guide 9.0. Rapport technique, Department of Mathematics, University of California, San Diego, California, 2004.

[12] R.-E. BANK, A.-H. SHERMAN et A. WEISER : Refinement algorithms and data structures for regular local mesh refinement. *In* R. STEPLEMAN, éditeur : *Scientific Computing*, pages 3–17. IMACS/North-Holland Publishing Company, Amsterdam, 1983.

[13] P. BASTIAN, K. BIRKEN, K. JOHANNSEN, S. LANG, H. RENTZ-REICHBERT et C. WIENERS : UG - a flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science*, 1:27–40, 1997.

[14] P. BASTIAN, K. BIRKEN, K. JOHANNSEN, S. LANG, C. WIENERS, G. WITTUM, V. REICHENBERGER et H. RENTZ-REICHBERT : A parallel software platform for solving problems of partial differential equations using unstructured grids and adaptive multigrid methods. *In* E. KRAUSE et W. JÄGER, éditeurs : *High Performance Computing in Science and Engineering'98*, pages 326–339. Springer, 1998.

[15] R. BECKER et R. RANNACHER : An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, 2001.

[16] Z. BELHACHMI : *A posteriori* error estimates for the 3D stabilized mortar finite element method applied to the LAPLACE equation. *Mathematical Modelling and Numerical Analysis*, 37(16):991–1011, 2003.

[17] Z. BELHACHMI : Residual a posteriori error estimates for a 3D mortar finite–element method : the STOKES problem. *Journal of Numerical Analysis*, 24(3):521–546, 2004.

[18] F. BEN BELGACEM : THE MORTAR FINITE ELEMENT METHOD WITH LAGRANGE MULTIPLIERS. *Numerische Mathematik*, 84(2):173–197, 1999.

[19] M.-J. BERGER : *Adaptive mesh refinement for hyperbolic partial differential equations*. Thèse de doctorat, Computer Science Department, Stanford University, 1982.

[20] M.-J. BERGER et P. COLLELA : Local adaptive mesh refinement for shock hydraudynamics. *Journal of Computational Physics*, 82:67–84, 1989.

[21] C. BERNARDI et F. HECHT : Mesh adaptivity in the mortar finite element method. *In* T. CHAN, T. KAKO, H. KAWARADA et O. PIRONNEAU, éditeurs : *12th International Conference on Domain Decomposition Methods*, 2001.

[22] C. BERNARDI et Y. MADAY : Mesh adaptivity in finite elements using the mortar method. *Revue européenne des Éléments Finis*, 9:451–465, 2000.

[23] C. BERNARDI, Y. MADAY et A. PATERA : A new nonconforming approach to domain decomposition : the mortar element method. *In* PITMAN, H. BREZIS et J.L. LIONS, éditeurs : *Séminaire du Collège de France*, 1990.

[24] J. BEY : Tetrahedral Grid Refinement. *Computing*, 55(4):355–378, 1995.

[25] J. BEY : Simplicial grid refinement : on FREUDENTHAL's algorithm and the optimal number of congruence classes. *Numerische Mathematik*, 85(1):1–29, 2000.

[26] R.-H. BISSELING : *Parallel Scientific Computation : A Structured Approach using BSP and MPI*. Oxford University Press, 2004.

[27] M. BOGOVSKIĬ : Solutions of some problems of vector analysis, associated with the operators div and grad. *In Theory of Cubature Formulas and the Applications of Functional Analysis to Problems of Mathematical Physics*, volume 149, pages 5–40. Trudy Sem. S. L. Soboleva, Akad. Nauk SSSR Sibirsk. Otdel. Inst. Mat., Novosibirsk, Russia, 1980.

[28] O. BONORDEN, B. JUURLINK, I. von OTTE et I RIEPING : The Paderborn University BSP (PUB) Library – Design, Implementation and Performance. *In IPPS '99/SPDP '99 : Proceedings of the 13th International Symposium on Parallel Processing and the 10th Symposium on Parallel and Distributed Processing*, pages 99–104, Washington, DC, USA, 1999. IEEE Computer Society.

[29] O. BONORDEN, B. JUURLINK, I. von OTTE et I RIEPING : The Paderborn University BSP (PUB) Library. *Parallel Computing*, 29(2):187–207, 2003.

[30] G. BOOCH, J. RUMBAUGH et I. JACOBSON : *Unified Modeling Language User Guide*. Addison–Wesley Professional, 2eédition, 2005. ISBN 0321367974.

[31] D.-R. BUTENHOF : *Programming with POSIX threads*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. ISBN 0-201-63392-2.

[32] E. BÄNSCH : Local mesh refinement in 2 and 3 dimensions. *Impact of Computing in Science and Engineering*, 3:181–191, 1991.

[33] C. CALVIN, O. CUETO et P. EMONOT : An object oriented approach to the design of fluid mechanics software. *Mathematical Modeling and Numerical Analysis (M2AN), Special Issue on Scientific Software Tools*, 36:907–921, 2002.

[34] C. CALVIN et P. EMONOT : The Trio–Unitaire project : a parallel CFD 3D code. *In ISCOPE*, 1997.

[35] J. G. CASTANOS : *Parallel Adaptive Unstructured Computation*. Thèse de doctorat, Brown University, USA, 2000.

[36] J. G. CASTANOS et J. SAVAGE : The dynamic adaptation of parallel mesh-based computation. *In Eighth SIAM Conference on Parallel Processing for Scientific Computation*, 1997.

[37] J. G. CASTANOS et J. SAVAGE : Parallel refinement of unstructured meshes. *In IASTED Conference on Parallel and Distributed Computing and Systems (PDCS'99)*, 1999.

[38] J. G. CASTANOS et J. SAVAGE : PARED : a framework for the adaptive solution of PDES. *In Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC'99)*, 1999.

[39] R. CHANDRA, R. MENON, L. DAGUM, D. KOHR, D. MAYDAN et J. MC DONALD : *Parallel Programming in OpenMP*. Morgan Kaufmann Publishers, 2000.

[40] P.-G. CIARLET : *The finite element method for elliptic problems*. Studies in mathematics and its applications. North-Holland, Amsterdam, 1976.

[41] P. CLÉMENT : Approximation by finite element functions using local regularization. *Revue Française d'Automatique, Informatique et Recherche Opérationnelle, Analyse Numérique*, 9(R-2):77–84, 1975.

[42] P. CLÉMENT : *Développement et applications de méthodes numériques volumes finis pour la description d'écoulements océaniques*. Thèse de doctorat, Université Joseph Fourier–Grenoble I, France, 1996.

[43] P. COLLELA, D.-T. DRAVES, T.-J. LIGOCKI, D.-F. MARTIN, D. MODIANO, D.-B SERAFINI et B. VAN STRAALEN : Chombo software package for AMR applications. Design document. Rapport technique, Lawrence Berkeley National Laboratory, Berkeley, CA, 2003.

[44] X. CORÉ : *Méthode adaptative de raffinement local multi–niveaux pour le calcul d'écoulements réactifs à faible nombre de* MACH. Thèse de doctorat, Université de Provence, 2002.

[45] M. CROUZEIX et P.-A. RAVIART : Conforming and nonconforming finite element methods for solving the stationary STOKES equations. *Revue Française d'Automatique, Informatique et Recherche Opérationnelle, Analyse Numérique*, 7(R-3):33–76, 1973.

[46] L. DAGUM et R. MENON : OpenMP : An Industry–Standard API for Shared–Memory Programming. *IEEE Computational Science and Engineering*, 5:46–55, 1998.

[47] F. DAREMA-ROGERS, A. NORTON et G.-F. PFISTER : Using a Single Program, Multiple Data Computational Model for Parallel Execution of Scientific Applications. Rapport technique RC11552, IBM Research, 1985.

[48] R. DAUTRAY et J.-L. LIONS : *Analyse Mathématique et Calcul Numérique pour les Sciences et les Techniques*. Masson, 1987.

[49] L. DEBREU et C. VOULAND : AGRIF, Adaptive Grid Refinement in Fortran, Tutorial. Rapport technique, LMC–IMAG, Grenoble, 2003.

[50] L. DEBREU et C. VOULAND : AGRIF, Adaptive Grid Refinement in Fortran, User's guide. Rapport technique, LMC–IMAG, Grenoble, 2003.

[51] L. EL ALAOUI et A. ERN : Residual and hierarchical *a posteriori* error estimates for nonconforming mixed finite element methods. *Mathematical Modeling and Numerical Analysis (M2AN)*, 38(6):903–929, 2004.

[52] P. ÉMONOT : *Méthodes de volumes-éléments finis : applications aux équations de* NAVIER–STOKES *et résultats de convergence*. Thèse de doctorat, Université Claude Bernard–Lyon I, France, 1992.

[53] K. ERIKSSON, D. ESTEP, P. HANSBO et C. JOHNSON : Introduction to Adaptive Methods for Differential Equations. *Acta Numerica*, pages 001–054, 1995.

[54] A. ERN et J.-L. GUERMOND : *Theory and Practice of Finite Elements*. Springer–Verlag, 2004. ISBN 0-387-20574-8.

[55] C.-M. FIDUCCIA et R.-M. MATTHEYSES : A linear time heuristic for improving network partitions. *In 19th IEEE Design Automation Conference*, pages 175–181, 1982.

[56] M. FIEDLER : A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.

[57] M.-J. FLYNN : Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960, 1972.

[58] M.-J. FLYNN et K.-W. RUDD : Parallel architectures. *ACM Computing Surveys*, 28 (1):67–70, 1996.

[59] M. FORTIN et M. SOULIÉ : A non–conforming piecewise quadratic finite element on triangles. *Internationnal Journal for Numerical Methods in Engineering*, 19:505–520, 1983.

[60] T. FORTIN : *Une méthode Éléments Finis à décomposition L^2 d'ordre élevé motivée par la simulation d'écoulements diphasiques bas* MACH. Thèse de doctorat, Université Paris VI, France, 2006.

[61] I. FOSTER : *Designing and Building Parallel Programs*. Addison–Wesley, 1995.

[62] P. FRAUENFELDER et C. LAGE : Concepts – An Object–Oriented Software Package for Partial Differential Equations. *Mathematical Modelling and Numerical Analysis*, 36 (5):937–951, 2002.

[63] H. FREUDENTHAL : Simplizialzerlegungen von beschränkter Flachheit. *Annals of Mathematics*, 43:580–582, 1942.

[64] P.-J. FREY et P.-L. GEORGE : *Maillages, applications aux éléments finis*. HERMÈS Science Publications, 1999. ISBN 2-7462-0024-4.

[65] E. GAMMA, R. HELM, R. JOHNSON et J. VLISSIDES : *Design Patterns, Elements of Reusable Object–Oriented Software*. Addison-Wesley Professional, 1reédition, 1995. ISBN 0-20163-361-2.

[66] M.-R. GAREY et D.-S. JOHNSON : *Computers and Intractability : A Guide to the Theory of NP-completeness*. Freeman, W.-H., 1979.

[67] P.-L. GEORGE et H. BOROUCHAKI : *Triangulation de Delaunay et maillage. Applications aux éléments finis*. HERMÈS Science Publications, 1997. ISBN 2-86601-625-4.

[68] F.-H. HARLOW et J.-E. WELCH : Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[69] S. HEIB : *Nouvelles discrétisations non–structurées pour des écoulements de fluides à incompressibilité renforcée*. Thèse de doctorat, Université Paris VI, France, 2003.

[70] R. HEMPEL : The MPI Standard for Message Passing. *In* W. GENTZSCH et U. HARMS, éditeurs : *High–Performance Computing and Networking, International Conference and Exhibition, Proceedings, Volume II : Networking and Tools*, volume 797, pages 247–252, 1994.

[71] R. HEMPEL et D. WALKER : The Emergence of the MPI Message Passing Standard for Parallel Computing. *Computer Standards & Interfaces*, 21:51–62, 1999.

[72] B. HENDRICKSON et R. LELAND : An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2): 452–469, 1995.

[73] J. HILL, S.-R. DONALDSON et D.-B. SKILLICORN : Portability of performance with the *BSPlib* communications library. *In Programming Models for Massively Parallel Computers, (MPPM'97)*, London, 1997. IEEE Computer Society Press.

[74] J. HILL, B. MC COLL, D. STEFANESCU, M. GOUDREAU, K. LANG, S. RAO, T. SUEL, T. TSANTILAS et R BISSELING : BSPlib : the BSP Programming Library. *Parallel Computing*, 24:1947–1980, 1998.

[75] IEEE : *IEEE Standard for Multithreaded Programming POSIX.1.c.* IEEE Computer Society Press, 1995.

[76] H. JOHANSEN et P. COLLELA : A cartesian grid embedded boundary method for POISSON's equation on irregular domains. *Journal of Computational Physics*, 147:60–85, 1998.

[77] V. JOHN : A posteriori L^2-error estimates for the non–conforming $\mathbb{P}^1/\mathbb{P}^0$–finite element discretisation of the STOKES equations. *Journal of Computational and Applied Mathematics*, 96(2):99–116, 1998.

[78] C. JOHNSON : *Numerical Solution of Partial Differential Equations by Finite Element Method.* Cambridge University Press, Cambridge, Royaume–Uni, 1987.

[79] J.-C. JOUHAUD, G. NICOLAS, G. FAUCHET, O. CIONI et P. ÉMONOT : Couplage de l'outil Homard (EDF) avec le code de calcul thermohydraulique Trio–U (CEA). Rapport technique, CERFACS–EDF–CEA, 2006.

[80] G. KARYPIS et V. KUMAR : A fast and high quality multilevel scheme for partitioning irregular graphs. Rapport technique TR 95–035, Department of Computer Science, University of Minnesota, Minneapolis, 1995.

[81] G. KARYPIS et V. KUMAR : Multilevel k–way partitioning scheme for irregular graphs. Rapport technique, Department of Computer Science, University of Minnesota, Minneapolis, 1995.

[82] G. KARYPIS et V. KUMAR : Parallel multilevel k–way partitioning scheme for irregular graphs. Rapport technique, Department of Computer Science, University of Minnesota, Minneapolis, 1996.

[83] G. KARYPIS et V. KUMAR : A coarse–grain parallel formulation of multilevel k–way graph partitioning algorithm. *In Eigth SIAM Conference on Parallel Processing for Scientific Computing*, Philadelphia, 1997.

[84] B. KERNIGHAN et S. LIN : An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, 1970.

[85] K. KHADRA : *Méthodes adaptatives de raffinement local multigrille : applications aux équations de* NAVIER–STOKES *et de l'énergie*. Thèse de doctorat, Université de Bordeaux I, 1994.

[86] R. LI, T. TANG et P. ZHANG : Moving mesh methods in multiple dimensions based on harmonic maps. *Journal of Computational Physics*, 170:562–588, 2001.

[87] R. LI, T. TANG et P. ZHANG : A moving mesh finite element algorithm for singular problems in two and three space dimensions. *Journal of Computational Physics*, 177:365–393, 2002.

[88] P.-L. LIONS : On the SCHWARZ alternating method. I. *In* R. GLOWINSKY, G.-H. GOLUB, G.-A. MEURANT et J. PÉRIAUX, éditeurs : *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 1–42, Paris, France, 1987. SIAM.

[89] P.-L. LIONS : On the SCHWARZ alternating method. II. *In* T. CHAN, R. GLOWINSKY, J. PÉRIAUX et O. WIDLUND, éditeurs : *Domain Decomposition Methods. Second International Symposium on Domain Decomposition Methods*, pages 47–70, Los Angeles, California, USA, 1988. SIAM.

[90] P.-L. LIONS : On the SCHWARZ alternating method. III : a variant for nonoverlapping subdomains. *In* T. CHAN, R. GLOWINSKY, J. PÉRIAUX et O. WIDLUND, éditeurs : *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 202–223, Houston, Texas, USA, 1989. SIAM.

[91] A. LIU et B. JOE : Quality local refinement of tetrahedral meshes based on bissection. *SIAM Journal of Scientific Computing*, 16:1269–1291, 1995.

[92] B. LUCQUIN et O. PIRONNEAU : *Introduction au calcul scientifique*. Masson, 1996. ISBN 2-225-85017-8.

[93] L. MARCINKOWSKI : The mortar element method with locally nonconforming elements. *BIT Numerical Mathematics*, 39(4):716–739, 1999.

[94] L. MARCINKOWSKI : Additive SCHWARZ Method for Mortar Discretisation of Elliptic Problems with P_1 Nonconforming Finite Elements. *BIT Numerical Mathematics*, 45(2):375–394, 2005.

[95] B. MATHIEU : *Documentation des structures de données parallèles de Trio-U*. CEA–DEN/DER/SSTH/LMDL, 2006. Documentation interne du projet Trio-U.

[96] J.-M.-L. MAUBACH : *Iterative Methods for Nonlinear Partial Differential Equations*. Thèse de doctorat, Université de Nijmegen, 1991.

[97] J.-M.-L. MAUBACH : Local bissection refinement for n–simplicial grids generated by reflection. *SIAM Journal of Scientific Computing*, 16:210—227, 1995.

[98] S.-F. MC CORMICK : Computational complexity of the fast adaptive composite grid (FAC) method. *Applied Numerical Mathematics*, 6:315–327, 1989.

[99] MESSAGE PASSING INTERFACE FORUM (MPIF) : MPI : A message-passing interface standard. Rapport technique UT-CS-94-230, University of Tennessee, Knoxville, 1994.

[100] MESSAGE PASSING INTERFACE FORUM (MPIF) : MPI-2 : Extensions to the Message-Passing Interface. Rapport technique, University of Tennessee, Knoxville, 1996.

[101] W.-F. MITCHELL : Adaptive refinement for arbitrary finite element spaces with hierarchical basis. *Journal of Computational and Applied Mathematics*, 36:65–78, 1991.

[102] W.-F. MITCHELL : Refinement tree based partitioning for adaptive grids. *In Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 587–592, 1995.

[103] W.-F. MITCHELL : The full domain partition approach for parallel multigrid on adaptive grids. *In Eighth SIAM Conference on Parallel Processing for Scientific Computing*, 1997.

[104] W.-F. MITCHELL : The design of a parallel adaptive multi–level code in fortran 90. *In 2002 International Conference on Computational Science*, 2002.

[105] W.-F. MITCHELL : Parallel adaptive multi–level methods with full domain partitions. *Applied Numerical Analysis and Computational Mathematics*, 1:36–48, 2004.

[106] G. NICOLAS : http ://www.code-aster.org/outils/homard. Site officiel du logiciel Homard.

[107] M.-E. ONG : Uniform Refinement of a Tetrahedron. *SIAM Journal of Scientific Computing*, 15(5):1134–1144, 1994.

[108] P. OSWALD : On a BPX–preconditionner for $\mathbb{P}^1$ elements. *Computing*, 51:125–133, 1993.

[109] A. POTHEN, H. SIMON et K. LIOU : Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11:430–452, 1990.

[110] M.-J. QUINN : *Parallel Programming in C with MPI and OpenMP*. McGraw–Hill, 2003.

[111] P. QUINTON et Y. ROBERT : *Algorithmes et architectures systoliques*. Études et Recherches en Informatique. Dunod, 1997.

[112] P.-A. RAVIART et J.-M. THOMAS : Primal Hybrid Finite Elements Methods for 2nd Order Elliptic Equations. *Mathematics of Computation*, 31(138):391–413, 1977.

[113] P.-A. RAVIART et J.-M. THOMAS : *Introduction à l'analyse numérique des équations aux dérivées partielles*. Dunod, 1993.

[114] J.-F. REMACLE, J. FLAHERTY et M. SHEPHARD : Some issues on distributed mesh representation. *In Eigth International Grid Conference*, 2002.

[115] J.-F. REMACLE, O. KLAAS, J. FLAHERTY et M. SHEPHARD : Parallel algorithm oriented mesh database. *In Tenth International Meshing Roundtable*, 2001.

[116] J.-F. REMACLE et M. SHEPHARD : An algorithm oriented mesh database. *International Journal for Numerical Methods in Engineering*, 58:349–374, 2003.

[117] M.-C. RIVARA : Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International Journal for Numerical Methods in Engineering*, 20:745–756, 1984.

[118] M.-C. RIVARA : Design and data structure of fully adaptive multigrid, finite–element software. *Transactions on Mathematical Software*, 10:242–264, 1984.

[119] M.-C. RIVARA : Mesh refinement processes based on the generalized bisection of simplices. *Journal of Numerical Analysis*, 21:604–613, 1984.

[120] M.-C. RIVARA : Local modification of meshes for adaptive and/or multigrid finite–element methods. *Journal of Computational and Applied Mathematics*, 36:79–89, 1991.

[121] Y. SAAD : *Iterative Methods for Sparse Linear Systems*. SIAM, 2^{e} édition, 2003. ISBN 0-89871-534-2.

[122] M. SATO : OpenMP : Parallel Programming API for Shared Memory Multiprocessors and On–Chip Multiprocessors. *In Proceedings of the 15th international symposium on System Synthesis (ISSS '02)*, pages 109–111, 2002.

[123] F. SCHIEWECK : Analysis of post-processing for nonconforming finite element solutions. Rapport technique 36/99, Fakultät für Mathematik, Universität Magdeburg, 1999.

[124] F. SCHIEWECK : A general transfert operator for arbitrary finite element spaces. Rapport technique 25/00, Fakultät für Mathematik, Universität Magdeburg, 2000.

[125] F. SCHIEWECK : A posteriori error estimates with post–processing for nonconforming finite elements. Rapport technique 11/00, Fakultät für Mathematik, Universität Magdeburg, 2000.

[126] F. SCHIEWECK : Upper and lower a posteriori error estimates with post–processing for nonconforming finite elements. Rapport technique 29/01, Fakultät für Mathematik, Universität Magdeburg, 2001.

[127] F. SCHIEWECK : A posteriori error estimates with post–processing for nonconforming finite elements. *Mathematical Modeling and Numerical Analysis (M2AN)*, 36 (3):489–503, 2002.

[128] K. SCHOEGEL, G. KARYPIS et V. KUMAR : Multilevel diffusion schemes for repartitioning of adaptive meshes. Rapport technique 97-013, Department of Computer Science, University of Minnesota, Minneapolis, 1997.

[129] K. SCHOEGEL, G. KARYPIS et V. KUMAR : Parallel multilevel diffusion schemes for repartitioning of adaptive meshes. Rapport technique 97-014, Department of Computer Science, University of Minnesota, Minneapolis, 1997.

[130] C. SCHWAB : *p– and hp–Finite Element Methods*. Clarenton Press, Oxford, 1998.

[131] H.-A. SCHWARZ : Gesammelte Mathematische Abhandlungen. *Vierteljahrsschrift der Naturforschenden Gessellshaft in Zürich*, 15:272–286, 1870.

[132] A. SCHWERTNER CHARÃO : *Multiprogrammation parallèle générique des méthodes de décomposition de domaine*. Thèse de doctorat, Institut National Polytechnique de Grenoble, France, 2001.

[133] L.-R. SCOTT et S. ZHANG : Finite element interpolation of nonsmooth functions satisfying boundary conditions. *Mathematics of Computation*, 54(190):483–493, 1990.

[134] E.-G. SEWELL : A finite element program with automatic user–controlled mesh grading. *In* R. STEPLEMAN, éditeur : *Advances in Computer Methods for Partial Differential Equations III*, pages 8–10. IMACS, 1979.

[135] D.-B. SKILLICORN, J.-M.-D. HILL et W.-F. MC COLL : Questions and answers about BSP. *Journal of Scientific Programming*, 6(3), 1997.

[136] B. SMITH, P. BJORSTAD et W. GROPP : *Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[137] S.-L. SOBOLEV : L'Algorithme de SCHWARZ dans la Théorie de l'Élasticité. *Comptes Rendus (Doklady) de l'Académie des Sciences de l'URSS*, IV((XIII)6):243–246, 1936.

[138] B. STROUSTRUP : *The C++ Programming Language, Special Edition*. Pearson Education, 2003. ISBN 2-7440-7003-3.

[139] B. SZABÓ et I. BABUŠKA : *Finite Element Analysis*. John Wiley & Sons, New–York, 1991.

[140] Z. TAN, Z.-R. ZHANG, Y. HUANG et T. TANG : Moving mesh methods with locally varying time steps. *Journal of Computational Physics*, 200:347–367, 2004.

[141] H.-Z. TANG et T. TANG : Adaptive mesh methods for one– and two–dimensional hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 41:487–515, 2003.

[142] A. TOSELLI et O. WIDLUND : *Domain Decomposition Methods – Algorithms and Theory*, volume 34 de *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, 2005. ISBN 3-540-20696-5.

[143] C.-T. TRAXLER : An algorithm for adaptive mesh refinement in n dimensions. *Computing*, 59:115–137, 1997.

[144] R. VERFÜRTH : Error estimates for a mixed finite element approximation of the STOKES equation. *Revue Française d'Automatique, Informatique et Recherche Opérationnelle, Analyse Numérique*, 18:175–182, 1984.

[145] R. VERFÜRTH : *A Review of a posteriori error estimation and adaptive mesh refinement techniques*. Wiley, Chichester, Royaume–Uni, 1996.

[146] S. VINCENT et J.-P. CALTAGIRONE : One cell local multigrid method for solving unsteady incompressible multi–phase flows. *Journal of Computational Physics*, 163:172–215, 2000.

[147] M. VOHRALÍK : A posteriori error estimates for lowest–order mixed finite element discretizations of convection–diffusion–reaction equations. *SIAM Journal on Numerical Analysis*, 45:1570–1599, 2007.

[148] M. VOHRALÍK : A posteriori error estimation in the conforming finite element method based on its local conservativity and using local minimization. *Comptes rendus de l'Académie des Sciences de Paris*, 346:687–690, 2008.

[149] S. ZHANG : *Multi–level Iterative Techniques*. Thèse de doctorat, Pennsylvania State University, 1988.

[150] S. ZHANG : Successive subdivision of tetrahedra and multigrid methods on tetrahedral meshes. *Houston Journal of Mathematics*, 21(3):541–556, 1995.

[151] Z.-R. ZHANG et T. TANG : An adaptive mesh redistribution algorithm for convection–dominated problems. *Communications on Pure and Applied Analysis*, 1:341–357, 2002.

Titre : Développement du parallélisme des méthodes numériques adaptatives pour un code indusctriel de simulation en mécanique des fluides.

Résumé : Les méthodes numériques adaptatives constituent un outil de choix pour assurer la pertinence et l'efficacité de la résolution numérique d'équations aux dérivées partielles. Le travail présenté dans ce mémoire porte sur la conception, l'implémentation, et la validation d'une telle méthode au sein d'une plate-forme industrielle de simulation en thermohydraulique. Du point de vue géométrique, la méthode proposée permet de prendre en compte tant le raffinement du maillage que son déraffinement, tout en garantissant la qualité des éléments qui le compose. Du point de vue numérique, nous utilisons le formalisme des éléments joints pour étendre la méthode des Volumes-Éléments Finis proposée par la plate-forme Trio-U et traiter convenablement les maillages non-conformes générés par la procédure d'adaptation. Enfin, l'implémentation proposée repose sur les concepts des méthodes de décomposition de domaine, afin d'en garantir le bon comportement dans un contexte d'exécution parallèle.

Mots-clefs : Adaptation dynamique de maillage, méthodes de décomposition de domaine, éléments joints, parallélisme, équilibrage de charge, mécanique des fluides.

Title : Development of parallel implementation of adaptive numerical methods with industrial applications in fluid mechanics.

Abstract : Numerical resolution of partial differential equations can be made reliable and efficient throught the use of adaptive numerical methods. We present here the work we have done for the design, the implementation and the validation of such a method within an industrial software platform with applications in thermohydraulics. From the geometric point of vue, this method can deal both with mesh refinement and mesh coarsening, while ensuring the quality of the mesh cells. Numerically, we use the mortar elements formalism in order to extend the Finite Volumes-Elements method implemented in the Trio-U platform and to deal with the non-conforming meshes arising from the adaptation procedure. Finally, we present an implementation of this method using concepts from domain decomposition methods for ensuring its efficiency while running in a parallel execution context.

Keywords : Adaptive mesh refinement, domain decomposition methods, Mortar method, parallelism, dynamic load balancing, fluid mechanics.