Vers la simulation des écoulements sanguins

Vincent Chabannes

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques Appliquées**

Arrêté ministérial : du 7 août 2006

Présentée par

Vincent Chabannes

Thèse dirigée par **Christophe Prud'homme**

préparée au sein **du Laboratoire Jean Kuntzmann**
et de **l'École doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Vers la simulation des écoulements sanguins

Thèse soutenue publiquement le 8 juillet 2013 ,
devant le jury composé de :

M. Jean-Frédéric Gerbeau
Directeur de Recherche, INRIA Paris-Rocquencourt, Rapporteur
M. Luca Formaggia
Professeur, École polytechnique de Milan, Rapporteur
M. Emmanuel Maitre
Professeur, Grenoble Institute of Technology, Examinateur
M. Frédéric Hecht
Professeur, Université Pierre et Marie, Examinateur
M. Christophe Prud'homme
Professeur, Université de Strasbourg, Directeur de thèse
M. Mourad Ismail
Maître de Conférences, Université Joseph Fourier, Co-Directeur de thèse

Rien ne nourrit mieux l'inspiration que l'amour.
La souffrance peut la stimuler, rarement l'épanouir.

Isabelle JARRY

Remerciements

Je voudrais tout d'abord remercier Christophe, mon directeur de thèse, de m'avoir donné l'opportunité de réaliser ce captivant sujet de thèse. Ça a été un réel plaisir d'être encadré par cette personne passionnée par son travail. Tous les savoir-faire et les connaissances qu'il a pu me transmettre ont épanoui ma culture scientifique. Merci encore pour la confiance et la liberté laissées dans mon travail de recherche et aussi pour les efforts de communication qui ont été faits malgré la distance nous séparant.

Un grand merci à Mourad qui m'a également encadré dans ce projet. Nos discussions m'ont souvent permis d'avancer dans mes travaux en mettant en lumière certains détails qui me semblaient obscurs. Les encouragements qu'il m'a exprimés tout au long de mon doctorat ont été appréciables et motivants pour l'avancement de ce projet.

Je suis très honoré que Jean-Frédéric Gerbeau et Luca Formaggia aient accepté d'être les rapporteurs de ma thèse. Je tiens à les remercier et à exprimer ma gratitude pour leur lecture approfondie de mon manuscrit. Merci également à Emmanuel Maitre et Frédéric Hecht d'avoir pris le temps de lire cet ouvrage.

Je tiens à remercier Gonçalo et Silvia pour les précieuses contributions qu'ils ont apportées dans ce projet. Cette collaboration fructueuse m'a permis d'avancer dans mes travaux de recherche.

Merci également à Stéphane, Abdoulaye, Vincent, Ranine, Céline et Pierre que j'ai côtoyés avec un réel plaisir dans mon travail, mais également hors de ce contexte. J'espère que notre amitié continuera au delà de cette thèse. Merci aussi aux autres personnes du bureau 64 (Aymen, Quentin, Albane et Affaf) avec qui j'ai pu passer de très bons moments au cours de ces années.

Je tiens aussi à adresser mes remerciements à tous les membres de l'équipe EDP dont j'ai fait parti pendant cette thèse. Je remercie aussi Catherine, Juana et Hélene pour leur gentillesse et les nombreux services administratifs qu'elles m'ont rendus.

Je n'oublie pas bien sûr de remercier mes amis de longue date avec qui j'ai pu me mettre au vert pour décompresser de ce long travail. Je parle évidemment de Julien, Nicolas, Mathieu, Jimmy, Guillaume, et de tous les autres. Aussi, un grand merci à Jean-Yves pour les bonnes soirées passées aux couleurs de la Belgique.

Enfin, je tiens à adresser mes plus sincères remerciements à la personne qui m'a toujours soutenu, Ingrid, et qui ne cesse d'épanouir ma vie. Je lui dois tant pour tous les soins, l'attention et l'amour qu'elle m'a donnés lors des ces deux accidents de vie que j'ai malheureusement rencontrés. Je la remercie infiniment pour toute cette force et ce courage qu'elle a mis en oeuvre pour me redonner le moral dans les moments difficiles. Elle a également accompli la lourde tâche de relire ce manuscrit bien qu'elle ne soit pas issue du monde des sciences, encore merci !

Table des matières

Notations

Géométrie et maillage

d	:	la dimension géométrique $d = 1$, 2 ou 3
Ω	:	un domaine borné quelconque de $\mathbb{R}^d$
$\boldsymbol{n}$	:	un vecteur normal unitaire de $\partial\Omega$ orienté vers l'extérieur
δ	:	un couple de paramètre de discrétisation géométrique $\delta \equiv (h, k)$
$\mathcal{T}_\delta$	:	un maillage de taille caractéristique h constitué d'éléments d'ordre k
N_{elt}	:	nombre d'élément contenu dans $\mathcal{T}_\delta$
Ω_δ	:	le domaine recouvert par la discrétisation $\mathcal{T}_\delta$
K	:	un élément de $\mathcal{T}_\delta$
K_e	:	l'élément de $\mathcal{T}_\delta$ d'indice e
$\hat{K}$	:	un élément de référence (de même type que K)
$\boldsymbol{\varphi}_K^{\text{geo}}$	:	la transformation géométrique de $\hat{K}$ vers K
$\boldsymbol{\varphi}_{K,k}^{\text{geo}}$	:	la transformation géométrique d'ordre k de $\hat{K}$ vers K

Espaces fonctionnels

$\mathbb{P}_N(K)$	:	l'espace vectoriel des polynômes de degrès total inférieur ou égale à N sur K
$\mathbb{Q}_N(K)$	:	l'espace vectoriel des polynômes de degrès inférieur ou égale à N sur K
$\hat{\Phi}_i^N$	:	la i^{me} fonctions de base de l'élément fini de Lagrange $(\hat{K}, \hat{\mathbb{P}}_N, \hat{\Sigma})$
$\Phi_{K,i}^N$	:	la i^{me} fonctions de base de l'élément fini de Lagrange $(K, \mathbb{P}_N, \Sigma_K)$
Φ_i^N	:	la i^{me} fonctions de base de l'espace $P_c^N(\Omega_\delta)$
$L^2(\Omega)$	:	l'espace des fonctions qui sont de carré intégrable au sens de l'intégrale de Lebesgue
$D^\alpha v$	:	la dérivée partielle généralisé de v au sens au sens des distributions
$H^m(\Omega)$	:	$\{v \in L^2(\Omega_\delta) \mid D^\alpha v \in L^2(\Omega_\delta) \quad \forall \alpha \in \mathbb{N}^d \text{ tel que } \lvert\alpha\rvert \leq m\}$
$H_g^m(\Omega)$	:	$\{v \in H^m(\Omega), v_{\lvert\partial\Omega} = g\}$
$H_{g,\Gamma}^m(\Omega)$	:	$\{v \in H^m(\Omega), v_{\lvert\Gamma} = g\}$
$\mathbf{H}_{\boldsymbol{g}}^m(\Omega)$	:	$\{\boldsymbol{v} \in [H^m(\Omega)]^d, \boldsymbol{v}_{\lvert\partial\Omega} = \boldsymbol{g}\}$
$\mathbf{H}_{\boldsymbol{g},\Gamma}^m(\Omega)$	:	$\{\boldsymbol{v} \in [H^m(\Omega)]^d, \boldsymbol{v}_{\lvert\Gamma} = \boldsymbol{g}\}$
$P_c^N(\Omega_\delta)$	:	$\{v \in C^0(\bar{\Omega}_\delta) \mid \ v \circ \boldsymbol{\varphi}_K^{\text{geo}} \in \mathbb{P}^N(\hat{K}) \ \forall K \in \mathcal{T}_\delta\}$
$P_d^N(\Omega_\delta)$	:	$\{v \in L^2(\Omega_\delta) \mid \ v \circ \boldsymbol{\varphi}_K^{\text{geo}} \in \mathbb{P}^N(\hat{K}) \ \forall K \in \mathcal{T}_\delta\}$

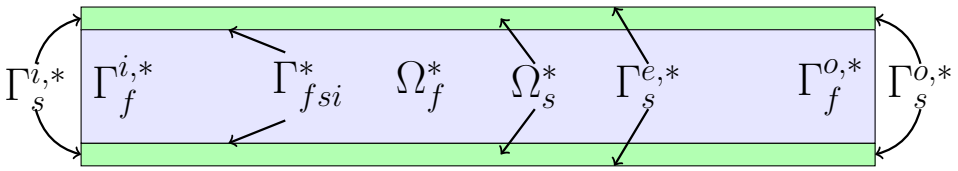
Transformation ALE

Ω^*	:	domaine de référence
$\mathbf{x}^*$	:	coordonnée d'un point dans le repère de référence
$\mathcal{A}^t$	:	la carte ALE continue au temps t
$\mathcal{A}^t_\delta$	:	la carte ALE discrète au temps t d'ordre k
$\boldsymbol{\zeta}$	:	le déplacement du maillage
$\boldsymbol{\zeta}_{(h,k)}$	:	le déplacement discret du maillage

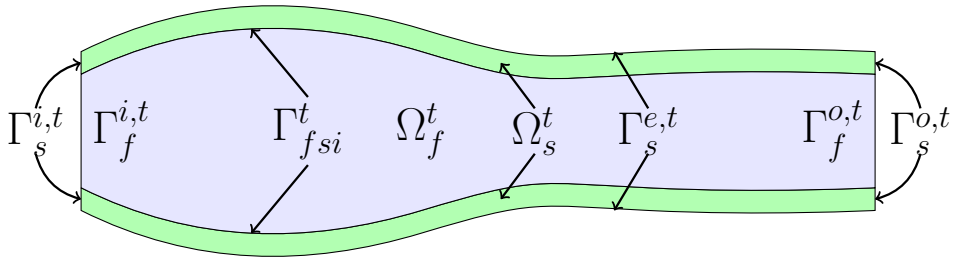
Méthode de la frontière élargie

Ω	:	le domaine global
ω	:	le domaine local
B	:	le domaine perforation
γ	:	la frontière autour de la perforation
γ'	:	la frontière fictive
u	:	la solution globale
$\hat{u}$	:	la solution locale
$\mathring{u}$	:	la solution perforation

Géométrie pour les écoulements sanguins



(a) Domaines de référence



(b) Domaines déformés

FIGURE 1 – Description des notations géométriques pour un problème FSI standard d'écoulement sanguin.

Notations pour le fluide

Ω_f^*	:	le domaine fluide de référence pour le repère ALE
Ω_f^t	:	le domaine fluide déformé à l'instant t
$\boldsymbol{u}_f$	:	le vecteur vitesse
p_f	:	la pression
$\boldsymbol{\sigma}_f$	:	le tenseur des contraintes
ρ_f	:	la masse volumique
μ_f	:	la viscosité dynamique

Notations pour la structure

Ω_s^*	:	le domaine structure de référence pour le repère lagrangien
Ω_s^t	:	le domaine structure déformé à l'instant t
$\boldsymbol{\eta}_s$	:	le vecteur déplacement
p_s	:	la pression
$\boldsymbol{F}_s$	:	le gradient de déformation
$\boldsymbol{\Sigma}_s$	:	le second tenseur de Piola-Kirchhoff
ρ_s^*	:	la masse volumique
E_s	:	le module de Young
ν_s	:	le coefficient de Poisson
λ_s	:	le premier coefficient de Lamé
μ_s	:	le second coefficient de Lamé (aussi appelé module de cisaillement)

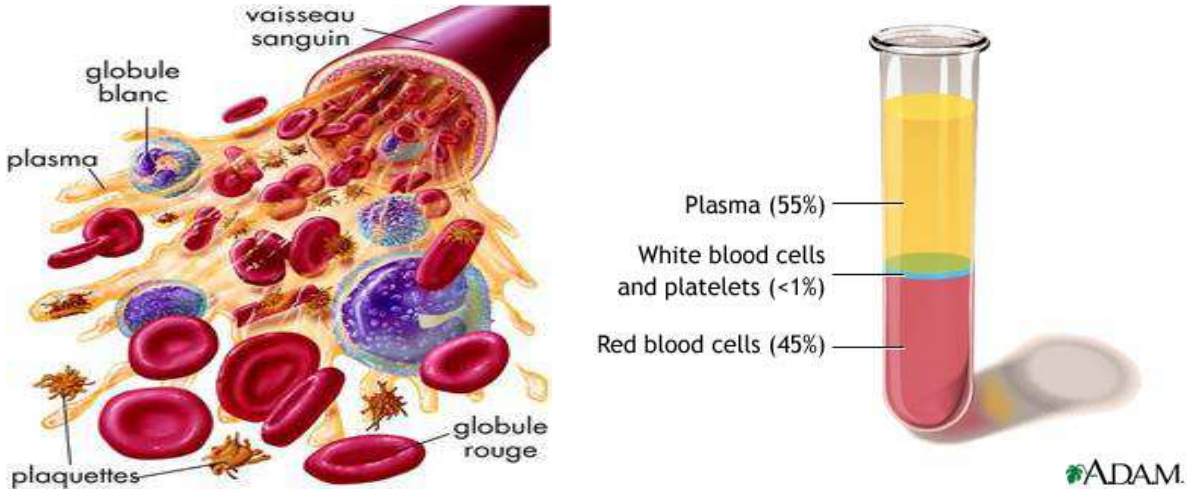
Introduction

La simulation des écoulements sanguins est un domaine en plein essor depuis ces dernières années. De nombreux groupes de recherche se sont impliqués dans cette étude qui vise à modéliser ces écoulements considérés comme fluides complexes. En effet, contrairement aux liquides ordinaires, les fluides complexes comme le sang exhibent des comportements étranges qui dépendent essentiellement des structures sous-jacentes qui les composent. Le sang est composé d'entités microscopiques (principalement de globules rouges, globules blancs et plaquettes) qui sont en suspension dans le plasma sanguin considéré comme un fluide ordinaire. Les comportements individuels et collectifs de ces entités ont alors un fort impact sur les propriétés hémodynamiques du sang à l'échelle globale. C'est cette rétroaction du micro à la macro échelle qui confère aux fluides complexes des comportements non triviaux et qui continue de poser un formidable défi pour les modélisations théoriques et numériques axès sur la rhéologie sanguine. Cette notion de rhéologie a été introduite par Eugene Bingham en 1920 pour désigner l'étude de la déformation et de l'écoulement de la matière. Elle a alors fait l'objet de nombreuses études théoriques dans le passé et grâce à l'arrivée et à l'augmentation croissante de la puissance des ordinateurs, la simulation numérique de ces phénomènes complexes a pu être envisagée. Les domaines d'expertise nécessaires à l'élaboration d'un tel projet vont de la médecine aux mathématiques en passant par la science du numérique et de l'informatique. L'objectif final est de développer des méthodes et des outils de simulation pour la communauté médicale, permettant de mener des expérimentations *in silico* et d'apporter des informations complémentaires, voire impossibles à obtenir, sur des patients. Cela pourrait être un enjeu important pour le traitement de certaines pathologies sanguines comme les sténoses artérielles, les anévrismes, l'athérosclérose ou les traitements thérapeutiques comme les stents endovasculaires ou les valves aortiques artificielles. Nous proposons dans cette thèse une contribution à ce projet qui sera majoritairement centré sur l'aspect numérique et informatique. Nous essaierons aussi de montrer dans chaque partie les intérêts médicaux et la rigueur mathématique qui sont nécessaires à l'élaboration de ce projet. Cependant, les résultats que nous obtiendrons n'auront pas la prétention de modéliser des applications vraiment réalistes ou physiologiques d'écoulements sanguins. Le but principal de cette thèse est de proposer des outils dans la direction de la rhéologie sanguine.

Hémodynamique

L'hémodynamique, la science des propriétés physiques du flux sanguin, représente notre point de départ pour cette étude sur la simulation des écoulements sanguins. La description qui va suivre a été faite dans de nombreux ouvrages médicaux dont notamment les suivants [65, 1, 111, 109] que nous avons utilisés. Le système cardiovasculaire est un

circuit fermé parcourant l'ensemble du corps humain et dans lequel circule le sang (circulation systémique et circulation pulmonaire). Cet écoulement prend en charge le transport de l'oxygène et des substances nutritives vers les milliards de cellules que contient un corps humain. Cet apport permet ainsi à ces cellules de « vivre ». Le sang récupère aussi tous les déchets des cellules qui sont ensuite éliminés par les reins, le foie et les poumons. La circulation du sang participe également au contrôle de la température corporelle ainsi qu'à la régulation du volume de certains liquides dans les tissus. De plus, le sang charrie les globules blancs qui ont pour rôle d'assurer une défense de notre organisme contre les virus et les microbes. Trois types de vaisseaux assurent le transport du sang : les artères, les capillaires et les veines. On parle également d'artériole et de veinule pour désigner les minuscules vaisseaux qui permettent le raccordement entre les capillaires et les artères ou les veines. Les contractions rythmiques du coeur (agissant comme une pompe) propulsent le sang dans les artères et circulent dans toutes les régions de l'organisme. Le sang arrive ensuite dans les artérioles puis dans les capillaires qui sont de minuscules vaisseaux. C'est ici que se produisent les échanges entre le sang et les cellules grâce à leur paroi extrêmement fine. Finalement, le sang est réacheminé vers le coeur par les veinules et les veines. D'après certains ouvrages, la longueur totale du vaste réseau formé par les vaisseaux sanguins serait estimée à 150000 km. On peut noter que l'aorte est la plus grosse des artères et les deux veines caves sont les plus grosses veines.



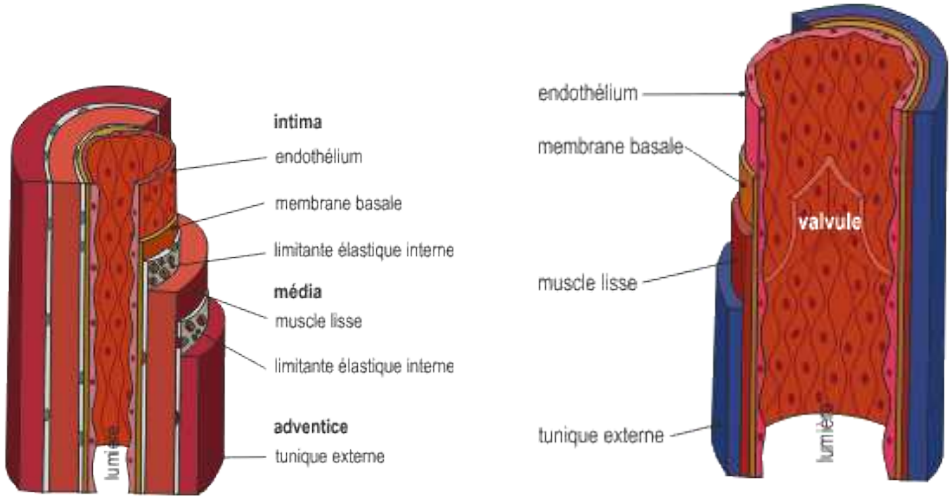
(a) Vue d'artiste (image issue de http ://sciencejunior.fr/).

(b) Composition quantitative (image issue de http ://www.pennmedicine.org).

FIGURE 2 – Schémas représentant la composition du sang.

Comme l'illustre la figure 2, le sang considéré comme un fluide complexe se compose d'un élément liquide, le plasma, dans lequel baignent les cellules sanguines : les globules rouges, les globules blancs et les plaquettes. En moyenne, le plasma représente 55% du volume total du sang, les globules rouges 45%, les globules blancs et les plaquettes moins de 1%. Le plasma est un liquide visqueux de couleur jaunâtre et qui est constitué à 90% d'eau. Les globules rouges (ou érythrocytes) transportent l'oxygène des poumons vers les cellules, et le gaz carbonique des cellules vers les poumons. Les globules blancs (ou leucocytes) ont la capacité de franchir la paroi des vaisseaux sanguins et de pénétrer dans les tissus pour défendre l'organisme contre les agents pathogènes. Les plaquettes sont des fragments de cellules, capables de s'agréger entre elles. Elles sont en partie responsables

de la coagulation du sang et de la formation des croûtes à la surface de la peau en cas de blessure.



(a) Artère (b) Veine

FIGURE 3 – Schéma d'une artère et d'une veine. (image issue de http ://www.fedecardio.org/).

La paroi des vaisseaux sanguins (artères, capillaires et veines) est composée de trois tuniques (ou couches) : l'intima, la média et l'adventice. On distingue trois constituants principaux qui sont : les fibres d'élastine, les fibres de collagène et les fibres musculaires lisses. Une description détaillée des constituants des parois vasculaires (artères et veines) est donnée dans les schémas de la figure 3. Les propriétés élastiques des différentes couches constituant les vaisseaux dépendent principalement du rapport entre leur quantité de fibre d'élastine et de fibre de collagène. Ce rapport varie avec le diamètre du vaisseau. Un autre facteur à prendre en compte dans le comportement mécanique est l'organisation tridimensionnelle complexe de ces fibres dans le tissu vasculaire. Chaque type de vaisseau est ainsi conçu de telle manière à pouvoir supporter la pression du flux sanguin en se déformant en conséquence. Une des propriétés fondamentales des grosses artères élastiques, et notamment de l'aorte, est de pouvoir amortir les importantes élévations de pression lors de la période de contraction cardiaque (systole ventriculaire) puis le retour élastique de cette même paroi pendant la période de repos cardiaque (diastole ventriculaire). Cette seconde phase permet de conserver dans le réseau artériel une pression minimale (ou pression diastolique). Ainsi, à la sortie du coeur, le débit sanguin pulsé tend à devenir continu lorsque l'on s'éloigne du coeur. On appelle parfois ce phénomène l'effet Windkessel.

Modèles physiques pour les écoulements sanguins

Les propriétés des écoulements sanguins sont affectées par un certain nombre de facteurs, dont l'hématocrite (pourcentage du volume des globules rouges par rapport au volume sanguin total), la viscosité du plasma, le regroupement et les déformations des

globules rouges. Comme le plasma est composé principalement d'eau, il peut-être considéré comme un fluide newtonien incompressible. Cependant, le sang qui est composé des cellules en suspension dans le plasma révèle un comportement complexe qualifié de fluide non newtonien. En particulier, la viscosité apparente du sang varie en fonction du taux de cisaillement. On peut ainsi remarquer que plus le taux d'hématocrite est élevé et plus le sang devient visqueux. En effet, pour les faibles taux de cisaillement, la viscosité du sang est grande. Ce phénomène est fortement lié à l'agrégation des globules rouges sous forme de microstructures appelées rouleaux et à la forte déformation qu'ils peuvent subir pour pouvoir circuler dans les micros vaisseaux (forme de « parachute »). Lorsque la vitesse de l'écoulement augmente (taux de cisaillement augmente), le regroupement des globules rouges se dissocie et entraine alors une diminution de la viscosité. Dans ce cas, c'est principalement l'hématocrite qui détermine la viscosité du sang. Ces comportements complexes du sang caractérisent ce fluide comme rhéofluidifiant. On peut également introduire la thixotropie et la viscoélasticité de ce fluide complexe. Néanmoins, des simplifications de modèles peuvent être envisagées lorsque le taux de cisaillement est élevé. Le sang peut être alors supposé newtonien avec une viscosité constante de l'ordre de 4-5 mPa.s. C'est typiquement le cas dans les larges vaisseaux sanguins comme l'aorte. Toutefois, pour la modélisation de certaines pathologies (ex : anévrisme), cette hypothèse pourrait être remise en cause. Cette description physique des écoulements sanguins est tirée des références suivantes [47, 136, 153].

Le modèle le plus simple qui peut être utilisé pour la modélisation des écoulements sanguins est le modèle newtonien. Il est assez bien adapté aux larges vaisseaux ou si nous ne sommes pas intéressés par des détails fins du fluide (comportements rhéofluidifiants et viscoélastiques). Les équations gouvernant ce modèle sont les équations classiques de Navier-Stokes incompressible. Elles sont décrites par le système suivant d'équations aux dérivées partielles :

$$\rho_f \frac{\partial \boldsymbol{u}_f}{\partial t} + \rho_f \left(\boldsymbol{u}_f \cdot \nabla\right) \boldsymbol{u}_f - \nabla \cdot \boldsymbol{\sigma}_f = \boldsymbol{f}_f$$
$$\nabla \cdot \boldsymbol{u}_f = 0$$

Les efforts internes du fluide sont représentés par le tenseur des contraintes $\boldsymbol{\sigma}_f$ défini par l'expression suivante :

$$\boldsymbol{\sigma}_f = -p_f \boldsymbol{I} + 2\mu_f \boldsymbol{D}\left(\boldsymbol{u}_f\right)$$

Dans cette dernière relation, nous faisons apparaitre la viscosité dynamique μ_f supposée constante pour le cas newtonien et le tenseur des déformations $\boldsymbol{D}\left(\boldsymbol{u}_f\right)$ choisi par la forme suivante :

$$\boldsymbol{D}\left(\boldsymbol{u}_f\right) = \frac{1}{2}\left(\nabla \boldsymbol{u}_f + \left(\nabla \boldsymbol{u}_f\right)^T\right)$$

Nous détaillerons ce modèle plus en détail dans le chapitre 4. De plus, nous le transformerons légèrement de manière à prendre en compte le possible mouvement du vaisseau.

Pour la modélisation du comportement rhéofluidifiant des écoulements sanguins, un modèle plus complexe est nécessaire. La viscosité μ_f n'est plus constante, mais dépend du tenseur des cisaillements $\boldsymbol{D}\left(\boldsymbol{u}_f\right)$. Le modèle dit non-newtonien est basé sur les mêmes équations énoncées précédemment, sauf que la viscosité doit être exprimée à l'aide d'une loi de comportement spécifique à l'application en fonction du taux de cisaillement. Cela ajoute une forte non-linéarité pour ces modèles non-newtoniens. Nous donnerons quelques exemples de ces types de modèle au chapitre 4.

Dans le système cardiovasculaire, les parois artérielles ont une structure assez similaire, mais la composition et l'épaisseur de celles-ci varient en fonction de leur localisation. Ces facteurs jouent un rôle important dans le comportement mécanique de ces vaisseaux. En effet, le diamètre des artères décroit au fur et à mesure que l'on s'éloigne du coeur. Les artères, en fonction de leur localisation, sont alors classées en trois groupes : les artères élastiques, les artères musculaires et les artérioles. Les artères élastiques se trouvent proches du coeur. Elles font parties des vaisseaux de grand diamètre et elles ont une paroi épaisse. C'est le cas par exemple de l'aorte. Ces types d'artères jouent un rôle important dans l'amortissement de la pression et du flux sanguin pulsé lors de la contraction cardiaque (systole). Elles assurent également la conservation d'une pression minimale dans le réseau artériel. Ce contrôle est réalisé grâce à l'élasticité de la paroi vasculaire des artères élastiques. D'après [101], les déformations peuvent être assez importantes (plus de 10% du rayon), c'est pourquoi les modèles de grande déformation, dits modèles hyperélastiques, sont privilégiés pour ce type de vaisseau. Les artères musculaires sont de taille moyenne et elles sont moins extensibles que les artères élastiques à cause de la diminution des fibres d'élastines. Elles jouent un rôle plus actif dans la vasoconstriction. Les grandes déformations pour ces types d'artères ne sont pas forcément nécessaires. D'autres propriétés mécaniques peuvent être prises en compte comme l'anisotropie et la viscoélasticité qui sont dues à la composition complexe de ces vaisseaux. Toutefois, nous négligerons ces aspects dans cette thèse. On considère aussi que les parois vasculaires sont incompressibles (ou quasi-incompressibles) [101, 100, 88].

Pour pouvoir décrire un modèle élastique en grande déformation, appelé généralement hyperélastique, il nous faut définir une loi de comportement qui nous permettra d'exprimer le tenseur des contraintes du modèle $\boldsymbol{\Sigma}_s$. Cette loi de comportement sera généralement caractérisée par des coefficients qui seront propres aux matériaux modélisés. Dans le cas d'un matériau compressible, les équations gouvernants ce modèle peuvent s'exprimer par :

$$\rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \nabla \cdot (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) = \boldsymbol{f}_s$$

dans le repère lagrangien. La quasi-incompressibilité peut aussi être prise en compte directement dans la formulation de la loi de comportement. Pour exprimer un modèle hyperélastique incompressible, une équation de contrainte d'incompressibilité doit être rajoutée à l'équation précédente. Cette contrainte est imposée à l'aide d'un multiplicateur de Lagrange qui sera intégré dans l'expression du tenseur des contraintes. Tous les détails de ces modèles de mécanique des solides seront présentés au chapitre 5. On peut aussi noter qu'il est possible d'utiliser des modèles réduits de structure. Dans [59], nous pouvons trouver la description de modèles de coque (2D) et d'un modèle 1D appelé *generalised string model*. Généralement, ces modèles sont applicables uniquement sur des géométries idéalisées.

Le globule rouge est une cellule du sang qui sert à capter l'oxygène lors de son passage dans les poumons et à le redistribuer aux autres cellules. Il est également appelé érythrocyte ou hématies. Les globules rouges sont dépourvus de noyau et se présentent sous la forme d'une lentille biconcave. Cette forme caractéristique lui confère un maximum de surface, ce qui lui permet un maximum d'échanges avec les autres cellules. Le globule rouge est constitué d'une fine membrane qui sépare l'extérieur de la cellule de l'intérieur appelé cytoplasme. Cette membrane est constituée de deux couches de lipides. Le cytoplasme contient en majorité l'hémoglobine qui est considérée comme un fluide newtonien

incompressible [55]. Les propriétés mécaniques de ces cellules jouent un rôle important dans la rhéologie sanguine. Grâce à la structure du squelette membranaire, elles peuvent entre autres se déformer considérablement pour se faufiler dans les capillaires, dont le diamètre peut être inférieur au leur. Une particularité de la membrane est qu'elle est inextensible. La viscosité de l'hémoglobine participe aussi aux propriétés mécaniques de la membrane [55, 136].

Ainsi, pour modéliser la dynamique des globules rouges, nous pouvons considérer le couplage de deux modèles physiques. Nous avons un modèle d'élasticité en très grande déformation avec une contrainte d'incompressibilité pour la membrane. L'hémoglobine est décrite comme un fluide newtonien incompressible, donc les équations de Navier-Stokes incompressible semblent adéquates pour modéliser le cytoplasme. Enfin, il faut prendre en compte l'interaction entre la membrane et le cytoplasme à l'aide d'un modèle d'interaction fluide structure.

Choix des méthodes numériques

Pour ce projet, nous sommes motivés par l'utilisation de méthodes numériques dont la précision pourra être accrue de manière arbitraire. Nous voulons ainsi améliorer la précision des solutions numériques et de la discrétisation géométrique en utilisant les notions d'approximation polynomiale d'ordre élevé et d'élément géométrique d'ordre élevé. Ce critère va donc motiver le choix de nos méthodes utilisées pour la modélisation des écoulements sanguins.

Tout d'abord, pour la résolution des équations aux dérivées partielles décrivant les modèles physiques, nous nous appuierons sur la méthode des éléments finis [39, 35, 31, 53, 91]. Cette méthode a l'avantage d'avoir une base théorique solide contrairement aux deux autres méthodes classiques : la méthode des différences finies et la méthode des volumes finis. Elle a ainsi fait l'objet de nombreuses analyses mathématiques donnant lieu à des propriétés d'interpolation et des estimations d'erreurs optimales. D'autre part, la méthode de Galerkin, associée à la méthode éléments finis, nous fournit un cadre d'approximation général pour une large gamme d'approximation d'ordre arbitrairement élevé en espace et géométrie.

Pour la description des équations aux dérivées partielles dans des domaines mobiles, nous avons choisi d'utiliser la méthode ALE (arbitrairement lagrangien-eulérien). C'est un choix que l'on retrouve assez souvent dans la littérature et notamment pour la simulation des écoulements sanguins [59, 124, 123, 37, 66, 57, 82]. Ce choix permet de représenter les modèles fluides et structures dans leurs repères naturels. Les équations de la mécanique des solides sont décrites dans le repère dit lagrangien, c'est-à-dire qu'elles suivent le déplacement des particules au cours du temps. Au contraire de cette idée, la mécanique des fluides est plus facilement décrite dans le repère dit eulérien, où l'observateur est à une position fixe et regarde passer les particules. Ainsi, seul le domaine de calcul fluide est en mouvement, les calculs effectués sur la structure sont faits uniquement dans un domaine de référence fixe. À l'aide de l'utilisation de la méthode ALE dans le modèle fluide, l'écoulement pourra alors suivre le mouvement lagrangien de l'interface fluide structure tout en conservant l'intérêt du repère eulérien lorsque l'on « s'éloigne » de cette interface. Cette

vision est possible grâce à l'introduction d'une vitesse de déformation (voir chapitre 2) définie sur l'ensemble du domaine fluide. De plus, cette méthode a l'avantage d'être précise dans la description de la déformation du domaine d'étude. Dans le but d'accroitre encore cette précision, nous proposons dans le chapitre 2 une nouvelle construction de cette méthode pour la prise en compte de déplacements géométriques d'ordre élevé. L'intérêt de cette approche d'ordre arbitraire a été démontré dans [124] dans un cas bidimensionnel. Notre nouvelle approche fournira un cadre plus général pour des applications 2D et 3D.

La modélisation de l'interaction fluide structure (entre le sang et la paroi vasculaire) est réalisée en couplant les équations de la mécanique des fluides et de la mécanique des solides. En plus de la contrainte géométrique reliant le domaine fluide au domaine solide, des conditions de transmission doivent être imposées sur l'interface fluide structure. Nous utilisons les conditions standards pour ce type de problème avec les relations suivantes :

$$\begin{aligned} \boldsymbol{u}_f &= \boldsymbol{u}_s \\ \boldsymbol{\sigma}_f \boldsymbol{n}_f + \boldsymbol{\sigma}_s \boldsymbol{n}_s &= \mathbf{0} \end{aligned}$$

Ces conditions assurent la continuité du champ de vitesse et l'équilibre des contraintes normales appliqués sur l'interface fluide structure. Les méthodes numériques pour résoudre un problème d'interaction fluide structure peuvent être classées en deux catégories : les méthodes monolithiques [83, 66, 57] et les méthodes partitionnées [124, 123, 56, 99, 30, 148]. L'approche monolithique est connue pour être plus robuste et performante que l'approche partitionnée où les problèmes fluides et structures sont résolus séparément. Cependant les solveurs monolithiques sont moins flexibles, leur implémentation est plus complexe à mettre en oeuvre et la résolution des systèmes algébriques est plus difficile. Dans cette thèse, nous nous sommes limités à l'approche partitionnée qui consiste à utiliser une procédure itérative de type point fixe avec relaxation. Elle est détaillée au chapitre 6. Un algorithme populaire de ce point fixe est appelé formulation Dirichlet-Neumann. Le problème fluide est résolu en utilisant le champ de vitesse de la structure. Dans le problème structure, on impose la contrainte normale du fluide. La relaxation opérée dans la procédure itérative est basée sur une méthode d'accélération d'Aitken qui est un ingrédient important à la convergence de ces algorithmes, notamment pour les applications d'écoulements sanguins. La procédure basée sur un schéma implicite est la méthode la plus précise, car elle assure une prise en compte forte des conditions de couplage. Toutefois, nous avons également utilisé dans cette thèse une approche dite semi-implicite qui « relâche » la contrainte de liaison géométrique à l'interface fluide structure. On pourra alors montrer au chapitre 10 que cette simplification permet d'accélérer les algorithmes tout en gardant une précision raisonnable de la solution. Cependant, la stabilité numérique de cette méthode sera vérifiée sous certaines conditions [56, 9].

Pour la modélisation des particules en mouvement dans un fluide, nous avons choisi d'utiliser la méthode de la frontière élargie. Initialement [115, 86], la méthode de la frontière élargie est dédiée à la simulation directe des fluides contenant des particules fixes. Typiquement, on part d'un problème initialement posé dans un domaine perforé (celui occupé par le fluide dans le cadre d'une suspension de particules) pour le ramener d'une manière équivalente à un couple de deux nouveaux problèmes :

- le premier est défini sur un domaine global de forme simple (celui occupé par le fluide et les particules) permettant l'utilisation d'un maillage régulier et par conséquent l'utilisation des solveurs rapides.
- le second se trouve dans une couche mince entourant chacune des particules. Le but de ce problème local est de corriger l'erreur introduite par la résolution du problème global au voisinage des particules. Comme ces domaines locaux sont de taille réduite, ils pourraient être maillés avec une haute résolution sans avoir une augmentation importante du coût de calcul.

De par sa conception, la méthode de la frontière élargie s'adapte facilement au cas des particules en mouvement. Dans ce cadre, les maillages des domaines locaux sont déplacés pour suivre les particules dans leurs mouvements. Le maillage global est bien évidemment fixe indépendamment des déplacements de ces dernières. Cela nous permet d'éviter les stratégies complexes de maillage adaptatif pour suivre le déplacement des particules (et le coût de calcul inhérent). De plus, l'avantage de la méthode de la frontière élargie comparée aux méthodes de domaines fictifs [70, 71, 72] est qu'elle a été conçue dans le but de récupérer la convergence optimale (au sens de l'interpolation éléments finis) à tous les ordres, en dépit de la non-conformité de ces maillages. Le comportement optimal de cette méthode a été prouvé théoriquement dans [26]. Nous proposerons au chapitre 3 une nouvelle formulation de cette méthode qui est mieux adaptée aux modèles de mécaniques des fluides. Nous étendrons ce travail au chapitre 6 pour le cas de l'interaction d'un fluide avec des particules mobiles et déformables.

Feel++ : un outil pour le calcul scientifique

Pour réaliser nos simulations d'écoulements sanguins à l'aide des méthodes numériques que nous venons de présenter, nous allons nous baser sur une librairie informatique C++ spécialisée dans le calcul scientifique. Elle se nomme Feel++, *Finite Element Embedded Language in* C++, et elle a fait l'objet de plusieurs publications [128, 135, 131, 129]. Feel++ fournit une interface C++ pour résoudre des systèmes d'équations aux dérivées partielles (EDP) en 1D, 2D et 3D au moyen de la méthode éléments finis combinée aux méthodes d'approximation de Galerkin. Cette librairie vise à rassembler la communauté scientifique pour la mise en oeuvre de méthodes numériques avancées et le calcul haute performance.

Le développement de Feel++ repose sur le paradigme des DSEL (*domain specific embedded language*). En d'autres mots, cet outil propose un langage embarqué dans le C++ pour la résolution d'EDP en restant très proche des mathématiques. Le principe fondamental de ces langages est de rendre transparentes la complexité et la puissance des algorithmes informatiques tout en gardant une large flexibilité et une vision claire des mathématiques. Ainsi, plus les mathématiques sont claires au niveau du numérique et plus il est facile et fiable de développer des méthodes numériques de complexité supérieure.

Le DSEL de Feel++ permet l'implémentation de nombreuses formulations variationnelles dans un langage très similaire à ce domaine. Il propose de manière simple et transparente une large variété de méthode d'approximation d'ordre arbitraire incluant l'ordre élevé géométrique, les formules de quadrature d'ordre élevé et des ensembles de points d'interpolations robustes pour l'ordre élevé. La souplesse des outils mis à la disposition de l'utilisateur permet de mettre en oeuvre des méthodes numériques qui couvrent

une grande combinaison de choix possibles (maillages, espaces de fonctions, points de quadrature) en utilisant le même langage. Feel++ dispose aussi d'une interface avec la librairie PETSc [13, 12, 14, 146] pour la résolution des systèmes algébriques. Combinée à cette puissante librairie de calcul, Feel++ offre un important choix de solveurs et de préconditionneurs qui peuvent être paramétrés directement en ligne de commande.

Au niveau de la programmation informatique, Feel++ utilise le langage C++ avancé (métaprogrammation par *templates*) et en particulier la dernière norme de ce langage programmation, le C++ 11, qui apporte des compléments très utiles tels que l'inférence des types. Ainsi, les nouveaux mots clés comme `auto` seront utilisés dans les exemples présentés dans cette thèse. Feel++ utilise également plusieurs parties de la librairie C++ Boost [28]. Nos exemples de script utiliseront aussi les concepts proposés par la librairie Boost.Parameter [2]. Elle permet d'augmenter la puissance du DSEL en donnant lieu à une nomination des arguments passés aux fonctions. Les arguments peuvent aussi être optionnels et il est possible de les donner dans un ordre quelconque.

Pour illustrer ces propos, nous présentons l'exemple le plus connu dans les mathématiques appliquées, le fameux laplacien. Le script suivant est le code C++ complet permettant de résoudre un laplacien 2D dans un carré unité avec une approximation P3.

```
#include <feel/feel.hpp>

int main(int argc, char**argv )
{
    using namespace Feel;
    Environment env(_argc=argc,_argv=argv);
    // definition du maillage et espace de fonction
    auto mesh = unitSquare();
    auto Vh = Pch<3>(mesh);
    auto u = Vh->element();
    auto v = Vh->element();
    // forme lineaire
    auto l = form1(_test=Vh);
    l = integrate(_range=elements(mesh),
                  _expr=id(v));
    // forme bilineaire
    auto a = form2(_trial=Vh,_test=Vh);
    a = integrate(_range=elements(mesh),
                  _expr=gradt(u)*trans(grad(v)) );
    a+=on(_range=boundaryfaces(mesh),_rhs=l,_element=u,
          _expr=cst(0.) );
    // resolution du systeme algebrique
    a.solve(_rhs=l,_solution=u);
    // exportation des resultats
    auto e = exporter(_mesh=mesh,_name="laplacian" );
    e->add( "u", u ); e->save();
    return 0;
}
```

Si l'on remplaçait `unitSquare()` par `unitCube()`, alors ce code correspondrait à un laplacien 3D dans un cube.

Au commencement de cette thèse, un certain nombre d'outils nécessaires à nos modélisations n'était pas implémenté dans Feel++. C'est pourquoi une grande partie de mon travail a consisté au développement de ces nouvelles fonctionnalités. Il s'agit en particulier du transfert d'informations entre les modèles et l'assemblage des termes non standards de

la méthode de la frontière élargie (chapitre 1). La parallélisation de la librairie a aussi été une étape cruciale pour l'obtention de résultats 3D (chapitre 7). Nous avons également développé un environnement spécifique à l'interaction fluide structure. Il contient tous les modèles de mécaniques des fluides et des solides que nous avons implémentés, ainsi que des outils de couplages de ces modèles pour l'interaction fluide structure et la méthode de la frontière élargie. Cet important travail de développement a aussi été utilisé par d'autres utilisateurs pour des applications diverses (écoulements sanguins, dynamiques des globules rouges avec *level-set*, décompositions de domaine, convection naturelle, champs magnétiques intenses). Ceci a ensuite donné lieu à des publications : [131] pour la librairie Feel++, [38] pour l'interaction fluide structure, [46, 45] pour le solveur de mécanique des fluides et [142, 34, 44, 145] pour la partie calcul haute performance.

Plan de la thèse

Dans une première partie, nous présenterons les méthodes numériques utilisées pour la résolution de nos modèles physiques. Nous commencerons avec le chapitre 1 par décrire les fondements de la méthode éléments finis et les nouveaux outils numériques qui ont été développés pour l'interaction fluide structure (opérateur d'interpolation) et pour la méthode de la frontière élargie (assemblage non standard). Nous passerons ensuite à la notion de carte ALE au chapitre 2 qui nous sera utile pour la résolution d'équations aux dérivées partielles dans des domaines mobiles. Entre autres, nous proposerons une nouvelle construction pour l'ordre élevé géométrique. Nous terminerons cette partie avec le chapitre 3 qui constitue une étude de deux formulations de la méthode de la frontière élargie dans le cas où les particules sont fixes.

La seconde partie de cette thèse sera consacrée aux modèles d'interaction fluide structure que nous avons implémentés. Nous débuterons avec les équations de la mécanique des fluides en domaines mobiles dans le chapitre 4. Ce travail s'appuiera sur la carte ALE présentée au chapitre 2. Nous aurons ensuite avec le chapitre 5 une description de quelques modèles gouvernant la mécanique des solides. Nous finirons avec le chapitre 6 par une présentation du modèle complet d'interaction fluide structure. Nous parlerons également des différentes stratégies de résolution possibles pour ce problème. Nous trouverons aussi dans ce chapitre une description d'un modèle d'interaction fluide particules que nous avons développé à l'aide de la méthode de la frontière élargie (chapitre 3).

Nous nous intéresserons dans une troisième partie à des aspects plus informatiques en illustrant d'autres contributions apportées à la librairie Feel++. Tout d'abord, le chapitre 7 présentera la mise en oeuvre du parallélisme dans la librairie. Nous effectuerons les premiers tests de scalabilités de cette nouvelle fonctionnalité. Puis, avec le chapitre 8, nous montrerons quelques fonctionnalités plus mineures que nous avons implémentées. Il se terminera avec la présentation de l'environnement qui a été développé pour la simulation des fluides et des structures.

Enfin, nous finirons cette thèse avec une quatrième partie illustrant principalement des résultats de simulation numérique. Avec le chapitre 9, une validation de nos modèles fluides, structures et d'interaction fluide structure sera effectuée. Pour cela, nous réaliserons les « benchmarks » proposés dans [154] et nous comparerons nos calculs avec d'autres résultats trouvés dans la littérature. Finalement, le chapitre 10 présentera des simulations obtenues pour des applications « se rapprochant » de la modélisation réaliste des écoule-

ments sanguins. Des tests 2D et 3D seront d'abord effectués sur des géométries idéalisées. Puis nous montrerons des résultats de simulations numériques réalisés sur des géométries réalistes.

Publications

Articles acceptés

[131] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Gonçalo Pena. Feel++ : A Computational Framework for Galerkin Methods and Advanced Numerical Methods. ESAIM : Proc., 38 :429-455, December 2012.

[46] Vincent Doyeux, Vincent Chabannes, Christophe Prud'Homme, and Mourad Ismail. Simulation of vesicle using level set method solved by high order finite element. ESAIM : Proceedings, 38 :335–347, August 2012.

[38] Vincent Chabannes, Gonçalo Pena, and Christophe Prud'homme. High-order fluide-structure interaction in 2D and 3D application to blood flow in arteries. Journal of Computational and Applied Mathematics, 246(0) : 1-9, 2013.

[45] V. Doyeux, Y. Guyot, V. Chabannes, C. Prud'homme, and M. Ismail. Simulation of two-fluid flows using a finite element/level set method. application to bubbles and vesicle dynamics. Journal of Computational and Applied Mathematics, 246(0) : 251-259, 2013.

[142] Abdoulaye Samake, Vincent Chabannes, Christophe Picard, and Christophe Prud'Homme. Domain decomposition methods in Feel++. pp 8, Accepted in DD21 proceedings, 2012.

Articles soumis

[34] Céline Caldini Queiros, Vincent Chabannes, Mourad Ismail, Gonçalo Pena, Christophe Prud'Homme, Marcela SZOPOS, and Ranine Tarabay. Towards large-scale three-dimensional blood flow simulations in realistic geometries. pp. 17, Submitted to ESAIM : Proc, CEMRACS 2012, 2012-12.

Conférences

[133] Christophe Prud'Homme, Gonçalo Pena, and Vincent Chabannes. High order fluid structure interaction in 2D and 3D : Application to blood flow in arteries. In ACOMEN 11 - 5th International Conference on Advanced COmputational Methods in ENgineering, pages 1–10, Liège, Belgium, October 2011.

[132] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, Gonçalo Pena, Cécile Daversin, and Christophe Trophime. Advances in Feel++ : a domain specific embedded language in C++ for partial differential equations. In Eccomas'12 - European Congress on Computational Methods in Applied Sciences and Engineering, Vienna, Autriche, September 2012.

Première partie

Méthodes numériques

Chapitre 1

Méthodes de Galerkin d'ordre arbitraire

Dans ce chapitre, nous présentons les principaux ingrédients mathématiques et numériques que nous avons choisis pour réaliser des simulations numériques d'écoulements sanguins. Toutes nos simulations vont s'appuyer sur la méthode des éléments finis. Celle-ci repose sur la formulation variationnelle (ou faible) du problème initial et sur une méthode d'approximation interne. Les domaines d'étude sont discrétisés à l'aide de maillages et le temps est traité par différences finis. La méthode d'approximation utilisée est la méthode de Galerkin, qui permet d'approcher la solution de problèmes continus dont la formulation variationnelle abstraite est la suivante :

$$\begin{cases} \text{Chercher } u \in V \text{ tel que} \\ a(u,v) = f(v) \quad ,\forall v \in W \end{cases} \tag{1.1}$$

où V et W sont des espaces fonctionnels, a est une forme bilinéaire définie sur $V \times W$ et f est une forme linéaire définie sur W. On dit que V est l'espace solution et que W est l'espace test. Les éléments de W sont appelés des fonctions tests. Nous rappelons aussi que la librairie Feel++ permet de résoudre des équation aux dérivées partielles (EDP) à l'aide de la méthode des éléments finis. Elle propose également un langage très proche de la formulation variationnelle et une gamme très large de méthode d'approximation de type Galerkin.

Après ce rappel sur les outils numériques de base, nous parlerons ensuite des outils plus spécifiques qui ont été développés pour les modèles d'interaction fluide structure et pour la méthode de la frontière élargie. Plus précisément, nous décrirons comment prendre en compte l'interaction des différentes physiques ou mathématiques, où le couplage s'effectuera entre plusieurs maillages qui n'auront pas forcément de conformité entre eux. Nous parlerons ainsi de l'opérateur d'interpolation et du calcul des termes non standard dans le cadre des méthodes de Galerkin. Les principaux exemples seront basés sur des problèmes simples, mais ils mettront suffisamment en jeu les besoins requis par les modèles décrits dans les chapitres suivants. Nous présenterons également quelques codes Feel++ montrant l'effort fait pour le développement du langage proche des mathématiques pour ces nouvelles fonctionnalités. Nous terminerons ce chapitre avec des vérifications des outils qui ont été développés.

1 Approximations par éléments finis

Nous commençons cette partie par une description de la méthode des éléments finis. Nous présenterons la construction de cette méthode pour des degrés d'approximation polynomiale d'ordre arbitraire en espace, mais aussi pour des approximations géométriques d'ordre arbitraire. Cette partie étant principalement issue de la thèse [124], le lecteur trouvera beaucoup plus de détails dans cet ouvrage. Dans cette thèse, les entités élémentaires utilisées dans les maillages sont des simplexes, c'est-à-dire segments (1D), triangles (2D) ou tétraèdres (3D). Toutes les descriptions faites par la suite seront basées sur ces entités, mais l'utilisation des hypercubes (quadrangles (2D) ou hexaèdres (3D)) est possible avec un traitement très similaire. Une description complète peut être trouvée dans [35, 124, 91, 31, 39]. Nous avons choisi de présenter cette partie afin d'introduire le vocabulaire, les notions et les notations nécessaires pour la suite de cette thèse.

1.1 Notion d'élément fini

Dans le cadre des méthodes des éléments finis et des éléments spectraux, nous nous basons sur la définition de Ciarlet[39]. Ce formalisme introduit la notion d'éléments finis qui correspond à la brique élémentaire de l'interpolation des fonctions. Pour construire un élément fini, nous avons besoin de définir un triplet $(K,\ P,\ \Sigma\ )$, avec :

- $K \in \mathbb{R}^d$ un élément géométrique compact, connexe et d'intérieur non vide ($\overset{\circ}{K} \neq \varnothing$) dont la frontière est lipschitzienne
- P un espace vectoriel de dimension finie de fonction $p : K \to \mathbb{R}^\mu$ où μ est un entier positif.
- Σ un ensemble de T_N formes linéaires σ_i défini sur P et tel que l'application linéaire :

$$P \ni p \longmapsto \left(\sigma_1\left(p\right), ..., \sigma_{T_N}\left(p\right)\right)^T \in \mathbb{R}^{T_N}$$

  soit bijective. Les formes linéaires $(\sigma_1, ..., \sigma_{T_N})$ sont appelées les degrés de liberté de l'élément fini.

De nombreux éléments finis existent, nous pouvons citer par exemple les éléments de Lagrange, Hermite, Raviart-Thomas, Crouzeix-Raviart, Nedelec, ... Une description complète de ces éléments est faite dans [39, 53, 91]. L'élément fini le plus simple et le plus classique est l'élément fini de Lagrange décrit par (K,$\mathbb{P}_N$,Σ). C'est celui-ci que nous utiliserons pour la suite de cette description. L'ensemble des formes linéaires $\Sigma = \{\sigma_i, i = 1, ..., T_N\}$ est ensuite défini par :

$$\sigma_i : \mathbb{P}_N(K) \longrightarrow \mathbb{R} \tag{1.2}$$

$$p \longmapsto p\left(\boldsymbol{a}_i\right) \tag{1.3}$$

où les $\boldsymbol{a}_i$ sont des points d'interpolation sur K et $\mathbb{P}_N(K)$ est un espace vectoriel des polynômes de degré total inférieur ou égal à N. L'espace vectoriel est défini sur l'élément K et est à valeur dans $\mathbb{R}$. Pour construire l'espace vectoriel $\mathbb{P}_N(K)$, nous devons choisir une base de cet espace, dite base primale. Plusieurs bases existent selon le type de l'élément K, mais certaines bases sont plus intéressantes grâce à des propriétés de L^2-orthonormalité et de construction hiérarchique : moments (tensorisé et simplexe), Jacobi (tensorisé), Legendre (tensorisé), Dubiner (simplexe) et adapté au bord (tensorisé et simplexe)

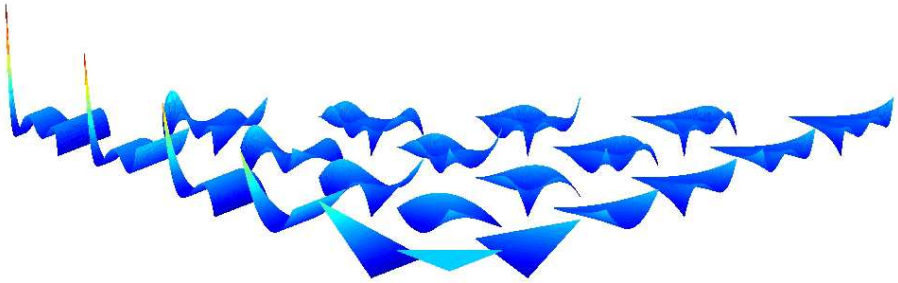


FIGURE 1.1 – Fonctions de base de Dubiner de degré 5 sur un triangle.

Definition 1. *Soit* $\left(\hat{K}, \hat{\mathbb{P}}_N, \hat{\Sigma}\right)$ *l'élément fini de Lagrange,* $\left\{\hat{\sigma}_i, i = 1, ..., \hat{T}_N\right\}$ *les formes linéaires associées à* $\hat{\Sigma}$ *et* $\hat{T}_N$ *la dimension de* $\hat{\mathbb{P}}_N(\hat{K})$*. On appelle fonctions de bases de l'élément fini* $\left(\hat{K}, \hat{\mathbb{P}}_N, \hat{\Sigma}\right)$*, les fonctions* $\left\{\hat{\Phi}_i^N \in \hat{\mathbb{P}}_N(\hat{K}), i = 1, ..., \hat{T}_N\right\}$ *telles que :*

$$\hat{\sigma}_j\left(\hat{\Phi}_i^N\right) = \delta_{ij}, \quad 1 \leq i, j \leq \hat{T}_N \tag{1.4}$$

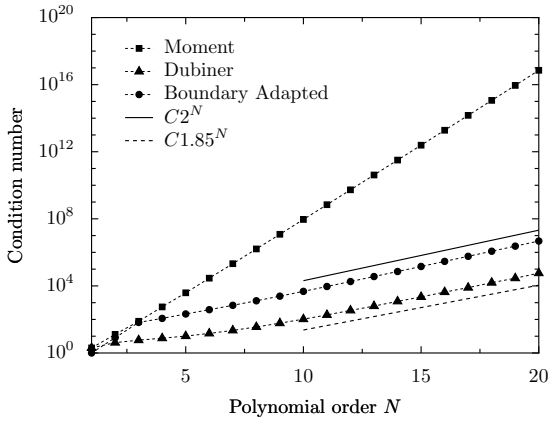
Avec le choix de notre construction élément fini, la base primale sera construite uniquement sur un élément de référence $\hat{K}$. On notera $\mathcal{B} = \{\psi_j\}_{j=1}^{\hat{T}_N}$ l'ensemble des fonctions de base de la base primale. Un exemple de fonctions de base primale $\mathcal{B}$ de type Dubiner sur un simplexe est illustré avec la figure 1.1. En reprenant l'exemple des éléments finis de Lagrange, chacune des fonctions de bases $\hat{\Phi}_i^N$ est déterminée par les $\hat{T}_N$ coefficients α_{ij} de telle sorte que :

$$\hat{\Phi}_i^N(x) = \sum_{j=1}^{\hat{T}_N} \alpha_{ij}\psi_j(x), \quad i = 1, ..., \hat{T}_N \tag{1.5}$$

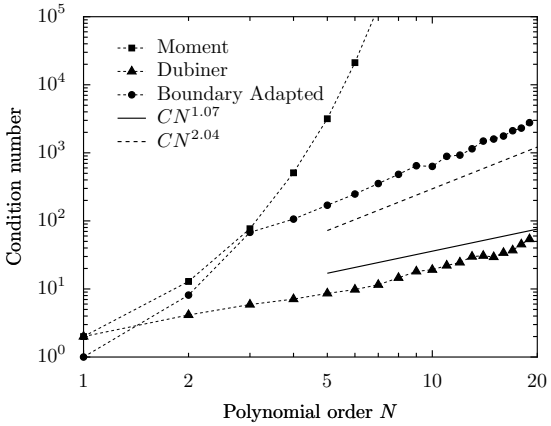
Le calcul des coefficients α_{ij} se ramène alors au choix des points d'interpolation que l'on doit définir sur l'élément $\hat{K}$. Il existe par exemple les points équidistribués, de Fekete et de Warp Blend. Ces derniers sont représentés avec l'exemple en figure 1.2(c). Une description de ces points d'interpolation est faite dans [150, 162, 124]. D'un point de vue algébrique, nous avons besoin de résoudre le système linéaire suivant :

$$\begin{bmatrix} \alpha_{11} & \dots & \alpha_{1\hat{T}_N} \\ \vdots & & \vdots \\ \alpha_{\hat{T}_N 1} & \dots & \alpha_{\hat{T}_N \hat{T}_N} \end{bmatrix} \underbrace{\begin{bmatrix} \psi_1(\hat{\boldsymbol{a}}_1) & \dots & \psi_1(\hat{\boldsymbol{a}}_{\hat{T}_N}) \\ \vdots & & \vdots \\ \psi_{\hat{T}_N}(\hat{\boldsymbol{a}}_1) & \dots & \psi_{\hat{T}_N}(\hat{\boldsymbol{a}}_{\hat{T}_N}) \end{bmatrix}}_{V(\mathcal{B},X)} = \boldsymbol{I}_{\hat{T}_N}$$

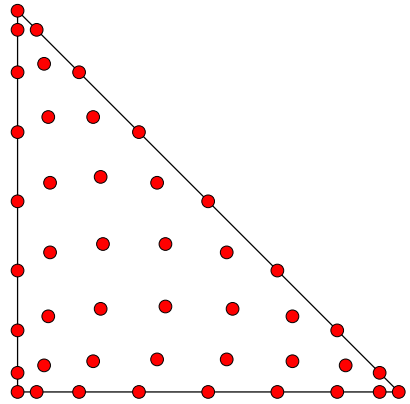
où $X = \{\hat{\boldsymbol{a}}_i\}_{i=1}^{\hat{T}_N}$ représente l'ensemble des points d'interpolation et $\boldsymbol{I}_{\hat{T}_N}$ désigne la matrice identitée de taille $(\hat{T}_N) \times (\hat{T}_N)$. Le choix des points d'interpolation et de la base primale est très important dès que l'on souhaite faire de l'ordre élevé. Comme démontré dans [124, 35, 91], ce choix est crucial pour calculer avec précision les coefficients α_{ij}, à cause du conditionnement de la matrice de Vandermonde $V(\mathcal{B}, X)$ (voir figures 1.2(a) et 1.2(b)). Il permet également d'avoir un meilleur conditionnement de la matrice de rigidité, noté $\mathcal{K}(A)$ pour le nombre de conditionnements, voir figure 1.2(d).



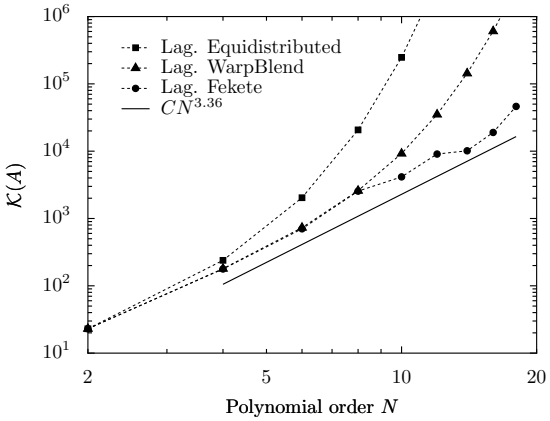
(a) Conditionnement de $V(\mathcal{B}, X)$ pour les points équidistribués



(b) Conditionnement de $V(\mathcal{B}, X)$ pour les points de Fekete



(c) Points d'interpolation de Warp Blend



(d) Conditionnement de la matrice de rigidité

FIGURE 1.2 – Illustration de l'influence du choix de la base primal et des points d'interpolation.

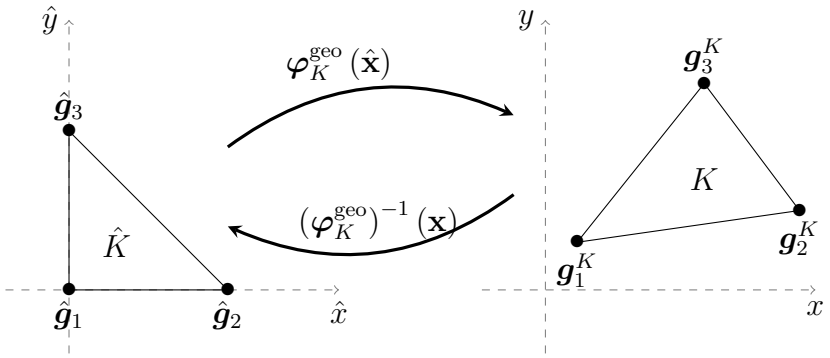
Definition 2. *Soit* $\left\{\hat{\Phi}_i, i = 1, ..., \hat{T}_N\right\}$ *l'ensemble des fonctions de base de l'élément fini de Lagrange* $\left(\hat{K}, \hat{\mathbb{P}}_N, \hat{\Sigma}\right)$. *On note* $\left\{(\hat{\sigma}_i, \hat{\boldsymbol{a}}_i), i = 1, ..., \hat{T}_N\right\}$ *l'ensemble des formes linéaires et des points d'interpolation associés à* $\hat{\Sigma}$. *On appelle opérateur d'interpolation locale sur l'élément de référence* $\hat{K}$, *l'opérateur défini par :*

$$\begin{aligned} \pi_{\hat{K}} : C^0\left(\hat{K}\right) &\longrightarrow \hat{\mathbb{P}}_N\left(\hat{K}\right) \\ v &\longmapsto \pi_{\hat{K}}(v) = \sum_{i=1}^{\hat{T}_N} \hat{\sigma}_i(v)\,\hat{\Phi}_i = \sum_{i=1}^{\hat{T}_N} v\left(\hat{\boldsymbol{a}}_i\right)\hat{\Phi}_i \end{aligned} \tag{1.6}$$

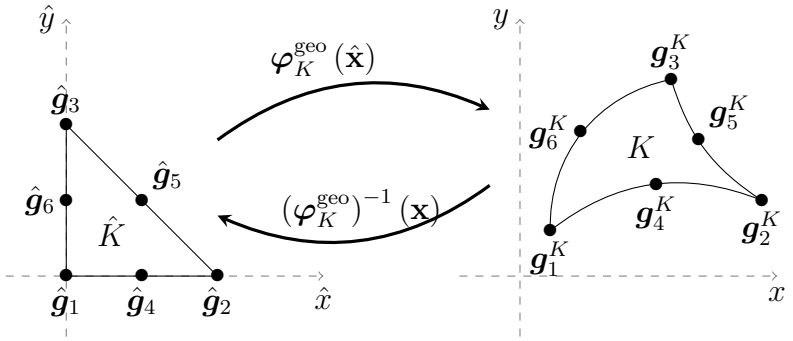
On dit que $\pi_{\hat{K}}(v)$ est l'interpolé de Lagrange de v sur $\hat{K}$. L'interpolé de Lagrange est tel que sa valeur aux points d'interpolation $\left\{\hat{\boldsymbol{a}}_i, i = 1, ..., \hat{T}_N\right\}$ coïncide avec celle de la fonction à interpoler v.

1.2 Transformation géométrique

Pour pouvoir définir un élément fini sur un élément K quelconque, il existe deux techniques. Soit nous pouvons directement appliquer le formalisme présenté dans la section précédente sur l'élément K, soit nous avons la possibilité d'utiliser la technique de l'élément de référence. C'est cette dernière solution que nous avons choisie, car elle a de nombreux avantages. L'approximation au sens éléments finis nous demande de définir les fonctions de base $\left\{\Phi_i^N\right\}_{i=1}^{T_N}$ associées à l'élément fini de Lagrange $(K, \mathbb{P}_N, \Sigma)$. Pour cela, nous utilisons les fonctions de base $\hat{\Phi}_i^N$ construites sur un élément de référence $\hat{K}$ et à l'aide d'une transformation géométrique $\boldsymbol{\varphi}_K^{\text{geo}}$, nous obtenons les fonctions de base sur l'élément réel K. Ainsi, beaucoup de calculs élémentaires sont effectués sur l'élément $\hat{K}$. Par exemple, c'est la cas pour l'évaluation ou la dérivation des fonctions de base aux points de quadratures. Ces calculs sont effectués avec précision et certains d'entre eux peuvent être ensuite réutilisés pour tous les mêmes types d'élément. Nous bénéficions toujours des propriétés énoncées précédemment pour les fonctions de base primale (construction hiérarchique et L^2-orthonormalité) et des points d'interpolation (conditionnement de la matrice de Vandermonde).



(a) Transformation linéaire sur un triangle



(b) Transformation quadratique sur un triangle

FIGURE 1.3 – Transformations géométriques d'ordre 1 et 2 sur un triangle.

La transformation géométrique est une application inversible de $\hat{K}$ vers K dès que l'élément K n'est pas dégénéré. Nous faisons l'hypothèse que $\boldsymbol{\varphi}_K^{\text{geo}}$ est C^1-difféomorphisme.

On définit ainsi $\boldsymbol{\varphi}_K^{\text{geo}}$ et son inverse $(\boldsymbol{\varphi}_K^{\text{geo}})^{-1}$:

$$\begin{array}{rccc} \boldsymbol{\varphi}_K^{\text{geo}}: & \hat{K}\in\mathbb{R}^d & \to & K\in\mathbb{R}^d \\ & \hat{\boldsymbol{x}} & \mapsto & \boldsymbol{x} \end{array} \quad \text{et} \quad \begin{array}{rccc} (\boldsymbol{\varphi}_K^{\text{geo}})^{-1}: & K\in\mathbb{R}^d & \to & \hat{K}\in\mathbb{R}^d \\ & \boldsymbol{x} & \mapsto & \hat{\boldsymbol{x}} \end{array} \tag{1.7}$$

L'élément de référence $\hat{K}$ peut être à priori choisi librement à condition de respecter :
- le type de l'élément (segment, triangle, quadrangle, tétraèdre, ...)
- le nombre de nœuds géométriques (ordre de la transformation géométrique)
- la numérotation des nœuds sur K et $\hat{K}$ doit être compatible pour garantir que $\boldsymbol{\varphi}_K^{\text{geo}}$ soit un C^1-difféomorphisme

Cependant, son choix peut être motivé par le domaine de définition de la base primale.

Nous allons maintenant utiliser l'élément fini de Lagrange pour définir la transformation géométrique $\boldsymbol{\varphi}_K^{\text{geo}}$. Ce formalisme nous permet de généraliser la définition de la transformation géométrique pour un ordre d'approximation arbitraire. Nous représentons avec la figure 1.3 deux types de transformation géométrique.

Definition 3. *On dit que $(\hat{K}, \hat{\mathbb{P}}_k^{geo}, \hat{\Sigma}^{geo})$ est l'élément fini géométrique d'ordre k. On pose $N_g = card(\hat{\Sigma}^{geo})$. On note par $\left\{\hat{\boldsymbol{g}}_1, ..., \hat{\boldsymbol{g}}_{N_g}\right\}$ l'ensemble des noeuds géométriques de $\hat{K}$ et ils correspondent aux points équidistribués sur cet élément. L'ensemble des fonctions de base de cet élément fini est noté par $\left\{\psi_1, ..., \psi_{N_g}\right\}$.*

L'élément K est constitué de N_g noeuds géométriques $\left\{\boldsymbol{g}_1^K, ..., \boldsymbol{g}_{N_g}^K\right\}$. Chaque point $\boldsymbol{g}_i^K$ est caractérisé par ses coordonnées $(g_{i,x}^K, g_{i,y}^K, g_{i,z}^K)$ dans le repère cartésien. On peut alors exprimer l'application $\boldsymbol{\varphi}_K^{\text{geo}}$ comme une combinaison linéaire de fonctions de base $\psi_i, i = 1, ..., N_g$:

$$\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}) = \sum_{i=1}^{N_g} \boldsymbol{g}_i^K \psi_i(\hat{\mathbf{x}}) \tag{1.8}$$

Sous sa forme algébrique (1.9), la transformation $\boldsymbol{\varphi}_K^{\text{geo}}$ fait apparaitre une matrice G représentant les points géométriques de l'élément de K. Le vecteur $L(\hat{\mathbf{x}})$, quant à lui, ne dépend pas de K, mais uniquement des fonctions de base évaluées aux points $\hat{\mathbf{x}} \in \hat{K}$.

$$\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}) = \begin{pmatrix} \varphi_{K,x}^{\text{geo}}(\hat{\mathbf{x}}) \\ \varphi_{K,y}^{\text{geo}}(\hat{\mathbf{x}}) \\ \varphi_{K,z}^{\text{geo}}(\hat{\mathbf{x}}) \end{pmatrix} = \sum_{i=1}^{N_g} \begin{pmatrix} g_{i,x}^K \\ g_{i,y}^K \\ g_{i,z}^K \end{pmatrix} \psi_i(\hat{\mathbf{x}}) = \underbrace{\begin{pmatrix} g_{1,x}^K & g_{2,x}^K & \cdots & g_{N_g,x}^K \\ g_{1,y}^K & g_{2,y}^K & \cdots & g_{N_g,y}^K \\ g_{1,z}^K & g_{2,z}^K & \cdots & g_{N_g,z}^K \end{pmatrix}}_{G} \underbrace{\begin{pmatrix} \psi_1(\hat{\mathbf{x}}) \\ \psi_2(\hat{\mathbf{x}}) \\ \vdots \\ \psi_{N_g}(\hat{\mathbf{x}}) \end{pmatrix}}_{L(\hat{\mathbf{x}})} \tag{1.9}$$

Nous avons aussi besoin de définir la matrice jacobienne de $\boldsymbol{\varphi}_K^{\text{geo}}$ que l'on note $K(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}} \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})$, le jacobien $J(\hat{\mathbf{x}}) = det(K(\hat{\mathbf{x}}))$ et la transposée de l'inverse de $K(\hat{\mathbf{x}})$ notée $B(\hat{\mathbf{x}}) = K(\hat{\mathbf{x}})^{-T}$. Ces expressions sont indispensables lorsque l'on souhaite faire le changement de variable $\boldsymbol{\varphi}_K^{\text{geo}}$ dans une intégrale sur K ou sur une face de K.

$$
\begin{aligned}
\nabla_{\hat{\mathbf{x}}} \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}) &= \begin{pmatrix} \frac{\partial \varphi_{K,x}^{\text{geo}}}{\partial \hat{x}}(\hat{\mathbf{x}}) & \frac{\partial \varphi_{K,x}^{\text{geo}}}{\partial \hat{y}}(\hat{\mathbf{x}}) & \frac{\partial \varphi_{K,x}^{\text{geo}}}{\partial \hat{z}}(\hat{\mathbf{x}}) \\ \frac{\partial \varphi_{K,y}^{\text{geo}}}{\partial \hat{x}}(\hat{\mathbf{x}}) & \frac{\partial \varphi_{K,y}^{\text{geo}}}{\partial \hat{y}}(\hat{\mathbf{x}}) & \frac{\partial \varphi_{K,y}^{\text{geo}}}{\partial \hat{z}}(\hat{\mathbf{x}}) \\ \frac{\partial \varphi_{K,z}^{\text{geo}}}{\partial \hat{x}}(\hat{\mathbf{x}}) & \frac{\partial \varphi_{K,z}^{\text{geo}}}{\partial \hat{y}}(\hat{\mathbf{x}}) & \frac{\partial \varphi_{K,z}^{\text{geo}}}{\partial \hat{z}}(\hat{\mathbf{x}}) \end{pmatrix} \\
&= \begin{pmatrix} \sum_{i=1}^{N_g} g_{i,x}^K \frac{\partial \psi_i}{\partial \hat{x}}(\hat{\mathbf{x}}) & \sum_{i=1}^{N_g} g_{i,x}^K \frac{\partial \psi_i}{\partial \hat{y}}(\hat{\mathbf{x}}) & \sum_{i=1}^{N_g} g_{i,x}^K \frac{\partial \psi_i}{\partial \hat{z}}(\hat{\mathbf{x}}) \\ \sum_{i=1}^{N_g} g_{i,y}^K \frac{\partial \psi_i}{\partial \hat{x}}(\hat{\mathbf{x}}) & \sum_{i=1}^{N_g} g_{i,y}^K \frac{\partial \psi_i}{\partial \hat{y}}(\hat{\mathbf{x}}) & \sum_{i=1}^{N_g} g_{i,y}^K \frac{\partial \psi_i}{\partial \hat{z}}(\hat{\mathbf{x}}) \\ \sum_{i=1}^{N_g} g_{i,z}^K \frac{\partial \psi_i}{\partial \hat{x}}(\hat{\mathbf{x}}) & \sum_{i=1}^{N_g} g_{i,z}^K \frac{\partial \psi_i}{\partial \hat{y}}(\hat{\mathbf{x}}) & \sum_{i=1}^{N_g} g_{i,z}^K \frac{\partial \psi_i}{\partial \hat{z}}(\hat{\mathbf{x}}) \end{pmatrix} \\
&= \begin{pmatrix} g_{1,x}^K & g_{2,x}^K & \cdots & g_{N_g,x}^K \\ g_{1,y}^K & g_{2,y}^K & \cdots & g_{N_g,y}^K \\ g_{1,z}^K & g_{2,z}^K & \cdots & g_{N_g,z}^K \end{pmatrix} \begin{pmatrix} \frac{\partial \psi_1}{\partial \hat{x}}(\hat{\mathbf{x}}) & \frac{\partial \psi_1}{\partial \hat{y}}(\hat{\mathbf{x}}) & \frac{\partial \psi_1}{\partial \hat{z}}(\hat{\mathbf{x}}) \\ \frac{\partial \psi_2}{\partial \hat{x}}(\hat{\mathbf{x}}) & \frac{\partial \psi_2}{\partial \hat{y}}(\hat{\mathbf{x}}) & \frac{\partial \psi_2}{\partial \hat{z}}(\hat{\mathbf{x}}) \\ \vdots & \vdots & \vdots \\ \frac{\partial \psi_{N_g}}{\partial \hat{x}}(\hat{\mathbf{x}}) & \frac{\partial \psi_{N_g}}{\partial \hat{y}}(\hat{\mathbf{x}}) & \frac{\partial \psi_{N_g}}{\partial \hat{z}}(\hat{\mathbf{x}}) \end{pmatrix} \\
&= G \, \nabla_{\hat{x}} L(\hat{\mathbf{x}})
\end{aligned}
\tag{1.10}
$$

Quand nous voulons effectuer un calcul d'intégral sur un élément K, nous nous ramenons toujours à l'intégration sur l'élément de référence. Soit $f : K \to \mathbb{R}$ une fonction. Nous avons les formules de changement de variable suivantes qui interviennent lors des calculs d'intégrale sur un élément :

$$
\int_K f(\mathbf{x}) \, d\mathbf{x} = \int_{\hat{K}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})) \, J(\hat{\mathbf{x}}) \, d\hat{\mathbf{x}} \tag{1.11}
$$

$$
\int_K \nabla_{\mathbf{x}} f(\mathbf{x}) \, d\mathbf{x} = \int_{\hat{K}} B(\hat{\mathbf{x}}) \, \nabla_{\hat{\mathbf{x}}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})) \, J(\hat{\mathbf{x}}) \, d\hat{\mathbf{x}} \tag{1.12}
$$

De la même manière, nous allons montrer les changements de variables et les objets à calculer sur l'élément de référence pour effectuer un calcul d'intégrale sur les faces des éléments. Soit $\Gamma \subset K$ une face de l'élément K et $\hat{\Gamma} \subset \hat{K}$ la face de l'élément de référence telle que $\Gamma = \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\Gamma})$. Nous notons également $\boldsymbol{n}_\Gamma(\mathbf{s})$ (respectivement $\boldsymbol{n}_{\hat{\Gamma}}(\hat{\mathbf{s}})$) la normale de Γ (respectivement $\hat{\Gamma}$) orientée vers l'extérieur par rapport à K (respectivement $\hat{K}$) et évaluée au point $\mathbf{s} \in \Gamma$ (respectivement $\hat{\mathbf{s}} \in \hat{\Gamma}$). Nous avons aussi besoin de définir le jacobien surfacique de la transformation géométrique, noté $J_s(\hat{\mathbf{s}})$, qui est défini par $J_s(\hat{\mathbf{s}}) = J(\hat{\mathbf{s}}) \| B(\hat{\mathbf{s}}) \boldsymbol{n}_{\hat{\Gamma}} \|$. De plus, nous avons la relation $\boldsymbol{n}_\Gamma(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{s}})) = \frac{J(\hat{\mathbf{s}})}{J_s(\hat{\mathbf{s}})} B(\hat{\mathbf{s}}) \boldsymbol{n}_{\hat{\Gamma}}(\hat{\mathbf{s}})$. Soit $f : K \to \mathbb{R}$ une fonction scalaire et $g : K \to \mathbb{R}^d$ une fonction vectorielle. Nous avons

alors :

$$\int_\Gamma f(\mathbf{s})\ d\mathbf{s} = \int_{\hat{\Gamma}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{s}}))\ J_s(\hat{\mathbf{s}})\ d\hat{\mathbf{s}} \tag{1.13}$$

$$= \int_{\hat{\Gamma}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{s}}))\ J(\hat{\mathbf{s}})\ \|B(\hat{\mathbf{s}})\boldsymbol{n}_{\hat{\Gamma}}(\hat{\mathbf{s}})\|\ d\hat{\mathbf{s}} \tag{1.14}$$

$$\int_\Gamma f(\mathbf{s})\cdot\boldsymbol{n}_\Gamma(\mathbf{s})\ d\mathbf{s} = \int_{\hat{\Gamma}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{s}}))\cdot\left(\frac{J(\hat{\mathbf{s}})}{J_s(\hat{\mathbf{s}})}B(\hat{\mathbf{s}})\boldsymbol{n}_{\hat{\Gamma}}(\hat{\mathbf{s}})\right) J(\hat{\mathbf{s}})\|B(\hat{\mathbf{s}})\boldsymbol{n}_{\hat{\Gamma}}(\hat{\mathbf{s}})\|d\hat{\mathbf{s}} \tag{1.15}$$

$$= \int_{\hat{\Gamma}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{s}}))\cdot(B(\hat{\mathbf{s}})\boldsymbol{n}_{\hat{\Gamma}}(\hat{\mathbf{s}}))\, J(\hat{\mathbf{s}})\ d\hat{\mathbf{s}} \tag{1.16}$$

1.3 Expansion au multi-domaine

Nous considérons tout d'abord le cas où l'on a un seul élément K. Nous pouvons naturellement étendre l'espace $\mathbb{P}_N\left(\hat{K}\right)$ en considérant l'image de tous les polynômes appartenant à cet espace par la transformation géométrique $\boldsymbol{\varphi}_K^{\text{geo}}$. Nous obtenons alors :

$$\mathbb{P}_N(K) = \left\{p : p = \hat{p}\circ(\boldsymbol{\varphi}_K^{\text{geo}})^{-1}, \hat{p}\in\mathbb{P}_N\left(\hat{K}\right)\right\} \tag{1.17}$$

Nous notons T_N la dimension de $\mathbb{P}_N(K)$ et celle-ci est évidemment égale à $\hat{T}_N$. Pour tout $i\in\{1,...,T_N\}$, nous posons $\boldsymbol{a}_i^K = \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\boldsymbol{a}}_i)$. On désigne par Σ_K les degrés de liberté associés aux noeuds $\left\{\boldsymbol{a}_1^K,...,\boldsymbol{a}_{T_N}^K\right\}$ dans K. Nous pouvons alors constater que le triplet $(K,\mathbb{P}_N,\Sigma_K)$ est un élément fini de Lagrange. Les fonctions de base de cet élément fini sont données par :

$$\Phi_{K,i}^N = \hat{\Phi}_i^N\circ(\boldsymbol{\varphi}_K^{\text{geo}})^{-1} \quad ,\forall i\in\{1,...,T_N\} \tag{1.18}$$

Definition 4. *Soit $\left\{\Phi_{K,i}^N, i=1,...,T_N\right\}$ l'ensemble des fonctions de base de l'élément fini de Lagrange $(K,\mathbb{P}_N,\Sigma_K)$ et $\left\{\boldsymbol{a}_i^K, i=1,...,T_N\right\}$ l'ensemble des points d'interpolation associée à Σ_K. L'opérateur d'interpolation locale sur l'élément K est donné par :*

$$\begin{aligned}
&\pi_K : C^0(K) \longrightarrow \mathbb{P}_N(K)\\
&\qquad v \longmapsto \pi_K(v) \quad = \sum_{i=1}^{T_N} v\left(\boldsymbol{a}_i^K\right)\Phi_{K,i}^N\\
&\qquad\qquad\qquad\ = \sum_{i=1}^{T_N} v\left(\boldsymbol{a}_i^K\right)\hat{\Phi}_i^N\circ(\boldsymbol{\varphi}_K^{\text{geo}})^{-1}
\end{aligned} \tag{1.19}$$

Nous passons maintenant à la discrétisation d'un domaine quelconque $\Omega\subset\mathbb{R}^d$. Cette discrétisation est constitué d'une famille d'élément $\{K_e\}_{1\le e\le N_{elt}}$, où N_{elt} est le nombre d'éléments formant cette discrétisation appelée maillage. Pour chaque élément K_e, nous posons :

$$h_{K_e} = diam(K_e) = \max_{x_1,x_2\in K_e}\|x_1-x_2\|_{\mathbb{R}^d} \tag{1.20}$$

correspondant à une mesure caractéristique de l'élément K_e. On définit également le pas de maillage (ou taille caractéristique) h comme étant :

$$h = \max_{1\le e\le N_{elt}} h_{K_e} \tag{1.21}$$

Associé à la taille caractéristique du maillage h, nous ajoutons le paramètre k correspondant à l'ordre géométrique utilisé pour la description des éléments K_e. Nous posons $\delta \equiv (h,k)$ comme étant le paramètre de discrétisation du maillage. Nous notons à partir de maintenant le maillage $\mathcal{T}_\delta$. Ce maillage est formé d'un ensemble d'éléments géométriques K_e et ces éléments peuvent être générés à l'aide de la transformation géométrique $\boldsymbol{\varphi}_K^{\text{geo}}$ sur l'élément de référence $\hat{K}$:

$$\mathcal{T}_\delta \equiv \mathcal{T}_{(h,k)} = \{K = \boldsymbol{\varphi}_K^{\text{geo}}(\hat{K})\} \tag{1.22}$$

Le domaine discrétisé ne recouvre pas forcément le domaine d'étude initial. C'est pourquoi, nous posons maintenant la définition du domaine de recouvrement associé à la discrétisation $\mathcal{T}_\delta$:

$$\Omega_\delta = \bigcup_{e=1}^{N_{elt}} K_e \tag{1.23}$$

Lorsque le domaine Ω est de dimension 1 ou un polygone (d=2) ou un polyèdre(d=3), le maillage construit permet de recouvrir exactement le domaine d'étude, c'est-à-dire $\Omega_\delta = \Omega$. Par contre, si le domaine Ω est à frontière courbe, le recouvrement n'est pas exact en général et on admet que l'on commet une erreur de discrétisation géométrique. Enfin, nous disons que le maillage généré $\mathcal{T}_\delta$ est conforme s'il vérifie les propriétés suivantes :

- $\overset{\circ}{K}_i \cap \overset{\circ}{K}_j = \varnothing$ si et seulement si $i \neq j$
- Deux éléments voisins ont seulement un sommet, une arête ou une face en commun.

On désigne maintenant par $\{(K, \mathbb{P}_N, \Sigma_K), K \in \mathcal{T}_\delta\}$ une famille d'éléments finis de Lagrange pour la discrétisation $\mathcal{T}_\delta$ et basé sur un élément fini de référence $\left(\hat{K}, \hat{\mathbb{P}}_N, \hat{\Sigma}\right)$. Pour chaque élément fini de la famille, nous notons par $\left\{\boldsymbol{a}_i^K\right\}_{i=1}^{T_n}$ l'ensemble des points d'interpolation associés à Σ_K. La méthode des éléments finis permet ensuite de construire des espaces d'approximation grâce à la famille d'éléments finis. Nous donnons la définition des espaces de fonctions scalaires continues $P_c^N(\Omega_\delta)$ et discontinues pour $P_d^N(\Omega_\delta)$ par :

$$P_c^N(\Omega_\delta) = \{v \in C^0(\bar{\Omega}_\delta) \mid \ v \circ \boldsymbol{\varphi}_K^{\text{geo}} \in \mathbb{P}^N(\hat{K}) \ \forall K \in \mathcal{T}_\delta\} \tag{1.24}$$

$$P_d^N(\Omega_\delta) = \{v \in L^2(\Omega_\delta) \mid \ v \circ \boldsymbol{\varphi}_K^{\text{geo}} \in \mathbb{P}^N(\hat{K}) \ \forall K \in \mathcal{T}_\delta\} \tag{1.25}$$

Nous étendons également les définitions de ces espaces au cas des fonctions vectorielles :

$$[P_c^N(\Omega_\delta)]^d = \prod_1^d P_c^N(\Omega_\delta) \quad , \qquad [P_d^N(\Omega_\delta)]^d = \prod_1^d P_d^N(\Omega_\delta) \tag{1.26}$$

Nous abordons à présent la construction d'une base de l'espace $P_c^N(\Omega_\delta)$. On commence par définir l'ensemble des degrés de liberté de cet espace de fonction. On note $\left\{\boldsymbol{a}_1, ..., \boldsymbol{a}_{N_{dof}}\right\} = \bigcup_{K\in\mathcal{T}_\delta} \left\{\boldsymbol{a}_1^K, ..., \boldsymbol{a}_{T_N}^K\right\}$ la réunion des noeuds ou points d'interpolation de tous les éléments du maillage, les noeuds situés aux interfaces n'étant comptabilisés qu'une fois. Nous avons posé N_{dof}, le nombre de degrés de liberté présents. Nous notons par $\left\{\Phi_i^N, i = 1, ..., \Phi_{N_{dof}}^N\right\}$ l'ensemble des fonctions de base de l'espace $P_c^N(\Omega_\delta)$. La méthode des éléments finis nous indique que ces fonctions de base doivent vérifier :

$$\Phi_i^N(\boldsymbol{a}_j) = \delta_{ij} \quad , 1 \leq i, j, \leq N_{dof} \tag{1.27}$$

Pour construire la fonction de base Φ_i^N, nous considérons l'un des deux cas suivants :

1. si le point $\boldsymbol{a}_i$ appartient à l'intérieur d'un élément K, on définit Φ_i^N comme la fonction qui coïncide avec la fonction de base de l'élément fini $(K, \mathbb{P}_N, \Sigma_K)$ associée au noeud $\boldsymbol{a}_i$ dans K et qui vaut zéro sur les autres mailles
2. si le point $\boldsymbol{a}_i$ est partagé par plusieurs éléments, on définit Φ_i^N comme la fonction qui coïncide avec la fonction de base associée au noeud $\boldsymbol{a}_i$ sur chacun des éléments et qui vaut zéro sur les autres mailles.

Definition 5. *Soit $\left\{\Phi_i^N, i = 1, ..., N_{dof}\right\}$ l'ensemble des fonctions de base globale de $P_c^N(\Omega_\delta)$ et $\{\boldsymbol{a}_i, i = 1, N_{dof}\}$ l'ensemble des degrés de liberté associé. L'opérateur d'interpolation global de Lagrange est défini par :*

$$\begin{aligned} \pi : C^0\left(\bar{\Omega}_\delta\right) &\longrightarrow P_c^N(\Omega_\delta) \\ v &\longmapsto \pi(v) = \sum_{i=1}^{N_{dof}} v\left(\boldsymbol{a}_i\right) \Phi_i^N \end{aligned} \tag{1.28}$$

On observe également que $\forall K \in \mathcal{T}_\delta$ et $\forall v \in C^0\left(\bar{\Omega}_\delta\right)$, on a :

$$\pi\left(v\right)|_K = \pi_K\left(v_{|K}\right) \tag{1.29}$$

Cependant, au vu du grand nombre de fonctions de base et de leur support restreint, celles-ci ne sont pas construites. On utilise une approche par élément en utilisant les fonctions de bases de l'élément fini de référence $\left(\hat{K}, \hat{\mathbb{P}}_N, \hat{\Sigma}\right)$ et les transformations géométriques. La clé de cette méthode est une table de degré de liberté $n(e, i)$ permettant de relier un degré de liberté local i d'un élément K_e aux degrés de liberté globaux j. Ce qui nous donne $j = n(e, i)$. Dans la pratique, cette construction est complexe à mettre en oeuvre pour les fonctions continues, et de plus la numérotation globale des degrés de liberté joue un rôle important sur les structures des matrices éléments finis. Soit K_e et K_f deux éléments voisins et $\boldsymbol{a}_i^{K_e}$ et $\boldsymbol{a}_j^{K_f}$ deux noeuds identiques. Nous avons alors la relation $n(e, i) = n(f, j)$ qui doit être vérifiée. Pour le cas de fonctions discontinues, nous n'avons pas besoin de cette contrainte. Il est alors assez simple de construire cette table.

Propriété 1. *Soit $K_e \in \mathcal{T}_\delta$ et $(K_e, \mathbb{P}_N, \Sigma_{K_e})$ un élément fini de Lagrange. On désigne par i l'indice du degré de liberté local tel que $1 \leq i \leq T_N$. La fonction de bases $\Phi_{K_e,i}^N$ associée à cet élément fini vérifie alors :*

$$\Phi_{K_e,i}^N\left(\boldsymbol{x}\right) = \hat{\Phi}_i^N \circ \left(\boldsymbol{\varphi}_{K_e}^{\text{geo}}\right)^{-1}\left(\boldsymbol{x}\right) = \Phi_{n(e,i)}^N\big|_{K_e}\left(\boldsymbol{x}\right) \quad , \forall \boldsymbol{x} \in K_e \tag{1.30}$$

Pour terminer cette partie, nous faisons un commentaire sur le lien entre élément fini géométrique et élément fini de référence. Lorsque le domaine d'étude est considéré comme un domaine à frontière courbe et qu'on souhaite utiliser un élément fini de référence de degré élevé, on n'obtiendra des propriétés d'interpolation optimales que si la frontière du domaine est décrite avec une précision suffisante. On doit donc utiliser un élément fini géométrique de degré élevé. Ces considérations motivent la définition suivante.

Definition 6. *Soit $\left(\hat{K}, \hat{\mathbb{P}}_N, \hat{\Sigma}\right)$ l'élément fini de référence et soit $(\hat{K}, \hat{\mathbb{P}}_k^{geo}, \hat{\Sigma}^{geo})$ l'élément fini géométrique. Lorsque ces deux éléments finis sont les mêmes, on parle d'interpolation isoparamétrique. On parle d'interpolation subparamétrique lorsque $\hat{\mathbb{P}}_k^{geo} \subset \hat{\mathbb{P}}_N$ et $\hat{\mathbb{P}}_k^{geo} \neq \hat{\mathbb{P}}_N$.*

2 Interpolation et non-conformité

Dans le contexte d'écoulement sanguin, plusieurs interactions entre en jeu comme celle du plasma avec la paroi artérielle. Pour simuler de tels phénomènes, nous allons utiliser des modèles définis sur des domaines distincts. Cependant, pour prendre en compte ces interactions, des termes de couplage apparaitront. Les maillages associés au modèle seront totalement indépendants, on parle alors de non-conformité entre ces entités.

Nous allons présenter dans cette partie les ingrédients mathématiques nécessaires pour le couplage de nos modèles. Nous commencerons par l'interpolation des fonctions entre des maillages différents. Nous parlerons ensuite des outils numériques développés pour utiliser une méthode de Galerkin non standard. Au cours de cette présentation, nous mettrons également en avant des extraits de code Feel++. L'intérêt est de montrer la similitude du langage informatique et du langage mathématique et la transparence de l'outil d'interpolation dans le langage.

2.1 Opérateurs d'interpolation

Dans de nombreux contextes, un besoin essentiel est le transfert d'informations aux interfaces de différents maillages. C'est le cas par exemple des applications d'interaction fluide-structure, présentées au chapitre 6, où les champs de vitesse, contrainte interne et déplacement doivent être échangés sur l'interface fluide-structure. D'autres applications comme les méthodes de décomposition de domaine requièrent une telle fonctionnalité de transfert. Dans certains cas, les maillages peuvent aussi se superposer et aucune relation n'existe entre eux. C'est le cas par exemple avec notre méthode de modélisation de globule rouge utilisant la méthode de la frontière élargie décrite dans le chapitre 3. Des exemples

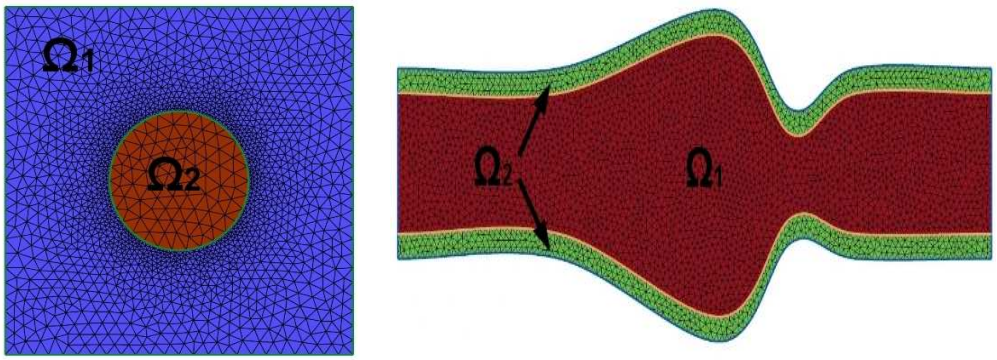


Figure 1.4 – Exemples de géométrie où l'on pourrait appliquer des outils d'interpolation

illustrant la connexion entre deux domaines d'application sont présentés avec la figure 1.4. Nous avons les domaines Ω_1 et Ω_2 et l'objectif est d'échanger des informations à l'interface $\Gamma = \Omega_1 \cap \Omega_2$. De plus, la connexion des domaines discrets ne sera pas forcément conforme. Nous verrons aussi qu'il est parfois nécessaire de recourir à l'extrapolation.

Dans la suite, nous ferons également appel à un outil de localisation de point dans un maillage. Cet outil se base sur une méthode de recherche nommée *kdtree* qui sera décrite dans la section 2.3 de ce chapitre.

Évaluation d'une intégrale

On présente dans cette partie la technique que nous utilisons pour évaluer l'intégrale d'une fonction u sur un domaine quelconque. Nous considérons deux domaines Ω et Ω' tels que $\Omega' \subset \Omega$. Nous générons le maillage $\mathcal{T}_\delta$ (resp $\mathcal{T}'_\delta$) associé au domaine Ω (resp Ω') formant un domaine de calcul Ω_δ (resp Ω'_δ). On fait aussi l'hypothèse que $\Omega'_\delta \subset \Omega_\delta$. On choisit u une fonction appartenant à $P_c^N(\Omega_\delta)$. Cette fonction est entièrement déterminée par ces N_{dof} degrés de liberté $u_i = u(\mathbf{a}_i), i = 1, ..., N_{dof}$.

Soit $K \in \mathcal{T}'_\delta$. On note par $J_K(\hat{\mathbf{x}}) = det\left(\nabla_{\hat{\mathbf{x}}} \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})\right)$ le jacobien de la transformation géométrique associée à K. Nous considérons également une formule de quadrature sur $\hat{K}$ que l'on définit par l'ensemble $\{(\hat{\boldsymbol{r}}_q, w_q), i = 1, ..., N_q\}$. Cet ensemble contient N_q points de quadrature $\hat{\boldsymbol{r}}_q$ et on associe à chacun de ses points un poids de quadrature w_q.

L'intégrale de la fonction u est décomposé sur chacun des éléments, puis elle est ramenée à l'élément de référence grâce à la transformation géométrique. Ce qui donne les relations suivantes :

$$\int_{\Omega'_\delta} u(\boldsymbol{x})\ d\boldsymbol{x} = \sum_{K' \in \mathcal{T}'_\delta} \int_{K'} u(\boldsymbol{x})\ d\boldsymbol{x} \tag{1.31}$$

$$= \sum_{K' \in \mathcal{T}'_\delta} \int_{\hat{K}} u \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{x}})\ J_{K'}(\hat{\boldsymbol{x}})\ d\hat{\boldsymbol{x}} \tag{1.32}$$

$$= \sum_{K' \in \mathcal{T}'_\delta} \sum_{q=1}^{N_q} w_q\ u \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q)\ \ J_{K'}(\hat{\boldsymbol{r}}_q) \tag{1.33}$$

Dans le cas classique où $\Omega'_\delta = \Omega_\delta$, l'évaluation de $u \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q)$ ne nécessite pas d'utiliser un outil de localisation. On pose e l'indice d'un élément $K' \in \mathcal{T}'_\delta$. Cette évaluation se traduit alors par :

$$u \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q) = \pi_{K'}(u) \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q) = \sum_{i=1}^{T_N} u(\boldsymbol{a}_{n(e,i)})\ \Phi^N_{(K',i)} \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q) \tag{1.34}$$

$$= \sum_{i=1}^{T_N} u(\boldsymbol{a}_{n(e,i)})\ \hat{\Phi}^N_i \circ (\boldsymbol{\varphi}_{K'}^{\text{geo}})^{-1} \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q) \tag{1.35}$$

$$= \sum_{i=1}^{T_N} u(\boldsymbol{a}_{n(e,i)})\ \hat{\Phi}^N_i(\hat{\boldsymbol{r}}_q) \tag{1.36}$$

Cependant, nous nous intéressons au cas $\Omega'_\delta \neq \Omega_\delta$. Il est alors nécessaire d'utiliser un outil de localisation de point dans un maillage. Ici, nous voulons localiser le point $\boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q)$ dans le maillage de définition de u, c'est-à-dire $\mathcal{T}_\delta$. Nous posons par $K_e \in \mathcal{T}_\delta$ l'élément qui contient ce point. Pour alléger les notations, nous noterons simplement e, l'indice de l'élément localisé et e dépend de K' et $\hat{\boldsymbol{r}}_q$. Ainsi, nous pouvons poser :

$$u \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q) = \pi_{K_e}(u) \circ \boldsymbol{\varphi}_{K'}^{\text{geo}}(\hat{\boldsymbol{r}}_q) \tag{1.37}$$

Toutefois, cet élément K_e n'est pas obligatoirement unique, mais comme la fonction $u \in P_c^N(\Omega_\delta)$, n'importe lequel d'entre eux peut être utilisé. Par contre, ceci n'est pas vrai si nous avons $u \in P_d^N(\Omega_\delta)$. Nous n'aborderons pas ce cas qui est un peu plus complexe.

Nous pouvons maintenant revenir à notre calcul d'intégral et obtenir l'expression détaillée suivante :

$$\int_{\Omega'_\delta} u(\boldsymbol{x})\, d\boldsymbol{x} = \sum_{K'\in\mathcal{T}'_\delta}\sum_{q=1}^{N_q} w_q\ \pi_{K_e}(u)\circ\boldsymbol{\varphi}^{\text{geo}}_{K'}(\hat{\boldsymbol{r}}_q)\ \ J_{K'}(\hat{\boldsymbol{r}}_q) \tag{1.38}$$

$$= \sum_{K'\in\mathcal{T}'_\delta}\sum_{q=1}^{N_q} w_q J_{K'}(\hat{\boldsymbol{r}}_q)\sum_{i=1}^{T_N} u(\boldsymbol{a}_{n(e,i)})\ \Phi^N_{(K_e,i)}\circ\boldsymbol{\varphi}^{\text{geo}}_{K'}(\hat{\boldsymbol{r}}_q) \tag{1.39}$$

$$= \sum_{K'\in\mathcal{T}'_\delta}\sum_{q=1}^{N_q} w_q J_{K'}(\hat{\boldsymbol{r}}_q)\sum_{i=1}^{T_N} u(\boldsymbol{a}_{n(e,i)})\ \hat{\Phi}^N_i\circ\left(\boldsymbol{\varphi}^{\text{geo}}_{K_e}\right)^{-1}\circ\boldsymbol{\varphi}^{\text{geo}}_{K'}(\hat{\boldsymbol{r}}_q) \tag{1.40}$$

Pour illustrer la transparence du langage FEEL++, l'intégrale de la fonction sur un domaine quelconque s'écrit simplement :

```
integrate(_range=elements(mesh),
          _expr=idv(u)).evaluate();
```

Dans cette expression, la librairie FEEL++ va automatiquement appliquer la formule d'intégration adéquate selon le cas que le maillage `mesh` et le maillage de définition de la fonction `u` sont identiques ou pas.

Opérateur linéaire d'interpolation

Les méthodes de localisation sont assez coûteuses dans la pratique, il faut donc essayer de minimiser leurs utilisations. En effet, la description précédente sur l'intégration de fonctions nous a montré qu'il fallait utiliser l'outil de localisation sur chaque point de quadrature. Ainsi, cette utilisation peut devenir très importante pour des schémas de quadrature d'ordre relativement élevé. Nous allons maintenant définir l'opérateur linéaire d'interpolation sous sa forme algébrique, c'est-à-dire qu'il peut être représenté par une matrice A. L'interpolé $v = \pi(u)$ s'écrira simplement sous la forme d'un produit matrice-vecteur, $v = \pi(u) = Au$. De plus, les coefficients de la matrice A sont totalement indépendantes des valeurs de u et sa structure est creuse Ainsi, lors d'une utilisation répétée, le calcul de l'interpolé est ramené à un produit matrice-vecteur ce qui réduit considérablement son coût de calcul.

On considère deux maillages $\mathcal{T}^1_\delta$ et $\mathcal{T}^2_\delta$ représentant respectivement deux domaines de calcul Ω^1_δ et Ω^2_δ tels que $\Omega^2_\delta \subset \Omega^1_\delta$. On choisit u^1_δ (resp u^2_δ) une fonction appartenant à $P_c^{N^1}(\Omega^1_\delta)$ (resp $P_c^{N^2}(\Omega^2_\delta)$) dont l'élément fini de Lagrange de référence admet T_{N^1} (resp T_{N^2}) degrés de liberté. On suppose que u^1_δ (resp u^2_δ) contient N^1_{dof} (resp N^2_{dof}) degrés de liberté repartis sur les points $\left\{\boldsymbol{a}_i, i = 1, ..., N^1_{dof}\right\}$ (resp $\left\{\boldsymbol{b}_i, i = 1, ..., N^2_{dof}\right\}$). On pose également $n^1(.,.)$ (resp $n^2(.,.)$) la table des degrés de liberté associée à $P_c^{N1}(\Omega^1_\delta)$ (resp $P_c^{N2}(\Omega^2_\delta)$). Soit $K^1 \in \mathcal{T}^1_\delta$ (resp $K^2 \in \mathcal{T}^2_\delta$) et j un indice local, on note $\boldsymbol{a}^{K^1}_j = \boldsymbol{a}_{n^1(K^1,j)}$ (resp $\boldsymbol{a}^{K^2}_j = \boldsymbol{a}_{n^2(K^2,j)}$).

On admet que pour tout $i \in \left\{1, ..., N^1_{dof}\right\}$ la valeur au degré de liberté $u(\boldsymbol{a}_i)$ est connue. L'objectif est de définir u^2_δ comme l'interpolé de Lagrange de u^1_δ. Cela revient à trouver les valeurs aux degrés de liberté $u(\boldsymbol{b}_i)$ de la fonction u^2_δ.

Nous commençons par présenter la méthode pour un degré de liberté. Nous souhaitons interpoler le point $\boldsymbol{b}_1^{K^2}$ à partir de la fonction u_δ^1. Pour cela, nous devons d'abord localiser le degré de liberté $\boldsymbol{b}_1^{K^2}$ dans le maillage $\mathcal{T}_\delta^1$. On note K^1 l'élément contenant ce point. Nous illustrons cet exemple avec la figure 1.5. La valeur de l'interpolé à ce degré de liberté est ensuite donnée par l'expression suivante :

$$\begin{aligned} u_\delta^2(\boldsymbol{b}_i^{K^2}) &= \sum_{j=1}^{T_{N^1}} u_\delta^1\left(\boldsymbol{a}_j^{K^1}\right)\ \Phi_{(K^1,j)}^{N^1}\left(\boldsymbol{b}_i^{K^2}\right) \\ &= \sum_{j=1}^{T_{N^1}} u_\delta^1\left(\boldsymbol{a}_j^{K^1}\right) \hat{\Phi}_i^{N^1} \circ \left(\boldsymbol{\varphi}_{K^1}^{\text{geo}}\right)^{-1}\left(\boldsymbol{b}_i^{K^2}\right) \end{aligned}$$

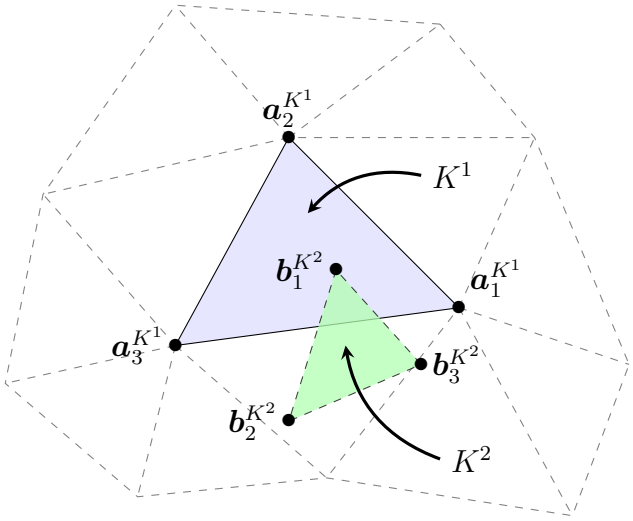


FIGURE 1.5 – Localisation du degré de liberté $\boldsymbol{b}_1^{K^2}$ dans le maillage $\mathcal{T}_\delta^1$. Le point est localisé dans l'élément K^1.

Nous allons maintenant définir les entrées non nulles de A de taille $N_{dof}^2 \times N_{dof}^1$. Pour chaque degré de liberté $\boldsymbol{b}_i$, on se donne un couple (K^2,l) tel que $\boldsymbol{b}_i = \boldsymbol{b}_l^{K^2}$. On appelle ensuite l'outil de localisation pour connaitre l'élément $K^1 \in \mathcal{T}_\delta^1$ tel que $\boldsymbol{b}_i \in K^1$. Nous avons alors $\forall k \in \{1, ..., T_{N^1}\}$:

$$A(i,j) = A\left(n^2(K^2,l), n^1(K^1,k)\right) \tag{1.41}$$

$$= \Phi_{(K^1,k)}^{N^1}\left(\boldsymbol{b}_l^{K^2}\right) \tag{1.42}$$

$$= \hat{\Phi}_k^{N^1} \circ \left(\boldsymbol{\varphi}_{K^1}^{\text{geo}}\right)^{-1}\left(\boldsymbol{b}_l^{K^2}\right) \tag{1.43}$$

Dans beaucoup de cas, nous avons uniquement besoin de calculer l'interpolé sur un bord de son domaine d'étude. Par exemple en fluide-structure, nous voulons effectuer cette opération uniquement sur l'interface fluide-structure. Dans ce cas, on parcourt uniquement les degrés de liberté $\boldsymbol{b}_i$ qui appartiennent à la zone désirée.

Avec la librairie Feel++, l'opérateur linéaire d'interpolation entre deux espaces de fonction `Xh` et `Yh` s'écrit :

```
auto Xh = Pch<N1>(mesh1); auto Yh = Pch<N2>(mesh2);
auto opI = opInterpolation(_domainSpace=Xh,_imageSpace=Yh );
```

La matrice A est calculée et elle est stockée dans l'objet `opI`. On peut ensuite calculer l'interpolé de Lagrange `u2` $\in$ `Yh` d'une fonction `u1` $\in$ `Xh` en appliquant le produit matrice vecteur (`u2`=A`u1`) :

```
auto u1 = vf::project(_space=Xh,_expr=cos(pi*Px()));
auto u2 = Yh->element();
opI->apply(u1,u2);
```

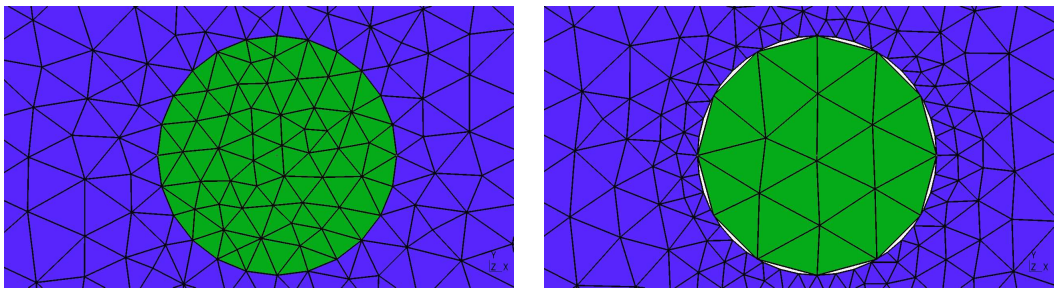
Dans certains cas, on souhaite seulement interpoler sur une partie ou un bord d'un domaine. Une option supplémentaire à l'opérateur d'interpolation est disponible :

```
auto opI = opInterpolation(_domainSpace=Xh,_imageSpace=Yh,
                           _range=markedfaces(mesh2,"interface"));
```

Ceci permet d'accélérer la construction de la matrice A. Beaucoup moins d'entrées seront construites et l'outil de localisation sera moins utilisé. Toutes les valeurs hors de cette interface seront mises à zéro.

Cas particulier : interpolation avec extrapolation

Dans cette partie, nous montrons un cas où la méthode de localisation échoue, car certains points à localiser ne se trouvent pas dans le domaine de recherche. Pour pallier à cette difficulté, nous avons alors recours à l'extrapolation.



(a) Interface conforme (b) Interface non conforme

Figure 1.6 – Exemples de discrétisation avec une interface conforme et non conforme

La figure 1.6(a) montre une interface conforme entre deux maillages. Chaque point de l'interface peut-être localisé dans les deux maillages. La figure 1.6(b) représente par contre une interface non conforme dans laquelle certains points ne pourront être localisés. Ce problème est dû à la discrétisation de chacun des domaines bleu et vert. Dans ce cas particulier, on peut tenter d'utiliser une technique d'extrapolation.

Soit Ω_δ^1 (resp Ω_δ^2) le domaine représenté en vert (resp bleu) sur la figure 1.6(b). On considère le point $\boldsymbol{x} \in \partial\Omega_\delta^1$ tel que $\boldsymbol{x} \notin \Omega_\delta^2$. Ce point ne peut donc pas être localisé. On pose K^1 un élément appartenant au domaine Ω_δ^1 tel que $\boldsymbol{x} \in K^1$. Dans le cas où la localisation fonctionne bien, on obtient un élément $K^2 \in \Omega_\delta^2$ et on évalue ensuite les fonctions de base de l'élément K^2 : $\Phi^N_{K^2,j}(\boldsymbol{x})$. Quand la localisation échoue, on utilise la stratégie suivante :

on cherche l'élément de Ω_δ^2 le plus proche de $\boldsymbol{x}$, noté $\tilde{K}^2$ et on extrapole les fonctions de bases définies sur cet élément :

$$\Phi^N_{\tilde{K}^2,j}(\boldsymbol{x}) = \hat{\Phi}^N_j \circ \left(\boldsymbol{\varphi}^{\text{geo}}_{\tilde{K}^2}\right)^{-1}(\boldsymbol{x}) \tag{1.44}$$

Cette technique fonctionne assez bien lorsque l'élément $\tilde{K}^2$ est très proche de $\boldsymbol{x}$. C'est généralement le cas quand il s'agit juste d'un problème de discrétisation géométrique. Pour des distances plus importantes, d'autres techniques devraient être alors envisagées.

2.2 Méthode de Galerkin non standard

Nous allons dans cette partie présenter la construction des structures algébriques représentant un problème discret approché par des méthodes d'approximation non standards. Ces méthodes peuvent intervenir lorsque des conditions aux limites sont imposées par des multiplicateurs de Lagrange ou encore pour des méthodes numériques complexes comme la méthode de la frontière élargie décrite dans le chapitre 3. Nous commencerons par la définition de ces types de problème, puis nous parlerons de la construction des opérateurs algébriques associés aux formes linéaires et bilinéaires. Nous nous baserons sur un problème type et tout au long de cette description, nous mettrons en lumière le code FEEL++ sous-jacent.

Soit X_h et Y_h deux espaces de fonction produits de dimension finie et $u \in X_h$:

$$X_h = \prod_{i=1}^{N_{xc}} X_h^i \quad , \quad Y_h = \prod_{i=1}^{N_{yc}} Y_h^i \quad , \quad u = (u_1, ..., u_{N_{xc}}) \tag{1.45}$$

À chaque sous-espace X_h^i (resp Y_h^i) est associé un maillage $\mathcal{T}^i_{X_h}$ (resp $\mathcal{T}^i_{Y_h}$) qui forme un domaine de calcul $\Omega^i_{X_h}$ (resp $\Omega^i_{Y_h}$). La méthode de Galerkin permet d'approcher des problèmes discrets dont la formulation abstraite est la suivante :

$$\begin{cases} \text{Chercher } u \in X_h \text{ tel que pour } i = 0...N_{yc} : \\ \displaystyle\sum_{j=1}^{N_{xc}} a_i^j(u_j, w_i) = f_i(w_i), \quad \forall w_i \in Y_h^i \end{cases} \tag{1.46}$$

où a_i^j est une forme bilinéaire définie sur $X_h^j \times Y_h^i$ et f_i est une forme linéaire sur Y_h^i. Nous parlerons alors de méthode de Galerkin non standard lorsque l'une de ces hypothèses sera vérifiée :

- X_h et Y_h sont différents.
- $\exists (i,j)$ tel que $\mathcal{T}^i_{X_h} \neq \mathcal{T}^j_{Y_h}$.
- $\exists (i,j)$ tel que a_i^j contient une intégrale $\int_{\mathcal{T}_q} ...$ avec $\left(\mathcal{T}_q \neq \mathcal{T}^j_{X_h} \vee \mathcal{T}_q \neq \mathcal{T}^i_{Y_h}\right)$.
- $\exists\ i$ tel que f_i contient une intégrale $\int_{\mathcal{T}_q} ...$ avec $\mathcal{T}_q \neq \mathcal{T}^i_{Y_h}$.

Nous allons maintenant présenter une construction des formes bilinéaires a_i^j et linéaires f_i dans un cas non standard. Pour la suite de cette partie, le problème (1.47) avec sa géométrie en figure 2.2 sera pris comme référence pour les notations. Nous tâcherons de décrire entièrement la construction de ce problème.

Soit $\mathcal{T}_\delta^\Omega$ (resp $\mathcal{T}_\delta^\omega$) le maillage associé à Ω (resp ω) et formant le domaine de calcul Ω_δ (resp ω_δ). On note $\mathcal{T}_\delta^\Gamma$ le maillage frontière de $\partial\Omega$ mais non-conforme avec $\mathcal{T}_\delta^\Omega$, il est pris plus grossier. Le domaine recouvert par $\mathcal{T}_\delta^\Gamma$ est noté Γ_δ. On défini ensuite deux espaces de fonctions $U_\delta \in P_c^N(\Omega_\delta)$ et $L_\delta \in P_c^N(\Gamma_\delta)$. Nous posons $X_\delta = U_\delta \times L_\delta$ un espace de fonctions composite dans lequel nous allons chercher la solution. Nous admettons que $N_{dof}^{U_\delta}$ (resp $N_{dof}^{L_\delta}$) est le nombre de degrés de liberté de l'espace U_δ (resp L_δ). Nous considérons que $\left\{\Phi_i^{U_\delta}, i = 1, ..., N_{dof}^{U_\delta}\right\}$ (resp $\left\{\Phi_i^{L_\delta}, i = 1, ..., N_{dof}^{L_\delta}\right\}$) est l'ensemble des fonctions de bases de l'espace U_δ (resp L_δ) et on associe également une table des degrés de liberté notés $n^{U_\delta}(.,.)$ (resp $n^{L_\delta}(.,.)$).

Problème de référence

Pour illustrer la construction de ces termes non standard, nous considérons le problème suivant :

$$\begin{cases} \text{Chercher } U = (u, l) \in X_\delta \text{ tel que} & & \\ a(u, v) + b(l, v) = f(v), & \forall v \in U_h & (1.47) \\ b(u, m) = 0, & \forall m \in L_h & \end{cases}$$

$$\text{avec : } a(u, v) = \int_{\Omega_\delta} \nabla u \cdot \nabla v - \int_{\partial\Omega_\delta} (\nabla u \cdot \boldsymbol{n})\, v$$

$$b(l, v) = \int_{\Gamma_\delta} l\ v$$

$$f(v) = \int_{\omega_\delta} g\ v \qquad \text{avec } g \in L^2(\omega_\delta) \text{ donnée}$$

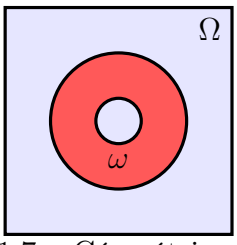


FIGURE 1.7 – Géométrie du problème de référence.

Ce problème correspond à la formulation variationnelle d'un problème elliptique avec une condition de Dirichlet sur Γ_δ imposé à l'aide d'un multiplicateur de Lagrange et un terme source défini localement sur le domaine ω_δ. Les termes non standards apparaissent à cause de la non-conformité entre Γ_δ, Ω_δ et ω_δ.

Forme linéaire

On considère la forme linéaire $f(v) = \int_{\omega_\delta} g\ v$. D'un point de vue algébrique, f correspond à un vecteur F de taille $N_{dof}^{U_\delta}$. On choisit la notation $\left\{(\boldsymbol{r}_q^K, w_q), q = 1..N_q\right\}$ pour décrire une formule de quadrature sur chaque $K \in \mathcal{T}_\delta^\omega$. Le point $\boldsymbol{r}_q^K$ représente un point de quadrature et w_q le poids associé. Chaque composante s'écrit par définition $F_i = f(\Phi_i^{U_\delta})$. D'où l'on peut développer :

$$f(\Phi_i^{U_\delta}) = \int_{\omega_\delta} g(\boldsymbol{x})\ \Phi_i^{U_\delta}(\boldsymbol{x})\ d\boldsymbol{x} \tag{1.48}$$

$$= \sum_{K \in \mathcal{T}_\delta^\omega} \int_K g(\boldsymbol{x})\ \Phi_i^{U_\delta}(\boldsymbol{x})\ d\boldsymbol{x} \tag{1.49}$$

$$= \sum_{K \in \mathcal{T}_\delta^\omega} \sum_{q=1}^{N_q} w_q\ g(\boldsymbol{r}_q^K)\ \Phi_i^{U_\delta}(\boldsymbol{r}_q^K) \tag{1.50}$$

Mais comme les fonctions de base globales $\Phi_i^{U_\delta}$ ne sont pas construites, on ne peut pas évaluer cette intégrale. Il nous faut exprimer $f(\Phi_i^{U_\delta})$ en fonction des fonctions de base locales $\Phi_{(K',m)}^{U_\delta}$ avec $K' \in \mathcal{T}_\delta^\Omega$ et m un indice local. Ainsi, $\forall K \in \mathcal{T}_\delta^\omega$, nous avons besoin

de localiser l'ensemble des points $\boldsymbol{r}_q^K$ dans le maillage $\mathcal{T}_\delta^\Omega$. La figure 1.8 nous illustre un exemple de points de quadrature sur un élément qu'il faut localiser. Pour la suite de notre construction, nous avons besoin de définir les ensembles suivants :

$$\begin{aligned} S_i &= \left\{ (K', m) \in \mathcal{T}_\delta^\Omega \times \left\{ 1, ..., T_N^{U_\delta} \right\} \text{ tel que } i = n^{U_\delta}(K', m) \right\} \\ R_{K'} &= \left\{ (K, q) \in \mathcal{T}_\delta^\omega \times \{1, ..., N_q\} \text{ tel que } \boldsymbol{r}_q^K \in K' \right\} \end{aligned} \tag{1.51}$$

On peut remarquer que la construction des ensembles $R_{K'}$ nécessite l'utilisation de l'outil de localisation. Ces ensembles permettent de répartir les points de quadrature sur chaque élément $K \in \mathcal{T}_\delta^\Omega$. Les contributions élémentaires correspondront à l'application d'une partie de la formule. Cette partie représente les points de quadrature présents dans l'élément.

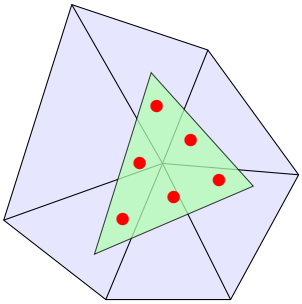


FIGURE 1.8 – Points d'intégration et intersection des maillages pour une forme linéaire. En bleu quelques éléments de $\mathcal{T}_\delta^\omega$, en vert un élément $K \in \mathcal{T}_\delta^\Omega$ et en rouge les points d'intégration sur K.

À l'aide de ces définitions, l'intégration pour la forme linéaire peut être maintenant accomplie. Cela se traduit par les expressions suivantes :

$$\begin{aligned} f(\Phi_i^{U_\delta}) &= \sum_{(K',m)\in S_i} \sum_{(K,q)\in R_{K'}} w_q \; g(\boldsymbol{r}_q^K) \; \Phi_i^{U_\delta}(\boldsymbol{r}_q^K) \\ &= \sum_{(K',m)\in S_i} \sum_{(K,q)\in R_{K'}} w_q \; g(\boldsymbol{r}_q^K) \; \Phi_{(K',m)}^{U_\delta}(\boldsymbol{r}_q^K) \end{aligned} \tag{1.52}$$

Cependant, la méthode que nous utilisons correspond à une approximation du calcul d'intégrale. Nous ne pouvons pas calculer l'intégrale exacte, mais nous pouvons améliorer la précision en augmentant l'ordre du schéma de quadrature. Pour réaliser un calcul exact, il faudrait créer un maillage représentant $\omega_\delta \cap \{K\}_{K\in\mathcal{T}_\delta^\Omega}$. En d'autres termes, chaque élément de ce maillage appartiendrait uniquement à un élément de $\mathcal{T}_\delta^\Omega$ et le domaine recouvert par ce maillage serait égal à ω_δ. Toutefois, cette opération peut se révéler complexe et coûteuse si le maillage est d'ordre élevé.

L'assemblage d'une telle forme linéaire s'écrit en FEEL++ de manière naturelle par :

```
auto F1 = backend->newVector(Uh);
form1(_test=Uh, _vector=F1) +=
    integrate(_range=elements(omega),
              _expr=g*id(v),_quad=_Q<15>() );
```

Le langage détecte que le maillage de définition de l'espace de fonction `Uh` n'est pas le même que le maillage d'intégration `omega`. Il utilise ainsi la méthode décrite précédemment. L'option `_quad` permet d'utiliser un ordre de quadrature spécifique (ici 15) qui correspond à l'ordre polynomial maximal tel que l'intégration numérique soit exacte. Cette option n'est pas obligatoire, mais dans ce genre de cas non conforme, il est préférable de la préciser pour obtenir une meilleure précision. La quadrature ne doit pas être d'ordre trop élevé, car cela nuirait considérablement aux performances. En effet, pour chaque point de quadrature, il faut utiliser l'outil de localisation.

Forme bilinéaire

Nous décrivons maintenant la forme bilinéaire $b(l,v) = \int_{\Gamma_\delta} l\ v$. La représentation algébrique de cet opérateur correspond à une matrice A de taille $N_{dof}^{U_\delta} \times N_{dof}^{L_\delta}$. Ici, le maillage d'intégration est le même que celui de l'espace L_h. Pour pouvoir décrire le cas le plus général, nous choisissons de définir une autre discrétisation de la frontière $\partial\Omega$, noté $\mathcal{T}_{\delta'}^\Gamma$. Nous aurons alors trois maillages différents et non-conforme qui interviendront dans une intégrale : $\mathcal{T}_{\delta'}^\Gamma$ le maillage d'intégration, $\mathcal{T}_\delta^\Gamma$ (resp $\mathcal{T}_\delta^\Omega$) le domaine de définition des fonctions tests (resp solutions). On pose Γ'_δ le domaine recouvert par la discrétisation $\mathcal{T}_{\delta'}^\Gamma$. Comme précédemment, nous utilisons la notation $\left\{(\boldsymbol{r}_q^K, w_q), q = 1..N_q\right\}$ pour décrire une formule de quadrature sur chaque $K \in \mathcal{T}_{\delta'}^\Gamma$. Chaque composante de la matrice A s'écrit par définition $A_{ij} = b(\Phi_j^{L_\delta}, \Phi_i^{U_\delta})$, d'où :

$$\begin{aligned} b(\Phi_j^{L_\delta}, \Phi_i^{U_\delta}) &= \int_{\Gamma'_\delta} \Phi_j^{L_\delta}(\boldsymbol{x})\ \Phi_i^{U_\delta}(\boldsymbol{x})\ d\boldsymbol{x} \\ &= \sum_{K\in\mathcal{T}_{\delta'}^\Gamma} \int_K \Phi_j^{L_\delta}(\boldsymbol{x})\ \Phi_i^{U_\delta}(\boldsymbol{x})\ d\boldsymbol{x} \\ &= \sum_{K\in\mathcal{T}_{\delta'}^\Gamma} \sum_{q=1}^{N_q} w_q \Phi_j^{L_\delta}(\boldsymbol{r}_q^K)\ \Phi_i^{U_\delta}(\boldsymbol{r}_q^K) \end{aligned}$$

De manière similaire à la forme linéaire, nous cherchons à exprimer $b(\Phi_j^{L_\delta}, \Phi_i^{U_\delta})$ en fonction des fonctions de base locales $\Phi_{(K'_1,m)}^{U_\delta}$ et $\Phi_{(K'_2,n)}^{L_\delta}$, avec $K'_1 \in \mathcal{T}_\delta^\Omega$, $K'_2 \in \mathcal{T}_\delta^\Gamma$ et m et n des indices locaux. Nous considérons alors les ensembles suivants :

$$\begin{aligned} S_i &= \left\{(K'_1, m) \in \mathcal{T}_\delta^\Omega \times \left\{1, ..., T_N^{U_\delta}\right\} \text{ tel que } i = n^{U_\delta}(K'_1, m)\right\} \\ T_j &= \left\{(K'_2, n) \in \mathcal{T}_\delta^\Gamma \times \left\{1, ..., T_N^{L_\delta}\right\} \text{ tel que } i = n^{L_\delta}(K'_2, n)\right\} \\ R(K'_1, K'_2) &= \left\{(K, q) \in \mathcal{T}_{\delta'}^\Gamma \times \{1, ..., N_q\} \text{ tel que } \boldsymbol{r}_q^K \in K'_1 \text{ et } \boldsymbol{r}_q^K \in K'_2\right\} \end{aligned} \tag{1.53}$$

Ces ensembles nous permettent de rassembler les points de quadratures appartenant à la fois à un élément $K^1 \in \mathcal{T}_\delta^\Omega$, $K^2 \in \mathcal{T}_\delta^\Gamma$ et $K' \in \mathcal{T}_{\delta'}^\Gamma$. Ainsi, nous pouvons utiliser les fonctions de base locales de chaque espace U_δ et L_δ mais également les transformations géométriques de chacun des éléments. L'outil de localisation est utilisé pour localiser les points de quadrature dans les deux maillages $\mathcal{T}_\delta^\Omega$ et $\mathcal{T}_\delta^\Gamma$. Le coût de construction de ces ensembles peut devenir très important. Nous avons représenté avec la figure 1.9 la superposition des maillages qui pourrait apparaitre dans cette application. Pour plus de clarté, les maillages $\mathcal{T}_\delta^\Gamma$ et $\mathcal{T}_{\delta'}^\Gamma$ sont représentés par des éléments 2d mais en réalité, ils devraient être constitués d'éléments 1D.

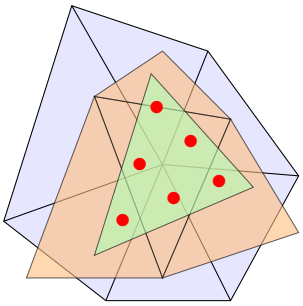


FIGURE 1.9 – Points d'intégrations et intersection des maillages pour une forme bilinéaire. En bleu quelques éléments de $\mathcal{T}_\delta^\Omega$, en rose quelques éléments de $\mathcal{T}_\delta^\Gamma$, en vert un élément K de $\mathcal{T}_{\delta'}^\Gamma$ et en rouge les points d'intégration sur K.

Nous ramenons l'intégrale de la forme bilinéaire sur ces ensembles de définition, ce qui permet d'exprimer :

$$\begin{aligned} b(\Phi_j^{L_\delta}, \Phi_i^{U_\delta}) &= \sum_{(K_1',m)\in S_i} \sum_{(K_2',n)\in T_j} \sum_{(K,q)\in R(K_1',K_2')} w_q \; \Phi_j^{L_\delta}(\boldsymbol{r}_q^K) \; \Phi_i^{U_\delta}(\boldsymbol{r}_q^K) \\ &= \sum_{(K_1',m)\in S_i} \sum_{(K_2',n)\in T_j} \sum_{(K,q)\in R(K_1',K_2')} w_q \; \Phi_{(K_2',n)}^{L_\delta}(\boldsymbol{r}_q^K) \; \Phi_{(K_1',m)}^{U_\delta}(\boldsymbol{r}_q^K) \end{aligned} \tag{1.54}$$

Comme pour la forme linéaire, l'intégrale que nous construisons est approchée en raison de la non-conformité des maillages. Malheureusement, aucune référence sur l'analyse de ce type d'intégration n'a été trouvée. D'un point de vue pratique, nous pouvons améliorer la précision de l'approximation numérique en utilisant un schéma de quadrature d'ordre « assez » élevé. Ce choix permet de mieux assurer la prise en compte des contributions pour les éléments se superposant, figure 1.9. En effet, ce sont les points de quadrature qui permettent de localiser les éléments nécessaires à l'intégration. Ainsi, plus l'ordre de quadrature est élevé, plus il y a de points de quadrature et donc plus il y aura de chance de repérer tous les éléments nécessaires. Pour un calcul exact des intégrales, il faudrait être capable de générer l'intersection des maillages, comme nous l'avons évoqué pour la forme linéaire.

Nous illustrons encore une fois le code FEEL++ associé à la construction des formes bilinéaires qui reste très proche de la formulation mathématique :

```
auto A21 = backend->newMatrix(_test=Lh, _trial=Uh);
form2(_test=Lh, _trial=Uh, _matrix=A21) +=
    integrate(_range=elements(GammaPrime),
              _expr= idt(u)*id(m),_quad=_Q<15>() );
auto A12 = backend->newMatrix(_test=Uh, _trial=Lh);
form2(_test=Uh, _trial=Lh, _matrix=A12) +=
    integrate(_range=elements(GammaPrime),
              _expr= idt(l)*id(v),_quad=_Q<15>() );
```

Nous imposons également ici un ordre de quadrature à utiliser pour le calcul des intégrales. On peut aussi remarquer que la matrice `A12` est égale à la transposée de `A21`. Ainsi, nous aurions pu simplement écrire pour définir la matrice `A12` le code suivant :

```
auto A12 = A21->transpose();
```

La dernière forme bilinéaire $a(u,v) = \int_{\Omega_\delta} \nabla u \cdot \nabla v - \int_{\partial\Omega_\delta} (\nabla u \cdot \boldsymbol{n})\, v$, correspond au cas standard. L'ordre de quadrature pour cette forme est automatiquement calculé avec Feel++ de manière à avoir une formule de quadrature exacte. Ce terme correspond au code :

```
auto A11 = backend->newMatrix(_test=Uh, _trial=Uh);
form2(_test=Uh, _trial=Uh, _matrix=A11) +=
    integrate(_range=elements(Omega),
              _expr= gradt(u)*trans(grad(v)) ) +
    integrate(_range=boundaryfaces(Omega),
              _expr=-gradt(u)*N()*id(v) );
```

Assemblage de la matrice par blocs

Nous venons de construire les structures algébriques représentant une décomposition des formes linéaires et bilinéaires associées au problème (1.47). Celles-ci sont définies sur les sous-espaces U_δ et L_δ. Nous voulons maintenant assembler la matrice A et le vecteur F correspondant au problème complet. Le système global s'écrit $AU = F$ avec la structure algébrique par bloc suivante :

$$\underbrace{\begin{pmatrix} A_{11} & B_{12} \\ B_{21} & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \mathrm{U}_u \\ \mathrm{U}_l \end{pmatrix}}_{U} = \underbrace{\begin{pmatrix} F_1 \\ 0 \end{pmatrix}}_{F} \tag{1.55}$$

Le vecteur U représente la solution de ce système algébrique et il contient l'ensemble des valeurs de la solution aux degrés de liberté de l'espace X_δ. Il se décompose en deux sous vecteurs U_u et U_l qui sont les représentations algébriques des deux sous solutions $u \in U_\delta$ et $l \in L_\delta$. Cette méthode de résolution du problème (1.47) est dite monolithique, car les solutions u et l sont obtenues simultanément.

Avec notre outil Feel++, nous avons deux possibilités pour assembler la matrice A et le vecteur F. Une première qui commence par construire chaque matrice bloc puis qui génère la matrice globale à partir des matrices blocs. La deuxième méthode permet de construire directement la matrice globale. Nous allons ainsi présenter les deux constructions en donnant à chaque fois le code informatique associé.

Construction 1 : Les sous-matrices (A_{11}, B_{12}, B_{21}) et le sous-vecteur F_1 sont construits en utilisant le code présenté dans la section précédente. Il nous faut aussi définir la matrice bloc nulle A_{22} de taille $N_{dof}^{L_\delta} \times N_{dof}^{L_\delta}$ et le vecteur nul F_2 de taille $N_{dof}^{L_\delta}$:

```
auto A22 = backend->newZeroMatrix( _trial=Lh, _test=Lh );
auto F2 = backend->newZeroVector( _test=Lh );
```

La matrice globale A et le vecteur F sont ensuite construits en rassemblant chaque bloc défini précédemment. Cet assemblage se traduit par :

```
auto blockMat = BlocksSparseMatrix<2,2>() << A11 << A12
                                          << A21 << A22 ;
auto A = backend->newBlockMatrix(_block=blockMat);

auto blockVec = BlocksVector<2,1>() << F1 << F2;
auto F = backend->newBlockVector(_block=blockVec);
```

Construction 2 : Une seule matrice A et un seul vecteur F sont construits. On utilise la connaissance de la structure de A avec les indices lignes et colonnes de début de chaque bloc. On commence par la structure des entrées non nulles de A :

```
auto pattern = BlocksStencilPattern(2,2)
    << size_type(Pattern::COUPLED) << size_type(Pattern::COUPLED)
    << size_type(Pattern::COUPLED) << size_type(Pattern::ZERO);

auto A = backend->newMatrix(_trial=Xh,_test=Xh,
                            _pattern_block=pattern);
auto F =  backend->newVector(_test=Xh);
```

Ensuite, on assemble toutes les contributions dans la matrice A :

```
form2(_test=Uh,_trial=Uh,_matrix=A,
      _rowstart=0,_colstart=0 )
  += integrate(_range=elements(Omega),
               _expr= gradt(u)*trans(grad(v)) )
  +  integrate(_range=boundaryfaces(Omega),
               _expr=-gradt(u)*N()*id(v) );

form2(_test=Uh,_trial=Lh,_matrix=A,
      _rowstart=0,_colstart=Uh->nDof())
  += integrate(_range=elements(GammaPrime),
               _expr=idt(l)*id(v),_quad=_Q<15>() );

form2(_test=Lh,_trial=Uh,_matrix=A,
      _rowstart=Uh->nDof(),_colstart=0 )
  += integrate(_range=elements(GammaPrime),
               _expr=idt(u)*id(m),_quad=_Q<15>() );
```

et dans le vecteur F :

```
form1(_test=Uh, _vector=F, _rowstart=0) +=
    integrate(_range=elements(omega),
              _expr= g*id(v),_quad=_Q<15>() );
```

Remarque : La deuxième méthode a l'avantage de déclarer une seule matrice (gain en mémoire) et il n'y a pas de recopie (gain en performance).

Application numérique

Nous avons ici implémenté le problème (1.47) avec une géométrie illustrée avec la figure 1.10. Le barycentre de Ω se trouve au point $(0.5, 0.5)$. L'inconnue l joue le rôle de multiplicateur de Lagrange. Il impose la contrainte $u = 0$ sur le bord de Ω. Quant à g, elle représente un terme source agissant uniquement sur ω. Nous choisissons de prendre :

$$g(x, y) = (y - 0.5)^2 \qquad (1.56)$$

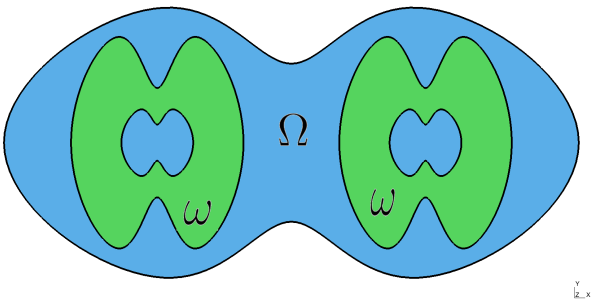


FIGURE 1.10 – Géométrie

La dernière étape de cette présentation est la résolution du système linéaire. On construit également le vecteur global U à partir des inconnues $u \in U_\delta$ et $l \in L_\delta$. Nous avons ainsi le code Feel++ suivant :

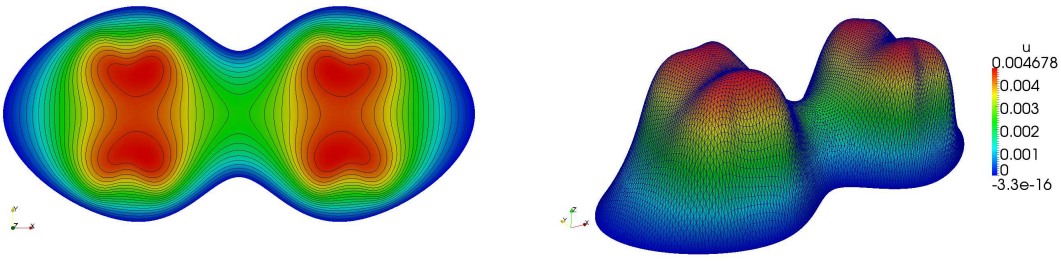
```
auto blockVecU = BlocksVector<2>() << u << l;
auto U = backend->newBlockVector(_block=blockVecU);

backend->solve(_matrix=A,_solution=U,_rhs=F);
```

Enfin, après avoir trouvé la solution U, on met à jour les sous-solutions u et l à l'aide de la fonction suivante :

```
blockVecU.localize(U);
```

Celle-ci permet de récupérer les valeurs de chacun des degrés de liberté de u et l à partir du vecteur algébrique U.

Les figures 1.11(a) et 1.11(b) montrent la solution obtenue. Cet exemple ne nous permet pas de valider rigoureusement notre implémentation, mais elle nous a permis de vérifier les nouveaux algorithmes numériques développés dans la librairie : assemblage de la matrice et du second membre, résolution d'un système non singulier, comportement qualitatif de solution. Toutefois, la vérification de l'ensemble de ces outils sera effectuée au chapitre 3 avec plusieurs tests de convergences en h et p.



(a) Solution u et iso-surfaces (b) Solution u et son élévation (facteur 200)

Figure 1.11 – Solution numérique u obtenue

2.3 Méthodes de localisation

Une partie importante des outils présentés précédemment est la localisation de points dans l'espace. On cherche à localiser un point dans un élément d'un maillage. Cette fonctionnalité est utilisée à de nombreuses reprises dans les outils de localisation. Il est donc important d'optimiser cette recherche. Nous avons choisi d'utiliser une technique relativement classique et efficace de recherche basée sur la structure de données kdtree.

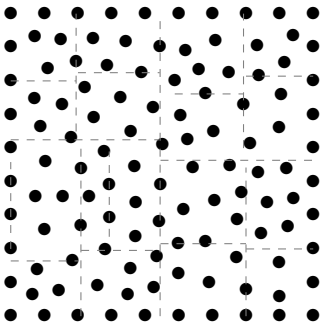
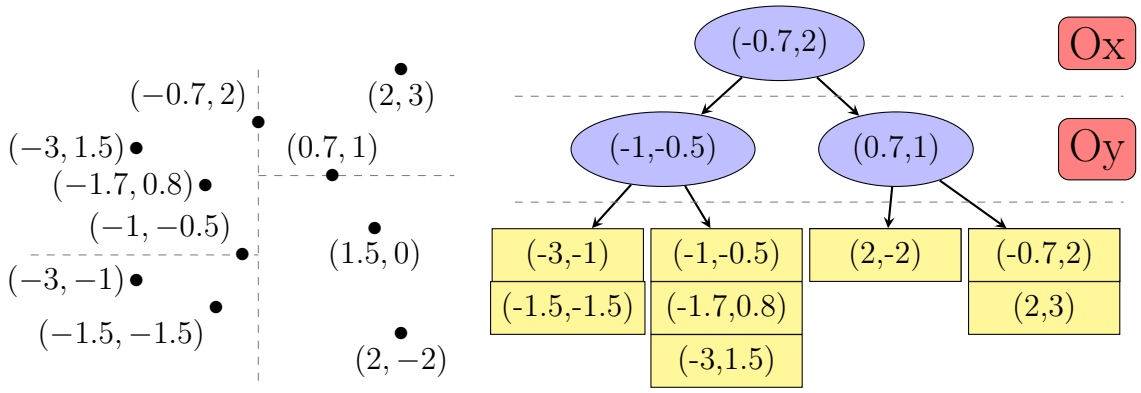


FIGURE 1.12 – Partitionnement de données

Structure de données kdtree

La notion de structure kdtree est basée sur le partitionnement de données. Elle a été initialement introduite dans [19]. Son principe est d'organiser intelligemment les points d'un espace de dimension k pour en accélérer leur traitement comme la recherche des points les plus proches. La structure de données est représentée sous la forme d'un arbre binaire dans lequel chaque nœud contient un point en dimension k. Chaque nœud non terminal divise l'espace en deux demi-espaces. Les points situés dans chacun des deux demi-espaces sont stockés dans les branches gauche et droite du nœud courant. Par exemple, si un nœud donné divise l'espace selon un plan normal à la direction (Ox), tous les points de coordonnée x inférieure à la coordonnée du point associé au nœud seront stockés dans la branche gauche du nœud. De manière similaire, les points de coordonnée x supérieure à celle du point considéré seront stockés dans la branche droite du nœud.



(a) Organisation des points de l'espace (b) Arbre kdtree

FIGURE 1.13 – Principe de la méthode kdtree

Il existe plusieurs possibilités de construction d'arbres kdtree [20, 158, 68, 107], notamment au niveau du choix des axes médians, mais le principe reste le même. À partir d'un nuage de points, on se donne une direction d'un hyperplan. L'axe médian est alors défini grâce au point qui divise en deux d'une manière équilibrée le nuage de point. La direction de l'hyperplan est choisie en fonction de la hauteur du point dans l'arbre. Pour un *kdtree* en dimension $\mathbb{R}^n, n = 1, 2, 3$, on utilise en général les vecteurs de la base canonique de $\mathbb{R}^n$.

En dimension 3 par exemple, le plan de la racine sera défini par un vecteur normal (1,0,0), le plan des deux « enfants » aura un vecteur normal (0,1,0), celui des « petits-enfants » un vecteur normal (0,0,1), puis à nouveau vecteur normal (1,0,0), et ainsi de suite... C'est un processus itératif, dont le critère d'arrêt est le nombre de points restant dans les feuilles de l'arbre. Cela correspond aux cases jaunes du graphique 1.13(b). Ce nombre de points est équivalent au nombre de points inclus dans chaque boite de partitionnement. La complexité algorithmique de construction d'un arbre *kdtree* est de $\mathcal{O}(n \log^2 n)$.

Algorithme de recherche

Nous présentons maintenant notre algorithme de recherche basé sur le partitionnement *kdtree*. À partir d'un point donné P, nous voulons chercher l'élément du maillage auquel ce point appartient. La première étape consiste à construire l'arbre kdtree avec un ensemble de points. Nous avons fait le choix de prendre tous les points géométriques des éléments constituant le maillage. À noter qu'il n'y a pas seulement les sommets dans le cas d'éléments d'ordre élevé, les points géométriques peuvent se trouver sur les faces ou à l'intérieur des éléments. Il faut aussi construire un conteneur très utile pour la suite, une carte de localisation, qui à partir d'un point géométrique nous donne l'ensemble des éléments pour lequel ce point géométrique appartient.

À partir du point de recherche P, nous pouvons utiliser la méthode kdtree pour localiser une boite de partitionnement contenant ce point. À partir de l'ensemble de points géométriques localisés, nous pouvons facilement accéder à une liste d'éléments L_{elt} du maillage contenant le point P. Cette liste est sans doublons. Ensuite, une méthode très simple consisterait simplement à parcourir cette liste et à tester si le point P appartient à l'élément. Cependant, le test d'appartenance à l'élément peut être coûteux, notamment pour les éléments d'ordre élevé, car cela nécessite la résolution d'un problème non linéaire. C'est pourquoi nous choisissons de trier la liste L_{elt} avec un critère de probabilité sur l'appartenance du point P. Pour cela, pour chaque point géométrique, nous comptons le nombre d'occurrences des éléments. Ainsi à la fin de ce processus, nous avons un classement qui nous donne le nombre d'occurrences des éléments contenus dans la liste L_{elt}, ce qui nous permet de trier cette liste.

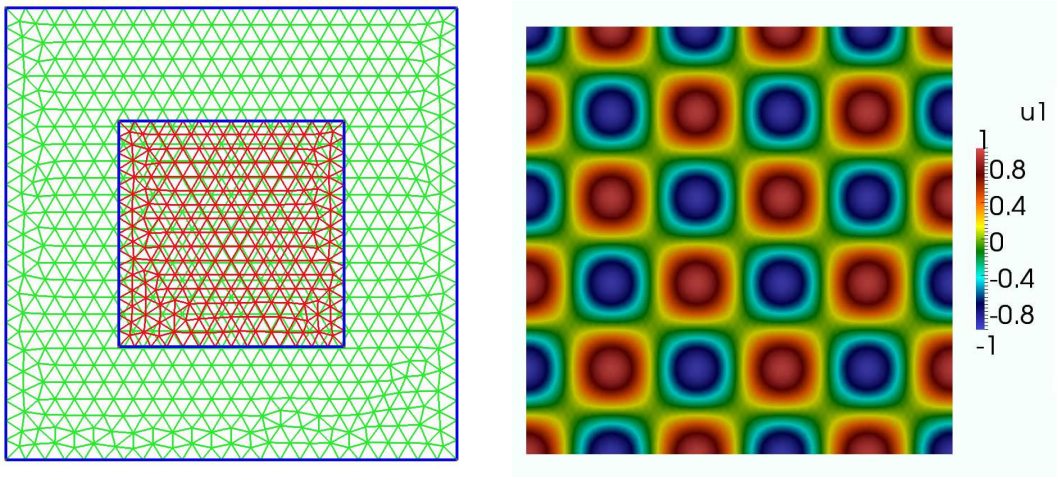
D'autres optimisations, plus mineures, sont aussi effectuées. Par exemple, dans le cas où nous voulons localiser les points des degrés de liberté d'une fonction vectorielle. On peut remarquer que pour les éléments finis de Lagrange, les composantes sont situées sur le même point. Il est alors totalement inutile et coûteux de localiser plusieurs fois le même point. Aussi nous pouvons remarquer que la recherche des points s'effectue souvent de proche en proche. Donc il est utile de sauvegarder le dernier élément localisé et de le tester directement pour le prochain point. Il y a de grandes chances pour qu'il soit le bon.

3 Tests Numériques

Pour clore ce chapitre, nous allons présenter deux applications numériques mettant en jeu l'opérateur d'interpolation décrit dans la partie 2.1. Nous commencerons par une vérification des propriétés théoriques de convergence de cet outil, puis nous illustrerons un exemple relativement simple de décomposition de domaine. Enfin, nous présenterons une application où les conditions aux limites de Dirichlet sont imposées par des multiplicateurs de Lagrange.

3.1 Convergence en h et p

Nous allons commencer par nous intéresser à l'interpolation d'une fonction u_1 définie sur un domaine Ω_1 vers une fonction u_2 définie sur un autre domaine Ω_2, tel que $\Omega_2 \subset \Omega_1$. Le domaine Ω_1 (resp Ω_2) correspond au carré $[0,1] \times [0,1]$ (resp $[0.25, 0.75] \times [0.25, 0.75]$).



(a) Maillages $\mathcal{T}_h^1$ et $\mathcal{T}_h^2$ (b) Fonction analytique f sur Ω_1

FIGURE 1.14 – Domaines de calcul (à gauche) et fonction de référence (à droite).

Le domaine Ω_1 (resp Ω_2) est ensuite discrétisé en un maillage $\mathcal{T}_h^1$ (resp $\mathcal{T}_h^2$) formant un domaine de calcul Ω_δ^1 (resp Ω_δ^1). La figure 1.14(a) illustre un exemple de maillages utilisés et elle montre leur non-conformité. Nous choisissons maintenant de construire la fonction $u_1 \in P_c^N(\Omega_\delta^1)$ comme étant la projection H^1 d'une fonction analytique u, représentée par la figure 1.14(b) :

$$u\,(x,y) = \cos\,(5\pi x)\cos\,(5\pi y) \tag{1.57}$$

Cette projection hérite des propriétés standards d'optimalité de convergence dans un contexte éléments finis. Ce problème classique est formalisé par l'équation suivante :

Trouver la fonction $u_1 \in P_c^N(\Omega_\delta^1)$ telle que $\forall v \in P_c^N(\Omega_\delta^1)$

$$\int_{\Omega_\delta^1} u_1\ v + \nabla u_1 \cdot \nabla v = \int_{\Omega_\delta^1} u\ v + \nabla u \cdot \nabla v$$

Nous souhaitons interpoler la fonction $u_2 \in P_c^N(\Omega_\delta^2)$ à partir de la solution calculée u_1. On pose $u_2 = \pi(u_1)$. La construction décrite dans la partie 2.1 admet des estimations d'erreur a priori que l'on peut trouver par exemple dans [35]. L'erreur d'interpolation aux points de Gauss-Lobatto est estimée par :

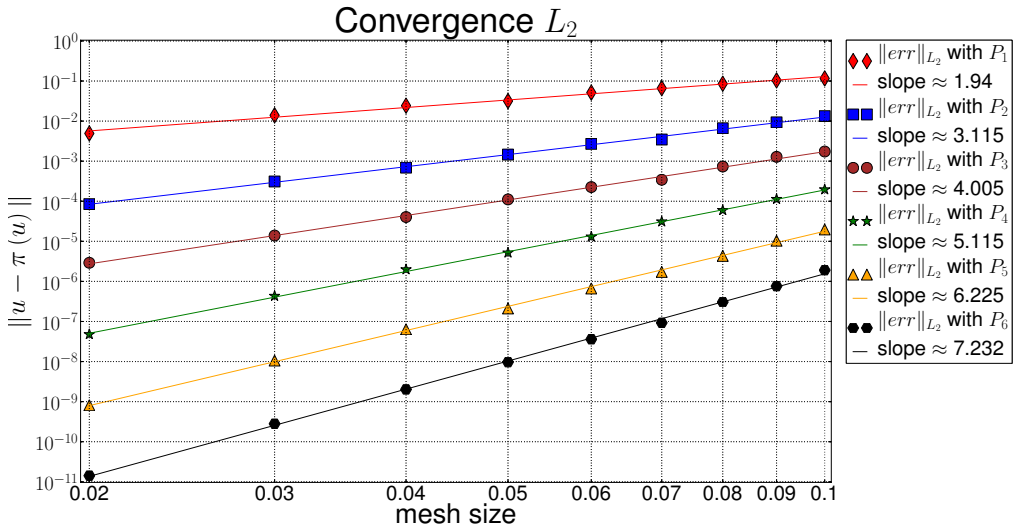
Soit $u \in H^m\,(\Omega_\delta^2)$ avec $m \geq 1$ l'indice de régularité de u et pour $k = 0, 1$, nous avons

$$\|u - \pi\,(u)\,\|_{H^k\left(\Omega_\delta^2\right)} \leq C h^{min(N+1,m)-k} N^{k-m} \|u\|_{H^m\left(\Omega_\delta^2\right)} \tag{1.58}$$

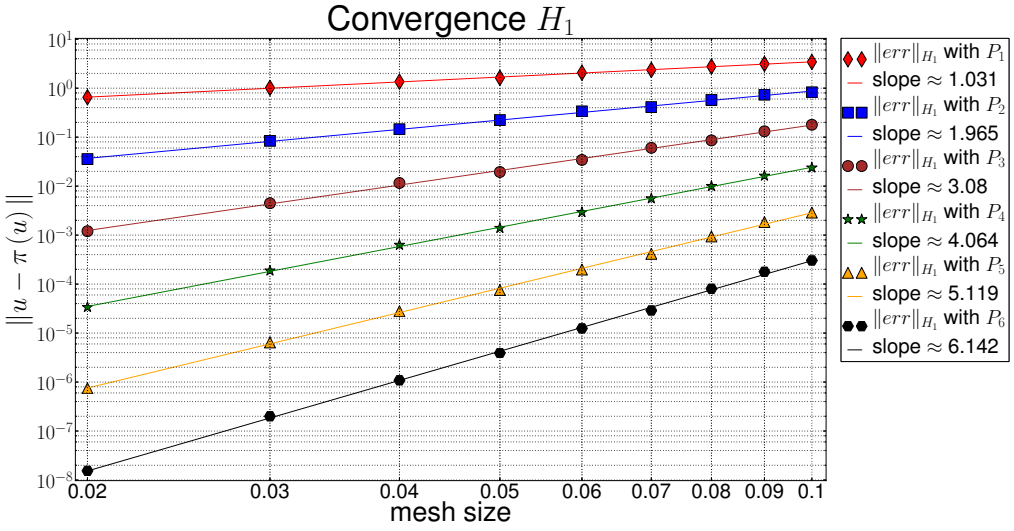
Comme la fonction u est infiniment régulière, l'indice m peut être pris relativement grand. Nous pouvons ainsi en déduire les estimations d'erreurs en norme L^2 et H^1 :

$$\|u - \pi(u_1)\|_{L^2(\Omega_\delta^2)} \leq Ch^{N+1}N^{-m}\|u\|_{H^m(\Omega_\delta^2)} \tag{1.59}$$

$$\|u - \pi(u_1)\|_{H^1(\Omega_\delta^2)} \leq Ch^{N}N^{1-m}\|u\|_{H^m(\Omega_\delta^2)} \tag{1.60}$$



(a) Convergence L^2 en h



(b) Convergence H^1 en h

FIGURE 1.15 – Tests de convergence en h pour l'opérateur d'interpolation (échelle log-log).

Enfin, nous avons tracé les courbes de convergence de l'opérateur d'interpolation avec les figures 1.15(a), 1.15(b) et 1.16. Elles représentent les erreurs d'approximation faites sur le domaine Ω_δ^2. Les deux premières figures expriment les erreurs L^2 et H^1 en fonction du pas du maillage h. Ces mesures sont effectuées pour plusieurs ordres polynomiaux. D'après les inégalités (1.59) et (1.60), on peut voir grâce aux pentes établies en échelle

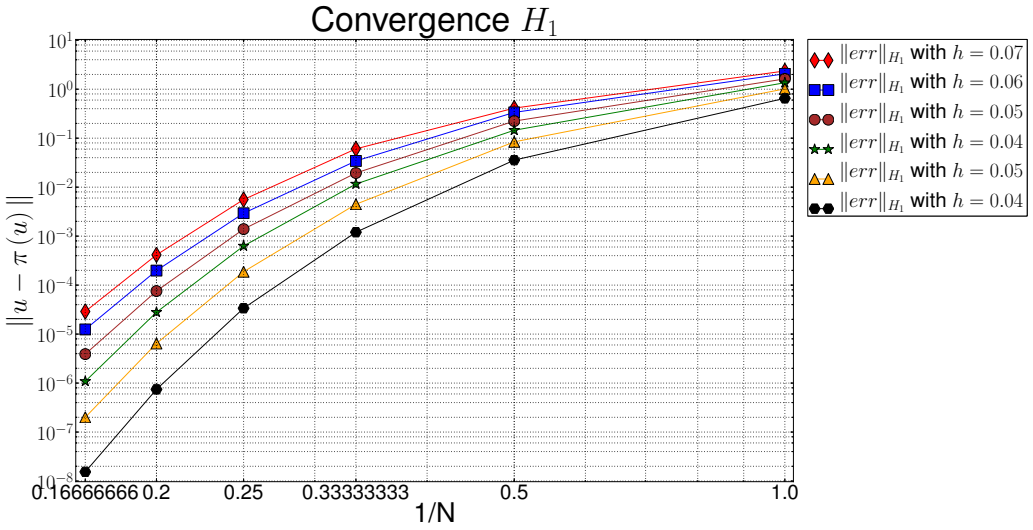


FIGURE 1.16 – Tests de convergence en N pour l'opérateur d'interpolation (échelle log-log).

log-log que la convergence est optimale. La dernière figure correspond à l'erreur H^1 selon l'ordre polynomial N qui converge de façon exponentielle. Ces dernières mesures sont comparées pour différents pas du maillage h.

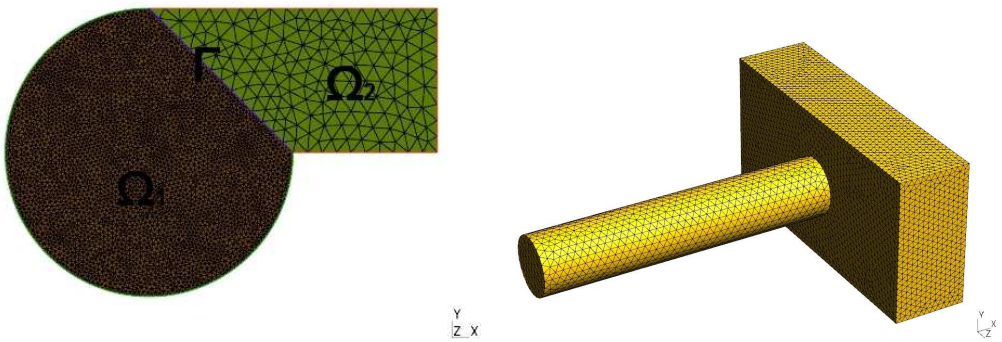
3.2 Décomposition de domaine

Nous souhaitons maintenant résoudre un problème en utilisant une méthode de décomposition de domaine sans recouvrement. Cet exemple nous permet de présenter une première application de l'opérateur d'interpolation. Nous présentons une application similaire à celle décrite dans [79]. Nous considérons les formulations fortes (à gauche) et variationnelles (à droite) écrites dans le cas mono-domaine par les équations suivantes :

$$\begin{cases} \text{Trouver } u \text{ tel que :} \\ \quad -\Delta u = f \quad \text{sur } \Omega \\ \quad\quad u = 0 \quad \text{sur } \delta\Omega \end{cases} \qquad \begin{cases} \text{Trouver } u \in H_0^1(\Omega) \text{ tel que :} \\ \displaystyle\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f\ v, \quad \forall v \in H_0^1(\Omega) \end{cases} \tag{1.61}$$

Nous découpons tout d'abord en 2 parties Ω_1 et Ω_2 le domaine d'étude Ω. On note Γ l'interface entre ces domaines ($\Gamma = \Omega_1 \cap \Omega_2$). Après discrétisation des domaines, nous notons par Ω_δ^1 et Ω_δ^2 les domaines de calculs. Comme les maillages ne sont pas forcément conformes, nous notons Γ_δ^i l'interface de Ω_δ^i. Nous posons également $\Sigma_\delta^i = \partial\Omega_\delta^i \setminus \Gamma_\delta^i$.

Nous avons représenté les domaines de calcul 2D et 3D que nous utilisons avec la figure 1.17. La décomposition du domaine 2D est directement donnée par la figure. Pour le cas 3D, le domaine Ω_1 (resp Ω_2) correspond au cylindre (resp parallélépipède). Pour les deux cas, nous utilisons une subdivision du domaine sans recouvrement. Ensuite, pour chaque sous-domaine, nous allons associer un nouveau problème. Le couplage entre les sous-domaines se fait grâce à l'ajout d'une nouvelle variable λ définie uniquement sur la frontière Γ. On utilise par la suite les espaces d'approximation $P_c^N(\Omega_\delta^1)$ et $P_c^N(\Omega_\delta^2)$ pour les solutions sur chacun des sous-domaines. Pour la fonction λ, on choisit librement l'espace



(a) Domaine 2D de calcul (b) Domaine 3D de calcul

FIGURE 1.17 – Domaines de calcul utilisés pour l'application de la méthode de décomposition de domaine.

$P_c^N(\Gamma_\delta^1)$ ou $P_c^N(\Gamma_\delta^2)$. La méthode de décomposition de domaine s'exprime sur chaque sous-domaine par le problème suivant :

Chercher la fonction u_i dans Ω_δ^i vérifiant :

$$\begin{cases} -\Delta u_i = & f & \text{sur } \Omega_\delta^i \\ u_i = & 0 & \text{sur } \Sigma_\delta^i \\ \nabla u_i \cdot \boldsymbol{n} = & -1^i \lambda & \text{sur } \Gamma_\delta^i \end{cases} \tag{1.62}$$

Ensuite, nous transformons la formulation des problèmes précédents sous leurs formes variationnelles, ce qui donne pour chaque sous-domaine :

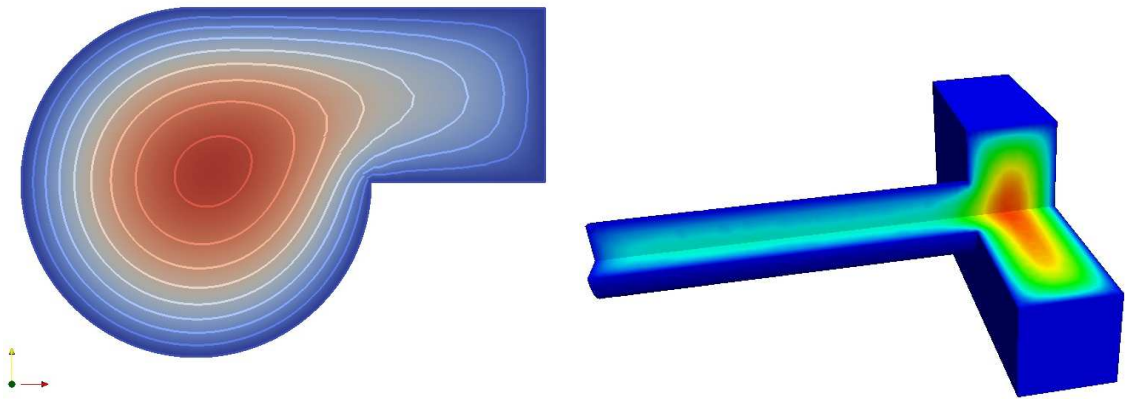
$$\begin{cases} \text{Trouver } u_i \in H^1_{0,\Sigma_\delta^i}(\Omega_\delta^1) \cap P_c^N(\Omega_\delta^1) \text{ tel que :} \\ \displaystyle\int_{\Omega_i} \nabla u_i \cdot \nabla v = \int_{\Omega_i} f\ v\ - 1^i \int_{\Gamma_i} \lambda\ v, \quad \forall v \in H^1_{0,\Sigma_\delta^i}(\Omega_\delta^i) \cap P_c^N(\Omega_\delta^i) \end{cases} \tag{1.63}$$

Dans l'écriture des équations ci-dessus, on voit apparaitre la nécessité d'interpoler λ sur chacune des interfaces Γ_δ^i. On remplace alors λ par son interpolé de Lagrange $\pi_i(\lambda)$ grâce à la définition d'un opérateur d'interpolation $\pi_i : P_c^N(\Gamma_\delta^1) \longrightarrow P_c^N(\Omega_\delta^i)$.

Enfin, nous utilisons l'algorithme itératif 1 pour résoudre le problème initial à l'aide de cette décomposition de domaine. Nous trouvons aussi dans cet algorithme l'expression de la mise à jour du champ de couplage λ à chaque itération. Les solutions obtenues après convergence sont illustrées avec les figures 1.18(a) et 1.18(b). Le critère de convergence est donné par l'évaluation de $\|u_1 - u_2\|_{L^2(\Gamma_\delta^1)} < \epsilon$ avec la tolérance $\epsilon = 1e^{-6}$.

Algorithm 1 Méthode de résolution itérative

while Non convergence **do**
  Résolution du problème sur Ω_1
  Résolution du problème sur Ω_2
  Calcul Convergence
  $\lambda = \lambda - \frac{1}{2}(u_1 - u_2)$
end while



(a) Solution 2D (b) Solution 3D

FIGURE 1.18 – Solutions numériques calculées avec la méthode de décomposition de domaine.

3.3 Condition de Dirichlet avec multiplicateurs de Lagrange

La prise en compte des conditions de Dirichlet est en général faite par élimination des degrés de liberté appartenant à cette frontière. On modifie directement la matrice, ce qui peut être vu comme une méthode algébrique. Une autre méthode, au niveau fonctionnel cette fois, consiste à considérer le problème initial sous la contrainte que la solution est imposée sur la frontière. On ajoute ainsi un multiplicateur de Lagrange comme une nouvelle inconnue dans notre système. La résolution d'un tel problème utilisera la méthode de Galerkin non standard que nous avons présentée dans ce chapitre.

Soit Ω un domaine borné de $\mathbb{R}^N$ et Γ la frontière de ce domaine. Le problème que l'on souhaite résoudre s'exprime par :

Chercher la fonction u vérfiant :

$$-\Delta u = f, \quad \text{dans } \Omega \tag{1.64}$$

$$u = g, \quad \text{sur } \Gamma \tag{1.65}$$

avec les hypothèses que $f \in L^2(\Omega)$ et $g \in H^{\frac{1}{2}}(\Gamma)$.

Nous intégrons maintenant la contrainte $u = g$ sur la frontière Γ à l'aide du multiplicateur de Lagrange λ. La formulation variationnelle de ce problème contraint est ensuite donnée par :

Trouver $(u, \lambda) \in (H^1(\Omega) \times H^{-\frac{1}{2}}(\Gamma))$ tel que $\forall (v, \mu) \in (H^1(\Omega) \times H^{-\frac{1}{2}}(\Gamma))$:

$$\int_\Omega \nabla u \cdot \nabla v + \int_\Gamma \lambda\, v = \int_\Omega f\, v \tag{1.66}$$

$$\int_\Gamma u\, \mu = \int_\Gamma g\, \mu \tag{1.67}$$

Pour réaliser un test de convergence pour cette formulation, nous utilisons la solution analytique suivante :

$$g \equiv g(x, y, z) = \cos(5\pi x)\cos(5\pi z)\cos(5\pi z) \tag{1.68}$$

Le second membre f est calculé à partir de la fonction g, c'est-à-dire $f = -\Delta g$. La géométrie utilisée pour cette application est une sphère de centre $(0, 0, 0)$ et de rayon 0.5.

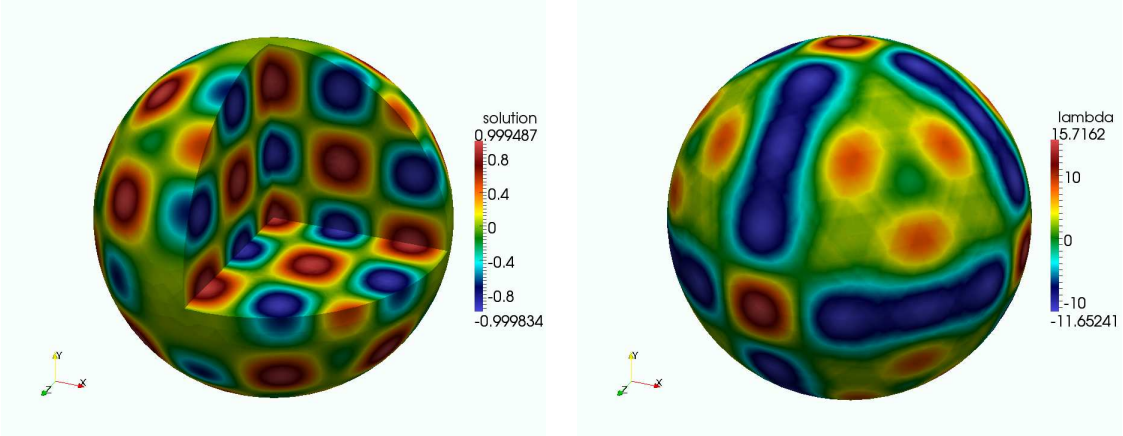
On note Ω_δ la discrétisation du domaine Ω. Pour la discrétisation géométrique de Γ, notée Γ_δ nous avons choisi de prendre exactement la trace de $\partial\Omega_\delta$, i.e. $\Gamma_\delta \equiv \partial\Omega_\delta$. L'ordre d'approximation polynomial du multiplicateur de Lagrange est pris égale à l'ordre d'approximation de la solution u. Ainsi, nous obtenons cette formulation variationnelle discrète :

Trouver $(u_\delta, \lambda_\delta) \in P_c^N(\Omega_\delta) \times P_c^N(\Gamma_\delta)$ tel que $\forall (v, \mu) \in P_c^N(\Omega_\delta) \times P_c^N(\Gamma_\delta)$:

$$\int_{\Omega_\delta} \nabla u_\delta \cdot \nabla v + \int_{\Gamma_\delta} \lambda_\delta \, v = \int_{\Omega_\delta} f \, v \tag{1.69}$$

$$\int_{\Gamma_\delta} u_\delta \, \mu = \int_{\Gamma_\delta} g \, \mu \tag{1.70}$$

Nous avons illustré une solution numérique obtenue pour ce problème avec les figures 1.19(a) et 1.19(b). Nous trouvons respectivement la solution u_δ et le multiplicateur de Lagrange λ_δ défini sur la frontière Γ_δ.



(a) Solution u calculée (b) Multiplicateur de Lagrange calculé λ

FIGURE 1.19 – Solution et multiplicateur de Lagrange calculé pour le cas 3D.

Nous présentons finalement les courbes de convergence mesurant l'erreur d'approximation en norme H^1 avec la figure 1.20. Les approximations polynomiales que nous avons testées vont de P_1 à P_6. Pour chaque cas, nous obtenons les ordres optimaux de convergence en h.

4 Conclusion

Dans ce chapitre, nous avons présenté un cadre mathématique permettant de résoudre divers types d'équations aux dérivées partielles. Pour cela, nous avons décrit les fondements de base de la méthode des éléments finis associés à une méthode d'approximation de Galerkin d'ordre arbitraire. Nous avons également développé quelques détails importants de la technique de l'élément de référence. Ce formalisme sera ainsi utilisé pour résoudre

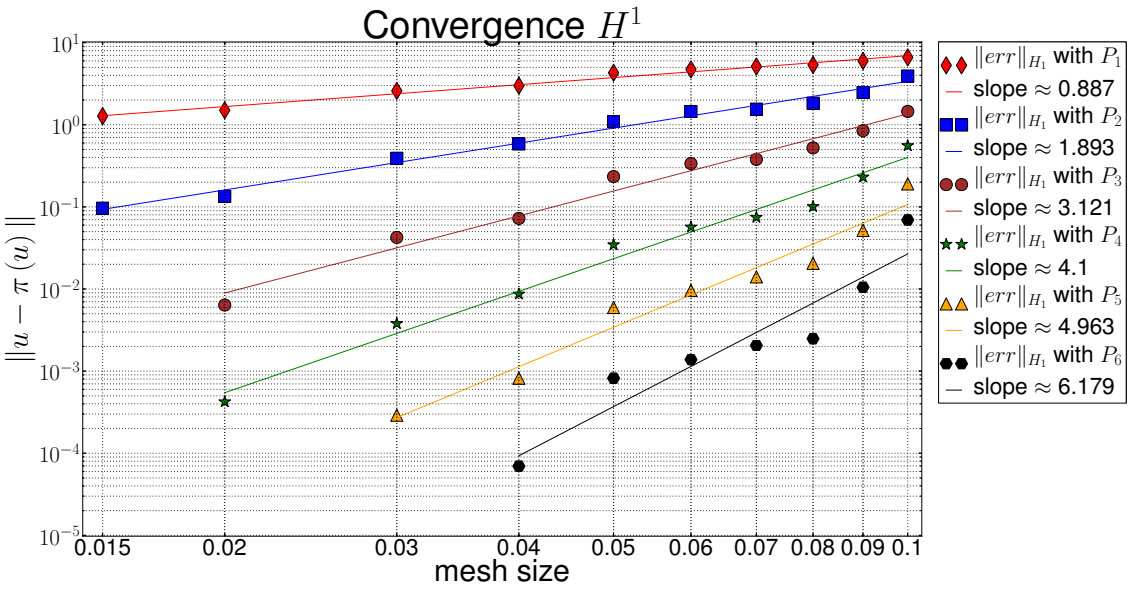


FIGURE 1.20 – Tests de convergence en h pour la norme H^1 (échelle log-log).

tous les modèles physiques que nous rencontrerons au cours de cette thèse. De plus, certains détails présents dans cette description seront d'une grande utilité pour le chapitre 7 sur le calcul haute performance. Nous parlerons entre autres de la table des degrés de liberté et de l'assemblage local.

Nous sommes ensuite passés à la notion d'interpolation entre des domaines de calcul. Cette fonctionnalité est un des points clés pour le modèle d'interaction fluide structure que nous présenterons au chapitre 6. Nous utiliserons également cet outil pour la méthode de domaines fictifs exposée dans le chapitre 3. Nous avons aussi décrit dans cette partie la construction d'un opérateur d'interpolation. Ce dernier est d'une très grande efficacité grâce à sa représentation algébrique.

Puis, nous nous sommes intéressés à la méthode de Galerkin non standard qui intervient lorsque la formulation variationnelle contient des termes couplés définis sur des domaines de calcul distincts. La méthode que nous proposons permet de traiter tous les cas de figure possibles (conformités/non-conformités, formes linéaires/bilinéaires). Les briques de base essentielles à cette construction sont l'outil de localisation et l'interpolation de fonction de base. Nous avons alors illustré l'intérêt de cette méthode avec la prise en compte de conditions aux limites de Dirichlet à l'aide d'un multiplicateur de Lagrange. Dans le contexte de cette thèse, nous l'utiliserons pour la nouvelle méthode de domaines fictifs que nous présenterons dans les chapitres 3 et 6.

Tout au long de ce chapitre, nous avons mis en avant un certain nombre de codes FEEL++. L'intérêt de ces illustrations est de montrer les contributions informatiques qui ont été réalisées dans cette thèse. Elles montrent également l'effort de développement qui a été fait pour enrichir le langage proposé par cette librairie. Principalement, nous avons montré la transparence de l'outil d'interpolation, la ressemblance entre la formulation variationnelle mathématique et informatique pour la méthode de Galerkin non standard et la manipulation de structures algébriques par blocs.

Chapitre 2

Carte ALE

Dans la modélisation des écoulements sanguins, un phénomène important est l'interaction fluide structure qui se produit entre le plasma et la paroi artérielle. Les battements du coeur permettent au sang de circuler dans le réseau artériel, veineux et capillaire en transportant également des entités comme les globules rouges. Cette impulsion entraine la propagation d'une onde de pression qui déforme les voies sanguines lors de son passage. Les globules rouges sont aussi des corps élastiques qui interagissent fortement avec le plasma. Dans ce contexte de rhéologie sanguine, il est nécessaire d'avoir des outils capables de simuler un écoulement sanguin soumis à diverses déformations au cours du temps.

Généralement, les équations de la mécanique des solides sont décrites dans le repère dit lagrangien, c'est-à-dire qu'elles suivent le déplacement des particules au cours du temps. Au contraire de cette idée, la mécanique des fluides est plus facilement décrite dans le repère dit eulérien, où l'observateur est à une position fixe et regarde passer les particules. Nous avons choisi d'utiliser ces descriptions pour modéliser le sang et le déplacement de la paroi artérielle. Dans ce cas, pour prendre en compte le déplacement du domaine fluide, c'est-à-dire le sang, nous allons utiliser une technique classique qui se nomme la méthode ALE (arbitrairement lagrangien-eulérien). Cette méthode permet à l'écoulement fluide de suivre le mouvement à l'interface fluide-structure (lagrangien près de la structure) et d'avoir à l'intérieur une vitesse de déformation qui ne suit pas nécessairement la vitesse du fluide. Une autre méthode utilisée pour aborder un problème d'interaction fluide-structure, est une approche totalement eulérienne [137, 48, 138, 41], les équations du fluide et de la structure sont écrites dans le même repère eulérien. Il existe également la méthode FM-ALE (fixed-mesh ALE) [40] qui utilise le principe de la méthode ALE, mais les équations du fluide sont traitées uniquement dans le domaine de référence. D'autres alternatives sont encore possibles comme les méthodes de type domaine fictif ou éventuellement combiné avec de l'adaptation de maillage [159]. Nous préférons choisir l'approche ALE car c'est une méthode robuste et précise, mais qui a la faiblesse d'être limitée dans l'importance des déformations imposées.

La méthode ALE a été mise en oeuvre dans de nombreux contextes et notamment pour la simulation des écoulements sanguins lorsqu'il s'agit de modéliser l'interaction plasma et paroi artérielle. Le mouvement du domaine de calcul est engendré par un déplacement imposé sur l'interface fluide-structure. Mais dans la plupart des publications existantes, les méthodes de déformations des maillages étaient d'ordre 1, c'est-à-dire que l'on déplaçait uniquement les sommets des éléments constituant le maillage. Nous voulons ici présenter une méthode plus précise, dans le sens où nous serons capables de déplacer les éléments

du maillage d'ordre élevé (>1). Ainsi nous voulons également faire évoluer les noeuds géométriques se trouvant sur les arêtes (en 2D) ou faces (3D). Les maillages d'ordre élevé seront générés grâce à la librairie GMSH [67], jusqu'à l'ordre 5 en 2D et jusqu'à l'ordre 4 en 3D. Au niveau des applications réalistes d'écoulement sanguin, les maillages issus de l'imagerie médicale ne seront pas généralement d'ordre élevé, dû à la difficulté d'obtenir de bons maillages à l'ordre 1. Cependant, à partir d'une telle discrétisation, nous pouvons générer un autre maillage, identique au sens où il représentera le même domaine, mais d'ordre élevé dans le sens où des points géométriques supplémentaires seront ajoutés. Ainsi dans un contexte d'interaction fluide-structure, les déformations engendrées pourront être modélisées de manière plus précise grâce à ces nouveaux degrés de liberté.

Nous évoquons maintenant les travaux [125] où a été proposé un cadre pour l'interaction fluide-structure en 2D pour l'ordre élevé en s'appuyant sur une construction efficace de la carte ALE d'ordre élevé décrite dans [134]. Un ingrédient fondamental dans ce cadre ALE est la transformation ALE discrète qui relie le domaine fluide de référence Ω_f^* au domaine fluide de calcul Ω_f^t à chaque pas de temps. La construction de la carte ALE discrète proposée dans [134] est basée sur l'opérateur d'extension harmonique et les transformations de Gordon-Hall [73, 74]. Cependant, l'utilisation de l'extension harmonique peut générer des maillages invalides, par exemple figure 2.5. De plus, les transformations de Gordon-Hall sont complexes à mettre en oeuvre en 2D et leurs extensions en 3D sont une tâche très difficile. C'est pourquoi nous voulons ici proposer une nouvelle méthode se basant toujours soit sur l'extension harmonique, soit sur les équations de Winslow(uniquement en 2D pour le moment) qui sont plus robustes [164]. Nous remplacerons également les transformations de Gordon-Hall par la résolution de petits problèmes locaux dans chaque élément du maillage en contact avec les frontières courbes. Notre construction de la carte ALE d'ordre élevé sera valable en 2D et 3D.

Dans ce chapitre, nous commencerons par donner quelques définitions et notations nécessaires pour présenter la construction de notre carte ALE d'ordre élevé. Ensuite, nous passerons à cette construction qui s'effectuera en deux étapes, tout d'abord une étape classique avec l'extension harmonique ou les équations de Winslow, puis nous expliquerons la correction à appliquer pour obtenir l'ordre élevé. Enfin, nous présenterons des propriétés théoriques et des résultats numériques de notre méthode.

1 Notations, hypothèses et modèles d'application FSI

Soit $\Omega^{t_0} \subset \mathbb{R}^d$, $d \geq 1$, un domaine borné représentant le domaine de calcul au temps initial $t = t_0$. Nous voulons résoudre des équations aux dérivées partielles dans ce domaine en mouvement au cours du temps. Le domaine déformé à l'instant t est noté Ω^t avec $t \in [t_0, t_f]$. Pour prendre en compte les déformations lagrangiennes du bord du domaine tout en gardant des caractéristiques eulériennes à l'intérieur, nous avons choisi d'utiliser la méthode ALE présentée en introduction dont l'ingrédient indispensable est la carte ALE. Cette carte, notée $\mathcal{A}^t$, permet d'avoir la position d'un point $\mathbf{x}$ du repère déformé à un instant t donné à partir de la position d'un point $\mathbf{x}^*$ dans une configuration de référence, notée Ω^*.

Comme le montre également la figure 2.1, nous notons par $\mathbf{x}^*$ les coordonnées d'un point dans le repère de référence Ω^* et par $\mathbf{x}$ la position de ce point dans le repère déformé. La carte ALE $\mathcal{A}^t$ est un homéomorphisme dans Ω^*, c'est-à-dire que $\mathcal{A}^t$ est une application

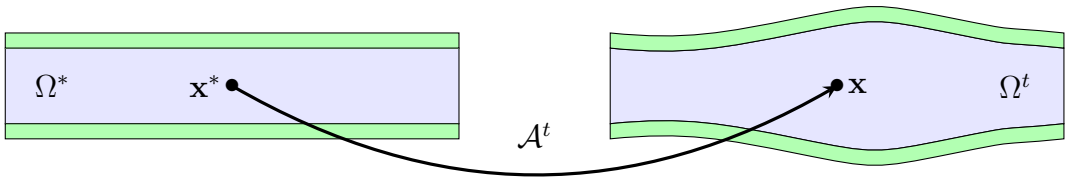


FIGURE 2.1 – Représentation de la carte ALE $\mathcal{A}^t$ avec le domaine de référence Ω^* et le domaine déformé Ω^t.

continue et bijective. Nous définissons alors $\forall t \in [t_0, t_f]$:

$$\begin{aligned} \mathcal{A}^t : \Omega^* &\longrightarrow \Omega^t \\ \mathbf{x}^* &\longmapsto \mathbf{x}\left(\mathbf{x}^*, t\right) = \mathcal{A}^t\left(\mathbf{x}^*\right) \end{aligned}$$

Nous supposons aussi que $\forall \mathbf{x}^* \in \Omega^*$, l'application :

$$\begin{aligned} \mathbf{x} : [t_0, t_f] &\longrightarrow \Omega^t \\ t &\longmapsto \mathbf{x}\left(\mathbf{x}^*, t\right) \end{aligned}$$

est différentiable presque partout.

De plus, nous n'avons pas vraiment de restriction dans le choix du domaine de référence Ω^*. C'est pourquoi nous le choisissons généralement comme étant le domaine de calcul au temps initial, c'est à dire $\Omega^* = \Omega^{t_0}$. Cependant, il est parfois nécessaire de remailler le domaine de calcul lorsque la déformation est trop importante par exemple. Dans ce cas particulier, nous devons choisir un nouveau domaine de référence qui correspond alors au dernier domaine déformé qui a été calculé et remaillé.

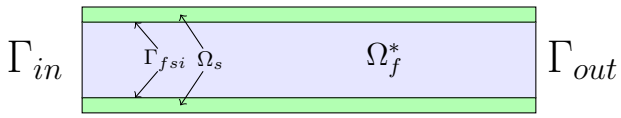


FIGURE 2.2 – Exemple de géométrie pour un problème FSI en 2D.

Contexte d'application FSI : Pour fixer les idées et bien comprendre comment la carte ALE va intervenir dans nos modèles d'écoulements, nous présentons un problème d'interaction fluide structure valable en 2D et 3D. Ce modèle sera décrit en détail au cours des chapitres 4, 5 et 6. Tout d'abord, nous commençons par un exemple de géométries 2D avec la figure 2.2, où Ω_f^* et Ω_s sont respectivement le domaine ALE de référence pour le fluide et le domaine occupé par la structure. Au niveau des frontières, nous avons Γ_{fsi}^t l'interface entre le fluide et la structure mais aussi des frontières supplémentaires Γ_{in} et Γ_{out} pouvant représenter par exemple l'entrée et la sortie d'un écoulement. Nous pouvons alors considérer le problème d'interaction fluide structure dans le cadre ALE suivant :

Trouver $(\mathcal{A}^t, \boldsymbol{u}_f, p_f, \boldsymbol{\eta}_s)$ telles que :

$$\begin{aligned}
\rho_f \left.\frac{\partial \mathbf{u}_f}{\partial t}\right|_{\mathbf{x}^*} + \rho_f((\mathbf{u}_f - \mathbf{w}_f)\cdot\nabla_{\mathbf{x}})\mathbf{u}_f - \nabla_{\mathbf{x}}\cdot\boldsymbol{\sigma}_f &= 0 \quad \text{dans } \Omega_f^t \times [t_0, t_f] \\
\nabla_{\mathbf{x}}\cdot\mathbf{u}_f &= 0 \quad \text{dans } \Omega_f^t \times [t_0, t_f] \\
\rho_s \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \nabla\cdot(\boldsymbol{F}_s\boldsymbol{\Sigma}_s) &= 0 \quad \text{dans } \Omega_s \times [t_0, t_f] \\
\mathbf{u}_f - \frac{\partial \boldsymbol{\eta_s}}{\partial t}\circ\left(\mathcal{A}^t\right)^{-1} &= 0 \quad \text{sur } \Gamma_{fsi}^t \times [t_0, t_f] \\
\boldsymbol{F}_s\boldsymbol{\Sigma}_s\hat{\boldsymbol{n}}_s + J_{\mathcal{A}^t}\boldsymbol{F}_{\mathcal{A}^t}^{-T}\hat{\boldsymbol{\sigma}}_f\hat{\boldsymbol{n}}_f &= 0 \quad \text{sur } \Gamma_{fsi}^0 \times [t_0, t_f]
\end{aligned}$$

où $\Omega_f^t = \mathcal{A}^t(\Omega_f^*)$ est le domaine occupé par le fluide en mouvement, $\mathbf{u}_f$ et p_f représente la vitesse et pression du fluide, $\left.\frac{\partial \mathbf{u}_f}{\partial t}\right|_{\mathbf{x}^*}$ est la dérivée temporelle dans le repère ALE, $\boldsymbol{\sigma}_f$ est le tenseur des contraintes du fluide. $\boldsymbol{\eta}_s$ représente le déplacement de la structure, $\boldsymbol{F}_s = \mathbf{I} + \nabla\boldsymbol{\eta}_s$ est le tenseur des déformations de la structure, $\boldsymbol{\Sigma}_s = \lambda_s(\operatorname{tr}\boldsymbol{E}_s)\boldsymbol{I} + 2\mu_s\boldsymbol{E}_s$ correspond au second tenseur de Piola-Kirchoff. Ces notations permettent de définir le tenseur des contraintes $\hat{\boldsymbol{\sigma}}_f = \boldsymbol{\sigma}_f \circ \mathcal{A}^t$ et le vecteur normal $\hat{\boldsymbol{n}}_f = \boldsymbol{n}_f \circ \mathcal{A}^t$ dans le domaine de référence. Nous définissons également $\boldsymbol{F}_{\mathcal{A}^t} = \nabla\mathcal{A}^t$ et $J_{\mathcal{A}^t} = \det\left(\boldsymbol{F}_{\mathcal{A}^t}\right)$. Le problème doit aussi être complété avec des conditions aux limites sur Γ_{in} et Γ_{out} pour avoir un problème correctement posé. Le type de condition, Neumann ou Dirichlet par exemple, n'a pas d'importance pour la description de la carte ALE, nous considérons seulement le fait que ces bords ne se déplacent pas.

Simplification des notations en temps : Dans la suite de ce chapitre, nous nous intéresserons uniquement à la construction de la carte ALE entre un domaine de référence Ω^* et un domaine déformé Ω. Nous enlevons donc les notations liées au temps t car il n'y a pas de réelle dépendance entre les cartes $\mathcal{A}^t$ et $\mathcal{A}^{t'}$, $\forall t_0 < t < t'$.

Discrétisation géométrique : Soit δ un paramètre de discrétisation. Nous introduisons alors Ω_δ une discrétisation de Ω et en général, on a $\Omega_\delta \neq \Omega$. A noter que si Ω est un domaine polygonal alors $\Omega_\delta = \Omega$. Nous définissons $\hat{K} \subset \mathbb{R}^d$ $d = 1, 2, 3$ l'élément de référence, e.g. un simplexe ou un hypercube. Nous notons par $\mathcal{T}_\delta$ un ensemble fini non vide et disjoint de simplexes ou hypercubes. Nous supposons que $\mathcal{T}_\delta \equiv \mathcal{T}_{(h,k)} = \{K = \boldsymbol{\varphi}_{K,k}^{\text{geo}}(\hat{K})\}$ forme une partition de Ω_δ telle que $h = \max_{K\in\mathcal{T}_\delta} h_K$, avec h_K le diamètre de l'élément $K \in \mathcal{T}_\delta$. $\boldsymbol{\varphi}_{K,k}^{\text{geo}}$ est le polynôme de degré k qui relie $\hat{K}$ à K et qui aussi s'appelle la transformation géométrique. Nous disons que l'hyperplan fermé F de $\overline{\Omega_\delta}$ est une face du maillage s'il a une mesure $(d-1)$-dimensionnelle positive et si de plus il existe K_1, $K_2 \in \mathcal{T}_\delta$ tel que $F = \partial K_1 \cap \partial K_2$ (et F est appelée une *face interne*) ou il existe $K \in \mathcal{T}_\delta$ tel que $F = \partial K \cap \partial\Omega_\delta$ (et F est appelée une *face frontière*). Les faces internes sont rassemblées dans l'ensemble $\mathcal{F}_\delta^i$, les faces frontières dans $\mathcal{F}_\delta^b$ et nous définissons $\mathcal{F}_\delta := \mathcal{F}_\delta^i \cup \mathcal{F}_\delta^b$. Pour tout $F \in \mathcal{F}_\delta$, nous definissons $\mathcal{T}_F := \{K \in \mathcal{T}_\delta \mid F \subset \partial K\}$. Pour chaque interface $F \in \mathcal{F}_\delta^i$ nous introduisons deux normales associées aux éléments de $\mathcal{T}_F$ et nous avons $\boldsymbol{n}_{K_1,F} = -\boldsymbol{n}_{K_2,F}$, où $\boldsymbol{n}_{K_i,F}$, $i \in \{1, 2\}$, dénote la normale unitaire de F pointant vers l'exterieur de $K_i \in \mathcal{T}_F$. Pour une face frontière $F \in \mathcal{F}_\delta^b$, $\boldsymbol{n}_F = \boldsymbol{n}_{K,F}$ dénote la normale unitaire pointant vers l'extérieur de Ω_δ.

Discrétisation spatiale : Nous supposons à partir de maintenant que nous travaillerons uniquement avec des éléments symplectiques pour simplifier un peu les notations, mais tout ce cadre sera aussi valable pour les hypercubes. Nous appellerons Ω_δ le domaine de *calcul* et nous introduisons Ω_δ^* qui sera représenté comme le domaine de *réference*. Nous supposons que le domaine de référence a un maillage composé de faces droites i.e. $\mathcal{T}_\delta^* \equiv \mathcal{T}_{(h,1)}^*$. De plus, nous admettons que le maillage $\mathcal{T}_\delta^*$ recouvre exactement le domaine Ω_δ^*, *i.e.*, $\Omega_\delta^* = \bigcup_{K^* \in \mathcal{T}_\delta^*} \overline{K^*}$. Nous notons alors $\mathcal{A}_\delta \equiv \mathcal{A}_{(h,k)}$ la transformation qui relie Ω_δ^* à Ω_δ. Dans la suite, on appellera cette transformation $\mathcal{A}_\delta$ la carte ALE discrète. Nous allons maintenant introduire les espaces discrets d'approximation nécessaire à la construction de $\mathcal{A}_\delta$. Nous définissons $P_c^N(\Omega_\delta^* \equiv \Omega_{(h,1)}^*)$ et $[P_c^N(\Omega_\delta^* \equiv \Omega_{(h,1)}^*)]^d$ les espaces de fonctions continues scalaires et vectoriels sur Ω_δ^* avec :

$$P_c^N(\Omega_\delta^*) = \{v \in C^0(\Omega_\delta^*) \mid v \circ \boldsymbol{\varphi}_{K,1}^{\text{geo}} \in \mathbb{P}^N(\hat{K}) \; \forall K \in \mathcal{T}_\delta^*\}, \quad [P_c^N(\Omega_\delta^*)]^d = \prod_1^d P_c^N(\Omega_\delta^*). \tag{2.1}$$

Ces espaces se caractérisent par l'utilisation de la transformation géométrique d'ordre 1 $\boldsymbol{\varphi}_{K,1}^{\text{geo}}$. Nous notons également par $\boldsymbol{\zeta}_{(h,k)} \equiv \boldsymbol{\zeta}_\delta$ la fonction déplacement permettant d'obtenir Ω_δ^* à partir de Ω_δ.

Conditions aux limites : Nous décomposons la frontière $\partial\Omega_\delta^*$ en 3 sous-ensembles : *(i)* Γ_M^*, la portion de la frontière qui se déplace selon le déplacement donné,*(ii)* Γ_F^*, la portion de la frontière qui reste fixe (ie, $\mathcal{A}_\delta(\mathbf{x}^*) = \mathbf{x}^*, \forall \mathbf{x}^* \in \Gamma_F^*$) et *(iii)* Γ_N^*, la partie sur laquelle il n'y a aucun déplacement prescrit. L'image de chaque sous-ensemble, Γ_M^*, Γ_F^* et Γ_N^* par $\mathcal{A}_\delta$ est notée respectivement par Γ_M, Γ_F et Γ_N. Ces 3 sous-ensembles ne se recouvrent pas et ils vérifient $\partial\Omega_\delta^* = \overline{\Gamma_M^*} \cup \overline{\Gamma_F^*} \cup \overline{\Gamma_N^*}$. Nous introduisons aussi $\mathcal{T}_\delta^{*,b} = \{K^* \in \mathcal{T}_\delta^* : \partial K^* \cap \Gamma_M^* \neq \emptyset\}$ l'ensemble d'éléments K^* partageant une face avec la frontière Γ_M^*.

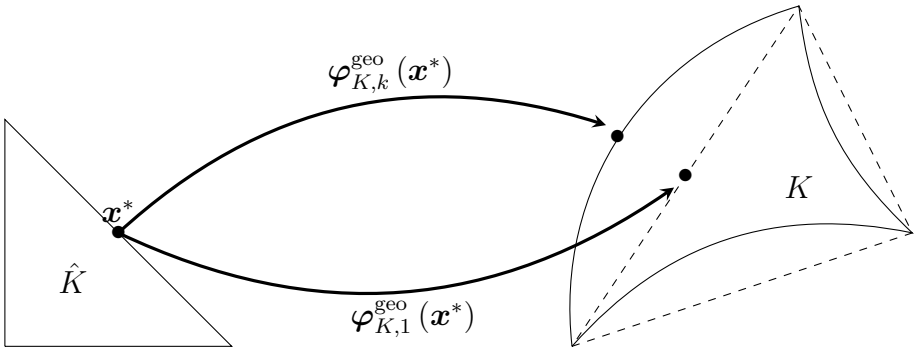


FIGURE 2.3 – Technique de raidification des éléments d'ordre élevé.

Hypothèse sur $\mathcal{T}_\delta$: Pour pouvoir appliquer la construction que nous allons présenter dans les sections suivantes, une hypothèse importante sur $\mathcal{T}_\delta$ doit avant tout être énoncée. Chacune des faces internes appartenant à l'ensemble $\mathcal{F}_\delta^i$ doit être droite ou autrement dit d'ordre 1. Par exemple dans le cas 2D, les points géométriques d'une arrête se trouvent tous exactement sur un segment droit. Malheureusement, les maillages d'ordre élevé que nous générons avec GMSH [67] ne vérifient pas cette hypothèse. Tous les éléments construits sont courbes et donc a fortiori toutes les faces. Pour obtenir les maillages d'ordre élevé

dont les faces courbes se trouvent uniquement dans l'ensemble $\mathcal{F}_\delta^b$, nous allons appliquer à ce maillage courbe un déplacement des noeuds géométriques qui va raidifier toutes les faces internes. La figure 2.3 présente la technique que nous utilisons pour calculer le déplacement à appliquer pour un élément $K \in \mathcal{T}_\delta$. À cet élément K est associé une transformation géométrique $\boldsymbol{\varphi}_{K,k}^{\text{geo}}$, d'ordre k, telle que $K = \boldsymbol{\varphi}_{K,k}^{\text{geo}}(\hat{K})$. La clé pour établir la correction est l'utilisation de la transformation géométrique d'ordre 1, notée $\boldsymbol{\varphi}_{K,1}^{\text{geo}}$. Sur la figure 2.3, le triangle en pointillé représente le triangle K après raidification de toutes ces faces, notons le K^{raid}. En fait, ce triangle peut être obtenu grâce à l'application de $\boldsymbol{\varphi}_{K,1}^{\text{geo}}$ sur $\hat{K}$, c'est-à-dire que nous avons la relation $K^{raid} = \boldsymbol{\varphi}_{K,1}^{\text{geo}}(K)$. Ensuite pour obtenir les points géométriques sur K, nous devons appliquer $\boldsymbol{\varphi}_{K,k}^{\text{geo}}$ sur les points géométriques de $\hat{K}$ qui correspondent aux points équidistribués. Ainsi, pour chaque élément $K \in \mathcal{T}_\delta$, le déplacement $\boldsymbol{\eta}_K$ à effectuer sur l'ensemble des noeuds géométriques de K est donné par la formule suivante :

$$\boldsymbol{\eta}_K(\boldsymbol{\varphi}_{K,k}^{\text{geo}}(\boldsymbol{x}^*)) = \underbrace{\left(\boldsymbol{\varphi}_{K,1}^{\text{geo}}(\boldsymbol{x}^*) - \boldsymbol{\varphi}_{K,k}^{\text{geo}}(\boldsymbol{x}^*)\right)}_{\text{raidification de toutes les faces}} - \underbrace{\left(\boldsymbol{\varphi}_{K\cap\partial\Omega_\delta,1}^{\text{geo}}(\boldsymbol{x}^*) - \boldsymbol{\varphi}_{K\cap\partial\Omega_\delta,k}^{\text{geo}}(\boldsymbol{x}^*)\right)}_{\text{correction pour conserver les bords courbes}}$$

Cette expression est composée de deux parties, la première raidifiant l'ensemble des faces du maillage et la seconde qui corrige le déplacement des faces appartenant à $\mathcal{F}_\delta^b$. Dans cette dernière partie, nous faisons apparaitre les applications $\boldsymbol{\varphi}_{K\cap\partial\Omega_\delta,1}^{\text{geo}}$ et $\boldsymbol{\varphi}_{K\cap\partial\Omega_\delta,k}^{\text{geo}}$ définies par :

$$\boldsymbol{\varphi}_{K\cap\partial\Omega_\delta,k}^{\text{geo}} \equiv \begin{cases} \boldsymbol{\varphi}_{K,k}^{\text{geo}} & \text{, si } K \cap \partial\Omega_\delta \neq \emptyset \\ 0 & \text{, sinon} \end{cases}$$

Cependant cette technique de raidification peut poser des problèmes dans certains cas très particuliers. En effet, si des éléments du bord du maillage d'ordre élevé ont une très forte courbure alors les éléments raidifiés peuvent être invalides. Cela pourrait être le cas si l'on souhaiter par exemple décrire une perforation circulaire avec très peu d'éléments. La figure 2.4 présente un tel cas où l'élément modifié K^{raid} n'est pas valide. Pour éviter ce genre de problème, les frontières à forte courbure doivent être composées d'un minimum d'élément.

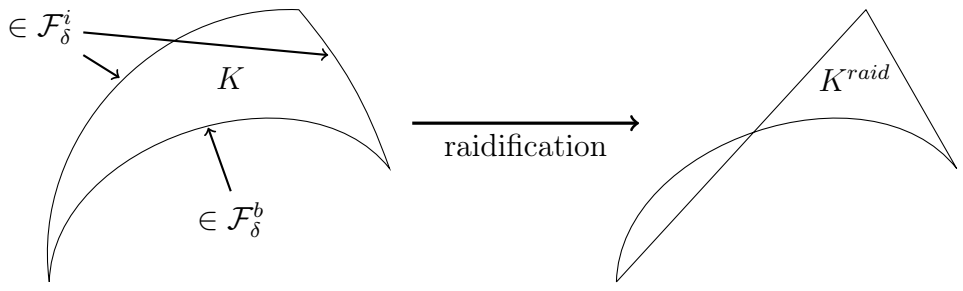


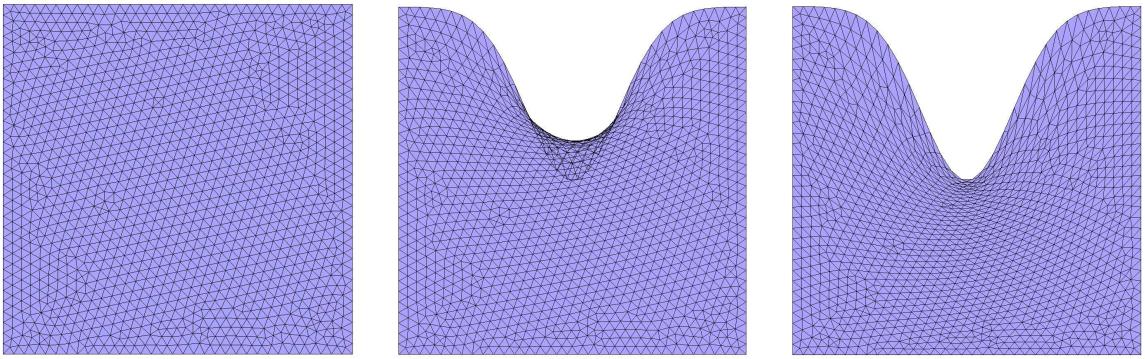
FIGURE 2.4 – Problème de raidification pouvant apparaitre pour les éléments ayant une face touchant le bord du domaine.

2 Transformation ALE d'ordre élevé

Dans cette section, nous allons présenter la construction de notre carte ALE discrète d'ordre élevé qui est une pièce fondamentale pour les simulations dans le cadre ALE. La construction de cette carte est faite en deux étapes :

- Construction de la carte ALE d'ordre 1 : $\mathcal{A}_{(h,1)}$
- Correction pour obtenir la carte ALE d'ordre élevé k : $\mathcal{A}_{(h,k)}$

où k représente également l'ordre géométrique de Ω_δ. L'un des intérêts de cette méthode est qu'elle se base sur la construction $\mathcal{A}_{(h,1)}$ où une large gamme de méthode existe. Nous pouvons citer par exemple l'extension harmonique [125, 138, 57, 83], le bi-laplacien [163], les modèles d'élasticité compressibles ou incompressibles [98, 7], les équations de Winslow [164, 108, 90] ou encore la méthode *ball-vertex* [29, 4]. Ensuite nous appliquons une correction pour obtenir la courbure d'ordre élevé. Celle-ci s'applique uniquement sur les éléments ayant une face sur le bord du domaine, ce qui permet d'avoir un coût relativement faible pour cette opération. Cette correction locale est uniquement possible grâce à l'hypothèse faite sur les faces internes du maillage $\mathcal{T}_\delta$.



(a) Maillage de référence (b) Harmonique (c) Winslow

FIGURE 2.5 – Comparaison des maillages générés par l'extension harmonique (invalide) et les équations de Winslow (valide).

2.1 Construction de la carte ALE d'ordre 1

Dans les deux papiers [134, 125], la carte ALE discrète créée est calculée en construisant une extension harmonique des déplacements imposés sur le bord du domaine. Cependant, si le déplacement est trop important, cet opérateur peut générer des maillages invalides, par exemple des *retournements de mailles* ou un *maillage plié*, voir figure 2.5(b). Une façon de contourner ce problème est de remplacer l'extension harmonique par un opérateur plus approprié et flexible qui permettrait d'obtenir un maillage valide et de meilleure qualité. Dans ce but, nous avons choisi d'utiliser les équations de Winslow qui d'un point de vue continu correspondent à la construction de l'extension harmonique pour l'inverse de la carte ALE. La figure 2.5(c) illustre la qualité de cet opérateur.

Extension harmonique standard

L'approche la plus simple et la plus couramment utilisée pour construire la carte $\mathcal{A}_{(h,1)}$ est de calculer l'extension harmonique du déplacement $\boldsymbol{\zeta}_{(h,1)}$ à l'aide du déplacement imposé sur le bord. On cherche ainsi à trouver $\boldsymbol{\zeta}_{(h,1)}$ qui vérifie le problème elliptique suivant :

$$\begin{cases} -\Delta\boldsymbol{\zeta}_{(h,1)} = & 0 & , & \text{dans } \Omega^*_\delta \\ \boldsymbol{\zeta}_{(h,1)} = & \boldsymbol{g}_{\boldsymbol{\zeta}} & , & \text{sur } \Gamma^*_M \\ \boldsymbol{\zeta}_{(h,1)} = & \mathbf{0} & , & \text{sur } \Gamma^*_F \\ \nabla\boldsymbol{\zeta}_{(h,1)}\ \boldsymbol{n}^* = & \mathbf{0} & , & \text{sur } \Gamma^*_N \end{cases} \tag{2.2}$$

Nous allons utiliser une méthode éléments finis pour résoudre ce système. Nous avons alors besoin de la formulation variationnelle associée à notre opérateur d'extension harmonique :

Trouver $\boldsymbol{\zeta}_{(h,1)} \in [H^1_{\boldsymbol{g}_{\boldsymbol{\zeta}},\Gamma_M}(\Omega^*_\delta)]^d \cap [H^1_{0,\Gamma_F}(\Omega^*_\delta)]^d \cap [P^1_c(\Omega^*_\delta)]^d$ telle que :

$$\int_{\Omega^*} \nabla\boldsymbol{\zeta}_{(h,1)} : \nabla\mathbf{z}\ dx = 0 \quad , \forall \mathbf{z} \in [H^1_{0,\Gamma_M\cup\Gamma_F}(\Omega^*_\delta)]^d \cap [P^1_c(\Omega^*_\delta)]^d$$

Après avoir calculé le déplacement $\boldsymbol{\zeta}_{(h,1)}$ en résolvant le problème précédent, la carte ALE discrète est déduite avec la relation suivante :

$$\mathcal{A}_{(h,1)}(\boldsymbol{x}^*) = \boldsymbol{x}^* + \boldsymbol{\zeta}_{(h,1)}(\boldsymbol{x}^*) \tag{2.3}$$

Nous illustrons maintenant des applications possibles avec cet opérateur harmonique. Nous considérons deux exemples, 2D et 3D, dont leur domaine de référence est défini par :

$$\Omega^{*,re} = \left\{(x^*, y^*) \in \mathbb{R}^2 :\ x^* \in [0,5]\,, y^* \in [-1,1]\right\} \tag{2.4}$$

$$\Omega^{*,cy} = \left\{(x^*, y^*, z^*) \in \mathbb{R}^3 :\ x^* \in [0,5]\,, y^{*2} + z^{*2} \leq 0.5^2\right\} \tag{2.5}$$

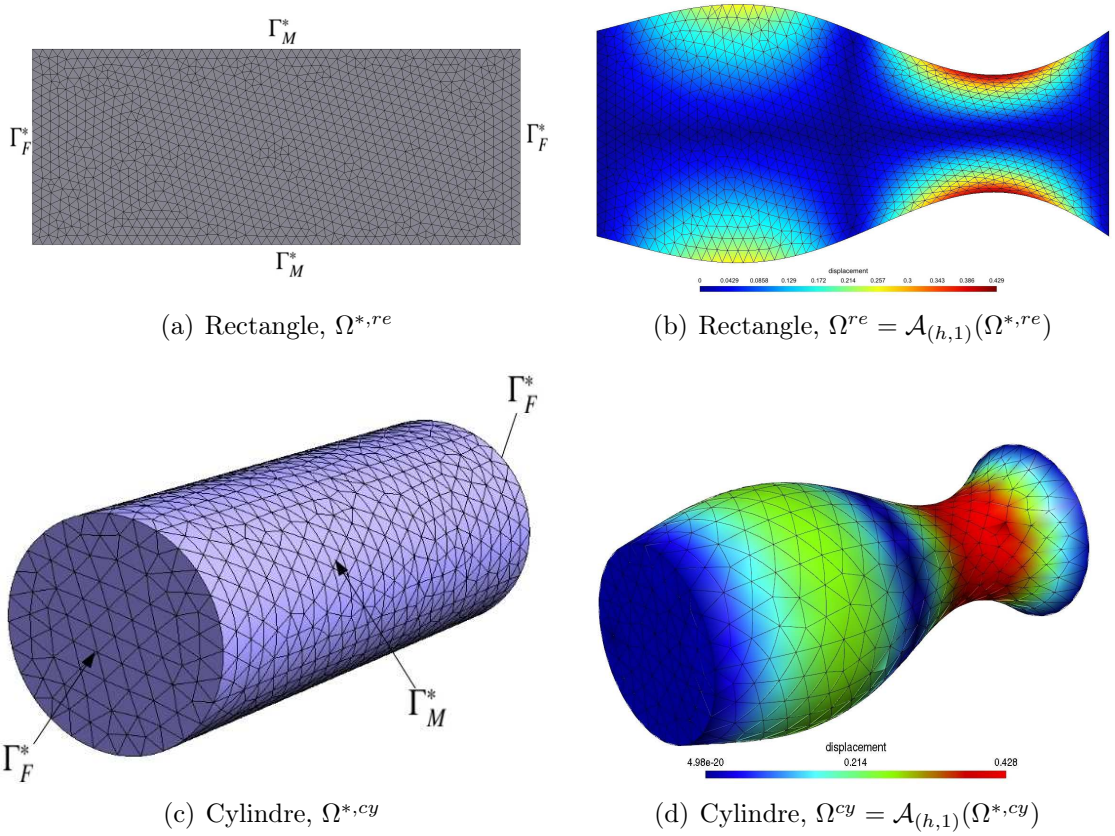
Ces domaines représentés respectivement par la figure 2.6(a) et 2.6(c). Nous appliquons ensuite sur les frontières Γ^*_M le déplacement suivant valable pour les deux exemples :

$$\boldsymbol{g}_{\boldsymbol{\zeta}}(\boldsymbol{x}^*) = 0.2\ \exp(\frac{x^*}{5}) \sin\left(\frac{\pi x^*}{2.5}\right) \boldsymbol{n}^*$$

Avec les figures 2.6(b) et 2.6(d), nous avons représenté les maillages déformés à l'aide du déplacement calculé par cette méthode. Toutefois, le déplacement qui a été imposé reste relativement faible et régulier.

Extension harmonique modifiée

L'extension harmonique présentée précédemment reste très limitée dans son champ d'application. Nous allons maintenant présenter un test qui montre ses faiblesses pour des déformations qui sont encore relativement petites. Nous prenons un rectangle perforé par une barre comme domaine de référence Ω^*, figure 2.7(a). Nous avons également affiché sur cette figure les bords en mouvement et le bord fixe autour de la barre. De plus la frontière extérieure au *grand* rectangle est fixe, elle fait donc partie de Γ^*_F. Pour ensuite nous donner un champ de déplacement sur Γ^*_M, nous allons appliquer un déplacement à la barre verte à l'aide d'une modélisation en mécanique des solides. Cette barre est fixée sur Γ^*_F et elle



(a) Rectangle, $\Omega^{*,re}$

(b) Rectangle, $\Omega^{re} = \mathcal{A}_{(h,1)}(\Omega^{*,re})$

(c) Cylindre, $\Omega^{*,cy}$

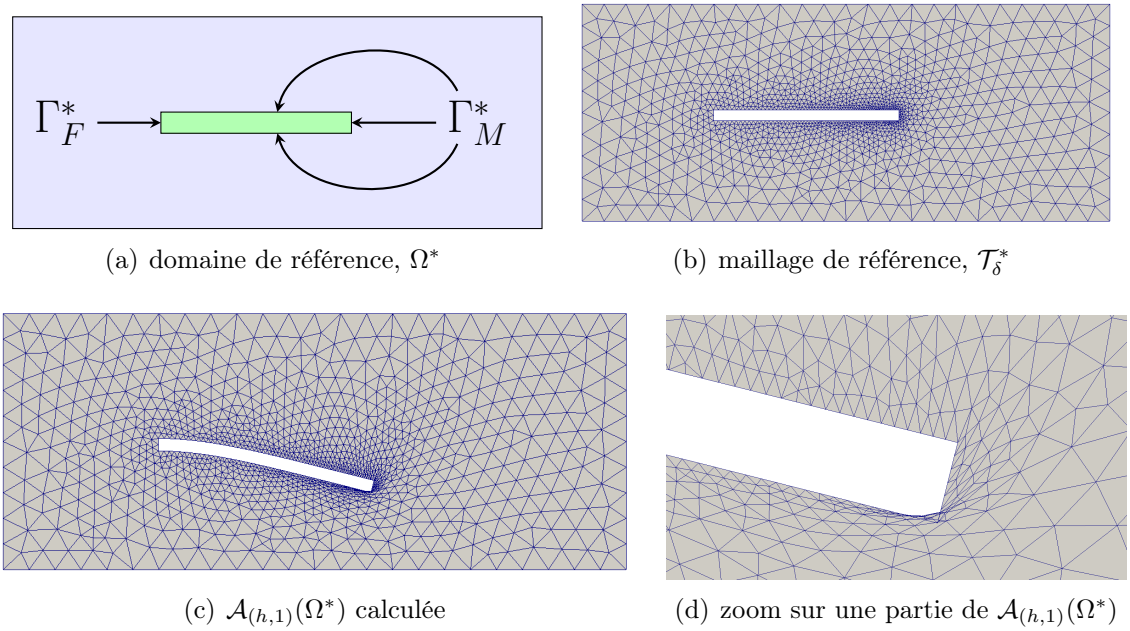
(d) Cylindre, $\Omega^{cy} = \mathcal{A}_{(h,1)}(\Omega^{*,cy})$

FIGURE 2.6 – Exemples d'application de la carte ALE calculée avec l'extension harmonique. Pour chaque exemple, nous avons le domaine de référence et le domaine déformé avec le champ de déplacement calculé.

va être soumise à son propre poids avec une gravité dirigée vers le bas. Cette structure va alors s'affaisser au cours du temps et en jouant sur le paramètre de masse volumique de la barre ou de force de gravité, nous pourrons obtenir des déformations réalistes de plus ou moins grandes amplitudes. De plus, nous faisons ce choix, car cela nous place dans un contexte d'application qui nous intéressera au chapitre 9. Nous ne décrirons pas plus la partie mécanique des solides dans ce chapitre, celle-ci sera faite en détail au chapitre 5.

Nous mettons maintenant en oeuvre ce test avec comme maillage de référence pour la carte ALE le maillage illustré avec la figure 2.7(b). Nous avons raffiné ce maillage proche de la barre pour avoir une description précise de la surface déformée. On commence par calculer un déplacement de la structure, puis nous appliquons l'extension harmonique standard pour la construction de $\mathcal{A}_{(h,1)}$. Le résultat est illustré avec les figures 2.7(c) et 2.7(d). Nous constatons alors que l'extension harmonique standard montre déjà ses faiblesses pour un premier déplacement pas très important. Des éléments du maillage sont devenus invalides, ils se sont *retournés*.

Pour pouvoir générer un maillage correct, nous allons essayer d'utiliser une technique proposée dans [89]. Le problème d'extension harmonique (2.2) est légèrement modifié et



(a) domaine de référence, Ω^*

(b) maillage de référence, $\mathcal{T}_\delta^*$

(c) $\mathcal{A}_{(h,1)}(\Omega^*)$ calculée

(d) zoom sur une partie de $\mathcal{A}_{(h,1)}(\Omega^*)$

FIGURE 2.7 – Géométrie du test et illustration des limites d'utilisation de la méthode basé sur l'extension harmonique standard.

consiste à trouver $\boldsymbol{\zeta}_{(h,1)}$ vérifiant :

$$\begin{cases} -(1+\tau)\,\Delta\boldsymbol{\zeta}_{(h,1)} = 0 \quad , & \text{dans } \Omega_\delta^* \\ \boldsymbol{\zeta}_{(h,1)} = \boldsymbol{g}_{\boldsymbol{\zeta}} \quad , & \text{sur } \Gamma_M^* \\ \boldsymbol{\zeta}_{(h,1)} = \mathbf{0} \quad , & \text{sur } \Gamma_F^* \\ \nabla\boldsymbol{\zeta}_{(h,1)}\,\boldsymbol{n}^* = \mathbf{0} \quad , & \text{sur } \Gamma_N^* \end{cases}$$

où τ est une fonction P^0 discontinu qui empêche la déformation des plus petits éléments du maillage. Pour chaque élément e du maillage $\mathcal{T}_\delta$, nous définissons :

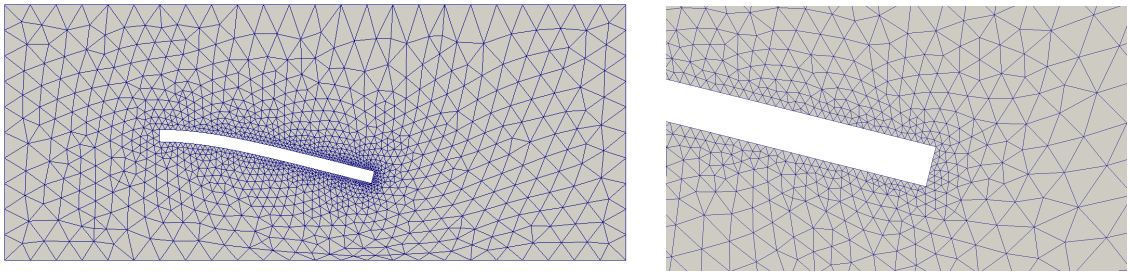
$$\tau|_e = \frac{1 + \Delta_{min}/\Delta_{max}}{\Delta_e/\Delta_{max}}$$

avec :

- Δ_{min} : l'aire/le volume du plus petit élément e
- Δ_{max} : l'aire/le volume du plus grand élément e
- Δ_e : l'aire/le volume de l'élément courant e

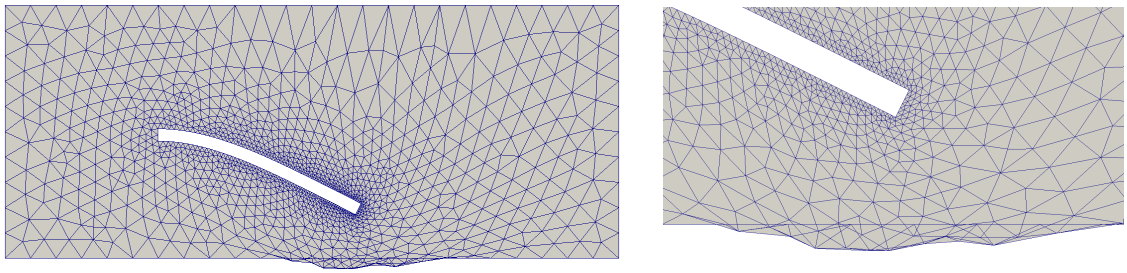
Ainsi nous cherchons plutôt à déformer les grands éléments, ce qui devrait permettre de résoudre le problème que nous avons rencontré, car les éléments les plus petits se trouvent proche de la déformation. On constate bien le bon comportement de cette extension harmonique modifiée avec les figures 2.8(a) et 2.8(b).

Cependant cette technique admet aussi ses limites quand le déplacement devient de plus en plus important. Comme le montre les figures 2.9(a) et 2.9(b), nous obtenons encore une fois un *retournement* d'élément, mais dans ce cas ce sont les grands éléments qui sont devenus invalides.



(a) $\mathcal{A}_{(h,1)}(\Omega^*)$ calculée (b) zoom sur une partie de $\mathcal{A}_{(h,1)}(\Omega^*)$

FIGURE 2.8 – Application de la méthode basée sur l'extension harmonique modifiée.



(a) $\mathcal{A}_{(h,1)}(\Omega^*)$ calculée (b) zoom sur une partie de $\mathcal{A}_{(h,1)}(\Omega^*)$

FIGURE 2.9 – Illustration des limites d'utilisation de la méthode basée sur l'extension harmonique modifiée.

Les équations de Winslow

Nous allons maintenant présenter une méthode alternative à l'extension harmonique, les équations de Winslow [164, 108, 90]. Cette méthode fait partie de la catégorie des *elliptic smoother* qui est une technique couramment utilisée pour la génération de maillage. Ces équations sont dérivées de l'opérateur de Laplace ou de Poisson mais elles sont appliquées au domaine physique (*i.e.* au domaine en mouvement). Nous considérons dans la suite l'opérateur de Laplace suivant comme modèle des équations de Winslow :

$$\nabla^2 \left(\left(\mathcal{A}_{(h,1)} \right)^{-1} \right) = 0 \tag{2.6}$$

Cet opérateur elliptique satisfait le principe du maximum, c'est-à-dire que les extremums de la solution de cette équation se trouvent sur le bord du domaine. Grâce à cette propriété, la carte ALE générée aura moins tendance à causer le pliage d'éléments du maillage comme cela pouvait être le cas avec l'extension harmonique, figures 2.5 ou 2.10. Une autre propriété importante de ce système elliptique est la *régularité* de sa solution qu'il génère, et par conséquent la qualité des éléments résultants à l'intérieur du domaine.

Cependant, nous ne pouvons résoudre directement ce problème dans le domaine déformé, car celui-ci est inconnu. Le principe des équations de Winslow est de transformer les équations précédentes dans le domaine de calcul. Nous obtenons alors un système d'équations non linéaires à résoudre où l'inconnue est la carte ALE $\mathcal{A}_{(h,1)}$. Dans ce système, nous faisons apparaitre le tenseur métrique $(g_{ij})_{i,j=1,..,d}$ qui permet de transformer les coordonnées contravariantes en coordonnées covariantes. Ce tenseur s'exprime par la

formule suivante :

$$g_{ij} = \frac{\partial}{\partial x_i}\mathcal{A}_{(h,1)} \cdot \frac{\partial}{\partial x_j}\mathcal{A}_{(h,1)}$$

Ainsi, nous pouvons maintenant présenter la formulation complète des équations de Winslow dans le repère de calcul. Nous commençons par décrire le cas 2D,

$$g_{11}\frac{\partial^2}{\partial x^2}\mathcal{A}_{(h,1)} - 2g_{12}\frac{\partial^2}{\partial x \partial y}\mathcal{A}_{(h,1)} + g_{22}\frac{\partial^2}{\partial y^2}\mathcal{A}_{(h,1)} = \mathbf{0}$$

et nous donnons également le détail des coefficients de la métrique qui interviennent dans l'équation précédente :

$$\begin{aligned} g_{11} &= \left(\frac{\partial}{\partial y}\mathcal{A}_{(h,1)x}\right)^2 + \left(\frac{\partial}{\partial y}\mathcal{A}_{(h,1)y}\right)^2 \\ g_{12} &= \frac{\partial}{\partial x}\mathcal{A}_{(h,1)x}\frac{\partial}{\partial y}\mathcal{A}_{(h,1)x} + \frac{\partial}{\partial x}\mathcal{A}_{(h,1)y}\frac{\partial}{\partial y}\mathcal{A}_{(h,1)y} \\ g_{22} &= \left(\frac{\partial}{\partial x}\mathcal{A}_{(h,1)x}\right)^2 + \left(\frac{\partial}{\partial x}\mathcal{A}_{(h,1)y}\right)^2 \end{aligned}$$

Pour l'extension au 3D, l'équation se complexifie davantage, mais l'ensemble des ingrédients nécessaires est identique au cas 2D. Voici donc la formulation des équations de Winslow 3D :

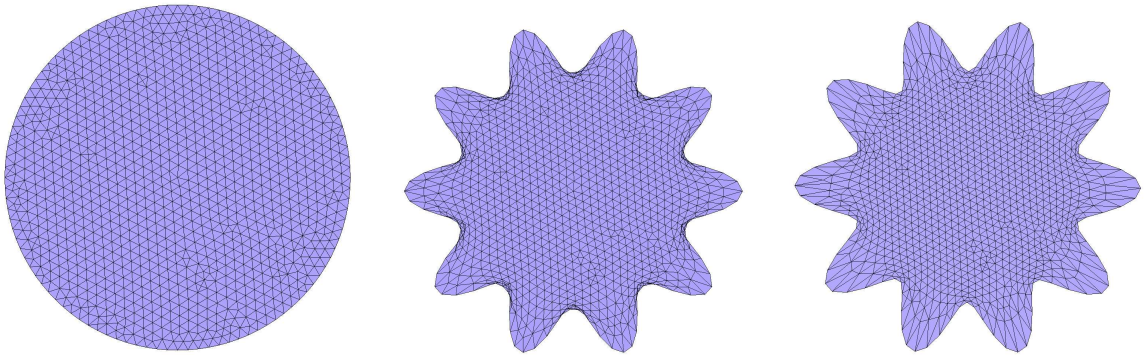
$$\begin{aligned} &\left(g_{22}g_{33} - (g_{23})^2\right)\frac{\partial^2}{\partial x^2}\mathcal{A}_{(h,1)} + \left(g_{11}g_{33} - (g_{13})^2\right)\frac{\partial^2}{\partial y^2}\mathcal{A}_{(h,1)} + \left(g_{11}g_{22} - (g_{12})^2\right)\frac{\partial^2}{\partial z^2}\mathcal{A}_{(h,1)} \\ &\quad + 2\left(g_{23}g_{13} - g_{12}g_{33}\right)\frac{\partial^2}{\partial x \partial y}\mathcal{A}_{(h,1)} + 2\left(g_{12}g_{23} - g_{22}g_{13}\right)\frac{\partial^2}{\partial x \partial z}\mathcal{A}_{(h,1)} \\ &\quad + 2\left(g_{13}g_{12} - g_{23}g_{11}\right)\frac{\partial^2}{\partial y \partial z}\mathcal{A}_{(h,1)} = \mathbf{0} \end{aligned}$$

D'un point de vue numérique, uniquement la version 2D a été implémentée. La version 3D est en cours de réalisation.

Comparaison entre l'extension harmonique et Winslow

Avec les résultats illustrés sur les figures 2.5 ou 2.10, nous voyons clairement que la qualité des maillages générés par les équations de Winslow est meilleure. L'extension harmonique est moins robuste dès que les déplacements deviennent importants.

Par contre, le désavantage des équations de Winslow est le coût de calcul de la solution numérique contrairement à l'extension harmonique qui est très efficace. Ce coût important est dû principalement au fait qu'il faille résoudre un système non linéaire. Pour cela, nous utilisons une méthode itérative avec un algorithme de point fixe ou une méthode de Newton. Le premier a l'avantage d'être plus simple à mettre en oeuvre et d'être aussi un peu plus robuste que l'algorithme de Newton, dans le sens où le Newton doit être initialisé avec une solution *assez proche* de la solution recherchée. Par contre, le nombre d'itérations nécessaire pour atteindre la convergence dans la résolution itérative est plus faible



(a) Maillage de référence (b) Harmonique (c) Winslow

FIGURE 2.10 – Comparaison entre la méthode basée sur l'extension harmonique et la méthode utilisant les équations de Winslow.

avec l'algorithme de Newton (ce qui peut être un atout). Une comparaison numérique de performance a été faite pour l'application représentée avec la figure 2.10. Les résultats se trouvent dans la table 2.1 et montrent que l'algorithme de point fixe est plus efficace que le Newton. Nous avons également utilisé une méthode mixte où l'on commence par quelques itérations de point fixe puis on applique la méthode de Newton. L'intérêt est de mieux initialiser l'algorithme de Newton. Cependant, on constate que le point fixe est toujours plus performant. Un dernier test a été réalisé avec la méthode mixte, mais cette fois-ci, nous utilisons un quasi-Newton à la place de Newton, c'est-à-dire que la jacobienne n'est calculée qu'une seule fois. On obtient alors une très bonne accélération de notre solveur, qui améliore presque d'un facteur 2 l'algorithme de point fixe. Cette alternative de quasi-Newton n'est cependant pas utilisable directement, il faut initialiser correctement le départ de cet algorithme pour qu'il ne diverge pas.

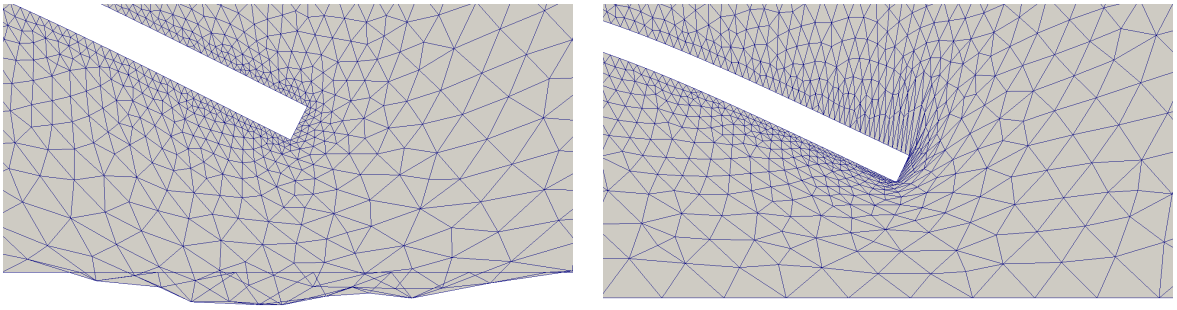
	Solveur	Nb Iterations	temps CPU (s)	Validité
Harmonique	Linéaire	1	0.02	Non
Winslow	Newton	16	13.8	Oui
Winslow	Point Fixe	29	7.7	Oui
Winslow	Point Fixe + Newton	7 + 10	9.5	Oui
Winslow	Point Fixe + Quasi-Newton	7 + 12	4.6	Oui

TABLE 2.1 – Performances entre l'extension harmonique et les équations de Winslow.

Enfin, nous appliquons le cas test présenté dans l'extension harmonique modifiée avec la barre qui déforme le rectangle. Nous rappelons (figure 2.11(a)) que l'application de l'extension harmonique était assez limitée pour ce test. Le résultat pour les équations de Winslow est montré avec la figure 2.11(b) et l'on constate que le maillage obtenu n'est pas de bonne qualité. Aucune de nos méthodes n'est donc employable pour ce cas test, et donc d'autres solutions doivent être envisagées comme l'adaptation de maillage, l'utilisation des modèles élastiques incompressibles en grandes déformations ou encore le remaillage complet du domaine.

2.2 Construction de la carte ALE d'ordre élevé

Nous présentons maintenant la correction à appliquer pour obtenir la carte ALE d'ordre élevé. Elle vise uniquement les éléments qui touchent le bord Γ_M^*, noté $\mathcal{T}_{(h,1)}^{*,b}$. Dans



(a) Extension harmonique (b) Winslow

FIGURE 2.11 – Illustration des faiblesses ou limites d'utilisation pour l'extension harmonique et les équations de Winslow.

chaque élément $K^* \in \mathcal{T}^{*,b}_{(h,1)}$, nous allons chercher la carte correctrice $\mathcal{A}_{(h_{K^*},N)}$ vérifiant le problème elliptique suivant :

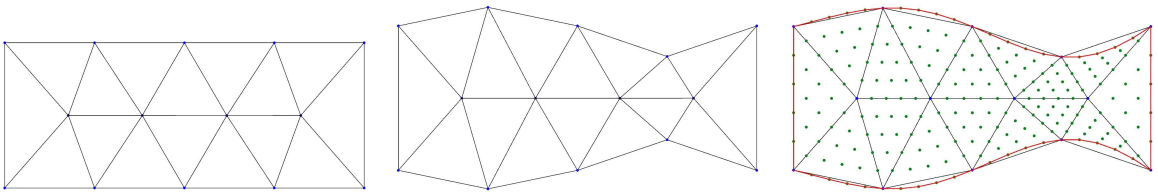
Trouver $\mathcal{A}_{(h_{K^*},N)} \in [\mathbb{P}^N(K^*)]^d$ telle que :

$$\begin{cases} \displaystyle\int_{K^*} \nabla \mathcal{A}_{(h_{K^*},N)} : \nabla \mathbf{z} \; dx = 0, & \forall \mathbf{z} \in [\mathbb{P}^N(K^*)]^d \\ \mathcal{A}_{(h_{K^*},N)}(\boldsymbol{x}^*) = \boldsymbol{x}^* + \boldsymbol{g_\zeta} - \mathcal{A}_{(h,1)}(\boldsymbol{x}^*), & \forall \boldsymbol{x}^* \in \partial K^* \cap \Gamma^*_M \\ \mathcal{A}_{(h_{K^*},N)} = \mathbf{0}, & \text{sinon sur } \partial K^*. \end{cases}$$

Ainsi, la carte ALE d'ordre élevé $\mathcal{A}_\delta \equiv \mathcal{A}_{(h,N)}$ est obtenue en ajoutant à $\mathcal{A}_{(h,1)}$ la correction $\mathcal{A}_{(h_{K^*},N)}$ dans chaque élément de $\mathcal{T}^{*,b}_{(h,1)}$:

$$\mathcal{A}_{(h,N)}(\mathbf{x}^*) = \mathcal{A}_{(h,1)}(\mathbf{x}^*) + \sum_{K^* \in \mathcal{T}^{*,b}_{(h,1)}} \mathcal{A}_{(h_{K^*},N)}(\mathbf{x}^*)$$

Pour terminer la présentation de la méthode, nous illustrons les deux étapes de construction avec la figure 2.12. Nous partons d'un maillage de référence représenté par la figure 2.12(a), puis nous calculons le déplacement d'ordre 1 à l'aide d'un déplacement imposé sur le bord haut et bas, figure 2.12(b). Et enfin, nous appliquons la correction sur les éléments frontière ce qui permet d'obtenir le déplacement d'ordre élevé, figure 2.12(c).



(a) Domaine de référence (b) Construction d'ordre 1 (c) Correction pour l'ordre élevé

FIGURE 2.12 – Etapes de construction d'une carte $\mathcal{A}_{(h,5)}$.

3 Propriétés de la carte ALE d'ordre élevé

Nous allons maintenant énoncer une propriété de la carte ALE d'ordre élevé que nous venons de construire.

Proposition 1 (Propriétés de $\mathcal{A}_{(h,N)}$)**.** *Sous les hypothèses de $\mathcal{T}_\delta$ présentées en début de chapitre et en admettant que le déplacement imposé $\boldsymbol{\eta} \in \boldsymbol{H}^{m-1/2}(\Gamma_M^*)$, pour $m \geqslant 1$, alors $\mathcal{A}_{(h,N)} \in [P_c^N(\mathcal{T}_{(h,1)}^{*,b})]^d$ — jouit de propriétés d'approximation optimale* i.e. *l'approximation sur la frontière est $\mathcal{O}\left(h^{N+1}\right)$ dans la L^2-norme — et $\mathcal{A}_{(h,N)} \in [P_c^1(\mathcal{T}_{(h,1)}^* \backslash \mathcal{T}_{(h,1)}^{*,b})]^d$.*

Démonstration. Nous allons présenter une démonstration pour le cas $d = 2$ mais le cas $d = 3$ peut être traitées de la même façon. On peut alors remarquer dans un premier temps que l'erreur :

$$\|\mathcal{A}_{h,N}(\mathbf{x}^*) - (\mathbf{x}^* + \boldsymbol{\eta}(\mathbf{x}^*))\|_{[L^2(\Gamma_M^*)]^d} \tag{2.7}$$

est égale à

$$\left\{ \|(\mathcal{A}_{h,N}(\mathbf{x}^*) - (\mathbf{x}^* + \boldsymbol{\eta}(\mathbf{x}^*))) \cdot \mathbf{e}_1\|^2_{L^2(\Gamma_M^*)} + \|(\mathcal{A}_{h,N}(\mathbf{x}^*) - (\mathbf{x}^* + \boldsymbol{\eta}(\mathbf{x}^*))) \cdot \mathbf{e}_2\|^2_{L^2(\Gamma_M^*)} \right\}^{1/2} \tag{2.8}$$

D'autre part, dans toutes les faces $F \in \partial K^* \cap \Gamma_M^*$, la carte ALE $\mathcal{A}_{h,N}$ est obtenue en déplaçant les degrés de liberté de la face F (correspondant aux points d'interpolation de l'espace de fonction $\mathbb{P}_N(K^*)$) par la fonction déplacement $\boldsymbol{\eta}$ imposée.

$$\mathcal{A}_{h,N}(\mathbf{x}^*) \cdot \mathbf{e}_i = \mathcal{I}_N(\eta_i)(\mathbf{x}^*) + x_i^*, \qquad \mathbf{x}^* \in F \tag{2.9}$$

où x_i^* (resp $\mathbf{x}^*$) est la $i^{\text{ème}}$ composante de $\mathbf{x}^*$ (resp $\boldsymbol{\eta}$) et $\mathcal{I}_N(\eta_i)$ désigne la fonction polynomiale appartenant à $\mathbb{P}_N(K^*)$ qui interpole η_i aux points de Gauss-Lobatto. En combinant les relations (2.8) et (2.9), nous arrivons alors à l'égalité suivante :

$$\begin{aligned}
\|\mathcal{A}_{h,N}(\mathbf{x}^*) - (\mathbf{x}^* + \boldsymbol{\eta}(\mathbf{x}^*))\|_{[L^2(\Gamma_M^*)]^d} &= \left\{ \sum_{i=1}^{d} \|(\mathcal{A}_{h,N}(\mathbf{x}^*) - (\mathbf{x}^* + \boldsymbol{\eta}(\mathbf{x}^*))) \cdot \mathbf{e}_i\|^2_{L^2(\Gamma_M^*)} \right\}^{1/2} \\
&= \left\{ \sum_{i=1}^{d} \|\mathcal{I}_N(\eta_i)(\mathbf{x}^*) + x_i^* - (x_i^* + \eta_i(\mathbf{x}^*))\|^2_{L^2(\Gamma_M^*)} \right\}^{1/2} \\
&= \left\{ \sum_{i=1}^{d} \|\mathcal{I}_N(\eta_i)(\mathbf{x}^*) - \eta_i(\mathbf{x}^*)\|^2_{L^2(\Gamma_M^*)} \right\}^{1/2}
\end{aligned} \tag{2.10}$$

Nous rappelons maintenant que n'importe quelle face F peut être reliée à l'intervalle $I = [-1, 1]$ par une transformation géométrique affine $\boldsymbol{\varphi}_F^{\text{geo}} : I \longrightarrow F$, dont le jacobien est constant. Une face F est caractérisée par ces deux extrémités de coordonnées (x_0, y_0) et (x_1, y_1). La transformation $\boldsymbol{\varphi}_F^{\text{geo}}$ est alors définie par :

$$\boldsymbol{\varphi}_F^{\text{geo}}(\hat{x}) = \left(\frac{h_x}{2}\hat{x} + \frac{x_1 + x_0}{2}, \frac{h_y}{2}\hat{x} + \frac{y_1 + y_0}{2} \right), \qquad \hat{x} \in I \tag{2.11}$$

avec $h_x = x_1 - x_0$ et $h_y = y_1 - y_0$. Les longueurs h_x et h_y sont reliées avec la taille de F, notée h_F, par la relation $h_F = \sqrt{h_x^2 + h_y^2}$. Nous appliquons ensuite à la norme présente dans (2.10) la décomposition suivante :

$$\|\mathcal{I}_N(\eta_i)(\mathbf{x}^*) - \eta_i(\mathbf{x}^*)\|^2_{L^2(\Gamma_M^*)} = \sum_{F \in \Gamma_M^*} \|\mathcal{I}_N(\eta_i)(\mathbf{x}^*) - \eta_i(\mathbf{x}^*)\|^2_{L^2(F)} \tag{2.12}$$

En utilisant l'équation (2.11), chaque norme du second membre de l'égalité précédente peut être exprimée à l'aide d'une norme similaire sur I :

$$\|\mathcal{I}_N(\eta_i)(\mathbf{x}^*) - \eta_i(\mathbf{x}^*)\|^2_{L^2(F)} = \frac{h_F}{2} \left\|\hat{\mathcal{I}}_N(\hat{\eta}_i) - \hat{\eta}_i\right\|^2_{L^2(I)} \tag{2.13}$$

où $\hat{\eta}_i = \eta_i \circ \boldsymbol{\varphi}_F^{\text{geo}}$ et $\hat{\mathcal{I}}_N$ est l'opérateur d'interpolation nodal équivalent sur I associé aux points de Gauss-Lobatto.

Si $\hat{\eta}_i \in H^m(I)$ alors l'estimation suivante est vérifiée (voir Quarteroni, Zang, Hussaini et Canutto [35]) :

$$\left\|\hat{\mathcal{I}}_N(\hat{\eta}_i) - \hat{\eta}_i\right\|_{L^2(I)} \leqslant \hat{C} N^{-m} |\hat{\eta}_i|_{H^{m;N}(I)} \tag{2.14}$$

avec

$$|\hat{\eta}_i|_{H^{m;N}(I)} = \left\{ \sum_{k=\min(m,N+1)}^{m} \left\| \frac{d^k}{dt^k} \hat{\eta}_i \right\|^2_{L^2(I)} \right\}^{1/2} \tag{2.15}$$

En utilisant un argument similaire qui a été appliqué pour obtenir l'égalité (2.13) et la règle de dérivation en chaîne, nous pouvons en déduire qu'il existe une constante $C_1 > 0$ indépendante de N, m et h_F telle que :

$$|\hat{\eta}_i|^2_{H^{m;N}(I)} \leqslant C_1 h_F^{2\min(m,N+1)-1} |\eta_i|^2_{H^{m;N}(F)} \tag{2.16}$$

où

$$|\eta_i|_{H^{m;N}(F)} = \left\{ \sum_{k=\min(m,N+1)}^{m} \sum_{j=1}^{d} \left\| D_j^k \eta_i \right\|^2_{L^2(F)} \right\}^{1/2} \tag{2.17}$$

et D_j^k représente la $k^{\text{ème}}$ dérivée dans la direction x_j. En combinant (2.16), (2.14) et (2.13), nous pouvons conclure qu'il existe une constante $C_2 > 0$ indépendante de N et h_F telle que :

$$\|\mathcal{I}_N(\eta_i)(\mathbf{x}^*) - \eta_i(\mathbf{x}^*)\|_{L^2(\Gamma_M^*)} \leqslant C_2 \left\{ \sum_{F \in \Gamma_M^*} h_F^{2\min(N+1,m)} N^{-2m} |\eta_i|^2_{H^{m;N}(F)} \right\}^{1/2} \tag{2.18}$$

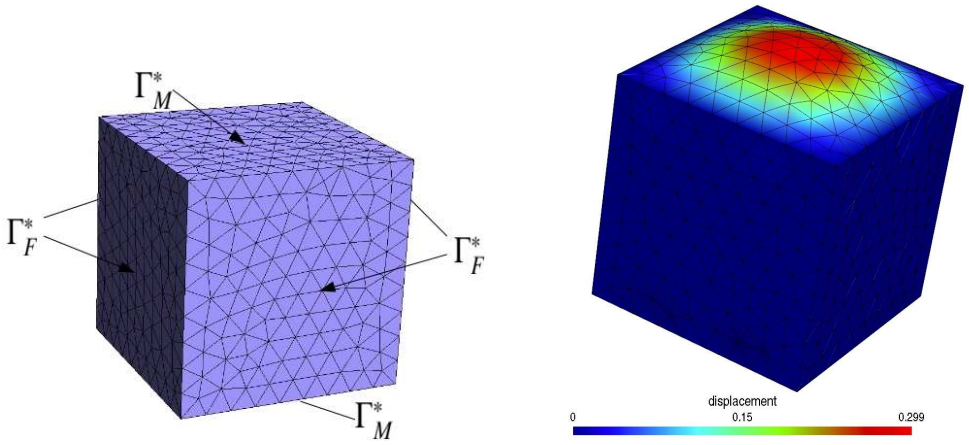
En supposant qu'il existe deux constantes $c < 1$ et $c' < 1$ telles que $ch < h_F < c'h$ pour tout F, nous obtenons finalement :

$$\|\mathcal{I}_N(\eta_i)(\mathbf{x}^*) - \eta_i(\mathbf{x}^*)\|_{L^2(\Gamma_M^*)} \leqslant C h^{\min(N+1,m)} N^{-m} |\eta_i|_{H^{m;N}(\Gamma_M^*)}. \tag{2.19}$$

□

4 Vérifications numériques

Nous passons à présent à des vérifications numériques de la proposition 1 avec les deux cas tests suivants. Pour chacun d'eux, nous calculerons la carte ALE pour des ordres géométriques allant de 1 à 4. Nous validerons les résultats à l'aide des graphes de convergence.



(a) $\Omega^{*,cu}$ (b) $\Omega^{cu} = \mathcal{A}_{(h,4)}(\Omega^{*,cy})$

FIGURE 2.13 – Domaine de référence $\Omega^{*,cu}$ (à gauche) et domaine déformé Ω^{cu} (à droite) utilisant un maillage d'ordre 4 et coloré avec la magnitude du déplacement $\boldsymbol{\zeta}_{(h,4)}$.

Nous commençons par choisir le domaine de référence $\Omega^{*,cu}$ comme un cube, représenté par la figure 2.13(a) et décrit par :

$$\Omega^{*,cu} = \left\{(x^*, y^*, z^*) \in [-1, 1]^3\right\} \tag{2.20}$$

Le déplacement imposé sur la frontière Γ_M^* est :

$$\boldsymbol{g}_{\boldsymbol{\zeta}}(\mathbf{x}^*) = 0.1 \cos(\frac{\pi}{2} x^*) \cos(\frac{\pi}{2} y^*) \boldsymbol{n}^* \tag{2.21}$$

Nous pouvons voir un résultat obtenu pour ce test à l'ordre 4 en géométrie avec la figure 2.13(b). Nous avons affiché sur la figure 2.14 les courbes de convergence mesurant la quantité $\|\mathcal{A}_{(h,N)}(\boldsymbol{x}^*) - (\boldsymbol{x}^* + \boldsymbol{g}_{\boldsymbol{\zeta}}(\boldsymbol{x}^*))\|_{[L^2(\Gamma_M^*)]^d}$. Celles-ci sont tracées en échelle log-log, ce qui permet de montrer l'optimalité de la méthode.

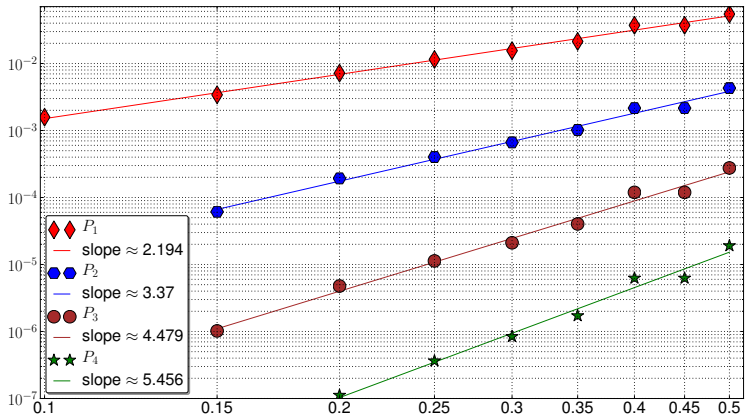


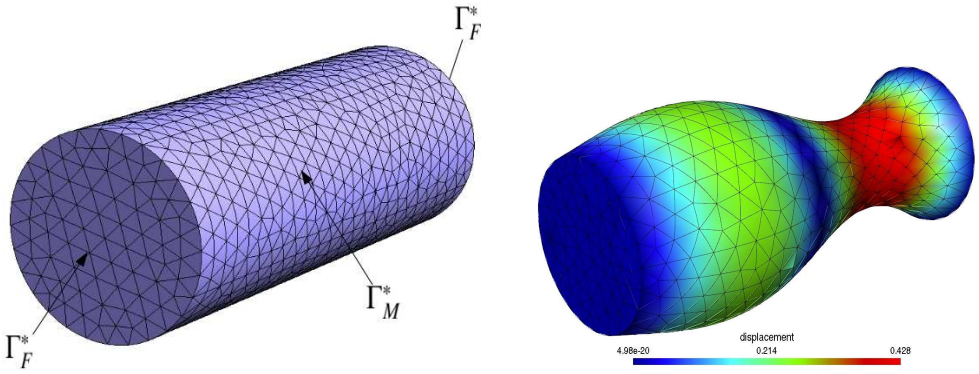
FIGURE 2.14 – Graphe de convergence de $\mathcal{A}_{(h,N)}$, $N = 1, 2, 3, 4$ en fonction de h (échelle log-log).

Nous considérons pour le deuxième cas test un cylindre représenté avec la figure 2.15(a) et défini par

$$\Omega^{*,cy} = \left\{(x^*, y^*, z^*) \in \mathbb{R}^3 : \ x^* \in [0,5], y^{*2} + z^{*2} \leq 0.5^2\right\} \tag{2.22}$$

Le déplacement imposé sur la partie mobile du cylindre s'exprime par :

$$\boldsymbol{\eta}^{cy}(\boldsymbol{x}^*) = 0.2 \exp\left(\frac{x^*}{5}\right) \sin\left(\frac{\pi x^*}{2.5}\right) \boldsymbol{n}^* \tag{2.23}$$



(a) $\Omega^{*,cy}$ (b) $\Omega^{cy} = \mathcal{A}_{(h,4)}(\Omega^{*,cy})$

FIGURE 2.15 – Domaine de référence $\Omega^{*,cy}$ (à gauche) et domaine déformé Ω^{cy} (à droite) utilisant un maillage d'ordre 4 et coloré avec la magnitude du déplacement $\boldsymbol{\zeta}_{(h,4)}$.

La figure 2.15(b) montre le domaine déformé calculé à l'aide de la carte $\mathcal{A}_{(h,4)}$. Finalement, nous montrons avec la figure 2.16 le taux de convergence de la quantité $\|\mathcal{A}_{(h,N)}(\boldsymbol{x}^*) - (\boldsymbol{x}^* + \boldsymbol{g}_{\boldsymbol{\zeta}}(\boldsymbol{x}^*))\|_{[L^2(\Gamma_M^*)]^d}$. Cela confirme encore le résultat de la proposition 1 pour les différents ordres géométriques.

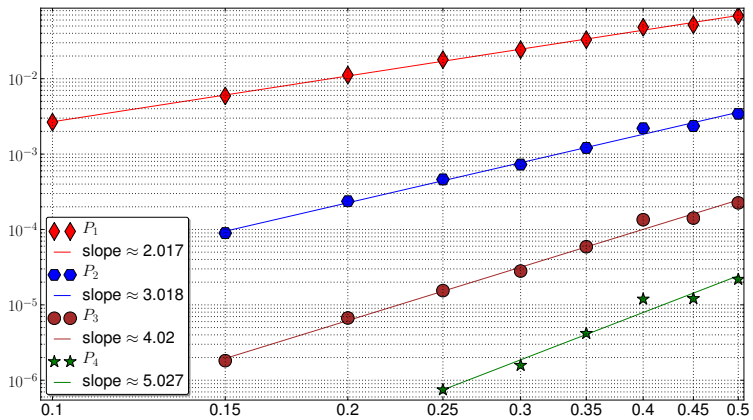


FIGURE 2.16 – Graphe de convergence de $\mathcal{A}_{(h,N)}$, $N = 1, 2, 3, 4$ en fonction de h (échelle log-log).

5 Conclusion

Nous venons de présenter dans ce chapitre un des outils indispensable pour l'implémentation de notre modèle d'interaction fluide structure. Il s'agit de la construction de la carte ALE qui permet de calculer le déplacement d'un maillage à partir d'un déplacement imposé sur l'un des bords. Cette technique est couramment utilisée dans la littérature pour la simulation des écoulements sanguins. Nous avons apporté dans ce chapitre la description d'une nouvelle méthode pour calculer une carte ALE d'ordre élevé. Cette opération faite en deux étapes permet d'obtenir le déplacement de maillage d'ordre élevé.

La première étape consiste à réutiliser les diverses méthodes qui existent pour calculer une carte ALE associée à un maillage d'ordre 1. Nous avons décrit la méthode standard, l'extension harmonique, qui est très efficace pour les petits déplacements. Cet opérateur a été implémenté en 2D et 3D. Nous avons également présenté une deuxième méthode basée sur les équations de Winslow. Elle permet de générer des déplacements de maillage plus importants et des maillages de meilleure qualité comparée à la méthode précédente. Cependant, le coût de construction de cet opérateur est beaucoup plus important. Dans cette thèse, nous avons pour le moment implémenté uniquement la version 2D. L'extension au 3D est en cours de réalisation.

La seconde étape de notre construction consiste à appliquer une correction à la carte ALE calculée précédemment pour obtenir le déplacement d'ordre élevé. Cette correction peut s'appliquer uniquement sur les éléments frontières du maillage grâce à l'hypothèse faite sur la nature du maillage d'ordre élevé (éléments courbes uniquement sur la frontière). Ainsi, cette restriction nous permet de réduire considérablement le coût de cette opération. Un autre avantage à cette construction est qu'elle ne dépend pas de la méthode utilisée pour le calcul de la carte d'ordre 1. Toutes les techniques sont donc compatibles. L'implémentation de cette correction a été faite en 2D et 3D. Cela nous a permis de vérifier numériquement la propriété d'optimalité sur la précision de la déformation de la frontière (proposition 1).

Chapitre 3

Méthode de la frontière élargie

Le sang est composé d'un fluide, le plasma, et d'un grand nombre de cellules telles que les globules rouges, les globules blancs et les plaquettes. Pour pouvoir modéliser avec précision la dynamique du plasma, il est important de prendre en compte la présence de ces entités. Ainsi, nous allons nous intéresser à la modélisation d'un fluide en présence d'un grand nombre d'obstacles. On parle alors de modèle défini dans un domaine perforé. Pour ce chapitre, nous considèrerons que les obstacles sont fixes. Nous étendrons ce travail au chapitre 6 pour la prise en compte d'obstacles en mouvement à l'aide d'un modèle d'interaction fluide structure.

Pour traiter ce type d'application, l'approche standard consisterait à utiliser directement les domaines perforés. Cependant, la complexité de ces géométries peut être une barrière à cette méthode. En effet, les domaines composés d'un grand nombre de perforations sont à priori difficiles à discrétiser. De plus, dans l'idée d'avoir des obstacles en mouvement, il faudrait être capable de pouvoir déformer et/ou remailler le maillage. C'est pourquoi nous avons choisi d'utiliser une méthode de type domaine fictif appelée méthode de la frontière élargie (*Fat Boundary Method-FBM*). Elle a été initialement proposée par B.Maury [115] pour résoudre des problèmes elliptiques dans des géométries complexes. Elle a ensuite été étendue pour la simulation numérique d'écoulement fluide tridimensionnelle dans [86]. L'idée principale consiste à remplacer le problème initial défini sur une géométrie complexe par un ensemble de sous-problèmes basé sur des géométries plus simples. Les maillages sous-jacents seront générés de manière indépendante, ce qui entrainera la non-conformité des maillages. L'avantage de la FBM comparé aux méthodes de domaines fictifs [70, 71, 72] est qu'elle a été conçue dans le but de récupérer la convergence optimale à tous les ordres, en dépit de la non-conformité de ces maillages. Le comportement optimal de cette méthode a été prouvé théoriquement [26] et vérifié numériquement par plusieurs tests numériques, 1D et 2D dans [26] et 3D dans [25, 86].

Dans ce chapitre, nous commencerons par vérifier le comportement de la FBM pour le cas spectral, c'est-à-dire avec des approximations d'ordre élevé en espace et géométrie. Nous apporterons également une modification à la méthode dans le but de faciliter la prise en compte des perforations de forme complexe. Nous passerons ensuite à l'introduction d'une nouvelle formulation de la FBM dite formulation point-selle. Contrairement à la précédente, cette nouvelle méthode autorisera d'un point de vue théorique le recouvrement des domaines locaux. Nous étendrons également cette formulation aux équations de Stokes et de Navier-Stokes.

1 Principe de la méthode et notations

‘Soit Ω un ouvert borné de $\mathbb{R}^n$ délimité par une frontière Γ. Nous notons par B une collection de N_{perfo} sous-domaines réguliers de mesures positives et par γ sa frontière. Chaque perforation est notée B_i et sa frontière est notée par γ_i (voir la figure 3.1 pour une illustration). Plus précisément

$$B = \bigcup_{i=1}^{N_{perfo}} B_i, \quad \gamma = \bigcup_{i=1}^{N_{perfo}} \gamma_i \tag{3.1}$$

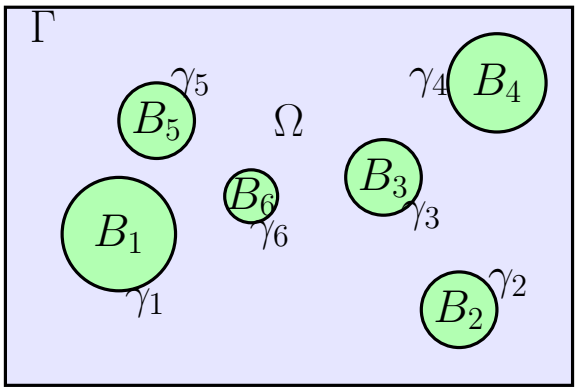


FIGURE 3.1 – Domaine réel avec perforations.

L'idée sous-jacente de la *Fat Boundary Method* est de remplacer le problème initialement posé dans le domaine perforé par plusieurs problèmes couplés de forme plus simple : un problème global associé à l'ensemble du domaine Ω (y compris les perfortations B), et plusieurs problèmes locaux définis sur chaque sous-domaine $\omega_i \subset \Omega \backslash \bar{B}$ délimité par γ_i et par une frontière artificielle de forme assez régulière $\gamma_i' \subset \Omega \backslash \bar{B}$ (voir figure 3.2). On définit aussi les ensembles des domaines locaux ω et frontières fictives γ' par :

$$\omega = \bigcup_{i=1}^{N_{perfo}} \omega_i, \quad \gamma' = \bigcup_{i=1}^{N_{perfo}} \gamma_i' \tag{3.2}$$

Le domaine global étant de forme simple, il est alors possible dans ce cas d'utiliser des solveurs et préconditionneurs très efficaces pour résoudre le problème global. Pour les problèmes locaux, nous pouvons envisager des maillages très fins dans le but d'avoir de meilleures approximations autour des particules. Un autre intérêt majeur de la FBM est de pouvoir traiter des problèmes avec un grand nombre de perforations de forme complexe. En effet, la création de maillages perforés peut se révéler très complexe et ne donne pas beaucoup de liberté dans les niveaux de raffinement. Aussi dans le cas où l'on souhaiterait par exemple déplacer des particules au cours du temps, la méthode FBM est bien adaptée, car il n'y aurait pas besoin de déformer le maillage perforé, voir de le remailler, mais seulement de déplacer les domaines locaux. Nous détaillerons cet aspect dans le chapitre 6.

Dans ce chapitre, nous notons par u une fonction définie sur le domaine global Ω et par $\hat{u}$ ou $\hat{u}_i$ une fonction définie sur le domaine local ω ou ω_i.

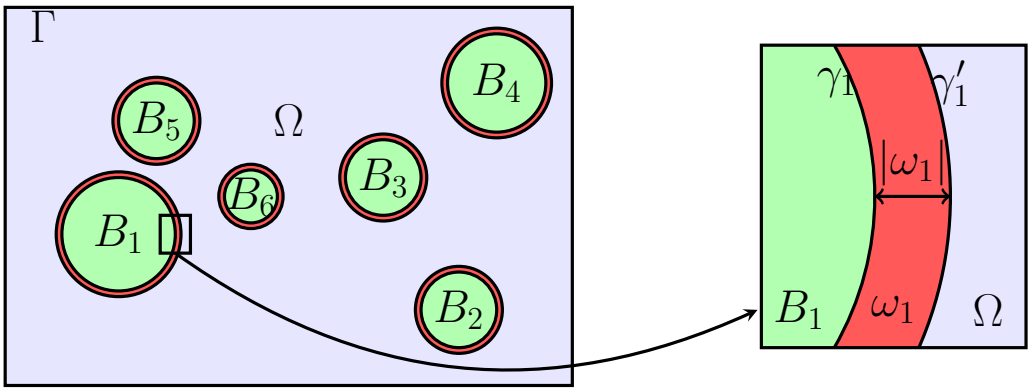


FIGURE 3.2 – Domaines de calcul : le domaine global Ω et plusieurs domaines locaux autour des perforations.

2 Formulation I : elliptique

Pour cette première formulation, nous considérons uniquement le problème elliptique suivant :

Trouver la fonction $\tilde{u}$ dans $\Omega\backslash\bar{B}$ telle que :

$$\begin{cases} -\Delta\tilde{u} = f & \text{dans } \Omega\backslash\bar{B} \\ \tilde{u} = g & \text{sur } \Gamma \\ \tilde{u} = 0 & \text{sur } \gamma \end{cases} \tag{3.3}$$

où $f \in L^2\left(\Omega\backslash\bar{B}\right)$ et $g \in H^{1/2}(\Gamma)$.

Avant de présenter la formulation classique de la FBM, nous commençons par introduire $\bar{f}$ le prolongement de $f \in L^2\left(\Omega\backslash\bar{B}\right)$ par 0 sur Ω, que l'on définit par :

$$\begin{cases} \bar{f}|_{\Omega\backslash\bar{B}} &= f \\ \bar{f}|_{B} &= 0 \end{cases} \tag{3.4}$$

Dans [86], il est démontré que le problème initial (3.3) est équivalent à un ensemble de sous-problèmes couplés. Un problème global qui est défini sur Ω et plusieurs problèmes locaux qui sont définis autour des perforations. Ce nouveau problème couplé se formalise par le système d'équations (3.5)-(3.6) suivant :

Trouver les fonctions $(u, \hat{u}_1, ..., \hat{u}_{N_{perfo}})$ qui vérifient :

$$\text{(Global)} \qquad \left\{ \begin{array}{l} -\Delta u = \bar{f} + \displaystyle\sum_{i=1}^{N_{perfo}} \delta_{\gamma_i} \partial_{\nu_i} \hat{u}_i \quad \text{dans } \Omega \\ u = g \quad \text{sur } \Gamma \end{array} \right\} \tag{3.5}$$

$$\text{(Local}_i\text{)}_{i=1}^{N_{perfo}} \qquad \left\{ \begin{array}{l} -\Delta\hat{u}_i = f \quad \text{dans } \omega_i \\ \hat{u}_i = 0 \quad \text{sur } \gamma_i \\ \hat{u}_i = u_i \quad \text{sur } \gamma_i' \end{array} \right\} \tag{3.6}$$

où nous notons par $\boldsymbol{\nu}_i$ la normale extérieure à γ_i et par $\partial_{\boldsymbol{\nu}_i} v$ la dérivée normale de v. On pose également par δ_{γ_i} la masse de Dirac associée à la frontière γ_i.

Definition 7. *Pour $\eta \in H^{-1/2}(\gamma)$, on note par $\eta\delta_\gamma$ l'élément de $H^{-1}(\Omega)$ défini par*

$$\langle \eta\delta_\gamma, w\rangle_{\frac{1}{2},\gamma} = \int_\gamma \eta w, \quad \forall w \in H_0^1(\Omega) \tag{3.7}$$

où $\langle\cdot,\cdot\rangle_{\frac{1}{2},\gamma}$ désigne le produit de dualité entre $H^{-1/2}(\gamma)$ et $H^{1/2}(\gamma)$.

2.1 Terme source et conditions aux limites

En pratique, le prolongement $\bar{f}$ n'est pas facile à implémenter, sauf dans le cas où l'on peut exprimer la perforation par des équations paramétriques. Dans le cas contraire, nous devons localiser les points des perforations B_i dans le domaine Ω, ce qui peut se révéler très coûteux. Pour pallier à cette difficulté et dans l'hypothèse où nous connaissons f dans tout Ω, nous pouvons transformer l'équation (3.5) de telle manière que $\bar{f}$, au second membre, devienne f. Le problème (3.5) devient :

Trouver la fonction u telle que :

$$\text{(Global)} \qquad \left\{ \begin{array}{c} -\Delta u = f + \displaystyle\sum_{i=1}^{N_{perfo}} \delta_{\gamma_i}\partial_{\nu_i}\hat{u}_i - \sum_{i=1}^{N_{perfo}} \delta_{\gamma_i}\partial_{\nu_i}\mathring{u}_i \ \text{ dans } \Omega \\ u = g \ \text{ sur } \Gamma \end{array} \right\} \tag{3.8}$$

où les $\mathring{u}_i$ sont les solutions d'un problème défini sur chaque B_i que nous allons décrire. Ces nouveaux problèmes sont de petite taille, indépendants les uns des autres et non couplés au problème FBM (3.5)-(3.6). Les $\mathring{u}_i$ sont donc calculés avant la résolution du problème FBM. Ces contributions sont ensuite intégrées au problème FBM. Le coût de ces calculs supplémentaires reste relativement négligeable et leurs indépendances entrainent une parallélisation de l'algorihtme de calcul très naturelle. Ces nouveaux problèmes consistent alors à trouver les fonctions $\mathring{u}_i$ dans chaque B_i telles que :

$$(\text{Perfo}_i)_{i=1}^{N_{perfo}} \left\{ \begin{array}{c} -\Delta\mathring{u}_i = f \ \text{ dans } B_i \\ \mathring{u}_i = 0 \ \text{ sur } \gamma_i \end{array} \right\} \tag{3.9}$$

Dans le problème initial (3.3), nous avons considéré un cas particulier, où nous avions des conditions de Dirichlet homogène sur les perforations γ_i. La formulation avec l'extension $\bar{f}$ peut s'étendre au cas non homogène comme l'explique [86]. Soit $u_i = h_i$ sur γ_i avec $h_i \in H^{1/2}(\gamma_i)$ et $1 \le i \le N_{perfo}$. On prend d'abord en compte cette condition dans les problèmes locaux en changeant la condition de Dirichlet homogène sur les γ_i. Il faut ensuite calculer le relèvement harmonique de h noté $\tilde{h}$ et ajouter une contribution de ce relèvement au second membre de l'équation (3.5) avec le terme :

$$-\sum_{i=1}^{N_{perfo}} \delta_{\gamma_i}\partial_{\nu_i}\tilde{h}_i$$

Pour la version sans l'extension $\bar{f}$, il suffit de modifier les conditions de Dirichlet homogène sur les γ_i dans les problèmes locaux (3.6) et perforations (3.9). Dans la suite, nous traiterons uniquement le cas homogène pour plus de simplicité.

2.2 Formulations variationnelles

Nous présentons maintenant la formulation variationnelle associée à ce problème. Après intégration sur les domaines de calculs et intégration par partie, nous obtenons le système d'équations suivant :

Trouver $(u, \hat{u}_1, ..., \hat{u}_{N_{perfo}}) \in \left(H_g^1(\Omega) \times \left(\otimes_{i=1}^{N_{perfo}} H^1_{0,\gamma_i}(\omega_i)\right)\right)$ telles que :

$$\text{(Global)} \qquad \left\{ \int_\Omega \nabla u \cdot \nabla v = \int_\Omega \bar{f} v + \sum_{i=1}^{N_{perfo}} \langle \delta_{\gamma_i} \partial_{\nu_i} \hat{u}_i, v \rangle_{\frac{1}{2},\gamma_i},\ \forall v \in H_0^1(\Omega) \right\} \tag{3.10}$$

$$(\text{Local}_i)_{i=1}^{N_{perfo}} \quad \left\{ \begin{array}{c} \displaystyle\int_{\omega_i} \nabla \hat{u}_i \cdot \nabla \hat{v}_i = \int_{\omega_i} f\ \hat{v}_i, \quad \forall \hat{v}_i \in H_0^1(\omega_i) \\ \hat{u}_i = u \ \text{ sur } \gamma_i' \end{array} \right\} \tag{3.11}$$

Comme nous cherchons les $\hat{u}_i$ dans $H^1_{0,\gamma_i}(\omega_i)$, chaque dérivée normale $\partial_{\nu_i}\hat{u}_i$ appartient à l'espace $H^{-1/2}(\partial\omega_i)$. Nous pouvons ainsi utiliser la définition 7, ce qui nous permet alors d'écrire :

$$\text{pour } i = 1, ..., N_{perfo}, \quad \langle \delta_{\gamma_i} \partial_{\nu_i} \hat{u}_i, v \rangle_{\frac{1}{2},\gamma_i} = \int_{\gamma_i} \partial_{\nu_i} \hat{u}_i\ v, \quad \forall v \in H_0^1(\Omega) \tag{3.12}$$

La formulation FBM sans l'extension $\bar{f}$ nécessite la résolution de sous-problèmes (3.9) sur chaque perforation B_i et chacun de ses sous-problèmes ne dépend pas des autres. Les formulations variationnelles de ces problèmes s'expriment par :

Trouver $\left(\mathring{u}_1, ..., \mathring{u}_{N_{perfo}}\right) \in \otimes_{i=1}^{N_{perfo}} H_0^1(B_i)$ telles que :

$$(\text{Perfo}_i)_{i=1}^{N_{perfo}} \left\{ \int_{B_i} \nabla \mathring{u}_i \cdot \nabla \mathring{v}_i = \int_{B_i} f\ \mathring{v}_i, \quad \forall \mathring{v}_i \in H_0^1(B_i) \right\} \tag{3.13}$$

Grâce aux solutions des équations (3.13), nous pouvons écrire la formulation variationnelle sans le prolongement de f qui est équivalent à la formulation variationnelle (3.10)-(3.11). Nous utilisons encore la définition 7 avec le nouveau terme ajouté sur Ω à l'aide de la masse de Dirac. Cette formulation est alors décrite par le problème suivant :

Trouver $(u, \hat{u}_1, ..., \hat{u}_{N_{perfo}}) \in \left(H_g^1(\Omega) \times \left(\otimes_{i=1}^{N_{perfo}} H^1_{0,\gamma_i}(\omega_i)\right)\right)$ telles que :

$$\text{(Global)} \qquad \left\{ \begin{array}{c} \displaystyle\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f\ v + \sum_{i=1}^{N_{Perfo}} \int_{\gamma_i} \left(\partial_{\nu_i}\hat{u}_i - \partial_{\nu_i}\mathring{u}_i\right) v, \\ \forall v \in H_0^1(\Omega) \end{array} \right\} \tag{3.14}$$

$$(\text{Local}_i)_{i=1}^{N_{perfo}} \quad \left\{ \begin{array}{c} \displaystyle\int_{\omega_i} \nabla \hat{u}_i \cdot \nabla \hat{v}_i = \int_{\omega_i} f\ \hat{v}_i, \quad \forall \hat{v}_i \in H_0^1(\omega_i) \\ \hat{u}_i = u \ \text{ sur } \gamma_i' \end{array} \right\} \tag{3.15}$$

Nous allons maintenant passer à la formulation discrète de la formulation FBM. Soient Ω_δ, $\omega_{i,\delta}$ et $B_{i,\delta}$ les discrétisations des domaines Ω, ω_i et B_i auxquelles nous associons respectivement les triangulations $\mathcal{T}_{\Omega_\delta}$, $\mathcal{T}_{\omega_{i,\delta}}$ et $\mathcal{T}_{B_{i,\delta}}$. Nous avons noté ici chacune des discrétisations par le même indice $\delta = (h, k)$ pour alléger les notations, mais dans la pratique

ce ne sera pas forcément le cas. En effet, l'intérêt de la FBM est de pouvoir utiliser différentes approximations entre ces domaines et notamment d'être beaucoup plus précis dans les domaines locaux, aussi bien en raffinement h qu'en ordre géométrique k, pour un coût relativement faible.

Nous définissons maintenant les espaces d'approximations $V_\delta(\Omega_\delta)$, $\hat{V}_{i,\delta}(\omega_{i,\delta})$ et $P_\delta(B_{i,\delta})$ associés respectivement au problème global, aux problèmes locaux et aux perforations. Nous avons fait le choix des espaces d'approximation éléments finis continus basés sur l'élément fini de Lagrange d'ordre respectif N, $\hat{N}$ et M :

$$\begin{aligned} V_\delta(\Omega_\delta) &= \left\{v \in H^1_g(\Omega_\delta) \cap P^c_N(\Omega_\delta)\right\} \\ \hat{V}_{i,\delta}(\omega_{i,\delta}) &= \left\{v \in H^1_{0,\gamma_i}(\omega_{i,\delta}) \cap P^c_{\hat{N}}(\omega_{i,\delta})\right\} \\ P_\delta(B_{i,\delta}) &= \left\{v \in H^1_0(B_{i,\delta}) \cap P^c_M(B_{i,\delta})\right\} \end{aligned}$$

Dans le cas où nous avons utilisé la formulation sans l'extension $\bar{f}$, nous présentons la formulation variationnelle discrète associée aux perforations B_i :

Trouver $\left(\mathring{u}_1, ..., \mathring{u}_{N_{perfo}}\right) \in \left(\otimes_{i=1}^{N_{perfo}} P_\delta(B_{i,\delta})\right)$ telles que :

$$(\text{Perfo}_i)_{i=1}^{N_{perfo}} \left\{ \int_{B_{i,\delta}} \nabla \mathring{u}_{i,\delta} \cdot \nabla \mathring{v}_i = \int_{B_{i,\delta}} f\ \mathring{v}_i, \quad \forall \mathring{v}_i \in P_\delta(B_{i,\delta}) \right\} \tag{3.16}$$

Enfin, nous avons la formulation variationnelle discrète de la FBM obtenue à partir des équations continues (3.14)-(3.15) :

Trouver $(u_\delta, \hat{u}_{1,\delta}, ..., \hat{u}_{N_{perfo},\delta}) \in \left(V_\delta(\Omega_\delta) \times \left(\otimes_{i=1}^{N_{perfo}} \hat{V}_{i,\delta}(\omega_{i,\delta})\right)\right)$ telles que :

$$\text{(Global)} \qquad \left\{ \int_{\Omega_\delta} \nabla u_\delta \cdot \nabla v = \int_{\Omega_\delta} f\ v + \sum_{i=1}^{N_{Perfo}} \int_{\gamma_i} \left(\partial_{\nu_i} \hat{u}_{i,\delta} - \partial_{\nu_i} \mathring{u}_{i,\delta}\right) v, \quad \forall v \in H^1_0(\Omega_\delta) \cap P^c_N(\Omega_\delta) \right\} \tag{3.17}$$

$$(\text{Local}_i)_{i=1}^{N_{perfo}} \qquad \left\{ \begin{array}{c} \displaystyle\int_{\omega_{i,\delta}} \nabla \hat{u}_{i,\delta} \cdot \nabla \hat{v}_i = \int_{\omega_{i,\delta}} f\ \hat{v}_i, \quad \forall \hat{v}_i \in H^1_0(\omega_{i,\delta}) \cap P^c_{\hat{N}}(\omega_{i,\delta}) \\ \hat{u}_{i,\delta} = u_\delta \quad \text{sur } \gamma_i' \end{array} \right\} \tag{3.18}$$

Nous allons terminer cette partie par donner une estimation d'erreur de la méthode FBM qui a été démontrée dans [26]. Soit $\tilde{u} \in H^m\left(\Omega \backslash \bar{B}\right)$ la solution exacte. Soit $u_\delta \in V_\delta(\Omega_\delta)$ la solution globale calculée avec la FBM. On note également h_Ω (resp h_ω) la taille caractéristique du maillage global (resp local). En posant $s = \min(N+1, m)$, nous pouvons affirmer :

$$\|\tilde{u} - u_\delta\|_{H^1\left(\Omega \backslash \left(\bar{B} \cup \bar{\omega}\right)\right)} \leq C\left(h_\Omega^{s-1} + h_\omega^{s-1}\right) \|\tilde{u}\|_{H^s\left(\Omega \backslash \bar{B}\right)} \tag{3.19}$$

Cette inégalité montre l'optimalité de la méthode FBM au sens de la méthode des éléments finis classiques. On peut également trouver d'autres estimations d'erreur sur Ω et ω dans [26]. Notamment, il est aussi démontré l'optimalité de la méthode pour la solution locale dans ω.

2.3 Algorithme de résolution

Pour pouvoir résoudre les systèmes d'équations (3.10)-(3.11) ou (3.14)-(3.15), nous allons utiliser un algorithme de point fixe. Le principe de cette méthode est de résoudre successivement les problèmes locaux et le problème global. On utilise la solution globale de l'itération précédente dans les problèmes locaux et les solutions locales dans le problème global. De plus, on applique à chaque itération une relaxation de la solution globale évaluée sur chaque frontière γ_i' dans les conditions aux limites des problèmes locaux. On obtient alors convergence lorsque les solutions locales (et par conséquent la solution globale) ne varient plus à une tolérance donnée. Une démonstration théorique de la convergence d'un tel algorithme est faite dans [26].

Soit k la $k^{\text{ème}}$ itération du point fixe. On note u_δ^k (resp $\hat{u}_{i,\delta}^k$) la solution globale (resp la solution locale du $i^{\text{ème}}$ sous-domaine) à l'itération k. On note également $\theta^k \in]0,1[$ le paramètre de relaxation à l'itération k. Ce paramètre est identique pour tous les sous domaines. On définit la fonction de relaxation $\mathcal{K}_i^k : \gamma_i' \longmapsto \mathbb{R}$ par :

$$\mathcal{K}_i^k = \theta^k u_\delta^k|_{\gamma_i'} + \left(1-\theta^k\right)\mathcal{K}_i^{k-1} \tag{3.20}$$

Ainsi, la formulation variationnelle des problèmes locaux à l'itération k est réécrite à l'aide de l'évaluation de $\mathcal{K}_i^{k-1}$:

Trouver $(\hat{u}_{1,\delta}^k, ..., \hat{u}_{N_{perfo},\delta}^k) \in \left(\otimes_{i=1}^{N_{perfo}} \hat{V}_{i,\delta}\left(\omega_{i,\delta}\right)\right)$ telles que :

$$\left(\text{Local}_i\right)_{i=1}^{N_{perfo}} \left\{ \begin{array}{c} \displaystyle\int_{\omega_{i,\delta}} \nabla\hat{u}_{i,\delta}^k \cdot \nabla\hat{v}_i = \int_{\omega_{i,\delta}} f\ \hat{v}_i, \quad \forall \hat{v}_i \in H_0^1\left(\omega_{i,\delta}\right) \cap P_{\hat{N}}^c\left(\omega_{i,\delta}\right) \\ \hat{u}_{i,\delta}^k = \mathcal{K}_i^{k-1} \ \text{ sur } \gamma_i' \end{array} \right\} \tag{3.21}$$

On peut remarquer dans (3.21) que la contrainte sur γ_i' ne dépend plus d'un terme inconnu, comme c'était le cas dans (3.18) avec u_δ. On peut alors prendre en compte cette contrainte comme une condition aux limites de Dirichlet classique et l'intégrer directement dans l'espace d'approximation. On définit ce nouvel espace :

$$\hat{V}_{i,\delta}^k\left(\omega_{i,\delta}\right) = \left\{ v \in H_{0,\gamma_i}^1\left(\omega_{i,\delta}\right) \cap P_{\hat{N}}^c\left(\omega_{i,\delta}\right), v = \mathcal{K}_i^{k-1} \ \text{ sur } \gamma_i' \right\}$$

Ce qui nous donne finalement les formulations variationnelles locales et globales à l'itération k :

Trouver $(\hat{u}_{1,\delta}^k, ..., \hat{u}_{N_{perfo},\delta}^k) \in \left(\otimes_{i=1}^{N_{perfo}} \hat{V}_{i,\delta}^k\left(\omega_{i,\delta}\right)\right)$ telles que :

$$\left(\text{Local}_i\right)_{i=1}^{N_{perfo}} \left\{ \int_{\omega_{i,\delta}} \nabla\hat{u}_{i,\delta}^k \cdot \nabla\hat{v}_i = \int_{\omega_{i,\delta}} f\ \hat{v}_i, \quad \forall \hat{v}_i \in H_0^1\left(\omega_{i,\delta}\right) \cap P_{\hat{N}}^c\left(\omega_{i,\delta}\right) \right\} \tag{3.22}$$

Trouver $u_\delta^k \in V_\delta\left(\Omega_\delta\right)$ telles que :

$$\text{(Global)} \qquad \left\{ \begin{array}{c} \displaystyle\int_{\Omega_\delta} \nabla u_\delta^k \cdot \nabla v = \int_{\Omega_\delta} f\ v + \sum_{i=1}^{N_{Perfo}} \int_{\gamma_i} \left(\partial_{\nu_i}\hat{u}_{i,\delta}^k - \partial_{\nu_i}\mathring{u}_{i,\delta}\right) v, \\ \forall v \in H_0^1\left(\Omega_\delta\right) \cap P_N^c\left(\Omega_\delta\right) \end{array} \right\} \tag{3.23}$$

Dans [86], les paramètres de relaxation θ^k sont choisis fixes au cours des itérations du point fixe. Nous préférons ici calculer ces paramètres via la méthode d'accélération d'Aitken, par exemple celle décrite dans [99]. En fait, lorsque $N_{perfo} > 1$, nous obtenons plusieurs paramètres θ_i^k pour un k donné, un pour chaque perforation. Nous choisissons de prendre θ^k comme le plus petit paramètre calculé, c'est-à-dire le paramètre le plus « relaxant ». Ainsi on écrit $\theta^k = \min_{i=1,..,N_{perfo}} \theta_i^k$.

Algorithm 2 Algorihtme de point fixe de la FBM sans l'extension $\bar{f}$

Require: ϵ une tolérance et $\{\mathcal{K}_i^0, i = 1, .., N_{perfo}\}$
  Résolution des problèmes sur les perforations B_i (3.16)
  $k = 1$
  while Non convergence **do**
    Résolution des problèmes locaux (3.22)
    Résolution du problème global (3.23)
    Calcul convergence
    if Non convergence **then**
      Calcul du paramètre de relaxation θ^k
      Mise à jour des $\mathcal{K}_i^k$
      k=k+1
    end if
  end while

Nous présentons avec l'algorithme 2.3 l'algorithme complet de la FBM où chacune des étapes a été décrite dans les paragraphes précédents hormis le critère de convergence. Nous utilisons pour cela le maximum des différences relatives entre deux itérations de la solution globale sur chaque γ_i', c'est-à-dire :

$$\max_{i=1,..,N_{perfo}} \frac{\| u_\delta^k|_{\gamma_i'} - \mathcal{K}_i^{k-1}\|_{L^2(\gamma_i')}}{\|\mathcal{K}_i^{k-1}\|_{L^2(\gamma_i')}} \leq \epsilon \tag{3.24}$$

avec ϵ un réel correspondant à une tolérance donnée.

Avant de passer à une vérification numérique à l'aide de nombreux tests de convergence, nous désirons tout d'abord montrer le comportement de cet algorithme avec quelques applications relativement simples. Avec la figure 3.3, nous avons voulu montrer l'efficacité du choix d'utiliser l'accélération d'Aitken pour la relaxation. Le problème qui est résolu est défini sur le carré $(-1, 1) \times (-1, 1)$ perforé par un disque centré en 0 et de rayon 0.1. Les données sont $f = 1$ et $g = 0$. Nous avons utilisé une approximation éléments finis P_2 dans le domaine global et local et pris un critère de convergence $\epsilon = 10^{-5}$. Nous avons ensuite appliqué la méthode FBM avec un domaine local ω représentant un anneau et cela pour différentes largeurs $|\omega|$. Pour θ fixé, on remarque que le nombre d'itérations est le plus faible pour les θ les plus proche de 0.5 et pour des largeurs $|\omega|$ assez grandes. Quand on raffine ces largeurs, les θ les plus rapides à converger se mettent à diverger, et plus θ est proche de 0.5 et plus l'algorithme diverge rapidement. Pour $\theta > 0.5$, le comportement de l'algorithme se détériore. Grâce à la relaxation dynamique d'Aitken, nous obtenons les plus faibles nombres d'itérations pour toute la gamme de largeur $|\omega|$. On remarque cependant une légère augmentation lorsque $|\omega|$ devient très petit.

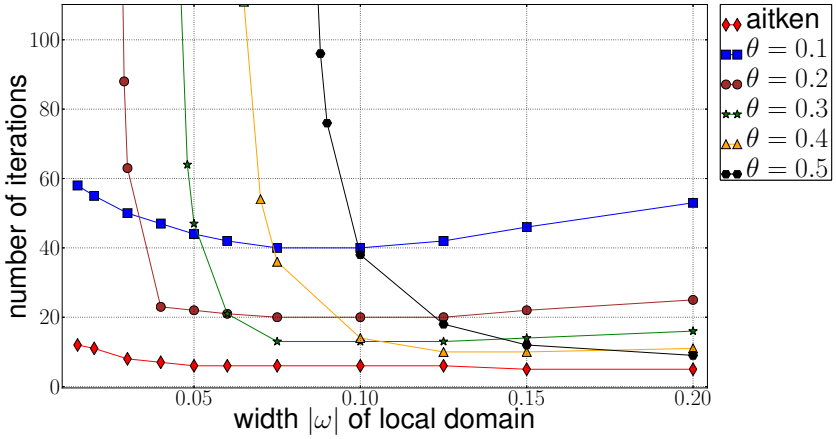


FIGURE 3.3 – Comparaison du nombre d'itérations du point fixe en fonction de la largeur $|\omega|$ du domaine local. Nous avons utilisé plusieurs paramètres de relaxation fixes et la relaxation dynamique d'Aitken.

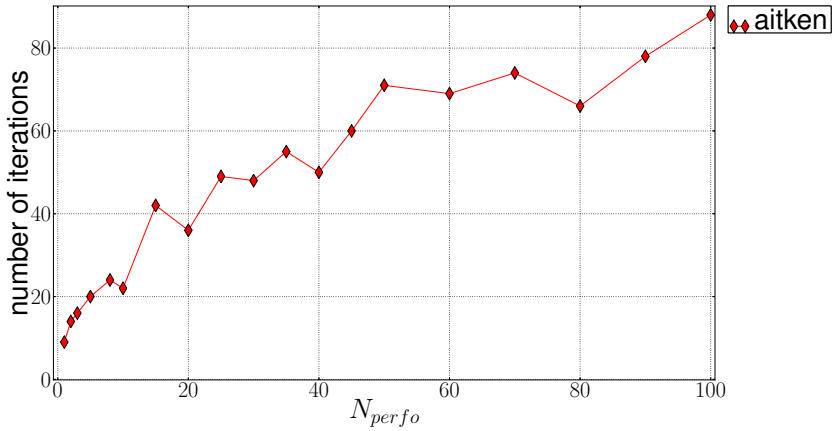
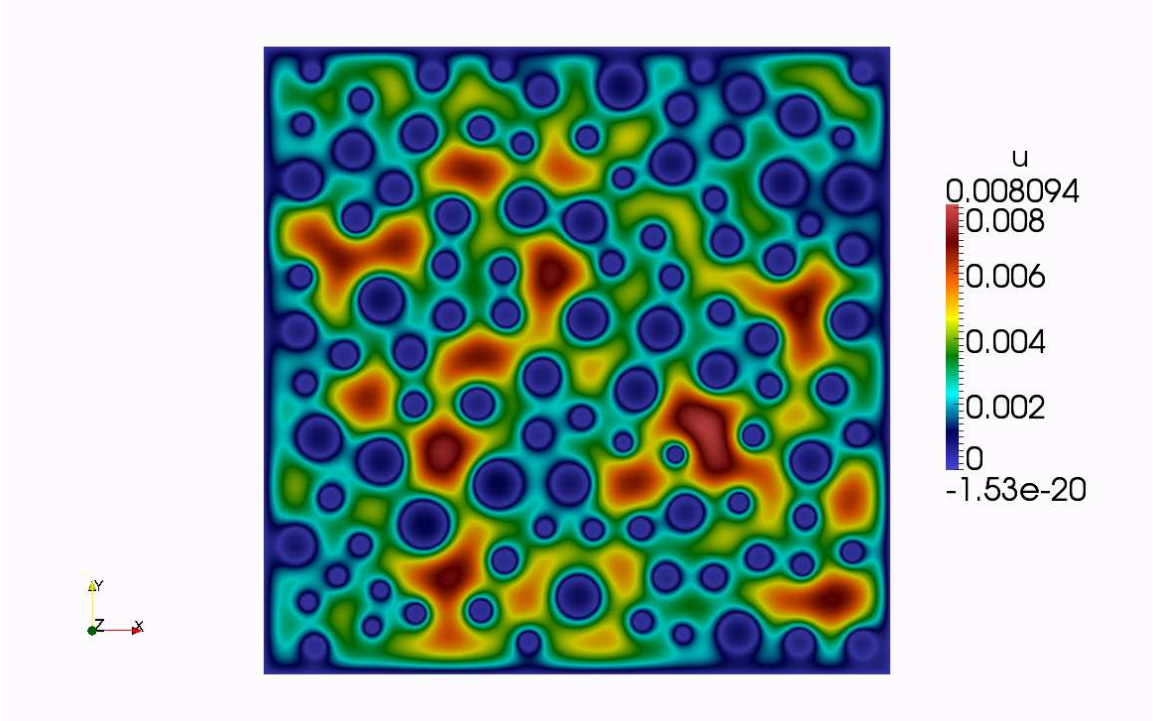


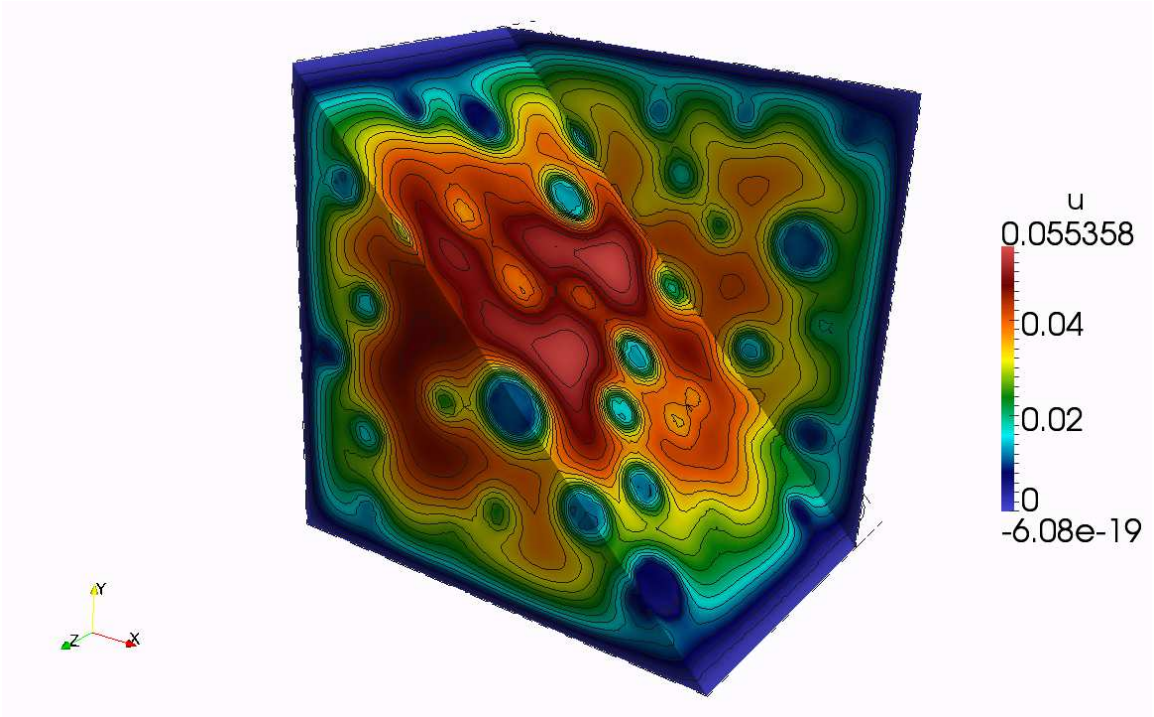
FIGURE 3.4 – Comparaison du nombre d'itération du point fixe en fonction du nombre de perforation N_{perfo}.

Sur la figure 3.4, nous avons tracé le nombre d'itérations de l'algorithme de point fixe en fonction du nombre de perforations pour la méthode FBM avec la relaxation d'Aitken. Pour cette étude nous avons utilisé la même configuration précédente, sauf que l'on fait varier le nombre de perforations. Ces perforations sont des disques de rayon 0.03 et leur placement est fait aléatoirement dans Ω. On voit que le nombre d'itérations croît avec les nombres de perforations N_{perfo}. Cependant, ce nombre d'itérations « n'explose pas », il reste raisonnable au vu du grand nombre de perforations.

Enfin, avec les figures 3.5(a) et 3.5(a), nous avons le résultat de deux applications avec 100 perforations de tailles variables. La première est un test 2D sur un carré avec des disques comme perforations. La deuxième est une application 3D sur un cube perforé par des boules sphériques. Pour chacun de ces exemples, nous avons utilisé $f = 1$ et $g = 0$.



(a) Problème 2D avec 100 perforations



(b) Problème 3D avec 100 perforations

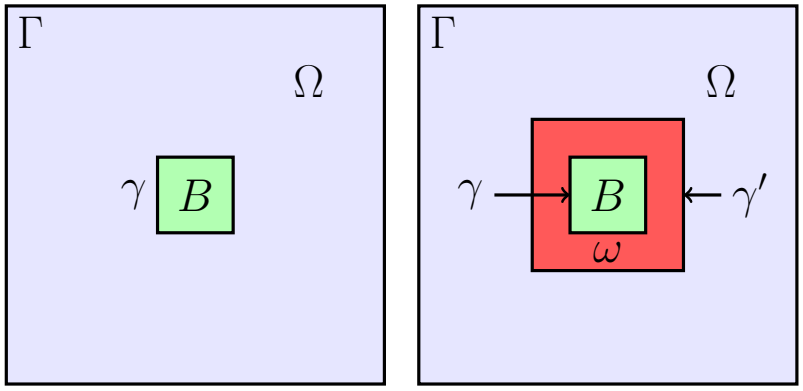
FIGURE 3.5 – Exemples de solutions obtenues avec la FBM.

2.4 Tests numériques

Nous présentons maintenant plusieurs résultats numériques obtenus avec la formulation I. Ce sont des résultats de convergence en h pour différents ordres polynomiaux d'approximation. Nous montrerons aussi le cas où l'on utilise des discrétisations géométriques d'ordre > 1. Le principal but de ces tests est d'étudier l'optimalité de la méthode grâce aux estimations d'erreur à priori que l'on peut trouver dans [26]. Nous voulons aussi vérifier l'implémentation de la méthode pour une large gamme d'approximations en espace et géométrie. Pour tous ces tests, nous avons pris le critère d'arrêt du point fixe $\epsilon = 10^{-12}$.

Cas test 1 :

Pour ce premier test, nous voulons comparer le comportement de la FBM avec et sans l'extension $\bar{f}$. Le but est de démontrer que la formulation avec l'extension $\bar{f}$ est équivalente à la formulation avec le terme source initial f et l'ajout du sous-problème sur la perforation B. Commençons par présenter le problème de départ. Le problème perforé est défini par un carré perforé par un autre carré B, figure 3.6(a).



(a) Géométrie du problème initial (b) Géométries utilisés avec la FBM

FIGURE 3.6 – Géométries utilisées pour le cas test 1.

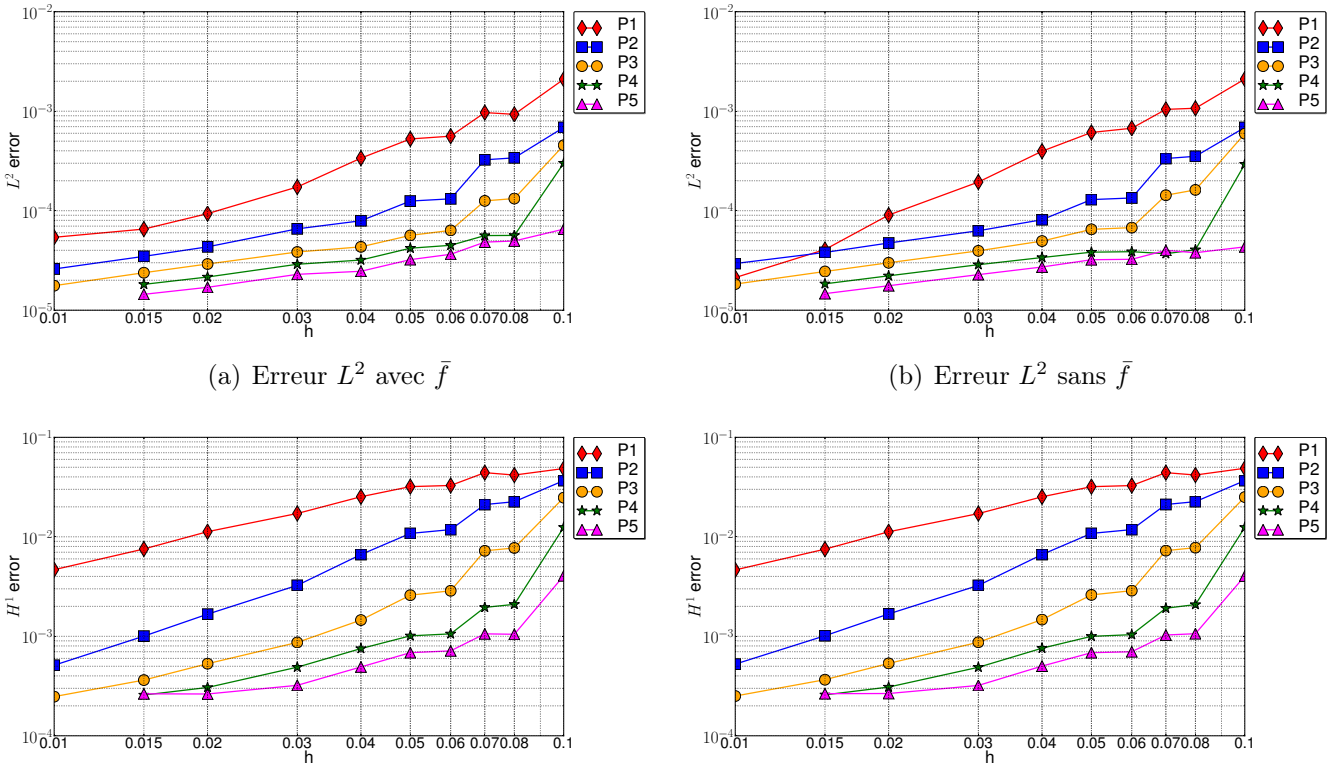
L'équation à résoudre pour ce problème est l'équation de Poisson avec des conditions de Dirichlet homogènes sur l'ensemble de la frontière. On cherche la fonction $\tilde{u} \in H^1\left(\Omega \backslash \bar{B}\right)$ telle que :

$$\begin{cases} -\Delta \tilde{u} = f & \text{dans } \Omega \backslash \bar{B} \\ \tilde{u} = 0 & \text{sur } \Gamma \\ \tilde{u} = 0 & \text{sur } \gamma \end{cases} \tag{3.25}$$

avec le terme source défini par $f = 10 \cos(10\pi x) \cos(10\pi y)$. La solution du problème (3.25) n'étant pas connue analytiquement, nous prenons comme solution de référence une solution numérique u_{num} calculée sur un maillage très fin. Nous notons aussi u_δ l'approximation numérique obtenue à l'aide de la méthode FBM.

Pour effectuer cette comparaison, nous choisissons de mesurer les erreurs $\|u_\delta - u_{num}\|$ en norme L^2 et H^1. Ces erreurs sont calculées uniquement sur la couronne ω car il est relativement couteux d'effectuer cette comparaison. En effet, comme les approximations

u_δ et u_{num} ne vivent pas sur le même maillage, il est nécessaire d'utiliser un opérateur d'interpolation entre ces deux fonctions pour pouvoir évaluer cette norme. Nous avons choisi d'interpoler l'approximation FBM sur le maillage fin. De plus, les deux problèmes étant fortement liés, la convergence de l'un entraine la convergence de l'autre. Les figures 3.7(a) et 3.7(c) représentent l'erreur avec l'extension $\bar{f}$. Les deux autres figures 3.7(b) et 3.7(d) n'ont pas utilisé $\bar{f}$ mais le sous-problème sur la perforation B. Les graphes sont tracés en échelle log-log en fonction du pas de maillage h.



(a) Erreur L^2 avec $\bar{f}$

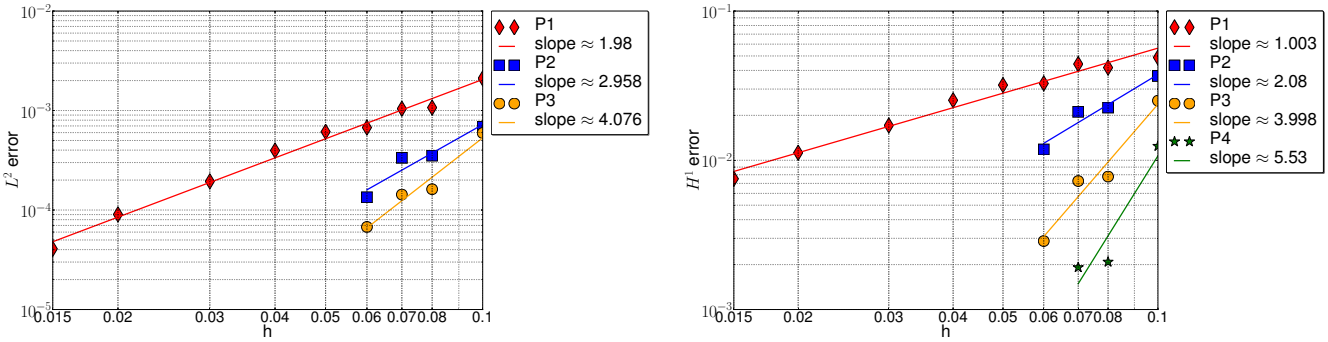
(b) Erreur L^2 sans $\bar{f}$

(c) Erreur H^1 avec $\bar{f}$

(d) Erreur H^1 sans $\bar{f}$

FIGURE 3.7 – Erreurs de convergence $\|u_\delta - u_{num}\|_{L^2}$ et $\|u_\delta - u_{num}\|_{H^1}$ sur ω (cas test 1).

Avec les figures 3.7, on observe bien que les deux formulations donnent la même précision. Le comportement et la valeur des courbes sont très similaires aussi bien en norme L^2 qu'en norme H^1. Cependant on remarque également que les erreurs de convergence semblent atteindre un seuil maximal. Cette saturation, qui reste toutefois relativement basse, ne peut-être expliquée clairement. Plusieurs choses peuvent être remises en cause. Tout d'abord, l'approximation u_{num} n'est peut être pas assez précise. L'intégration non conforme n'est pas totalement validée. Cela peut aussi venir du fait que la perforation n'est pas de forme régulière (ici à cause des quatre angles). Par contre, le fait de monter en ordre polynomial nous assure une approximation très précise dès les premiers pas de maillage. Pour terminer ce premier cas test, nous avons enlevé les points qui arrivaient à saturation pour le cas sans extension $\bar{f}$. C'est ce qui est tracé sur les figures 3.8(a) et 3.8(b) avec les droites de convergence et leur pente en légende. On retrouve alors les approximations théoriques optimales pour les deux normes.



(a) Erreur L^2 sans $\bar{f}$ (b) Erreur H^1 sans $\bar{f}$

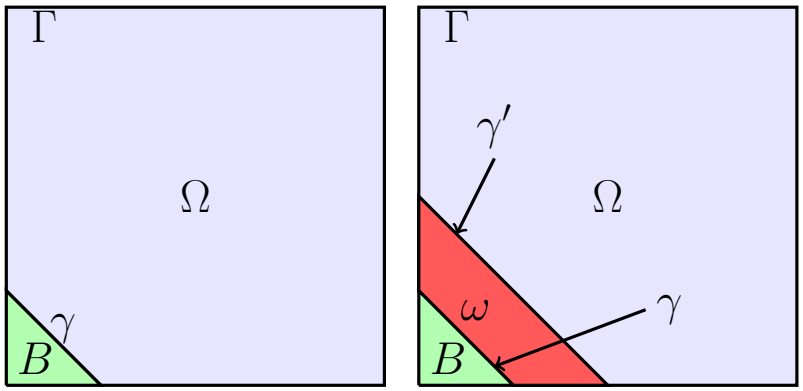
FIGURE 3.8 – Erreurs de convergence $\|u_\delta - u_{num}\|_{L^2}$ et $\|u_\delta - u_{num}\|_{H^1}$ sur ω pour le cas sans $\bar{f}$ (cas test 1).

Cas test 2 :

Ce deuxième test a été réalisé dans [26] et montre l'optimalité de la méthode FBM jusqu'à l'ordre 4 en norme H^1. Les figures 3.9(a) et 3.9(b) illustrent les géométries utilisées. Nous considérons pour le domaine global Ω le carré $]0,1[^2$. La frontière γ et l'interface artificielle γ' sont donnée par :

$$\gamma = \left\{(x,y) \in [0,1]^2,\ x+y = \frac{1}{4}\right\} \quad et \quad \gamma' = \left\{(x,y) \in [0,1]^2,\ x+y = \frac{1}{2}\right\}$$

La perforation B correspond alors au triangle défini par le coin inférieur gauche du carré Ω et par le segment γ. Par conséquence, le domaine local ω est le quadrangle défini par les deux segments γ et γ'. On constate que dans ce cas $\partial\Omega \cap \partial\omega \neq \emptyset$, mais cela ne pose aucun problème théorique et pratique pour l'application de la méthode FBM. Il suffit de rajouter dans le problème local ω la même condition aux limites que dans le problème global Ω sur $\partial\Omega \cap \partial\omega$.



(a) Géometrie du problème initial (b) Géométries utilisées avec la FBM

FIGURE 3.9 – Géométries utilisées pour le cas test 2.

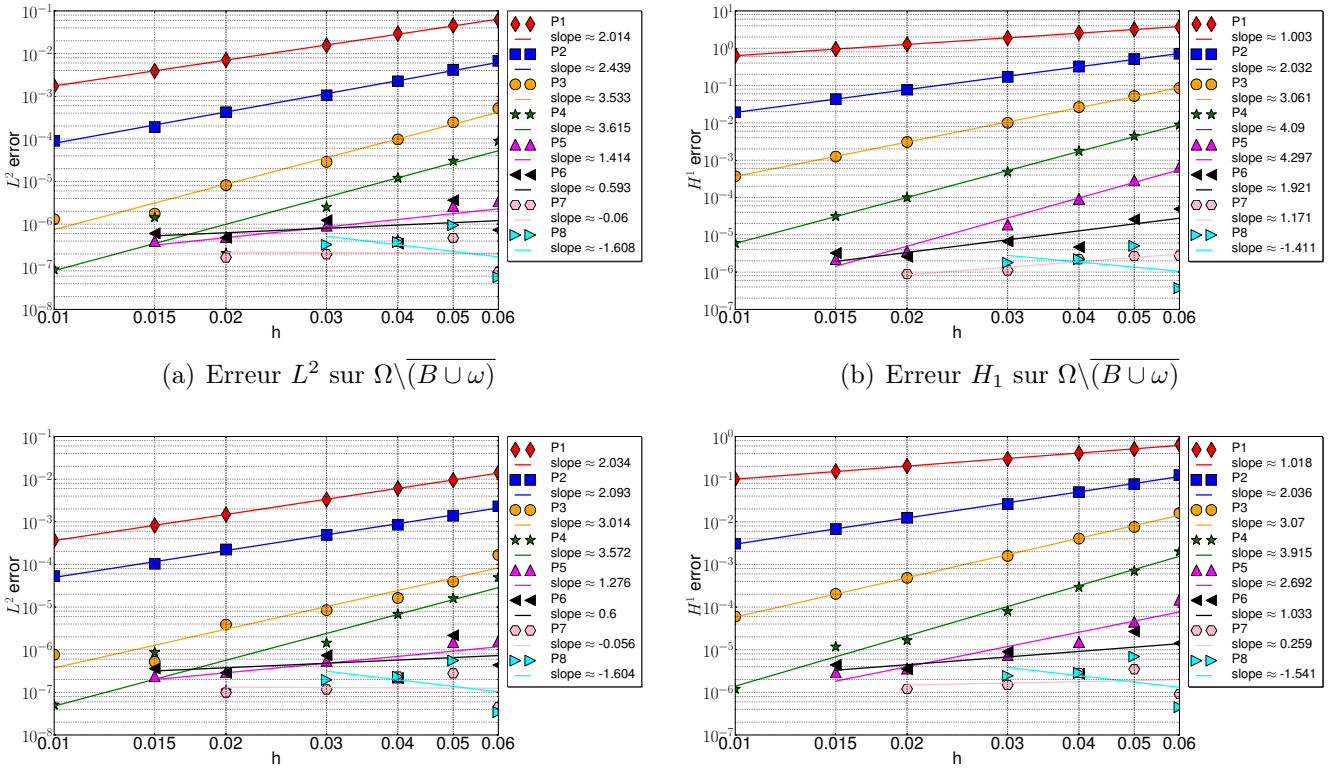
Nous cherchons alors la solution du problème de Poisson suivant :

$$\begin{cases} -\Delta\tilde{u} = f & \text{dans } \Omega\backslash\bar{B} \\ \tilde{u} = g & \text{sur } \partial\left(\Omega\backslash\bar{B}\right) \end{cases} \tag{3.26}$$

Pour ce test, nous avons choisi le terme source f à partir de la fonction g qui est supposé être la solution exacte du problème et défini par :

$$g\left(x,y\right) = (x+y-1)(x+y)\cos(6\pi(x+y)) + 3\pi(x+y)(x+y) - 3\frac{\pi}{16} \tag{3.27}$$

L'intérêt de cette solution g est que la fonction $g = 0$ sur le segment γ, ce qui nous place dans le cas FBM homogène sur la particule.



(a) Erreur L^2 sur $\Omega\backslash\overline{(B\cup\omega)}$

(b) Erreur H_1 sur $\Omega\backslash\overline{(B\cup\omega)}$

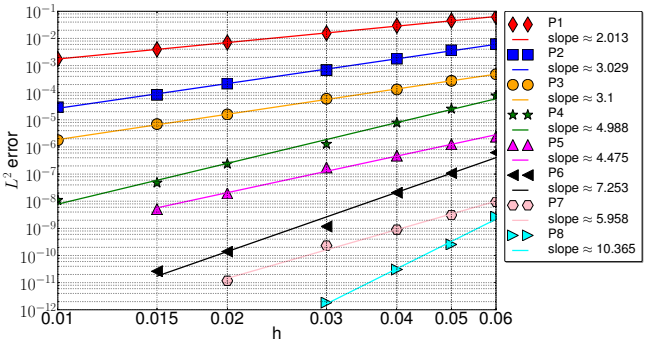
(c) Erreur L^2 sur ω

(d) Erreur H_1 sur ω

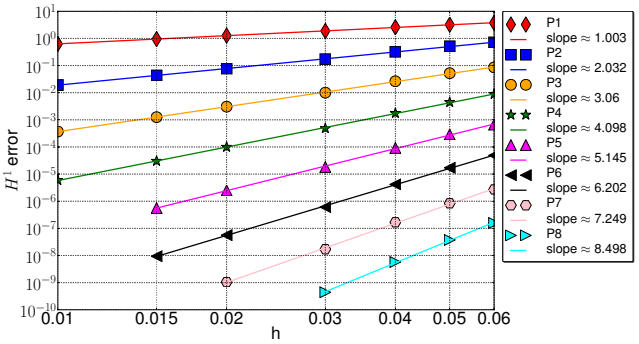
FIGURE 3.10 – Erreurs de convergence de la FBM $\|u_\delta - g\|_{L^2}$ et $\|u_\delta - g\|_{H^1}$ pour le cas avec l'extension $\bar{f}$ (cas test 2).

Nous présentons avec les figures 3.10(a), 3.10(b), 3.10(c) et 3.10(d) le résultat des tests de convergence de la FBM avec l'extension $\bar{f}$. Ces tests sont effectués pour différents ordres polynomiaux, de 1 à 8, qui sont identiques à chaque fois dans le domaine global et local. On remarque tout d'abord que nous retrouvons les résultats d'optimalité présentés dans [26] jusqu'à l'ordre 4 en norme H^1. L'optimalité en norme L^2 est aussi présente jusqu'à l'ordre 4. Cependant, le taux de convergence pour les approximations P_2 et P_4 est légèrement plus faible que le taux optimal espéré. Par contre, pour les ordres supérieurs à

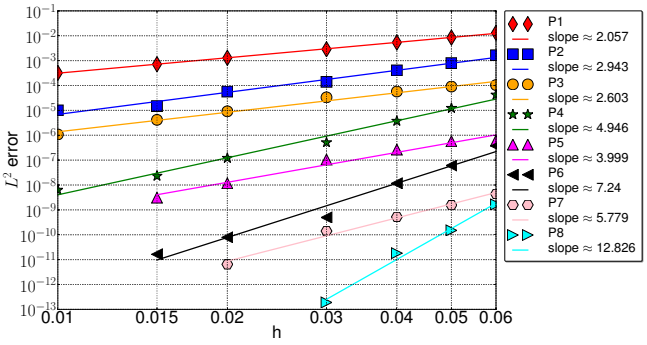
4, les erreurs L^2 et H^1 ne semblent plus converger, comme si elles atteignaient leur limite dès les pas de maillage les plus « grossiers ». Ce phénomène s'explique peut-être par des erreurs commises dans les intégrales non conformes, mais la question reste toujours ouverte. Nous pouvons aussi constater avec ces graphiques le fait très naturel que le comportement de la convergence locale et globale est identique.



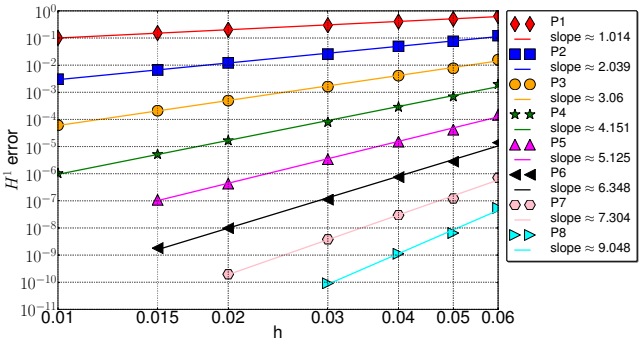
(a) Erreur L^2 sur $\Omega\backslash\overline{(B \cup \omega)}$



(b) Erreur H_1 sur $\Omega\backslash\overline{(B \cup \omega)}$



(c) Erreur L^2 sur ω



(d) Erreur H_1 sur ω

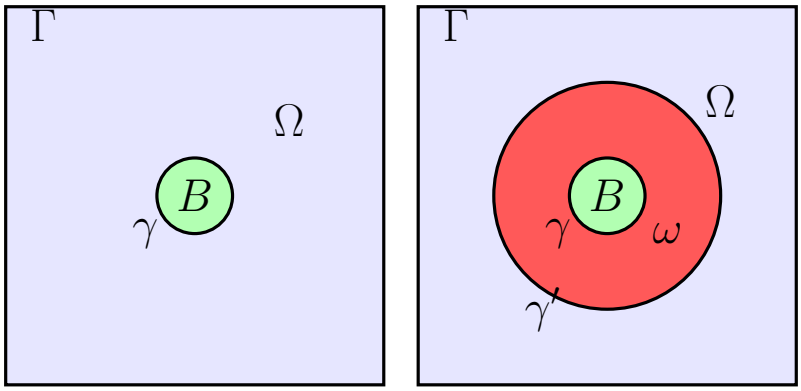
FIGURE 3.11 – Erreurs de convergence de la FBM $\|u_\delta - g\|_{L^2}$ et $\|u_\delta - g\|_{H^1}$ pour le cas sans l'extension $\bar{f}$ (cas test 2).

Maintenant, nous effectuons les mêmes tests mais avec l'ajout du sous-problème sur la perforation et en utilisant le terme source f sur l'ensemble de Ω. Il est important de noter que ce test est moins révélateur de l'optimalité de la méthode, car le fait d'utiliser f sur tout Ω pousse fortement à faire converger le problème global vers la solution exacte. Cependant ce test nous permet de contrôler le comportement des intégrales non conforme dans la méthode FBM et de vérifier le couplage global/local. Les figures 3.11(a), 3.11(b), 3.11(c) et 3.11(d) illustrent la convergence de la FBM dans ce cas particulier. Nous constatons immédiatement que cette fois-ci la convergence est établie pour tous les ordres polynomiaux. L'ordre optimal est atteint parfaitement en norme H^1. En norme L^2, les ordres de convergence P_3, P_5 et P_7 sont légérement inférieurs à l'ordre optimal. Nous avons encore peut-être des résidus d'erreur dans les non-conformités. Enfin, on remarque aussi que la converge P_8 est étrangement plus grande que la convergence optimale. Mais pour cette approximation, les valeurs de l'erreur sont de l'ordre de 10^{-12}, donc les erreurs d'arrondi numérique sont plutôt envisagées pour expliquer ce phénomène.

Cas test 3 :

Un des grands intérêts de la méthode FBM est de pouvoir utiliser des approximations différentes entre domaine global et domaine local, et ainsi d'avoir une plus grande précision proche de la particule. C'est pourquoi dans ce dernier test, nous voulons contrôler le comportement de l'ordre élevé en géométrie dans le domaine local en restant affine dans le domaine global. Pour cela, nous utilisons une géométrie de la perforation qui n'est plus affine. La Figure 3.12(a) représente le domaine de calcul initial, où la perforation B est représentée par un disque centré en $(0.5, 0.5)$ de rayon $R_B = 0.1$. Le bord γ est donc représenté par la frontière de ce disque. Le domaine global est encore le carré $]0,1[^2$. Comme le montre ensuite la figure 3.12(b), la géométrie du domaine local est un anneau, de rayon extérieur $R_\omega = 0.3$. La frontière fictive γ' représente le cercle décrit par l'équation :

$$\gamma' = \left\{(x,y) \in \mathbb{R}^2,\ (x-0.5)^2 + (y-0.5)^2 = R_\omega^2\right\}$$



(a) Géometrie initial (b) Géometrie FBM

FIGURE 3.12 – Géométries utilisées pour le cas test 3.

Le problème que l'on souhaite résoudre est de trouver la fonction $\tilde{u} \in H^1\left(\Omega \backslash \bar{B}\right)$ vérifiant l'équation aux dérivées partielles suivante :

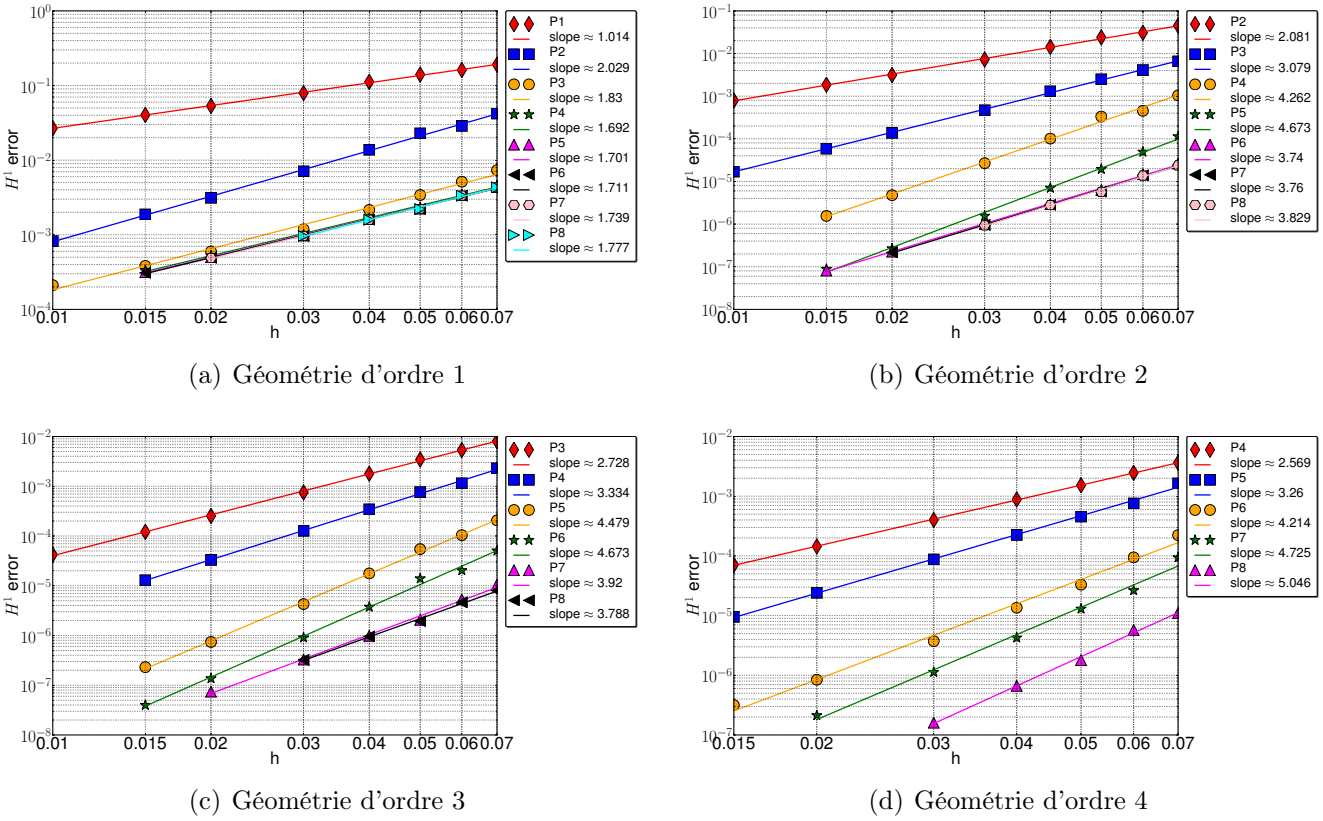
$$\begin{cases} -\Delta\tilde{u} = f & \text{dans } \Omega\backslash\bar{B} \\ \tilde{u} = g & \text{sur } \Gamma \\ \tilde{u} = 0 & \text{sur } \gamma \end{cases} \tag{3.28}$$

De la même manière que le test précédent, nous allons nous donner le terme source f à l'aide de la fonction g donnée par l'expression suivante en supposant que g est la solution du problème (3.28) :

$$g\,(x,y) = \left((x-0.5)^2 + (y-0.5)^2 - R_B^2\right)\cos(8\pi x) \tag{3.29}$$

La solution g vérifie bien-sûr la condition $g = 0$ sur γ au niveau continu. Cependant au niveau discret, lorsque nous aurons maillé le domaine ω, g ne vérifiera plus exactement cette condition sur γ_δ, la discrétisation de γ. Si l'on considère une discrétisation élément

fini P_N et un maillage d'ordre 1, les degrés de liberté se trouvant sur γ_δ ne se situent pas exactement sur le cercle γ hormis les degrés de liberté se trouvant sur les noeuds du maillage. Ainsi la solution discrète calculée u_δ vaudra 0 en ces points, ce qui n'est pas le cas pour la solution exacte g du problème. Cependant dans le cas isoparamétrique, ce problème ne devrait pas apparaitre, car les degrés de liberté se trouveront exactement sur les noeuds géométriques situés exactement sur γ.



(a) Géométrie d'ordre 1

(b) Géométrie d'ordre 2

(c) Géométrie d'ordre 3

(d) Géométrie d'ordre 4

FIGURE 3.13 – Erreurs de convergence de la FBM $\|u_\delta - g\|_{H^1}$ sur ω pour le cas sans l'extension $\bar{f}$ (cas test 3).

Des tests de convergence ont été effectués pour différents ordres géométriques, de 1 à 4, comme le montrent les figures 3.13(a), 3.13(b), 3.13(c) et 3.13(d). Ces tests ont été effectués sans l'extension $\bar{f}$ avec des approximations isoparamétriques et subparamétriques. On observe alors que des problèmes de convergence apparaissent. Par exemple pour la discrétisation géométrique d'ordre 1, la convergence optimale est atteinte jusqu'à l'ordre 2, ensuite le fait de monter en ordre ne nous apporte plus rien. Pour la discrétisation géométrique d'ordre 2, nous atteignons la convergence optimale jusqu'à l'ordre 4. L'intérêt d'approcher au mieux la géométrie nous permet donc d'assurer une meilleure convergence pour les ordres d'approximation spatiale élevés.

3 Formulation II : point-selle

Dans cette partie, nous allons maintenant présenter une nouvelle formulation de la FBM, dite formulation point-selle, qui a été récemment développée en collaboration avec Silvia Bertoluzza. Cette formulation a été initialement motivée pour résoudre les équations de Stokes dans un domaine perforé, mais son principe s'applique aussi aux problèmes elliptiques. Le principe de la FBM reste toujours le même, dans le sens où l'on va avoir un problème global et plusieurs sous-problèmes locaux, mais c'est le couplage entre les équations qui va changer. Nous allons cette fois-ci utiliser des multiplicateurs de Lagrange pour assurer la continuité faible entre la solution globale et les solutions locales sur les interfaces fictives γ_i' et aussi pour imposer la condition de Dirichlet sur les frontières γ dans les problèmes locaux. De plus, pour le cas de Stokes, une contrainte supplémentaire sur la pression devra être ajoutée. En raison du récent développement de cette méthode et de sa complexité en matière d'analyse théorique, les démonstrations des théorèmes d'existence, d'unicité, de stabilité et d'estimation d'erreur à priori ne seront pas présentées dans cette thèse. Une description complète de cette nouvelle formulation est en cours de réalisation avec les prochaines publications [23, 24, 22] qui apporteront en plus les preuves théoriques.

Nous allons commencer par présenter la résolution du problème de Poisson avec cette nouvelle formulation FBM, illustrée avec quelques résultats numériques. Nous passerons ensuite dans une étude plus approfondie sur le comportement numérique des équations de Stokes et Navier-Stokes.

3.1 Problème de Poisson

Nous souhaitons résoudre le problème elliptique présenté en (3.3). Pour simplifier les notations, nous supposerons ici n'avoir qu'une seule perforation, donc $N_{perfo} = 1$, mais l'extension à un nombre arbitraire de sous domaines est naturel. Comme évoqué dans le début de cette partie, nous allons assurer certaines contraintes à l'aide de multiplicateurs de Lagrange. On note λ le multiplicateur assurant la condition de Dirichlet sur le bord γ et ϕ le multiplicateur assurant la contrainte de continuité faible des solutions globale et locale sur l'interface fictive γ'. La formulation point-selle de la FBM se traduit par le problème suivant :

Trouver $(u, \hat{u}, \lambda, \phi) \in H_g^1(\Omega) \times H^1(\omega) \times H^{-1/2}(\gamma) \times H^{-1/2}(\gamma')$ telles que :

$$\int_\Omega \nabla u \cdot \nabla v - \int_\gamma \lambda\, v = \int_\Omega f\, v, \quad \forall v \in H_0^1(\Omega) \tag{3.30}$$

$$\int_\omega \nabla \hat{u} \cdot \nabla \hat{v} - \int_\gamma \lambda\, \hat{v} - \int_{\gamma'} \phi\, \hat{v} = \int_\omega f\, \hat{v}, \quad \forall \hat{v} \in H^1(\omega) \tag{3.31}$$

$$\int_\gamma \hat{u}\, \mu = 0, \quad \forall \mu \in H^{-1/2}(\gamma) \tag{3.32}$$

$$\int_{\gamma'} (u - \hat{u})\, \psi = 0, \quad \forall \psi \in H^{-1/2}(\gamma') \tag{3.33}$$

Théorème 1. *Le problème* (3.30)-(3.33) *admet une unique solution* $(u, \hat{u}, \lambda, \phi)$.

Ensuite, nous choisissons des espaces d'approximations associés respectivement à la solution globale, locale et aux multiplicateurs de Lagrange sur γ et γ' :

$$\begin{aligned} V_\delta &= \left\{v \in H_g^1(\Omega_\delta) \cap P_N^c(\Omega_\delta)\right\} \\ \hat{V}_\delta &= \left\{v \in P_{\hat{N}}^c(\omega_\delta)\right\} \\ L_\delta^\gamma &= \left\{v \in P_Q^c(\gamma_\delta)\right\} \\ L_\delta^{\gamma'} &= \left\{v \in P_{Q'}^c(\gamma'_\delta)\right\} \end{aligned}$$

avec N,$\hat{N}$,Q et Q' leur degré d'approximation respectif. Ainsi nous écrivons la formulation variationnelle discrète :

Trouver $(u_\delta, \hat{u}_\delta, \lambda_\delta, \phi_\delta) \in \left(V_\delta \times \hat{V}_\delta \times L_\delta^\gamma \times L_\delta^{\gamma'}\right)$ telles que :

$$\int_{\Omega_\delta} \nabla u_\delta \cdot \nabla v - \int_{\gamma_\delta} \lambda_\delta\, v = \int_{\Omega_\delta} f\, v, \quad \forall v \in H_0^1(\Omega_\delta) \cap P_N^c(\Omega_\delta) \tag{3.34}$$

$$\int_{\omega_\delta} \nabla \hat{u}_\delta \cdot \nabla \hat{v} - \int_{\gamma_\delta} \lambda_\delta\, \hat{v} - \int_{\gamma'_\delta} \phi_\delta\, \hat{v} = \int_{\omega_\delta} f\, \hat{v}, \quad \forall \hat{v} \in \hat{V}_\delta \tag{3.35}$$

$$\int_{\gamma_\delta} \hat{u}_\delta\, \mu = 0, \quad \forall \mu \in L_\delta^\gamma \tag{3.36}$$

$$\int_{\gamma'} (u_\delta - \hat{u}_\delta)\, \psi = 0, \quad \forall \psi \in L_\delta^{\gamma'} \tag{3.37}$$

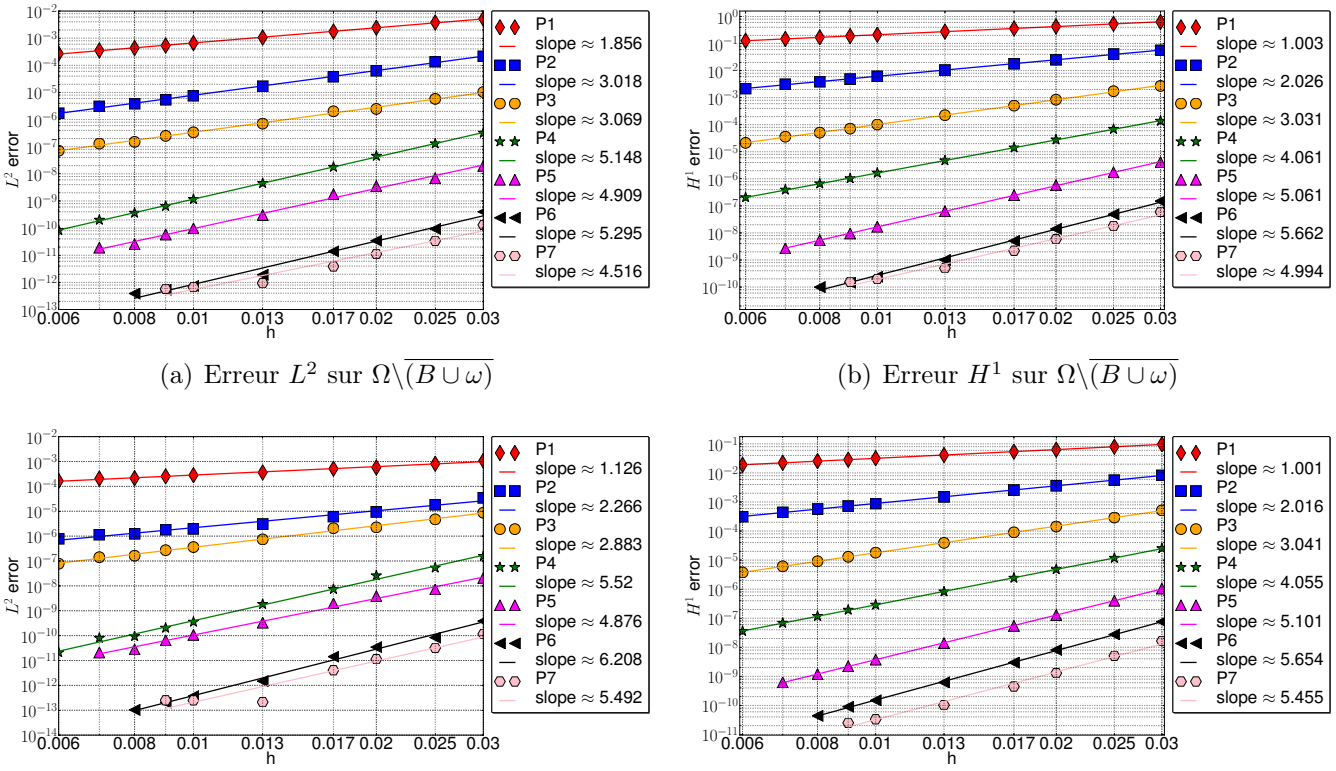
Le système d'équations (3.34)-(3.37) est ensuite résolu par une approche monolithique, c'est-à-dire que toutes ces équations sont résolues en même temps. D'un point de vue algébrique, cela nécessite la construction de la matrice A et du vecteur F associé à ces équations. On présente ci-dessous la structure de ces objets :

$$\underbrace{\left(\begin{array}{c|c|c|c} A_{uu} & 0 & A_{u\lambda} & 0 \\ \hline 0 & A_{\hat{u}\hat{u}} & A_{\hat{u}\lambda} & A_{\hat{u}\phi} \\ \hline 0 & A_{\lambda\hat{u}} & 0 & 0 \\ \hline A_{\phi u} & A_{\phi\hat{u}} & 0 & 0 \end{array}\right)}_{A} \underbrace{\left(\begin{array}{c} u \\ \hline \hat{u} \\ \hline \lambda \\ \hline \phi \end{array}\right)}_{U} = \underbrace{\left(\begin{array}{c} F_u \\ \hline F_{\hat{u}} \\ \hline 0 \\ \hline 0 \end{array}\right)}_{F} \tag{3.38}$$

Pour terminer cette partie sur la formulation point-selle du problème de Poisson, nous présentons les résultats de convergence que nous avons mesurés. Pour cela, nous reprenons le cas test 3 de la partie précédente avec les géométries représentées par les figures 3.12(a)-3.12(b) et la solution de référence g défini par l'expression (3.29). Nous commençons par considérer l'équation aux dérivées partielles suivante qui consiste à trouver la fonction $\tilde{u}$ vérifiant :

$$\begin{cases} -\Delta\tilde{u} = f & \text{dans } \Omega\backslash\bar{B} \\ \tilde{u} = g & \text{sur } \Gamma \\ \tilde{u} = g & \text{sur } \gamma \end{cases} \tag{3.39}$$

Ce problème correspond à une condition de Dirichlet homogène sur γ dans le cas continu à cause de la définition de g. Cependant, au niveau discret, la géométrie approchée



(a) Erreur L^2 sur $\Omega\backslash\overline{(B \cup \omega)}$

(b) Erreur H^1 sur $\Omega\backslash\overline{(B \cup \omega)}$

(c) Erreur L^2 sur ω

(d) Erreur H^1 sur ω

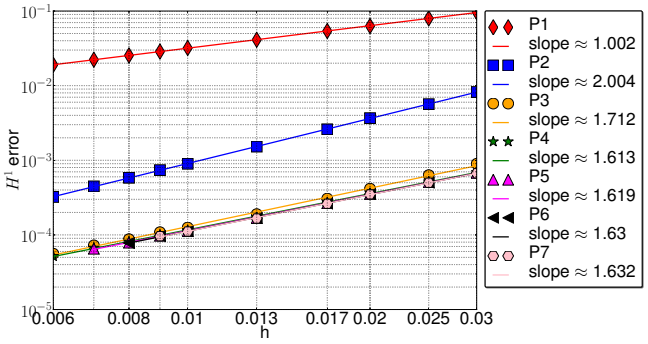
FIGURE 3.14 – Test de convergence de la FBM pour le problème de poisson avec $N = \hat{N} = Q = Q'$ et une géométrie d'ordre 1.

ne permet pas d'obtenir cette propriété, sauf dans le cas isoparamétrique. Pour prendre en compte cette condition non homogène, nous utilisons la même technique proposée à la section 2.1 de la partie précédente. Nous devons donc avant résoudre un problème supplémentaire sur la perforation, puis rajouter cette contribution dans le problème global.

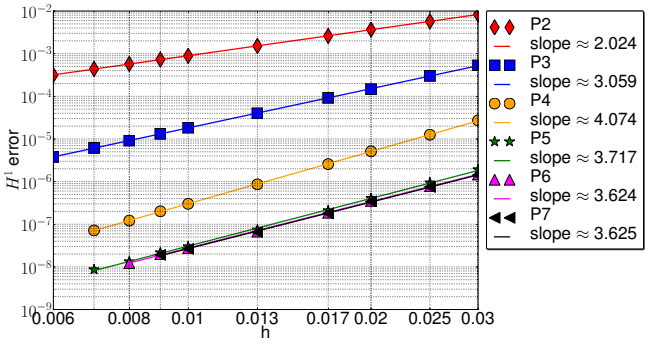
Tout d'abord, nous avons les figures 3.14(b)-3.14(d) qui représente l'erreur H^1 sur le domaine global et local. Les taux de convergence obtenus sont optimums jusqu'à l'ordre 6. Pour l'ordre 7, nous voyons que la convergence n'est plus optimale. Cet effet est peut-être dû aux schémas de quadrature utilisés pour les intégrales non conformes dont l'ordre a été fixé à 12. Pour la norme L^2, figures 3.14(a)-3.14(c), l'optimalité de la méthode n'est plus parfaitement visible. Certains ordres d'approximation ont une pente inférieure à la pente théorique optimale. Toutefois, le comportement de l'erreur reste intéressant avec notamment la présence de la convergence en p.

Nous avons ensuite réalisé un deuxième test similaire avec cette fois-ci des conditions homogènes sur γ. Nous reprenons le problème (3.39) mais en utilisant la condition aux limites $\tilde{u} = 0$ sur γ. Comme nous l'évoquions précédemment, cette condition sur γ ne sera pas forcément vérifiée par la solution exacte g à cause de l'approximation géométrique de cette frontière. Nous voulons ici montrer l'intérêt de l'ordre géométrique dans le but de mieux approcher la frontière γ continue et donc la solution g.

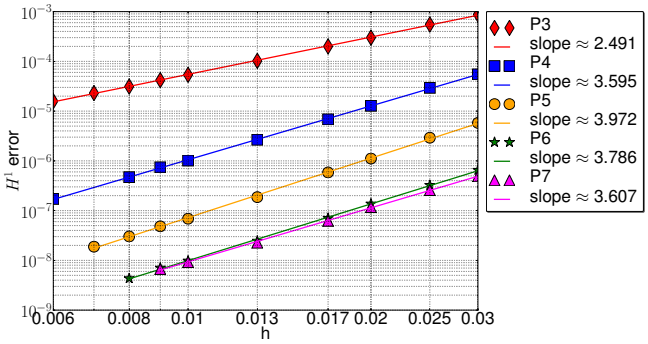
Nous avons effectué plusieurs tests de convergence sur quatre types de discrétisation



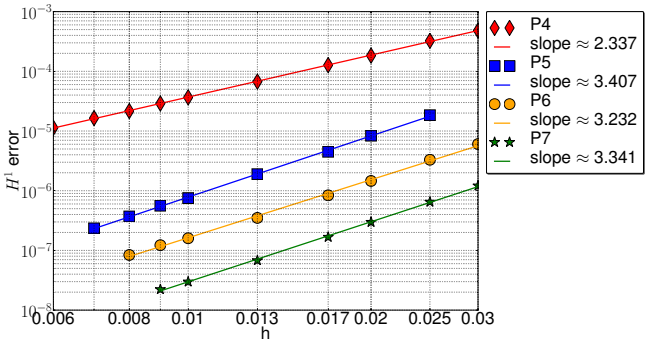
(a) Géométrie d'ordre 1



(b) Géométrie d'ordre 2



(c) Géométrie d'ordre 3



(d) Géométrie d'ordre 4

FIGURE 3.15 – Test de convergence de la FBM pour la norme H^1 sur ω et avec $N = \hat{N} = Q = Q'$.

géométrique de la perforation, de l'ordre 1 à 4. Nous illustrons nos résultats avec la figure 3.15 qui montre l'erreur H^1 sur le domaine local ω. Nous pouvons remarquer avec l'ordre 1 en géométrie (figure 3.15(a) que la convergence sature très vite. L'ordre 2 en géométrie (figure 3.15(b)) permet d'améliorer nettement les résultats avec une convergence optimale obtenue jusqu'à P4. La convergence atteint ensuite un seuil et l'augmentation de l'ordre polynomial ne permet plus d'améliorer la solution. L'erreur géométrique devient dominante. Enfin, nous avons les figures 3.15(c) et 3.15(d) qui représentent l'ordre géométrique 3 et 4. Nous pouvons voir l'amélioration de la convergence en p avec la disparition des seuils de saturation. Cependant, les pentes de convergences s'affaiblissent avec l'augmentation de l'ordre géométrique. Ces effets sont peut-être encore dus à l'utilisation de formules de quadrature qui ne sont pas assez précises dans la construction des termes non standards.

3.2 Problème de Stokes

Nous passons maintenant à la formulation point-selle de la FBM pour les équations de Stokes. Comme précédemment, nous allons traiter la formulation mathématique pour le cas où l'on a une seule perforation. L'extension à de multiples perforations se fera très naturellement. On suppose aussi que la frontière Γ délimitant le domaine Ω est décomposée en deux parties $\Gamma = \Gamma_D \cup \Gamma_N$. Pour présenter la méthode FBM pour les équations de Stokes, nous allons considérer le problème suivant :

Trouver $(\tilde{\boldsymbol{u}}, \tilde{p}) \in \mathbf{H}^1\left(\Omega\backslash\bar{B}\right) \times L^2\left(\Omega\backslash\bar{B}\right)$ telles que :

$$
\begin{aligned}
-\Delta\tilde{\boldsymbol{u}} + \nabla\tilde{p} &= \boldsymbol{f} \quad \text{dans } \Omega\backslash\bar{B} && (3.40)\\
\nabla\cdot\tilde{\boldsymbol{u}} &= 0 \quad \text{dans } \Omega\backslash\bar{B} && (3.41)\\
\tilde{\boldsymbol{u}} &= \boldsymbol{g} \quad \text{sur } \Gamma_D && (3.42)\\
\left(-\tilde{p}\boldsymbol{I} + \nabla\tilde{\boldsymbol{u}}\right)\boldsymbol{n} &= \boldsymbol{0} \quad \text{sur } \Gamma_N && (3.43)\\
\tilde{\boldsymbol{u}} &= \boldsymbol{0} \quad \text{sur } \gamma && (3.44)
\end{aligned}
$$

Nous préférons résoudre ce problème avec des conditions de Dirichlet et Neumann, car c'est ce type de conditions aux limites qui pourrait intervenir dans des écoulements de fluide avec entrée et sortie. Aussi cela nous évite d'avoir le problème de définition de la pression à une constante près dans le cas où l'on aurait uniquement des conditions de Dirichlet sur l'ensemble de Γ. La formulation variationnelle de ce problème prend la forme suivante :

Trouver $(\boldsymbol{u}, p, \hat{\boldsymbol{u}}, \hat{p}, \bar{p}, \boldsymbol{\lambda}, \boldsymbol{\phi}) \in \mathbf{H}^1_{\boldsymbol{g},\Gamma_D}(\Omega) \times L^2(\Omega) \times \mathbf{H}^1(\omega) \times L^2(\omega) \times \mathbb{P}_0(\omega) \times \mathbf{H}^{-1/2}(\gamma) \times \mathbf{H}^{-1/2}(\gamma')$ telles que :

$$
\begin{aligned}
\int_\Omega \nabla\boldsymbol{u} : \nabla\boldsymbol{v} - \int_\Omega p\nabla\cdot\boldsymbol{v} - \int_\gamma \boldsymbol{\lambda}\cdot\boldsymbol{v} &= \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}, \quad \forall\boldsymbol{v} \in \mathbf{H}^1_{0,\Gamma_D}(\Omega) && (3.45)\\
\int_\Omega q\nabla\cdot\boldsymbol{u} &= 0, \quad \forall q \in L^2(\Omega) && (3.46)\\
\int_\omega \nabla\hat{\boldsymbol{u}} : \nabla\hat{\boldsymbol{v}} - \int_\omega \hat{p}\nabla\cdot\hat{\boldsymbol{v}} - \int_\gamma \boldsymbol{\lambda}\cdot\hat{\boldsymbol{v}} - \int_{\gamma'} \boldsymbol{\phi}\cdot\hat{\boldsymbol{v}} &= \int_\omega \boldsymbol{f}\cdot\hat{\boldsymbol{v}}, \quad \forall\hat{\boldsymbol{v}} \in \mathbf{H}^1(\omega) && (3.47)\\
\int_\omega \hat{q}\nabla\cdot\hat{\boldsymbol{u}} - \int_\omega \bar{p}\hat{q} &= 0, \quad \forall\hat{q} \in L^2(\omega) && (3.48)\\
\int_\omega p\bar{q} - \int_\omega \hat{p}\bar{q} &= 0, \quad \forall\bar{q} \in \mathbb{P}_0(\omega) && (3.49)\\
\int_\gamma \hat{\boldsymbol{u}}\cdot\boldsymbol{\mu} &= 0, \quad \forall\boldsymbol{\mu} \in \mathbf{H}^{-1/2}(\gamma) && (3.50)\\
\int_{\gamma'} (\boldsymbol{u} - \hat{\boldsymbol{u}})\cdot\boldsymbol{\psi} &= 0, \quad \forall\boldsymbol{\psi} \in \mathbf{H}^{-1/2}(\gamma') && (3.51)
\end{aligned}
$$

Théorème 2. *Le problème* (3.45)-(3.51) *admet une unique solution* $(\boldsymbol{u}, \hat{\boldsymbol{u}}, p, \hat{p}, \bar{p}, \boldsymbol{\lambda}, \boldsymbol{\phi})$.

Dans le système précédent, plusieurs contraintes ont été intégrées à l'aide de multiplicateurs de Lagrange. La condition aux limites de Dirichlet sur γ a été imposée à l'aide du multiplicateur $\boldsymbol{\lambda}$ et de l'équation (3.50). Nous avons assuré la contrainte de continuité de la vitesse sur γ' à l'aide du multiplicateur $\boldsymbol{\phi}$ et de l'équation (3.51). Enfin, une dernière contrainte sur la pression est nécessaire, assurant la même moyenne de la pression globale p et locale $\hat{p}$ sur ω. Celle-ci est imposée par le multiplicateur $\bar{p}$ et l'équation (3.49).

Condition de dirichlet non homogène

Dans le cas où l'on souhaiterait avoir une condition non homogène sur le bord γ, il est nécessaire de rajouter comme pour la formulation I de la FBM un sous-problème

sur la perforation B. La solution de ce problème nous permettra ensuite d'ajouter une contribution dans les équations définies sur le domaine global Ω pour assurer la bonne prise en compte de cette condition aux limites. Soit $\boldsymbol{h} \in \mathbf{H}^{1/2}(\gamma)$ la condition qui doit être imposée. Le problème à résoudre sur la perforation B est encore un problème de Stokes qui correspond à :

Trouver $(\mathring{\boldsymbol{u}}, \mathring{p}) \in \mathbf{H}^1(B) \times L^2(B)$ telles que :

$$\begin{aligned} -\Delta\mathring{\boldsymbol{u}} + \nabla\mathring{p} &= \boldsymbol{f} \quad \text{dans } B && (3.52)\\ \nabla \cdot \mathring{\boldsymbol{u}} &= 0 \quad \text{dans } B && (3.53)\\ \mathring{\boldsymbol{u}} &= \boldsymbol{h} \quad \text{sur } \gamma && (3.54) \end{aligned}$$

Par contre, ce problème n'admet pas une unique solution parce que la pression est définie à une constante près. Pour rendre ce problème bien posé, il faut ajouter une contrainte sur la pression. On impose alors que la moyenne de la pression dans la perforation soit égale à la moyenne de la pression globale dans B.

Ensuite nous devons ajouter la contribution suivante au second membre de l'équation (3.45) :

$$\int_\gamma \mathring{\boldsymbol{\sigma}}\boldsymbol{n}_B \cdot v \tag{3.55}$$

avec $\mathring{\boldsymbol{\sigma}} = -\mathring{p}I + \nabla\mathring{\boldsymbol{u}}$ et $\boldsymbol{n}_B$ la normale extérieure par rapport à la perforation B sur la frontière γ. Enfin, il reste à prendre en compte la condition de Dirichlet imposée par un multiplicateur de Lagrange en remplaçant l'équation (3.50) par :

$$\int_\gamma \hat{\boldsymbol{u}} \cdot \boldsymbol{\mu} = \int_\gamma \boldsymbol{h} \cdot \boldsymbol{\mu} \tag{3.56}$$

Discrétisation

Nous présentons maintenant les espaces d'approximation discrets choisis pour représenter les inconnues et multiplicateurs de la formulation. Commençons par les espaces associés respectivement à la vitesse et pression sur le domaine global :

$$\begin{aligned} V_\delta &= \mathbf{H}^1_{g,\Gamma_D}(\Omega_\delta) \cap \left(P^c_N(\Omega_\delta)\right)^d \\ P_\delta &= P^c_M(\Omega_\delta) \end{aligned}$$

Ensuite nous avons les espaces pour la vitesse et la pression définis dans le problème local et l'espace du multiplicateur de Lagrange assurant la moyenne de la pression sur ω :

$$\begin{aligned} \hat{V}_\delta &= \left(P^c_{\hat{N}}(\omega_\delta)\right)^d \\ \hat{P}_\delta &= P^c_{\hat{M}}(\omega_\delta) \\ \bar{P}_\delta &= P^c_0(\omega_\delta) \end{aligned}$$

Enfin, nous avons les deux espaces associés respectivement aux multiplicateurs de Lagrange sur γ et γ' :

$$\begin{aligned} L^\gamma_\delta &= \left(P^c_Q(\gamma_\delta)\right)^d \\ L^{\gamma'}_\delta &= \left(P^c_{Q'}(\gamma'_\delta)\right)^d \end{aligned}$$

Pour assurer la stabilité des équations discrètes, le choix des degrés d'approximation ne peut pas être pris au hasard. Nous choisissons d'utiliser les éléments de Taylor-Hood, un choix très classique qui assure la condition inf-sup au niveau discret (voir chapitre 4). Cela se traduit par les relations $M = N-1$ et $\hat{M} = \hat{N}-1$. Au niveau des multiplicateurs, nous prenons des ordres d'approximation égaux à l'ordre de la vitesse locale, c'est-à-dire $\hat{N} = Q = Q'$.

Méthode de résolution monolithique

La première méthode utilisée pour résoudre le système (3.45)-(3.51) est une méthode monolithique, c'est-à-dire que l'ensemble des équations va être résolu simultanément. C'est la même technique que celle utilisée pour le problème de Poisson précédemment. L'avantage est d'obtenir directement la solution sans passer par l'intermédiaire d'un algorithme itératif en résolvant une suite de sous-problèmes. L'inconvénient principal est qu'il faut assembler tous les termes de l'équation dans une matrice représentant l'opérateur linéaire de ce problème. En particulier, les termes non standards peuvent être assez coûteux en temps de calcul. Nous donnons ci-dessous la structure algébrique du système monolithique :

$$\underbrace{\left(\begin{array}{cc|ccc|cc} A_{\boldsymbol{uu}} & A_{\boldsymbol{u}p} & 0 & 0 & 0 & A_{\boldsymbol{u\lambda}} & 0 \\ A_{p\boldsymbol{u}} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & A_{\hat{\boldsymbol{u}}\hat{\boldsymbol{u}}} & A_{\hat{u}\hat{p}} & 0 & A_{\hat{\boldsymbol{u}}\boldsymbol{\lambda}} & A_{\hat{\boldsymbol{u}}\boldsymbol{\phi}} \\ 0 & 0 & A_{\hat{p}\hat{\boldsymbol{u}}} & 0 & A_{\hat{p}\bar{p}} & 0 & 0 \\ 0 & A_{\bar{p}p} & 0 & A_{\bar{p}\hat{p}} & 0 & 0 & 0 \\ \hline 0 & 0 & A_{\boldsymbol{\lambda}\hat{\boldsymbol{u}}} & 0 & 0 & 0 & 0 \\ \hline A_{\boldsymbol{\phi u}} & 0 & A_{\boldsymbol{\phi}\hat{\boldsymbol{u}}} & 0 & 0 & 0 & 0 \end{array}\right)}_{A} \underbrace{\left(\begin{array}{c} \boldsymbol{u} \\ p \\ \hline \hat{\boldsymbol{u}} \\ \hat{p} \\ \bar{p} \\ \hline \boldsymbol{\lambda} \\ \hline \boldsymbol{\phi} \end{array}\right)}_{U} = \underbrace{\left(\begin{array}{c} F_{\boldsymbol{u}} \\ 0 \\ \hline F_{\hat{\boldsymbol{u}}} \\ 0 \\ 0 \\ \hline 0 \\ \hline 0 \end{array}\right)}_{F} \tag{3.57}$$

Méthode de résolution point fixe

Une autre manière pour résoudre le problème (3.45)-(3.51) est de découpler ces équations. L'idée est exactement la même que pour l'algorithme de résolution de la formulation I de la FBM. Nous allons résoudre successivement le problème local puis le problème global et répéter ce processus jusqu'à convergence. Nous utiliserons également cette méthode de point fixe avec une relaxation de la solution globale évaluée dans le problème local à chaque itération. Nous notons par $\boldsymbol{u}^k$ la représentation d'une fonction $\boldsymbol{u}$ à l'itération k. On définit alors la fonction de relaxation $\boldsymbol{\mathcal{K}}^k : \gamma' \longmapsto \mathbb{R}^d$ par :

$$\boldsymbol{\mathcal{K}}^k = \theta^k \boldsymbol{u}_\delta^k\big|_{\gamma'} + \left(1-\theta^k\right)\boldsymbol{\mathcal{K}}^{k-1} \tag{3.58}$$

L'algorithme de résolution est similaire à celui présenté pour la formulation elliptique, algorithme 2.3. Soit $\hat{\boldsymbol{u}}^0$, $\hat{p}^0$, $\bar{p}^0$, $\boldsymbol{\lambda}^0$, $\boldsymbol{\phi}^0$ et $\boldsymbol{\mathcal{K}}^0$ donnés. Pour $k > 1$, le problème local et le problème global sont ensuite donnés respectivement par les deux formulations variationnelles suivantes :

Trouver $(\hat{\boldsymbol{u}}_\delta^k, \hat{p}_\delta^k, \bar{p}_\delta^k, \boldsymbol{\lambda}_\delta^k, \boldsymbol{\phi}_\delta^k) \in \hat{V}_\delta \times \hat{P}_\delta \times \bar{P}_\delta \times L_\delta^\gamma \times L_\delta^{\gamma'}$ tel que :

$$\int_\omega \nabla \hat{\boldsymbol{u}}_\delta^k \cdot \nabla \hat{\boldsymbol{v}} - \int_\omega \hat{p}_\delta^k \nabla \cdot \hat{\boldsymbol{v}} - \int_\gamma \boldsymbol{\lambda}_\delta^k \cdot \hat{\boldsymbol{v}} - \int_{\gamma'} \boldsymbol{\phi}_\delta^k \cdot \hat{\boldsymbol{v}} = \int_\omega \boldsymbol{f} \cdot \hat{\boldsymbol{v}}, \quad \forall \hat{\boldsymbol{v}} \in \hat{V}_\delta \tag{3.59}$$

$$\int_\omega \hat{q} \nabla \cdot \hat{\boldsymbol{u}}_\delta^k - \int_\omega \bar{p}_\delta^k \hat{q} = 0, \quad \forall \hat{q} \in \hat{P}_\delta \tag{3.60}$$

$$\int_\omega \hat{p}_\delta^k \bar{q} = \int_\omega p_\delta^{k-1} \bar{q}, \quad \forall \bar{q} \in \bar{P}_\delta \tag{3.61}$$

$$\int_\gamma \hat{\boldsymbol{u}}_\delta^k \cdot \boldsymbol{\mu} = 0, \quad \forall \boldsymbol{\mu} \in L_\delta^\gamma \tag{3.62}$$

$$\forall \boldsymbol{\psi} \in L_\delta^{\gamma'}, \quad \int_{\gamma'} \hat{\boldsymbol{u}}_\delta^k \cdot \boldsymbol{\psi} = \int_{\gamma'} \boldsymbol{\mathcal{K}}^{k-1} \cdot \boldsymbol{\psi} \tag{3.63}$$

Trouver $(\boldsymbol{u}_\delta^k, p_\delta^k) \in V_\delta \times P_\delta$ tel que $\forall (\boldsymbol{v}, q) \in \left[\mathbf{H}^1_{\mathbf{0},\Gamma_D}(\Omega_\delta) \cap (P_N^c(\Omega_\delta))^d\right] \times P_\delta$:

$$\int_\Omega \nabla \boldsymbol{u}_\delta^k \cdot \nabla \boldsymbol{v} - \int_\Omega p^k \nabla \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} + \int_\gamma \boldsymbol{\lambda}_\delta^k \cdot \boldsymbol{v} \tag{3.64}$$

$$\int_\Omega q \nabla \cdot \boldsymbol{u}_\delta^k = 0 \tag{3.65}$$

La convergence de l'algorithme sera considérée avec le même critère que nous avons proposé pour la formulation elliptique de la FBM, inéquation (3.24). Les paramètres de relaxation θ^k seront également calculés par la méthode d'Aitken.

3.3 Applications numériques

Dans cette partie, nous présenterons quelques résultats numériques de la nouvelle formulation FBM pour Stokes. Nous montrerons des applications physiques et des tests de convergence. Pour tous ces tests, nous avons utilisé la version monolithique de notre formulation FBM.

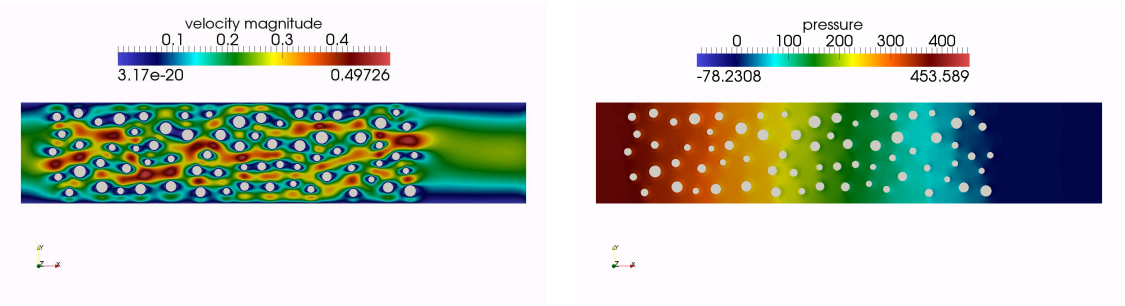
Application 1 : écoulement autour de particules fixes

Pour ce premier exemple d'application de la FBM dans un contexte de la mécanique des fluides, nous avons réalisé quelques simulations d'écoulement autour de plusieurs particules fixes. Les équations traitées sont les équations de Stokes avec un profil parabolique en entrée, une condition de non-glissement sur la paroi et une sortie à contrainte libre.

Le premier exemple est un écoulement 2D avec 70 perforations. La géométrie du domaine Ω est un rectangle $(0, 5) \times (-0.5, 0.5)$ avec en entrée l'expression suivante :

$$\boldsymbol{u}(x, y) = -(0.5 - y)(0.5 + y)\, \boldsymbol{n}$$

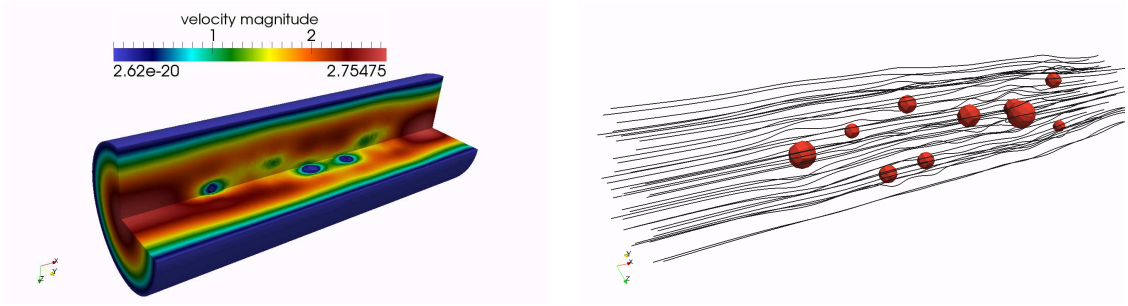
Les figures 3.16(a) et 3.16(b) montrent les champs de vitesse et pression obtenus. Sur ces illustrations, nous avons la solution globale et superposées à celle-ci les solutions locales qui sont plus précises. Nous avons aussi rajouté en blanc les perforations pour avoir un rendu similaire au problème perforé initialement posé. Avec la figure 3.16(c),



(a) Magnitude la vitesse (b) Pression

(c) Lignes de courant

FIGURE 3.16 – Solution 2D de Stokes obtenue avec la FBM



(a) Magnitude la vitesse (b) Lignes de courant

FIGURE 3.17 – Solution 3D de Stokes obtenue avec la FBM

nous avons tracé quelques lignes de courant calculées à partir de l'entrée sur le domaine global uniquement. En rouge, nous avons aussi représenté les perforations.

Nous avons également effectué un test similaire, mais en 3D avec 10 perforations. La géométrie de Ω est un cylindre de longueur 3, de rayon 0.5 et d'axe (Ox). Les particules à l'intérieur du cylindre sont représentées par des sphères. Le profil parabolique en entrée est donnée par :

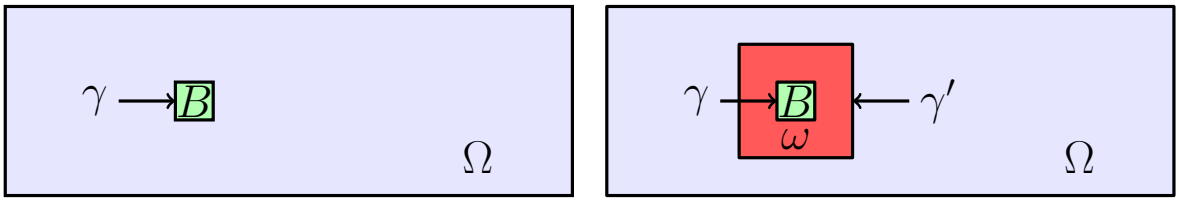
$$\boldsymbol{u}(x,y,z) = \frac{1.8}{0.1681}\left(y^2+z^2-0.25\right)\boldsymbol{n}$$

Des résultats de ce test 3D sont présentés avec les figures 3.17(a) et 3.17(b).

Application 2 : convergence numérique avec condition aux limites homogène

Nous voulons maintenant étudier le comportement numérique de la nouvelle formulation (3.45)-(3.51). L'idée est de se donner une solution analytique et de mesurer l'erreur commise en norme L^2 et H^1 avec les solutions calculées (global et locales). Cependant, il

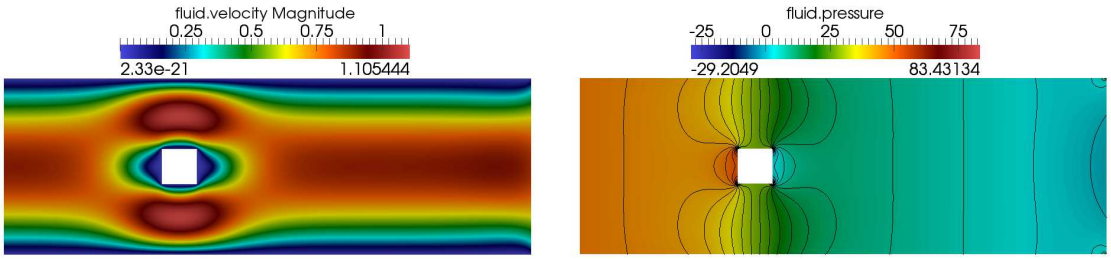
n'existe pas de solutions analytiques relativement simples du problème de Stokes autour d'un ou plusieurs obstacles, avec non-glissement autour de ceux-ci. Pour pallier à cette difficulté, nous choisissons d'utiliser une solution numérique calculée sur un maillage très fin comme solution de référence.



(a) Géométrie du probleme initial (b) Géométrie FBM

FIGURE 3.18 – Géométries utiliseés dans l'application 2

Le problème initial modélise un écoulement de Stokes autour d'un obstacle carré, figure 3.18(a). Le fluide est impulsé par une entrée parabolique et la sortie est imposée par une condition de contrainte libre. Le reste des frontières correspond au non-glissement du fluide. La solution de référence est calculée par une méthode éléments finis standard avec une approximation P_2 en vitesse et P_1 en pression. Nous avons avec les figures 3.19(a) et 3.19(b) les solutions vitesse et pression de référence.

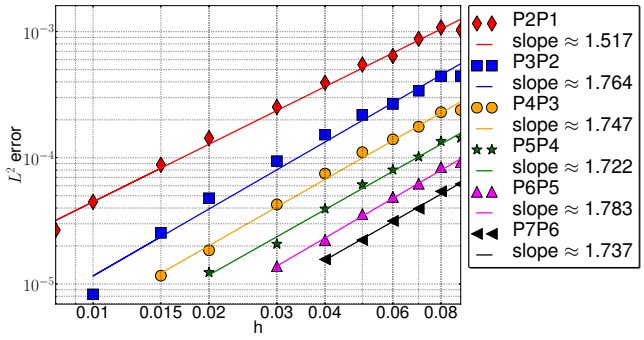


(a) Magnitude de la vitesse (b) Champ de pression

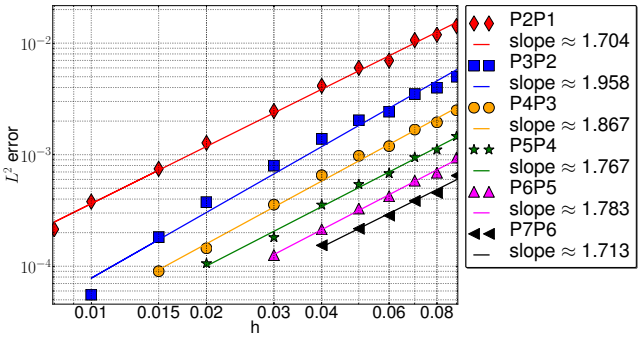
FIGURE 3.19 – Solution de référence calculée sur le maillage fin. Le maillage contient 271775 triangles. Le nombre de degrés de liberté (vitesse+pression) est de 1227370.

Nous avons effectué les tests de convergence pour différents ordres polynomiaux, mais à chaque fois nous avons $M = N-1 = \hat{M} = \hat{N}-1 = Q-1 = Q'-1$. Une difficulté dans ce genre de test est de comparer les solutions FBM qui ne sont pas représentées sur le même maillage que celui de la solution de référence. Nous avons alors recours à l'interpolation. L'erreur de convergence est calculée sur le maillage très fin, les solutions FBM sont alors interpolées sur ce maillage.

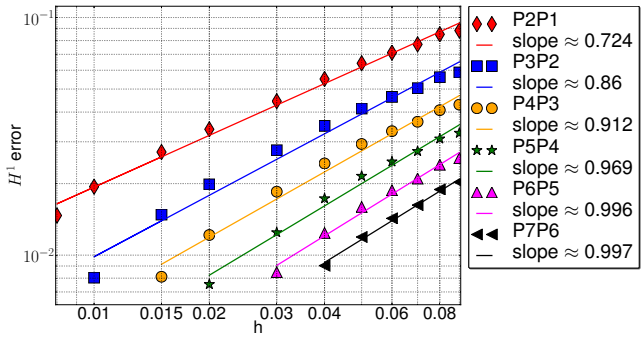
Comme le montrent les figures 3.20(a), 3.20(b), 3.20(c) et 3.20(d) la convergence du champ vitesse n'est pas optimale. Cependant, contrairement aux problèmes de Poisson, nous pouvons augmenter l'ordre polynomial et réussir à résoudre le système FBM. De plus, la précision s'accroit avec l'augmentation de l'ordre polynomial. Malheureusement, les ordres de convergence ne sont pas significativement meilleurs. On remarque aussi que la convergence semble s'accélérer dès que le pas de maillage devient petit, on le voit particulièrement bien sur la figure 3.20(d).



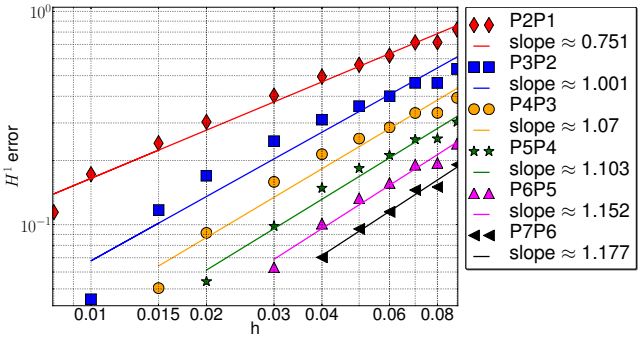
(a) Erreur L^2 sur $\Omega\backslash\overline{(B \cup \omega)}$



(b) Erreur L^2 sur ω



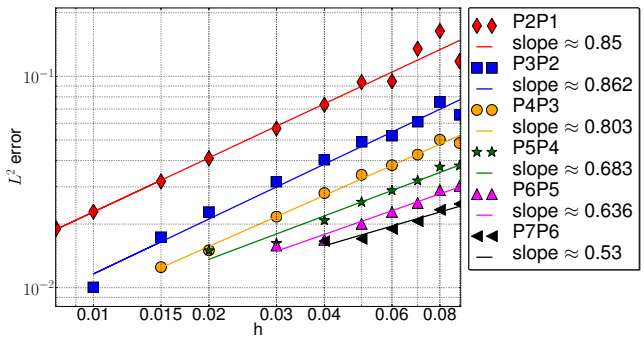
(c) Erreur H^1 sur $\Omega\backslash\overline{(B \cup \omega)}$



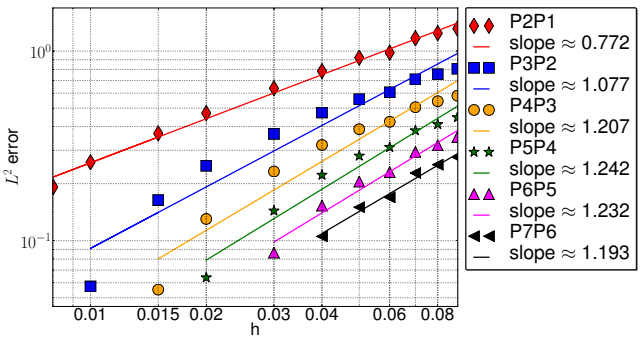
(d) Erreur H^1 sur ω

FIGURE 3.20 – Erreur L^2 et H^1 de la vitesse entre la solution FBM et la solution de référence.

Au niveau de la pression, un comportement similaire apparait, figures 3.21(a) et 3.21(b). Cependant les erreurs sur $\Omega\backslash\overline{(B \cup \omega)}$ semblent atteindre un seuil de saturation, particulièrement à l'ordre élevé.



(a) Erreur L^2 sur $\Omega\backslash\overline{(B \cup \omega)}$



(b) Erreur L^2 sur ω

FIGURE 3.21 – Erreur L^2 de la pression entre la solution FBM et la solution de référence.

Ces tests nous montrent la convergence de notre méthode FBM, mais malheureusement nous indiquent aussi que l'optimalité de la méthode n'est pas au rendez-vous. Par contre, on peut très bien voir sur cet exemple l'intérêt de monter en ordre polynomial. Une étude plus approfondie, aussi bien numérique que théorique, est nécessaire pour essayer de comprendre ce comportement.

Application 3 : convergence numérique avec condition aux limites non homogène

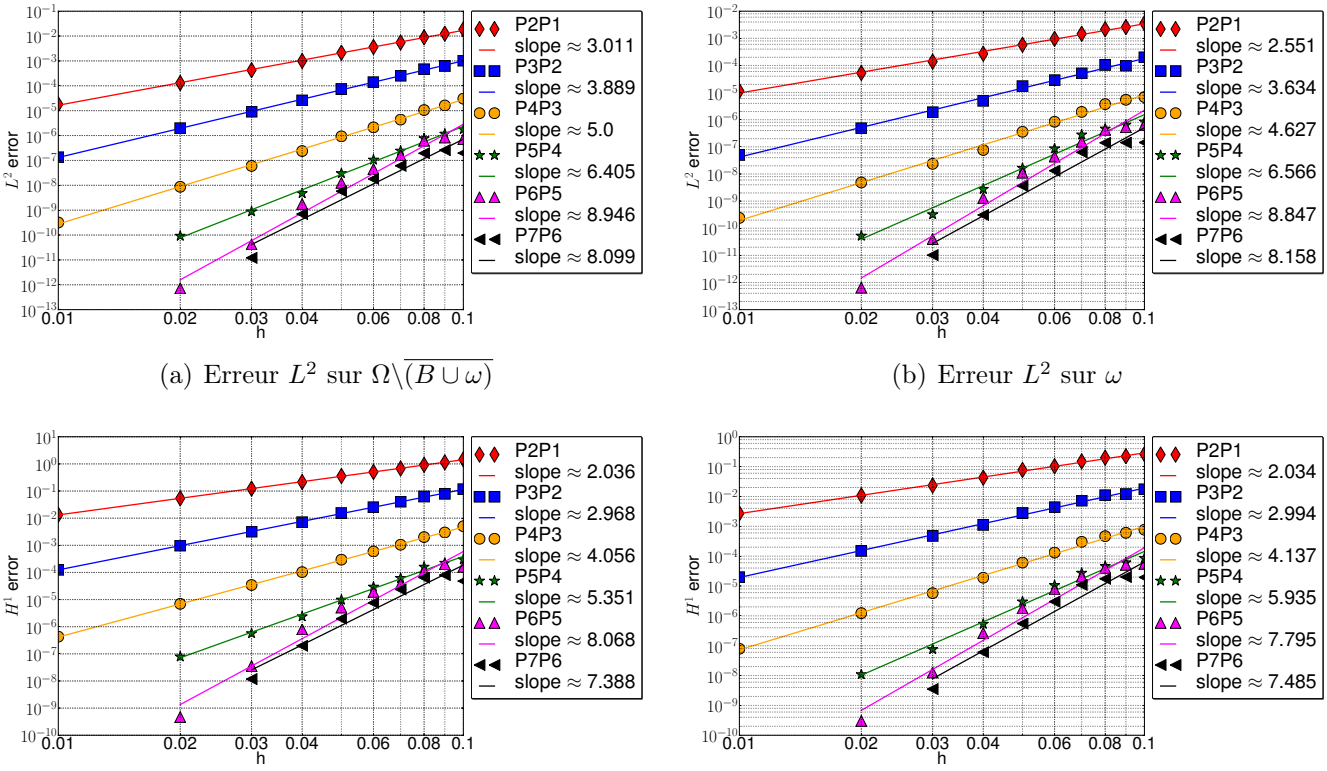
On s'intéresse maintenant à la résolution du problème de Stokes avec des conditions de Dirichlet non homogènes sur le bord de la perforation γ. Nous voulons effectuer un test de convergence sur les solutions FBM en vitesse et pression. Nous choisissons d'utiliser pour ce cas la solution analytique de Kovaznay $\boldsymbol{K} = (K_u, K_v, K_p)$ comme solution de référence :

$$K_u(x,y) = 1 - e^{ax}\cos(2\pi y) \tag{3.66}$$

$$K_v(x,y) = \frac{a}{2\pi}e^{ax}\sin(2\pi y) \tag{3.67}$$

$$K_p(x,y) = \frac{1}{2}\left(1 - e^{2ax}\right) \tag{3.68}$$

avec $a = 1/2\nu - \sqrt{1/4\nu^2 + 4\pi^2}$. La variable K_u (resp K_v) est la composante x (resp y) de la vitesse et K_p de la pression. Le second membre $\boldsymbol{f}$ et les conditions aux limites sur Ω et γ sont ensuite déduits grâce à la solution exacte. Au niveau de la géométrie, le domaine Ω est représenté par le rectangle $(-0.5, 1) \times (-0.5, 1.5)$ et la perforation B est représentée par le cercle centré en $(-0.15, 0.5)$ de rayon 0.15.



(a) Erreur L^2 sur $\Omega\backslash\overline{(B \cup \omega)}$

(b) Erreur L^2 sur ω

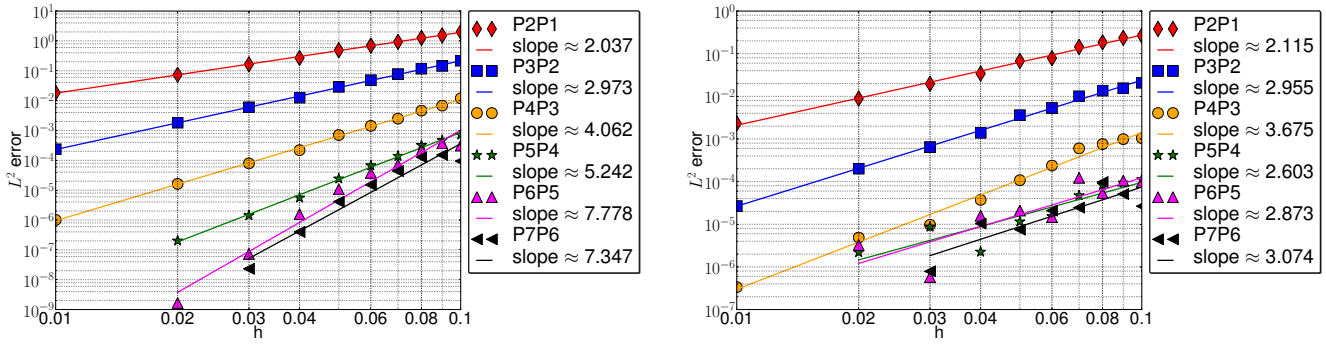
(c) Erreur H^1 sur $\Omega\backslash\overline{(B \cup \omega)}$

(d) Erreur H^1 sur ω

FIGURE 3.22 – Graphes de convergence mesurant l'erreur L^2 et H^1 de la vitesse en fonction du pas de maillage h. Plusieurs approximations polynomiales ont été utilisées dans ce test. Leurs pentes de convergence sont données dans les légendes associées.

Cependant, le fait de connaitre $\boldsymbol{f}$ sur l'ensemble du domaine Ω implique que la solution calculée u sera naturellement convergente vers $\boldsymbol{K}$, si toutefois l'intégrale sur γ de l'équation (3.45) correspond bien à la contribution recherchée. Dans le cas contraire, des problèmes de convergence sont attendus. Ce test permet de vérifier des parties essentielles de cette formulation FBM, en particulier la convergence de la solution locale et par conséquent le rôle des multiplicateurs de Lagrange, ainsi que l'apport du problème sur la perforation B.

Pour le test sur la vitesse, nous obtenons les graphes de convergence 3.22(a), 3.22(b), 3.22(c) et 3.22(d). Les erreurs L^2 et H^1 sur $\Omega\backslash\overline{(B \cup \omega)}$ se comportent de manière optimale jusqu'à P_5P_4. Pour les ordres supérieurs, les erreurs semblent ne pas bien converger pour les pas de maillage grossiers, du même ordre que P_5P_4, mais on peut remarquer que ce problème se dissipe en raffinant le maillage. On peut voir que les courbes sur ω ont le même comportement que celle de l'erreur globale. Ceci nous permet de valider le rôle des multiplicateurs de Lagrange $\boldsymbol{\lambda}$ et $\boldsymbol{\phi}$.



(a) Erreur L^2 sur $\Omega\backslash\overline{(B \cup \omega)}$

(b) Erreur L^2 sur ω

FIGURE 3.23 – Graphes de convergence mesurant l'erreur L^2 et H^1 de la pression en fonction du pas de maillage h. Plusieurs approximations polynomiales ont été utilisées dans ce test. Leurs pentes de convergence sont données dans les légendes associées.

Pour la pression, nous avons le même phénomène sur l'erreur globale évoquée précédemment, figure 3.23(a). Par contre, nous pouvons voir avec la figure 3.23(b) que l'erreur locale en pression pour les ordres supérieurs à P_4P_3 se comporte assez étrangement. Nous pouvons peut-être remettre en cause le multiplicateur de Lagrange $\bar{p}$.

3.4 Extension à Navier-Stokes

Précédemment, nous avons utilisé les équations de Stokes pour modéliser la dynamique des fluides. Nous souhaitons maintenant étendre ce travail au modèle de fluide non linéaire de Navier-Stokes. Nous nous placerons d'abord dans le cadre d'un régime stationnaire, puis nous considérerons le régime transitoire.

Navier-Stokes stationnaire

Pour obtenir ce modèle de fluide, nous définissons tout d'abord les tenseurs des contraintes $\boldsymbol{\sigma}$ et $\hat{\boldsymbol{\sigma}}$ associés respectivement au domaine globale et au domaine local :

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\boldsymbol{D}(\boldsymbol{u}) \tag{3.69}$$
$$\hat{\boldsymbol{\sigma}} = -\hat{p}\boldsymbol{I} + 2\mu\boldsymbol{D}(\hat{\boldsymbol{u}}) \tag{3.70}$$
$$\boldsymbol{D}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right) \tag{3.71}$$

où $\boldsymbol{D}(\boldsymbol{u})$ correspond au tenseur des déformations et la constante μ est la viscosité dynamique du fluide. On notera également par ρ la masse volumique du fluide supposé aussi constante.

De la même manière que le système d'équations (3.45)-(3.47), nous décrivons la formulation FBM de Navier-Stokes par le problème suivant :

Trouver $(\boldsymbol{u}, p, \hat{\boldsymbol{u}}, \hat{p}, \bar{p}, \boldsymbol{\lambda}, \boldsymbol{\phi}) \in \mathbf{H}^1_{g,\Gamma_D}(\Omega) \times L^2(\Omega) \times \mathbf{H}^1(\omega) \times L^2(\omega) \times \mathbb{P}_0(\omega) \times \mathbf{H}^{-1/2}(\gamma) \times \mathbf{H}^{-1/2}(\gamma')$ telles que $\forall \boldsymbol{v} \in \mathbf{H}^1_{0,\Gamma_D}(\Omega)$, $\forall q \in L^2(\Omega)$, $\forall \hat{\boldsymbol{v}} \in \mathbf{H}^1(\omega)$, $\forall \hat{q} \in L^2(\omega)$, $\forall \bar{q} \in \mathbb{P}_0(\omega)$, $\forall \boldsymbol{\mu} \in \mathbf{H}^{-1/2}(\gamma)$, $\forall \boldsymbol{\psi} \in \mathbf{H}^{-1/2}(\gamma')$:

$$\rho\int_\Omega (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v} + \int_\Omega \boldsymbol{\sigma} : \nabla\boldsymbol{v} - \int_\gamma \boldsymbol{\lambda}\cdot\boldsymbol{v} = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v} \tag{3.72}$$
$$\int_\Omega q\nabla\cdot\boldsymbol{u} = 0 \tag{3.73}$$
$$\rho\int_\omega (\hat{\boldsymbol{u}}\cdot\nabla)\hat{\boldsymbol{u}}\cdot\hat{\boldsymbol{v}} + \int_\omega \hat{\boldsymbol{\sigma}} : \nabla\hat{\boldsymbol{v}} - \int_\gamma \boldsymbol{\lambda}\cdot\hat{\boldsymbol{v}} - \int_{\gamma'} \boldsymbol{\phi}\cdot\hat{\boldsymbol{v}} = \int_\omega \boldsymbol{f}\cdot\hat{\boldsymbol{v}} \tag{3.74}$$
$$\int_\omega \hat{q}\nabla\cdot\hat{\boldsymbol{u}} - \int_\omega \bar{p}\hat{q} = 0 \tag{3.75}$$
$$\int_\omega p\bar{q} - \int_\omega \hat{p}\bar{q} = 0 \tag{3.76}$$
$$\int_\gamma \hat{\boldsymbol{u}}\cdot\boldsymbol{\mu} = 0 \tag{3.77}$$
$$\int_{\gamma'} (\boldsymbol{u} - \hat{\boldsymbol{u}})\cdot\boldsymbol{\psi} = 0 \tag{3.78}$$

Théorème 3. *Le problème* (3.72)-(3.78) *admet une unique solution* $(\boldsymbol{u}, \hat{\boldsymbol{u}}, p, \hat{p}, \bar{p}, \boldsymbol{\lambda}, \boldsymbol{\phi})$.

Pour tester numériquement notre méthode FBM avec les équations de Navier-Stokes, nous avons choisi de reproduire le « benchmark » proposé par [144]. Il représente un écoulement laminaire (Re=20) en 2D autour d'un obstacle circulaire. La géométrie de ce problème est représentée par la figure 3.24. La frontière gauche est l'entrée d'un écoulement parabolique donné par :

$$\boldsymbol{u}(x, y) = -\frac{4U_{max}y(0.41 - y)}{0.41^2}\boldsymbol{n} \tag{3.79}$$

La frontière de droite correspond à la sortie du fluide. On impose sur ce bord une condition de Neumann homogène, $\boldsymbol{\sigma n} = \mathbf{0}$. Les deux autres frontières ont des conditions de non-glissement, c'est à dire $\boldsymbol{u} = (0, 0)$.

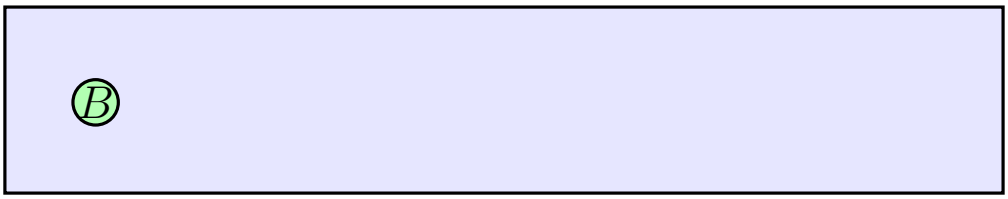


FIGURE 3.24 – Géométrie du probleme initial [144]

Dans notre formulation, le domaine global Ω va alors représenter le rectangle sans la perforation circulaire. Le domaine local ω est la zone rouge du schéma 3.25. Il est choisi sous la forme d'un disque englobant la perforation B et paramétré par l'épaisseur $|\omega|$.

FIGURE 3.25 – Géometrie du problème global et local

Les quantités mesurées dans ce benchmark sont la portance (*lift*), la traînée (*drag*) et la différence de pression Δp entre les deux points A(0.15,0.2) et B(0.25,02). Ces mesures sont calculées par les formules suivantes :

$$C_D = \frac{2F_D}{\rho U_{mean}^2 L} \tag{3.80}$$

$$C_L = \frac{2F_L}{\rho U_{mean}^2 L} \tag{3.81}$$

$$\Delta p = p(A) - p(B) \tag{3.82}$$

Les forces F_D et F_L exercées sur le cercle sont exprimées à l'aide des expressions (3.83). On voit apparaitre $\boldsymbol{\sigma}$ le tenseur de contrainte interne au fluide et $\boldsymbol{n}$ le vecteur normal par rapport au cercle et dirigé vers l'intérieur.

$$\boldsymbol{F} = \begin{pmatrix} F_D \\ F_L \end{pmatrix} = -\int \boldsymbol{\sigma}\ \boldsymbol{n} \quad \text{avec} \quad \boldsymbol{\sigma} = -p\boldsymbol{I} + \mu\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right) \tag{3.83}$$

Nous présentons les résultats obtenus de ce benchmark avec les tables 3.1 et 3.2. Pour tous les tests, nous avons utilisé le même maillage et une approximation spatiale P_2P_1 pour le domaine global Ω. Le nombre de degrés de liberté associé (vitesse + pression) est de 263002. Au niveau du domaine local, la largeur de l'anneau ω est également fixé à 0.025. Par contre nous avons utilisé différentes approximations géométriques N_{geo} et discrétisations spatiales $P_{\hat{N}}P_{\hat{M}}$ où $\hat{N}$ (resp $\hat{M}$) est l'ordre d'approximation de la vitesse

$\hat{\boldsymbol{u}}$ (resp de la pression $\hat{p}$). Nous avons aussi à chaque fois 6 niveaux de raffinement de ω_δ, notés symboliquement de M0 à M5 (du plus grossier au plus fin). Au début de la table nous donnons également les bornes supérieures et inférieures de validité calculées à partir des résultats obtenus par plusieurs codes [144]. Nous avons également affiché les résultats obtenus par un calcul ordre élevé [87] et par un calcul effectué avec notre logiciel Feel++ avec une méthode éléments finis classiques.

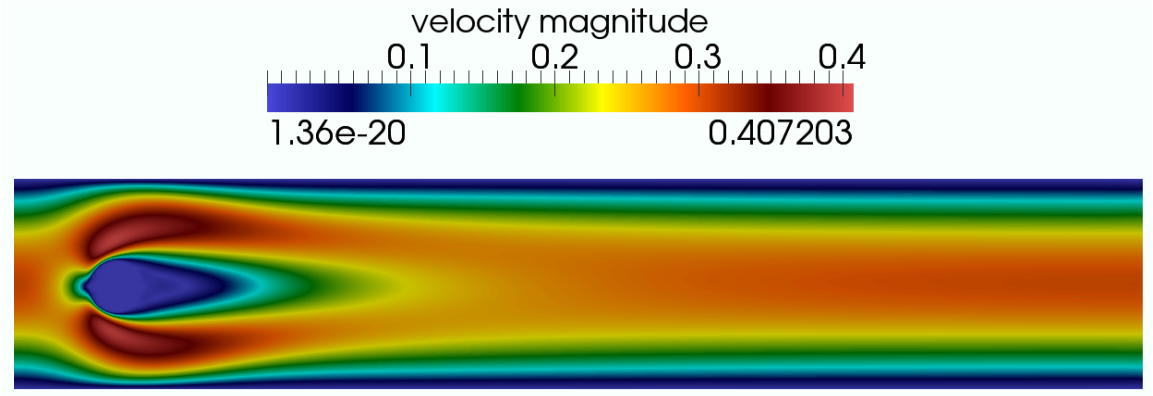
La table 3.1 nous montre que pour un choix d'approximation géométrique, les quantités mesurées convergent assez bien dans l'intervalle de validité. Seul le coefficient C_L pour l'ordre géométrique 1 en P_2P_1 n'atteint pas l'intervalle, cependant l'approximation reste encore relativement « grossière ». Les approximations géométriques d'ordre supérieur donnent de bien meilleurs résultats. Avec la table 3.2 nous montrons l'intérêt de l'approximation géométrique qui apporte de meilleurs résultats pour une même discrétisation en espace avec le même nombre d'éléments dans le maillage. Enfin, nous pouvons voir un résultat obtenu avec la FBM avec les figures 3.26(a) et 3.26(b). Elles illustrent la magnitude du champ de vitesse et la pression.

	N_{geo}	$P_{\hat{N}}P_{\hat{M}}$	N_{dof}	C_D	C_L	Δp
lower bound				5.5700	0.0104	0.1172
upper bound				5.5900	0.0110	0.1176
REF [87]				5.579535	0.010618	0.11752
Feel++	1	P_2P_1	4159615	5.57779	0.0106035	0.117519
(M0)	1	P_2P_1	1235	5.5043	0.00835463	0.117259
(M1)	1	P_2P_1	2041	5.5309	0.00982127	0.117421
(M2)	1	P_2P_1	3368	5.55414	0.0121961	0.117363
(M3)	1	P_2P_1	7092	5.56471	0.010117	0.117439
(M4)	1	P_2P_1	12297	5.57034	0.0107688	0.117465
(M5)	1	P_2P_1	26869	5.57419	0.0103372	0.117508
(M0)	2	P_3P_2	2857	5.57878	0.0103238	0.117365
(M1)	2	P_3P_2	4785	5.57963	0.0105141	0.117435
(M2)	2	P_3P_2	8001	5.57972	0.0103966	0.117469
(M3)	2	P_3P_2	16991	5.57966	0.010538	0.117505
(M4)	2	P_3P_2	29509	5.57961	0.0105497	0.117512
(M5)	2	P_3P_2	64899	5.57957	0.0105608	0.117519
(M0)	3	P_4P_3	5175	5.57984	0.0106074	0.117585
(M1)	3	P_4P_3	8729	5.57967	0.010576	0.117561
(M2)	3	P_4P_3	14492	5.57959	0.0105189	0.117542
(M3)	3	P_4P_3	31244	5.57958	0.0105451	0.117526
(M4)	3	P_4P_3	54608	5.57955	0.0105518	0.117523
(M5)	3	P_4P_3	120062	5.57954	0.0105552	0.117521

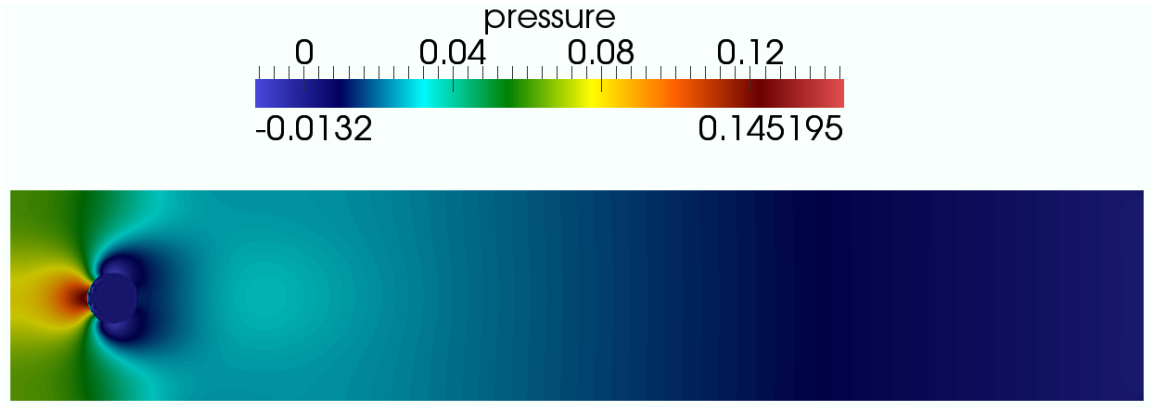
Table 3.1 – Résultats du benchmark obtenus avec la FBM comparés à des mesures de *référence*.

	N_{geo}	$P_{\hat{N}}P_{\hat{M}}$	N_{dof}	C_D	C_L	Δp
(M0)	1	P_3P_2	2857	5.52926	0.01013	0.116754
(M0)	2	P_3P_2	2857	5.57878	0.0103238	0.117365
(M1)	1	P_3P_2	4785	5.54537	0.0105122	0.117088
(M1)	2	P_3P_2	4785	5.57963	0.0105141	0.117435
(M0)	1	P_4P_3	5175	5.52884	0.0104191	0.117842
(M0)	2	P_4P_3	5175	5.57982	0.0105537	0.117501
(M0)	3	P_4P_3	5175	5.57984	0.0106074	0.117585

TABLE 3.2 – Résultats du benchmark obtenus avec la FBM en confrontant l'ordre géométrique utilisé.



(a) Champ de vitesse



(b) Champ de pression

FIGURE 3.26 – Solution calculée avec la FBM. La solution locale est superposée sur la solution globale. On visualise aussi l'intérieur de la perforation, qui résulte de la solution globale.

Navier-Stokes transitoire

Nous voulons maintenant passer à un modèle d'écoulement transitoire, c'est-à-dire qu'il faut ajouter le terme d'inertie $\rho\frac{\partial \boldsymbol{u}}{\partial t}$ à l'équation du mouvement de Navier-Stokes. Au niveau de la formulation FBM, cela revient à ajouter ce terme dans le problème global et le problème local. Pour discrétiser le terme $\frac{\partial \boldsymbol{u}}{\partial t}$, nous utilisons un schéma *backward differentiation formulation* d'ordre q (BDF_q). Nous notons par Δt le pas de temps supposé fixe. On définit également la fonction $\boldsymbol{u}$ au temps t^n par $\boldsymbol{u}^n$. Le terme d'inertie au temps t^{n+1} est alors approché par le schéma suivant :

$$\frac{\partial \boldsymbol{u}^{n+1}}{\partial t} = \frac{\beta_{-1}}{\Delta t}\boldsymbol{u}^{n+1} - \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\boldsymbol{u}^{n-j} \tag{3.84}$$

où les β_j sont les coefficients de ces schémas. Ces coefficients sont donnés dans la table 4.1 du chapitre 4.

Soient $\boldsymbol{u}^0$ et $\hat{\boldsymbol{u}}^0$ les conditions initiales données. La formulation FBM de Navier-Stokes au temps t^{n+1} est donnée par le système :

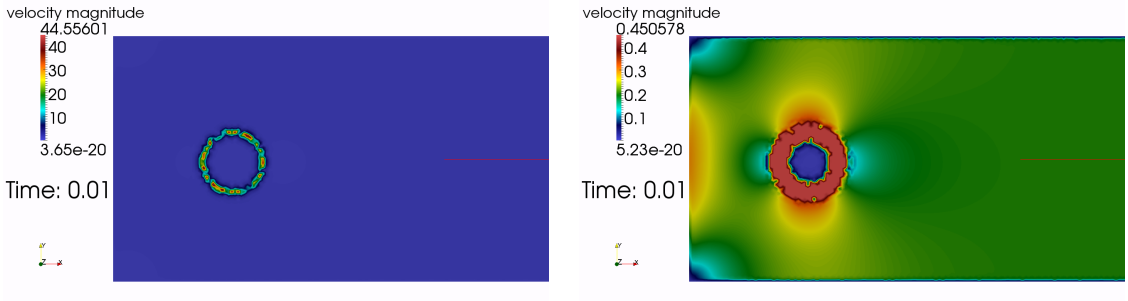
Trouver $(\boldsymbol{u}^{n+1}, p^{n+1}, \hat{\boldsymbol{u}}^{n+1}, \hat{p}^{n+1}, \bar{p}^{n+1}, \boldsymbol{\lambda}^{n+1}, \boldsymbol{\phi}^{n+1}) \in \mathbf{H}^1_{g,\Gamma_D}(\Omega)\times L^2(\Omega)\times\mathbf{H}^1(\omega)\times L^2(\omega)\times \mathbb{P}_0(\omega)\times\mathbf{H}^{-1/2}(\gamma)\times\mathbf{H}^{-1/2}(\gamma')$ telles que $\forall \boldsymbol{v}\in\mathbf{H}^1_{0,\Gamma_D}(\Omega)$, $\forall q\in L^2(\Omega)$, $\forall\hat{\boldsymbol{v}}\in\mathbf{H}^1(\omega)$, $\forall\hat{q}\in L^2(\omega)$, $\forall\bar{q}\in\mathbb{P}_0(\omega)$, $\forall\boldsymbol{\mu}\in\mathbf{H}^{-1/2}(\gamma)$, $\forall\boldsymbol{\psi}\in\mathbf{H}^{-1/2}(\gamma')$:

$$\begin{aligned}
\rho\int_\Omega\left(\frac{\beta_{-1}}{\Delta t}\boldsymbol{u}^{n+1} + (\boldsymbol{u}^{n+1}\cdot\nabla)\,\boldsymbol{u}^{n+1}\right)\cdot\boldsymbol{v} + \int_\Omega \boldsymbol{\sigma}^{n+1}:\nabla\boldsymbol{v} & \\
-\int_\gamma \boldsymbol{\lambda}^{n+1}\cdot\boldsymbol{v} &= \int_\Omega \boldsymbol{f}^n\cdot\boldsymbol{v} && (3.85)\\
\int_\Omega q\nabla\cdot\boldsymbol{u}^{n+1} &= 0 && (3.86)\\
\rho\int_\Omega\left(\frac{\beta_{-1}}{\Delta t}\hat{\boldsymbol{u}}^{n+1} + (\hat{\boldsymbol{u}}^{n+1}\cdot\nabla)\hat{\boldsymbol{u}}^{n+1}\right)\cdot\hat{\boldsymbol{v}} + \int_\omega \hat{\boldsymbol{\sigma}}^{n+1}:\nabla\hat{\boldsymbol{v}} & \\
-\int_\gamma \boldsymbol{\lambda}^{n+1}\cdot\hat{\boldsymbol{v}} - \int_{\gamma'}\boldsymbol{\phi}^{n+1}\cdot\hat{\boldsymbol{v}} &= \int_\omega \hat{\boldsymbol{f}}^n\cdot\hat{\boldsymbol{v}} && (3.87)\\
\int_\omega \hat{q}\nabla\cdot\hat{\boldsymbol{u}}^{n+1} - \int_\omega \bar{p}^{n+1}\hat{q} &= 0 && (3.88)\\
\int_\omega p^{n+1}\bar{q} - \int_\omega \hat{p}\bar{q}^{n+1} &= 0 && (3.89)\\
\int_\gamma \hat{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{\mu} &= 0 && (3.90)\\
\int_{\gamma'}\left(\boldsymbol{u}^{n+1}-\hat{\boldsymbol{u}}^{n+1}\right)\cdot\boldsymbol{\psi} &= 0 && (3.91)
\end{aligned}$$

où l'on a noté : $\boldsymbol{f}^n = \boldsymbol{f} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\boldsymbol{u}^{n-j}$ et $\hat{\boldsymbol{f}}^n = \boldsymbol{f} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\hat{\boldsymbol{u}}^{n-j}$.

Pour illustrer ce modèle, nous reprenons l'exemple du benchmark précédent avec les mêmes conditions aux limites. Nous utilisons les conditions initiales $\boldsymbol{u}^0 \equiv \mathbf{0}$ et $\hat{\boldsymbol{u}}^0 \equiv \mathbf{0}$ avec un pas de temps $\Delta t = 0.01$. Les paramètres physiques sont $\rho = 1$ et $\mu = 0.00025$, ce qui nous donnent un nombre de Reynolds de 80.

Le système de Navier-Stokes transitoire avec la FBM n'arrive pas à être résolu dès le premier pas de temps. Nous utilisons une méthode de Newton pour la résolution du système non linéaire. Pour essayer de comprendre ce problème, nous avons essayé un modèle plus simple, le modèle d'Oseen. Ces équations sont identiques aux équations de Navier-Stokes, seul le terme convectif est extrapolé ce qui rend le problème à nouveau linéaire. Dans ce cas, le solveur algébrique arrive à résoudre le système linéaire, mais nous fait apparaitre de fortes instabilités au niveau de la perforation. La figure 3.27(a) nous montre cette instabilité et les grandes valeurs que prend le champ de vitesse au niveau de la perforation au premier pas de temps. Nous avons la même solution représentée avec la figure 3.27(b), mais l'échelle du champ de vitesse a été modifiée. On s'aperçoit que la solution reste *physiquement* correcte lorsque l'on s'éloigne de la perforation B. Après quelques pas de temps, le calcul « crash » à cause des instabilités grandissantes. On peut aussi remarquer que l'amplitude des instabilités est proportionnelle la valeur du nombre de Reynolds. Plus ce nombre est grand, plus les instabilités sont grandes. Pour des Reynolds faibles, les instabilités apparaissent toujours, mais arrivent à être dissipées avec l'apparition de petites perturbations *non physiques* dans l'écoulement.



(a) Echelle min-max (b) Echelle physique

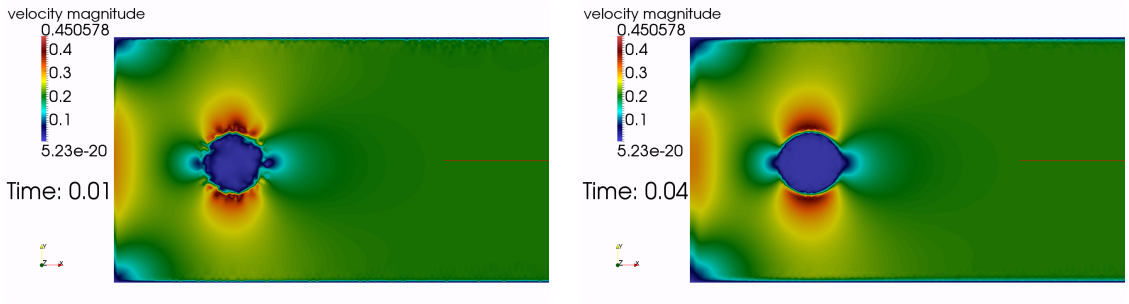
FIGURE 3.27 – Champ de vitesse obtenue avec des équations d'Oseen pour la FBM non stabilisée à t=0.01. Deux échelles de visualisation sont présentées.

Nous proposons maintenant une solution à ce problème d'instabilités numériques. Pour cela, nous ajoutons un terme de stabilisation au problème global qui va renforcer la contrainte d'incompressibilité du fluide. Cette contrainte est prise en compte de manière faible en ajoutant à la formulation variationnelle le terme suivant dans l'équation (3.85) :

$$\int_\Omega (\nabla \cdot \boldsymbol{u}^{n+1})(\nabla \cdot \boldsymbol{v}) \tag{3.92}$$

La figure 3.28(a) est la solution du premier pas de temps pour le modèle simplifié d'Oseen. L'ajout de ce terme de stabilisation a bien fait disparaitre les fortes instabilités. Il reste toutefois quelques irrégularités proches de B, mais la solution locale corrige cela. De plus, ces irrégularités disparaissent complètement après quelques pas de temps comme le montre la figure 3.28(b).

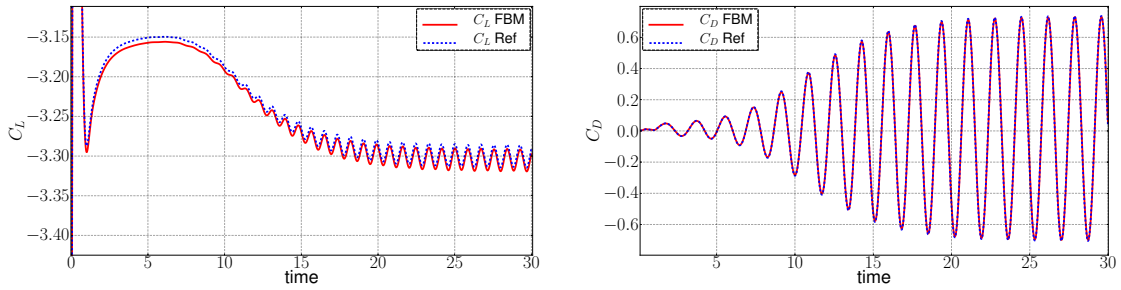
Remarque : Nous avons défini le terme de stabilisation (3.92) sur tout Ω. Comme les instabilités sont localisées près de la perforation B, cette intégrale pourrait être restreinte à la zone d'instabilité et ainsi réduire les coûts de calculs.



(a) t=0.01 (b) t=0.04

FIGURE 3.28 – Champ de vitesse obtenue avec les équations d'Oseen pour la FBM stabilisée.

Grâce à ce terme de stabilisation, le système non linéaire de Navier-Stokes peut être résolu avec succès. Nous avons mesuré les coefficients C_L et C_D du « benchmark » à chaque pas de temps en comparaison à une solution éléments finis classique que nous notons Ref. C'est ce que l'on a représenté avec 3.29(a) et 3.29(b). Les solutions FBM et Ref sont très similaires, ce qui nous permet de valider ce modèle. En effet, les courbes du coefficient C_D se superposent parfaitement, celles du coefficient C_L ont un léger décalage lié au démarrage de la simulation, mais globalement le comportement est identique.



(a) Coefficient C_L (b) Coefficient C_D

FIGURE 3.29 – Comparaison des coefficients C_L et C_D entre FBM et éléments finis pour le benchmark transitoire de Navier-Stokes.

4 Conclusion

Nous venons de présenter dans ce chapitre une description de la méthode FBM. Celle-ci est conçue pour la résolution de problèmes définis dans des domaines perforés. Le principe est de remplacer le problème initialement posé dans le domaine perforé par plusieurs sous-problèmes couplés de géométrie moins complexe. Des propriétés d'optimalité de la méthode (au sens élément fini) ont été démontrées dans [26]. De plus, un intérêt majeur

à cette méthode est que l'on peut avoir des discrétisations différentes entre le problème global et les problèmes locaux. Ainsi, on peut obtenir une très bonne précision autour des perforations (ordre élevé en espace et géométrie) et une approximation moins précise, mais plus efficace dans le domaine global. Deux types de formulation de la FBM ont été décrits et implémentés dans ce chapitre. La première correspond à la formulation classique qui a été utilisée dans [115, 86, 25, 26]. La seconde est une nouvelle approche qui a été initiée dans cette thèse. Son analyse théorique est en cours. De plus, l'implémentation de ces méthodes nous a permis d'utiliser et de vérifier le comportement des outils présentés au chapitre 1, à savoir l'opérateur d'interpolation et la construction de la méthode de Galerkin non standard.

La première formulation (nommé formulation I) est adaptée aux problèmes elliptiques. Elle correspond à la formulation initiale étudiée dans [86, 26]. Nous avons toutefois apporté une modification qui a permis de simplifier la prise en compte de géométries complexes. Cette amélioration consiste à rajouter un sous-problème sur chaque perforation. Chacun d'eux est de petite taille et indépendant du problème FBM couplé. Nous avons ensuite vérifié notre implémentation de la méthode à l'aide de plusieurs tests de convergence en h et p. Nous avons pu ainsi montrer l'optimalité de la méthode pour des ordres d'approximation relativement bas (≤ 4). Pour les ordres supérieurs, l'erreur de convergence semble atteindre un seuil de saturation. Nous ne pouvons pas encore donner une explication à ce phénomène. Cette formulation a aussi été étendue aux équations de la mécanique des fluides dans [86]. Grâce à l'utilisation des schémas de projection [3] ou des schémas de *splitting* [77], le modèle fluide utilisé peut être ramené sous la forme de plusieurs problèmes elliptiques, ce qui permet de rentrer dans le cadre cette formulation FBM.

Une nouvelle formulation de la FBM (nommée formulation II) a été introduite dans ce chapitre. La principale différence avec la version précédente est que le couplage entre le problème global et les problèmes locaux s'effectue à l'aide de multiplicateur de Lagrange. Le problème FBM couplé est alors de type point-selle. Cette nouvelle formulation a l'avantage d'être mieux adaptée aux équations de la dynamique des fluides, car elle s'applique directement sur le problème couplé vitesse pression. De plus, le recouvrement des domaines locaux est autorisé d'un point de vue théorique, ce qui n'était pas le cas pour la formulation initiale. L'analyse théorique de cette formulation est en cours et elle devrait faire l'objet de plusieurs publications prochainement [23, 24, 22]. Les tests de convergences effectués ne nous ont pas permis de montrer l'optimalité de la méthode. On remarque toutefois le bon comportement de l'approximation avec l'augmentation de l'ordre d'approximation. Nous avons également validé notre modèle de Navier-Stokes avec la FBM en comparant nos résultats avec des simulations éléments finis standards sur un maillage perforé. Cette formulation va être ensuite étendue au chapitre 6 pour la modélisation de l'interaction fluide-particules. Les perforations (et donc les domaines locaux) se déplaceront dans l'écoulement.

Deuxième partie

Interaction fluide-structure

Chapitre 4

Navier-Stokes en domaine mobile

Les vaisseaux sanguins permettent l'acheminement du sang entre les divers organes du corps. Nous rappelons que le sang est composé d'un fluide, le plasma, et de cellules sanguines en suspension. Les divers types de vaisseaux sanguins sont des matériaux plus ou moins élastiques [15, 114], c'est-à-dire qu'ils peuvent subir des déformations. En effet, lors de cette circulation du sang, celui-ci interagit avec la paroi du vaisseau qui l'entoure et peut provoquer son déplacement. Nous souhaitons donc pouvoir modéliser un écoulement sanguin dans un vaisseau mobile. Pour cela, nous utiliserons un modèle newtonien qui est valable pour les vaisseaux de diamètre relativement large ($> 1\ mm$). C'est le cas des principales grandes artères élastiques. Pour les vaisseaux de diamètre inférieur (le cas de certains capillaire), le comportement rhéofluidifiant du sang ne peut plus être négligé et cela nécessite l'utilisation de modèle non newtonien [136, 47, 69, 59].

Dans ce chapitre, nous allons présenter les équations de la dynamique des fluides en domaine mobile. La prise en compte du mouvement du domaine est faite par une méthode ALE dont nous avons fait une description dans le chapitre 2. Nous commencerons par énoncer les notions de repère ALE et de vitesse de déformation d'un domaine nécessaire pour la prise en compte du mouvement. Celles-ci pourront être après intégrées dans les équations de Navier-Stokes modélisant le fluide. Ensuite, nous présenterons le modèle, la formulation variationnelle ainsi que les discrétisations numériques utilisées dans ce contexte. Ce travail s'appuie fortement sur la stratégie utilisée dans [134], où le lecteur pourra trouver plus de détails théoriques. Puis, nous présenterons des applications numériques 2D et 3D dans le but de vérifier une propriété du schéma numérique en temps. Ce test est appelé *Geometric Conservation Law* (GCL), il a été utilisé dans [123, 124] pour des modèles de fluide 2D. Nous terminerons par une description de quelques spécificités qui peuvent être utilisées pour la simulation des écoulements sanguins.

1 Équations dans le repère ALE

Soit Ω_f^t le domaine représentant le fluide à un instant t. Nous posons par t_i et t_f le temps initial et le temps final. Nous utilisons la notation $\boldsymbol{u}_f$ pour décrire la vitesse du fluide et p_f pour la pression. Au niveau des paramètres physiques, nous notons ρ_f la densité ou masse volumique du fluide et μ_f la viscosité dynamique du fluide. En mécanique des fluides, les équations de Navier-Stokes sont des équations aux dérivées partielles non linéaires qui permettent de décrire le mouvement des fluides newtoniens. Nous admettrons également l'hypothèse d'incompressibilité dans ce modèle de fluide. Ainsi, dans le repère

eulérien, les équations de Navier-Stokes incompressible se présentent respectivement par les équations de conservation de la quantité de mouvement et de conservation de la masse :

$$\rho_f \left.\frac{\partial \mathbf{u}_f}{\partial t}\right|_{\mathbf{x}} + \rho_f \left(\boldsymbol{u}_f \cdot \nabla_{\mathbf{x}}\right) \boldsymbol{u}_f - \nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma}_f = \boldsymbol{f}_f^t, \quad \text{dans } \Omega_f^t \times [t_i, t_f] \tag{4.1}$$

$$\nabla_{\mathbf{x}} \cdot \boldsymbol{u}_f = 0, \quad \text{dans } \Omega_f^t \times [t_i, t_f] \tag{4.2}$$

Ce modèle doit ensuite être complété avec des conditions aux limites. Nous en parlerons dans la section 2 de ce chapitre. Le comportement du fluide est caractérisé par le tenseur des contraintes $\boldsymbol{\sigma}_f$ que l'on décompose généralement par la forme suivante :

$$\boldsymbol{\sigma}_f = -p_f \boldsymbol{I} + \boldsymbol{\tau}$$

où $\boldsymbol{I}$ est le tenseur identité. Cette décomposition du tenseur des contraintes fait apparaitre un tenseur sphérique $-p_f \boldsymbol{I}$ dont la forme correspond à celle du tenseur des contraintes pour une pression hydrostatique. Nous avons également un tenseur déviatorique $\boldsymbol{\tau}$ qui dépend de la viscosité μ_f et du tenseur des déformations $\boldsymbol{D}\left(\boldsymbol{u}_f\right)$. Pour un modèle newtonien incompressible, le tenseur déviatorique s'exprime par :

$$\boldsymbol{\tau} = 2\mu_f \boldsymbol{D}\left(\boldsymbol{u}_f\right) \ , \ \boldsymbol{D}\left(\boldsymbol{u}_f\right) = \frac{1}{2}\left(\nabla_{\mathbf{x}} \boldsymbol{u}_f + \left(\nabla_{\mathbf{x}} \boldsymbol{u}_f\right)^T\right)$$

Le symbole $\nabla_{\mathbf{x}}$ désigne le gradient dans le repère eulérien.

Nous allons maintenant transformer les équations de Navier-Stokes 4.1-4.2 dans un nouveau repère, le repère ALE. Nous commençons par introduire la notion de carte ALE, notée $\mathcal{A}^t$, dont la construction a été détaillée dans le chapitre 2. On considère Ω_f^* un domaine de référence. Généralement, on choisit $\Omega_f^* = \Omega_f^{t_i}$. La carte ALE permet de relier le domaine de référence et le domaine déformé. Cette application est définie par :

$$\begin{aligned} \mathcal{A}^t : \Omega_f^* &\longrightarrow \Omega_f^t \\ \mathbf{x}^* &\longmapsto \mathbf{x} \equiv \mathbf{x}\left(\mathbf{x}^*, t\right) = \mathcal{A}^t\left(\mathbf{x}^*\right) \end{aligned}$$

Nous avons souvent besoin de transporter des fonctions définies dans le repère eulérien vers le repère ALE et inversement. Soit $f : \Omega_f^t \times [t_i, t_f] \longrightarrow \mathbb{R}$ une fonction définie dans le repère eulérien et $\hat{f} = f \circ \mathcal{A}^t$ la fonction équivalente dans le repère ALE définie par :

$$\begin{aligned} \hat{f} : \Omega_f^* \times [t_i, t_f] &\longrightarrow \mathbb{R} \\ \left(\mathbf{x}^*, t\right) &\longmapsto \hat{f}(\mathbf{x}^*, t) = f\left(\mathcal{A}^t\left(\mathbf{x}^*\right), t\right) \end{aligned}$$

Nous introduisons également la notation suivante $\left.\frac{\partial}{\partial t}\right|_{\mathbf{x}^*}$ pour décrire la dérivée temporelle dans le repère ALE. Celle-ci est définie par :

$$\begin{aligned} \left.\frac{\partial f}{\partial t}\right|_{\mathbf{x}^*} : \Omega_f^t \times [t_i, t_f] &\longrightarrow \mathbb{R} \\ (\mathbf{x}, t) &\longmapsto \left.\frac{\partial f}{\partial t}\right|_{\mathbf{x}^*} (\mathbf{x}, t) = \frac{\partial \hat{f}}{\partial t}\left(\left(\mathcal{A}^t\right)^{-1}(\mathbf{x}), t\right) \end{aligned}$$

Par analogie, nous notons la dérivée temporelle dans le repère eulérien avec la notation $\frac{\partial}{\partial t}\big|_{\mathbf{x}}$. Nous avons aussi besoin de définir la vitesse de déformation du domaine, également appelé vitesse ALE. Nous notons par $\boldsymbol{w}_f$ cette quantité qui s'exprime par :

$$\boldsymbol{w}_f(\mathbf{x},t) = \left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{x}^*}(\mathbf{x},t) = \frac{\partial \mathbf{x}}{\partial t}\left(\left(\mathcal{A}^t\right)^{-1}(\mathbf{x}),t\right) \tag{4.3}$$

Cette notion de vitesse de déformation est nécessaire pour l'écriture des dérivées temporelles dans le repère ALE. En effet, nous ne souhaitons plus exprimer l'équation de la conservation de la quantité de mouvement 4.1 à l'aide de la dérivée eulérienne. Nous voulons plutôt l'écrire en utilisant la dérivée ALE. Ainsi, la relation entre les dérivées eulériennes et ALE est donnée par l'équation suivante :

$$\left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}^*} = \left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}} + \boldsymbol{w}_f \cdot \nabla_{\mathbf{x}} \boldsymbol{u}_f \tag{4.4}$$

Après l'introduction de ces nouvelles notions, nous pouvons maintenant écrire les équations de Navier-Stokes incompressible dans le repère ALE :

$$\rho_f \left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}^*} + \rho_f\left(\left(\boldsymbol{u}_f - \boldsymbol{w}_f\right)\cdot\nabla_{\mathbf{x}}\right)\boldsymbol{u}_f - \nabla_{\mathbf{x}}\cdot\boldsymbol{\sigma}_f = \boldsymbol{f}_f^t, \quad \text{dans } \Omega_f^t \times [t_i,t_f] \tag{4.5}$$

$$\nabla_{\mathbf{x}}\cdot\boldsymbol{u}_f = 0, \quad \text{dans } \Omega_f^t \times [t_i,t_f] \tag{4.6}$$

2 Formulation variationnelle

Après avoir décrit le modèle de mécanique des fluides pour un domaine mobile, nous allons présenter l'écriture de la formulation variationnelle associée à un problème type. Tout d'abord, nous supposons avoir une condition initiale sur la vitesse notée $\boldsymbol{u}_{f,0}$, c'est-à-dire que $\boldsymbol{u}_f = \boldsymbol{u}_{f,0}$ dans $\Omega_f^{t_i}$. Nous admettons aussi que la frontière $\partial\Omega_f^*$ est composée de deux parties Γ_D^* et Γ_N^*. Nous posons $\Gamma_D^t = \mathcal{A}^t(\Gamma_D^*)$ et $\Gamma_N^t = \mathcal{A}^t(\Gamma_N^*)$ et nous supposons connaitre les fonctions $\boldsymbol{g}_D^t \in [L^2(\Gamma_D^t)]^d$ et $\boldsymbol{g}_N^t \in [L^2(\Gamma_N^t)]^d$, $\forall t \in [t_i,t_f]$. Nous pouvons ainsi définir les conditions aux limites de notre problème de référence :

$$\boldsymbol{u}_f = \boldsymbol{g}_D^t, \quad \text{sur } \Gamma_D^t \times [t_i,t_f] \tag{4.7}$$

$$\boldsymbol{\sigma}_f \boldsymbol{n}_f = \boldsymbol{g}_N^t, \quad \text{sur } \Gamma_N^t \times [t_i,t_f] \tag{4.8}$$

Dans le but d'écrire la formulation variationnelle, nous introduisons maintenant les espaces de fonction définie $\forall t \in [t_i,t_f]$. Nous avons les espaces de fonction solution associée respectivement à la vitesse et pression :

$$V^t = \left\{\boldsymbol{v} : \Omega_f^t \longmapsto \mathbb{R}^d,\ \boldsymbol{v} = \hat{\boldsymbol{v}} \circ \left(\mathcal{A}^t\right)^{-1},\ \hat{\boldsymbol{v}} \in \mathbf{H}^1_{(\hat{\boldsymbol{g}}_D^t, \Gamma_D^*)}\left(\Omega_f^*\right)\right\} \tag{4.9}$$

$$Q^t = \left\{q : \Omega_f^t \longmapsto \mathbb{R},\ q = \hat{q} \circ \left(\mathcal{A}^t\right)^{-1},\ \hat{q} \in L^2\left(\Omega_f^*\right)\right\} \tag{4.10}$$

ainsi que l'espace des fonctions tests de la vitesse du fluide :

$$W^t = \left\{\boldsymbol{v} : \Omega_f^t \longmapsto \mathbb{R}^d,\ \boldsymbol{v} = \hat{\boldsymbol{v}} \circ \left(\mathcal{A}^t\right)^{-1},\ \hat{\boldsymbol{v}} \in \mathbf{H}^1_{(0,\Gamma_D^*)}\left(\Omega_f^*\right)\right\} \tag{4.11}$$

Ensuite, nous multiplions l'équation 4.5 (resp 4.6) avec une fonction test $\boldsymbol{v} \in W^t$ (resp $q \in Q^t$). On intègre ensuite chacune des équations sur le domaine Ω_f^t, et nous effectuons une intégration par partie du terme contenant le tenseur des contraintes $\boldsymbol{\sigma}_f$, ce qui nous donne :

$$\rho_f \int_{\Omega_f^t} \left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}^*} \cdot \boldsymbol{v} + \rho_f \int_{\Omega_f^t} \left((\boldsymbol{u}_f - \boldsymbol{w}_f) \cdot \nabla_{\mathbf{x}}\right) \boldsymbol{u}_f \cdot \boldsymbol{v} + \int_{\Omega_f^t} \boldsymbol{\sigma}_f : \nabla_{\mathbf{x}} \boldsymbol{v} - \int_{\partial\Omega_f^t} \boldsymbol{\sigma}_f \boldsymbol{n}_f \cdot v = \int_{\Omega_f^t} \boldsymbol{f}_f^t \cdot \boldsymbol{v} \tag{4.12}$$

$$\int_{\Omega_f^t} q \, \nabla_{\mathbf{x}} \cdot \boldsymbol{u}_f = 0 \tag{4.13}$$

Pour finir, nous devons prendre en compte les conditions aux limites sur Γ_D^t et Γ_N^t. Ainsi, nous obtenons la formulation variationnelle de Navier-Stokes incompressible en domaine mobile pour le problème de référence que nous avons considéré :

Trouver $(\boldsymbol{u}_f, p_f) \in V^t \times Q^t$ tel que $\forall\, (\boldsymbol{v}, q) \in W^t \times Q^t$ on a :

$$\rho_f \int_{\Omega_f^t} \left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}^*} \cdot \boldsymbol{v} + \rho_f \int_{\Omega_f^t} \left((\boldsymbol{u}_f - \boldsymbol{w}_f) \cdot \nabla_{\mathbf{x}}\right) \boldsymbol{u}_f \cdot \boldsymbol{v} + \int_{\Omega_f^t} \boldsymbol{\sigma}_f : \nabla_{\mathbf{x}} \boldsymbol{v} = \int_{\Omega_f^t} \boldsymbol{f}_f^t \cdot \boldsymbol{v} + \int_{\Gamma_N^t} \boldsymbol{g}_N^t \cdot \boldsymbol{v} \tag{4.14}$$

$$\int_{\Omega_f^t} q \, \nabla_{\mathbf{x}} \cdot \boldsymbol{u}_f = 0 \tag{4.15}$$

3 Stratégies de discrétisation

Nous allons commencer par introduire quelques notations associées à la discrétisation géométrique. Soit $\mathcal{T}_{f,\delta}^*$ le maillage issu de la discrétisation du domaine Ω_f^* et formant ainsi le domaine de calcul de référence $\Omega_{f,\delta}^*$. On note $\mathcal{A}_\delta^t$ la carte ALE discrète d'ordre k à l'instant t qui permet de donner la position du domaine déformé $\Omega_{f,\delta}^t$. On pose $\mathcal{T}_{f,\delta}^t$ le maillage formé par l'ensemble d'éléments suivants :

$$\mathcal{T}_{f,\delta}^t = \left\{K \mid K = \mathcal{A}_\delta^t\left(K^*\right), K^* \in \mathcal{T}_{f,\delta}^*\right\}$$

Ce maillage forme donc une partition pour le domaine $\Omega_{f,\delta}^t$. Nous pouvons aussi en déduire que $\Omega_{f,\delta}^t := \mathcal{A}_\delta^t(\Omega_{f,\delta}^*)$. On pose également $\Gamma_{D,\delta}^*$ et $\Gamma_{N,\delta}^*$ la frontière discrète associée respectivement à Γ_D^* et Γ_N^*. Nous définissons ainsi $\Gamma_{D,\delta}^t = \mathcal{A}_\delta^t(\Gamma_{D,\delta}^*)$ et $\Gamma_{N,\delta}^t = \mathcal{A}_\delta^t(\Gamma_{N,\delta}^*)$.

Nous échantillonnons la dimension temporelle en N_T pas de temps, notés $\{t_n\}_{1 \le n \le N_T}$. Nous avons alors $t_i \le t_n \le t_f, \forall n \in \{1, ..., N_T\}$ et $t_i < t_j$ si et seulement si $i < j$. On note $\boldsymbol{u}_{f,\delta}^n$ (resp $p_{f,\delta}^n$) le champ de vitesse (resp pression) discret au temps t_n.

3.1 Discrétisation spatiale

Nous commençons par décrire les espaces d'approximation que nous allons utiliser pour écrire la formulation variationnelle discrète. Soit :

$$\begin{aligned} V_\delta^t &= \left\{ \boldsymbol{v} : \Omega_{f,\delta}^t \longmapsto \mathbb{R}^d, \boldsymbol{v} = \hat{\boldsymbol{v}} \circ \left(\mathcal{A}_\delta^t\right)^{-1}, \hat{\boldsymbol{v}} \in \mathbf{H}^1_{(\hat{\boldsymbol{g}}_D^t, \Gamma_{D,\delta}^*)} \left(\Omega_{f,\delta}^*\right) \cap \left[P_c^M \left(\Omega_{f,\delta}^*\right)\right]^d \right\} \\ Q_\delta^t &= \left\{ q : \Omega_{f,\delta}^t \longmapsto \mathbb{R}, q = \hat{q} \circ \left(\mathcal{A}_\delta^t\right)^{-1}, \hat{q} \in P_c^N \left(\Omega_{f,\delta}^*\right) \right\} \end{aligned}$$

les espaces discrets associés respectivement au fluide et à la pression. Les degrés polynomiaux M et N ne peuvent être choisit librement, car le problème discret obtenu à partir de la formulation variationnelle continue ne sera pas forcément bien posée selon ce choix. Ce problème discret est de type point selle, les espaces d'approximation doivent alors vérifier une condition de compatibilité pour assurer l'existence et l'unicité du problème. Nous pouvons trouver par exemple les détails mathématiques dans [31]. Dans la littérature, cette condition est souvent appelée la condition de Babuška–Brezzi ou la condition de Ladyshenskaya–Babuška–Brezzi. Cette condition stipule que le problème discret est bien posé si et seulement si les espaces V_δ^t et Q_δ^t sont tels qu'il existe une constante β_δ telle que :

$$\inf_{q_\delta \in Q_\delta^t} \sup_{\boldsymbol{v}_\delta \in V_\delta^t} \frac{\int_{\Omega_{f,\delta}^t} q_\delta \, \nabla \cdot \boldsymbol{v}_\delta}{\|q_\delta\|_{L^2(\Omega_{f,\delta}^t)} \|\boldsymbol{v}_\delta\|_{\mathbf{H}^1(\Omega_{f,\delta}^t)}} \geq \beta_\delta \tag{4.16}$$

Si cette contrainte n'est pas vérifiée, alors la solution numérique peut faire apparaitre des instabilités. C'est le cas lorsque $M{=}N$. Des solutions existent pour résoudre ce problème. Elles consistent à rajouter des termes de stabilisation aux équations. Plusieurs exemples de stabilisation existent, nous pouvons citer par exemple les méthodes PSPG [151], SUPG [84] , GLS [85] et CIP [32]. Cependant nous préférons éviter l'ajout de ces termes en utilisant des éléments finis inf-sup qui vérifient directement la condition de compatibilité. Nous choisissons d'utiliser un des choix les plus courants en éléments finis, les éléments de Taylor-Hood [31, 53]. Les ordres polynomiaux M sont alors reliés par $N = M - 1$. De plus, ce choix permet d'obtenir des estimations d'erreurs optimales. Considérons par exemple le problème de Stokes (sans le terme d'inertie et de convection) sur un domaine Ω. Soit $(\boldsymbol{u}, p)$ la solution exacte supposée suffisamment régulière et $(\boldsymbol{u}_\delta, p_\delta)$ la solution éléments finis. Nous avons alors l'estimation d'erreur suivante :

$$\|\boldsymbol{u} - \boldsymbol{u}_\delta\|_{\mathbf{H}^1(\Omega)} + \|p - p_\delta\|_{\mathbf{L}^2(\Omega)} \leq C h^M \left(\|\boldsymbol{u}\|_{\mathbf{H}^M(\Omega)} + \|p\|_{H^N(\Omega)} \right) \tag{4.17}$$

Comme nous avons des conditions de Dirichlet non homogène, nous avons aussi besoin d'introduire l'espace discret des fonctions tests associé à la vitesse :

$$W_\delta^t = \left\{ v : \Omega_{f,\delta}^t \longmapsto \mathbb{R}^d, \boldsymbol{v} = \hat{\boldsymbol{v}} \circ \left(\mathcal{A}_\delta^t\right)^{-1}, \hat{\boldsymbol{v}} \in \mathbf{H}^1_{(0,\Gamma_{D,\delta}^*)} \left(\Omega_{f,\delta}^*\right) \cap \left[P_c^N \left(\Omega_{f,\delta}^*\right)\right]^d \right\} \tag{4.18}$$

Soit t_{n+1} un temps appartenant à l'intervalle de discrétisation du temps. La formulation variationnelle discrète au temps t_{n+1} se traduit par le problème suivant :

Trouver $(\boldsymbol{u}_{f,\delta}^{n+1}, p_{f,\delta}^{n+1}) \in V_\delta^{t_{n+1}} \times Q_\delta^{t_{n+1}}$ tel que $\forall (\boldsymbol{v}, q) \in W_\delta^{t_{n+1}} \times Q_\delta^{t_{n+1}}$:

$$\rho_f \int_{\Omega_{f,\delta}^{t_{n+1}}} \left.\frac{\partial \boldsymbol{u}_{f,\delta}^{n+1}}{\partial t}\right|_{\mathbf{x}^*} \cdot \boldsymbol{v} + \rho_f \int_{\Omega_{f,\delta}^{t_{n+1}}} \left(\left(\boldsymbol{u}_{f,\delta}^{n+1} - \boldsymbol{w}_{f,\delta}^{n+1}\right) \cdot \nabla_{\mathbf{x}}\right) \boldsymbol{u}_{f,\delta}^{n+1} \cdot \boldsymbol{v} + \int_{\Omega_{f,\delta}^{t_{n+1}}} \boldsymbol{\sigma}_{f,\delta}^{n+1} : \nabla_{\mathbf{x}} \boldsymbol{v} = \int_{\Omega_{f,\delta}^{t_{n+1}}} \boldsymbol{f}_f^{t_{n+1}} \cdot \boldsymbol{v} + \int_{\Gamma_{N,\delta}^{t_{n+1}}} \boldsymbol{g}_N^{t_{n+1}} \cdot \boldsymbol{v} \tag{4.19}$$

$$\int_{\Omega_{f,\delta}^{t_{n+1}}} q \, \nabla_{\mathbf{x}} \cdot \boldsymbol{u}_{f,\delta}^{n+1} = 0 \tag{4.20}$$

Cependant, même en utilisant les éléments finis de Taylor-Hood, la discrétisation numérique peut encore faire apparaitre des instabilités. C'est par exemple le cas lorsque l'on modélise un fluide avec un phénomène de convection dominante, la convection domine les autres forces mises en jeu. Pour résoudre ce problème, nous avons choisi d'utiliser une formulation éléments finis stabilisés basée sur la technique appelée *Continuous Interior Penalty* (CIP). Cette méthode de stabilisation est présentée dans [32, 124]. Celle-ci consiste à ajouter à l'équation 4.19 le terme présenté ci-dessous :

$$s\left(\boldsymbol{u}_{f,\delta}^n, \boldsymbol{v}; \boldsymbol{\beta}^n\right) = \gamma \sum_{F \in \mathcal{F}} \int_F \left|\left(\boldsymbol{\beta}^n - \boldsymbol{w}_{f,\delta}^n\right) \cdot \boldsymbol{n}_f\right| \frac{h_F^2}{N^{3.5}} [\![\boldsymbol{\nabla} \boldsymbol{u}_{f,\delta}^n]\!]_F \cdot [\![\boldsymbol{\nabla} \boldsymbol{v}]\!]_F \, ds \tag{4.21}$$

où $\mathcal{F}$ désigne un ensemble de faces de $\mathcal{T}_{f,\delta}^{t_n}$, $\boldsymbol{\beta}^n$ est une extrapolation de la vitesse $\boldsymbol{u}_{f,\delta}^n$, γ représente un paramètre, h_F désigne la taille caractéristique de la face F et N l'ordre polynomial de la vitesse. L'opérateur $[\![\cdot]\!]_F$ représente le saut à travers la face F. Plus précisément, ce saut est défini à partir des informations des deux éléments connectés à la face F, que l'on note T_1 et T_2. On note $\boldsymbol{n}_1$ (resp $\boldsymbol{n}_2$) la normale extérieure de l'élément T_1 (resp T_2). On note aussi $\boldsymbol{u}_1$ (resp $\boldsymbol{u}_2$) la restriction d'une fonction $\boldsymbol{u}$ sur T_1 (resp T_2). Le saut du gradient qui se trouve dans l'équation (4.21) peut alors s'exprimer :

$$[\![\boldsymbol{\nabla} \boldsymbol{u}]\!]_F = \nabla \boldsymbol{u}_1 \boldsymbol{n}_1 + \nabla \boldsymbol{u}_2 \boldsymbol{n}_2$$

Le principal inconvénient de cette stabilisation est que ce terme augmente le nombre d'entrées non nulles dans les matrices éléments finis. Cependant, l'ensemble des faces $\mathcal{F}$ peut être limité à la zone où la convection domine. Ce sera le cas dans l'une des applications que nous présenterons au chapitre 10 et cela permettra de réduire considérablement le coût d'assemblage de ce terme supplémentaire.

3.2 Discrétisation en temps

Nous allons maintenant approcher les dérivées en temps en utilisant des schémas implicites nommés *backward differentiation formulation* (BDF). Ces schémas peuvent être écrits pour un ordre q arbitraire et ils sont ainsi notés par BDF_q. Nous présenterons par la suite des schémas BDF_q jusqu'à l'ordre 4. Soit Δt le pas de temps supposé constant au cours du temps, nous avons $t_0 = t_i$ et $t_n = t_0 + n\Delta t$.

q	β_{-1}	β_0	β_1	β_2	β_3
1	1	1			
2	3/2	2	-1/2		
3	11/6	3	-3/2	1/3	
4	25/12	4	-3	4/3	-1/4

TABLE 4.1 – Coefficients des schémas BDF_q jusqu'à l'ordre 4.

La formule suivante décrit l'approximation de la dérivée en temps d'ordre q de la vitesse à l'aide des coefficients β_j décrits dans la table 4.1 :

$$\left.\frac{\partial \boldsymbol{u}_{f,\delta}^{n+1}}{\partial t}\right|_{\mathbf{x}^*} \approx \left(\frac{\beta_{-1}}{\Delta t}\hat{\boldsymbol{u}}_{f,\delta}^{n+1} - \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\hat{\boldsymbol{u}}_{f,\delta}^{n-j}\right) \circ \left(\mathcal{A}_\delta^{t_{n+1}}\right)^{-1} \tag{4.22}$$

$$= \frac{\beta_{-1}}{\Delta t}\boldsymbol{u}_{f,\delta}^{n+1} - \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\hat{\boldsymbol{u}}_{f,\delta}^{n-j} \circ \left(\mathcal{A}_\delta^{t_{n+1}}\right)^{-1} \tag{4.23}$$

Cette expression est composée de deux termes. Le premier contient l'inconnu $\boldsymbol{u}_{f,\delta}^{n+1}$. Le second fait apparaitre les solutions des pas de temps précédents, c'est-à-dire qu'elles sont connues. On remarque aussi que le domaine de définition des solutions précédentes a été transporté sur $\Omega_{f,\delta}^{n+1}$ grâce aux cartes ALE $\left\{\mathcal{A}_\delta^{t_{n+1-k}}, 0 \leq k \leq q\right\}$. Ainsi, nous pouvons ajouter au terme source $\boldsymbol{f}_f^{t_{n+1}}$ cette deuxième partie. On définit alors :

$$\tilde{\boldsymbol{f}}_f^{t_{n+1}} = \boldsymbol{f}_f^{t_{n+1}} + \sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\hat{\boldsymbol{u}}_{f,\delta}^{n-j} \circ \left(\mathcal{A}_\delta^{t_{n+1}}\right)^{-1} \tag{4.24}$$

La vitesse de déformation du maillage est elle aussi approchée à l'aide d'un schéma BDF_q. Nous rappelons tout d'abord la définition de cette vitesse :

$$\boldsymbol{w}_{f,\delta}^{n+1} = \frac{\partial \mathcal{A}_\delta^{t_{n+1}}}{\partial t} \circ \left(\mathcal{A}_\delta^{t_{n+1}}\right)^{-1} \tag{4.25}$$

Nous pouvons ensuite illustrer une approximation faite en utilisant un schéma BDF_2, ce qui nous donne la formule suivante :

$$\boldsymbol{w}_{f,\delta}^{n+1} \approx \frac{1}{\Delta t}\left(\frac{3}{2}\mathcal{A}_\delta^{t_{n+1}} - 2\mathcal{A}_\delta^{t_n} + \frac{1}{2}\mathcal{A}_\delta^{t_{n-1}}\right) \circ \left(\mathcal{A}_\delta^{t_{n+1}}\right)^{-1} \tag{4.26}$$

Nous terminons la discrétisation temporelle en ajoutant les conditions initiales nécessaires pour pouvoir correctement construire les dérivées temporelles. Si l'on utilise un schéma BDF_1, une seule condition initiale est nécessaire $\boldsymbol{u}_{f,\delta}^0 = \boldsymbol{u}_0$ dans $\Omega_{f,\delta}^0$. On peut ensuite calculer les solutions au pas de temps suivant t_n pour $n > 1$. Lorsque que l'on utilise un schéma q d'ordre supérieur à 1, nous avons deux possibilités. Soit on utilise q condition initiale, ce qui représente la solution la plus propre mathématiquement, mais ces informations ne sont pas toujours accessibles dans la pratique. Soit on incrémente progressivement l'ordre du schéma, c'est-à-dire pour le premier pas de temps on utilise un schéma BDF_1 qui a été initialisé avec une seule condition initiale $\boldsymbol{u}_{f,\delta}^0$, puis pour le

deuxième pas de temps on utilise BDF_2 jusqu'à arriver jusqu'à l'ordre désiré. Cependant, cette méthode peut présenter le désavantage de créer des discontinuités à ce démarrage si les phénomènes simulés sont importants.

Finalement, la formulation variationnelle discrète de notre problème type (sans les termes de stabilisations) est donnée par le problème suivant :

Trouver $(\boldsymbol{u}_{f,\delta}^{n+1}, p_{f,\delta}^{n+1}) \in V_\delta^{t_{n+1}} \times Q_\delta^{t_{n+1}}$ tel que $\forall (\boldsymbol{v}, q) \in W_\delta^{t_{n+1}} \times Q_\delta^{t_{n+1}}$:

$$\rho_f \frac{\beta_{-1}}{\Delta t} \int_{\Omega_f^{t_{n+1}}} \boldsymbol{u}_{f,\delta}^{n+1} \cdot \boldsymbol{v} + \rho_f \int_{\Omega_f^{t_{n+1}}} \left(\left(\boldsymbol{u}_{f,\delta}^{n+1} - \boldsymbol{w}_{f,\delta}^{n+1} \right) \cdot \nabla_{\mathbf{x}} \right) \boldsymbol{u}_{f,\delta}^{n+1} \cdot \boldsymbol{v} + \int_{\Omega_f^{t_{n+1}}} \boldsymbol{\sigma}_{f,\delta}^{n+1} : \nabla_{\mathbf{x}} \boldsymbol{v} = \int_{\Omega_f^{t_{n+1}}} \tilde{\boldsymbol{f}}_f^{t_{n+1}} \cdot \boldsymbol{v} + \int_{\Gamma_{N,\delta}^{t_{n+1}}} \boldsymbol{g}_N^{t_{n+1}} \cdot \boldsymbol{v} \tag{4.27}$$

$$\int_{\Omega_f^{t_{n+1}}} q \, \nabla_{\mathbf{x}} \cdot \boldsymbol{u}_{f,\delta}^{n+1} = 0 \tag{4.28}$$

4 Vérifications numériques

Nous allons maintenant présenter un test numérique assez classique appelé *Geometric Conservation Law* (GCL). Le principe de cette vérification est de pouvoir conserver une solution constante au cours du temps malgré les déformations du domaine et sans terme source. Bien évidemment, les conditions aux limites doivent être appropriées et la solution initiale égale à la solution constante. La GCL est vérifiée lorsque la solution discrète n'est pas affectée par le mouvement du domaine. En effet, au niveau continu, la formulation ALE 4.14-4.15 satisfait la GCL mais ce n'est pas forcément le cas au niveau discret. La GCL a été analysée dans [123, 61, 62] pour le cas éléments finis. On y trouve des propriétés de stabilité numérique pour plusieurs schémas temporels, en particulier pour les schémas BDF_q.

Nous allons commencer par un test 3D. Nous choisissons comme solution constante, la solution de Poiseuille dans un cylindre. Ce cylindre a une longueur de 5 et un rayon de 1. La solution de Poiseuille dans un tel contexte est :

$$\boldsymbol{u}^*(x,y,z) = (1 - y^2 - z^2, 0, 0)^T \qquad et \qquad p^*(x,y,z) = -4x + 20 \tag{4.29}$$

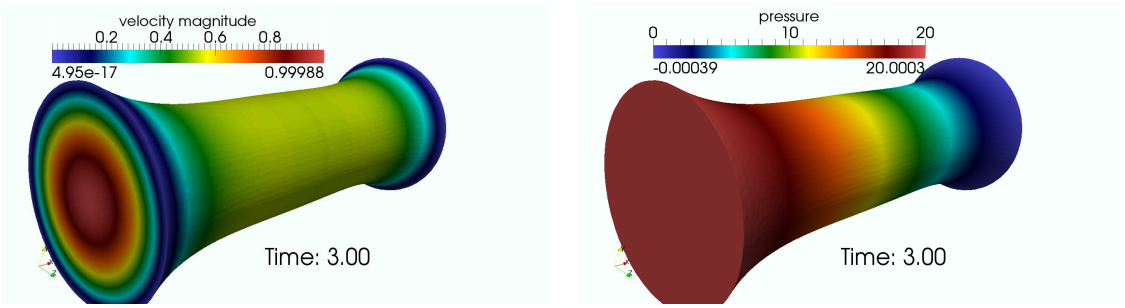
Nous voulons aussi rendre mobile ce cylindre. Les deux disques aux extrémités sont considérés comme fixes, seul le bord courbe subira un déplacement. Le déplacement imposé est $\boldsymbol{d}$, il s'effectuera progressivement au cours du temps. comme le montre la prochaine expression suivante :

$$\boldsymbol{d}(x,y,z,t) = 0.01 \left((x - 2.5)^2 + 5 \right) x \, (5 - x) \left(f(t) \chi(t \in [1,3]) + \chi(t > 3) \right) \boldsymbol{n} \tag{4.30}$$

où nous choisissons de prendre pour f, le facteur dépendant du temps, la même expression que l'on peut trouver dans [124].

$$\begin{aligned} f(t) = \quad & -0.15625t^7 + 2.1875t^6 + 12.46875t^5 + 37.1875t^4 - 62.34375t^3 \\ & +59.0625t^2 - 29.53125t + 6.0625 \end{aligned}$$

Cette fonction est l'unique polynôme de degrés 7 vérifiant $f(1) = 0, f(3) = 1, f^{(k)}(1) = f^{(k)}(3) = 0, k = 1,2,3$. Les figures 4.1(a) et 4.1(b) montrent le résultat obtenu pour le déplacement maximal au temps $t = 3$.



(a) Vitesse (b) Pression

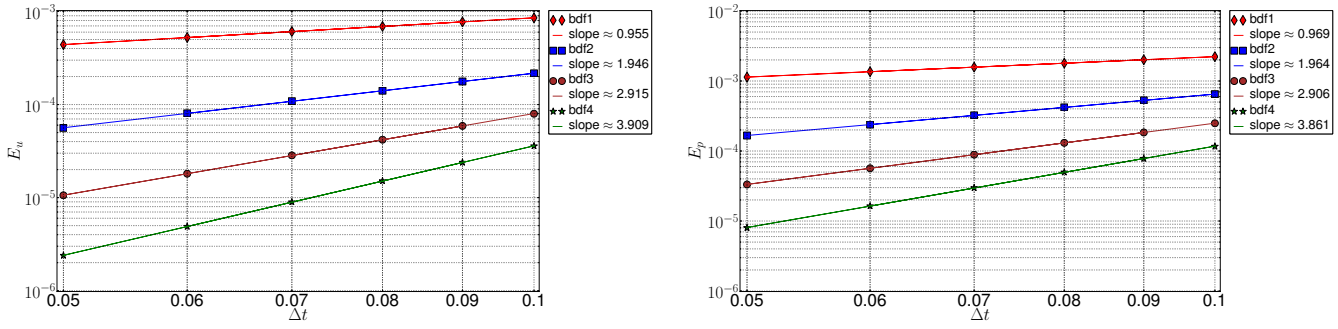
FIGURE 4.1 – Exemple de solution calculée.

Les erreurs sur la vitesse et pression sont décrites ci-dessous. Elles représentent l'accumulation des erreurs L^2 au cours du temps. Celles-ci sont redimensionnées par le pas de temps Δt.

$$E_u = \left(\Delta t \sum_{n=0}^{N_T} \|\boldsymbol{u}^* - \boldsymbol{u}_{f,\delta}^n\|^2_{L^2\left(\Omega_{f,\delta}^{t_n}\right)} \right)^{\frac{1}{2}} \tag{4.31}$$

$$E_p = \left(\Delta t \sum_{n=0}^{N_T} \|p^* - p_{f,\delta}^n\|^2_{L^2\left(\Omega_{f,\delta}^{t_n}\right)} \right)^{\frac{1}{2}} \tag{4.32}$$

Nous avons représenté avec les figures 4.2(a) et 4.2(b) les courbes de convergence associées aux erreurs décrites précédemment en fonction du pas de temps Δt. On observe que la convergence optimale est atteinte pour les quatre schémas BDF.



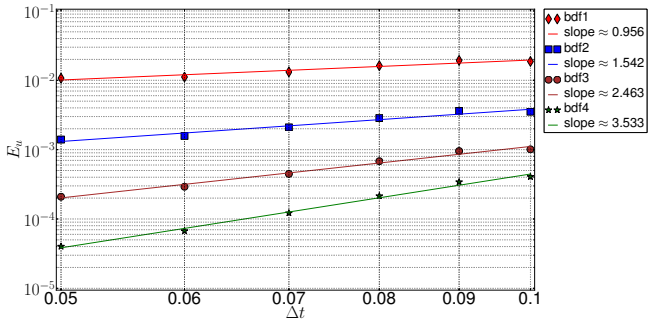
(a) Erreur sur la vitesse (b) Erreur sur la pression

FIGURE 4.2 – Erreur E_u et E_p pour le cas 3D ($N_{geo} = 2$)

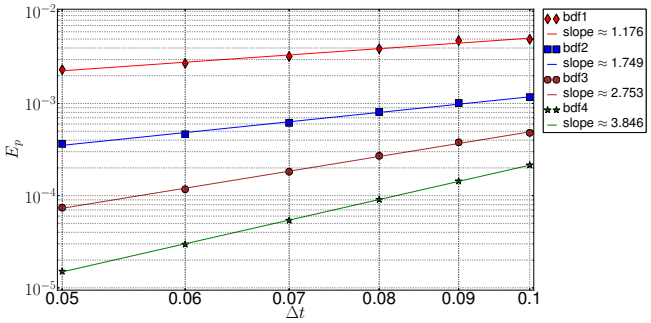
Maintenant, nous voulons tester la stabilisation CIP que nous avons présentée dans ce chapitre. Cette fois-ci, nous faisons une expérimentation 2D. Le domaine initial est un rectangle sur $(0, 5) \times (-1, 1)$. Le déplacement, très similaire, mais légèrement supérieur au cas test 3d est donné par :

$$\boldsymbol{d}(x, y, z, t) = 0.015 \left((x - 2.5)^2 + 5 \right) x \, (5 - x) \left(f(t)\chi(t \in [1, 3]) + \chi(t > 3) \right) \boldsymbol{n} \tag{4.33}$$

Les coefficients du modèle de Navier-Stokes sont $\rho_f = 1$ et $\mu_f = 10^{-3}$. Nous utiliserons des éléments géométriques d'ordre 2 pour discrétiser le domaine. Tout d'abord, nous effectuons un test sans le terme de stabilisation et comme le montrent les figures 4.3(a) et 4.3(b). On observe que la convergence optimale n'est pas atteinte à partir du schéma d'ordre 2. Le phénomène de convection dominante apparait, il est liée au maillage relativement grossier et au paramètre de l'équation. On peut aussi remarquer que la convergence se détèriore un peu plus pour la vitesse que pour la pression.



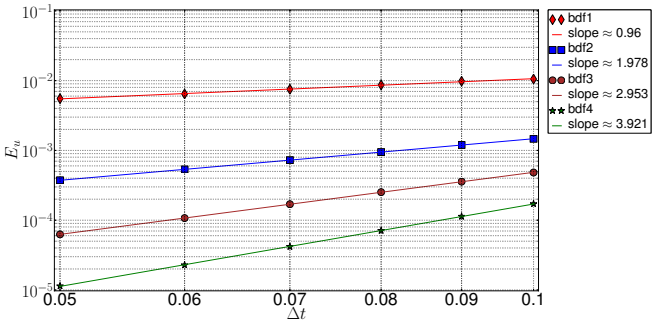
(a) Erreur sur la vitesse



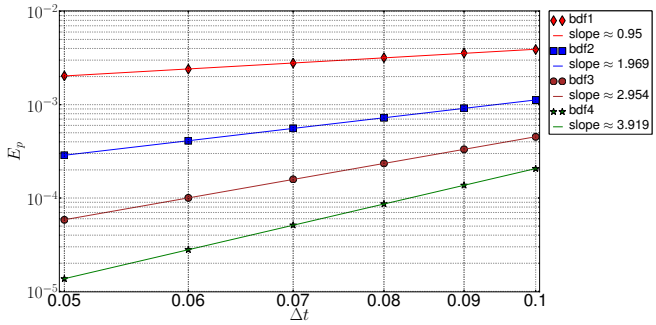
(b) Erreur sur la pression

FIGURE 4.3 – Erreur E_u et E_p pour le cas 2D sans stabilisation ($N_{geo} = 2$)

Enfin, nous recommençons ce test avec le terme de stabilisation (4.21). Le paramètre γ est pris égal à 0.1. L'ensemble des faces $\mathcal{F}$ est choisi comme l'ensemble des faces internes avec aussi les faces de la frontière Neumann. Cette fois, on retrouve parfaitement les ordres de convergence. Les résulats sont tracés sur les figures 4.4(a) et 4.4(b)



(a) Erreur sur la vitesse



(b) Erreur sur la pression

FIGURE 4.4 – Erreur E_u et E_p pour le cas 2D avec la stabilisation CIP ($N_{geo} = 2$)

5 Modèles spécifiques aux écoulements sanguins

Nous allons présenter dans cette dernière partie quelques particularités qui peuvent être nécessaires pour nos simulations. Nous commencerons par exposer une technique permettant de mieux prendre en compte la condition de sortie du sang dans un vaisseau. Cette condition sera notamment utilisée dans le chapitre 10. Nous terminerons en décrivant brièvement quelques modèles de sang prenant en compte une propriété complexe du sang, le comportement rhéofluidifiant.

5.1 Modèle de Windkessel comme condition aux limites

Dans le contexte des écoulements sanguins, la définition des conditions aux limites réaliste en entrée et sortie d'un écoulement n'est pas simple à mettre en oeuvre. Une condition couramment utilisée pour décrire une sortie d'écoulement est de supposer que le tenseur des contraintes normales est nul (i.e. $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0}$). Ce type de condition modélise la sortie de l'écoulement se déversant dans un grand réservoir. Ce n'est pas le cas pour notre modélisation. Ces conditions génèrent un phénomène de réflexion des ondes de pression en sortie. Plusieurs solutions existent pour modéliser une sortie d'écoulement plus réaliste. Elles permettent de prendre en compte la continuité du vaisseau sanguin. Certains auteurs [121, 59, 123, 60] proposent un couplage de l'écoulement fluide (2D ou 3D) avec un modèle fluide réduit (1D) en sortie. On trouve également des couplages avec des modèles 0d basés sur les équations de Windkessel [21, 59] ou sur des conditions aux limites *défectueuses* [63]. Dans chacun des cas, le couplage entre le modèle 2D ou 3D et le modèle réduit peut-être effectué de manière faible ou forte.

Pour nos applications, nous avons parfois utilisé le modèle de Windkessel 3-éléments comme conditions aux limites de sortie. On suppose avoir n_{out} sortie dans notre écoulement que l'on note $\Gamma_f^{out,l}$ avec $1 \leq l \leq n_{out}$ Pour chacune de ces sorties, nous définissons une pression P_l et un débit Q_l. Ces quantités doivent alors vérifier les systèmes d'équations différentielles suivants :

$$\begin{cases} C_{d,l} \dfrac{\partial \pi_l}{\partial t} + \dfrac{\pi_l}{R_{d,l}} = Q_l \\ P_l = R_{p,l} Q_l + \pi_l \end{cases} \tag{4.34}$$

Les coefficients $R_{p,l}$ and $R_{d,l}$ représentent respectivement la résistance proximale et distale. La dernière constante $C_{d,l}$ correspond à la capacitance du vaisseau sanguin. Les valeurs P_l et π_l sont aussi appelées pression proximale et distale, respectivement.

Il nous faut maintenant définir les conditions de couplage entre la sortie du modèle fluide et le modèle réduit de Windkessel. Ce couplage 3D-0D ou 2D-0D est alors décrit par les deux relations suivantes :

$$Q_l = \int_{\Gamma_l} \boldsymbol{u}_f \cdot \boldsymbol{n}_f \tag{4.35}$$

$$\boldsymbol{\sigma}_f \boldsymbol{n}_f = -P_l \boldsymbol{n}_f \tag{4.36}$$

Il permet de relier le débit Q_l et la pression proximale P_l du modèle réduit aux inconnues $\boldsymbol{u}_f$ et p_f du modèle fluide.

Nous passons maintenant à la discrétisation en temps du problème (4.34). Pour cela, nous utilisons encore un schéma BDF_q pour la dérivée en temps. Soit $\{t_n\}_{1 \leq n \leq N_T}$ un

échantillonnage de l'intervalle de temps d'étude. Nous notons alors Q_l^n, π_l^n et P_l^n le débit, la pression distale et la pression proximale aux temps t_n. En utilisant les notations présentées à la section 3.2 pour ce schéma, nous obtenons :

$$\begin{cases} C_{d,l}\dfrac{\beta_{-1}}{\Delta t}\pi_l^{n+1} + \dfrac{\pi_l^{n+1}}{R_{d,l}} = Q_l^{n+1} + \displaystyle\sum_{j=0}^{q-1}\frac{\beta_j}{\Delta t}\pi_l^{n-j} \\ P_l^{n+1} = R_{p,l}Q_l^{n+1} + \pi_l^{n+1} \end{cases} \tag{4.37}$$

En regroupant certains termes du système d'équation précédent, ce dernier peut-être reformulé par :

$$\begin{cases} \pi_l^{n+1} = \alpha_l \displaystyle\sum_{j=0}^{q-1}\beta_j\pi_l^{n-j} + \gamma_l Q_l^{n+1} \\ P_l^{n+1} = \kappa_l Q_l^{n+1} + \alpha_l \displaystyle\sum_{j=0}^{q-1}\beta_j\pi_l^{n-j} \end{cases} \tag{4.38}$$

$$\xi_l = R_{d,l}C_{d,l}\beta_{-1} + \Delta t \quad , \quad \alpha_l = \frac{R_{d,l}C_{d,l}}{\xi_l} \quad , \quad \gamma_l = \frac{R_{d,l}\Delta t}{\xi_l} \quad , \quad \kappa_l = \frac{R_{p,l}\xi_l + R_{d,l}\Delta t}{\xi_l}$$

Nous terminons cette partie avec la description de la méthode de couplage que nous avons utilisée. Nous avons choisi de réaliser un couplage faible pour sa simplicité d'implémentation. Ainsi, les conditions de couplage (4.35)-(4.36) ne seront pas vérifiées exactement. Pour cela, nous allons utiliser une technique d'extrapolation du champ de vitesse $\boldsymbol{u}_f^{n+1}$. On pose $\tilde{\boldsymbol{u}}_f^{n+1}$ l'extrapolation de ce champ de vitesse. Le débit Q_l^{n+1} peut-être ensuite approché en utilisant $\tilde{\boldsymbol{u}}_f^{n+1}$ dans l'équation (4.35). Enfin, nous présentons l'algorithme 3 qui représente les principales étapes effectuées pour la prise en compte de ces conditions aux limites connectées au modèle 0d dans le modèle fluide.

Algorithm 3 Couplage faible entre le modèle fluide et le modèle de Windkessel

Calcul de $\tilde{\boldsymbol{u}}_f^{n+1}$ par extrapolation de $\boldsymbol{u}_f^{n+1}$
Approximation de Q_l^{n+1} avec $\tilde{\boldsymbol{u}}_f^{n+1}$
Calcul de π_l^{n+1} et P_l^{n+1} en utilisant (4.38)
Résolution du modèle fluide en utilisant la condition (4.36)

5.2 Extension aux fluides non newtoniens

Dans cette partie, nous allons présenter rapidement quelques modèles de fluides, dis non newtoniens, qui peuvent être nécessaires pour la simulation des écoulements sanguins. Cependant, nous ne les avons pas expérimentés d'un point de vue numérique. Nous nous sommes limités au cas des fluides newtoniens avec le modèle de Navier-Stokes.

D'après certaines publications comme [47], le sang peut être considéré comme un fluide newtonien dans les vaisseaux de diamètre assez large. C'est le cas pour les principales artères. Cependant, lorsque cette taille diminue ($< 1mm,$), cette hypothèse n'est plus vraie. En effet, lorsque le diamètre de ces vaisseaux se rapproche de la taille du globule rouge, des modifications dans les propriétés physiques de l'écoulement apparaissent. Par

exemple, la viscosité du fluide n'est plus constante. Ces phénomènes sont principalement dus au transport des globules rouges dans le plasma. Ces derniers doivent se regrouper et subir de grandes déformations pour pouvoir circuler dans le plasma et à l'intérieur de ces petits vaisseaux. Pour décrire ces comportements complexes du sang, il existe plusieurs types de modèles qui sont regroupés dans la catégorie des fluides non newtoniens.

Dans ce contexte, le sang est souvent modélisé comme un fluide rhéofluidifiant. Ce comportement peut être décrit par le principe suivant : lorsque la viscosité baisse, le taux de cisaillement augmente. La viscosité n'est donc pas une constante comme dans le cas newtonien, mais c'est une fonction de tenseur des contraintes $\boldsymbol{D}(\boldsymbol{u}_f)$. Selon [8], nous pouvons introduire une métrique du taux de déformation noté par $\dot{\gamma}$:

$$\dot{\gamma} \equiv \sqrt{2\ tr\left(\boldsymbol{D}(\boldsymbol{u}_f)^2\right)}$$

et de cette manière, nous représentons la viscosité μ_f comme une fonction dépendant de $\dot{\gamma}$. Le tenseur des contraintes déviatoriques $\boldsymbol{\tau}$ est ensuite obtenu à l'aide du modèle de fluide newtonien généralisé qui se caractérise par la forme suivante :

$$\boldsymbol{\tau} = 2\mu_f(\dot{\gamma})\,\boldsymbol{D}(\boldsymbol{u}_f)$$

Cette loi permet de définir une grande variété de modèles de fluide. En particulier, le fluide est dit newtonien lorsque μ_f est une constante.

L'exemple le plus simple de modèle non newtonien est le modèle d'Ostwald ou la loi puissance qui a une viscosité donnée par :

$$\mu_f(\dot{\gamma}) = k\dot{\gamma}^{n-1}$$

où k et $n < 1$ sont deux paramètres dépendants des caractéristiques biologiques du sang. Pour décrire certains modèles non newtoniens, nous pouvons utiliser la fonction Φ qui est définie par la relation suivante :

$$\Phi(\dot{\gamma}, \mu_\infty, \mu_0) = \frac{\mu(\dot{\gamma}) - \mu_\infty}{\mu_0 - \mu_\infty}$$

avec μ_0 est la viscosité pour un cisaillement nul et μ_∞ est la viscosité pour un cisaillement infini. Les modèles les plus couramment utilisés pour la modélisation du sang sont résumés dans la table 4.2 et s'expriment par l'intermédiaire de la fonction Φ. Les constantes matérielles présentes sont issues de [59]. Elles correspondent à des valeurs réalistes pour les écoulements sanguins.

Modèle	$\Phi(\dot{\gamma}, \mu_\infty, \mu_0)$	Constantes matérielles pour le sang
Powell-Eyring	$\frac{sinh^{-1}(\lambda\dot{\gamma})}{\lambda\dot{\gamma}}$	$\lambda = 5.383s$
Cross	$\frac{1}{1+(\lambda\dot{\gamma})^m}$	$\lambda = 1.007s$, $m = 1.028$
Carreau-Yasuda	$(1+(\lambda\dot{\gamma})^a)^{(n-1)/a}$	$\lambda = 1.902s$, $n = 0.22$, $a = 1.25$

TABLE 4.2 – Quelques exemples de modèles non newtoniens pour le sang avec $\mu_0 = 0.056$ Pa.s and $\mu_\infty = 0.00345$ Pa.s

Nous pouvons aussi prendre en compte d'autres propriétés présentes dans les écoulements sanguins comme la viscoélasticité et la thixotropie. La viscoélasticité permet de modéliser un fluide qui est non seulement visqueux, mais qu'il est aussi élastique. Autrement dit, le fluide possède une mémoire. La réponse du fluide à une déformation présente alors à la fois un aspect visqueux (où les contraintes sont proportionnelles aux vitesses de déformation) et un aspect élastique (où les contraintes sont proportionnelles aux déformations). Enfin, la thixotropie permet de prendre en compte des phénomènes très complexes comme le passage de l'état liquide à l'état solide et inversement.

Chapitre 5

Modèles de mécanique des solides

La structure et les propriétés d'élasticité des vaisseaux sanguins peuvent jouer un rôle important dans la modélisation des écoulements sanguins. Par exemple, une des propriétés fondamentales des grosses artères élastiques et notamment de l'aorte est de réguler le débit cardiaque [111, 18, 114]. Le débit discontinu pulsé du sang est transformé en un débit continu grâce à la nature élastique de la paroi vasculaire. On appelle ce phénomène l'effet Windkessel. On peut distinguer deux types de vaisseaux dans le corps, les artères et les veines, qui ont une structure assez similaire. La paroi de ces vaisseaux sanguins est composée de trois couches : l'intima, la média et l'adventice. Les propriétés mécaniques des différentes couches constituant les artères dépendent de leur quantité de fibres d'élastine qui varie avec le diamètre du vaisseau. Dans certains vaisseaux, les déformations engendrées peuvent être assez importantes (plus de 10 % du rayon) [101]. C'est pourquoi les modèles de grande déformation, dits modèles hyperélastiques, sont privilégiés. On considère aussi que les vaisseaux sanguins sont de nature anisotrope à cause de cette structure composite. Plusieurs modèles hyperélastiques anisotropes peuvent être trouvés dans [114]. Une autre propriété qui est démontrée dans [140] est la viscoélasticité de la paroi artérielle. Les déformations subies à un instant t dépendent des déformations antérieures. On pourra trouver dans [88] une description complète des modèles mécaniques utilisés pour la modélisation de la paroi artérielle. Cependant, dans les modèles utilisés dans cette thèse, nous négligerons les caractéristiques anisotropes et viscoélastiques des vaisseaux. Il est également constaté dans [105, 55] que les globules rouges sont composés d'une membrane élastique et incompressible. Ces cellules peuvent subir de très grandes déformations lorsqu'elles voyagent à travers les différents capillaires. En effet, le diamètre des capillaires est plus petit que la taille des globules rouges.

L'objectif de ce chapitre est de présenter dans un contexte très général les modèles que nous avons développés dans cette thèse. Nous commencerons par présenter les bases des modèles d'élasticités. Nous passerons après à la description de quelques lois de comportement des matériaux hyperélastiques isotropes. Ces lois permettent de caractériser le comportement des matériaux et le niveau de déformabilité qu'ils peuvent subir. Nous aborderons les cas compressibles, incompressibles et quasi-incompressibles. Ces lois peuvent être utilisées dans la mécanique des vaisseaux sanguins si l'on néglige leurs caractères anisotropes et viscoélastiques. Ces deux premières parties ont été écrites en partie en s'inspirant des ouvrages [49, 43]. Ensuite, nous présenterons les formulations variationnelles et les discrétisations numériques que nous utilisons dans la pratique. Nous illustrons ces modèles par des applications numériques. Elles ne décriront pas des applications liées

aux écoulements sanguins. Leur intérêt est de mettre à l'épreuve notre implémentation numérique des modèles de structure sous diverses contraintes. Enfin, nous terminerons par la description d'un modèle axisymétrique réduit qui sera utilisé au chapitre 10.

1 Cinématique du solide

Soit Ω_s^* le domaine d'étude appelé domaine de référence ou configuration initiale. On note Ω_s^t le domaine déformé à l'instant t. On désigne par $\mathbf{x}$ la position d'une particule dans le milieu actuel déformé et $\mathbf{x}^*$ la position de cette même particule dans la configuration initiale. Ces deux points de vue caractérisent les repères lagrangiens et eulériens, et ils sont reliés par la déformation $\boldsymbol{\varphi}_s^t$ définie par l'application suivante :

$$\begin{array}{rcl} \boldsymbol{\varphi}_s^t : \Omega_s^* & \longrightarrow & \Omega_s^t \\ \mathbf{x}^* & \longmapsto & \mathbf{x} = \boldsymbol{\varphi}_s^t\left(\mathbf{x}^*\right) \end{array}$$

Cette application est supposée inversible et permet d'avoir $\Omega_s^t = \boldsymbol{\varphi}_s^t\left(\Omega_s^*\right)$. Par la suite, nous allons nous placer uniquement dans le repère lagrangien pour modéliser les matériaux solides. Ce repère est mieux adapté pour décrire la mécanique des solides. C'est pourquoi, nous avons besoin de définir la fonction déplacement $\boldsymbol{\eta}_s(\mathbf{x}^*, t) = \boldsymbol{\varphi}_s^t(\mathbf{x}^*) - \mathbf{x}^*$. L'ensemble de ces notations est représenté avec la figure 5.1.

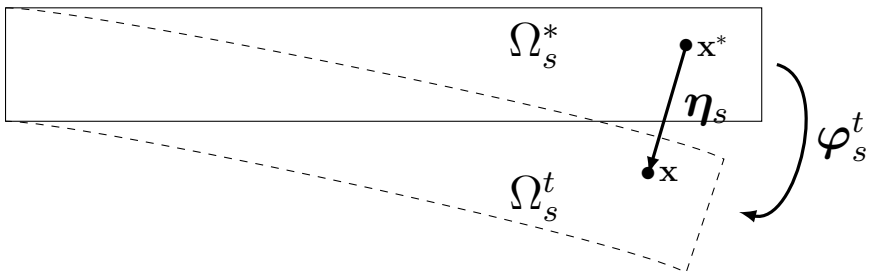


FIGURE 5.1 – Représentation des repères lagrangiens et eulériens.

Pour mesurer les déformations du solide, on introduit tout d'abord le gradient de déformation défini par :

$$\boldsymbol{F}_s = \boldsymbol{F}_s(\mathbf{x}^*, t) = \frac{\partial \boldsymbol{\varphi}_s^t}{\partial \mathbf{x}^*} = \boldsymbol{I} + \nabla \boldsymbol{\eta}_s$$

Ce tenseur n'étant pas invariant dans un changement de repère, une mesure plus objective de la déformation serait le tenseur de Cauchy-Green ou tenseur des dilatations :

$$\boldsymbol{C}_s = \boldsymbol{F}_s^T \boldsymbol{F}_s$$

Ce tenseur est symétrique et il a la propriété d'être invariant par translation et rotation, c'est-à-dire dans un changement de repère de référence. Cependant, un autre tenseur est utilisé, c'est le tenseur de Green-Lagrange :

$$\boldsymbol{E}_s = \frac{1}{2}\left(\boldsymbol{C}_s - \boldsymbol{I}\right)$$

Il possède les mêmes propriétés que le tenseur précédent, mais il a la caractéristique de se décomposer en deux termes :

$$\boldsymbol{E}_s = \underbrace{\frac{1}{2}\left(\nabla\boldsymbol{\eta}_s + (\nabla\boldsymbol{\eta}_s)^T\right)}_{\boldsymbol{\epsilon}_s} + \underbrace{\frac{1}{2}\left((\nabla\boldsymbol{\eta}_s)^T\,\nabla\boldsymbol{\eta}_s\right)}_{\boldsymbol{\gamma}_s}$$

– $\boldsymbol{\epsilon}_s$ est linéaire par rapport au déplacement $\boldsymbol{\eta}_s$
– $\boldsymbol{\gamma}_s$ est quadratique par rapport au déplacement $\boldsymbol{\eta}_s$

On parle alors d'hypothèse des petites perturbations lorsque l'on néglige le terme quadratique $\boldsymbol{\gamma}_s$. Ce cadre reste valable lorsque les déformations et les déplacements restent petits, typiquement inférieurs à 10 % de la taille de l'objet.

On pose ρ_s^* la masse volumique donnée dans le domaine de référence au temps initial. Pour pouvoir établir la relation fondamentale de la dynamique, nous avons également besoin d'exprimer deux quantités définies sur le domaine déformé à un instant t. Ce sont la vitesse de déformation $\boldsymbol{u}_s = \dot{\boldsymbol{\eta}}_s \circ (\boldsymbol{\varphi}_s^t)^{-1}$ et la masse volumique ρ_s.

Le principe fondamental de la dynamique est issu de la 2ème loi de Newton qui stipule que : si un corps subit un ensemble de forces extérieures $\boldsymbol{F}_i$, alors la variation de la quantité de mouvement par unité de temps est égale à la somme des forces $\boldsymbol{F}_i$. $\forall\omega^t \in \Omega_s^t$, les forces extérieures agissant sur ω^t sont d'une part des forces massiques de densité volumique $\rho_s\boldsymbol{F}_{vol}^t$ et des actions de contact de densité $\boldsymbol{F}_{cont}^t$. Il en résulte alors l'équation de la conservation de la quantité de mouvement :

$$\forall\omega^* \in \Omega_s^*\ ,\ \omega^t = \boldsymbol{\varphi}_s^t\left(\omega^*\right)\ ,\quad \frac{d}{dt}\int_{\omega^t}\rho_s\boldsymbol{u}_s dx = \int_{\omega^t}\rho_s\boldsymbol{F}_{vol}^t dx + \int_{\partial\omega^t}\boldsymbol{F}_{cont}^t dS \tag{5.1}$$

Le théorème de Cauchy nous permet d'affirmer l'existence d'un champ de tenseur $\boldsymbol{\sigma}_s$, appelé tenseur des contraintes de Cauchy, tel que $\boldsymbol{F}_{cont}^t = \boldsymbol{\sigma}_s\boldsymbol{n}_s$. Ce qui conduit à :

$$\forall\omega^* \in \Omega_s^*\ ,\ \omega^t = \boldsymbol{\varphi}_s^t\left(\omega^*\right)\ ,\quad \frac{d}{dt}\int_{\omega^t}\rho_s\boldsymbol{u}_s dx = \int_{\omega^t}\rho_s\boldsymbol{F}_{vol}^t dx + \int_{\partial\omega^t}\boldsymbol{\sigma}_s\boldsymbol{n}_s dS \tag{5.2}$$

Nous allons maintenant transformer les équations dans le repère lagrangien grâce au changement de variable $\mathbf{x} = \boldsymbol{\varphi}_s^t(\mathbf{x}^*)$. Tout d'abord, commençons par écrire l'expression lagrangienne de la conservation de la masse :

$$\forall\omega^* \in \Omega_s^*\ ,\ \omega^t = \boldsymbol{\varphi}_s^t\left(\omega^*\right)\ ,\quad \int_{\omega^t}\rho_s\ d\mathbf{x} = \int_{\omega^*}\rho_s^*\ d\mathbf{x}^* \tag{5.3}$$

Grâce à ce changement de variable et au jacobien $J_s = |\det\left(\boldsymbol{F}_s\right)|$, nous pouvons obtenir :

$$\forall\omega^* \in \Omega_s^*\ ,\quad \int_{\omega^*}\rho_s\circ\boldsymbol{\varphi}_s^t\ J_s\ d\mathbf{x}^* = \int_{\omega^*}\rho_s^*\ d\mathbf{x}^* \quad\Longleftrightarrow\quad J_s = \frac{\rho_s^*}{\rho_s\circ\boldsymbol{\varphi}_s^t} \tag{5.4}$$

A l'aide de cette relation et en posant $\hat{\boldsymbol{\sigma}}_s = \boldsymbol{\sigma}_s\circ\boldsymbol{\varphi}_s^t$ et $\hat{\boldsymbol{F}}_{vol}^t = \boldsymbol{F}_{vol}^t\circ\boldsymbol{\varphi}_s^t$, l'équation (5.2) devient après le changement de variable :

$$\forall\omega^* \in \Omega_s^*\ ,\quad \frac{d}{dt}\int_{\omega^*}\rho_s^*\dot{\boldsymbol{\eta}}_s\ d\mathbf{x}^* = \int_{\omega^*}\rho_s^*\hat{\boldsymbol{F}}_{vol}^t\ d\mathbf{x}^* + \int_{\partial\omega^*}J_s\hat{\boldsymbol{\sigma}}_s\boldsymbol{F}_s^{-T}\boldsymbol{n}_s\ dS^* \tag{5.5}$$

Le premier tenseur des contraintes de Piola-Kirchhoff $\boldsymbol{\Pi}_s = J_s \hat{\boldsymbol{\sigma}}_s \boldsymbol{F}_s^{-T}$ apparaît, mais ce tenseur n'est pas symétrique. Nous préférons alors définir le second tenseur de Piola-Kirchhoff $\boldsymbol{\Sigma}_s$ définit par :

$$\boldsymbol{\Sigma}_s = J_s \boldsymbol{F}_s^{-1} \hat{\boldsymbol{\sigma}}_s \boldsymbol{F}_s^{-T} = \boldsymbol{F}_s^{-1} \boldsymbol{\Pi}_s$$

À l'aide de la conservation de la masse, nous pouvons aussi démontrer la relation suivante :

$$\forall \omega^* \in \Omega_s^* \ , \quad \frac{d}{dt} \int_{\omega^*} \rho_s^* \dot{\boldsymbol{\eta}}_s \ d\mathbf{x}^* = \int_{\omega^*} \rho_s^* \frac{d}{dt} \dot{\boldsymbol{\eta}}_s \ d\mathbf{x}^* = \int_{\omega^*} \rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} \ d\mathbf{x}^* \tag{5.6}$$

En écrivant $\boldsymbol{\Pi}_s = \boldsymbol{F}_s \boldsymbol{\Sigma}_s$, nous obtenons alors :

$$\forall \omega^* \in \Omega_s^* \ , \quad \int_{\omega^*} \rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} \ d\mathbf{x}^* = \int_{\omega^*} \rho_s^* \hat{\boldsymbol{F}}_{vol}^t \ d\mathbf{x}^* + \int_{\partial \omega^*} \boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s^* \ dS^* \tag{5.7}$$

En supposant que $\boldsymbol{F}_s \boldsymbol{\Sigma}_s$ possède des dérivées partielles intégrables, on peut appliquer le théorème de la divergence sur l'intégrable de bord qui apparait dans l'équation précédente, ce qui donne :

$$\forall \omega^* \in \Omega_s^* \ , \quad \int_{\omega^*} \left[\rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \rho_s^* \hat{\boldsymbol{F}}_{vol}^t - \nabla \cdot (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) \right] \ d\mathbf{x}^* = 0 \tag{5.8}$$

Finalement, comme l'équation (5.8) est vrai $\forall \omega^* \in \Omega_s^*$, nous obtenons en posant $\boldsymbol{f}_s^t = \rho_s^* \hat{\boldsymbol{F}}_{vol}^t$ la relation fondamentale de la dynamique du solide dans la configuration initiale :

$$\rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \nabla \cdot (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) = \boldsymbol{f}_s^t \tag{5.9}$$

2 Lois de comportement pour les matériaux hyperélastiques

Il est ensuite nécessaire d'ajouter une loi de comportement à notre modèle, car celui-ci possède plus d'inconnues que d'équations. La loi de comportement permet de relier le tenseur des contraintes au tenseur des déformations en tout point du solide. Cette loi de comportement doit de plus vérifier le principe d'objectivité qui s'énonce par : la loi de comportement doit être invariante dans tout changement de référentiel.

L'élasticité linéaire : Nous commençons tout d'abord par décrire brièvement le comportement de l'élasticité linéaire en petite déformation. On dit qu'un matériau est élastique si le tenseur des contraintes à l'instant t dépend uniquement de l'état de déformation à ce même instant. La relation peut alors s'écrire $\boldsymbol{\Sigma}_s = \mathbb{C} : \boldsymbol{E}_s$ avec $\mathbb{C}$ un tenseur d'ordre 4. On peut remarquer que $\mathbb{C}$ est un tenseur symétrique car $\boldsymbol{\Sigma}_s$ et $\boldsymbol{E}_s$ le sont, ce qui limite à 36 le nombre de coefficients de ce tenseur. De plus, si le matériau est isotrope, on obtient la loi de Hooke qui ne dépend que de deux coefficients :

$$\boldsymbol{\Sigma}_s = \lambda_s tr\left(\boldsymbol{\epsilon}_s\right) \boldsymbol{I} + 2\mu_s \boldsymbol{\epsilon}_s \ \text{,où } \boldsymbol{\epsilon}_s \text{ est la partie linéaire du tenseur } \boldsymbol{E}_s$$

L'hyperélasticité : Un matériau élastique est dit hyperélastique si le tenseur des contraintes dérive d'une fonction d'énergie du matériau. Ceci implique que le travail mis en jeu pour aller d'un état de déformation à un autre ne dépend pas du chemin suivi, mais uniquement de l'état initial et final. Pour écrire une loi de comportement hyperélastique, on postule ainsi l'existence d'une énergie libre Ψ.

Notre point de départ est l'inégalité de Clausius Duhem qui est obtenue en utilisant le premier et le second principe de la thermodynamique. Elle se formule en description lagrangienne par :

$$\Phi_0 = \underbrace{\boldsymbol{\Sigma}_s \dot{\boldsymbol{E}}_s - \rho_s^* \left(\dot{\Psi} + s\dot{T} \right)}_{\Phi_{int}} - \underbrace{\frac{\boldsymbol{Q} \cdot \nabla T}{T}}_{\Phi_{th}} \geq 0$$

avec :

- Φ_0 est la dissipation, qui peut se décomposer en une dissipation intrinsèque mécanique, et une dissipation thermique.
- Ψ est l'énergie libre massique
- $\boldsymbol{Q}$ est le flux de chaleur
- s est l'entropie

L'énergie libre (de Helmotz) est définie par : $\Psi = e - Ts$, avec e la densité massique d'énergie interne. Pour respecter le principe d'objectivité énoncé plus haut, l'énergie interne et l'entropie sont considérées comme une fonction du tenseur des déformations $\boldsymbol{E}_s$. Si de plus le matériau est isotrope, on pourra les exprimer en fonction des invariants du tenseur C. Elles dépendent également de la température T et ainsi il en va de même pour l'énergie libre, $\Psi = \Psi\left(\boldsymbol{E}_s, T\right)$.

Pour simplifier notre loi de comportement, nous allons négliger les effets thermiques. De plus, les matériaux élastiques ont un comportement réversible, ce qui se traduit comme une dissipation intrinsèque nulle et une dissipation thermique nulle. D'où la relation :

$$\boldsymbol{\Sigma}_s \dot{\boldsymbol{E}}_s - \rho_s^* \dot{\Psi} = \left(\boldsymbol{\Sigma}_s - \rho_s^* \frac{\partial \Psi}{\partial \boldsymbol{E}_s} \right) : \dot{\boldsymbol{E}}_s = 0 \quad \Longleftrightarrow \quad \boldsymbol{\Sigma}_s = \rho_s^* \frac{\partial \Psi}{\partial \boldsymbol{E}_s}$$

On désigne par $\mathcal{W}$ l'énergie de déformation par unité de volume dans la configuration initiale définie par :

$$\mathcal{W} = \rho_s^* \Psi \tag{5.10}$$

On obtient ainsi la forme générale de la loi de comportement hyperélastique écrit dans le repère lagrangien :

$$\boldsymbol{\Sigma}_s = \frac{\partial \mathcal{W}}{\partial \boldsymbol{E}_s} = 2 \frac{\partial \mathcal{W}}{\partial \boldsymbol{C}_s}$$

En supposant que le matériau est isotrope, l'énergie libre s'exprime également en fonction des invariants I_1, I_2 et I_3 du tenseur C. Les invariants d'un tenseur C sont les coefficients du polynôme caractéristique $\det\left(\boldsymbol{C}_s - \lambda \boldsymbol{I}\right)$ et le calcul nous donne :

$$I_1 = tr\left(\boldsymbol{C}_s\right), \quad I_2 = \frac{1}{2}\left(tr\left(\boldsymbol{C}_s\right)^2 - tr(\boldsymbol{C}_s^2)\right), \quad I_3 = \det(\boldsymbol{C}_s)$$

De plus, la dérivation de ces invariants par rapport à $\boldsymbol{C}_s$ conduit à :

$$\frac{\partial I_1}{\partial \boldsymbol{C}_s} = \boldsymbol{I} \quad , \quad \frac{\partial I_2}{\partial \boldsymbol{C}_s} = I_1 \boldsymbol{I} - \boldsymbol{C}_s \quad , \quad \frac{\partial I_3}{\partial \boldsymbol{C}_s} = I_3 \boldsymbol{C}_s^{-1} \tag{5.11}$$

On obtient finalement la relation entre contraintes et déformations :

$$\boldsymbol{\Sigma}_s = 2\rho_s^* \frac{\partial \Psi}{\partial \boldsymbol{C}_s} = 2\frac{\partial \mathcal{W}}{\partial \boldsymbol{C}_s} = 2\sum_{i=1}^{3} \frac{\partial \mathcal{W}}{\partial I_i}\frac{\partial I_i}{\partial \boldsymbol{C}_s} \tag{5.12}$$

$$= 2\left[\left(\frac{\partial \mathcal{W}}{\partial I_1} + I_1\frac{\partial \mathcal{W}}{\partial I_2}\right)\boldsymbol{I} - \frac{\partial \mathcal{W}}{\partial I_2}\boldsymbol{C}_s + I_3\frac{\partial \mathcal{W}}{\partial I_3}\boldsymbol{C}_s^{-1}\right] \tag{5.13}$$

Une autre approche fréquemment utilisée pour des milieux isotropes et popularisée par Ogden, consiste à considérer les élongations principales λ_1, λ_2 et λ_3 du tenseur des dilatations $\boldsymbol{C}_s$. Elles correspondent aux valeurs propres du tenseur des déformations pures. Nous pouvons alors écrire les invariants de $\boldsymbol{C}_s$ en fonction des élongations :

$$\begin{aligned} I_1 &= \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \\ I_2 &= \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2 \\ I_3 &= \lambda_1^2\lambda_2^2\lambda_3^2 \end{aligned}$$

Ainsi, l'énergie de déformation $\mathcal{W}$ peut être définie en fonction des élongations principales. Finalement, le second tenseur de Piola-Kirchhoff peut s'exprimer par :

$$\boldsymbol{\Sigma}_s = \sum_{i=1}^{3} \frac{1}{\lambda_i}\frac{\partial \mathcal{W}}{\partial \lambda_i}(\boldsymbol{N}_i \otimes \boldsymbol{N}_i) \tag{5.14}$$

où les $\boldsymbol{N}_i$ sont les vecteurs propres associés à chaque valeur propre λ_i.

L'expression mathématique de l'énergie de déformation $\mathcal{W}$ est ensuite déterminée pour reproduire au mieux les réponses du matériau obtenues sous des sollicitations variées. Dans ce but, $\mathcal{W}$ peut être élaborée à partir de résultats expérimentaux, de considérations statistiques, microstructurales, ... Nous allons maintenant présenter les deux modèles hyperélastiques compressibles les plus simples. Nous commençons par le modèle hyperélastique de Saint-Venant-Kirchhoff. L'énergie de déformation est alors donnée par :

$$\mathcal{W} = \frac{\lambda_s}{2}\left(tr\left(\boldsymbol{E}_s\right)\right)^2 + \mu_s\ \boldsymbol{E}_s : \boldsymbol{E}_s \tag{5.15}$$

Après dérivation, nous obtenons l'expression du second tenseur de Piola-Kirchhoff :

$$\boldsymbol{\Sigma}_s = \lambda_s tr\left(\boldsymbol{E}_s\right)\boldsymbol{I} + 2\mu_s\boldsymbol{E}_s \tag{5.16}$$

Nous citons également le modèle néo-hookéen compressible, présenté dans [81, 27]. Il se caractérise par la formule suivante :

$$\mathcal{W} = \frac{\mu_s}{2}\left(I_1 - 3 - 2\ln(J_s)\right) + \frac{\lambda_s}{2}\left(\ln(J_s)\right)^2$$

où le symbol ln désigne le logarithme népérien. Et finalement, nous obtenons la formule du second tenseur de Piola-Kirchhoff :

$$\boldsymbol{\Sigma}_s = \mu_s\left(\boldsymbol{I} - \boldsymbol{C}_s^{-1}\right) + \lambda_s\left(\ln(J_s)\right)\boldsymbol{C}_s^{-1} \tag{5.17}$$

Ces deux exemples de loi de comportement restent relativement simples. Elles ne permettent pas de modéliser de très grandes déformations, mais elles semblent suffisantes pour les déformations des écoulements sanguins. Cependant, les propriétés des artères supposent d'autres hypothèses comme l'incompressibilité. Il est donc nécessaire d'enrichir les modèles avec cette caractéristique.

2.1 Matériaux incompressibles

Nous voulons maintenant prendre en compte une hypothèse d'incompressibilité des matériaux. Cela signifie que la masse volumique est constante au cours du temps, c'est-à-dire que $\rho_s^* = \rho_s \circ \boldsymbol{\varphi}_s^t$ à chaque instant t. En utilisant la relation (5.4), nous obtenons la contrainte $J_s = det\,(\boldsymbol{F}_s) = 1$.

Pour assurer cette contrainte, la condition d'incompressibilité est introduite directement dans l'écriture de l'énergie de déformation en écrivant celle-ci sous la forme :

$$\tilde{\mathcal{W}} = \mathcal{W} - p_s(J_s - 1) \tag{5.18}$$

où p_s joue le rôle d'un multiplicateur de Lagrange. L'énergie de déformation $\mathcal{W}$ ne dépend plus de l'invariant I_3 car on peut remarquer :

$$I_3 = det(\boldsymbol{C}_s) = J_s^2 = 1 \tag{5.19}$$

Ainsi $\tilde{\mathcal{W}}$ dépend de I_1, I_2 et J. Nous pouvons aussi dire que le multiplicateur de Lagrange p_s peut être identifié à une pression hydrostatique. Par dérivation de l'énergie de déformation, on obtient alors :

$$\begin{aligned}\boldsymbol{\Sigma}_s &= 2\frac{\partial \tilde{\mathcal{W}}}{\partial \boldsymbol{C}_s} = 2\frac{\partial \mathcal{W}}{\partial \boldsymbol{C}_s} - p_s J_s \boldsymbol{C}^{-1} = 2\sum_{i=1}^{2} \frac{\partial \mathcal{W}}{\partial I_i}\frac{\partial I_i}{\partial \boldsymbol{C}_s} - p_s J_s \boldsymbol{C}^{-1} \\ &= 2\left[\left(\frac{\partial \mathcal{W}}{\partial I_1} + I_1 \frac{\partial \mathcal{W}}{\partial I_2}\right)\boldsymbol{I} - \frac{\partial \mathcal{W}}{\partial I_2}\boldsymbol{C}_s\right] - p_s J_s \boldsymbol{C}^{-1}\end{aligned}$$

Différentes lois de comportement isotropes peuvent être trouvées dans la littérature [43, 96, 160]. Ces lois sont classiquement utilisées pour étudier le comportement de matériaux caoutchoutiques, ainsi que de certains tissus biologiques comme la paroi artérielle.

Modèle de Rivlin

Le modèle de Rivlin décrit l'énergie de déformation en une série de polynômes :

$$\mathcal{W}(I_1, I_2) = \sum_{i,j=0}^{\infty} C_{ij}\,(I_1 - 3)^i\,(I_2 - 3)^j \tag{5.20}$$

avec $C_{00} = 0$ afin que l'énergie de déformation s'annule lorsque le matériau n'est pas sollicité. Trois modèles classiques découlent de ce modèle de Rivlin :

- le modèle néo-hookéen : $\mathcal{W}(I_1, I_2) = C_{10}\,(I_1 - 3)$ avec $C_{10} = \mu_s/2$
- le modèle de Mooney-Rivlin : $\mathcal{W}(I_1, I_2) = C_{10}\,(I_1 - 3) + C_{01}\,(I_2 - 3)$
- le modèle de Yeoh : $\mathcal{W}(I_1, I_2) = C_{10}\,(I_1 - 3) + C_{20}\,(I_1 - 3)^2 + C_{30}\,(I_1 - 3)^3$

Modèle de Ogden

Dans le modèle de Ogden, l'énergie de déformation est écrite en fonction des élongations principales λ_1, λ_2 et λ_3 :

$$\mathcal{W}\,(\lambda_1, \lambda_2, \lambda_3) = \sum_{n=1}^{N} \frac{\mu_n}{\alpha_n}\,(\lambda_1^{\alpha_n} + \lambda_2^{\alpha_n} + \lambda_3^{\alpha_n} - 3)$$

où les doublets de réels (μ_n, α_n) sont des constantes matérielles et doivent vérifier $\mu_n \alpha_n > 0, \forall n$. On peut montrer que les modèles néo-hookéen et de Mooney-Rivlin sont également des cas particuliers du modèle d'Ogden. Ce modèle est particulièrement adapté aux très grandes déformations.

2.2 Matériaux quasi-incompressibles

L'hypothèse d'incompressibilité reste une approximation souvent justifiée par les valeurs élevées du module de compressibilité de ces matériaux par rapport à leur rigidité en cisaillement. En réalité, ces matériaux sont qualifiables de faiblement compressibles ou quasi-incompressibles. Par ailleurs, cette propriété est souvent mise à profit sur le plan numérique, pour relâcher la contrainte sévère que constitue la condition d'incompressibilité dans la résolution des problèmes d'équilibre en hyperélasticité. Une prise en compte de cette faible compressibilité nécessite rigoureusement une reformulation de la loi de comportement en décomposant les tenseurs $\boldsymbol{F}_s$ et $\boldsymbol{C}_s$ en une partie sphérique et une partie isochore :

$$\tilde{\boldsymbol{F}}_s = J_s^{-1/3} \boldsymbol{F}_s \quad \text{et} \quad \tilde{\boldsymbol{C}}_s = J_s^{-2/3} \boldsymbol{C}_s \tag{5.21}$$

Le tenseur des dilations de Cauchy isochore $\tilde{\boldsymbol{C}}_s$ vérifie la relation :

$$\frac{\partial \tilde{\boldsymbol{C}}_s}{\partial \boldsymbol{C}_s} = J_s^{-2/3} \left(\mathbb{I} - \frac{1}{3} \boldsymbol{C}_s \otimes \boldsymbol{C}_s^{-1} \right) \tag{5.22}$$

On introduit alors les invariants réduits, invariants des tenseurs $\tilde{\boldsymbol{F}}_s$ et $\tilde{\boldsymbol{C}}_s$, reliés aux invariants I_1 et I_2 de la manière suivante :

$$\tilde{I}_1 = J_s^{-2/3} I_1 \quad \text{et} \quad \tilde{I}_2 = J_s^{-4/3} I_2 \tag{5.23}$$

La densité d'énergie de déformation peut être ensuite découplée en une partie sphérique $\mathcal{W}_S$ et une partie déviatorique $\mathcal{W}_D$:

$$\mathcal{W}(\tilde{I}_1, \tilde{I}_2, J_s) = \mathcal{W}_D(\tilde{I}_1, \tilde{I}_2) + \mathcal{W}_S(J_s) \tag{5.24}$$

On obtient alors la loi de comportement hyperélastique quasi-incompressible :

$$\boldsymbol{\Sigma}_s = 2 \frac{\partial \mathcal{W}}{\partial \boldsymbol{C}_s} = 2 \frac{\partial \mathcal{W}_D}{\partial \tilde{\boldsymbol{C}}_s} : \frac{\partial \tilde{\boldsymbol{C}}_s}{\partial \boldsymbol{C}_s} + 2 \frac{d\mathcal{W}_S}{dJ_s} \frac{\partial J_s}{\partial \boldsymbol{C}_s} \tag{5.25}$$

$$= 2 J_s^{-2/3} \frac{\partial \mathcal{W}_D}{\partial \tilde{\boldsymbol{C}}_s} : \left(\mathbb{I} - \frac{1}{3} \boldsymbol{C}_s \otimes \boldsymbol{C}_s^{-1} \right) + J_s \frac{d\mathcal{W}_S}{dJ_s} \boldsymbol{C}_s^{-1} \tag{5.26}$$

Pour expliciter les lois hyperélastiques compressibles, l'expression de l'énergie de déformation déviatorique $\mathcal{W}_D$, fonction des invariants réduits $(\tilde{I}_1, \tilde{I}_2)$, est généralement choisie sous la même forme que pour les matériaux incompressibles. On retrouve donc les modèles présentés dans la partie précédente. Pour la densité d'énergie sphérique $\mathcal{W}_S$, un grand nombre de modèles existe. Ils font intervenir une constante matérielle assimilable au module de compressibilité κ et parfois un paramètre empirique β. Nous pouvons commencer par citer la forme la plus simple :

$$\mathcal{W}_S(J_s) = \frac{\kappa}{2} (J_s - 1)^2$$

D'autres fonctions énoncées dans [104] s'expriment de la manière suivante :

$$\mathcal{W}_S(J_s) = \frac{\kappa}{4}\left((J_s-1)^2 + (ln(J_s))^2\right) \qquad [\text{ SIMO-TAYLOR (1982)}]$$

$$\mathcal{W}_S(J_s) = \frac{\kappa}{2}\left(ln(J_s)\right) \qquad [\text{ SIMO (1985)}]$$

$$\mathcal{W}_S(J_s) = \frac{\kappa}{\beta^2}\left(\beta ln(J_s) + J_s^{-\beta} - 1\right) \qquad [\text{ OGDEN (1972)}]$$

$$\mathcal{W}_S(J_s) = \frac{\kappa}{2}\left(J_s^2 - 1 - 2ln(J_s)\right) \qquad [\text{ SIMO-TAYLOR (1982)}]$$

3 Approximation éléments finis

Nous allons maintenant établir la formulation variationnelle d'un problème d'hyperélasticité pour les cas compressibles et incompressibles. Nous présenterons ensuite les discrétisations éléments finis associées à ces formulations. Dans ce but, nous considérerons un problème de référence composé de certaines conditions aux limites standards. On désigne toujours le domaine de référence $\Omega_s^* \subset \mathbb{R}^d$. On suppose que la frontière $\partial\Omega_s^*$ est décomposée en deux parties Γ_D^* et Γ_N^*. Notre intervalle de temps d'étude sera compris entre t_i le temps initial et t_f le temps final.

3.1 Formulations variationnelles

Nous commençons par rappeler la forme générique des modèles d'hyperélasticité. Nous avons les modèles compressibles représentés par la recherche du déplacement $\boldsymbol{\eta}_s$ vérifiant l'équation suivante :

$$\rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \nabla \cdot (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) = \boldsymbol{f}_s^t, \quad \text{dans } \Omega_s^* \times [t_i, t_f] \tag{5.27}$$

avec $\boldsymbol{\Sigma}_s = 2\frac{\partial \mathcal{W}}{\partial \boldsymbol{C}_s}$ et $\mathcal{W}$ correspondant à une loi de comportement compressible ou quasi-incompressible présentée précédemment.

Pour les modèles incompressibles, nous devons ajouter une contrainte supplémentaire pour assurer l'incompressibilité des matériaux. La pression p_s apparait comme une inconnue supplémentaire. On obtient alors le modèle suivant :

$$\rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \nabla \cdot (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) = \boldsymbol{f}_s^t, \quad \text{dans } \Omega_s^* \times [t_i, t_f] \tag{5.28}$$

$$J_s = 1, \quad \text{dans } \Omega_s^* \times [t_i, t_f] \tag{5.29}$$

avec $\boldsymbol{\Sigma}_s = 2\frac{\partial \mathcal{W}}{\partial \boldsymbol{C}_s} - p_s J_s \boldsymbol{C}^{-1}$ et $\mathcal{W}$ une loi de comportement.

Pour pouvoir établir les formulations variationnelles, nous devons compléter ces modèles de conditions aux limites. De nombreux types de conditions aux limites existent, nous avons choisi de prendre deux types de conditions très classiques et qui interviendront dans nos problèmes d'interaction fluide structure. La première correspond à un déplacement imposé et la seconde représente l'action d'une force normale à la frontière.

$$\boldsymbol{\eta}_s^* = \boldsymbol{g}_D^t \text{ sur } \Gamma_D^* \tag{5.30}$$

$$(\boldsymbol{F}_s \boldsymbol{\Sigma}_s)\, \boldsymbol{n}_s^* = \boldsymbol{g}_N^t \text{ sur } \Gamma_N^* \tag{5.31}$$

Nous présentons maintenant les espaces de fonctions continus définissant les inconnus des modèles d'hyperélasticité. Nous définissons $V_s = \mathbf{H}^1_{\boldsymbol{g}_D,\Gamma^*_D}(\Omega^*_s)$ l'espace associé au déplacement et $Q_s = L^2\left(\Omega^*_f\right)$ l'espace associé à la pression. Nous posons également $W_s = \mathbf{H}^1_{0,\Gamma^*_D}$. Ainsi, en multipliant l'équation (5.27) par $\boldsymbol{\xi} \in W_s$, en intégrant sur Ω^*_s et en faisant une intégration par partie sur le terme qui contient le tenseur des contraintes, on obtient :

$$\int_{\Omega^*_s} \rho^*_s \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} \cdot \boldsymbol{\xi} + \int_{\Omega^*_s} (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) : \nabla \boldsymbol{\xi} - \int_{\partial\Omega^*_s} (\boldsymbol{F}_s \boldsymbol{\Sigma}_s)\, \boldsymbol{n}^*_s \cdot \boldsymbol{\xi} = \int_{\Omega^*_s} \boldsymbol{f}^t_s \cdot \boldsymbol{\xi} \tag{5.32}$$

Nous rajoutons ensuite les conditions aux limites à l'équation précédente. Nous obtenons alors la formulation variationnelle pour un modèle compressible ou quasi-incompressible avec le problème suivant :

Trouver $\boldsymbol{\eta}_s \in V_s$ tel que $\forall \boldsymbol{\xi} \in W_s$:

$$\int_{\Omega^*_s} \rho^*_s \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} \cdot \boldsymbol{\xi} + \int_{\Omega^*_s} (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) : \nabla \boldsymbol{\xi} = \int_{\Omega^*_s} \boldsymbol{f}^t_s \cdot \boldsymbol{\xi} + \int_{\Gamma^*_N} \boldsymbol{g}^t_N \cdot \boldsymbol{\xi} \tag{5.33}$$

Pour le modèle incompressible, nous appliquons la même procédure décrite précédemment à l'équation (5.28). Pour l'équation (5.29), nous la multiplions par $q \in W_s$ et nous l'intégrons sur Ω^*_s. Ainsi, le problème suivant nous donne la formulation variationnelle pour le modèle incompressible :

Trouver $(\boldsymbol{\eta}_s, p_s) \in V_s \times Q_s$ tel que $\forall (\boldsymbol{\xi}, q) \in W_s \times Q_s$:

$$\int_{\Omega^*_s} \rho^*_s \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} \cdot \boldsymbol{\xi} + \int_{\Omega^*_s} (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) : \nabla \boldsymbol{\xi} = \int_{\Omega^*_s} \boldsymbol{f}^t_s \cdot \boldsymbol{\xi} + \int_{\Gamma^*_N} \boldsymbol{g}^t_{s,N} \cdot \boldsymbol{\xi} \tag{5.34}$$

$$\int_{\Omega^*_s} (J_s - 1)\ q = 0 \tag{5.35}$$

3.2 Discrétisations numériques

Nous considérons $\mathcal{T}^*_{s,\delta}$ le maillage représentant une discrétisation du domaine Ω^*_s. Cette discrétisation recouvre un domaine de calcul noté $\Omega^*_{s,\delta}$. On note également $\Gamma^*_{\delta,D}$ (resp $\Gamma^*_{\delta,N}$) la frontière discrète représentant Γ^*_D (resp Γ^*_N). Nous définissons maintenant les espaces d'approximations $V_{s,\delta}$ pour le déplacement et $Q_{s,\delta}$ pour la pression dans le modèle incompressible :

$$V_{s,\delta} = \mathbf{H}^1_{\boldsymbol{g}^t_D,\Gamma^*_D}(\Omega^*_s) \cap \left[P^M_c\left(\Omega^*_{s,\delta}\right)\right]^d \tag{5.36}$$

$$Q_{s,\delta} = P^N_c\left(\Omega^*_{s,\delta}\right) \tag{5.37}$$

Dans le cas incompressible, le choix des ordres polynomiaux se fait de la même manière que pour les équations de Navier-Stokes. Ce choix est basé sur les éléments de Taylor-Hood et il permet ainsi d'assurer la condition de compatibilité présentée avec l'équation (4.16). Les ordres polynomiaux M sont alors reliés par $N = M - 1$. Nous avons également besoin de l'espace des fonctions tests pour le déplacement donné par :

$$W_{s,\delta} = \mathbf{H}^1_{0,\Gamma^*_D}(\Omega^*_s) \cap \left[P^M_c\left(\Omega^*_{s,\delta}\right)\right]^d \tag{5.38}$$

L'intervalle de temps est échantillonné en N_T pas de temps, notés $\{t_n\}_{1\leq n\leq N_T}$. On note $\boldsymbol{\eta}^n_{s,\delta}$ (resp $p^n_{s,\delta}$) le champ de déplacement (resp pression) discret au temps t_n. Pour la discrétisation du terme d'accélération, nous choisissons le schéma de Newmark présenté dans [156, 97]. Pour chaque pas de temps t_n, nous pouvons exprimer l'accélération et la vitesse de déformation à partir du déplacement au temps t_n et des champs déplacement, vitesse et accélération au temps t_{n-1}. Les équations suivantes expriment ces relations :

$$\ddot{\boldsymbol{\eta}}^{n+1}_{s,\delta} = \frac{1}{\beta\Delta t^2}\left(\boldsymbol{\eta}^{n+1}_{s,\delta} - \boldsymbol{\eta}^{n}_{s,\delta}\right) - \frac{1}{\beta\Delta t}\dot{\boldsymbol{\eta}}^{n}_{s,\delta} - \left(\frac{1}{2\beta} - 1\right)\ddot{\boldsymbol{\eta}}^{n}_{s,\delta} \tag{5.39}$$

$$\dot{\boldsymbol{\eta}}^{n+1}_{s,\delta} = \frac{\gamma}{\beta\Delta t}\left(\boldsymbol{\eta}^{n+1}_{s,\delta} - \boldsymbol{\eta}^{n}_{s,\delta}\right) - \left(\frac{\gamma}{\beta} - 1\right)\dot{\boldsymbol{\eta}}^{n}_{s,\delta} - \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\ddot{\boldsymbol{\eta}}^{n}_{s,\delta} \tag{5.40}$$

Les deux paramètres γ and β sont choisis de manière à obtenir un schéma inconditionnellement stable d'ordre 2. Pour cela, nous pouvons prendre $\gamma = 0.5$ et $\beta = 0.25$ d'après [97].

Enfin, nous pouvons présenter les formulations variationnelles discrètes pour les modèles d'hyperélasticié. Nous présentons uniquement la formulation pour le cas incompressible. Le cas compressible ou quasi-incompressible est identique sauf qu'il n'y a pas l'équation sur la contrainte d'incompressibilité et donc l'inconnue de pression. Pour chaque pas de temps t_{n+1}, le problème variationnel se formule par :

Trouver $\left(\boldsymbol{\eta}^{n+1}_{s,h}, p^{n+1}_{s,\delta}\right) \in V_{s,\delta} \times Q_{s,\delta}$ tel que $\forall\,(\boldsymbol{\xi}, q) \in V_{s,\delta} \times Q_{s,\delta}$:

$$\int_{\Omega^*_{s,\delta}} \rho^*_s \ddot{\boldsymbol{\eta}}^{n+1}_{s,\delta} \cdot \boldsymbol{\xi} + \int_{\Omega^*_{s,\delta}} \left(\boldsymbol{F}^{n+1}_{s,\delta}\boldsymbol{\Sigma}^{n+1}_{s,\delta}\right) : \nabla\boldsymbol{\xi} = \int_{\Omega^*_{s,\delta}} \boldsymbol{f}^{t_{n+1}}_s \cdot \boldsymbol{\xi} + \int_{\Gamma^*_{\delta,N}} \boldsymbol{g}^{t_{n+1}}_N \cdot \boldsymbol{\xi} \tag{5.41}$$

$$\int_{\Omega^*_{s,\delta}} \left(J^{n+1}_{s,\delta} - 1\right)\ q = 0 \tag{5.42}$$

4 Applications numériques

Nous voulons maintenant tester quelques modèles décrits dans les parties précédentes. L'objectif principal est une vérification des algorithmes mis en jeu et une vérification qualitative des solutions. Les résultats pourront aussi être comparés à d'autres librairies numériques de calcul.

La première application représente la simulation d'une plaque en 3D accrochée à un des bords et soumise à la gravité. Elle a été construite à partir d'un benchmark 2D [154] étendu au cas 3D. Ce benchmark 2D est étudié en détail dans le chapitre 9. La deuxième application est un test de déformation d'un solide pour des modèles compressibles et incompressibles.

4.1 Plaque soumise à son propre poids

Ce premier exemple modélise la déformation d'une plaque élastique accrochée uniquement sur un côté. L'état initial de la plaque est un parallélépipède et la seule force agissant sur cet objet sera une force de gravité. La géométrie de la plaque est représentée avec la

figure 5.2. On note Γ_F la surface grisée correspondant à la partie fixe et Γ_L les autres surfaces. Pour cette application, nous avons pris les valeurs des paramètres suivantes :

- $\mathbf{P} = (0.6, 0, 0.19)$
- $(L_x, L_y, L_z) = (0.35101, 0.41, 0.02)$

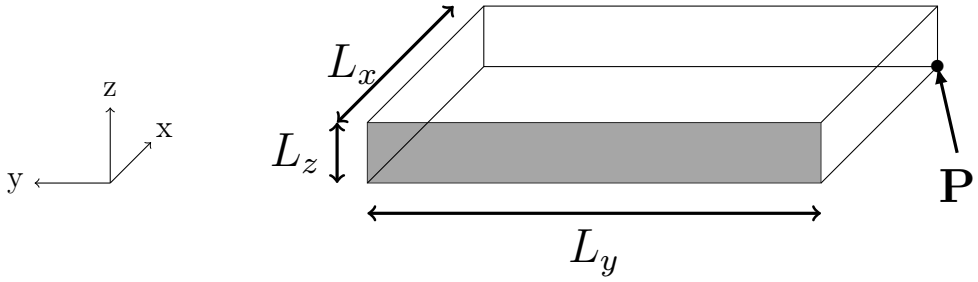


FIGURE 5.2 – Géométrie du problème de plaque soumise à son propre poids

La loi de comportement utilisée pour modéliser cette structure est un modèle de Saint-Venant-Kirchhoff compressible. Le matériau est aussi caractérisé par sa masse volumique $\rho_s^* = 1000$, son module de Young $E_s = 1.4 \cdot 10^6$ et son coefficient de Poisson $\nu_s = 0.4$. À partir de ces deux dernières données, nous pouvons en déduire les deux coefficients de Lamé λ_s et μ_s :

$$\lambda_s = \frac{E_s \nu_s}{(1+\nu_s)(1-2\nu_s)} \quad , \quad \mu_s = \frac{E_s}{2(1+\nu_s)} \tag{5.43}$$

Nous imposons ensuite les conditions aux limites pour ce problème. Nous avons la surface Γ_F qui doit rester fixe et les autres surfaces sont libres de contraintes Γ_L. Cela se traduit par :

- $\boldsymbol{\eta}_s = 0$ sur Γ_F
- $(\boldsymbol{F}_s \boldsymbol{\Sigma}_s)\, \boldsymbol{n}_s^* = \mathbf{0}$ sur Γ_L

Enfin, nous ajoutons le terme source $\boldsymbol{f}_s^t = (0, 0, -\rho_s^* g)$ qui dépend d'une constante de gravité g que nous prenons égale à 2. Ce terme va permettre de mettre la structure en mouvement grâce à l'action de son propre poids.

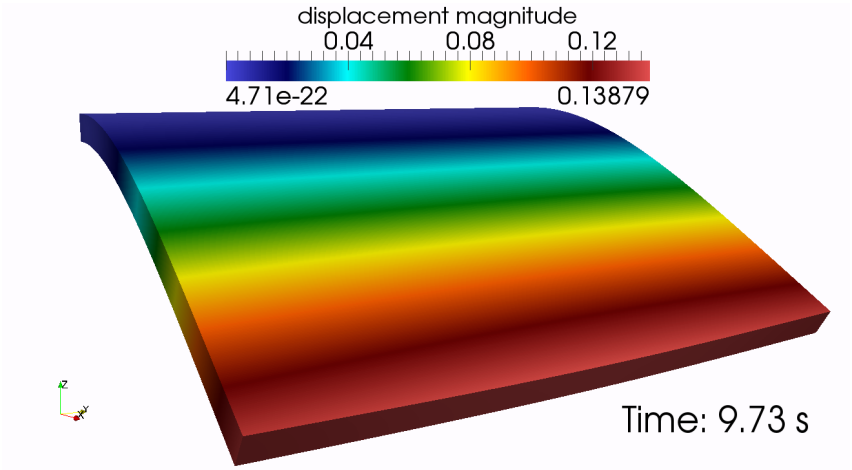
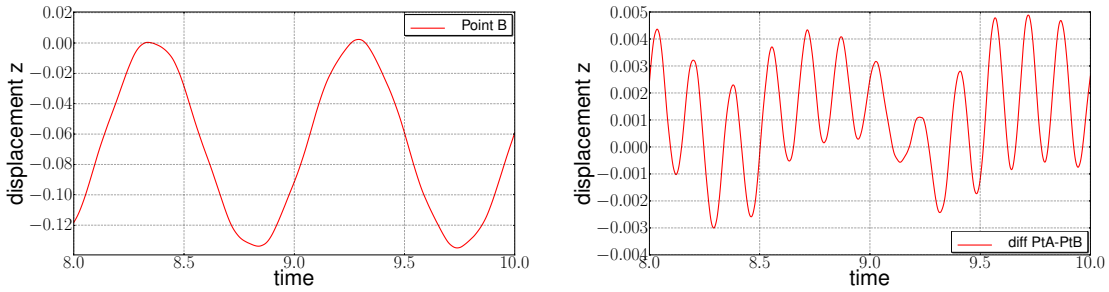


FIGURE 5.3 – Magnitude du déplacement calculé au temps $t = 9.73$.

Nous présentons maintenant les résultats numériques de cette application transitoire. Nous mesurons à chaque pas de temps le déplacement de deux points $\mathbf{A} = (0.6, 0, 0.2)$ et $\mathbf{B} = (0.6, 0.205, 0.2)$. Ces points sont situés sur la surface opposée à la surface fixe. Tout d'abord, nous pouvons voir une illustration du déplacement avec la figure 5.3. Nous avons ensuite le déplacement du point **B** avec la figure 5.4(a). On retrouve le comportement oscillatoire de la plaque qui est très similaire au benchmark 2D présenté [154]. Par contre, dans ce cas 3D, on voit apparaitre de nouvelles sortes d'oscillations, plus rapides en fréquence, mais beaucoup plus petites en amplitude. La différence entre le déplacement en z du point **A** et du point **B** nous permet d'en avoir un aperçu. C'est ce que représente la figure 5.4(b).



(a) Point **B** (b) Différence $\mathbf{A} - \mathbf{B}$

FIGURE 5.4 – Comportement de la plaque au cours du temps à partir des mesures du déplacement en z des points **A** et **B**.

4.2 Torsion d'une barre élastique

Dans ce nouvel exemple, nous allons modéliser la torsion d'une barre à l'aide d'un mouvement circulaire imposé à l'une des extrémités. L'autre extrémité sera fixée. Nous prenons comme domaine de référence un parallélépipède représenté sur la figure 5.5. La surface où la torsion est imposée est notée Γ_T et la surface fixe est notée Γ_F. On pose Γ_L les autres frontières de cette barre. Les valeurs des paramètres géométriques utilisées pour cette modélisation sont :

- $\mathbf{P} = (0, 0, 0)$
- $(L_x, L_y, L_z) = (5, 1, 1)$

Au niveau des caractéristiques du matériau, nous prenons le module de Young $E_s = 10$ et le coefficient de Poissson $\nu_s = 0.3$. Nous utiliserons aussi plusieurs lois de comportement pour cette modélisation tel que Mooney-Rivlin incompressible, Néo-Hookéen incompressible, Néo-Hookéen compressible et Saint-Venant-Kirchhoff compressible. Tous ces modèles dépendent des coefficients de Lamé sauf pour le modèle de Mooney-Rivlin qui dépend de deux constantes matérielles C_{10} et C_{01}. Ces constantes sont généralement déterminées à l'aide de données expérimentales. Par exemple, on peut calculer ces constantes en effectuant une combinaison de tests de tension et de compression sur le matériau. Elles sont aussi reliées par le coefficient de Lamé $\mu_s = 2(C_{10}+C_{01})$. Ainsi, la connaissance d'une seule des constantes est suffisante. On peut trouver dans [58, 17] une approximation de la constante C_{10}. Celle-ci est exprimée en fonction du module de Young : $C_{10} = E_s/6$.

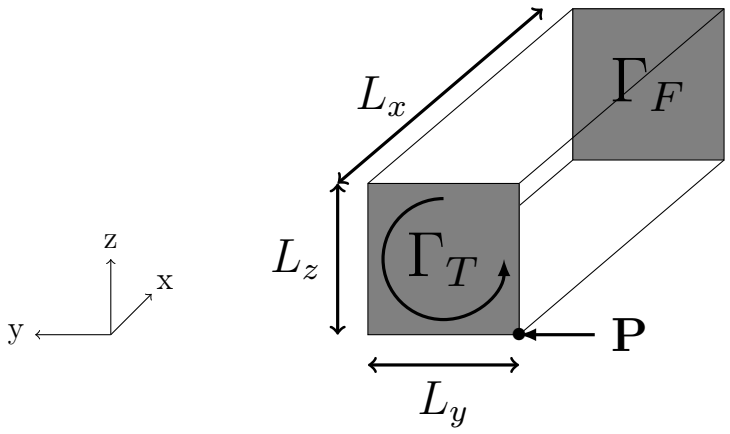


FIGURE 5.5 – Géométrie du problème de torsion de barre.

En ce qui concerne les conditions aux limites, nous avons :
- $\boldsymbol{\eta}_s = \boldsymbol{g}_T$ sur Γ_T
- $\boldsymbol{\eta}_s = \mathbf{0}$ sur Γ_F
- $(\boldsymbol{F}_s \boldsymbol{\Sigma}_s)\, \boldsymbol{n}_s^* = \mathbf{0}$ sur Γ_L

avec $\boldsymbol{g}_T$ le vecteur déplacement assurant une torsion circulaire de la barre d'un angle θ. Ce déplacement défini sur le domaine de référence Ω_s^* est donné par l'expression suivante :

$$\boldsymbol{g}_T(x, y, z, \theta) = \begin{pmatrix} 0 \\ 0.5 + (y - 0.5)\cos(\theta) - (z - 0.5)\sin(\theta) - y \\ 0.5 + (y - 0.5)\cos(\theta) + (z - 0.5)\sin(\theta) - z \end{pmatrix} \tag{5.44}$$

Nous supposons aussi qu'il n'y a pas de forces volumiques, c'est-à-dire que $\boldsymbol{f}_s^t = \mathbf{0}$.

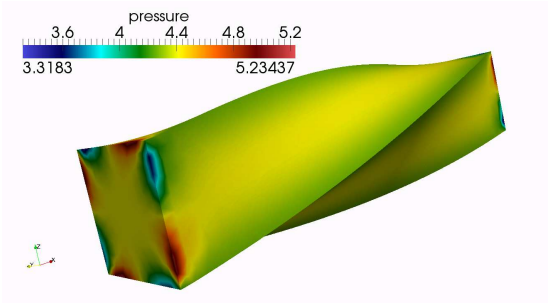
Le but de notre application est de calculer l'état stationnaire de déformation de la barre pour un angle de rotation θ maximal. On cherche ainsi à aller à la limite de la loi de comportement. Cette limite apparaitra comme un déplacement engendrant un maillage non valide ou encore si le solveur numérique échoue. Cependant, il est impossible de réaliser un calcul direct avec un θ relativement grand. Nous devons échelonner le paramètre θ en une suite de paramètre θ_i et ainsi arriver à l'angle de torsion désiré par ces calculs intermédiaires. On considère cette méthode comme une simulation quasi-statique. Pour toutes nos simulations, on considérera la suite θ_i définie par $\theta_i = i/10$.

Nous présentons maintenant les valeurs maximales de θ que nous avons pu calculer pour les différents modèles. C'est valeurs se trouvent dans le tableau suivant :

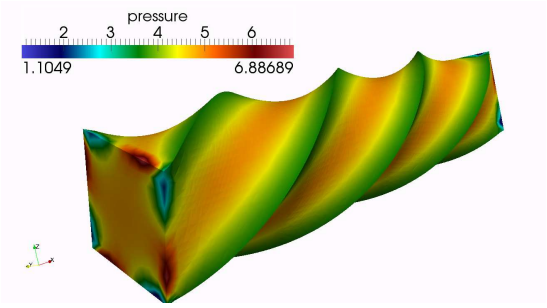
Modèle	θ_{max}	Modèle	θ_{max}
Saint-Venant-Kirchhoff compressible	6.5	Néo-Hookéen incompressible	4.5
Néo-Hookéen compressible	8.6	Mooney-Rivlin incompressible	5.4

On constate que les modèles compressibles permettent d'obtenir des torsions plus fortes. C'est un résultat qui semble naturel, car la compression du matériau permet une

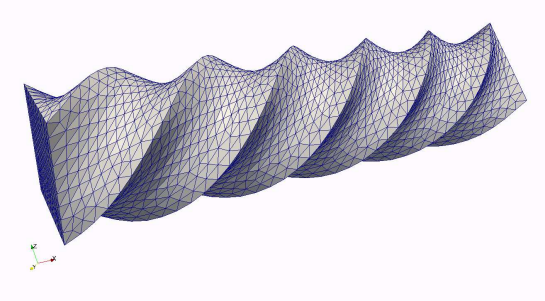
plus grande déformabilité. Comme le prévoit la théorie des lois de comportement, le modèle Néo-Hookéen compressible permet d'obtenir une torsion plus grande que le modèle de Saint-Venant-Kirchhoff. Pour le cas incompressible, on retrouve une meilleure robustesse du modèle de Mooney-Rivilin par rapport au modèle Néo-Hookéen, comme le suppose également la théorie. Les deux figures 5.6(a) et 5.6(b) montrent des résultats obtenus avec le modèle de Mooney-Rivlin et elles illustrent la pression sur le domaine déformé. La figure 5.6(c) montre le modèle Néo-Hookéen compressible à sa torsion maximale. Enfin, nous avons tracé les valeurs de J_s en fonction de θ pour chacun des modèles. La contrainte d'incompressibilité est bien respectée pour les deux modèles incompressibles. Enfin, on remarque que le modèle de Saint-Venant-Kirchhoff a des effets de compression plus importants que le cas Néo-Hookéen.



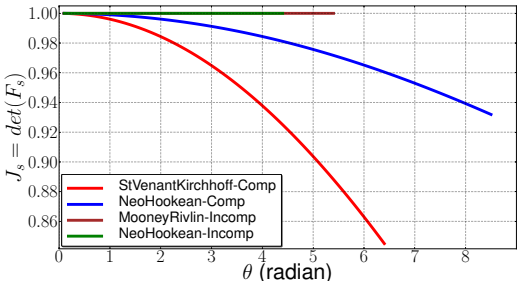
(a) Mooney-Rivlin incompressible pour $\theta = 1.6$



(b) Mooney-Rivlin incompressible pour $\theta = 5.1$



(c) Néo-Hookéen compressible pour $\theta = 8.6$



(d) Mesure de J_s en fonction de θ

FIGURE 5.6 – Résultats numériques de l'application modélisant la torsion d'une barre.

5 Modèle axisymétrique réduit

Pour pouvoir modéliser la propagation d'une onde de pression dans un fluide bidimensionnel, nous avons besoin d'utiliser un modèle d'élasticité axisymétrique. Dans ce repère, on peut reproduire le déplacement de la paroi vasculaire 3D à partir du déplacement radial calculé η_s (champs scalaires). On conserve le fait que cette paroi entoure le fluide. Ce modèle peut aussi encore être simplifié en supposant que l'épaisseur de la paroi vasculaire h est très fine. On obtient alors un modèle réduit à une dimension. Dans cette thèse, nous avons opté pour un modèle réduit 1D nommé *generalized string*. Il a été souvent décrit et utilisé dans la littérature, voir par exemple [59, 124, 123, 66]. Ce modèle sera ensuite utiliser ce modèle au chapitre 10.

La forme axi-symétrique qui nous intéresse est un tube droit de longeur L, de rayon R_0, d'éppaisseur h et orienté dans l'axe $\boldsymbol{e}_z$. La figure 5.7 nous montre le repère axisymétrique avec l'axe d'orientation $\boldsymbol{e}_z$ et l'axe radial $\boldsymbol{e}_r$. Le domaine réduit 1D est représenté par la ligne en pointillé marqué Ω_s^*. Le domaine de calcul du déplacement radial η_s est donc $\Omega_s^* = [0, L]$. Cependant, le fluide n'utilisera pas un formalisme axisymétrique, mais un modèle de fluide bidimensionnel. Le domaine noté $\Omega_s'^*$ devra alors aussi calculer un déplacement radial avec le même type de modèle. Seule la direction de l'axe radial sera inversée.

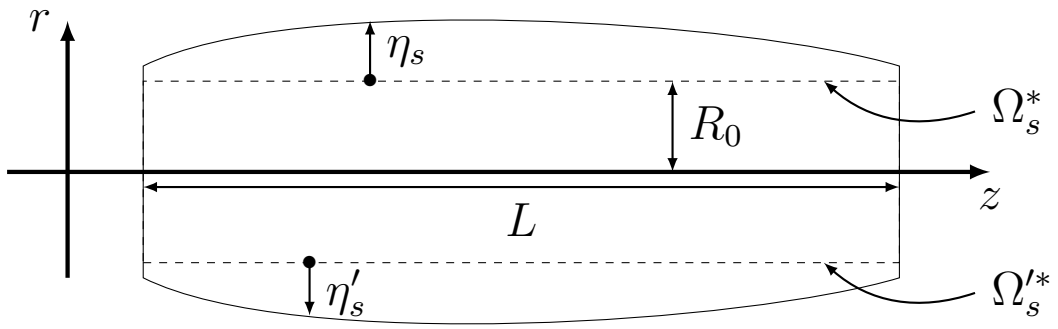


FIGURE 5.7 – Géométrie du modèle réduit 1D.

Nous présentons maintenant le problème mathématique qui décrit le modèle réduit de structure sur Ω_s^*. Il consiste à trouver le déplacement radial $\eta_s : \Omega_s^* \to \mathbb{R}$ vérifiant :

$$\rho_s^* h \frac{\partial^2 \eta_s}{\partial t^2} - k G_s h \frac{\partial^2 \eta_s}{\partial x^2} + \frac{E_s h}{1 - \nu_s^2} \frac{\eta_s}{R_0^2} - \gamma_v \frac{\partial^3 \eta}{\partial x^2 \partial t} = f_s.$$

Ici, h est l'épaisseur de la structure (supposée constante) et R_0 est le rayon du cylindre. La constante k désigne le facteur de correction de Timoshenko et γ_v est un paramètre de viscoélasticité. Les caractéristiques du matériau sont définies à l'aide de la masse volumique ρ_s^*, du module de Young E_s, du coefficient de Poisson ν_s et du module de cisaillement G_s. Nous avons aussi besoin de la fonction de déformation $\boldsymbol{\varphi}_s^t$ qui relie Ω_s^* et Ω_s^t. Elle s'exprime par :

$$\boldsymbol{\varphi}_s^t(z) = (z, R_0 + \eta_s(z, t)) \tag{5.45}$$

Pour les conditions aux limites, nous prenons simplement $\eta_s = 0$ sur $\partial\Omega_s^*$. La paroi vasculaire est donc fixée à ces extrémités. Il existent d'autres conditions aux limites plus réalistes, voir [123], mais elles ne seront pas abordées ici. Pour les conditions de couplage entre le fluide 2D et la structure 1D, nous modifions les relations classiques (6.1)-(6.3) du chapitre 6 par les suivantes :

$$\dot{\eta}_s \boldsymbol{e}_r - \boldsymbol{u}_f = \boldsymbol{0} \quad \text{(Continuité des vitesses)} \tag{5.46}$$

$$f_s + \left(J_{\mathcal{A}_f^t} \boldsymbol{F}_{\mathcal{A}_f^t}^{-T} \hat{\boldsymbol{\sigma}}_f \boldsymbol{n}_f^* \right) \cdot \boldsymbol{e}_r = 0 \quad \text{(Continuité des contraintes)} \tag{5.47}$$

$$\boldsymbol{\varphi}_s^t - \mathcal{A}_f^t = \boldsymbol{0} \quad \text{(Continuité géométrique)} \tag{5.48}$$

Lorsque nous résoudrons le modèle réduit pour un problème d'interaction fluide structure, nous utiliserons la condition de couplage (5.47). Cela conduira à la formulation variationnelle suivante :

Trouver $\eta_s \in H_0^1(\Omega_s^*)$ tel que $\forall \xi \in H_0^1(\Omega_s^*)$:

$$\int_0^L \rho_s^* h \frac{\partial^2 \eta_s}{\partial t^2} \xi + \int_0^L \frac{E_s h}{(1-\nu_s^2) R_0^2} \eta_s \xi + \int_0^L \left(k G_s h \frac{\partial \eta_s}{\partial x} + \gamma_v \frac{\partial^2 \eta_s}{\partial x \partial t} \right) \frac{\partial \xi}{\partial z} = \int_0^L f_s \xi.$$

Pour la discrétisation du terme d'accélération, nous utilisons le schéma de Newmark présenté au chapitre 5. Pour le terme viscoélastique, nous appliquons un schéma BDF_q.

Chapitre 6

Interaction fluide-structure

Pour la modélisation des écoulements sanguins, il est souvent nécessaire de considérer l'interaction entre le plasma et les parois vasculaires. En effet, nous avons déjà évoqué le rôle de l'aorte qui a pour but de réguler le débit sanguin pulsé [18, 114]. Plus généralement, cette interaction semble essentielle pour la modélisation du sang dans les artères. Dans le réseau veineux, cet effet semble moins influent. Ce phénomène joue aussi un rôle important pour la prédiction et la compréhension de certaines pathologies comme l'athérosclérose [50] et les anévrismes [165, 152]. La simulation de traitement thérapeutique comme les stents endovasculaires [166, 161] ou les valves aortiques artificielles [78, 36] nécessite de considérer cette interaction.

Ce type d'application a été traité dans de nombreuses études en mathématiques appliquées et en ingénierie biomédicale [106, 120, 157, 157, 16, 126]. La technique la plus couramment utilisée pour ces modèles d'interaction fluide-structure est l'approche ALE [59, 124, 123, 37, 66, 57, 82]. Deux types de techniques existent pour résoudre ce couplage. Nous avons les formulations monolithiques et les formulations partitionnées. Nous avons implémenté ces dernières, car elles sont plus simples à mettre en oeuvre et elles sont plus flexibles. Nous présenterons également une nouvelle méthode d'interaction fluide-structure pour la modélisation du mouvement de particules élastiques dans un écoulement. Cet outil sera basé sur la méthode FBM décrite au chapitre 3. L'ambition de ce modèle est de pouvoir modéliser une suspension de globules rouges dans le sang. Cependant, nous sommes encore à un stade d'expérimentation numérique. Ces premières applications ne nous permettront pas encore d'étudier la rhéologie des globules rouges.

Nous commencerons ce chapitre par une présentation de notre modèle d'interaction fluide structure. Il sera basé sur la carte ALE, la mécanique des fluides en domaine mobile et la mécanique des solides qui ont été décrits dans les chapitres précédents. Puis, nous montrerons les algorithmes de résolution que nous avons utilisés pour résoudre ce problème. Nous trouverons les schémas de couplage implicite et semi-implicite qui ont été présentés dans [124]. Une première application du modèle d'interaction fluide-structure sera ensuite effectuée sur un cas test classique. Nous finirons ce chapitre par la description de notre modèle d'interaction fluide-particule avec la FBM. Des exemples d'application de cette méthode seront également présentés dans le but de montrer les premières capacités de ce modèle.

1 Couplage de modèles

Pour décrire les problèmes d'interaction fluide-structure, nous choisissons de considérer la géométrie représentée par la figure 6.1. Celle-ci nous permet de donner les différentes notations associées à chacun des domaines fluides et structures. Ainsi, à l'instant t, nous considérons Ω_f^t comme le domaine fluide dans le repère ALE et Ω_s^t le domaine structure dans le repère eulérien. Pour le fluide, nous utilisons le formalisme de la mécanique des fluides en domaine mobile présenté au chapitre 4. Nous définissons alors le domaine de référence Ω_f^* qui est relié à Ω_f^t à l'aide de la carte ALE $\mathcal{A}_f^t$. Nous avons $\Omega_f^t = \mathcal{A}_f^t\left(\Omega_f^*\right)$. Nous posons aussi $\boldsymbol{\eta}_f = \mathcal{A}_f^t - \mathbf{x}^*$ le déplacement du domaine Ω_f^*. Pour les équations de la dynamique des solides, nous utilisons la formulation lagrangienne que nous avons décrite au chapitre 5. Ces modèles sont définis sur un domaine de référence Ω_s^* qui permet d'exprimer le domaine déformé par la relation $\Omega_s^t = \boldsymbol{\varphi}_s^t\left(\Omega_s^*\right)$. La transformation $\boldsymbol{\varphi}_s^t$ est définie à l'aide du déplacement $\boldsymbol{\eta}_s$ et s'exprime par $\boldsymbol{\varphi}_s^t(\mathbf{x}^*) = \mathbf{x}^* + \boldsymbol{\eta}_s(\mathbf{x}^*, t)$. Nous avons besoin de définir les frontières $\Gamma_s^{i,*}$, $\Gamma_s^{o,*}$ et $\Gamma_s^{e,*}$ appartenant au domaine Ω_s^*. L'image de ces frontières par la transformation $\boldsymbol{\varphi}_s^t$ nous donne respectivement les frontières $\Gamma_s^{i,t}$, $\Gamma_s^{o,t}$ et $\Gamma_s^{e,t}$. Les autres notations présentes et non décrites dans cette partie sont identiques à celles présentées dans les chapitres précédents.

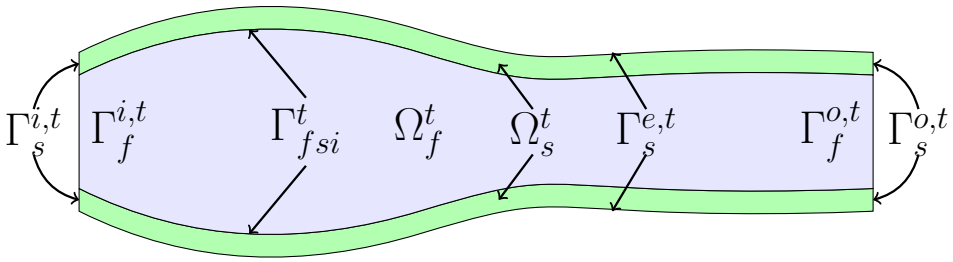


FIGURE 6.1 – Géométrie de référence pour un problème d'interaction fluide-structure.

Nous passons à présent aux conditions de couplage que le problème d'interaction fluide-structure doit vérifier. Celles-ci expriment plusieurs relations de continuité sur l'interface Γ_{fsi}^* entre les inconnues de la structure et les inconnues du fluide. Nous pouvons les exprimer par les expressions suivantes :

$$\frac{\partial \boldsymbol{\eta_s}}{\partial t} - \boldsymbol{u}_f \circ \mathcal{A}_f^t = \mathbf{0} \quad \text{(Continuité des vitesses)} \tag{6.1}$$

$$\boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s^* + J_{\mathcal{A}_f^t} \boldsymbol{F}_{\mathcal{A}_f^t}^{-T} \hat{\boldsymbol{\sigma}}_f \boldsymbol{n}_f^* = \mathbf{0} \quad \text{(Continuité des contraintes)} \tag{6.2}$$

$$\boldsymbol{\varphi}_s^t - \mathcal{A}_f^t = \mathbf{0} \quad \text{(Continuité géométrique)} \tag{6.3}$$

avec $\boldsymbol{F}_{\mathcal{A}_f^t} = \mathbf{x}^* + \nabla \mathcal{A}_f^t$ et $J_{\mathcal{A}_f^t} = det(\boldsymbol{F}_{\mathcal{A}_f^t})$.

Nous présentons maintenant notre modèle d'interaction fluide-structure que nous utiliserons dans la suite de cette thèse. Pour cela, nous utilisons le modèle de Navier-Stokes incompressible en domaine mobile pour la partie fluide. Pour la structure, nous considérons un modèle hyperélastique compressible. Cependant, le modèle incompressible pourra aussi être utilisé. Il faudra alors rajouter à ce modèle la contrainte d'incompressibilité que nous avons présentée dans le chapitre 5. Pour la carte ALE continue $\mathcal{A}_f^t$, c'est l'opérateur d'extension harmonique qui est présenté. Toutefois, d'autres opérateurs comme les

équations de Winslow auraient pu être posés. Ainsi, nous pouvons établir un modèle d'interaction fluide-structure. Il consiste à trouver les fonctions $\left(\boldsymbol{u}_f, p_f, \boldsymbol{\eta}_s, \mathcal{A}_f^t\right)$ qui vérifient le système d'équations suivant :

$$\rho_f \left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}^*} + \rho_f \left(\left(\boldsymbol{u}_f - \boldsymbol{w}_f\right) \cdot \nabla\right) \boldsymbol{u}_f - \nabla \cdot \boldsymbol{\sigma}_f = \boldsymbol{f}_f, \quad \text{dans } \Omega_f^t \times [t_i, t_f] \tag{6.4}$$

$$\nabla \cdot \boldsymbol{u}_f = 0, \quad \text{dans } \Omega_f^t \times [t_i, t_f] \tag{6.5}$$

$$\rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \nabla \cdot \boldsymbol{F}_s \boldsymbol{\Sigma}_s = \boldsymbol{f}_s, \quad \text{dans } \Omega_s^* \times [t_i, t_f] \tag{6.6}$$

$$-\Delta \boldsymbol{\eta}_f = \mathbf{0}, \quad \text{dans } \Omega_f^* \times [t_i, t_f] \tag{6.7}$$

$$\frac{\partial \boldsymbol{\eta_s}}{\partial t} - \boldsymbol{u}_f \circ \mathcal{A}_f^t = \mathbf{0}, \quad \text{sur } \Gamma_{fsi}^* \times [t_i, t_f] \tag{6.8}$$

$$\boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s^* + J_{\mathcal{A}_f^t} \hat{\boldsymbol{\sigma}}_f \boldsymbol{F}_{\mathcal{A}_f^t}^{-T} \boldsymbol{n}_f^* = \mathbf{0}, \quad \text{sur } \Gamma_{fsi}^* \times [t_i, t_f] \tag{6.9}$$

$$\boldsymbol{\varphi}_s^t - \mathcal{A}_f^t = \mathbf{0}, \quad \text{sur } \Gamma_{fsi}^* \times [t_i, t_f] \tag{6.10}$$

avec les termes sources $\boldsymbol{f}_f \in \left[L^2\left(\Omega_f^t\right)\right]^d$ et $\boldsymbol{f}_s \in \left[L^2\left(\Omega_f^t\right)\right]^d$, $\forall t \in [t_i, t_f]$. Les équations (6.4)-(6.5) représentent le modèle de Navier-Stokes. L'équation (6.6) modélise la dynamique de la structure. L'opérateur d'extension harmonique pour le mouvement du fluide est décrit par l'équation (6.7). Enfin, nous avons les conditions du couplage fluide-structure avec les équations (6.8)-(6.10).

Pour décrire un modèle complet d'application fluide-structure, nous devons aussi définir des conditions aux limites sur l'entrée $\Gamma_f^{i,t}$ et la sortie $\Gamma_f^{o,t}$ du fluide et sur les frontières $\Gamma_s^{i,t}$ et $\Gamma_s^{o,t}, \Gamma_s^{e,t}$ de la structure. Nous proposons alors les conditions suivantes :

$$\boldsymbol{\sigma}_f \boldsymbol{n}_f = \boldsymbol{g}_{in} \quad \text{sur } \Gamma_f^{i,t} \times [t_i, t_f] \tag{6.11}$$

$$\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0} \quad \text{sur } \Gamma_f^{o,t} \times [t_i, t_f] \tag{6.12}$$

$$\boldsymbol{\eta}_s = \mathbf{0} \quad \text{sur } \Gamma_s^{i,*} \cup \Gamma_s^{o,*} \times [t_i, t_f] \tag{6.13}$$

$$\boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s^* = \mathbf{0} \quad \text{sur } \Gamma_s^{e,*} \times [t_i, t_f] \tag{6.14}$$

$$\boldsymbol{\eta}_f = \mathbf{0} \quad \text{sur } \Gamma_f^{i,*} \cup \Gamma_f^{o,*} \times [t_i, t_f] \tag{6.15}$$

avec $\boldsymbol{g}_{in} \in \mathbf{H}^{1/2}\left(\Omega_f^t\right)$, $\forall t \in [t_i, t_f]$.

Ce modèle complet permet par exemple de modéliser la propagation d'une onde de pression dans une artère. La fonction $\boldsymbol{g}_{in}$ représente l'impulsion de l'onde de pression. Toutefois, ce modèle n'est pas très réaliste à cause des conditions aux limites en entrée et sortie. Nous trouverons au chapitre 10 des simulations numériques pour ce modèle d'écoulements sanguins.

2 Méthode partitionnée de résolution

Les méthodes numériques pour résoudre des problèmes d'interaction fluide-structure peuvent être classées en deux catégories : les méthodes partitionnées et les méthodes monolithiques. Dans une approche monolithique, le système d'équations non linéaires complet (6.4)-(6.10) complété de conditions aux limites est généralement résolu en utilisant une méthode de Newton ou une variante de celle-ci. Cette approche est reconnue pour être plus robuste qu'une approche partitionnée dans laquelle les problèmes fluides et solides sont

résolus séparément. L'approche monolithique a été utilisée par exemple dans [83, 66, 57]. Dans ce dernier document, on peut trouver une comparaison avec une méthode partitionnée et celui-ci démontre la supériorité de l'approche monolithique. Cependant les solveurs monolithiques sont moins flexibles, leur implémentation est plus complexe à mettre en oeuvre et la résolution algébrique est plus difficile. Dans cette thèse, nous nous sommes limités à l'approche partitionnée. Celle-ci est décrite dans de nombreuses références comme [124, 123, 10, 56, 99, 30, 148].

Nous allons donc découpler le modèle fluide-structure en sous-problèmes identifiés comme un problème fluide, un problème structure et un problème de construction de la carte ALE. Nous utilisons une formulation Dirichlet-Neumann pour décrire les problèmes fluides et structures. La solution du problème fluide-structure sera ensuite obtenue en utilisant un algorithme itératif. Nous commençons par poser $\tilde{\boldsymbol{w}}_f = \frac{\partial \boldsymbol{\eta}_s}{\partial t} \circ \left(\mathcal{A}_f^t\right)^{-1}\Big|_{\Gamma_{fsi}^t}$. En supposant connaitre $\tilde{\boldsymbol{w}}_f$ et $\boldsymbol{w}_f$, le problème associé au fluide à l'instant t consiste à trouver $(\boldsymbol{u}_f, p_f)$ vérifiant :

$$\begin{aligned}
\rho_f \left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}^*} + \rho_f \left((\boldsymbol{u}_f - \boldsymbol{w}_f)\cdot\nabla\right)\boldsymbol{u}_f - \nabla\cdot\boldsymbol{\sigma}_f &= \boldsymbol{f}_f, \quad \text{dans } \Omega_f^t \times [t_i, t_f] && (6.16)\\
\nabla\cdot\boldsymbol{u}_f &= 0, \quad \text{dans } \Omega_f^t \times [t_i, t_f] && (6.17)\\
\mathbf{u}_f &= \tilde{\boldsymbol{w}}_f, \quad \text{sur } \Gamma_{fsi}^t \times [t_i, t_f] && (6.18)\\
\boldsymbol{\sigma}_f \boldsymbol{n}_f &= \boldsymbol{g}_{in}, \quad \text{sur } \Gamma_f^{i,t} \times [t_i, t_f] && (6.19)\\
\boldsymbol{\sigma}_f \boldsymbol{n}_f &= \mathbf{0}, \quad \text{sur } \Gamma_f^{o,t} \times [t_i, t_f] && (6.20)
\end{aligned}$$

Pour la structure, nous supposons connaitre $\mathcal{A}_f^t$, $\boldsymbol{u}_f$ et p_f. Ce problème consiste à trouver $\boldsymbol{\eta}_s$ vérifiant :

$$\begin{aligned}
\rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} - \nabla\cdot\boldsymbol{F}_s\boldsymbol{\Sigma}_s &= \boldsymbol{f}_s, \quad \text{dans } \Omega_s^* \times [t_i, t_f] && (6.21)\\
\boldsymbol{F}_s\boldsymbol{\Sigma}_s\boldsymbol{n}_s^* &= -J_{\mathcal{A}_f^t}\boldsymbol{F}_{\mathcal{A}_f^t}^{-T}\hat{\boldsymbol{\sigma}}_f\boldsymbol{n}_f^*, \quad \text{sur } \Gamma_{fsi}^* \times [t_i, t_f] && (6.22)\\
\boldsymbol{\eta}_s &= \mathbf{0}, \quad \text{sur } \Gamma_s^{i,*} \cup \Gamma_s^{o,*} \times [t_i, t_f] && (6.23)\\
\boldsymbol{F}_s\boldsymbol{\Sigma}_s\boldsymbol{n}_s^* &= \mathbf{0}, \quad \text{sur } \Gamma_f^{e,*} \times [t_i, t_f] && (6.24)
\end{aligned}$$

Enfin, il reste à calculer la carte ALE $\mathcal{A}_f^t$. Nous considérons toujours l'extension harmonique définie par le déplacement $\boldsymbol{\eta}_s$ sur l'interface Γ_{fsi}^* :

$$\begin{aligned}
-\Delta\boldsymbol{\eta}_f &= \mathbf{0}, \quad \text{dans } \Omega_f^* \times [t_i, t_f] && (6.25)\\
\boldsymbol{\eta}_f &= \boldsymbol{\eta}_s, \quad \text{sur } \Gamma_{fsi}^* \times [t_i, t_f] && (6.26)\\
\boldsymbol{\eta}_f &= \mathbf{0}, \quad \text{sur } \Gamma_f^{i,*} \cup \Gamma_f^{o,*} \times [t_i, t_f] && (6.27)
\end{aligned}$$

La description des formulations variationnelles associées aux problèmes (6.16)-(6.20), (6.21)-(6.24) et (6.25)-(6.27) a déjà été faite dans les chapitres précédents. Nous ne les décrirons pas ici et nous ramenons donc le lecteur aux chapitres 2, 4 et 5.

2.1 Discrétisations numériques

Nous passons maintenant à la discrétisation spatiale et temporelle du modèle d'interaction fluide-structure. La méthode numérique utilisée pour résoudre ses sous-problèmes

est toujours la méthode éléments finis. Nous échantillonnons l'intervalle de temps $[t_i, t_f]$ en N_T pas de temps, notés $\{t_n\}_{1\leq n\leq N_T}$. Pour la discrétisation géométrique, nous considérons le paramètre $\delta_f \equiv (h_f, k_f)$ (resp $\delta_s \equiv (h_s, k_s)$) pour le domaine fluide (resp structure). Ces paramètres ne seront pas forcément identiques pour les deux domaines. Toutefois, pour simplifier les notations, nous considérons le même paramètre noté δ pour chaque domaine. Lorsqu'il sera nécessaire de faire la différence entre les paramètres du fluide et de la structure, nous expliciterons ces données. Dans la suite de cette description, toutes les notations utilisées seront les mêmes que celles utilisées dans les chapitres précédents. Par exemple, nous notons $\boldsymbol{u}_{f,\delta}^{n+1}$ le champ de vitesse discret du fluide au temps t_{n+1}.

Nous allons commencer par décrire les espaces d'approximation dans lesquels nous allons chercher les solutions $\boldsymbol{u}_{f,\delta}^n$, $p_{f,\delta}^n$, $\boldsymbol{\eta}_{s,\delta}^n$ et $\mathcal{A}_{f,\delta}^n$, $\forall n \in \{1, ..., N_T\}$. Ces solutions vérifient le système d'équation (6.4)-(6.10) ainsi que les conditions aux limites (6.11)-(6.15) d'un point de vue discret.

Nous avons tout d'abord l'espace d'approximation pour la carte ALE $\mathcal{A}_{f,\delta}^n$ et donc aussi le déplacement du domaine $\boldsymbol{\eta}_{f,\delta}^n$:

$$A_{f,\delta}^{t_n} = \left\{ \boldsymbol{v} : \Omega_{f,\delta}^* \longmapsto \mathbb{R}^d, \boldsymbol{v} \in \mathbf{H}^1_{(\boldsymbol{\eta}_{s,\delta}^n, \Gamma_{fsi,\delta}^*)} \left(\Omega_{f,\delta}^*\right) \cap \mathbf{H}^1_{(\mathbf{0}, \Gamma_{f,\delta}^{i,*} \cup \Gamma_{f,\delta}^{o,*})} \left(\Omega_{f,\delta}^*\right) \cap \left[P_c^k \left(\Omega_{f,\delta}^*\right)\right]^d \right\}$$

Nous avons ensuite les espaces d'approximation pour la vitesse et la pression du fluide en considérant $\tilde{\boldsymbol{w}}_{f,\delta}^n = \dot{\boldsymbol{\eta}}_{s,\delta}^n \circ \left(\mathcal{A}_{f,\delta}^n\right)^{-1}\Big|_{\Gamma_{fsi,\delta}^{t_n}}$:

$$\begin{aligned} V_{f,\delta}^{t_n} &= \left\{ \boldsymbol{v} : \Omega_{f,\delta}^{t_n} \longmapsto \mathbb{R}^d, \boldsymbol{v} = \hat{\boldsymbol{v}} \circ \left(\mathcal{A}_{f,\delta}^n\right)^{-1}, \hat{\boldsymbol{v}} \in \mathbf{H}^1_{(\tilde{\boldsymbol{w}}_{f,\delta}^n, \Gamma_{fsi,\delta}^*)} \left(\Omega_{f,\delta}^*\right) \cap \left[P_c^M \left(\Omega_{f,\delta}^*\right)\right]^d \right\} \\ Q_{f,\delta}^{t_n} &= \left\{ q : \Omega_{f,\delta}^t \longmapsto \mathbb{R}, q = \hat{q} \circ \left(\mathcal{A}_{f,\delta}^n\right)^{-1}, \hat{q} \in P_c^N \left(\Omega_{f,\delta}^*\right) \right\} \end{aligned}$$

Puis, l'espace d'approximation du déplacement de la structure en remarquant qu'il ne dépend pas du temps :

$$V_{s,\delta} = \left\{ \boldsymbol{v} : \Omega_{s,\delta}^* \longmapsto \mathbb{R}^d, \boldsymbol{v} \in \mathbf{H}^1_{\mathbf{0}, \Gamma_{s,\delta}^{i,*}}(\Omega_s^*) \cap \mathbf{H}^1_{\mathbf{0}, \Gamma_{s,\delta}^{o,*}}(\Omega_s^*) \cap \left[P_c^S \left(\Omega_{s,\delta}^*\right)\right]^d \right\}$$

Enfin, nous avons aussi besoin de définir l'espace d'approximation associé aux fonctions tests de la vitesse du fluide :

$$W_{f,\delta}^{t_n} = \left\{ \boldsymbol{v} : \Omega_{f,\delta}^{t_n} \longmapsto \mathbb{R}^d, \boldsymbol{v} = \hat{\boldsymbol{v}} \circ \left(\mathcal{A}_{f,\delta}^n\right)^{-1}, \hat{\boldsymbol{v}} \in \mathbf{H}^1_{(\mathbf{0}, \Gamma_{fsi,\delta}^*)} \left(\Omega_{f,\delta}^*\right) \cap \left[P_c^M \left(\Omega_{f,\delta}^*\right)\right]^d \right\}$$

Pour le choix des ordres polynomiaux M et N nous utilisons la même stratégie que celle présenté au chapitre 4, c'est-à-dire que l'on utilise les éléments de Taylor-Hood en prenant la relation $N = M - 1$. On contraint également que $M > k$ pour assurer que les champs vitesse et pression soient isoparamétriques ou subparamétriques. Pour le choix de l'ordre polynomial du déplacement S, on choisit généralement $S = k_f$ lorsque l'interface discrète entre le fluide et la structure est conforme dans les états de référence $\Omega_{f,\delta}^*$ et $\Omega_{s,\delta}^*$. Ce choix nous permet d'avoir une description suffisamment précise pour décrire le déplacement du domaine fluide. Un ordre supérieur n'apporterait pas plus de précision pour le déplacement du domaine fluide. Cependant, ce choix n'est pas toujours possible lorsque l'on utilise par exemple un modèle incompressible (contrainte de Taylor-Hood). Ce constat n'est plus vrai aussi lorsque l'interface n'est plus conforme. Dans ce cas, l'ordre élevé de S nous permettra de mieux approximer le déplacement à imposer sur la frontière $\Gamma_{fsi,\delta}^*$.

En utilisant les formulations variationnelles discrètes des chapitres 4 et 5, nous obtenons les problèmes représentant respectivement le modèle fluide et le modèle structure :

Trouver $(\boldsymbol{u}^n_{f,\delta}, p^n_{f,\delta}) \in V^{t_n}_{f,\delta} \times Q^{t_n}_{f,\delta}$ tel que $\forall (v,q) \in W^{t_n}_{f,\delta}$:

$$\rho_f \int_{\Omega^{t_n}_{f,\delta}} \left.\frac{\partial \boldsymbol{u}^n_{f,\delta}}{\partial t}\right|_{\mathbf{x}^*} \cdot \boldsymbol{v} \; + \; \rho_f \int_{\Omega^{t_n}_{f,\delta}} \left(\left(\boldsymbol{u}^n_{f,\delta} - \boldsymbol{w}^n_{f,\delta} \right) \cdot \nabla_{\mathbf{x}} \right) \boldsymbol{u}^n_{f,\delta} \cdot \boldsymbol{v}$$
$$+ \int_{\Omega^{t_n}_{f,\delta}} \boldsymbol{\sigma}^n_{f,\delta} : \nabla_{\mathbf{x}} \boldsymbol{v} = \int_{\Omega^{t_n}_{f,\delta}} \boldsymbol{f}^{t_n}_f \cdot \boldsymbol{v} + \int_{\Gamma^{i,t_n}_{f,\delta}} \boldsymbol{g}^{t_n}_{in} \cdot \boldsymbol{v} \tag{6.28}$$

$$\int_{\Omega^{t_n}_{f,\delta}} q \; \nabla_{\mathbf{x}} \cdot \boldsymbol{u}^n_{f,\delta} = 0 \tag{6.29}$$

Trouver $\boldsymbol{\eta}^n_{s,h} \in V_{s,\delta}$ tel que $\forall \boldsymbol{\xi} \in V_{s,\delta}$:

$$\int_{\Omega^*_{s,\delta}} \rho^*_s \ddot{\boldsymbol{\eta}}^n_{s,\delta} \cdot \boldsymbol{\xi} + \int_{\Omega^*_{s,\delta}} \left(\boldsymbol{F}^n_{s,\delta} \boldsymbol{\Sigma}^n_{s,\delta} \right) : \nabla \boldsymbol{\xi}$$
$$= \int_{\Omega^*_{s,\delta}} \boldsymbol{f}^{t_n}_s \cdot \boldsymbol{\xi} + \int_{\Gamma^*_{fsi,\delta}} J_{\mathcal{A}^n_{f,\delta}} \boldsymbol{F}^{-T}_{\mathcal{A}^n_{f,\delta}} \hat{\boldsymbol{\sigma}}^n_{f,\delta} \boldsymbol{n}^*_{s,\delta} \cdot \boldsymbol{\xi} \tag{6.30}$$

La construction de la carte ALE discrète $\mathcal{A}^n_{f,\delta}$ ne sera pas décrite dans ce chapitre. Nous référons le lecteur au chapitre 2 dédié à cette construction. Pour résumer nous calculons la carte $\mathcal{A}^n_{f,(h,1)}$ par une méthode classique comme l'extension harmonique ou les équations de Winslow. Puis nous appliquons une correction sur les éléments frontières pour obtenir la carte $\mathcal{A}^n_{f,\delta}$ d'ordre élevé, si celle-ci est nécessaire.

Pour les discrétisations des dérivées temporelles, nous référons également le lecteur à ces mêmes précédents chapitres. Nous pouvons rappeler que pour le fluide, nous utilisons un schéma BDF_q et pour la structure un schéma de Newmark.

2.2 Algorithmes de résolution

Nous allons présenter les algorithmes itératifs de type point fixe permettant de résoudre le problème fluide-structure avec la méthode de résolution partitionnée. Plusieurs types d'algorithmes existent dans ce contexte. Nous avons les schémas implicites qui assurent la vérification exacte du modèle d'interaction fluide-structure. On parle alors de couplage fort. Nous avons aussi les schémas explicites et semi-implicites. Dans ces cas, les conditions de couplage du modèle fluide-structure ne sont pas vérifiées exactement. Ces techniques permettent de simplifier la méthodologie numérique et elles sont très performantes comparées aux schémas implicites. Ces méthodes représentent un couplage fluide-structure dit faible. Dans de nombreux contextes, en particulier dans l'aéroélasticité, le couplage faible est suffisant pour obtenir des résultats précis. Cependant, dans le contexte d'écoulement sanguin, cette facilité n'est plus permise. Quelques études montrent que lorsque la masse volumique du fluide ρ_f et la masse volumique de la structure ρ^*_s sont proches, alors des instabilités numériques peuvent apparaitre avec un couplage fluide structure faible (ou approché). Avec un couplage fort, c'est le temps de calcul qui est affecté, car le nombre d'itérations pour réaliser ce couplage augmente fortement. On parle du phénomène de masse ajoutée qui est démontré dans [37, 11].

Dans cette thèse, nous avons testé les schémas implicites et semi-implicites. Ces derniers peuvent souffrir d'instabilités numériques dans notre contexte de rhéologie sanguine. Cependant, lorsque le pas de temps est petit, ces effets non désirés peuvent être atténués.

Dans la même référence [37], une correction à ces perturbations est proposée pour un modèle simplifié. Pour chacun de ces schémas, une méthode de point fixe avec relaxation de Aitken est utilisée. Cette technique d'accélération, décrite par exemple dans [99], est parfois indispensable pour certaines applications. C'est le cas pour la simulation de la propagation d'onde de pression dans les vaisseaux sanguins.

Avant de donner les algorithmes de résolution implicites et semi-implicites, nous commençons par décrire les principales étapes incluses dans ces stratégies de résolution itérative. Nous ajoutons dans nos notations l'indice k représentant la $k^{\text{ème}}$ itération de l'algorithme itératif (ex : $\boldsymbol{u}_{f,\delta}^{k,n+1}$). Sans cet indice, on considère que la variable a atteint l'état de convergence. Nous trouvons alors les fonctions suivantes :

- $\boldsymbol{\eta}_{s,\delta}^{0,n+1} = \textbf{PrédictionDéplacement}(\boldsymbol{\eta}_{s,\delta}^{n}, \boldsymbol{\eta}_{s,\delta}^{n-1}, ...)$: $\boldsymbol{\eta}_{s,\delta}^{0,n+1}$ est la prédiction du déplacement sur l'interface fluide structure. Nous pouvons donner à titre d'exemple une formule d'ordre 2 pour cette prédiction :

$$\boldsymbol{\eta}_{s,\delta}^{0,n+1} = \boldsymbol{\eta}_{s,\delta}^{n} + \Delta t \left(\frac{3}{2} \dot{\boldsymbol{\eta}}_{s,\delta}^{n} - \frac{1}{2} \dot{\boldsymbol{\eta}}_{s,\delta}^{n-1} \right)$$

- $\mathcal{A}_{f,\delta}^{k,n+1} = \textbf{SolveurALE}(\boldsymbol{\eta}_{s,\delta}^{k-1,n+1})$: cette étape représente le calcul de la carte ALE du domaine fluide permettant le déplacement de celui-ci. Il est défini à partir du déplacement de la structure sur l'interface $\Gamma_{fsi,\delta}^{*}$. La carte ALE permet alors le calcul de la vitesse de déformation $\boldsymbol{w}_{f,\delta}^{k,n+1}$.
- $(\boldsymbol{u}_{f,\delta}^{k,n+1}, p_{f,\delta}^{k,n+1}) = \textbf{SolveurFluide}(\mathcal{A}_{f,\delta}^{k,n+1}, \tilde{\boldsymbol{w}}_{f,\delta}^{n+1})$: résolution du problème fluide décrit par les équations 6.28-6.29. Ce problème dépend de la vitesse de déformation du domaine fluide $\boldsymbol{w}_{f,\delta}^{n+1}$ issue de la carte ALE. Nous avons aussi besoin de la vitesse de déplacement de la structure sur l'interface fluide-structure $\tilde{\boldsymbol{w}}_{f,\delta}^{n+1}$ qui a été transformée dans le repère ALE. On pourra constater que $\boldsymbol{w}_{f,\delta}^{n+1} = \tilde{\boldsymbol{w}}_{f,\delta}^{n+1}$ dans le cas du schéma implicite.
- $\tilde{\boldsymbol{\eta}}_{s,\delta}^{k,n+1} = \textbf{SolveurStructure}(\hat{\boldsymbol{\sigma}}_{f}^{k,n+1}, \mathcal{A}_{f,\delta}^{k,n+1})$: résolution du problème structure décrit par l'équation 6.30. La condition de couplage sur l'interface fluide structure est imposée à l'aide du tenseur de contrainte du fluide et de la carte ALE.
- $\epsilon = \textbf{CalculConvergence}(\tilde{\boldsymbol{\eta}}_{s,\delta}^{k,n+1}, \boldsymbol{\eta}_{s,\delta}^{k-1,n+1})$: fonction qui évalue la convergence de l'algorithme itératif en évaluant la norme L^2 relative suivante :

$$\epsilon = \frac{\left\| \tilde{\boldsymbol{\eta}}_{s,\delta}^{k,n+1} - \boldsymbol{\eta}_{s,\delta}^{k-1,n+1} \right\|_{L^2(\Omega_{s,\delta}^{*})}}{\left\| \boldsymbol{\eta}_{s,\delta}^{k-1,n+1} \right\|_{L^2(\Omega_{s,\delta}^{*})}}$$

- $\theta^k = \textbf{CalculParamètreRelaxation}(\tilde{\boldsymbol{\eta}}_{s,\delta}^{n+1}, \boldsymbol{\eta}_{s,\delta}^{k-1,n+1}, \theta^{k-1})$: les algorithmes itératifs présentés dans la suite nécessitent d'utiliser une relaxation du déplacement $\boldsymbol{\eta}_{s,\delta}^{n+1}$. Cette étape calcule le paramètre de relaxation θ^k par la méthode d'Aitken.

Schéma implicite

Nous commençons par le schéma implicite avec l'algorithme 4. C'est la technique la plus robuste des méthodes partitionnées pour le modèle d'interaction fluide-structure. Cet algorithme nous permettra de résoudre exactement les équations du modèle avec ces conditions de couplage. Le principal désavantage de cette méthode est que le domaine

de calcul fluide se déforme à chaque résolution. Ainsi, il est nécessaire de recalculer tous les termes du problème fluide à chaque itération du point fixe. Toutefois, pour résoudre le modèle non linéaire de Navier-Stokes, nous pouvons utiliser une méthode de quasi-Newton en sauvegardant la matrice jacobienne dans l'itération du solveur algébrique, mais aussi dans les itérations du point fixe fluide-structure. Le préconditionneur est donc aussi sauvegardé. On utilise ces données jusqu'à ce que le solveur algébrique diverge ou ne soit plus efficace. Dans le cas où l'on utiliserait un modèle linéaire, le modèle d'Oseen par exemple, la matrice ne peut plus être sauvegardée. On peut cependant réutiliser le préconditionneur au cours des itérations du point fixe.

Algorithm 4 Solveur FSI implicite

Require: $\{t_n\}_{1\leq n\leq T_N}$, $\boldsymbol{u}_{f,\delta}^0$ et $\boldsymbol{\eta}_{s,\delta}^0$

for $n = 0$ à $N_T - 1$ **do**
  $\boldsymbol{\eta}_{s,\delta}^{0,n+1} = \textbf{PrédictionDéplacement}(\boldsymbol{\eta}_{s,\delta}^{n}, \boldsymbol{\eta}_{s,\delta}^{n-1}, ...)$
  $k = 1$
  while $\epsilon > TOL$ **do**
    $\mathcal{A}_{f,\delta}^{k,n+1} = \textbf{SolveurALE}(\boldsymbol{\eta}_{s,\delta}^{k-1,n+1})$
    $\tilde{\boldsymbol{w}}_{f,\delta}^{k,n+1} = \dot{\boldsymbol{\eta}}_{s,\delta}^{k-1,n+1} \circ \left(\mathcal{A}_{f,\delta}^{k,n+1}\right)^{-1}\Big|_{\Gamma_{fsi,\delta}^{t_n}}$
    $(\boldsymbol{u}_{f,\delta}^{k,n+1}, p_{f,\delta}^{k,n+1}) = \textbf{SolveurFluide}(\mathcal{A}_{f,\delta}^{k,n+1}, \tilde{\boldsymbol{w}}_{f,\delta}^{k,n+1})$
    $\hat{\boldsymbol{\sigma}}_{f,\delta}^{k,n+1} = \boldsymbol{\sigma}_{f,\delta}^{k,n+1} \circ \mathcal{A}_{f,\delta}^{k,n+1}$
    $\tilde{\boldsymbol{\eta}}_{s,\delta}^{k} = \textbf{SolveurStructure}(\hat{\boldsymbol{\sigma}}_{f}^{k,n+1}, \mathcal{A}_{f,\delta}^{k,n+1})$
    $\epsilon = \textbf{CalculConvergence}(\tilde{\boldsymbol{\eta}}_{s,\delta}^{k}, \boldsymbol{\eta}_{s,\delta}^{k-1,n+1})$
    if $\epsilon > TOL$ **then**
      $\theta^k = \textbf{CalculParamètreRelaxation}(\tilde{\boldsymbol{\eta}}_{s,\delta}^{k}, \boldsymbol{\eta}_{s,\delta}^{k-1,n+1}, \theta^{k-1})$
      $\boldsymbol{\eta}_{s,\delta}^{k,n+1} = \theta^k \tilde{\boldsymbol{\eta}}_{s,\delta}^{k} + (1-\theta^k)\boldsymbol{\eta}_{s,\delta}^{k-1,n+1}$
      $k = k + 1$
    end if
  end while
  $\boldsymbol{u}_{f,\delta}^{n+1} = \boldsymbol{u}_{f,\delta}^{k,n+1}$, $p_{f,\delta}^{n+1} = p_{f,\delta}^{k,n+1}$, $\boldsymbol{\eta}_{s,\delta}^{n+1} = \tilde{\boldsymbol{\eta}}_{s,\delta}^{k}$, $\mathcal{A}_{f,\delta}^{n+1} = \mathcal{A}_{f,\delta}^{k,n+1}$
end for

Schéma semi-implicite

Pour ce schéma, la condition de couplage fluide-structure ne sera pas exactement vérifiée. Il est décrit par l'algorithme 5. Nous explicitons la partie de calcul de la carte ALE en l'éjectant de la procédure itérative du point fixe. La carte ALE $\mathcal{A}_{f,\delta}^{n+1}$ et ainsi la vitesse de déformation du domaine $\boldsymbol{w}_{f,\delta}^{n+1}$ seront constantes dans ces itérations. Cela induit que le domaine ne bouge pas. Nous pouvons alors optimiser le code de calcul en stockant tous les termes linéaires des équations de Navier-Stokes apparaissant dans la matrice jacobienne. On remarque aussi que sur l'interface FSI, la vitesse de déformation $\boldsymbol{w}_{f,\delta}^{n+1}\big|_{\Gamma_{fsi,\delta}^{t_n}}$ n'est pas égale à la vitesse de déformation de la structure $\tilde{\boldsymbol{w}}_{f,\delta}^{n+1}$. Par contre, cet algorithme nécessite plus d'itération de point fixe que le schéma implicite pour obtenir convergence. Cependant, grâce à l'optimisation décrite précédemment et en réutilisant le

Algorithm 5 Solveur FSI semi-implicite

Require: $\{t_n\}_{1\leq n\leq T_N}$, $\boldsymbol{u}^0_{f,\delta}$ et $\boldsymbol{\eta}^0_{s,\delta}$

for $n = 0$ à $N_T - 1$ **do**
  $\mathcal{A}^{n+1}_{f,\delta} = \textbf{SolveurALE}(\boldsymbol{\eta}^n_{s,\delta})$
  $\boldsymbol{\eta}^{0,n+1}_{s,\delta} = \textbf{PrédictionDéplacement}(\boldsymbol{\eta}^n_{s,\delta}, \boldsymbol{\eta}^{n-1}_{s,\delta}, ...)$
  $k = 1$
  while $\epsilon > TOL$ **do**
    $\tilde{\boldsymbol{w}}^{k,n+1}_{f,\delta} = \dot{\boldsymbol{\eta}}^{k-1,n+1}_{s,\delta} \circ \left(\mathcal{A}^{n+1}_{f,\delta}\right)^{-1}\Big|_{\Gamma^{t_n}_{fsi,\delta}}$
    $(\boldsymbol{u}^{k,n+1}_{f,\delta}, p^{k,n+1}_{f,\delta}) = \textbf{SolveurFluide}(\mathcal{A}^{n+1}_{f,\delta}, \tilde{\boldsymbol{w}}^{k,n+1}_{f,\delta})$
    $\hat{\boldsymbol{\sigma}}^{k,n+1}_{f,\delta} = \boldsymbol{\sigma}^{k,n+1}_{f,\delta} \circ \mathcal{A}^{n+1}_{f,\delta}$
    $\tilde{\boldsymbol{\eta}}^{k}_{s,\delta} = \textbf{SolveurStructure}(\hat{\boldsymbol{\sigma}}^{k,n+1}_{f}, \mathcal{A}^{n+1}_{f,\delta})$
    $\epsilon = \textbf{CalculConvergence}(\tilde{\boldsymbol{\eta}}^{k}_{s,\delta}, \boldsymbol{\eta}^{k-1,n+1}_{s,\delta})$
    if $\epsilon > TOL$ **then**
      $\theta^k = \textbf{CalculParamètreRelaxation}(\tilde{\boldsymbol{\eta}}^{k}_{s,\delta}, \boldsymbol{\eta}^{k-1,n+1}_{s,\delta}, \theta^{k-1})$
      $\boldsymbol{\eta}^{k,n+1}_{s,\delta} = \theta^k \tilde{\boldsymbol{\eta}}^{k}_{s,\delta} + (1-\theta^k)\boldsymbol{\eta}^{k-1,n+1}_{s,\delta}$
      $k = k + 1$
    end if
  end while
  $\boldsymbol{u}^{n+1}_{f,\delta} = \boldsymbol{u}^{k,n+1}_{f,\delta}$, $p^{n+1}_{f,\delta} = p^{k,n+1}_{f,\delta}$, $\boldsymbol{\eta}^{n+1}_{s,\delta} = \tilde{\boldsymbol{\eta}}^{k}_{s,\delta}$
end for

préconditionneur, nous obtenons un algorithme numérique qui est plus performant que le précédent. Dans le cas du modèle linéaire d'Oseen, cette optimalité est très visible. Mais nous rappelons que ce schéma est moins précis et peut faire apparaitre des instabilités. Un pas de temps relativement petit est nécessaire pour l'utiliser.

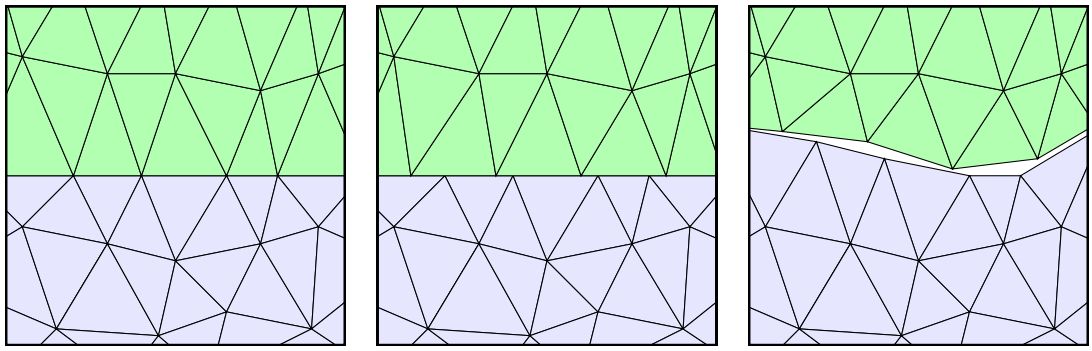
Stabilité numérique

La stabilité numérique des schémas de couplage n'a pas été profondément étudiée dans cette thèse, mais nous avons toutefois fait une recherche bibliographique sur ce sujet. Généralement, la stabilité numérique est analysée au sens de la norme énergie. Dans [103, 102, 123, 59], il est montré que le schéma implicite est inconditionnellement stable grâce à la l'imposition exacte des conditions de couplage (6.1)-(6.3). Pour le schéma semi-implicite, ces conditions ne sont plus vérifiées exactement. On peut citer les travaux de [56, 9] qui ont montrer sur un problème un peu plus simple (Stokes et élasticité linéaire) qu'il y avait une condition de stabilité à vérifier. En particulier, la taille caractéristique du maillage et le pas de temps ne doivent pas être trop grand. Toutefois, les discrétisations temporelles faites dans ces références diffèrent légèrement des nôtres, en particulier le schéma du point milieu est majoritairement utilisé pour la structure. Une investigation plus profonde devra être faite dans le futur avec nos schémas discrets.

2.3 Conformités et non conformités

Pour finir cette description, nous allons parler de la gestion des conformités et des non-conformités de l'interface fluide-structure discrète Γ^t_{fsi}. Dans notre modèle d'interaction

fluide structure, nous devons échanger des informations sur cette interface. Cependant, les équations du fluide et de la structure ne vivent pas dans le même repère. Nous avons choisi de réaliser ce transfert en ramenant le domaine fluide sur son domaine de référence. Ainsi, l'échange des données s'effectue sur l'interface $\Gamma^*_{fsi,\delta}$ à l'aide de la transformation ALE. Les données transmises sont le champ de déplacement de la structure $\boldsymbol{\eta}_{s,\delta}$ et le tenseur des contraintes du fluide $\hat{\boldsymbol{\sigma}}_{f,\delta}$. Dans le schéma semi-implicite, nous devons aussi transmettre le champ de vitesse de la structure $\dot{\boldsymbol{\eta}}_{s,\delta}$. Ces informations sont transmises à l'aide de l'opérateur d'interpolation que nous avons décrit au chapitre 1. L'interface $\Gamma^*_{fsi,\delta}$ n'est pas forcément identique dans le domaine fluide et dans le domaine structure. Ainsi, la conformité de l'interface fluide structure peut être de plusieurs types. Nous les avons illustrés avec la figure 6.2. En bleu clair, nous avons le domaine fluide $\Omega^*_{f,\delta}$ et en vert, nous avons le domaine structure $\Omega^*_{s,\delta}$. On note pour cette partie $\Sigma^*_{f,\delta}$ (resp $\Sigma^*_{s,\delta}$) l'interface fluide-structure discrète de la partie fluide (structure).



(a) Interface FSI conforme (b) Interface FSI non conforme (c) Interface FSI disjointe

FIGURE 6.2 – Types de conformité et non-conformité de l'interface fluide structure. En bleu le domaine fluide, en vert le domaine structure.

Lorsque l'interface est conforme, figure 6.2(a), les champs de déplacement et le tenseur des contraintes peuvent être transmis parfaitement. Toutefois, la discontinuité du tenseur de contrainte $\hat{\sigma}_{f,\delta}$ nécessite d'utiliser l'opérateur d'interpolation avec précaution. Il faut interpoler les valeurs de $\hat{\sigma}_{f,\delta}$ entre les mêmes éléments pour assurer une transmission précise. Considérons par un exemple l'interpolation du point $\mathbf{x}^*_s \in K_s$ et $K_s \in \Sigma^*_{s,\delta}$. On interpolera ce point sur l'élément $K_f \in \Sigma^*_{f,\delta}$ tel que $K_f = K_s$.

Pour le deuxième cas, figure 6.2(b), nous avons une interface non-conforme. Cependant, tous les points appartenant à $\Sigma^*_{f,\delta}$ peuvent être localisés dans $\Sigma^*_{s,\delta}$ et vice-versa. L'interpolé du champ de déplacement $\boldsymbol{\eta}_{s,\delta}$ (et aussi $\dot{\boldsymbol{\eta}}_{s,\delta}$) sur l'interface $\Sigma^*_{f,\delta}$ aura des propriétés d'interpolation optimale (dans le sens de l'interpolation de Lagrange). Cela est possible à cause de la continuité de cette fonction. Par contre, ce ne sera pas le cas pour le tenseur des contraintes $\hat{\boldsymbol{\sigma}}_{f,\delta}$. Plusieurs possibilités existent pour réaliser ce transfert. Une première solution très simple consiste à considérer l'interpolé de $\hat{\boldsymbol{\sigma}}_{f,\delta}$ comme une fonction continue. On évalue chaque point $\mathbf{x}^*_s \in K_s\Sigma^*_{s,\delta}$ de l'interpolé en cherchant l'élément $K_f \in \Sigma^*_{f,\delta}$ tel que $\mathbf{x}^*_s \in K_f$. Si l'on tombe sur un noeud appartenant à plusieurs éléments, on choisit aléatoirement un des éléments pour calculer l'interpolation du point. Une solution plus propre mathématiquement consiste à calculer l'interpolation L^2 du champ discontinu $\hat{\boldsymbol{\sigma}}_{f,\delta}$ en une fonction continue sur $\Sigma^*_{f,\delta}$. On applique ensuite l'interpolation de Lagrange sur cette nouvelle donnée pour obtenir une approximation continue de ce tenseur

des contraintes sur $\Sigma^*_{s,\delta}$. Le principal désavantage de cette méthode est qu'elle nécessite la résolution d'un problème supplémentaire pour calculer l'interpolé L^2.

Enfin, nous avons une dernière configuration présentée avec la figure 6.2(c). Elle correspond au cas où les interfaces fluide-structure ne se superposent pas, c'est-à-dire que l'on a $\Sigma^*_{f,\delta} \neq \Sigma^*_{s,\delta}$. Pour ce cas dit interfaces disjointes, nous allons reprendre les techniques employées pour le cas non conforme avec en plus l'utilisation de l'extrapolation. Cette méthode d'extrapolation utilisée dans l'opérateur d'interpolation est celle que nous avons décrite au chapitre 1. La différence du cas précédent est que nous n'avons plus l'estimation d'erreur optimale dans le transfert du déplacement $\boldsymbol{\eta}_{s,\delta}$.

Pour conclure cette partie, nous pouvons dire qu'il est préférable d'utiliser dans la pratique une interface fluide-structure conforme. En effet, cette configuration nous permettra d'obtenir une meilleure prise en compte de ce couplage. Cependant, il peut être parfois nécessaire d'utiliser des méthodes non conformes à cause de l'imposition de maillage. Cela pourra aussi être un choix si l'on ne veut pas raffiner un des maillages fluide et structure pour pouvoir optimiser les performances du code de calcul. Cependant, il faudra vérifier que les erreurs d'interpolation commises pourront être négligées dans la modélisation. En particulier, ce dernier point pourra être proposé pour l'application que nous présenterons dans la section suivante à cause de l'épaisseur très fine de la structure.

3 Application numérique

Pour vérifier l'implémentation de notre modèle d'interaction fluide-structure, nous allons mettre en oeuvre une série de benchmarks trouvée dans la littérature. Le chapitre 9 sera consacré à la comparaison d'un problème fluide, d'un problème de structure et d'un problème d'interaction fluide-structure en 2D. Ces tests sont issus de [154]. Nous présenterons ensuite au chapitre 10 des benchmarks classiques d'application à des écoulements sanguins dans des tubes 2D et 3D. Toutefois, dans cette partie, nous proposons de vérifier dans un premier temps notre code de calcul FSI sur un benchmark 2D initialement proposé dans [119]. Cette application représente l'écoulement d'un fluide incompressible dans une cavité qui contient une paroi élastique. D'autres auteurs ont également testé ce benchmark comme [66, 156, 99, 94]. La validation de cette simulation sera basée sur la mesure du déplacement d'un point sur la paroi mobile.

3.1 Description du benchmark

Nous commençons par présenter la géométrie représentant la cavité avec la figure 6.3. Elle illustre les domaines initiaux et de référence pour le fluide et la structure. Le domaine Ω^*_f est décrit par le carré $[0,1]^2$ avec l'unité du mètre m. Nous pouvons trouver dans ce schéma l'entrée de l'écoulement avec $\Gamma^{i,*}_f$ et la sortie avec $\Gamma^{o,*}_f$. Nous avons aussi la frontière $\Gamma^{h,*}_f$ où la vitesse de l'écoulement sera supposée constante dans la direction x. Les parois notées par $\Gamma^{f,*}_f$ imposeront une vitesse nulle, ce qui représentera une condition de non-glissement pour le fluide. Enfin, nous avons le domaine structure en bas qui interagit avec le fluide sur l'interface Γ^*_{fsi}. Cette structure est fixée sur les bords verticaux gauche et droit. Nous les avons notés $\Gamma^{f,*}_s$. Pour alléger légèrement les notations de la figure 6.3, seul le bord droit est représenté par cette notation.

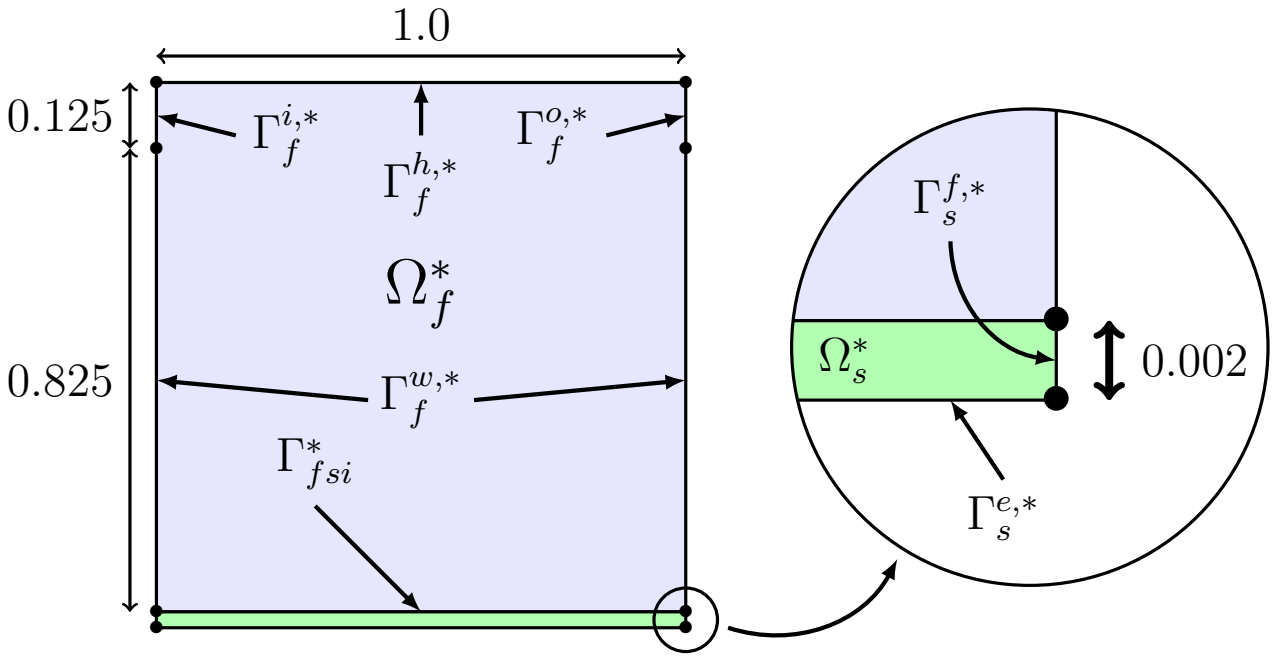


FIGURE 6.3 – Géométrie du problème FSI pour la simulation d'un écoulement dans une cavité contenant une paroi élastique.

Pour décrire formellement les conditions aux limites, nous considérons la vitesse oscillatoire suivante en fonction du temps dans la direction x :

$$v_{in} = 1 - \cos\left(\frac{2\pi t}{5}\right) \ m/s \tag{6.31}$$

Ainsi, nous avons l'expression des conditions aux limites de cette application :
- $\boldsymbol{u}_f = (v_{in}, 0)$ sur $\Gamma_f^{h,*}$
- $\boldsymbol{u}_f = (v_{in}(8y-7), 0)$ sur $\Gamma_f^{i,*}$
- $\boldsymbol{u}_f = \mathbf{0}$ sur $\Gamma_f^{f,*}$
- $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0}$ sur $\Gamma_f^{o,*}$
- $\boldsymbol{\eta}_s = \mathbf{0}$ sur $\Gamma_s^{f,*}$
- $\boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s = \mathbf{0}$ sur $\Gamma_s^{e,*}$

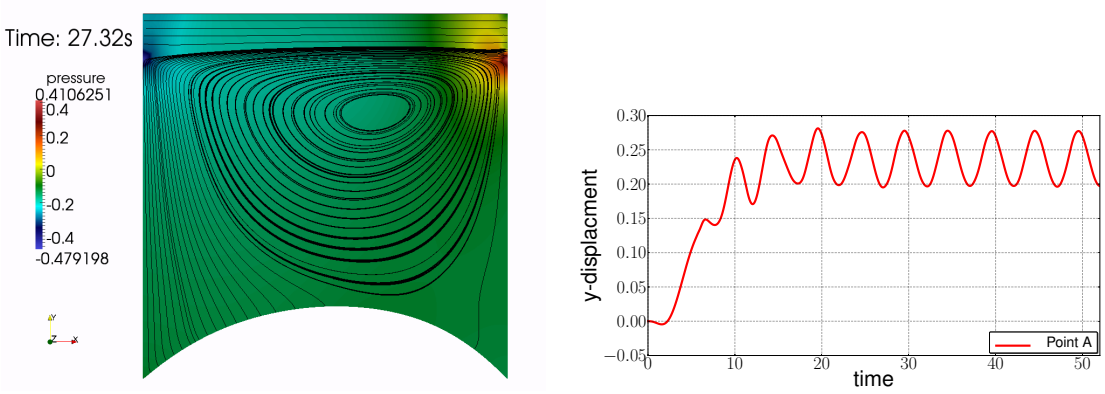
Les conditions aux limites sur l'interface fluide structure Γ_{fsi}^* correspondent aux conditions de couplage FSI présentées avec les équations (6.1)-(6.2)-(6.3).

Les propriétés physiques du fluide sont prises avec une densité $\rho_f = 1 \ Kg/m^3$ et une viscosité $\mu_f = 0.001 \ m^2/s$. Pour la structure, nous avons une densité $\rho_s^* = 500 \ Kg/m^3$, un module de Young $E_s = 250 \ N/m^2$ et un coefficient de Poisson $\nu_s = 0$.

3.2 Résultats

Pour les simulations qui ont été réalisées, nous avons utilisé le pas de temps $\Delta t = 0.01 \ s$. Pour vérifier le comportement du solveur FSI, nous avons mesuré le déplacement dans la direction y au point $\mathbf{A} = (0.5, 0)$. Celui-ci a pu être comparé à divers résultats trouvés dans la littérature.

Un certain nombre de configurations numériques a été utilisé pour résoudre cette application. Pour chacun de ces tests, l'ordre géométrique du domaine fluide était égal à l'ordre d'approximation du déplacement de la structure. Nous avons commencé par une approximation P2P1 pour le fluide avec une géométrie d'ordre 1 et une interface fluide structure conforme. Nous avons recommencé avec une approximation P3P2 et une géométrie d'ordre 2 sur un maillage équivalent. Les résultats de ces tests étaient très similaires. Nous avons relancé cette application avec une approximation P3P2 pour le fluide et une géométrie d'ordre 2. L'ordre polynomial d'approximation du déplacement de la structure restait toujours d'ordre 2 mais cette fois-ci le maillage de l'interface fluide structure devenait non conforme. Les performances de cette étude ont été bien meilleures que dans les cas conformes évoqués précédemment. On pouvait remarquer que la finesse de la paroi élastique engendrait un raffinement important du domaine fluide pour le cas conforme. Celui-ci devait assurer un meilleur contrôle du couplage FSI. Cependant, les résultats des tests non conformes sont très proches des mesures effectuées pour le cas conforme. Les illustrations présentées avec les figures 6.4(a) et 6.4(b) ont été faites avec cette configuration non conforme. Elles sont en accord (pour la norme de l'oeil) avec les résultats trouvés dans la littérature.



(a) Champ de pression avec les lignes de courant (b) Déplacement y du point **A** en fonction du temps

FIGURE 6.4 – Résultats numériques de l'application de la cavité entrainée avec une paroi élastique (approximation P3P2, géométrie d'ordre 2 et interface non conforme).

Ce test nous a permis une première vérification de notre modèle d'interaction fluide structure. Le contexte simulé reste toutefois « assez simple » dans le sens où $\mu_s >> \mu_f$. La non conformité de l'interface fluide structure n'a pas eu une grande influence sur la simulation de cette application. Cependant, cette caractéristique ne sera plus valable dans le contexte des écoulements sanguins.

4 Méthode de la frontière élargie avec interaction fluide-particule

Nous allons maintenant introduire une nouvelle méthode numérique motivée par l'idée de modéliser le transport d'une particule élastique B^t dans un fluide Ω_f, figure 6.5. Cette méthode est basée sur la FBM présentée au chapitre 3 couplée au modèle d'interaction

fluide-structure présenté dans ce chapitre. Toutefois, la technique qui va être présentée peut être considérée comme des expériences numériques. Une étude théorique serait nécessaire pour valider complètement la méthodologie. Nous nommons ce modèle FBM-FSI.

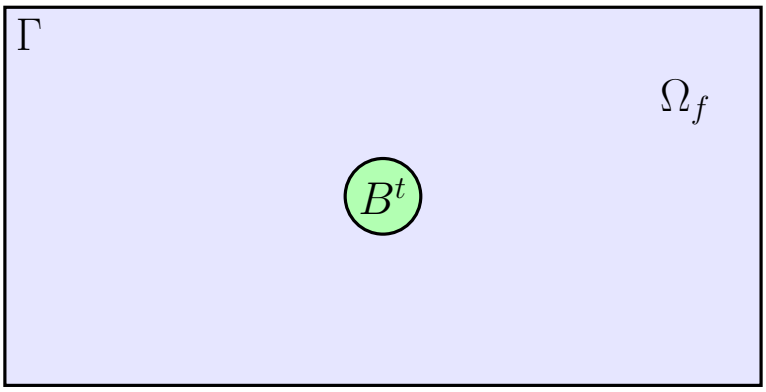


FIGURE 6.5 – Exemple de géométrie pour la modélisation du transport d'une particule déformable dans une écoulement de fluide.

Nous rappelons que la méthode FBM permet de transformer un problème défini sur un domaine perforé en plusieurs problèmes couplés de formes plus simples. Nous avons un problème global qui représente l'ensemble du domaine sans les perforations et des problèmes locaux situés autour des perforations. Nous souhaitons ici modéliser des écoulements de fluide contenant des particules rigides ou déformables. On utilise les équations de Navier-Stokes pour modéliser le fluide. On fait l'hypothèse que le domaine de l'écoulement fluide (domaine global) n'est pas mobile pour simplifier la description. Les perforations du domaine initial représentent ces particules en mouvement. Nous utilisons un modèle de mécanique des fluides dans le problème global et les problèmes locaux. Nous ajouterons ensuite des modèles élastiques sur chaque perforation. On parlera des problèmes de structure. L'interaction fluide-structure aura lieu entre les problèmes locaux du fluide et les problèmes de structure. Ce couplage a pour but de mettre en mouvement les particules. Cela induit également que les domaines locaux seront en mouvement, car ils doivent suivre les particules. Ainsi, nous devrons utiliser un modèle de fluide en domaine mobile pour ces problèmes locaux. De plus, nous aurons des conditions de Dirichlet non-homogènes sur l'interface fluide-particule. Il nous faudra donc également ajouter un problème de mécanique des fluides en domaine mobile sur la perforation.

Pour les notations géométriques, nous utilisons celles données dans la figure 6.6 qui représente les domaines à un instant t. On considérera uniquement une seule particule pour simplifier les notations. On note Ω_f le domaine fluide global qui est fixe au cours du temps. On admet que la frontière Γ est décomposée en deux parties disjointes notées Γ_D et Γ_N. Nous imposons une condition de Dirichlet $\boldsymbol{u}_f = \boldsymbol{g}$ sur la frontière Γ_D et une condition $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0}$ sur Γ_N. Nous trouvons aussi les perforations B^t qui sont entourées du domaine fictif ω_f^t. Les conditions de couplage pour l'interaction fluide structure sont représentées par les équations (6.1), (6.2) et (6.3).

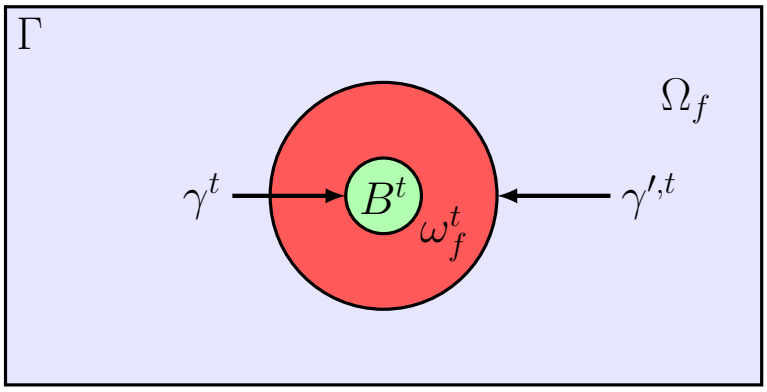


FIGURE 6.6 – Description des géométries utilisées pour le modèle FBM-FSI.

4.1 Formulation mathématique

Nous allons commencer par présenter les notations utilisées pour décrire la formulation FBM-FSI. Nous reprenons une grande partie des notations utilisées dans la chapitre 3. On note $(\boldsymbol{u}_f, p_f)$ (resp $(\hat{\boldsymbol{u}}_f, \hat{p}_f)$) le couple vitesse-pression sur le domaine Ω_f (resp ω_f^t). Nous avons aussi besoin du champ de vitesse $\mathring{\boldsymbol{u}}_f$ et du champ de pression $\mathring{p}_f$ défini sur la perforation B^t. Les domaines ω_f^t et B^t sont en mouvement au cours du temps. Nous utilisons le calcul d'une carte ALE sur ces domaines pour pouvoir les déplacer et intégrer ce mouvement dans la dynamique du fluide. Nous notons $\hat{\mathcal{A}}_f^t$ (resp $\mathring{\mathcal{A}}_f^t$) la carte ALE associée au domaine ω_f^t (resp B^t). Pour définir ces transformations, nous considérons les domaines de référence ω_f^* et B^* correspondant à l'état initial du problème. Ils vérifient $B^t = \mathring{\mathcal{A}}_f^t(B^*)$ et $\omega_f^t = \hat{\mathcal{A}}_f^t(\omega_f^*)$. Le domaine B^* sera également utilisé pour exprimer le modèle de structure de la particule. Cette approche lagrangienne consiste à trouver le déplacement $\boldsymbol{\eta}_s$. Nous nous placerons aussi dans l'hypothèse d'incompressibilité de la particule, ce qui nécessitera le calcul d'une pression p_s.

Nous décrivons maintenant les formulations variationnelles qui vont être utilisées dans la méthode FBM-FSI. Nous commençons par le problème FBM qui couple la dynamique des fluides dans le domaine global et le domaine local. Nous décrivons ici l'approche monolithique, mais la formulation point fixe est aussi adaptée (voir chapitre 3). Ce problème dépend de la solution du problème fluide sur B^t avec l'apparition du tenseur des contraintes $\mathring{\boldsymbol{\sigma}}_f$. Il dépend également de la vitesse de déformation de la particule avec :

$$\tilde{\boldsymbol{w}}_f = \dot{\boldsymbol{\eta}}_s \circ \left(\hat{\mathcal{A}}_f^t\right)^{-1}\Big|_{\gamma^t} :$$

Pour un instant t, la formulation variationnelle du problème FBM-FSI est décrite par le problème suivant :

Trouver $(\boldsymbol{u}, p, \hat{\boldsymbol{u}}, \hat{p}, \bar{p}, \boldsymbol{\lambda}, \boldsymbol{\phi}) \in \mathbf{H}^1_{\boldsymbol{g},\Gamma_D}(\Omega_f) \times L^2(\Omega_f) \times \mathbf{H}^1\left(\omega_f^t\right) \times L^2\left(\omega_f^t\right) \times \mathbb{P}_0\left(\omega_f^t\right) \times \mathbf{H}^{-1/2}\left(\gamma_f^t\right) \times \mathbf{H}^{-1/2}\left(\gamma_f'^{,t}\right)$ telles que $\forall \boldsymbol{v} \in \mathbf{H}^1_{0,\Gamma_D}(\Omega_f)$, $\forall q \in L^2(\Omega_f)$, $\forall \hat{\boldsymbol{v}} \in \mathbf{H}^1\left(\omega_f^t\right)$, $\forall \hat{q} \in$

$L^2\left(\omega_f^t\right)$, $\forall \bar{q} \in \mathbb{P}_0\left(\omega_f^t\right)$, $\forall \boldsymbol{\mu} \in \mathbf{H}^{-1/2}\left(\gamma^t\right)$, $\forall \boldsymbol{\psi} \in \mathbf{H}^{-1/2}\left(\gamma'^{,t}\right)$:

$$\rho_f \int_{\Omega_f} \left.\frac{\partial \boldsymbol{u}_f}{\partial t}\right|_{\mathbf{x}} \cdot \boldsymbol{v} + \rho_f \int_{\Omega_f} (\boldsymbol{u}_f \cdot \nabla)\boldsymbol{u}_f \cdot \boldsymbol{v} + \int_{\Omega_f} \boldsymbol{\sigma}_f : \nabla \boldsymbol{v} - \int_{\gamma^t} \boldsymbol{\lambda} \cdot \boldsymbol{v} = \int_{\Omega_f} \boldsymbol{f}_f \cdot \boldsymbol{v} + \int_{\gamma^t} (\mathring{\boldsymbol{\sigma}}_f \boldsymbol{n}_B) \cdot \boldsymbol{v} \quad (6.32)$$

$$\int_{\Omega_f} q \, \nabla \cdot \boldsymbol{u}_f = 0 \quad (6.33)$$

$$\rho_f \int_{\omega_f^t} \left.\frac{\partial \hat{\boldsymbol{u}}_f}{\partial t}\right|_{\mathbf{x}^*} \cdot \hat{\boldsymbol{v}} + \rho_f \int_{\omega_f^t} \left((\hat{\boldsymbol{u}}_f - \hat{\boldsymbol{w}}_f) \cdot \nabla\right) \hat{\boldsymbol{u}}_f \cdot \hat{\boldsymbol{v}} + \int_{\omega_f^t} \hat{\boldsymbol{\sigma}}_f : \nabla \hat{\boldsymbol{v}} - \int_{\gamma^t} \boldsymbol{\lambda} \cdot \hat{\boldsymbol{v}} - \int_{\gamma'^{,t}} \boldsymbol{\phi} \cdot \hat{\boldsymbol{v}} = \int_{\omega_f^t} \boldsymbol{f}_f \cdot \hat{\boldsymbol{v}} \quad (6.34)$$

$$\int_{\omega_f^t} \hat{q} \, \nabla \cdot \hat{\boldsymbol{u}}_f - \int_{\omega_f^t} \bar{p}_f \hat{q} = 0 \quad (6.35)$$

$$\int_{\omega_f^t} p_f \bar{q} - \int_{\omega} \hat{p}_f \bar{q} = 0 \quad (6.36)$$

$$\int_{\gamma_f^t} \hat{\boldsymbol{u}}_f \cdot \boldsymbol{\mu} = \tilde{\boldsymbol{w}}_f \quad (6.37)$$

$$\int_{\gamma'^{,t}} (\boldsymbol{u}_f - \hat{\boldsymbol{u}}_f) \cdot \boldsymbol{\psi} = 0 \quad (6.38)$$

Comme nous avons une condition de Dirichlet non homogène sur l'interface γ^t, nous devons résoudre un problème de Navier-Stokes sur la particule. Celui-ci est défini à l'aide de la vitesse $\tilde{\boldsymbol{w}}_f$ sur γ^t et de la pression p_f sur B^t. Nous pouvons écrire ce problème par le système d'équation suivant :

Trouver $(\mathring{\boldsymbol{u}}_f, \mathring{p}_f, \bar{r}_f) \in \mathbf{H}^1_{\tilde{\boldsymbol{w}}_f, \gamma^t}\left(B^t\right) \times L^2\left(B^t\right) \times \mathbb{P}_0\left(B^t\right)$ tel que $\forall (\boldsymbol{v}, q, \bar{q}) \in \mathbf{H}^1_{\mathbf{0}, \gamma^t}\left(B^t\right) \times L^2\left(B^t\right) \times \mathbb{P}_0\left(B^t\right)$:

$$\rho_f \int_{B^t} \left.\frac{\partial \mathring{\boldsymbol{u}}_f}{\partial t}\right|_{\mathbf{x}^*} \cdot \boldsymbol{v} + \rho_f \int_{B^t} \left((\mathring{\boldsymbol{u}}_f - \mathring{\boldsymbol{w}}_f) \cdot \nabla\right) \mathring{\boldsymbol{u}}_f \cdot \boldsymbol{v} + \int_{B^t} \mathring{\boldsymbol{\sigma}}_f : \nabla \boldsymbol{v} = \int_{B^t} \boldsymbol{f}_f \cdot \boldsymbol{v} \quad (6.39)$$

$$\int_{B^t} q \, \nabla \cdot \mathring{\boldsymbol{u}}_f - \int_{B^t} \bar{r}_f \; q = 0 \quad (6.40)$$

$$\int_{B^t} \mathring{p}_f \; \bar{q} = \int_{B^t} p_f \; \bar{q} \quad (6.41)$$

Enfin, nous devons résoudre un problème d'hyperélasticité sur la particule pour modéliser la déformation de celle-ci. Tout d'abord, on pose $\hat{\hat{\boldsymbol{\sigma}}}_f = \hat{\boldsymbol{\sigma}}_f \circ \hat{\mathcal{A}}_f^t$ qui représente la force hydrodynamique exercée sur la frontière extérieure de la particule. Le modèle de structure de la particule est défini à l'aide du problème suivant :

Trouver $(\boldsymbol{\eta}_s, p_s) \in \mathbf{H}^1(B^*) \times L^2(B^*)$ tel que $\forall (\boldsymbol{\xi}, q) \in \mathbf{H}^1(B^*) \times L^2(B^*)$:

$$\int_{B^*} \rho_s^* \frac{\partial^2 \boldsymbol{\eta}_s}{\partial t^2} \cdot \boldsymbol{\xi} + \int_{B^*} (\boldsymbol{F}_s \boldsymbol{\Sigma}_s) : \nabla \boldsymbol{\xi} = \int_{B^*} \boldsymbol{f}_s^t \cdot \boldsymbol{\xi} + \int_{\gamma^*} J_{\hat{\mathcal{A}}_f^t} \boldsymbol{F}_{\hat{\mathcal{A}}_f^t}^{-T} \hat{\boldsymbol{\sigma}}_f \boldsymbol{n}_B^* \cdot \boldsymbol{\xi} \tag{6.42}$$

$$\int_{B^*} (J_s - 1)\ q = 0 \tag{6.43}$$

Après avoir décrit les formulations variationnelles des principaux problèmes qui sont résolus dans la méthode FBM-FSI, nous présentons l'algorithme 6 qui nous permet de résoudre le problème complet. Il se base sur une stratégie de couplage implicite pour l'interaction fluide-structure. Une version semi-implicite a également été implémentée. Cette dernière est plus performante, car le domaine ne se déplace pas lors des itérations de l'algorithme point fixe. Ainsi, les opérateurs d'interpolation et les termes non standards ne sont pas recalculés.

Algorithm 6 FBM Solveur avec couplage FSI implicite

for $n = 0$ à $N_T - 1$ **do**
  $\boldsymbol{\eta}_{s,\delta}^{0,n+1} = \textbf{PrédictionDéplacement}(\boldsymbol{\eta}_{s,\delta}^{n}, \boldsymbol{\eta}_{s,\delta}^{n-1}, ...)$
  $k = 1$
  while $\epsilon > TOL$ **do**
    $\mathring{\mathcal{A}}_{f,\delta}^{k,n+1} = \textbf{SolveurALE}(\boldsymbol{\eta}_{s,\delta}^{k-1,n+1})$
    $\hat{\mathcal{A}}_{f,\delta}^{k,n+1} = \textbf{SolveurALE}(\boldsymbol{\eta}_{s,\delta}^{k-1,n+1})$
    $\tilde{\boldsymbol{w}}_{f,\delta}^{k,n+1} = \dot{\boldsymbol{\eta}}_{s,\delta}^{k-1,n+1} \circ \left(\hat{\mathcal{A}}_{f,\delta}^{k,n+1}\right)^{-1} \Big|_{\gamma^t}$
    $(\mathring{\boldsymbol{u}}_{f,\delta}^{k,n+1}, \mathring{p}_{f,\delta}^{k,n+1}) = \textbf{SolveurPerfo}(\ \tilde{\boldsymbol{w}}_{f,\delta}^{k,n+1}, p_{f,\delta}^{k-1,n+1})$
    $(\boldsymbol{u}_{f,\delta}^{k,n+1}, p_{f,\delta}^{k,n+1}, \hat{\boldsymbol{u}}_{f,\delta}^{k,n+1}, \hat{p}_{f,\delta}^{k,n+1}) = \textbf{SolveurFBM}(\hat{\mathcal{A}}_{f,\delta}^{k,n+1}, \tilde{\boldsymbol{w}}_{f,\delta}^{k,n+1}, \mathring{\boldsymbol{\sigma}}_{f,\delta}^{k,n+1})$
    $\hat{\boldsymbol{\sigma}}_{f,\delta}^{k,n+1} = \hat{\boldsymbol{\sigma}}_{f,\delta}^{k,n+1} \circ \hat{\mathcal{A}}_{f,\delta}^{k,n+1}$
    $\tilde{\boldsymbol{\eta}}_{s,\delta}^{k} = \textbf{SolveurStructure}(\hat{\boldsymbol{\sigma}}_{f,\delta}^{k,n+1}, \hat{\mathcal{A}}_{f,\delta}^{k,n+1})$
    $\epsilon = \textbf{CalculConvergence}(\tilde{\boldsymbol{\eta}}_{s,\delta}^{k}, \boldsymbol{\eta}_{s,\delta}^{k-1,n+1})$
    if $\epsilon > TOL$ **then**
      $\theta^k = \textbf{CalculParamètreRelaxation}(\tilde{\boldsymbol{\eta}}_{s,\delta}^{k}, \boldsymbol{\eta}_{s,\delta}^{k-1,n+1}, \theta^{k-1})$
      $\boldsymbol{\eta}_{s,\delta}^{k,n+1} = \theta^k \tilde{\boldsymbol{\eta}}_{s,\delta}^{k} + (1 - \theta^k) \boldsymbol{\eta}_{s,\delta}^{k-1,n+1}$
      $k = k + 1$
    end if
  end while
  $\boldsymbol{u}_{f,\delta}^{n+1} = \boldsymbol{u}_{f,\delta}^{k,n+1}$, $p_{f,\delta}^{n+1} = p_{f,\delta}^{k,n+1}$, $\hat{\boldsymbol{u}}_{f,\delta}^{n+1} = \hat{\boldsymbol{u}}_{f,\delta}^{k,n+1}$, $\hat{p}_{f,\delta}^{n+1} = \hat{p}_{f,\delta}^{k,n+1}$
  $\hat{\mathcal{A}}_{f,\delta}^{n+1} = \hat{\mathcal{A}}_{f,\delta}^{k,n+1}$, $\mathring{\mathcal{A}}_{f,\delta}^{n+1} = \mathring{\mathcal{A}}_{f,\delta}^{k,n+1}$
  $\boldsymbol{\eta}_{s,\delta}^{n+1} = \tilde{\boldsymbol{\eta}}_{s,\delta}^{k}$
end for

4.2 Applications numériques

Nous présentons maintenant deux applications numériques qui ont pour but de tester notre méthode FBM-FSI. Nous souhaitons vérifier le comportement des solveurs et obtenir quelques résultats qualitatifs des solutions. On rappelle que cette méthode reste pour le

moment une expérimentation numérique et qu'une étude théorique de ce modèle serait nécessaire.

Écoulement en cisaillement avec particule

Dans cette première expérience, nous souhaitons modéliser le comportement d'une particule soumis à un écoulement en cisaillement. Le domaine global Ω_f est représenté par un rectangle $[0,3] \times [-0.5, 0.5]$. La particule B^t est de forme elliptique à son état initial B^{t_0}. Elle est centrée sur le point $(1.5, 0)$, son grand rayon R_B est défini par le point $(1.5, 0.1)$ et son petit rayon r_B par le point $(1.45, 0)$. La frontière fictive $\gamma'^{,t_0}$ est également une ellipse centrée sur le même point, son grand rayon est égale à $R_B + 0.05$ et son petit rayon est égale à $r_B + 0.05$. Le domaine local $\omega_f^{t_0}$ est ensuite défini par les deux frontières γ^{t_0} et $\gamma'^{,t_0}$. On peut donc voir ω_f comme un anneau en forme d'ellipse et d'épaisseur 0.05.

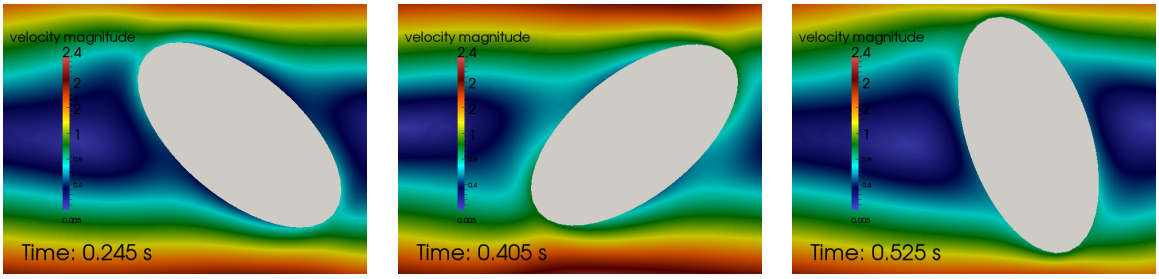
Nous utilisons le modèle de Navier-Stokes incompressible pour le fluide et un modèle hyperélastique incompressible avec une loi de Saint-Venant-Kirchhoff pour la structure. Le déplacement du domaine local ω_f^t est calculé avec les équations de Winslow et le déplacement de la particule avec l'opérateur d'extension harmonique. Au niveau des paramètres physiques, nous avons choisi $\rho_f = 1$, $\mu_f = 1$, $\rho_s^* = 500$, $E_s = 10^7$ et $\nu_s = 0.3$. Pour les conditions sur le domaine Ω_f, nous imposons :

- $\boldsymbol{\sigma_f n}_f = \mathbf{0}$ sur $(\{0\} \cup \{3\}) \times [-0.5, 0.5]$
- $\boldsymbol{u}_f = (-10, 0)$ sur $[0,3] \times \{0.5\}$
- $\boldsymbol{u}_f = (10, 0)$ sur $[0,3] \times \{-0.5\}$

Les autres conditions sont décrites dans le modèle FBM-FSI. Aussi, nous n'utilisons pas de termes sources volumiques, c'est-à-dire $\boldsymbol{f}_f^t = \boldsymbol{f}_s^t = \mathbf{0}$.

Les illustrations se trouvant dans la figure 6.7 représentent trois captures d'écran de la magnitude du champ de vitesse faite à trois instants. Ces résultats ont été obtenus avec une approximation P2P1 pour les problèmes fluides (global, local et perforation). L'ordre polynomial d'approximation utilisé pour le modèle de structure est également P2P1. Nous avons pris un pas de temps $\Delta t = 0.005$ s. La simulation nous montre alors que la particule se met en mouvement de rotation en gardant un centre de rotation fixe. En réalisant la simulation jusqu'à $t = 10$ s, nous avons pu effectuer plusieurs rotations qui illustrent la périodicité de ce mouvement. Nous obtenons ainsi le résultat qualitatif attendu par cette application. Les figures 6.7(a), 6.7(b) et 6.7(c) montre ce mouvement pour la première demi-rotation. Au niveau de l'algorithme de l'interaction fluide particule, le nombre d'itérations moyen est de 4 pour obtenir la convergence avec un critère d'arrêt $\epsilon = 10^{-6}$. Ce faible nombre d'itérations est principalement dû au fait que la particule est quasiment rigide.

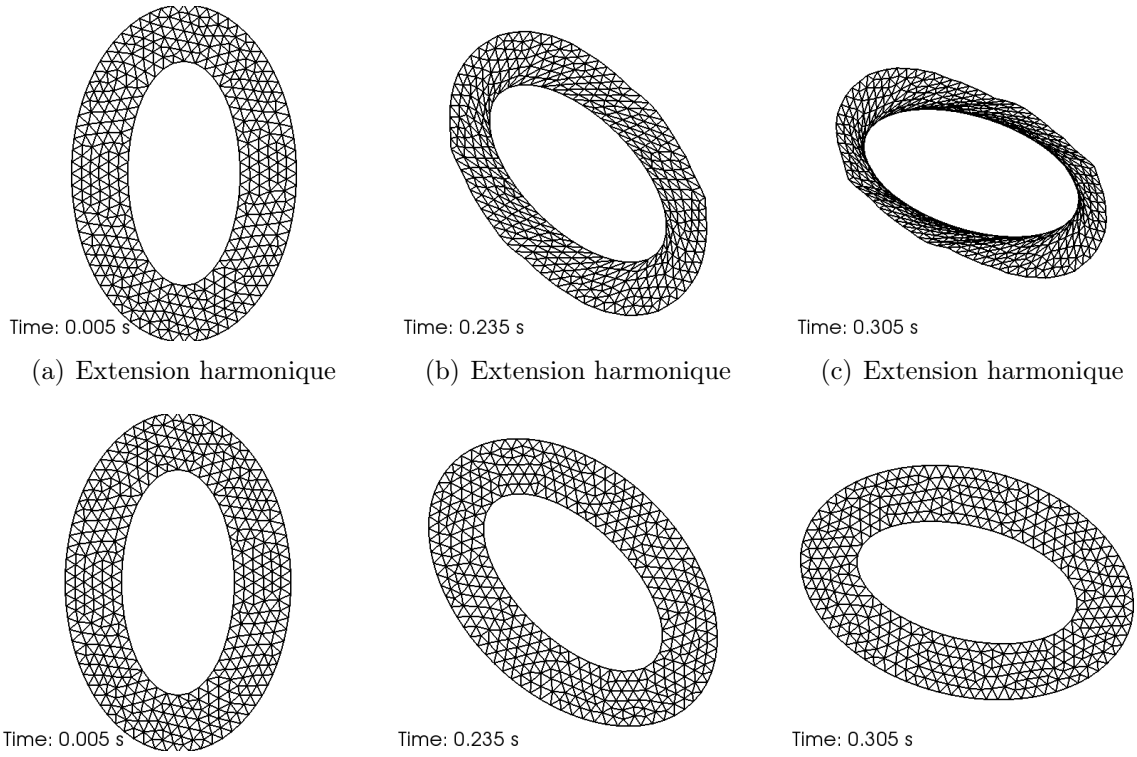
Nous avons utilisé les équations de Winslow (voir chapitre 2) pour le déplacement du domaine local ω_f. En faite, ce choix est crucial pour le déplacement rotatif de cette forme d'anneau. En effet, l'approche classique qui consiste à utiliser le calcul de l'extension harmonique (voir chapitre 2) n'est pas fonctionnelle pour cette application. Nous avons illustré avec la figure 6.8 la comparaison entre l'extension harmonique et les équations de Winslow pour ce mouvement du domaine local. Les figures 6.8(a) et 6.8(d) représente le maillage local au premier pas de temps pour chacune des deux méthodes. On constate ensuite que la méthode utilisant l'extension harmonique génère une détérioration de la qualité du maillage déformé (figure 6.8(b)) jusqu'à l'obtention d'un maillage invalide (figure 6.8(c)). Cette dernière figure montre le dernier pas de temps effectué avant que notre



(a) t=0.245 s (b) t=0.405 s (c) t=0.525 s

FIGURE 6.7 – Magnitude de la vitesse du fluide à trois instants pour la modélisation d'une particule dans un écoulement en cisaillement.

programme « crash ». Par contre, on constate la qualité de la déformation obtenue avec les équations de Winslow (figure 6.8(e) et 6.8(f)). Elles montrent ainsi les très bonnes propriétés de cette méthode et cette qualité est conservée pour les multiples rotations de cet objet. Pour le déplacement de la particule B^t, l'utilisation de l'extension harmonique est suffisante pour cette application satisfaisante. Cependant, si en plus de cette rotation la particule subissait des déformations beaucoup plus importantes, l'utilisation des équations de Winslow devrait être envisagée.



(a) Extension harmonique (b) Extension harmonique (c) Extension harmonique

(d) Winslow (e) Winslow (f) Winslow

FIGURE 6.8 – Comparaison du déplacement du domaine local entre l'extension harmonique (en haut) et les équations de Winslow (en bas).

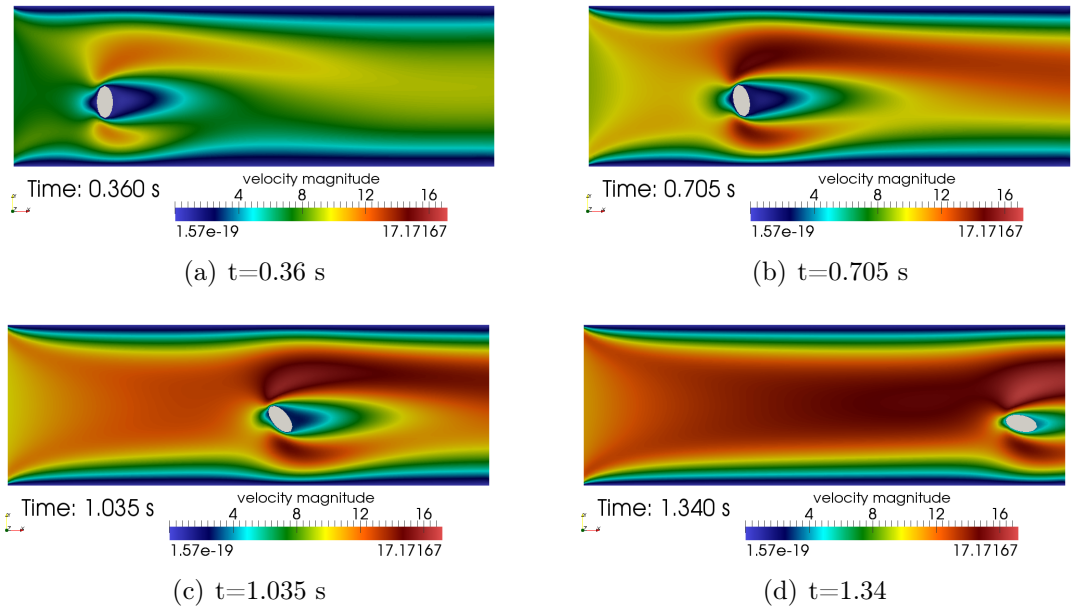
Écoulement d'une particule dans un canal rectangulaire

Cette seconde application a pour but de modéliser le transport d'une particule dans un canal rectangulaire $\Omega_f = [0, 3] \times [-0.5, 0.5]$. La particule B^t est de forme elliptique à son état initial B^{t_0}. Elle est centrée sur le point $(0.5, -0.1)$, son grand rayon R_B est défini par le point $(0.5, 0)$ et son petit rayon r_B par le point $(0.45, -0.1)$. La frontière fictive $\gamma'^{,t_0}$ est également une ellipse centrée sur le même point, son grand rayon est égale à $R_B + 0.1$ et son petit rayon est égale à $r_B + 0.1$.

Les équations de Navier-Stokes incompressible sont choisies comme modèle pour le fluide. La déformation de la particule est basée un modèle hyperélastique incompressible avec une loi de Saint-Venant-Kirchhoff. Comme pour l'application précédente, nous utilisons les équations de Winslow pour le déplacement du domaine local ω_f^t et l'opérateur d'extension harmonique pour la particule. Au niveau des paramètres physiques, nous avons pris $\rho_f = 1$, $\mu_f = 0.1$, $\rho_s^* = 500$, $E_s = 10^7$ et $\nu_s = 0.3$. Pour les conditions sur le domaine Ω_f, nous avons celles du modèle FBM-FSI auxquelles on ajoute :

- $\boldsymbol{\sigma_f n}_f = -100\boldsymbol{n}_f$ sur $\{0\} \times [-0.5, 0.5]$
- $\boldsymbol{\sigma_f n}_f = \mathbf{0}$ sur $\{3\} \times [-0.5, 0.5]$
- $\boldsymbol{u}_f = \mathbf{0}$ sur $[-0.5, 0.5] \times (\{-0.5\} \cup \{-0.5\})$

Nous avons aussi $\boldsymbol{f}_f^t = \boldsymbol{f}_s^t = \mathbf{0}$.



(a) t=0.36 s (b) t=0.705 s

(c) t=1.035 s (d) t=1.34

FIGURE 6.9 – Magnitude du champ de vitesse du fluide à différents instants pour l'écoulement de la particule dans un canal.

L'approximation polynomiale P2P1 a été choisie pour tous les problèmes fluides et pour la structure aussi. Le pas de temps utilisé est $\Delta t = 0.005\ s$. Avec la figure 6.9, nous avons affiché plusieurs instants de la simulation avec l'affichage de la magnitude du champ de vitesse. On constate le déplacement de la particule dans le canal. Du fait de la position décentrée de la particule au départ, celle-ci subit une rotation lors de son déplacement.

D'un point de vue qualitatif, les solutions obtenues semblent cohérentes avec la physique sous-jacente. Cependant, pour valider réellement notre modèle FBM-FSI, il faudrait

comparer nos résultats avec d'autres méthodes ou bien trouver une application basée sur une expérimentation physique. Ces applications nous ont permis de vérifier le comportement numérique de notre solveur FBM-FSI et de montrer les capacités de cette méthode pour la modélisation de particules immergées dans un fluide.

Troisième partie

Contributions informatiques à Feel++

Chapitre 7

Calcul haute performance

Les simulations numériques nécessitent un très grand nombre de calculs qui augmentent avec la complexité des modèles, du domaine d'étude et des méthodes mathématiques mises en jeu. Elles peuvent durer plusieurs heures, plusieurs jours, voire plusieurs semaines de calcul. C'est pourquoi l'optimisation de codes de calcul à l'aide de l'exploitation des nouvelles architectures parallèles devient un passage incontournable pour le calcul scientifique. On parle alors de calcul haute performance.

Les architectures parallèles sont devenues le paradigme dominant pour tous les ordinateurs depuis ces dernières années. Les processeurs sont devenus multicoeurs, traitant ainsi plusieurs instructions simultanément. Les architectures hybrides couplant des processeurs traditionnels à des processeurs spécialisés (GPU) ont également fait leur apparition et elles ont permis d'accélérer encore certains calculs. Le calcul sur ces nouvelles architectures spécialisées est appelé GPGPU (*General-Purpose computation on Graphics Processing Units*). Tout ceci a permis l'évolution récente des supercalculateurs qui s'orientent désormais vers un parallélisme massif avec un nombre de coeur de plus en plus grand. Un supercalculateur contient de nombreux noeuds de calcul qui sont reliés entre eux par un réseau. Chaque noeud peut contenir plusieurs coeurs et processeurs spécialisés. Avec la

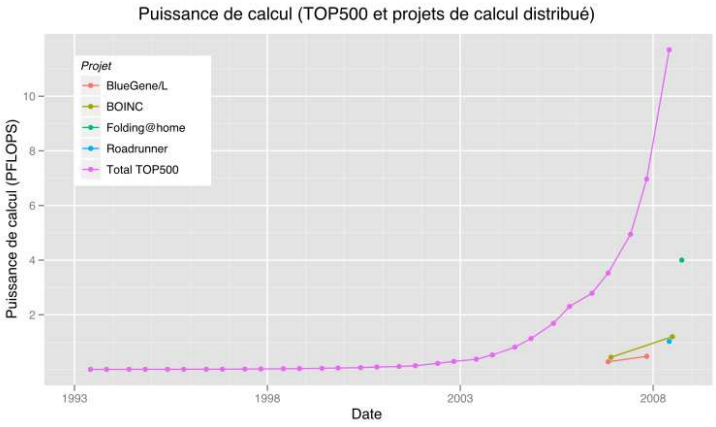


FIGURE 7.1 – Évolution de la puissance des supercalculateurs (image issue de http ://fr.wikipedia.org/wiki/Superordinateur).

figure 7.1, nous pouvons voir l'évolution de la puissance des supercalculateurs (de 1993 à 2008) et on constate l'explosion de la puissance depuis ces dix dernières années. En 2012,

le supercalculateur le plus puissant du monde se trouve aux États-Unis. Cette machine se nomme Titan et elle a été développée par Cray avec une puissance de 17.59 petaFLOPS et 560640 processeurs. Les systèmes de calcul pétaflopiques se présentent comme des architectures massivement parallèles hétérogènes. Ces systèmes sont basés sur des noeuds à architecture mémoire non uniforme (NUMA) interconnectés par des réseaux rapides, chaque noeud s'organisant autour d'un ensemble d'unités de calcul hétérogène : cartes multicoeurs, processeurs graphiques (GPU) ou autres types de carte accélératrice.

Pour tirer le meilleur profit de ces supercalculateurs, il est nécessaire de développer des logiciels capables d'exploiter efficacement ces architectures parallèles. Dans ce but, nous allons présenter la stratégie de calcul haute performance que nous souhaitons employer dans la librairie Feel++ [129, 130, 131] spécialisée dans la résolution d'EDP par des méthodes de Galerkin. Nous voulons utiliser trois technologies de calcul parallèle : MPI, Multithreading (MT) et GPGPU. Ces différents niveaux de parallélisation sont présentés dans la figure 7.2. Nous commençons par répartir le découpage du domaine de calcul sur les processeurs à mémoire distribuée. La couche MPI nous permet de relier les sous-domaines en communiquant des informations entre ces processeurs. Ensuite nous pouvons accélérer certaines parties de nos calculs à l'aide de processeurs multicoeurs et de la technologie Multithreading. La présence de carte GPU permettrait également d'accélérer des algorithmes de calcul.

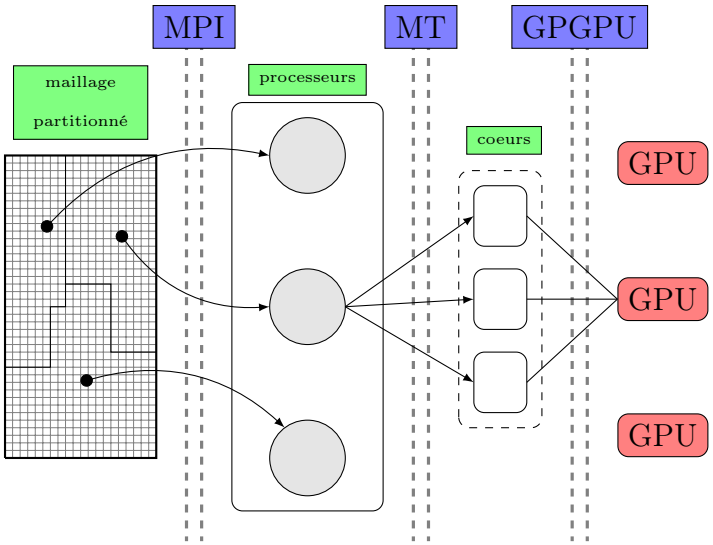


Figure 7.2 – Stratégie de calcul haute performance dans Feel++.

Dans ce chapitre, nous allons aborder deux types de parallélisation. Tout d'abord, nous présenterons notre méthode de distribution des calculs avec la technologie MPI à partir du partitionnement du domaine de calcul. Nous montrerons également des tests de scalabilité sur des exemples réalistes de mécanique des fluides et des solides. Enfin, nous illustrerons les possibilités d'accélération des cartes GPU sur une partie de notre code éléments finis, l'assemblage des structures algébriques.

1 Décomposition de domaine algébrique

Dans cette section, nous allons décrire la méthodologie employée pour paralléliser la librairie Feel++. Cette description sera valable pour la résolution d'EDP à l'aide de méthodes de Galerkin standards. De plus, l'implémentation parallèle sera totalement transparente pour un simple utilisateur. Toutes les communications MPI seront cachées dans la librairie. Nous commencerons cette partie par le partitionnement de maillage permettant ainsi de distribuer le problème sur plusieurs processeurs. Nous décrirons ensuite la parallélisation de la table des degrés de liberté et de la construction des espaces de fonctions. Nous finirons avec l'interface qui a été réalisée pour accéder aux fonctionnalités de la librairie PETSc comme les structures et solveurs algébriques parallèles.

1.1 Partitionement du maillage

Soit Ω_δ un maillage quelconque. La première étape de parallélisation est le découpage du domaine de calcul Ω_δ. Chaque processeur doit avoir accès seulement à une partie du maillage. Ainsi nous définissons Ω_δ^p la partition associée au processeur p. Nous notons également par Γ_{IP}^p la frontière de Ω_δ^p entre les sous-domaines nommée *«frontière interprocessus»*. La connexion entre les différents sous-domaines s'effectuera par l'ajout d'éléments dits *«éléments fantômes»* pour chaque processus. Les éléments fantômes sont les éléments du maillage ayant une face ou une arête ou un point sur Γ_{IP}^p et n'appartenant pas au domaine de calcul du processus. Les éléments non-fantômes appartenant à la partition sont dits *«éléments réels»* ou *«éléments actifs»*. Cette subdivision permet donc de réduire la mémoire vive locale utilisée par chaque processeur comparé au cas séquentiel. Mais évidemment, la mémoire globale sur l'ensemble des processus sera supérieure, à cause de l'ajout des éléments fantômes.

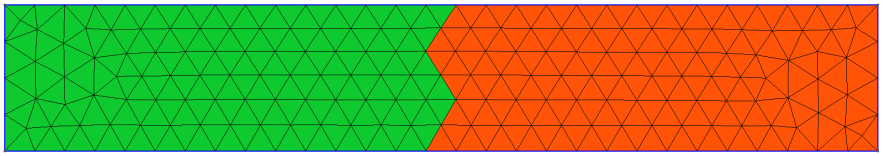
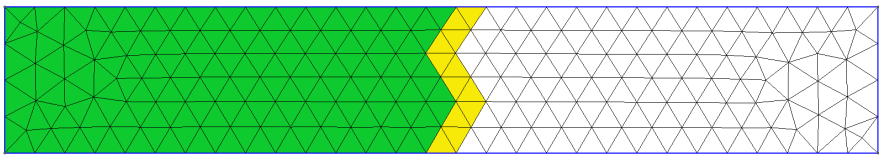


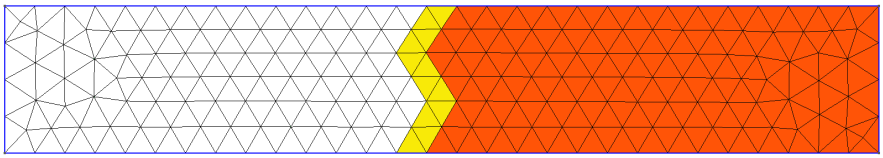
Figure 7.3 – Exemple de partition d'un maillage 2D.

Nous avons choisi d'utiliser les partitionneurs 2D et 3D fournis par Gmsh [67] qui sont Chaco [80] et Metis [93]. Un exemple de partitionnement est montré avec la figure 7.3. Nous avons mis en évidence (en jaune) les éléments fantômes pour chacune des partitions, figures 7.4(a) et 7.4(b). Gmsh nous permet donc de générer et de partitionner le maillage, mais malheureusement ces opérations sont purement séquentielles pour le moment. Une fois fait, Gmsh nous crée un fichier .msh qui contient toutes les informations relatives aux points, faces et éléments contenus dans le maillage Ω_δ. Les points se trouvent dans la partie \$Nodes. Les faces et les éléments dans la partie \$Elements de ce fichier .msh Les informations liées au partitionnement se trouvent dans cette dernière partie.

Au niveau de Feel++, la création du maillage Ω_δ^p associé au processeur p s'effectue par la lecture du fichier .msh. Cette lecture est faite simultanément par tous les processeurs. Tout d'abord, tous les processeurs vont enregistrer l'ensemble des points (partie \$Nodes du fichier). Ensuite, chaque processeur va stocker les éléments ou faces qui appartiennent



(a) proc0



(b) proc1

FIGURE 7.4 – Partition d'un maillage avec les cellules fantômes.

à leur partition ou qui sont fantômes. Par exemple, dans la partie $Elements du fichier .msh, nous pouvons trouver cette ligne :

```
14 2 7 0 1 4 3 -1 -2 -4 1 2 5
```

Nous décrivons cette ligne de gauche à droite. Les deux premiers nombres signifient que cette ligne correspond au $14^{\text{ème}}$ élément du maillage de Ω_δ et qu'il est de type 2, c'est-à-dire un triangle. Ensuite, le 7 nous donne le nombre de *tag* disponible pour cet élément. Les deux nombres suivants sont des marqueurs. Puis nous avons les informations liées au partitionnement. Le 4 dit que l'élément appartient à quatre partitions en tant qu'élément réel ou fantôme. Le premier nombre (positif) est l'indice de partition pour l'élément réel. Les trois autres nombres (négatifs) correspondent aux indices de partition pour lequel cet élément est fantôme. Les trois derniers nombres donnent les points décrivant ce triangle. Une fois la lecture du fichier terminée, tous les éléments nécessaires à Ω_δ^p ont été ajoutés. Il reste à donner une information importante aux éléments fantômes qui est le numéro de processeur dans lequel cet élément est réel et aussi l'indice de cet élément réel. Cette opération nécessite des communications entre les processeurs. Enfin, les points ne se trouvant pas sur la partition du processeur sont supprimés.

Pour terminer le partitionnement, nous montrons un exemple de création de maillage parallèle avec le code ci-dessous. Il s'agit d'un rectangle partitionné en 2 morceaux et l'objet mesh a été construit avec la description faite précédemment.

```
GeoTool::Node x1(0,-0.5);
GeoTool::Node x2(6,0.5);
GeoTool::Rectangle R(meshSize,"Omega",x1,x2);
auto mesh = R.createMesh(_mesh=new mesh_type,_name="mymesh",
                         _partitions=2);
```

L'utilisation de l'outil `GeoTool` pour la création de maillage sera décrite en détail au chapitre 8.

1.2 Table des degrés de liberté parallèle

Nous allons maintenant présenter la construction de la table des degrés de liberté parallèle associée à un espace de fonction $P_N^c(\Omega_\delta)$ ou $P_N^d(\Omega_\delta)$. Cette table permet d'obtenir l'indice global du degré de liberté à partir d'un numéro de processeur p, d'un indice local

i associé à un élément $K_e \in \Omega_\delta^p$. Nous notons $n(e,i,p)$ un élément de cette table. La numération globale doit être unique, ce qui va nécessiter un effort de communication entre les processeurs voisins. C'est l'un des ingrédients clés pour la parallélisation complète de la librairie, elle représente la distribution des structures de données sur l'ensemble des processeurs et donc aura un rôle très important pour la suite.

Pour construire l'ensemble de la table $n(e,i,p)$, nous allons nous baser sur la construction séquentielle locale à chaque processeur p de la table des degrés de liberté, $n^p(e,i)$. Cet objet est propre à chacun des processeurs, il n'y a aucune relation entre eux, et elle nous donne une numérotation unique des degrés de liberté sur ce processeur. Le but ensuite est de construire une numérotation unique pour l'ensemble de tous les processeurs, en prenant également en compte la présence des degrés de liberté fantômes, c'est-à-dire n'appartenant pas au processeur, mais présents dans celui-ci. On peut remarquer que dans le cas de l'espace $P_N^d(\Omega_\delta)$, où l'espace de fonction est discontinu par élément, il n'y a pas de degrés de liberté fantômes. Dans le cas des fonctions continues, les degrés de liberté fantômes se trouvent sur la frontière interprocessus.

Dans la suite de cette section, nous parlerons de représentation locale, la table des degrés de liberté séquentielle qui est indépendante pour chaque processeur. Nous décrirons par représentation globale, la table commune à l'ensemble des processeurs et correspondant à la numérotation unique des degrés de liberté. Nous notons aussi $\mathcal{M}_j^p$ la carte reliant les degrés de liberté locaux aux degrés de liberté globaux. Nous avons alors la relation $n(e,i,p) = \mathcal{M}_j^p$ avec $j = n^p(e,i)$. Ainsi, notre méthode de construction de la table parallèle va être faite en deux étapes :

- Étape 1 : construction de la table locale séquentielle $n^p(e,i)$ (contenant les fantômes)
- Étape 2 : construction de la table globale $n(e,i,p)$ à l'aide de $\mathcal{M}_j^p$

L'étape 1 consiste donc simplement à une réutilisation d'une fonctionnalité séquentielle, et de plus il n'y a aucune communication entre les processus. Pour illustrer l'étape 2, nous avons représenté un exemple très simple avec les figures 7.5. On considère Ω_δ comme l'union de deux éléments. On considère qu'il est distribué sur deux processeurs. Chaque point numéroté représente les degrés de liberté (ddl). À gauche de cette figure, nous avons les tables représentant les états locaux et globaux des ddl. Ici, chaque processeur contient localement 6 ddl contrairement au niveau global où le processeur 1 contient 3 ddl contre 6 pour le processeur 0. Le processeur 1 possède donc 3 ddl fantômes dont l'indice global est donné par la carte $\mathcal{M}_j^p$.

- numérotation locale $n^p(e,i)$:

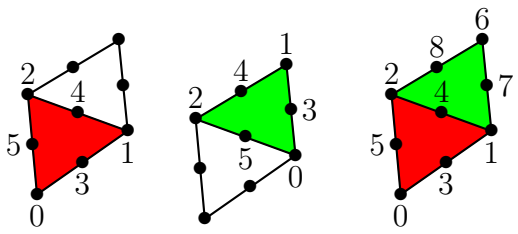
proc0	0	1	2	3	4	5
proc1	0	1	2	3	4	5

- numérotation globale $n(e,i,p)$:

proc0	0	1	2	3	4	5
proc1		6		7	8	

- carte $\mathcal{M}_j^p$ reliant indices locaux et globaux :

proc0	0	1	2	3	4	5
proc1	1	6	2	7	8	4



(a) proc0 (b) proc1 (c) global

FIGURE 7.5 – Représentation des degrés de liberté parallèles sur deux éléments.

Pour établir la carte des degrés de liberté globaux, nous avons tout d'abord besoin d'une règle permettant de savoir si un degré de liberté est réel ou bien fantôme. Pour une

partition Ω_δ^p, tous les degrés de liberté internes à ce maillage (ne se trouvant pas sur une frontière interprocessus) appartiennent au processeur p. Pour les ddl de la frontière interprocessus, nous choisissons de dire qu'un ddl de coordonnée $\boldsymbol{x}$ est réel pour sa partition Ω_δ^p si et seulement si $p = \min\left\{q, 1 \leq q \leq N_{proc} \text{ et } \boldsymbol{x} \in \Gamma_q^p\right\}$. Dans le cas contraire, ce ddl sera appelé ddl fantôme, la vision de ce degré de liberté sera uniquement locale. Cette règle qui a l'avantage d'être très simple n'équilibre pas parfaitement le nombre de degré de liberté dans chaque processus. On pourrait alors choisir d'autres règles qui pourraient par exemple contrôler le nombre de ddl locaux ou encore le nombre d'éléments. Avec la figure 7.6, nous pouvons voir la distribution des degrés de liberté sur trois processeurs appliquant cette règle.

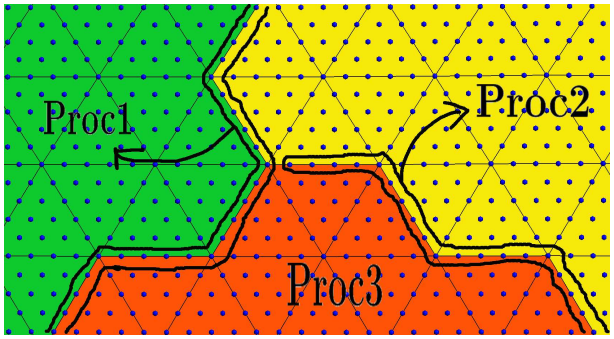


FIGURE 7.6 – Principe de la distribution des degrés de liberté fantômes.

La partie délicate de la parallélisation de la table des degrés de liberté est la construction de la numérotation globale avec unicité des ddl sur l'ensemble des partitions. Pour pouvoir établir la numérotation des ddls sur l'ensemble des processeurs, nous devons savoir si un ddl interprocessus est fantôme ou pas. Si c'est le cas, il faut également récupérer l'indice de processeur et l'indice du ddl dans cette partition. L'algorithme que nous avons implémenté se résume par les étapes suivantes :

- Pour chaque face de la frontière interprocessus Γ_{IP}^p, on envoie les ddl présents sur cette face vers le processeur p' partageant la face. On renvoie au processeur p le numéro local du ddl dans la partition de p' mais également des informations représentant la nécessité de rechercher le ddl dans une autre partition. C'est le cas pour les ddl connectés à plus de deux processeurs. On itère ce procédé jusqu'à ce que tous les degrés de liberté interprocessus aient communiqué avec les partitions dans lesquelles les ddl apparaissent.
- Pour chaque ddl de Γ_{IP}^p, nous connaissons les processeurs auquel il appartient. En appliquant la règle du processeur de rang le plus petit, nous pouvons savoir si le ddl appartient ou non au processeur courant.
- Grâce à toutes les informations collectées dans les étapes précédentes, il est maintenant possible de construire la carte des degrés de liberté globaux et la carte reliant la numérotation locale avec la numérotation globale.

Beaucoup de communications sont nécessaires pour cette construction. L'implémentation actuelle fonctionne très bien, mais elle n'est pas encore parfaitement scalable. Certaines parties doivent être optimisées. Cependant, dans beaucoup d'applications, cette table est construite une seule fois. Ainsi la scalabilité globale de l'application n'est pas détériorée.

1.3 Espaces de fonction

Dans cette partie, nous allons présenter la construction des espaces de fonction parallèles dans la librairie FEEL++. Celle-ci se base sur la table des degrés de liberté parallèle que nous avons établie précédemment. Le but de cette partie est de montrer les différentes façons de construire un espace de fonction dans un environnement parallèle. Nous présenterons d'abord le cas non composites, c'est-à-dire que l'espace est composé d'un seul champ. Nous passerons ensuite au cas composite qui est utilisé par exemple dans un contexte multiphysique. Pour chaque cas, nous aborderons également l'impact sur la structure des matrices creuses parallèles pour un code éléments finis standard.

Espaces de fonction standards

Nous commençons par les espaces de fonction non composites scalaires ou vectoriels. C'est le cas le plus simple qui correspond principalement à la création de la table des degrés de liberté parallèle. Pour construire un espace de fonction dans FEEL++, nous avons besoin du type d'espace et d'un objet décrivant le maillage. Si le maillage est distribué sur plusieurs processeurs, alors celui-ci va engendrer automatiquement la construction de l'espace de fonction parallèle associé à la table des degrés de liberté parallèle. Le code suivant montre la construction d'un espace de fonction de Lagrange P2 continu sur trois processeurs.

```
auto mesh = R.createMesh(_mesh=new mesh_type,_name="mymesh",
                         _partitions=3);
typedef Lagrange<2,Scalar> basis_type;
typedef FunctionSpace<mesh_type,bases<basis_type> > space_type;
auto Xh = space_type::New(_mesh=mesh);
```

Si l'environnement possédait N processeurs mais que le maillage était défini seulement sur M processeurs avec $M < N$, alors automatiquement l'espace de fonction associé serait défini sur ces M processeurs.

PROC 0	A_{00}	A_{01}	A_{02}
PROC 1	A_{10}	A_{11}	A_{12}
PROC 2	A_{20}	A_{21}	A_{22}

FIGURE 7.7 – Structure de la matrice éléments finis standards pour un espace de fonction non composite.

Nous décrivons maintenant la structure de la matrice éléments finis pour ce type d'espace avec la représentation faite dans la figure 7.7. Soit v une entrée de la matrice en (i,j). La valeur v sera alors stockée sur le processeur qui contiendra l'indice global i défini par la table des degrés de liberté. Nous avons mis en évidence les blocs diagonaux, surlignés en rouge. La majorité des entrées non nulles de la matrice se trouve dans ces blocs. Les blocs A_{pq} avec $p \neq q$ contiennent les entrées non nulles dont l'indice global de colonne n'appartient pas aux processeurs p, et sont très minoritaires par rapport aux blocs diagonaux.

Espaces de fonction produits (I)

Nous passons maintenant au cas des espaces composites, c'est-à-dire que l'espace de fonction est composé de plusieurs sous-espaces scalaires ou vectoriels. Nous commençons par construire la table des degrés de liberté parallèle associée à chaque sous-espace. Le partitionnement utilisé est le même pour chacun des sous-espaces et donc chaque processeur va avoir accès à une partie de la table des degrés de liberté de chaque sous-espace. Ensuite, nous devons construire la table des degrés de liberté de l'espace composite. Pour cela, chaque processeur va construire sa numérotation locale des degrés de liberté. Nous commençons par réutiliser la numérotation locale du premier sous-espace, puis celle du second sous-espace en translatant chaque indice local par le nombre de degrés de liberté du premier. On itère ce processus pour tous les espaces et l'on obtient une numérotation locale contiguë. Ces opérations sont locales et donc ne nécessitent aucune communication. Après, nous devons construire la numérotation globale à l'ensemble des processeurs. Nous utilisons le même procédé que pour la numérotation locale sauf que nous avons besoin d'un indice global de départ pour chaque processeur pour avoir une numérotation globale contiguë sur l'ensemble des processeurs. Nous supposons que le processeur de rang 0 commence à 0, puis l'indice de départ du processeur 1 est égal au nombre de degré de liberté globale du processeur 0. Pour le processeur 3, l'indice de départ sera égal à la somme du nombre de degré de liberté globale du processeur 0 et 1. On itère ce procédé sur l'ensemble des processeurs. Avant cette procédure, il est nécessaire de mettre en oeuvre des communications pour obtenir l'indice global de départ de chaque processeur.

Au niveau de Feel++, la génération de l'espace composite est également faite automatiquement à l'aide du partitionnement du maillage. Nous avons le code suivant qui permet de créer un espace de fonction P2P1 parallèle :

```
typedef Lagrange<2,Vectorial> basis_u_type;
typedef Lagrange<1,Scalar> basis_p_type;
typedef FunctionSpace<mesh_type,bases<basis_u_type,
                                      basis_p_type> > space_type;
auto Xh = space_type::New(_mesh=mesh);
```

Cet exemple peut représenter la création d'un espace de fonction pour un modèle de mécanique des fluides. On retouve ici l'élément fini standard de Taylor-Hood pour la mécanique des fluides.

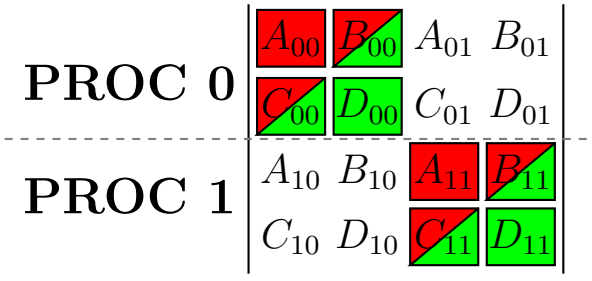


Figure 7.8 – Structure de la matrice éléments finis standards pour un espace de fonction composite (I).

La figure 7.8 représente la structure de la matrice éléments finis pour un espace de fonction composé de deux sous-espaces et distribué sur deux processeurs. Nous avons mis

en couleur les blocs comportant la majorité des entrées non nulles. En rouge, nous avons les entrées associées au premier sous-espace et en vert les entrées du second sous-espace. Les blocs colorés de rouge et de vert correspondent aux blocs de couplage entre les sous-espaces. Dans certaines situations, ces blocs peuvent ne pas contenir d'entrées non nulles. Pour le calcul de préconditionneurs de type Schwarz additif par exemple, la structure des blocs diagonaux est importante. Dans notre cas, le bloc diagonal du processeur p est représenté par l'ensemble des blocs A_{pp}, B_{pp}, C_{pp} et D_{pp}.

Espaces de fonction produits (II)

Nous proposons dans cette partie une autre construction pour les espaces de fonction composites. Cependant, cette alternative est souvent moins efficace que la précédente et elle est également plus restrictive dans son utilisation. En fait, cette construction a été la première implémentation parallèle des espaces composites, car sa mise en oeuvre était plus pratique dans le développement informatique. Toutefois, des extensions sont possibles pour améliorer son efficacité et c'est pourquoi nous trouvons utile de décrire cette procédure dans cette section.

Le principe de cette construction est d'associer un groupe de processeurs à un sous-espace. Un processeur ne possèdera qu'une seule partie d'un seul sous-espace. Par exemple, nous avons un environnement parallèle avec quatre processeurs et nous souhaitons construire un espace composé de deux sous-espaces. Nous voulons alors distribuer chaque sous-espace sur deux processeurs. Pour cela, le maillage doit être partitionné en deux parties, le processeur de rang 0 (respectivement 1) aura la même partition que le processeur de rang 2 (respectivement 3). Cette fois-ci, la construction de la table des degrés de liberté pour l'espace composite est plus simple à mettre en oeuvre. Nous créons tout d'abord la table des degrés de liberté pour chaque sous-espace dans son groupe de processeurs. Ensuite, pour l'espace composite, la numérotation locale est identique à celle créée par la table des sous-espaces. Pour la numérotation globale, le premier groupe de processeurs associé au premier sous-espace a encore la même numérotation. Pour le second, nous devons juste translater la numérotation globale du sous-espace par le nombre de degrés de liberté globaux du premier sous-espace. Et l'on itère sur l'ensemble des sous-espaces.

Nous présentons maintenant la construction d'un tel espace de fonction avec Feel++. L'environnement parallèle est composé de quatre processeurs et l'espace composite possède deux sous-espaces. Nous commençons par définir un objet `mapWorldComm` décrivant la répartition des processeurs pour les sous-espaces (détaillé au chapitre 8).

```
std::vector<int> mapColor(4);
for (int proc = 0 ; proc < 4 ; ++proc)
    if (proc < 2 ) mapColor[proc] = 0;//sous-espace 1
    else mapColor[proc] = 1;//sous-espace 2
auto mapWorldComm = WorldComm(mapColor);
```

Ensuite, nous construisons un maillage avec deux partitions. Les processeurs de rang 0 et 2 (respectivement 1 et 3) vont charger la partition 1 (respectivement 2). Cette distribution des partitions est gérée par l'objet `mapWorldComm`.

```
mesh = R.createMesh(_mesh=new mesh_type,_name="mymesh",
                    _partitions=mapWorldComm.localSize(),
                    _worldcomm=mapWorldComm);
```

Enfin, l'espace de fonction produit est créé à partir du maillage et de la description de la répartition des processeurs sur les sous-espaces.

```
std::vector<WorldComm> mysubWorldComm(2);
mysubWorldComm[0]=mapWorldComm.subWorldComm(0);//sous-espace 1
mysubWorldComm[1]=mapWorldComm.subWorldComm(1);//sous-espace 2
auto Xh = space_type::New(_mesh=mesh,
                          _worldscomm=mysubWorldComm);
```

Comme nous l'évoquions au début de cette partie, cette méthode admet plus de contraintes que la précédente. Si nous avons N sous-espaces, alors le nombre de processeurs nécessaire doit être un multiple de N, disons kN. L'entier k correspond au nombre de partitions dans le maillage. Une autre source de mauvaise efficacité est la distribution de la charge de calcul sur l'ensemble des processeurs. Si les sous-espaces de fonction sont très différents, par exemple un espace vectoriel P3 et un espace scalaire P1, alors les processeurs associés à l'espace vectoriel utiliseront beaucoup plus de mémoire et également ils auront plus de travail. Cependant, cet inconvénient pourrait être corrigé si l'on avait la possibilité de gérer plusieurs partitionnements. On aurait alors un partitionnement par sous-espace.

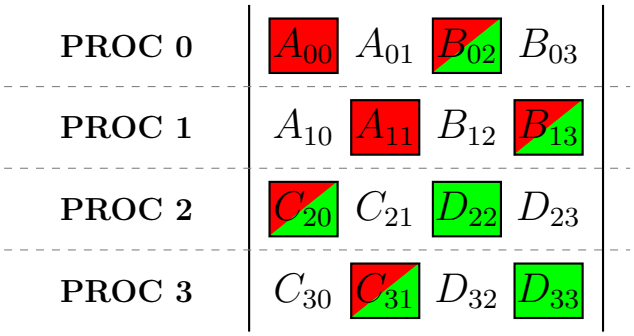


FIGURE 7.9 – Structure de la matrice éléments finis standards pour un espace de fonction produit (II).

Avec la définition de notre espace produit, la structure d'une matrice éléments finis est illustrée avec la figure 7.9. Ce cas représente un environnement de quatre processeurs avec deux sous-espaces. Les principales entrées non nulles sont représentées par les blocs en couleurs. Les blocs rouges (respectivement verts) correspondent aux entrées appartenant au premier (respectivement second) sous-espace. Les blocs rouges et verts sont les entrées non nulles de couplage entre les deux sous-espaces. L'avantage de cette construction est la structure des blocs diagonaux, car ils sont composés d'un seul sous-espace. Ce sont les blocs A_{00},A_{11},D_{22},D_{33} dans notre exemple. Ainsi, ces blocs sont plus facilement inversibles grâce à la possibilité d'utiliser une large gamme de préconditionneurs. Au niveau du préconditionneur global, celui-ci risque d'être moins efficace si l'on utilise un préconditionneur de type Schwarz additif. Cela est dû au nombre relativement important d'entrées non nulles ne se trouvant pas dans la diagonale. Cependant, d'autres types de préconditionneurs pourraient être testés pour améliorer l'efficacité comme ceux basés sur le complément de Schur.

1.4 Structures algébriques et solveurs parallèles : PETSc

Nous allons maintenant parler de l'implémentation des structures algébriques parallèles ainsi que des solveurs disponibles dans Feel++. Ces outils algébriques se basent sur la librairie PETSc [13, 12, 14, 146]. Nous avons développé une interface avec cette librairie, ce qui permet l'accès à de nombreux outils de PETSc. En effet, PETSc permet de construire des vecteurs et matrices parallèles. Il propose également une large gamme des solveurs linéaires et non linéaires et de préconditionneurs parallèles.

Les vecteurs

Nous commençons par la description de l'interface pour les vecteurs algébriques parallèles. Pour construire un vecteur, nous avons uniquement besoin d'un espace de fonction décrivant la distribution des degrés de liberté locale et globale. Dans le code Feel++, une nouvelle classe a été créée pour les vecteurs parallèles, **VectorPetscMPI**. Cette structure de donnée comporte deux vues, une globale parallèle et une locale séquentielle. La vue globale est utilisée pour l'assemblage des formes linéaires dans notre contexte éléments finis. Cette structure de donnée sera aussi utilisée pour la résolution des systèmes algébriques. Pour assembler les formes linéaires, nous calculons la contribution d'une entrée d'indice locale i. Ensuite, grâce à la carte reliant la table des degrés de liberté locale à la numérotation globale, cette contribution sera assemblée dans la structure parallèle. Si l'entrée est une entrée fantôme, alors elle sera stockée puis communiquée au processeur lors de l'appel de la méthode `close()`. Quant à la vision locale, elle permet d'avoir accès aux entrées présentes dans le processeur, mais aussi aux entrées fantômes sans avoir besoin de faire appel à une communication. Cela permet ensuite de réaliser des opérations algébriques purement locales, c'est-à-dire lorsque l'on veut faire des opérations basiques comme l'addition, calculer des normes sur chaque processeur indépendamment ou encore exporter une solution. Pour transférer les valeurs du vecteur global parallèle vers le vecteur local séquentiel, il faut appeler la méthode `localize()`. Cette fonction utilise des routines de PETSc pour synchroniser le conteneur local.

Les matrices

Nous passons à l'interface des matrices creuses parallèles. Pour construire une matrice éléments finis, nous avons besoin de deux espaces de fonctions associées aux fonctions *tests* et *solutions*. L'implémentation actuelle fonctionne uniquement pour deux espaces basés sur le même maillage. Dans le code Feel++, c'est la classe **MatrixPetscMPI** qui gère les matrices parallèles et elle se base sur la structure matricielle de PETSc. Cette fois-ci, nous avons une seule vision globale de la matrice. Pour définir une matrice creuse avec PETSc, il nous faut construire le graphe représentant les entrées non nulles. Ce graphe est une structure parallèle assez complexe à mettre en oeuvre et nécessite des communications entre les processeurs partageant des degrés de liberté. Nous avons illustré avec la figure 7.10 les entrées non nulles de la matrice de Stokes pour une approximation P2P1. Pour l'assemblage des formes bilinéaires, nous utilisons le même principe que pour les vecteurs parallèles. À partir des indices locaux (i,j), nous assemblons les contributions dans la matrice à l'aide de la carte reliant les indices locaux et indices globaux. De même, les contributions se trouvant sur des entrées fantômes sont assemblées à l'appel de la méthode `close()`.

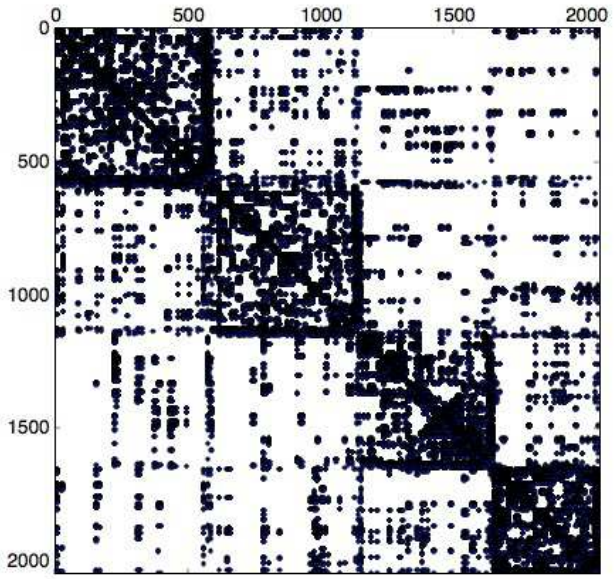


FIGURE 7.10 – Entrées non nulles de la matrice associé au problème de Stokes avec 4 processeurs (vitesse sur 4 proc, pression sur 4 proc).

Les solveurs et préconditionneurs

Nous nous intéressons maintenant à la résolution des systèmes linéaires en parallèle de la forme :

$$AX = F \tag{7.1}$$

où A dénote la matrice représentant l'opérateur linéaire, F est le vecteur second membre et X le vecteur solution inconnu. Pour résoudre ce système, nous appliquons des méthodes itératives utilisant des projections dans des sous-espaces particuliers, les espaces de Krylov. Les espaces de Krylov permettent de construire, par de simples opérations de type produit matrice vecteur, produit scalaire ou combinaison linéaire de vecteurs, des sous-espaces affines très pertinents pour chercher des approximations de la solution du système linéaire. Une grande variété de méthode de Krylov est implémentée dans PETSc. Nous pouvons citer les méthodes plus connues : CG (Conjugate gradient), GMRES (generalized minimum residual), BiCGSTAB (biconjugate gradient stabilized), QMR (quasi minimal residual), TFQMR (transpose-free QMR) et MINRES (minimal residual). Le détail de ces méthodes de Krylov peut être trouvé dans [139].

Cependant, la vitesse de convergence des méthodes de Krylov est liée au conditionnement de la matrice A. Plus il est petit et plus l'algorithme convergera vite. Or, le conditionnement de la matrice A n'est pas a priori petit et généralement augmentent avec h et p. C'est pourquoi nous préconditionnons le système 7.1 dans le but d'avoir une matrice avec un conditionnement plus petit. Nous obtenons par exemple le système suivant pour un préconditionnement à gauche :

$$PAX = PF \tag{7.2}$$

avec P une approximation de A^{-1}. La combinaison du préconditionnement et des méthodes de Krylov permet ainsi d'avoir des procédures de résolution très efficace.

La librairie PETSc propose plusieurs préconditionneurs parallèles. Nous faisons une énumération des préconditionneurs accessibles depuis notre librairie FEEL++ :

– **LU avec MUMPS** : La librairie MUMPS [5, 6] est également accessible depuis PETSc et elle propose un préconditionneur LU parallèle. Avec ce type de préconditionneur, la méthode de Krylov n'est plus réellement nécessaire, car ce solveur peut-être vu comme un solveur direct. Après expérimentation, ce préconditionneur est plutôt efficace en 2D ou lorsque le système n'est pas trop grand. Le coût de calcul de ce préconditionneur est en effet assez conséquent. Cependant, dans le cas d'un problème transitoire, nous pouvons le calculer une seule fois et le réutiliser le plus possible. Il s'avère que cette technique est très efficace. Pour les très grands systèmes, ce préconditionneur n'est plus du tout efficace et il est également très gourmand en mémoire.
– **Schwarz additif** : PETSc propose plusieurs préconditionneurs de type Schwarz additif. À partir d'un domaine divisé en p sous-domaines, le préconditionneur de Schwarz additif classique, noté P, prend la forme suivante :

$$P = \sum_{i=1}^{p} R_i^T A_i^{-1} R_i \tag{7.3}$$

avec R_i l'opérateur de restriction et $A_i = R_i A R_i^T$ la restriction de A au $i^{\text{ème}}$ sous-domaine. Les sous-matrices A_i correspondent aux blocs diagonaux présentés dans la partie 1.3 de ce chapitre. Les matrices A_i^{-1} ne sont pas calculées explicitement. Nous utilisons également une méthode de Krylov avec préconditionnement. Nous parlons alors de sous-solveurs et de sous-préconditionneurs. Par contre, les sous-solveurs et sous-préconditionneurs sont séquentiels. Ils sont associés à chaque sous-domaine. Nous pouvons ainsi utiliser toute la gamme de préconditionneurs séquentiels disponible dans PETSc. Nous pouvons citer les plus couramment utilisés : LU, ILU (Incomplète LU), Cholesky, ICC (Incomplète Cholesky), Jacobi, ML (algébrique multigrille). La librairie PETSc propose plusieurs types de préconditionneurs de Schwarz additif. Nous avons le préconditionneur Block Jacobi (noté bjacobi) qui correspond à l'algorithme de Schwarz additif sans recouvrement. Nous disposons également des préconditionneurs de Schwarz additif avec recouvrement algébrique. On notera par la suite ce préconditionneur asm(k), avec k la taille du recouvrement. Nous avons également accès à un dernier préconditionneur avec recouvrement que l'on note gasm(k). Ce dernier est une variante du préconditionneur asm(k) conçu pour le cas où l'on a beaucoup de sous-domaines, voir [147] pour les détails.
– **FieldSplit** : Nous disposons également de préconditionneurs spécifiques aux matrices blocs (par exemple, pour les discrétisations monolithiques multiphysiques). Nous avons trois types de préconditionneurs utilisables pour un nombre arbitraire de blocs : block Jacobi, block Gauss-Seidel et symmetric block Gauss-Seidel. Les blocs peuvent être définis en indiquant les lignes qui appartiennent à un bloc. Par exemple, dans le cas de matrice de Stokes parallèle, nous avons montré que chaque processeur contenait à la fois une partie des entrées de la vitesse et de la pression. Nous pouvons alors regrouper les lignes correspondant à la vitesse (respectivement la pression) et ainsi créer une matrice bloc 2×2. De plus, pour ce type de structure, nous pouvons utiliser une autre famille de solveurs basés sur le complément de Schur. Ces types de préconditionneurs fonctionnent en parallèle, mais leurs performances semblent pour le moment inférieures au préconditionneur Schwarz additif. Pour cette raison, les précondtionneurs de type FieldSplit n'apparaitront pas dans

nos tests de scalabilité. Cependant, nous avons peut-être mal configuré cet outil qui requiert un grand nombre d'options. De plus, ce préconditionneur est censé être plus robuste pour des problèmes difficiles à résoudre. Le détail de ces méthodes peut être trouvé dans [12, 52, 116, 51].

2 Vérifications et scalabilités

Nous allons maintenant expérimenter l'implémentation de librairie FEEL++ en parallèle. L'objectif de cette partie est de vérifier les résultats numériques et d'étudier la scalabilité de nos modèles. Nous effectuerons des tests de scalabilité forte, c'est-à-dire que l'on fixe la taille du problème et l'on augmente le nombre de processeurs. Dans ce cas, nous regarderons particulièrement le *speed-up* $= t1/tN$ avec t1 le temps de calcul avec un processeur et tN avec N processeur. Nous lancerons également des tests de scalabilité faible, où la taille du problème varie en fonction du nombre de processeurs. Nous calculerons alors un pourcentage *d'efficacité* ou *efficiency* $= (t1/tN)100$. Les tests de scalabilités ont été fait sur des processeurs Amd 6272 (Interlagos) 2.1 Ghz.

2.1 Test de converge numérique

Un bon moyen de vérifier numériquement notre implémentation parallèle est de tester les propriétés de convergence théorique d'un problème éléments finis. Nous choisissons le modèle de Stokes avec comme solution analytique de référence la solution de Ethier-Steinman [54]. Nous notons par $\boldsymbol{u}^{ES}$ et p^{ES} le couple solution vitesse et pression. Leurs expressions sans la dépendance en temps sont données par les équations suivantes :

$$
\begin{aligned}
u_1^{ES}(\boldsymbol{x}) &= -a(e^{ax}\sin(ay+dz) + e^{az}\cos(ax+dy)) \\
u_2^{ES}(\boldsymbol{x}) &= -a(e^{ay}\sin(az+dx) + e^{ax}\cos(ay+dz)) \\
u_3^{ES}(\boldsymbol{x}) &= -a(e^{az}\sin(ax+dy) + e^{ay}\cos(az+dx)) \\
p^{ES}(\boldsymbol{x}) &= -\frac{a^2}{2}[e^{ax} + e^{ay} + e^{ax} \\
&\quad + 2\sin(ax+dy)\cos(az+dx)e^{y+z} \\
&\quad + 2\sin(ay+dz)\cos(ax+dy)e^{z+x} \\
&\quad + 2\sin(az+dx)\cos(ay+dz)e^{x+y}]
\end{aligned}
$$

avec $a = \frac{\pi}{4}$ et $d = \frac{\pi}{2}$.

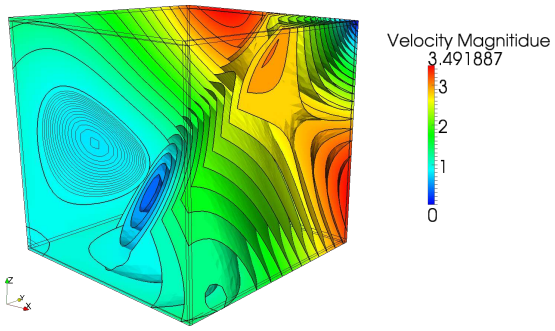
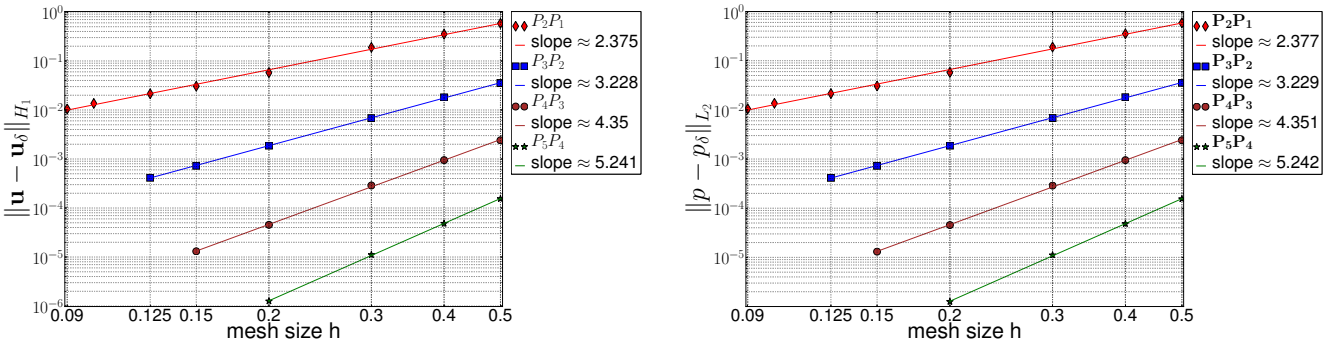


FIGURE 7.11 – Magnitude de la vitesse de la solution de Ethier-Steinman

Le champ $\boldsymbol{u}^{ES}$ est aussi représenté avec la figure 7.11. Le domaine de calcul utilisé est le cube $[-1,1]^3$, ce qui permet alors d'assurer la divergence égale à 0 de la solution vitesse $\boldsymbol{u}^{ES}$. Nous considérons également avoir des conditions aux limites de Dirichlet et de Neumann, où leurs expressions est déduite de $\boldsymbol{u}^{ES}$ et p^{ES} et de même pour le second membre. Pour nos simulations, nous utilisons des éléments de Taylor-Hood sur des maillages tétraédriques. Chaque test a été effectué sur 16 processeurs en utilisant le préconditionneur gasm1 avec un préconditionneur LU dans chaque bloc. Comme le montre la figure 7.12, nous obtenons les pentes de convergence optimales au sens de la méthode éléments finis. Cette vérification est donc réussie.



(a) Erreur H_1 sur la vitesse

(b) Erreur L_2 sur la pression

FIGURE 7.12 – Courbes de convergence pour la solution d'Ethier-Steinman pour différents ordres polynomiaux. Les graphes sont en échelle log-log.

2.2 Scalabilités fortes et faibles en mécanique des solides

Nous allons maintenant étudier la scalabilité de nos outils pour des applications en mécanique des solides. Le modèle étudié est un modèle hyperélastique compressible avec une loi de Saint-Venant-Kirchhoff, présenté au chapitre 5, dont on rappelle l'équation :

$$\rho_s^* \frac{\partial^2 \boldsymbol{\eta_s}}{\partial t^2} - \nabla \cdot (\boldsymbol{F_s \Sigma_s}) = \boldsymbol{f}_s \quad , \quad \boldsymbol{\Sigma}_s = \lambda_s \left(tr \boldsymbol{E}_s\right) \boldsymbol{I} + 2\mu_s \boldsymbol{E}_s$$

L'application qui va être testée est également similaire à celle décrite dans le chapitre 5, section 4.1. Cette application décrit le mouvement d'une plaque soumise à son propre poids. Le second membre $\boldsymbol{f}_s = (0, 0, -2)$ représente la force de gravité que nous imposons. Nous avons représenté la géométrie de cette plaque avec la figure 7.13. Elle est caractérisée par 3 longueurs L_x,L_y et L_z. Nous fixons $L_x = 0.24899$ et $L_z = 0.02$, la dernière longueur dépendra du type de scalabilité. Au niveau des conditions aux limites, nous avons un déplacement nul sur la partie grisée et le reste est libre de contraintes.

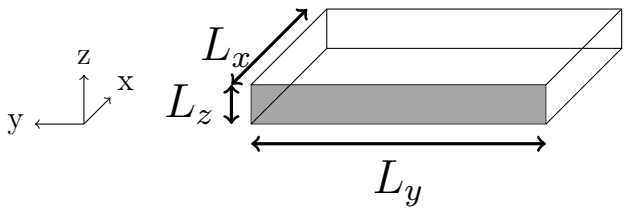
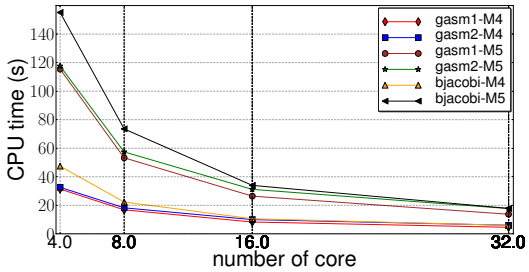


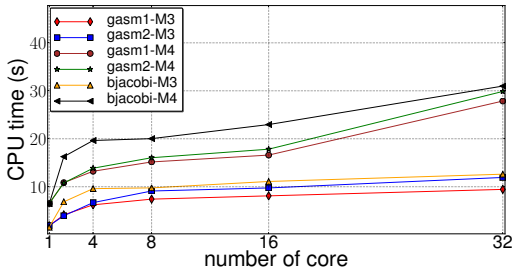
FIGURE 7.13 – Géométrie du problème de plaque soumise à son propre poids

Nous présentons maintenant les configurations utilisées pour ces tests de scalabilité. Pour tous les tests, nous avons choisi une approximation $P2$ pour le déplacement. La longueur L_y ainsi que le nombre de degrés de liberté se trouvent dans la table 7.1 pour la scalabilité forte et dans la table 7.2 pour la scalabilité faible. Plusieurs préconditionneurs ont aussi été utilisés. Nous avons comparé block jacobi(dénoté bjacobi), gasm avec un recouvrement de 1 (gasm1) et gasm avec un recouvrement de 2 (gasm2).

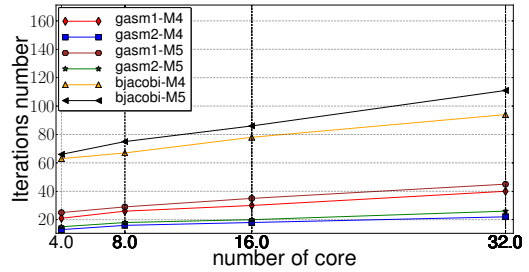
Pour ces tests, nous avons mesuré le temps de calcul pour l'assemblage de la matrice jacobienne et du vecteur résidu ainsi que le temps de résolution du système algébrique. À



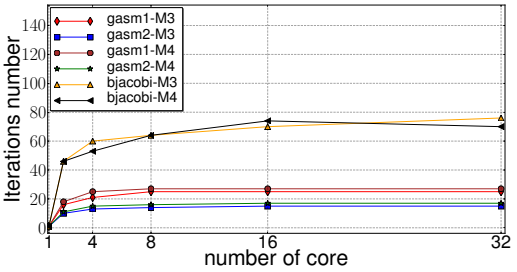
(a) Temps CPU pour la résolution du système algébrique pour la scalabilité forte



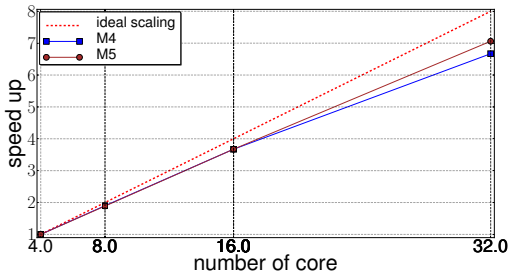
(b) Temps CPU pour la résolution du système algébrique pour la scalabilité faible



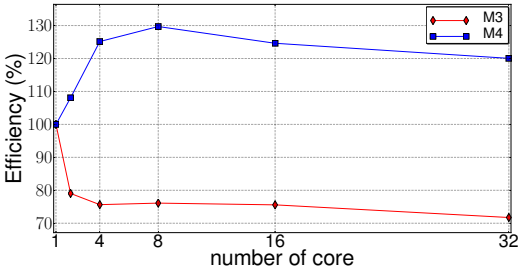
(c) Nombre d'itération du solveur pour la scalabilité forte



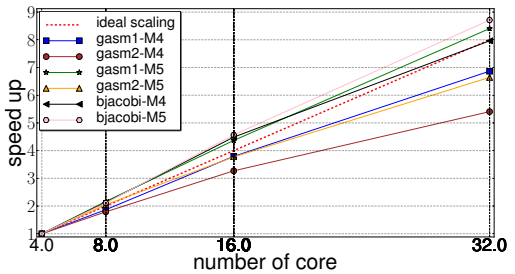
(d) Nombre d'itération du solveur pour la scalabilité faible



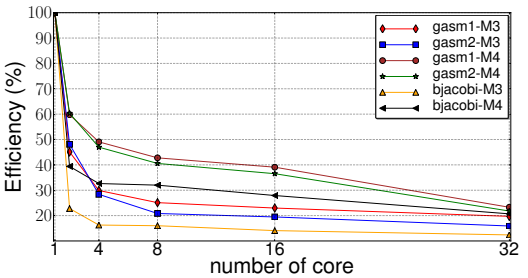
(e) Speed-up du temps d'assemblage pour la scalabilité forte



(f) Efficacité du temps d'assemblage pour la scalabilité faible



(g) Speed-up du temps de résolution pour la scalabilité forte



(h) Efficacité du temps de résolution pour la scalabilité faible

FIGURE 7.14 – Tests de scalabilité du modèle hyper-élastique avec une approximation P2

config	L_y	N_{elt}	N_{dof}
M4	0.41	28593	149562
M5	0.41	63216	315450

TABLE 7.1 – Configurations utilisées pour la scalabilité forte du modèle hyper-élastique avec une approximation P2.

nProc	L_y	config	N_{dof}	config	N_{dof}
32	1.6	M3	329835	M4	565179
16	0.8	M3	163782	M4	281475
8	0.4	M3	84003	M4	140190
4	0.2	M3	43887	M4	73650
2	0.1	M3	23343	M4	43803
1	0.05	M3	12966	M4	25575

TABLE 7.2 – Configurations utilisées pour la scalabilité faible du modèle hyper-élastique avec une approximation P2.

noter que pour cette application le temps d'assemblage est nettement supérieur au temps de résolution. Les résultats se trouvent dans la figure 2.2. Pour les scalabilités fortes et faibles, nous voyons que le préconditionneur bjacobi est le plus *lent*, figures 7.14(a) et 7.14(b). Les préconditionneurs gasm1 et gasm2 sont assez similaires. Cela s'explique par le fait que le nombre d'itérations nécessaires pour résoudre le système est clairement supérieur aux préconditionneurs gasm1 et gasm2, figures 7.14(c) et 7.14(d). Au niveau de l'assemblage des vecteurs et matrices, la scalabilité est assez bonne avec un comportement meilleur pour les maillages les plus fins. Le speed-up associé à la scalabilité forte est légèrement inférieur à la scalabilité idéale, figure 7.14(e). L'efficacité de la scalabilité faible est excellente comme le montre la figure 7.14(f). Enfin, nous voyons que la scalabilité pour la résolution du système se détériore un peu quand le nombre de processeurs augmente, figures 7.14(g) et 7.14(h). On peut remarquer que le préconditionneur bjacobi a une scalabilité forte parfaite, mais il est moins efficace que les préconditionneurs gasm1 et gasm2. On voit également que le préconditionneur gasm1 a une meilleure scalabilité que gasm2.

2.3 Scalabilités fortes et faibles en mécanique des fluides

Nous voulons établir des caractéristiques de scalabilité appliquées sur des problèmes en mécanique des fluides. Nous traitons ici le cas des écoulements de Stokes avec une géométrie simple, un cylindre, et le modèle de Navier-Stokes avec une géométrie complexe et réaliste, l'aorte.

Le modèle de Stokes

Pour le problème de Stokes, nous nous intéressons au temps d'assemblage et de résolution du système linéaire. La géométrie de ce problème est un cylindre de longueur L_c et de

rayon 0.5. Sur l'une des faces du cylindre, nous imposons un profil de vitesse parabolique et une sortie libre sur l'autre face. La vitesse est également nulle sur la paroi.

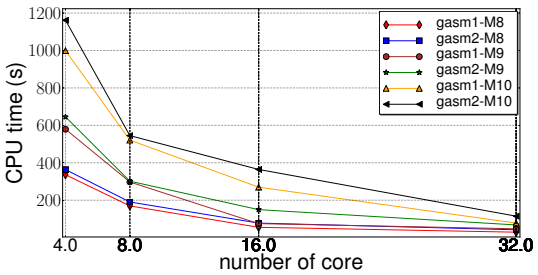
config	L_c	N_{elt}	N_{dof}
M8	6	111475	530275
M9	6	151443	712545
M10	6	210526	980105

TABLE 7.3 – Configurations utilisées pour la scalabilité forte du modèle de Stokes avec une approximation P2P1.

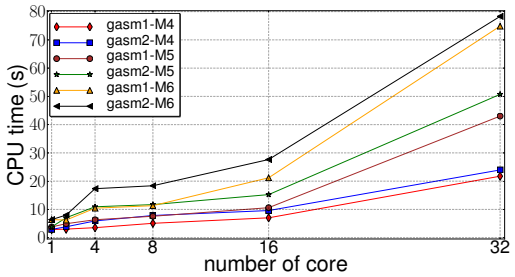
nProc	L_c	config	N_{dof}	config	N_{dof}	config	N_{dof}
32	16	M4	297372	M5	398181	M6	563574
16	8	M4	146094	M5	197262	M6	290423
8	4	M4	77255	M5	105296	M6	148027
4	2	M4	39834	M5	56075	M6	75005
2	1	M4	23950	M5	31971	M6	41869
1	0.5	M4	15705	M5	19207	M6	26523

TABLE 7.4 – Configurations utilisées pour la scalabilité faible du modèle de Stokes avec une approximation P2P1.

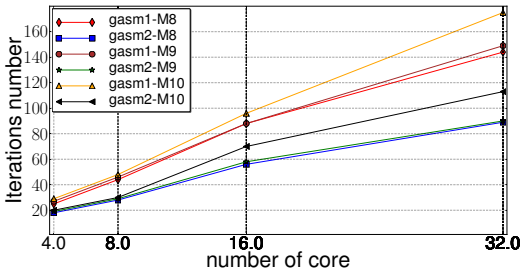
Les tests de scalabilité du modèle de Stokes présentés dans cette partie ont été faits avec une approximation P2P1 (vitesse/pression) et une géométrie d'ordre 1. D'autres tests ont été réalisés avec une approximation P3P2 et une géométrie d'ordre 2 et les résultats se trouvent dans l'annexe A. Les configurations utilisées pour l'approximation P2P1 se trouvent dans la table 7.3 pour la scalabilité forte et dans la table 7.4 pour la scalabilité faible. Les mesures de scalabilité pour P2P1 sont illustrées dans la figure 2.3. Nous avons comparé dans ces expériences plusieurs maillages et deux préconditionneurs parallèles, gasm1 et gasm2. Le préconditionneur bjacobi ne se trouve pas dans nos résultats, car il n'est pas assez robuste pour ce type d'application, la convergence est très lente et même diverge quand le nombre de processeurs devient grand. Comme le montrent les figures 7.15(a) et 7.15(b), le préconditionneur gasm1 est plus performant que gasm2 pour les deux types de scalabilités. Au niveau du nombre d'itérations du solveur parallèle, la figure 7.15(c) nous montre que le préconditionneur gasm2 permet d'en réduire ce nombre. Mais pour les deux préconditionneurs, le nombre d'itérations augmente fortement avec l'ajout de processeur. Cette augmentation apparait clairement avec la scalabilité faible, figure 7.15(c), où le nombre d'itérations « explose » pour le cas 32 processeurs. Au niveau du temps de résolution du système algébrique, le speed-up associé à la scalabilité forte est très bon, on est même au-dessus du speed-up idéal, figure 7.15(g). Pour la scalabilité faible, les performances du solveur algébrique décroissent avec l'augmentation du nombre de processeurs, figure 7.15(h). La forte augmentation du nombre d'itérations en est la cause. Pour l'assemblage, les performances sont idéales avec les scalabilités fortes et faibles jusqu'à 16 processeurs puis se détériorent pour le cas 32 processeurs, figures 7.15(e) et 7.15(f). On remarque toutefois que le speed-up s'améliore avec les maillages les plus raffinés.



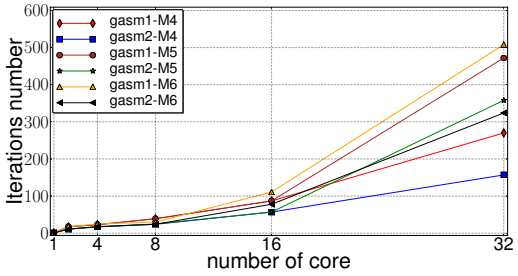
(a) Temps CPU pour la résolution du système algébrique pour la scalabilité forte



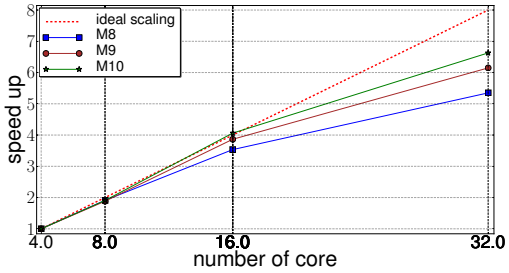
(b) Temps CPU pour la résolution du système algébrique pour la scalabilité faible



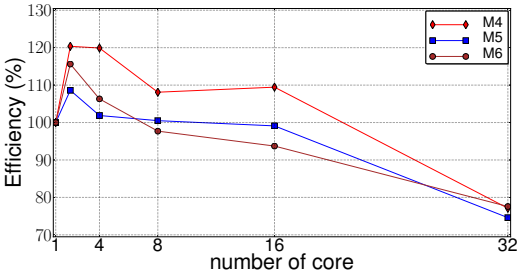
(c) Nombre d'itération du solveur pour la scalabilité forte



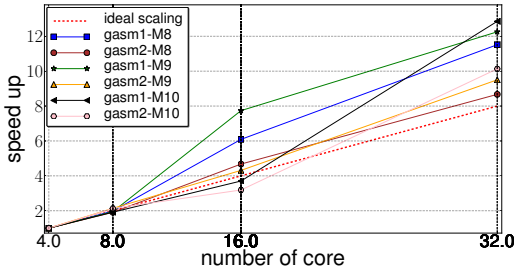
(d) Nombre d'itération du solveur pour la scalabilité faible



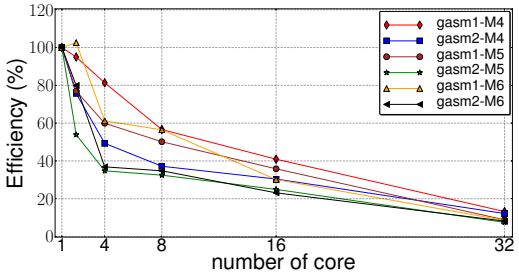
(e) Speed-up du temps d'assemblage pour la scalabilité forte



(f) Efficacité du temps d'assemblage pour la scalabilité faible



(g) Speed-up du temps de résolution du solveur algébrique pour la scalabilité forte



(h) Efficacité du temps de résolution du solveur algébrique pour la scalabilité faible

FIGURE 7.15 – Tests de scalabilité du modèle de Stokes avec une approximation P2P1 et une géométrie d'ordre 1.

Le modèle de Navier-Stokes incompressible

Nous allons maintenant présenté un test de scalabilité forte pour Navier-Stokes avec une géométrie réaliste, une aorte, représentée en 7.16. La formulation de ce modèle non linéaire est décrite avec les équations 4.1 et 4.1. Encore une fois, les temps de calcul mesurés seront faits sur le premier pas de temps.

config	N_{elt}	N_{dof}
M2	42744	199371
M3	109997	497525
M4	223120	989237

TABLE 7.5 – Configurations utilisées pour la scalabilité forte.

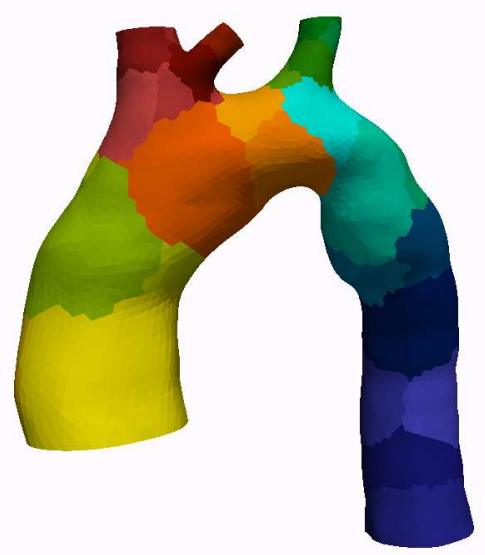
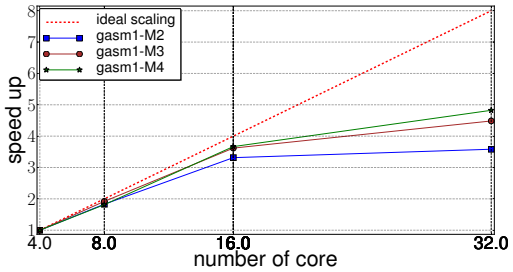
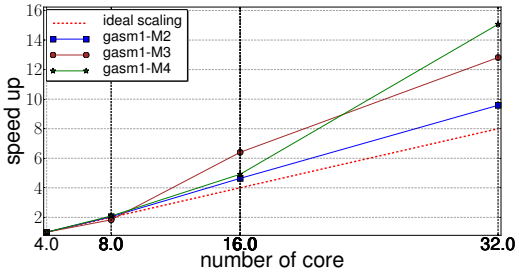


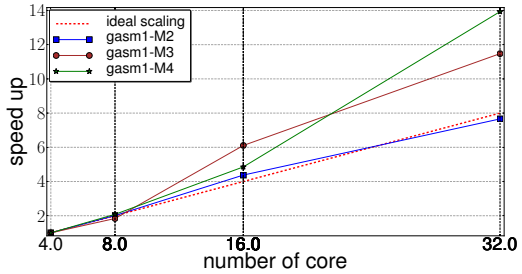
FIGURE 7.16 – Maillage M4 de l'aorte avec 32 partitions. Chaque couleur représente une partition.



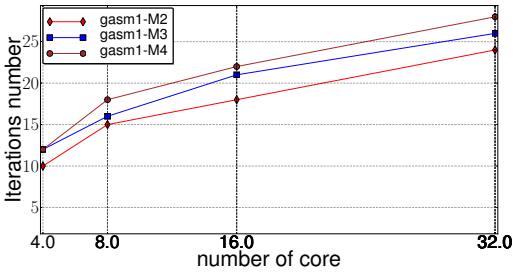
(a) Speed-up pour l'assemblage des matrices et vecteurs



(b) Speed-up pour les résolutions des systèmes algébriques



(c) Speed-up pour l'assemblage+résolutions



(d) Nombre d'itération du premier solveur linéaire

FIGURE 7.17 – Tests de scalabilité du modèle de Navier-Stokes avec une approximation $P2P1$ et une géométrie d'ordre 1.

Ce test de scalabilité forte a été fait avec une approximation $P2P1$ pour le couple

vitesse-pression. Le nombre d'éléments pour chaque maillage ainsi que le nombre de degrés de liberté associé se trouvent dans la table 7.5. Nous pouvons trouver les résultats de la scalabilité dans la figure 2.3. Nous avons tout d'abord le speed-up pour l'assemblage avec la figure 7.17(a). Il se comporte de manière idéale jusqu'à 16 processeurs, mais ensuite les performances stagnent. Les communications semblent devenir dominantes par rapport au coût d'assemblage local. On constate que les performances sont meilleures avec les maillages les plus fins, là où l'assemblage est le plus important. Par contre, la scalabilité du solveur est excellente, figure 7.17(b). Cela s'explique en partie par le fait que le nombre d'itérations du solveur linéaire reste relativement faible, figure 7.17(d). On peut également voir avec la figure 7.17(d) que la scalabilité de l'application est dominée par celle du solveur, ce qui nous permet d'avoir cette excellente scalabilité.

3 GPU : accélération de l'assemblage local

Depuis leur création, les cartes graphiques ont été utilisées uniquement pour afficher des données en 2D ou 3D. Mais avec l'augmentation de la demande pour l'affichage en temps réel de graphiques 3D complexes et de très haute définition, la structure physique des cartes graphiques a évolué très rapidement pour devenir massivement parallèle, multi-thread, multi-coeur avec une grande puissance de calcul et une haute bande passante mémoire. Comme nous pouvons le voir dans la figure 7.18, la puissance des cartes graphiques est maintenant supérieure à celle des CPU. Cette différence est principalement due à la structure de la carte GPU qui est spécialisée dans le calcul massivement parallèle. Les cartes GPU contiennent beaucoup plus d'unités de calcul (ALU) indépendantes que le CPU. En contrepartie les GPU fonctionnent bien uniquement avec des données contigües en mémoire, car chacune des unités de calcul possède peu de mémoires caches pour stocker les données à calculer. Cette structure est schématisée sur la figure 7.19. Ces nouvelles architectures pourront montrer leur supériorité face aux CPU sur certains algorithmes de calcul.

En 2006, Nvidia a introduit CUDA, une architecture informatique qui permet d'utiliser la puissance massivement parallèle des cartes graphiques. Cette architecture est livrée avec un ensemble d'instructions basées sur le langage de programmation C. Ceci permet aux développeurs d'utiliser directement les cartes graphiques pour faire des calculs parallèles. Le but de cette section est de montrer le potentiel d'accélération des GPU pour une étape de la construction du problème élément fini, l'assemblage local. L'implémentation numérique a été réalisée ici avec la bibliothèque CUDA développée par NVIDIA.

3.1 Assemblage local dans la méthode éléments finis

Dans notre librairie de calcul éléments finis, la construction de la matrice globale du problème s'effectue en deux étapes. Tout d'abord, nous avons l'assemblage des matrices élémentaires. Elles correspondent aux contributions apportées par les fonctions de base sur chaque élément. Ces matrices sont obtenues à partir de calculs très localisés et sans dépendance par rapport aux autres. Cette première étape peut alors se révéler très coûteuse lorsque le nombre de degrés de liberté élémentaire augmente (ordre d'approximation polynomial, dimension). Ensuite, nous obtenons la matrice globale en utilisant les valeurs obtenues dans les matrices élémentaires. Nous avons la même procédure pour le calcul

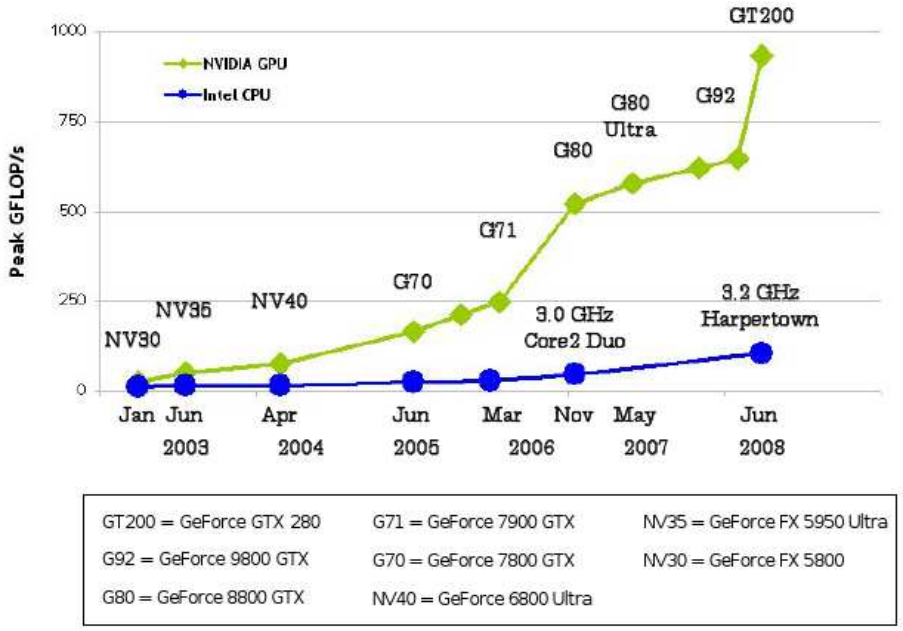


FIGURE 7.18 – Évolution de la puissance de calcul pour un CPU et GPU.

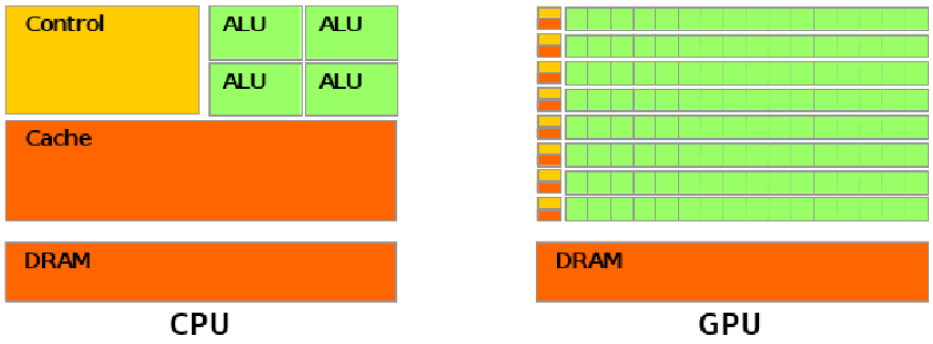


FIGURE 7.19 – Schéma simplifié de l'architecture physique CPU et GPU.

des seconds membres élémentaires. Mais pour cette étude, nous avons choisi de ne pas décrire cette partie très similaire. La nature très localisée des calculs élémentaires et de leur indépendance permet de penser que cette partie devrait être adaptée aux architectures GPU. Pour illustrer ces propos, nous allons considérer un problème mathématique « assez simple ». Il s'agit de trouver la fonction $u : \Omega \in \mathbb{R}^d \to \mathbb{R}$ vérifiant :

$$\begin{cases} u - \Delta u = f \text{ dans } \Omega \\ u = g \text{ sur } \Gamma_D \\ \dfrac{\partial u}{\partial n} = h \text{ sur } \Gamma_N \end{cases} \tag{7.4}$$

avec f, g et h des fonctions données et $\partial\Omega = \Gamma_D \cap \Gamma_N$. La formulation variationnelle de cette EDP s'obtient ensuite en multipliant (7.4) par une fonction test $v \in H^1_{0,\Gamma_D}(\Omega)$, puis en intégrant l'équation sur Ω et enfin en utilisant les formules de Green, ce qui nous donne :

Trouver $u \in H^1_{g,\Gamma_D}(\Omega)$ tel que $\forall v \in H^1_{0,\Gamma_D}(\Omega)$:

$$\int_\Omega u(\mathbf{x})\, v(\mathbf{x}) + \int_\Omega \nabla u(\mathbf{x}) \cdot \nabla v(\mathbf{x}) = \int_\Omega f(\mathbf{x})\, v(\mathbf{x}) + \int_{\Gamma_1} h(\mathbf{x})\, v(\mathbf{x}) \tag{7.5}$$

Le calcul des matrices élémentaires permet d'obtenir les contributions apportées par le modèle pour chaque élément. On a donc N_{elt} matrices élémentaires à construire. Chacune de ces matrices est de taille $(T_N \times T_N)$, où T_N est le nombre de degrés de liberté de l'élément fini. On suppose que ce nombre est identique pour tous les éléments. Pour pouvoir évaluer les valeurs de ces matrices, nous utilisons la technique de l'élément de référence que nous avons présenté au chapitre 1. Ainsi, nous notons par $\boldsymbol{\varphi}_K^{\text{geo}}$ la transformation géométrique de $\hat{K} \to K$. Nous posons également $J_K = det(\nabla_{\hat{\mathbf{x}}} \boldsymbol{\varphi}_K^{\text{geo}})$ et $\boldsymbol{B}_K = (\nabla_{\hat{\mathbf{x}}} \boldsymbol{\varphi}_K^{\text{geo}})^{-T}$. Pour calculer une de ces matrices, nous utilisons la formulation variationnelle élémentaire du problème. Elle s'obtient de la même manière que pour la formulation (7.5) mais en restreignant l'intégrale à un élément K. Pour notre exemple, nous obtenons deux termes (terme de masse + terme de raideur). Pour l'assemblage de chaque matrice élémentaire, nous choisissons de séparer le calcul de ces termes (matrice de masse et matrice de raideur). Cela nous permet d'optimiser ce calcul en utilisant des formules de quadrature différentes (car ces termes ne sont pas du même ordre polynomial).

Calcul de la matrice de masse élémentaire

On pose $\left\{(\hat{\mathbf{x}}_q^m, w_q^m), 1 \leq\ q \leq N_q^m\right\}$ l'ensemble définissant une formule de quadrature pour le terme de masse. Il contient les couples points de quadrature et poids de quadrature. Par définition et utilisation de la technique de l'élément de référence, nous obtenons l'expression des entrées $(i,j)_{1 \leq i,j \leq T_N}$ de la la matrice élémentaire M_K, $\forall K \in \mathcal{T}_\delta$:

$$\begin{aligned}
M_K(i,j) &= \int_K \Phi_{K,i}^N(\mathbf{x})\, \Phi_{K,j}^N(\mathbf{x})\, d\mathbf{x} \\
&= \int_{\hat{K}} \left[\Phi_{K,i}^N \circ \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})\right] \left[\Phi_{K,j}^N \circ \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})\right] |J_K(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}} \\
&= \int_{\hat{K}} \hat{\Phi}_i^N(\hat{\mathbf{x}})\, \hat{\Phi}_j^N(\hat{\mathbf{x}})\, |J_K(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}} \\
&= \sum_{q=1}^{N_q} w_q^m\ \hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q^m\right) \hat{\Phi}_j^N\left(\hat{\mathbf{x}}_q^m\right) \left|J_K\left(\hat{\mathbf{x}}_q^m\right)\right|
\end{aligned}$$

Calcul de la matrice de raideur élémentaire

Pour pouvoir utiliser la technique de l'élément de référence dans ce cas, nous avons tout d'abord besoin de la relation suivante :

$$\nabla_{\mathbf{x}} \Phi_{K,i}^N(\mathbf{x}) = \left(\boldsymbol{B}_K\ \nabla_{\hat{\mathbf{x}}} \hat{\Phi}_i^N\right) \circ (\boldsymbol{\varphi}_K^{\text{geo}})^{-1}(\mathbf{x}) \tag{7.6}$$

qui est vrai $\forall K \in \mathcal{T}_\delta$ et avec $1 \leq i \leq T_N$. Celle-ci se démontre facilement en utilisant la règle de dérivation en chaine. On pose $\left\{(\hat{\mathbf{x}}_q^r, w_q^r), 1 \leq\ q \leq N_q^r\right\}$ l'ensemble définissant une formule de quadrature pour le terme de raideur. La matrice élémentaire de raideur notée R_K est calculée à l'aide de la formule ci-dessous :

$$
\begin{aligned}
R_K(i,j) &= \int_K \nabla_{\mathbf{x}} \Phi^N_{K,i}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} \Phi^N_{K,j}(\mathbf{x})\, d\mathbf{x} \\
&= \int_{\hat{K}} \left[\boldsymbol{B}_K(\hat{\mathbf{x}}) \ \nabla_{\hat{\mathbf{x}}} \hat{\Phi}^N_i(\hat{\mathbf{x}}) \right] \cdot \left[\boldsymbol{B}_K(\hat{\mathbf{x}}) \ \nabla_{\hat{\mathbf{x}}} \hat{\Phi}^N_j(\hat{\mathbf{x}}) \right] |J_K(\hat{\mathbf{x}})|\, d\hat{\mathbf{x}} \\
&= \sum_{q=1}^{\tilde{N}_q} w^r_q \ \left[\boldsymbol{B}_K(\hat{\mathbf{x}}^r_q) \ \nabla_{\hat{\mathbf{x}}} \hat{\Phi}^N_i(\hat{\mathbf{x}}_q) \right] \cdot \left[\boldsymbol{B}_K(\hat{\mathbf{x}}^r_q) \ \nabla_{\hat{\mathbf{x}}} \hat{\Phi}^N_j(\hat{\mathbf{x}}^r_q) \right] \left| J_K(\hat{\mathbf{x}}^r_q) \right|
\end{aligned}
$$

Algorithme d'assemblage local sur CPU

Pour terminer cette partie, nous proposons l'algorithme 7 permettant la construction de l'ensemble des matrices élémentaires sur un CPU pour les termes de masse et de raideur. Cet algorithme sera ensuite comparé à celui qui été implémenté avec le GPU. Ces tests nous permettront d'avoir des indications sur la scalabilité de l'algorithme.

Algorithm 7 Assemblage local de la matrice de masse+raideur sur CPU

Require: $(w^m_q, \hat{\mathbf{x}}^m_q)$ et $(w^r_q, \hat{\mathbf{x}}^r_q)$: deux formules de quadrature
Require: $\hat{\Phi}^N_i(\hat{\mathbf{x}}_q)$: fonctions de base de l'élément fini sur $\hat{K}$
Require: $\mathcal{T}_\delta$: un maillage

Ensure: MR_K : les matrices élémentaires

```
for K ∈ T_δ do
  for i = 1, ..., T_N do
    for i = 1, ..., T_N do
      sum = 0
      for q = 1, ..., N^m_q do
        sum+ = w^m_q Φ̂^N_i(x̂^m_q) Φ̂^N_j(x̂^m_q) |J_K(x̂^m_q)|
      end for
      for q = 1, ..., N^r_q do
        sum+ = w^r_q [B_K(x̂^r_q) ∇_x̂ Φ̂^N_i(x̂^r_q)] · [B_K(x̂^r_q) ∇_x̂ Φ̂^N_j(x̂^r_q)] |J_K(x̂^r_q)|
      end for
      MR_K(i, j) = sum
    end for
  end for
end for
```

3.2 Implémentation sur GPU avec CUDA

Le langage CUDA offre la possibilité d'exécuter des fonctions directement sur les unités de calcul de la carte graphique en parallèle. Ces fonctions définies par l'utilisateur sont appelées noyaux et elles sont également en charge de définir le nombre de threads souhaités pour l'exécution. Les threads GPU sont répartis dans des structures de type blocs, qui sont elles-mêmes étagées dans une structure de type grille. Cette structure est représentée

avec la figure 7.20. La grille est définie comme un tableau à deux dimensions et le bloc comme un tableau à une, deux ou trois dimensions. On obtient le nombre de threads avec la formule suivante :

$$\text{nb threads} = \text{gridDim.x*gridDim.y*blockDim.x*blockDim.y*blockDim.z.}$$

La structure des blocs et la grille sont nécessaires pour savoir quels threads partagent la même section locale mémoire. Il existe plusieurs types de mémoire dans les cartes graphiques. Il y a la mémoire globale qui est partagée par tous les threads, la mémoire locale qui est partagée par tous les threads d'un bloc, la mémoire constante qui est accessible par tous les threads en lecture seule et la mémoire texture qui est optimisé pour un espace à deux dimensions. Hormis la mémoire globale, les autres types de mémoire sont de taille relativement petite, mais plus rapide que la mémoire globale dont la taille dépend de la carte graphique. L'accessibilité des différentes mémoires est répertoriée dans la table 7.6.

Mémoire	Disponible dans	Avec cache
globale	grille	non
local	bloc	oui
constante	grille	oui
texture	grille	non

TABLE 7.6 – Les types de mémoire dans un GPU.

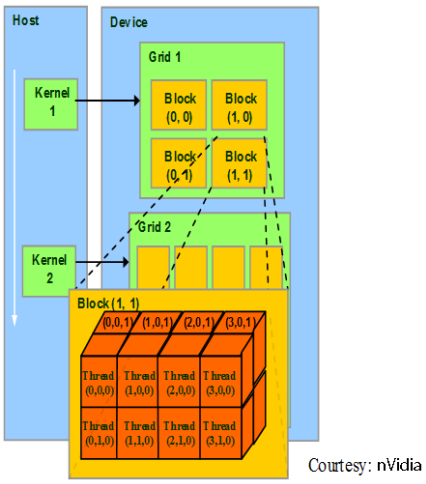


FIGURE 7.20 – Exemple de structure de grille de calcul avec CUDA.

Une autre chose importante avec CUDA est le transfert de mémoire entre GPU et CPU. Ces transferts de mémoire peuvent être un goulot d'étranglement. Ils pourraient détériorer complètement les performances du GPU. L'utilisation des communications asynchrones permet d'éviter ce désagrément.

Application à l'assemblage local

Nous allons maintenant construire un code CUDA permettant de réaliser l'assemblage des matrices élémentaires. On commence par définir la distribution des calculs sur les différents processus de la carte GPU. Nous avons alors fait le choix de répartir le calcul de chaque composante d'une matrice élémentaire sur les processus. On itère ce procédé sur tous les éléments $K \in \mathcal{T}_\delta$. Chaque processus va donc calculer pour un K donné, la composante $MR_K(i,j)$. Ainsi, la grille et les blocs seront représentés par une topologie bidimensionnelle. La figure 7.21 représente cette topologie, avec $\mathbf{B_{xy}}$ un bloc et $\mathbf{T_{xy}}$ un processus d'un bloc.

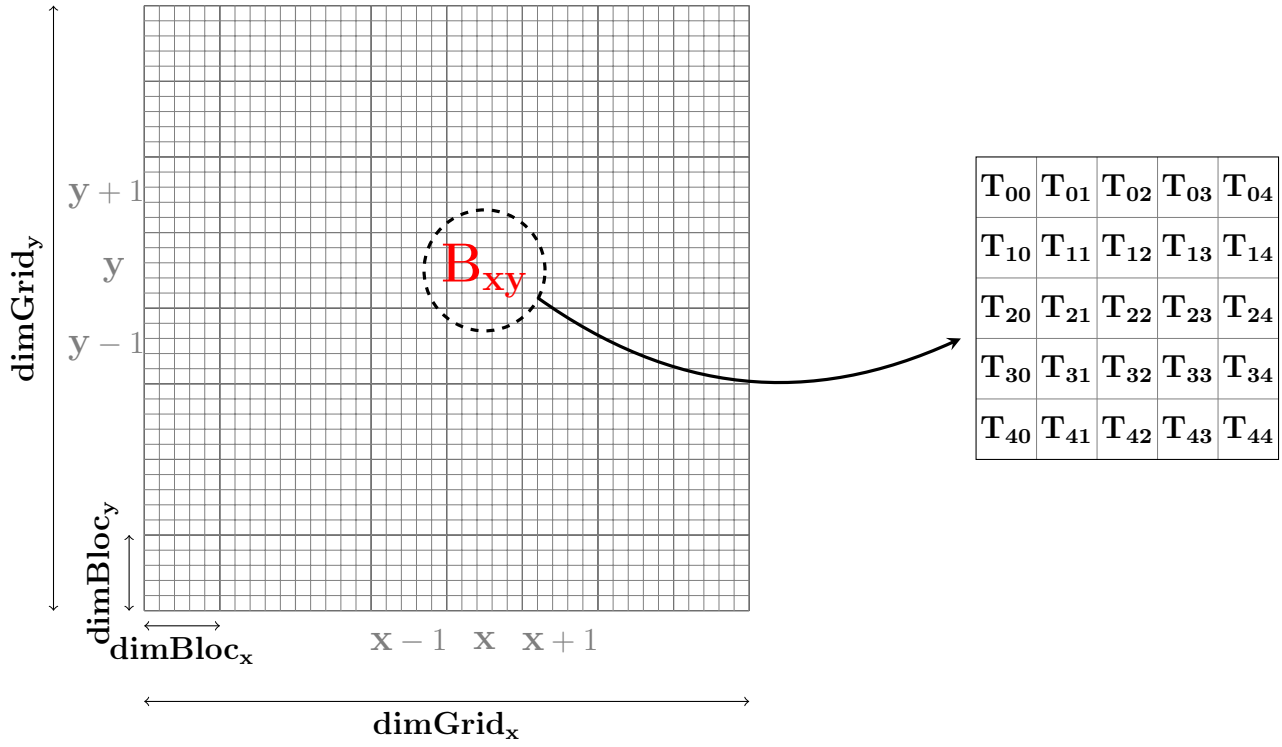


FIGURE 7.21 – Topologie bi-dimensionnelle des processus

Pour décrire avec CUDA cette structure, nous commençons par donner la taille d'un bloc en fixant deux entiers $\mathbf{dimBloc_x}$ et $\mathbf{dimBloc_y}$. Ensuite, il faut définir la taille de la grille en fonction de T_N, le nombre de degrés de liberté de l'élément, ce qui donne dans notre cas :

$$\mathbf{dimGrid_x} = \frac{T_N}{\mathbf{dimBloc_x}} \qquad \mathbf{dimGrid_y} = \frac{T_N}{\mathbf{dimBloc_y}}$$

Comme $\mathbf{dimBloc_x}$ et $\mathbf{dimBloc_y}$ ne sont pas forcément des diviseurs de T_N, un choix plus sûr est :

$$\mathbf{dimGrid_x} = \frac{T_N}{\mathbf{dimBloc_x}} + 1 \qquad \mathbf{dimGrid_y} = \frac{T_N}{\mathbf{dimBloc_y}} + 1$$

Avec cette définition, on aura un petit peu plus de processus que nécessaire. Il faudra alors vérifier qu'il n'y a pas de calculs sur les processus de trop et pour cela, on utilise les relations suivantes. Elle permet de connaitre l'indice de la matrice élémentaire (i, j) à partir de l'indice de bloc $\mathbf{B_{xy}} = (B_\mathbf{x}, B_\mathbf{y})$ et de l'indice de processus par bloc $\mathbf{T_{xy}} = (T_\mathbf{x}, T_\mathbf{y})$:

$$i = B_\mathbf{x}\ \mathbf{dimBloc_x} + T_\mathbf{x}$$
$$j = B_\mathbf{y}\ \mathbf{dimBloc_y} + T_\mathbf{y}$$

Avec cette topologie des processus, nous pouvons établir un algorithme de calcul composé des deux parties, une première sur CPU et une autre sur le GPU. L'algorithme 8 présente le transfert de calcul effectué depuis le CPU vers le GPU. On commence par

envoyer les données sur la carte graphique. Ensuite on appelle la fonction de calcul sur GPU pour tous les éléments du maillage. Cette fonction est décrite par l'algorithme 9. Tous les processus du GPU vont exécuter cette suite d'instruction. Une fois terminer, une dernière étape de récupération des résultats est nécessaire.

Algorithm 8 Assemblage local sur GPU (Partie CPU)

Require: Précalculs (w_q^m, $\hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q^m\right)$, $J_K\left(\hat{\mathbf{x}}_q^m\right)$, ...) disponible sur le CPU
Ensure: Les matrices élémentaires

≪ Envoi des données d'entrées du CPU vers le GPU ≫
for $K = 1, ..., N_{elt}$ **do**
  ≪ Appel à la fonction Assemblage local (Partie GPU) ≫
end for
≪ Réception des résultats du GPU vers le CPU ≫

Algorithm 9 Assemblage local sur GPU (Partie GPU)

Require: Précalculs (w_q^m, $\hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q^m\right)$, $J_K\left(\hat{\mathbf{x}}_q^m\right)$, ...) envoyés par le CPU
Ensure: Une matrice élémentaire

$i = B_{\mathbf{x}}\ \mathbf{dimBloc_x} + T_{\mathbf{x}}$
$j = B_{\mathbf{y}}\ \mathbf{dimBloc_y} + T_{\mathbf{y}}$
if $(i < T_N$ et $i < T_N)$ **then**
  $sum = 0$
  for $q = 1, ..., N_q^m$ **do**
    $sum+ = w_q^m\ \hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q^m\right)\hat{\Phi}_j^N\left(\hat{\mathbf{x}}_q^m\right)\left|J_K\left(\hat{\mathbf{x}}_q^m\right)\right|$
  end for
  for $q = 1, ..., N_q^r$ **do**
    $sum+ = w_q^r\ \left[\boldsymbol{B}_K\left(\hat{\mathbf{x}}_q^r\right)\ \nabla_{\hat{\mathbf{x}}}\hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q^r\right)\right]\cdot\left[\boldsymbol{B}_K\left(\hat{\mathbf{x}}_q^r\right)\ \nabla_{\hat{\mathbf{x}}}\hat{\Phi}_j^N\left(\hat{\mathbf{x}}_q^r\right)\right]\left|J_K\left(\hat{\mathbf{x}}_q^r\right)\right|$
  end for
  $MR_K(i,j) = sum$
end if

Cet algorithme fonctionne bien, mais il a le désavantage de copier toutes les données sur la carte GPU. Ainsi, lorsque le nombre d'éléments devient trop grand, la mémoire « explose ». Pour pallier à cette contrainte, il faut découper le transfert de données en une liste de paquet. Tout d'abord, on se donne le nombre d'éléments que l'on va copier sur le GPU, **Size_batch**. On en déduit le nombre de paquets par :

$$\mathbf{Nb\_batch} = \frac{N_{elt}}{\mathbf{Size\_batch}}$$

avec N_{elt} le nombre d'éléments du maillage. Cependant, **Size_batch** n'est pas forcément un diviseur de N_{elt}. On considère alors **Nb_batch** comme étant le quotient de cette division euclidienne. il faut alors rajouter une dernière étape sur les éléments manquants

avec le reste de la division. Ainsi, nous utilisons l'algorithme 10 pour la partie CPU de notre implémentation avec CUDA.

Algorithm 10 Assemblage local sur GPU avec batch (Partie CPU)

Require: Précalculs $(w_q^m,\ \hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q^m\right),\ J_K\left(\hat{\mathbf{x}}_q^m\right),\ ...)$ disponibles sur le CPU
Ensure: Les matrices élémentaires

for $i = 1, \mathbf{Nb\_batch}$ **do**
  $\ll$ *Envoi d'un paquet de données d'entrées du CPU vers le GPU* $\gg$
  for $K = i * Size\_batch, (i+1) * Size\_batch$ **do**
    $\ll$ *Appel à la fonction Assemblage local (Partie GPU)* $\gg$
  end for
  $\ll$ *Réception d'un paquet de résultats du GPU vers le CPU* $\gg$
end for
$\ll$ *Envoi du dernier paquet de données du CPU vers le GPU* $\gg$
$\ll$ *Appel à la fonction Assemblage local (Partie GPU)* $\gg$
$\ll$ *Réception du dernier paquet de résultats du GPU vers le CPU* $\gg$

Résultats de scalabilité

Pour finir, nous avons effectué des tests de scalabilité entre les versions CPU et GPU de l'assemblage. Pour ce test, nous n'avons pas utilisé de données réelles (fonctions de base, éléments, maillages,...). Ces données ont été initialisées avec des valeurs aléatoires. Cependant, le nombre de ces entrées est choisi de manière réaliste $(N_q, T_N, ...)$. On note N l'ordre polynomial. Le nombre de point de quadrature N_q sera choisie en fonction de la dimension et de l'ordre polynomial de telle sorte que les formules d'intégration soient exactes. Dans ces tests, nous mesurerons uniquement les opérations d'assemblage et de communication. Les blocs de processus seront toujours choisis carrés. On pose alors $\mathbf{dimBloc} = \mathbf{dimBloc_x} = \mathbf{dimBloc_y}$.

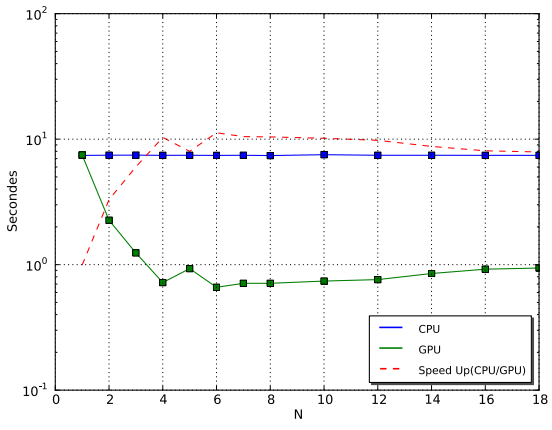


FIGURE 7.22 – Temps d'assemblage en 3D (masse+raideur) en fonction de **dimBloc** avec $N = 6$, $N_{elt} = 200$.

dimBloc	CPU	GPU	Speed Up
2	7.45	2.26	3.2964
3	7.46	1.24	6.01613
4	7.43	0.72	10.3194
5	7.44	0.93	8.0
6	7.42	0.66	11.2424
7	7.44	0.71	10.4789
8	7.4	0.71	10.4225
10	7.52	0.74	10.1622
12	7.44	0.76	9.7894
14	7.44	0.85	8.75294
16	7.43	0.92	8.07609
18	7.43	0.94	7.90426

TABLE 7.7 – Table associée à la figure 7.22.

Nous commençons par présenter la figure 7.22 et la table 7.7. Elle illustre l'influence de la taille des blocs par rapport aux performances obtenues avec la carte GPU. On constate une forte détérioration pour **dimBloc** < 4. Les meilleurs speed-up sont obtenus entre 4 et 12, avec un maximum pour **dimBloc** $= 6$.

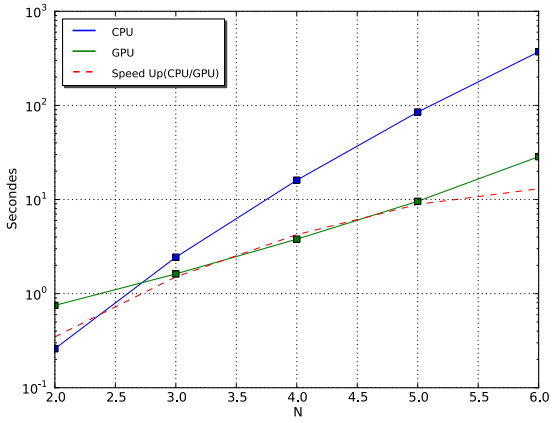


FIGURE 7.23 – Temps d'assemblage en 3D (masse+raideur) en fonction de l'ordre polynomial N avec $N_{elt} = 10000$ et **dimBloc** $= 6$.

N	CPU	GPU	Speed Up
2	0.26	0.75	0.346667
3	2.44	1.62	1.50617
4	16.04	3.79	4.23219
5	84.97	9.57	8.87879
6	371.06	28.52	13.0105

TABLE 7.8 – Table associée à la figure 7.23.

Quant à la figure 7.23 et la table 7.8, elles illustrent l'influence de l'ordre polynomial. Plus cet ordre est élevé et plus notre stratégie est efficace. On commence à obtenir des performances intéressantes à partir de l'ordre 4. Nous avons un speed-up de 13 pour le cas $N = 6$. D'autres facteurs d'influence sont illustrés dans l'annexe B. Nous retrouvons toujours la remarque précédente sur l'ordre élevé. Nous avons aussi une comparaison entre l'assemblage séparé de la matrice de masse et de raideur. On constate alors que l'augmentation de la complexité des termes assemblés permet d'avoir un meilleur speed-up. On montrera aussi que le nombre d'éléments n'a pas d'influence sur les performances. Toutefois, d'autres stratégies peuvent être envisagées. Par exemple, nous pouvons essayer d'assembler plus de matrices élémentaires en même temps. Cela devrait permettre d'améliorer les performances dans le cas des ordres d'approximation faibles et/ou de dimension petite (1D/2D).

Enfin, nous faisons la remarque que cette stratégie d'assemblage sur GPU n'est pas encore intégré à FEEL++. Il s'agit juste d'un programme indépendant permettant de réaliser des tests de scalabilité.

Chapitre 8

Autres outils intégrés dans Feel++

La librairie Feel++ est un outil numérique qui permet de résoudre des équations aux dérivées partielles. Son développement repose sur le paradigme des DSEL (*domain specific embedded language*). En d'autres mots, cet outil propose un langage embarqué dans le C++ pour la résolution d'EDP en restant très proche des mathématiques. En particulier, Feel++ est un outil conçu pour la résolution d'EDP via les méthodes de Galerkin. Le principe fondamental de ces langages est de rendre transparentes la complexité et la puissance des algorithmes informatiques tout en gardant une large flexibilité et une vision claire des mathématiques. Ainsi, plus les mathématiques sont claires au niveau du numérique et plus il est facile et fiable de développer des méthodes numériques de complexité supérieure.

Pour le développement numérique de cette thèse, un certain nombre d'outils numériques ont été développés dans le sens de la DSEL. Dans les chapitres précédents, nous avons déjà illustré l'intégration d'outils majeurs pour la simulation des écoulements sanguins : les opérateurs d'interpolation, les méthodes de Galerkin non standards et le parallélisme. Nous souhaitons dans cette partie présenter d'autres fonctionnalités qui ont été développées. La complexité de ces nouveaux outils est moins importante, mais leur intérêt reste très fort dans le développement de ce langage informatique pour les mathématiques. Cette partie est essentiellement destinée aux développeurs et utilisateurs de la librairie Feel++.

Dans ce chapitre, nous allons commencer par présenter des simplifications et des optimisations faites pour le calcul intégral. Nous passerons ensuite à la description d'un outil d'aide à la définition de géométrie. Ce développement sera particulièrement utile pour la FBM. Puis, nous parlerons de parallélisme en présentant brièvement deux outils : une interface pour la communication de processus et un opérateur d'interpolation entre des espaces de fonctions parallèles. Enfin, nous finirons par la description de l'utilisation des modèles physiques développés pour la mécanique des fluides, des solides, de la FBM et de l'interaction fluide structure.

1 Calculs d'intégrales et optimisations

Nous allons présenter ici quelques aspects importants qui ont été améliorés dans Feel++ pour le calcul intégral. Nous parlerons tout d'abord du calcul automatique de l'ordre utilisé dans les formules de quadrature. Nous présenterons ensuite les optimisations faites pour le calcul d'intégrale basé sur des géométries d'ordre élevé (>1).

1.1 Choix automatique de la formule de quadrature

Pour illustrer cette fonctionnalité, nous allons considérer un exemple de calcul d'intégrale. On note par $\Omega_\delta \equiv \Omega_{h,k}$ un domaine de calcul représentant le recouvrement d'un maillage $\mathcal{T}_\delta$. On note $\boldsymbol{\varphi}_K^{\text{geo}}$ la transformation géométrique entre un élément de référence $\hat{K}$ et un élément réel $K \in \mathcal{T}_\delta$. On pose aussi $J_K = det(\nabla \boldsymbol{\varphi}_K^{\text{geo}})$. On choisit une fonction $u \in P_c^N(\Omega_\delta)$. Pour établir une intégration numérique, nous considérons avoir une formule de quadrature définie sur l'élément de référence $\hat{K}$ et caractérisé par l'ensemble $\{(\hat{\mathbf{r}}_q, w_q), i = 1, ..., N_q\}$. Elle contient les points de quadrature $\hat{\mathbf{r}}_q \in \hat{K}$ et des poids de quadrature w_q.

À l'aide de ces notations, nous pouvons exprimer la formulation numérique utilisée pour l'évaluation de l'intégrale de la fonction u sur le domaine Ω_δ :

$$\int_{\Omega_\delta} u(\mathbf{x})\ d\mathbf{x} = \sum_{K \in \mathcal{T}_\delta} \int_K u(\mathbf{x})\ d\mathbf{x} = \sum_{K \in \mathcal{T}_\delta} \int_{\hat{K}} u \circ \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})\ \ J_K(\hat{\mathbf{x}})\ d\hat{\mathbf{x}} \tag{8.1}$$

$$= \sum_{K \in \mathcal{T}_\delta} \sum_{q}^{N_q} w_q\ u \circ \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{r}}_q)\ J_K(\hat{\mathbf{r}}_q) \tag{8.2}$$

Par définition, la fonction u est une fonction polynomiale d'ordre N sur chaque élément K. La transformation géométrique $\boldsymbol{\varphi}_K^{\text{geo}}$ et le jacobien J_K sont aussi des fonctions polynomiales d'ordre respectif k et $k-1$. Ainsi, l'expression $u \circ \boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})\ \ J_K(\hat{\mathbf{x}})$ est une fonction polynomiale d'ordre $N * k + k - 1$. Nous pouvons alors obtenir une intégration exacte si l'on utilise une formule de quadrature assez précise. Dans cet exemple, nous avons besoin d'une quadrature qui soit au minimum d'ordre égal à l'ordre polynomial de l'expression. Ainsi, il est possible d'effectuer des intégrations numériques de manière exacte tant que l'expression à intégrer est polynomiale sur chaque élément du maillage. L'utilisation de l'ordre de quadrature minimal est un choix optimal. Elle permet d'obtenir une intégration parfaite avec un minimum d'évaluation des points de quadrature.

Nous décrivons maintenant un exemple d'intégration dans FEEL++. On choisi u1 $\in P_c^1(\Omega_\delta)$, u2 $\in P_c^2(\Omega_\delta)$ et u3 $\in P_c^3(\Omega_\delta)$. On suppose que l'ordre géométrique des éléments est $k = 2$. On considère alors l'intégrale numérique exprimée par :

```
integrate(_range=elements(mesh),
          _expr=Px()*(Px()-1)*idv(u1)*gradv(u2)*trans(gradv(u3))
          _quad=_Q<13>() )
```

Dans cette formule, nous avons imposé un ordre de quadrature de 13 avec le dernier argument. C'est l'ordre minimal pour obtenir une intégration numérique exacte. Nous souhaitons alors pouvoir rendre automatique le calcul de cet ordre. Cela permettrait de rendre optionnelle l'option `_quad`. Dans notre librairie, la principale difficulté pour calculer cette constante d'intégration est que les opérations doivent s'effectuer à la compilation et non à l'exécution. En effet, l'ordre de quadrature est transmis à la fonction `integrate` à l'aide à l'object `_Q<13>()`. L'entier 13 exprime cet ordre et correspond à un argument template de la classe `_Q`. Nous allons donc devoir utiliser le concept de la méta-programmation par template C++.

La première étape consiste à ajouter chaque classe C++ représentant un mot clé du langage des données représentant l'ordre polynomial de ce mot clé. Nous trouverons alors

dans ces classes la déclaration suivante avec `uint16_type` un type pour les entiers positifs représentés sur 16 bits :

```
static const uint16_type imorder = ...
```

Par exemple, le mot clé `Px()` est une expression d'ordre 1. Nous trouverons explicitement dans le code l'entier 1. Pour `gradv(u3)`, cette expression est d'ordre 2 (ordre de u3 moins un). L'ordre recherché dépend donc de l'ordre polynomial de l'espace de fonction de u3. Nous trouverons dans ce cas :

```
static const uint16_type imorder = space_type::basis_type::nOrder-1
```

Une fois que l'on a ajouté cette donnée de base dans tous les mots clés de base, il faut faire la même chose pour toutes les opérations (+,-,*,...) possible entre les mots clés. Ces opérations génèrent un type C++ complexe (une structure d'arbre) représentant l'expression. La dernière étape est opérée dans la fonction `integrate`. Nous avons accès à l'ordre polynomial N de l'expression et à l'ordre k de la transformation géométrique. Nous pouvons ainsi calculer l'ordre de quadrature avec $N * k + k - 1$. Ainsi, l'expression FEEL++ suivante est identique à l'expression présentée avant :

```
integrate(_range=elements(mesh),
          _expr=Px()*(Px()-1)*idv(u1)*gradv(u2)*trans(gradv(u3)) )
```

Dans le cas où l'expression est composée de fonctions non polynomiales, par exemple en utilisant cos(...), l'intégration ne peut plus être calculée exactement. Pour améliorer la précision des calculs, il faut augmenter l'ordre de quadrature. Nous proposons quand même un calcul automatique de l'ordre de quadrature pour ce cas. Cependant, il peut être parfois nécessaire de donner explicitement cette donnée pour obtenir un meilleur résultat.

1.2 Optimisation des intégrales pour l'ordre élevé géométrique

L'intégration sur les maillages d'ordre élevé est assez coûteuse, car elle nécessite d'utiliser des ordres de quadrature assez importants. Ce coût important se constate avec l'ordre d'intégration minimal égal à $N * k + k - 1$. Pour limiter ce coût, nous allons proposer une optimisation pour les calculs d'intégrale sur des maillages d'ordre élevé. Tout d'abord, nous considérons l'hypothèse cruciale présentée au chapitre 2 qui suppose que les éléments internes au maillage sont d'ordre 1. Les faces des éléments ont été raidifiées. Nous allons alors décomposer en deux parties l'intégration numérique présentée avec les équations (8.1)-(8.2). Nous utiliserons toujours la transformation géométrie d'ordre élevé pour les éléments qui auront une courbure. Pour le reste des éléments supposés droits, nous pourrons appliquer simplement une transformation géométrie d'ordre 1. Nous avons représenté ces deux types de transformation, notées $\boldsymbol{\varphi}^{\text{geo}}_{K,k}$ et $\boldsymbol{\varphi}^{\text{geo}}_{K,1}$, avec la figure 8.1. Pour les éléments droits, nous avons naturellement l'équivalence $\boldsymbol{\varphi}^{\text{geo}}_{K,k} \equiv \boldsymbol{\varphi}^{\text{geo}}_{K,1}$.

Nous considérons un maillage $\mathcal{T}_\delta$ formant un domaine de calcul Ω_δ. Nous utilisons la notation ∂K pour désigner l'ensemble des faces de l'élément K et la notation $\partial^2 K$ pour désigner l'ensemble des sommets de l'élément K. Cela nous permet de définir le sous-ensemble $\mathcal{T}^b_\delta = \{K \in \mathcal{T}_\delta : (\partial K \backslash \partial^2 K) \cap \partial\Omega_\delta \neq \emptyset\}$ représentant l'ensemble des éléments K qui partagent une face (en 2D et 3D) ou une arrête (en 3D) avec la frontière de Ω_δ. Nous posons $\mathcal{T}^i_\delta = \mathcal{T}_\delta \backslash \mathcal{T}^b_\delta$ l'ensemble des éléments internes (seul un sommet peut toucher

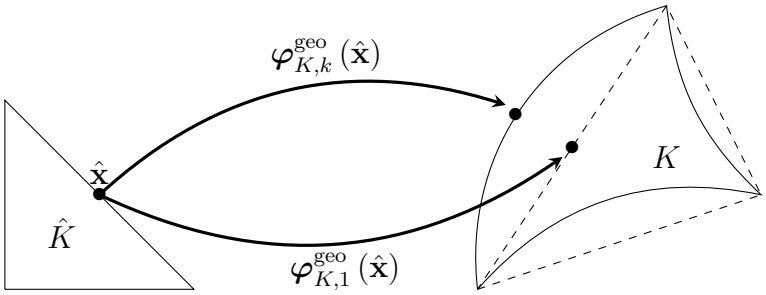


FIGURE 8.1 – Transformations géométriques d'ordre 1 et d'ordre élevé.

le bord). Nous pouvons ainsi exprimer le calcul d'intégrale avec cette décomposition :

$$\int_{\Omega_\delta} u(\mathbf{x})\ d\mathbf{x} = \sum_{K\in\mathcal{T}_\delta^b}\int_K u(\mathbf{x})\ d\mathbf{x} + \sum_{K\in\mathcal{T}_\delta^i}\int_K u(\mathbf{x})\ d\mathbf{x} \tag{8.3}$$

$$= \sum_{K\in\mathcal{T}_\delta^b}\int_{\hat{K}} u\circ\boldsymbol{\varphi}^{\text{geo}}_{K,k}(\hat{\mathbf{x}})\ \ J_{K,k}(\hat{\mathbf{x}})\ d\hat{\mathbf{x}} \tag{8.4}$$

$$+ \sum_{K\in\mathcal{T}_\delta^i}\int_{\hat{K}} u\circ\boldsymbol{\varphi}^{\text{geo}}_{K,1}(\hat{\mathbf{x}})\ \ J_{K,1}(\hat{\mathbf{x}})\ d\hat{\mathbf{x}} \tag{8.5}$$

L'intégration numérique va maintenant s'effectuer à l'aide de deux formules de quadrature. Une première est utilisée pour l'ordre élevé et elle est définie par l'ensemble $\left\{\left(\hat{\mathbf{r}}_q^k, w_q^k\right), i = 1, ..., N_q^k\right\}$. Une seconde, définie par $\left\{\left(\hat{\mathbf{r}}_q^1, w_q^1\right), i = 1, ..., N_q^1\right\}$, nous permettra l'intégration des éléments internes. Ainsi, nous obtenons les formules d'intégration suivantes :

$$\int_{\hat{K}} u\circ\boldsymbol{\varphi}^{\text{geo}}_{K,k}(\hat{\mathbf{x}})\ \ J_{K,k}(\hat{\mathbf{x}})\ d\hat{\mathbf{x}} = \sum_q^{N_q^k} w_q^k\ u\circ\boldsymbol{\varphi}^{\text{geo}}_{K,k}\left(\hat{\mathbf{r}}_q^k\right)\ \ J_{K,k}(\hat{\mathbf{r}}_q^k) \qquad ,\forall K\in\mathcal{T}_\delta^b \tag{8.6}$$

$$\int_{\hat{K}} u\circ\boldsymbol{\varphi}^{\text{geo}}_{K,1}(\hat{\mathbf{x}})\ \ J_{K,1}(\hat{\mathbf{x}})\ d\hat{\mathbf{x}} = \sum_q^{N_q^1} w_q^1\ u\circ\boldsymbol{\varphi}^{\text{geo}}_{K,1}\left(\hat{\mathbf{r}}_q^1\right)\ \ J_{K,1}(\hat{\mathbf{r}}_q^1) \qquad ,\forall K\in\mathcal{T}_\delta^i \tag{8.7}$$

Ainsi, pour obtenir des intégrations numériques optimales, nous devrons choisir des ordres de quadrature. L'ordre de quadrature utilisé pour les éléments $K \in \mathcal{T}_\delta^b$ sera toujours égale à $N * k + k - 1$. Par contre, pour les éléments $K \in \mathcal{T}_\delta^i$, une quadrature d'ordre N sera suffisante. Cela permettra d'accélérer considérablement les calculs, car généralement le cardinal de $\mathcal{T}_\delta^i$ est bien plus important que celui de $\mathcal{T}_\delta^b$.

Dans la librairie FEEL++, ces ordres d'intégration seront également calculés automatiquement. Cependant, il peut être nécessaire de les imposer. En reprenant l'exemple présenté dans la partie précédente, nous pouvons écrire :

```
integrate(_range=elements(mesh),
          _expr=Px()*(Px()-1)*idv(u1)*gradv(u2)*trans(gradv(u3))
          _quad=_Q<13>(),
          _quad1=_Q<6>() )
```

avec l'option `_quad` (resp `_quad1`) désignant la quadrature utilisée pour les éléments $K \in \mathcal{T}_\delta^b$ (resp $K \in \mathcal{T}_\delta^i$). Les ordres de quadratures utilisés dans ce code correspondent aux ordres minimaux pour obtenir un calcul numérique exact.

2 Classe `GeoTool` : un outil de description de géométrie

Nous allons maintenant aborder la description d'un outil qui a été développé tout au long de cette thèse. Ce développement a été motivé dans l'idée de simplifier la description des géométries dans les codes FEEL++. Toutes les fonctionnalités décrites dans cette partie reposent sur l'implémentation d'une classe C++ nommée `GeoTool`. Elle propose la définition de plusieurs géométries de base qui sont caractérisées par quelques paramètres pratiques. Nous pouvons aussi interagir entre ces objets pour créer des formes plus complexes. Les opérations disponibles sont l'union et la différence. Nous montrerons également l'implémentation d'une géométrie de base. Cette description a pour but de montrer l'effort de développement qui a été fait dans la classe `GeoTool` pour permettre l'ajout de nouvelles formes par un procédé relativement simple. Tout ce travail permet de générer un code qui est peut-être décrypté par la libraire GMSH [67]. Celle-ci nous permet ensuite de générer le maillage associé à la géométrie.

2.1 Formes de base

Un certain nombre de formes géométriques a déjà été implémenté dans la classe `GeoTool`. Nous trouverons les formes classiques telles que les triangles, quadrangles, divers autres polygones, cercles, hexaèdres, sphères, cylindres, etc. D'autres figures plus spécifiques sont aussi présentes.

Pour illustrer cet environnement, nous commençons par présenter le code correspondant à la définition d'un domaine représenté par un quadrangle. Il sera donc paramétré par ses quatre sommets. Le code suivant permet de définir une telle géométrie :

```
GeoTool::Node x1(0,0);GeoTool::Node x2(2.3,0.2);
GeoTool::Node x3(2.6,2.2);GeoTool::Node x4(-0.2,2.4);
GeoTool::Quadrangle Q( meshSize,"OMEGA",x1,x2,x3,x4 );
Q.setMarker(_type="line",_name="Boundary1",_marker3=true);
Q.setMarker(_type="line",_name="Boundary2",_marker1=true,
            _marker2=true,_marker4=true);
Q.setMarker(_type="surface",_name="Omega",_markerAll=true);
```

Dans les deux premières lignes de ce code, nous trouvons la définition des quatre sommets décrivant le quadrangle. Nous avons ensuite la déclaration de l'objet géométrique de type Quadrangle caractérisé par :

- `meshSize` : une taille caractéristique des éléments du maillage.
- `''OMEGA''` : un identificateur unique de la forme (a de l'importance pour les opérations entre les objets `GeoTool`).
- `x1,...,x4` : les paramètres décrivant la géométrie (dans cet exemple ce sont les quatre sommets).

Nous avons ensuite la description des marqueurs physiques qui pourront être utilisés avec FEEL++. À l'aide des marqueurs élémentaires définis dans la forme géométrique, voir figures 8.2 avec les $\hat{\Gamma}_i$, nous pouvons définir des marqueurs physiques regroupés dans une liste de marqueurs élémentaires. Dans notre exemple, les marqueurs physiques sont illustrés dans la figure 8.3 avec les Γ_i. Comme le montre le code précédent, nous avons $\Gamma_1 = \hat{\Gamma}_3$ et $\Gamma_2 = \hat{\Gamma}_1 \cup \hat{\Gamma}_2 \cup \hat{\Gamma}_4$.

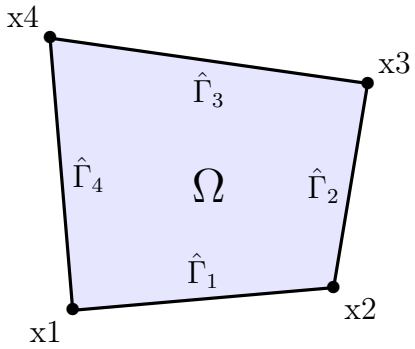


FIGURE 8.2 – Géométrie avec les marqueurs élémentaires.

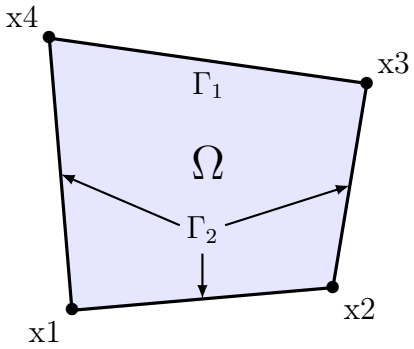


FIGURE 8.3 – Géométrie avec les marqueurs physiques.

Une fois que nous avons décrit la géométrie, il nous faut générer le maillage et le charger dans une structure de donnée de FEEL++. Cette opération est réalisée simplement à l'aide du code ci-dessous :

```
typedef Mesh<Simplex<2,1,2> > mesh_type;
auto mesh = Q.createMesh(_mesh=new mesh_type,_name="mymesh");
```

Cette opération génère deux fichiers. Un premier, nommé mymesh.geo, contient la description géométrique dans le format GMSH. Le deuxième, nommé mymesh.msh, représente le maillage généré à partir du fichier géométrique à l'aide de la librairie GMSH. D'autres options sont disponibles dans la fonction `createMesh`. Les principales sont :

- `_refine` : permet de demander un niveau de rafinement du maillage par subdivision des éléments.
- `_partitions` : nombre de partitions souhaitées (par défaut, le nombre de processeurs dans l'environement).
- `_hmin` et `_hmax` : le taille caractéristique minimum et maximum des éléments dans le maillage.
- `_optimize3d_netgen` : pouvoir utiliser un outil d'optimisation de maillage 3D.

2.2 Opérations sur les géométries

À partir de ces formes élémentaires, nous pouvons ensuite appliquer des opérations d'union et de différence entre les géométries GeoTool. Les opérateurs d'union et de différence sont respectivement symbolisés par le signe « + » et le signe « - » :

```
// operation union
GeoTool::Quadrangle Q1(....);
GeoTool::Quadrangle Q2(....);
auto geoUnion = Q1+Q2;
// operation difference
GeoTool::Quadrangle Q(....);
GeoTool::Circle C(....);
auto geoDiff = Q-C;
```

Dans ce code, nous trouvons deux exemples de création de géométrie composite. Ils peuvent être représenté par les figures 8.4 et 8.5. Les maillages associés sont ensuite créés en appelant la fonction `createMesh`.

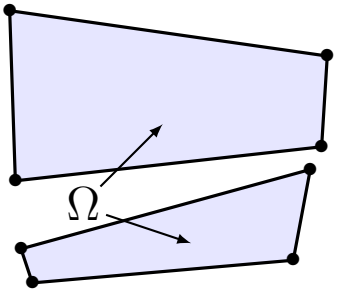


FIGURE 8.4 – Géométrie avec les marqueurs élémentaires.

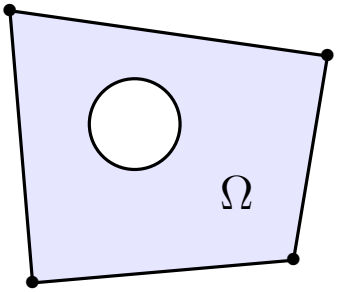
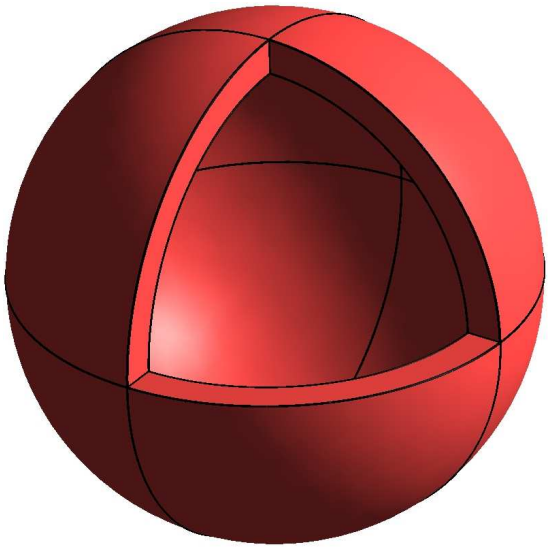
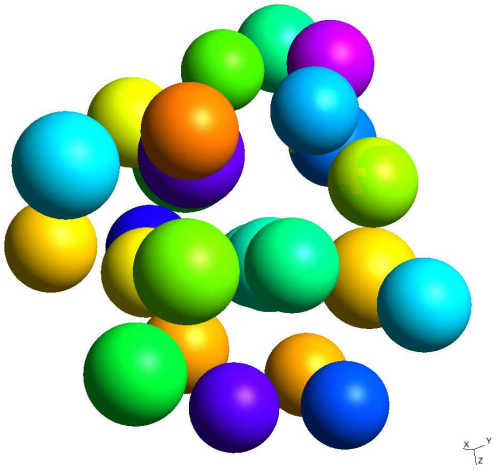


FIGURE 8.5 – Géométrie avec les marqueurs physiques.

Les marqueurs physiques définis dans chacune des formes de base sont également présents dans la géométrie composite. Si deux marqueurs physiques ont des noms identiques dans deux objets différents, alors ce marqueur les représentera tous dans la forme composite. Il existe néanmoins quelques contraintes sur ces opérations. L'opérateur différence fonctionne uniquement dans le cas où la géométrie à soustraire est strictement incluse dans la géométrie support (i.e. $A - B$ fonctionne ssi $B \subset A$). L'opérateur d'union n'est pas un opérateur de fusion de forme. Il permet de créer des géométries composées de plusieurs formes de base, mais ces formes ne sont pas vraiment connectées entre elles. Enfin, nous avons parlé d'un identificateur unique dans la définition des formes de bases. Cet identifiant est utilisé lors d'opérations relativement complexes. Toutefois, cet argument devrait disparaitre dans le futur.



(a) Un domaine local



(b) 25 domaines locaux

FIGURE 8.6 – Application des GeoTools pour la FBM.

Ces opérations sont d'une grande utilité pour la méthode FBM. Elle permet de générer très facilement un grand nombre de domaines locaux et de perforations. Par exemple, on considère `S1a` une sphère de rayon $r + \epsilon$ centrée en **c** et `S1b` une sphère de rayon r centrée en **c**. L'opération `(S1a-S1b)` permet d'obtenir la géométrie du domaine local pour la perforation sphérique S1b. Cela nous donne par exemple la figure 8.6(a).

Grâce à l'opérateur d'union, on peut créer un groupe de domaine local en écrivant : `(S1a-S1b)+(S2a-S2b)+...+(SNa-SNb)`. Nous illustrons cet exemple avec la figure 8.6(b) où l'on a 25 domaines locaux.

2.3 Implémentation d'une géométrie

Pour terminer la description de cet outil géométrique, nous allons présenter l'implémentation d'une forme `GeoTool`. Pour cela, nous allons considérer l'exemple de la forme « `Quadrangle` ». Avant de commencer, il est important de noter que l'implémentation de cette classe est fortement basée sur la librairie Boost.preprocessor [92].

La première étape consiste à définir les caractéristiques de base de la forme. Nous avons le nom (Quadrangle), la dimension topologique (2), le nombre de paramètres (4), le nombre de surfaces (1), le nombre de volumes (0). Ces informations sont ajoutées à la liste des formes déjà disponibles qui est décrite sous forme de macro C++.

Ensuite, nous devons ajouter la fonction suivante pour la description de la forme géométrique :

```
void runQuadrangle( data_geo_ptrtype dg )
{
  // les 2 parametres
  node_type PtA = param<0>( dg ); node_type PtC = param<2>( dg );
  node_type PtB = param<1>( dg ); node_type PtD = param<3>( dg );
  // les 4 sommets
  writePoint( 1, dg , PtA( 0 ), PtA( 1 ) );
  writePoint( 2, dg , PtB( 0 ), PtB( 1 ) );
  writePoint( 3, dg , PtC( 0 ), PtC( 1 ) );
  writePoint( 4, dg , PtD( 0 ), PtD( 1 ) );
  // les 4 lignes
  writeLine( 1, dg , 1 , 2 );
  writeLine( 2, dg , 2 , 3 );
  writeLine( 3, dg , 3 , 4 );
  writeLine( 4, dg , 4 , 1 );
  // la surface
  writeLineLoop( 1, dg, Loop()»1»2»3»4 );
  writePlaneSurface( 1, dg, 1 );
}
```

Ce langage a été conçu en inspirant fortement du langage de géométrie de Gmsh. La variable `dg` est un objet assez complexe qui sauvegarde les informations présentes dans cette fonction. Pour chaque entité géométrique, nous lui associons un index qui est représenté par le premier entier (avant `dg`).

Enfin, il nous reste à définir les marqueurs élémentaires pour les points, lignes, surface et volume à partir des entités géométriques définies dans la fonction précédente. On relie un marqueur élémentaire à une liste d'index identifiant les entités géométriques. Cela s'opère en implémentant plusieurs macros dans un format assez simple. Dans notre exemple, nous devrons implémenter les trois macros suivantes :

- `GEOTOOL_MARKER_POINT_QUADRANGLE`
- `GEOTOOL_MARKER_LINE_QUADRANGLE`
- `GEOTOOL_MARKER_SURFACE_QUADRANGLE`

3 Quelques détails importants du parallélisme

Dans cette partie, nous voulons présenter des aspects du parallélisme qui n'ont pas été décrits dans le chapitre 7. Nous commencerons par la description d'un objet de gestion de processus parallèle. Cet outil permet de communiquer entre des groupes de processus. Nous parlerons ensuite de l'implémentation de l'opérateur d'interpolation en parallèle.

3.1 Classe `WorldComm` : gestion des processus

Dans le développement du parallélisme de FEEL++, un objet très utile a été construit. Il s'agit de la classe `WorldComm`. Elle permet la gestion des communications entre plusieurs processus. Son implémentation repose sur la librairie Boost.mpi [75] qui représente une interface des fonctions du standard MPI. Pour pouvoir faire interagir les processus entre eux, nous avons besoin d'un communicateur. Cet objet gère la connexion d'un ensemble de processus. Il permet ainsi de réaliser des communications dites point-à-point (entre deux processus) ou collective (sur l'ensemble des processus). Les communicateurs seront implémentés à l'aide du type `boost::mpi::communicator`.

Pour représenter une structure de donnée parallèle dans FEEL++, l'objet de type `WorldComm` devra être présent. Il permet de donner une représentation de la structure dans l'environnement parallèle. Cet objet contient trois types de communicateurs :

- `godComm` : un communicateur initial qui englobe tous les processus de l'environnement (MPI_COMM_WORLD). Il sera identique pour tous les objets.
- `globalComm` : un communicateur qui contient un sous-ensemble de processus du communicateur `godComm`.
- `localComm` : un communicateur qui contient un sous-ensemble de processus du communicateur `globalComm`.

Dans le cas standard, où toutes les structures de données sont réparties sur l'ensemble des processus de l'environnement, les trois communicateurs mentionnés précédemment sont identiques. L'objet `WorldComm` ne représente pas un outil indispensable pour cette configuration. Il permet toutefois d'avoir regroupé l'accès aux fonctions MPI dans une classe. Si l'on devait utiliser une autre librairie que Boost.mpi, alors seulement ce fichier devrait être modifié. Cependant, nous avons déjà vu l'utilité de cet outil pour le cas des espaces de fonction produit (II) au chapitre 7. Dans l'exemple présenté, le communicateur `globalComm` restait identique `godComm`. Par contre, ce n'était plus le cas pour le communicateur `localComm`. Il permettait de décrire les processus appartenant à l'un des deux sous-espaces. Selon le processus, ce communicateur contiendra les processus de l'espace vitesse ou les processus de l'espace pression.

Une autre application qui montre la forte utilité de cet objet est la résolution simultanée d'EDP définie sur des domaines différents. Nous pouvons prendre par exemple le cas de la FBM où les problèmes locaux peuvent être résolus de manière totalement indépendante. Chaque sous-domaine peut-être partitionné sur un sous-groupe de processus. Chacun des sous domaines est alors construit et résolu indépendamment des autres. Pour illustrer ce procédé, nous allons considérer la résolution de deux problèmes indépendants avec un environnement parallèle composé de cinq processus. La première étape consiste à définir les groupes de processus pour chacun des deux sous-domaines. Nous les distribuons

aléatoirement en représentant les groupes par des couleurs (entiers 3 et 5 par ex). Cette distribution s'exprime par le code suivant :

```
// definition des couleur
int color1=3; int color2=5;
std::vector<int> MapWorld(5);
// attribution des couleurs
MapWorld[0]=color2;MapWorld[1]=color1;MapWorld[2]=color2;
MapWorld[3]=color2;MapWorld[4]=color1;
// definition du WorldComm avec couleur
auto worldCommColored = WorldComm(MapWorld);
```

L'objet `worldCommColored` contient un communicateur `globalComm` de taille 5 pour tous les processus et un communicateur `localComm` de taille 2 ou 3 selon le processus.Nous voulons maintenant obtenir les objets `WorldComm` associés à chaque couleur. Cette opération est faite par le code suivant :

```
// definition de chaque WorldComm (par couleur)
auto worldCommD1 = worldCommColored.subWorldComm(color1);
auto worldCommD2 = worldCommColored.subWorldComm(color2);
```

Pour la couleur `color1`, nous obtenons l'objet `worldCommD1` qui contient un communicateur `localComm` de taille 2 ou 3 identique à `globalComm`. Il est de taille 2 pour les processus de rang 1 et 4 avec un état d'activité pour ces processus. Il est de taille 3 pour les autres processus avec des états de non-activité.

Une fois ces objets créés, nous devons donner ces informations à deux structures de données qui sont les maillages et les backends. Nous illustrons ces déclarations pour un des deux domaines avec le code ci-dessous :

```
auto mesh1 = R1.createMesh(_mesh=new mesh_type,_name ="domain1",
                           _partitions=worldCommD1.localSize(),
                           _worldcomm=worldCommD1 );
auto backend1 = Backend<double>::build(worldCommD1);
```

C'est exactement la même chose pour l'autre domaine. Il n'y a pas de condition du style « si je suis le rang x ... ». Tous les processeurs appellent ces fonctions, mais grâce à la notion d'activité, leur renvoi de la fonction est quasi immédiat.

Une fois ces structures initialisées, les informations de l'environnement parallèle vont être transmises automatiquement aux autres structures de données telles que les espaces de fonction, matrices, vecteurs, ... Également, aucune condition sur les rangs n'apparaitra au niveau de l'assemblage, la résolution, les exports. Ces concepts permettent de manière assez transparente de résoudre deux problèmes simultanément et chaque problème est résolu en parallèle avec PETSc. Cet outil a été utilisé dans [142] pour l'implémentation de méthodes de décomposition de domaine avec Feel++.

3.2 Opérateur d'interpolation en parallèle

Un des outils indispensables pour notre méthode d'interaction fluide structure est l'opérateur d'interpolation. Il a donc été nécessaire de paralléliser cet opérateur pour pouvoir réaliser des modélisations FSI de taille assez importante. Le but de cette partie est d'expliquer brièvement la méthode utilisée pour construire la matrice représentant cet opérateur.

Nous avons deux domaines de calcul qui sont chacun répartis sur un ensemble de processus. Chaque sous-domaine est donc partitionné et il n'y a aucune dépendance vis-à-vis

de ces parutions. L'objectif est d'interpoler une fonction définie sur un des domaines vers l'autre domaine. Nous utiliserons la description qui a été faite dans le chapitre 1 pour l'interpolation des fonctions éléments finis. Nous voulons décrire dans cette partie la technique de localisation dans l'environnement parallèle. Elle est représentée par l'algorithme 11. Une étape importante pour la performance de cet algorithme est l'étape de prédiction. Pour un point donné, elle doit permettre d'identifier la partition qui contient ce point avec une probabilité assez bonne. Notre méthode assez simple utilise les barycentres des partitions. On estime cette probabilité par la distance du point au barycentre de la partition. On utilisera ainsi la partition qui aura son barycentre le plus proche.

Algorithm 11 Opérateur d'interpolation parallèle

Require: ListPts : une liste de point à localiser
  while ListPts n'est pas vide **do**
    Prédiction d'une partition non visité pour chaque point
    Envoie des points à localiser aux processeurs correspondants
    Reception des points, tentative de localisation et renvoie d'une réponse
    Reception de la réponse et mise à jour de ListPts
  end while

Une autre partie importante est la construction de la matrice parallèle associée à l'opérateur d'interpolation. Lorsque les deux domaines sont repartis sur les mêmes processus (et donc avec le même le nombre de partitions), cette étape est relativement facile et a été implémentée. La matrice construite utilise la table de degré de liberté parallèle des deux espaces de fonction. Le produit matrice vecteur a également un« sens » avec cette répartition. Si la répartition des processus sur les domaines n'est plus identique, alors cette tâche devient très dure. Nous pouvons distinguer deux cas pour cette configuration. Le premier correspond au fait que les processus se « superposent ». Nous avons par exemple les rangs 0 et 1 pour un des deux domaines et le rang 0,1,2,3,4 pour l'autre. Le dernier cas, très complexe à traiter, est lorsque les processus sont disjoints. Nous avons par exemple les rangs 0 et 1 pour un des deux domaines et le rang 2,3,4 pour l'autre Le développement de ces deux types de répartition est en cours de développement.

4 Environnement pour l'interaction fluide structure

Nous finissons ce chapitre par une présentation de l'environnement informatique qui a été développé pour la simulation de l'interaction fluide structure. Nous commencerons par une description des fonctionnalités disponibles pour la modélisation de la mécanique des fluides et des solides. Nous parlerons ensuite de l'utilisation de cet environnement.

4.1 Modèles et méthodes numériques disponibles

La mécanique des fluides

Dans la catégorie des modèles incompressibles newtoniens, nous trouverons : Stokes, Oseen et Navier-Stokes. Un modèle non newtonien a aussi été implémenté, mais il a été peu expérimenté dans la pratique. Il s'agit d'un modèle basé sur une loi puissance. Tous ces modèles peuvent être aussi utilisés dans le contexte de domaine mobile. Ils sont connectés à

notre solveur ALE d'ordre élevé qui se base sur l'extension harmonique ou sur les équations de Winslow. Au niveau de la discrétisation numérique, nous utilisons les éléments finis de Lagrange pour l'approximation vitesse et pression. Le choix des ordres polynomiaux est totalement arbitraire. On utilise généralement les éléments de Taylor-Hood pour leur stabilité intrinsèque. Les maillages d'ordre arbitraire ($\leq$ 5 en 2D et $\leq$ 4 en 3D) sont supportés. Pour les schémas de discrétisation en temps, nous pouvons utiliser les schémas BDF_q avec $q \in \{1, 2, 3, 4\}$. Plusieurs méthodes de stabilisation basées sur la méthode CIP sont disponibles : cip-convection, cip-divergence, cip-pressure. Il existe aussi une méthode de stabilisation de la divergence par pénalisation. Ce solveur de mécanique des fluides permet également de réaliser des simulations bi-fluides. Les auteurs de [45] ont utilisé ce solveur pour ces simulations basées sur la méthode level set. Pour les conditions aux limites, nous trouverons dans notre langage les suivantes : `dirichlet_vec, neumann_scal, pressure, paroi_mobile, slip, bloodflow_outlet`.

La mécanique des solides

Nous commençons par citer la présence du modèle élastique linéaire basé sur la loi de Hooke. Ensuite, nous avons les modèles hyperélastiques que celles regroupés en deux catégories : les modèles compressibles et incompressibles. Pour les modèles compressibles, nous avons les lois de comportement de Saint-Venant-Kirchhoff et Néo-Hookéen. Pour les modèles incompressibles, nous trouverons les lois de comportement Saint-Venant-Kirchhoff, Néo-Hookéen et Mooney-Rivlin. Un modèle réduit a aussi été implémenté. C'est un modèle 1D nommé generalized string. Pour les approximations polynomiales et la discrétisation géométrique, nous avons les mêmes caractéristiques qui sont mentionnées pour le fluide. Au niveau de la discrétisation en temps, nous utilisons un schéma de Newmark. Les conditions aux limites disponibles sont de types : `dirichlet_vec, neumann_scal, neumann_vec, paroi_mobile, robin_vec`.

L'interaction fluide structure

Le couplage entre le solveur fluide et structure est réalisé par une méthode partitionnée avec des schémas implicites et semi-implicites. Nous proposons plusieurs outils pour faciliter l'implémentation de ce modèle. Nous avons notamment un outil de gestion des transferts d'information sur l'interface fluide structure. Nous trouvons aussi un outil dédié à la relaxation dynamique d'Aitken et à la convergence du point fixe. Tous les modèles décrits précédemment sont compatibles.

La méthode FBM

Toute la description faite dans cette thèse sur cette méthode a été implémentée dans cet environnement. Nous pouvons modéliser les écoulements de fluide autour d'obstacles fixes (chapitre 3). L'interaction fluide-particule est également possible (chapitre 6). Pour toutes ces applications, nous pouvons utiliser l'ensemble des modèles fluide et structure décrits précédemment. Nous avons aussi deux types de couplage pour l'interaction fluide particule : implicite et semi-implicite. Seule la construction de ce modèle FBM n'a pas encore été parallèlisée.

Les méthodes de résolution algébrique

Pour la résolution des systèmes linéaires, nous pouvons utiliser toute la gamme de méthode de Krylov préconditionnée disponible dans la librairie PETSc. Pour les équations non linéaires, nous utilisons également les outils proposés par cette libraire. Nous utilisons ainsi des méthodes de Newton ou quasi-Newton pour la résolution. Plusieurs optimisations sont disponibles dans notre environnement. Nous pouvons stocker les termes constants des matrices et vecteurs et les réutiliser le plus longtemps possible. La réutilisation du préconditionneur est une option qui s'avère souvent très efficace. Dans l'algorithme de Newton, nous pouvons également optimiser la construction du vecteur résidu en utilisant la partie linéaire de la matrice jacobienne (application d'un produit matrice vecteur). De plus, tous nos solveurs sont parallèles.

4.2 Utilisation de l'environnement

Pour pouvoir réaliser plusieurs types de simulations, avec par exemple des conditions aux limites différentes, il est nécessaire d'avoir plusieurs programmes. En effet, les expressions du langage Feel++ peuvent correspondre à un type C++ très complexe et nécessitent d'être exprimé à la compilation. Il était impensable de recopier les codes à la main en modifiant à chaque fois ces conditions. C'est pourquoi un travail important a été fait pour aider l'utilisateur dans la création de nouvelles applications en se basant sur un code unique décrivant les modèles présentés dans la partie précédente. Pour illustrer les principales étapes de construction d'une application, nous allons prendre un exemple assez simple de mécanique des fluides.

Pour définir son application, l'utilisateur doit d'abord commencer par définir les caractéristiques de l'application. Ces informations sont décrites sous la forme de macro c++ et elles se trouvent dans un fichier que l'on nomme par exemple `monappli.bc`. Il contient le code source suivant :

```
#define FLUIDMECHANICS_DIM 2
#define FLUIDMECHANICS_ORDERGEO 1
#define FLUIDMECHANICS_ORDER_VELOCITY 2
#define FLUIDMECHANICS_ORDER_PRESSURE 1

#define FLUIDMECHANICS_BC(OBJECT_PTR)                                  \
  addCL<cl::dirichlet_vec,cl::Entree>(INPUTVELOCITY(OBJECT_PTR)) \
  +                                                                    \
  addCL<cl::neumann_scal,cl::Sortie>(vf::cst(0.))                      \
  +                                                                    \
  addCL<cl::dirichlet_vec,cl::Paroi>(0*vf::one())                      \
  /**/
#define FLUIDMECHANICS_VOLUME_FORCE(OBJECT_PTR) \
  -OBJECT_PTR->rho()*vf::cst(9.81)*vf::oneY()    \
  /**/
```

Nous trouverons tout d'abord la définition de la dimension, de l'ordre géométrique et des degrés d'approximation de la vitesse et pression. Nous avons ensuite la description des conditions aux limites avec la macro `FLUIDMECHANICS_BC(OBJECT_PTR)`. L'argument `OBJECT_PTR` est un pointeur sur l'objet décrivant le modèle de mécanique des fluides. Il permet ainsi d'exprimer les conditions aux limites selon certains paramètres (physiques ou donnés par l'utilisateur). Dans cet exemple, nous avons trois types de conditions qui permettent de représenter un écoulement : un profil de vitesse donné en entrée,

une paroi de non-glissement et une sortie libre de contraintes. Pour chaque condition, nous utilisons la fonction `addCL` qui est décrite par trois arguments. Les deux premiers (arguments « template ») correspondent respectivement aux types de conditions aux limites (ex : `dirichlet_vec`) et au marqueur physique sur lequel agit cette condition (ex : `Entree`). Le troisième argument correspond à l'expression de la condition aux limites (ex : `INPUTVELOCITY(OBJECT_PTR)` doit représenter une expression vectorielle pour le profil de vitesse).

Ensuite, nous devons implémenter la fonction principale qui permet d'initialiser et d'exécuter l'application. Nous présentons le fichier que l'on nomme `main.cpp`. Il contient cette fonction représentée par le code suivant :

```
int main( int argc, char** argv )
{
  // objet qui manage tous les types de modeles
  AppliManagement Master(argc,argv);
  // creation du modele de fluide
  auto FM = Master.FluidMechanics();
  // initialisation (structures algebriques)
  FM->init();
  // boucle en temps
  for (double time = ti+dt ; time <= tf ; time += dt)
    {
      FM->updateTime(time);
      FM->solve();
      FM->exportResults();
      FM->updateBdf();
    }
}
```

Il contient les routines de bases nécessaires pour effectuer notre simulation fluide. Avec les quelques commentaires et le nom des fonctions, le rôle de chacune d'elles semble assez clair. C'est aussi dans cette partie que l'utilisateur peut effectuées des opérations de post-traitement (ex : mesure de quantités physiques). La géométrie aurait également pu être décrite dans cette partie. Cependant, d'autres alternatives plus simples existent. Nous y reviendrons à la fin de cette partie.

Enfin, pour pouvoir créer le programme, nous allons utiliser l'outil CMake [112, 113] qui va nous générer les `makefiles` nécessaires pour la compilation. La création de ces fichiers s'effectue pour le cas d'un modèle fluide par l'ajout d'une ligne dans un `CMakeLists.txt` : `fluidAppli( monprog monappli.bc main.cpp)`. Cette macro CMake que nous avons développée permet de recopier les fichiers de la librairie du modèle fluide, le fichier de conditions aux limites et le programme principal. La compilation générera ensuite un exécutable nommé `monprog`. Nous ne décrivons pas les détails des opérations complexes effectués à cette étape à l'aide de CMake. Nous pouvons donner un argument supplémentaire à cette fonction. Il correspond au nom d'un fichier contenant la description d'une géométrie. Celle-ci peut aussi être donnée directement dans un fichier de configuration (.cfg) contenant les options de ligne de commande. C'est dans ce fichier que l'on décrit entièrement le modèle physique, les paramètres, les préconditionneurs, le pas de temps, le type d'exportation, ...

Pour les autres modèles (structure, FSI, FBM), on utilise le même type de procédure pour construire les applications. Ce procédé nous a permis de créer de nombreuses applications très diverses.

Quatrième partie

Simulations numériques

Chapitre 9

Benchmarks CFD [1], CSM [2] et FSI [3]

Pour valider l'implémentation de notre solveur d'interaction fluide-structure, nous allons dans ce chapitre essayer de reproduire des simulations numériques qui ont été réalisées par d'autres codes de calcul numérique. Le benchmark que nous avons testé a été proposé initialement par [154]. Il modélise l'écoulement d'un fluide incompressible dans un régime laminaire. Cet écoulement contient un obstacle qui est en partie déformable. Ce matériau pourra subir des déformations relativement importantes. Il est considéré comme hyper-élastique et compressible. Nous pouvons illustrer cette simulation avec la figure 9.1. Elle montre le champ de vitesse du fluide et le déplacement de l'obstacle à un instant $t = 6s$. Ce benchmark est devenu un des benchmarks classiques pour l'interaction fluide-structure.

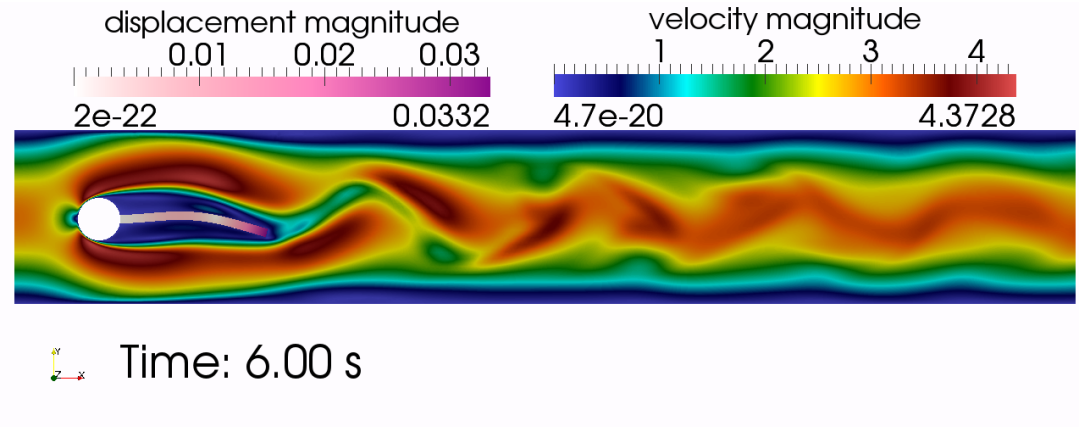


FIGURE 9.1 – Champ de vitesse obtenu avec le benchmark FSI.

Dans la référence [154], nous pouvons trouver trois configurations pour le benchmark FSI. Elles sont notées **FSI1**, **FSI2** et **FSI3**. La première configuration représente le calcul d'un état stationnaire. Les deux autres configurations transitoires sont distinguées par leur nombre de Reynolds (Re). On a $Re = 100$ pour **FSI2** et $Re = 200$ pour **FSI3**. Dans ce chapitre, uniquement le cas test **FSI3** a été expérimenté.

Pour vérifier les résultats numériques, les auteurs de ce benchmark proposent de calculer des mesures caractérisant des propriétés physiques de la simulation. Nous avons

1. Computational Fluid Dynamics
2. Computational Solid Mechanics
3. Fluid Structure Interaction

l'évaluation du déplacement d'un point qui appartient à la partie élastique de l'obstacle. Nous mesurerons aussi les forces hydrodynamiques exercées par le fluide sur l'obstacle. Pour chacune de ces mesures, nous trouverons les résultats sous la forme :

$$\text{moyenne} \pm \text{amplitude}\,[\text{fréquence}]$$

Dans [154], nous trouverons également des applications pour la mécanique des fluides (CFD) et pour la mécanique du solide (CSM). Ces exemples utilisent les mêmes géométries que pour le cas FSI. Ils permettent dans un premier temps de vérifier séparément les solveurs fluides et structures. Ainsi, nous commencerons ce chapitre par l'expérimentation des exemples CSM et CFD. Puis, nous passerons à l'application FSI. Les résultats obtenus seront comparés à plusieurs codes de calcul.

1 Benchmark CSM

Cette première application modélise la déformation d'une barre hyperélastique soumise à son propre poids. Cette barre est fixée à l'une des extrémités et l'action de la force de gravité entraine un mouvement oscillatoire de cette structure. Elle est nommée par **CSM3** dans [154]. Nous avons la description de la géométrie de la structure Ω_s^* avec la figure 9.2. Nous avons le bord fixe noté par Γ_F^*. Les autres bords sont notés $\Gamma_L^* = \partial\Omega_s^* \backslash \Gamma_F^*$ Les paramètres géométriques sont donnés par :

- $\mathbf{A} = (0.6, 0.2)$
- $\mathbf{B} = ((0.4 + \sqrt{0.0096})/2, 0.19)$
- $\mathbf{C} = (0.2, 0.2)$
- $(l, h, r) = (0.35, 0.2, 0.05)$

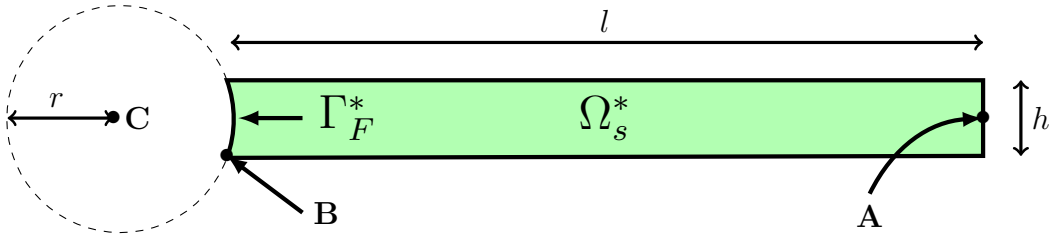


FIGURE 9.2 – Géométrie du benchmark CSM.

Le matériau est supposé hyperélastique avec une loi de comportement Saint-Venant-Kirchhoff compressible. Il est caractérisé par son module de Young $E_s = 1.4 \times 10^6\ kg/ms^2$ et son coefficient de Poisson $\nu_s = 0.4$. La masse volumique ρ_s^* est prise égale à 1000 kg/m^3. Pour les conditions aux limites, nous avons :

- $\boldsymbol{\eta}_s = 0$ sur Γ_F^*
- $(\boldsymbol{F}_s \boldsymbol{\Sigma}_s)\, \boldsymbol{n}_s^* = \mathbf{0}$ sur Γ_L^*

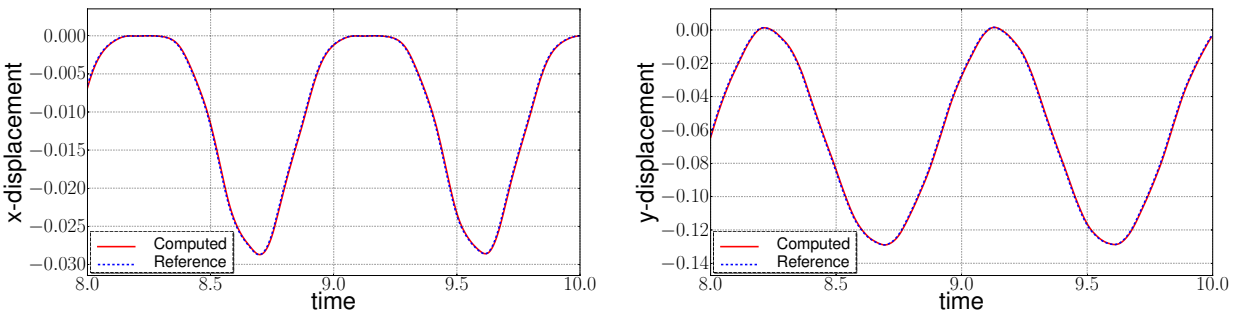
Enfin, le terme source nécessaire pour mettre en mouvement la barre élastique est choisi par $\boldsymbol{f}_s = (0, -\rho_s^* g)$. La constante $g = 2\ m/s^2$ représente une constante de gravité.

Pour vérifier le bon comportement de notre solveur de mécanique des solides, nous mesurons le déplacement du point **A** dans la direction x et y. Cependant, la description du benchmark ne précise pas l'intervalle de temps pour lesquelles ses mesures sont effectuées.

Nous avons choisi d'effectuer cette étude sur l'intervalle de temps $[0, 10]$. Nos résultats sont donnés dans la table 9.1. Plusieurs raffinements de maillage et plusieurs approximations polynomiales pour le déplacement ont été utilisés. Nos résultats sont en assez bon accord avec les valeurs de référence obtenues par [154]. Nous avons toutefois un léger décalage dans nos résultats. Le changement d'intervalle d'étude semble en être la cause. Cette information paraît nécessaire pour obtenir des résultats plus cohérents. Les figures 9.3(a) et 9.3(b) nous montrent aussi que les courbes de déplacement en x et y que nous avons mesurées avec la dernière ligne de la table 9.1 se superposent très bien avec les résultats de [154].

N_{elt}	N_{dof}	x $[\times 10^{-3}]$	y $[\times 10^{-3}]$
[154]		-14.305 ± 14.305 [1.0995]	-63.607 ± 65.160 [1.0995]
4199	17536(P_2)	-14.585 ± 14.590 [1.0953]	-63.981 ± 65.521 [1.0930]
4199	38900(P_3)	-14.589 ± 14.594 [1.0953]	-63.998 ± 65.522 [1.0930]
1043	17422(P_4)	-14.591 ± 14.596 [1.0953]	-64.009 ± 65.521 [1.0930]
4199	68662(P_4)	-14.590 ± 14.595 [1.0953]	-64.003 ± 65.522 [1.0930]
4199	17536(P_2)	-14.636 ± 14.640 [1.0969]	-63.937 ± 65.761 [1.0945]
4199	38900(P_3)	-14.642 ± 14.646 [1.0969]	-63.949 ± 65.771 [1.0945]
1043	17422(P_4)	-14.645 ± 14.649 [1.0961]	-63.955 ± 65.778 [1.0945]
4199	68662(P_4)	-14.627 ± 14.629 [1.0947]	-63.916 ± 65.739 [1.0947]
4199	17536(P_2)	-14.645 ± 14.645 [1.0966]	-64.083 ± 65.623 [1.0951]
4199	38900(P_3)	-14.649 ± 14.650 [1.0966]	-64.092 ± 65.637 [1.0951]
1043	17422(P_4)	-14.652 ± 14.653 [1.0966]	-64.099 ± 65.645 [1.0951]
4199	68662(P_4)	-14.650 ± 14.651 [1.0966]	-64.095 ± 65.640 [1.0951]

TABLE 9.1 – Résultats obtenus pour **CSM3** avec $\Delta t = 0.02, 0.01, 0.005$. La première ligne montre les valeurs de référence obtenues dans [154].



(a) Déplacement en x

(b) Déplacement en y

FIGURE 9.3 – Déplacement calculé du point **A** pour **CSM3** comparé à la référence [154].

2 Benchmark CFD

Nous passons maintenant au benchmark de mécanique des fluides nommé **CFD3** dans [154]. Cette application modélise l'écoulement d'un fluide incompressible autour d'un obstacle rigide. Cet obstacle est identique à la géométrie de l'application **CSM3**. La géométrie du fluide est représentée avec la figure 9.4. Le domaine fluide Ω_f est délimité par le rectangle $[0, 2.5] \times [0, 0.41]$.

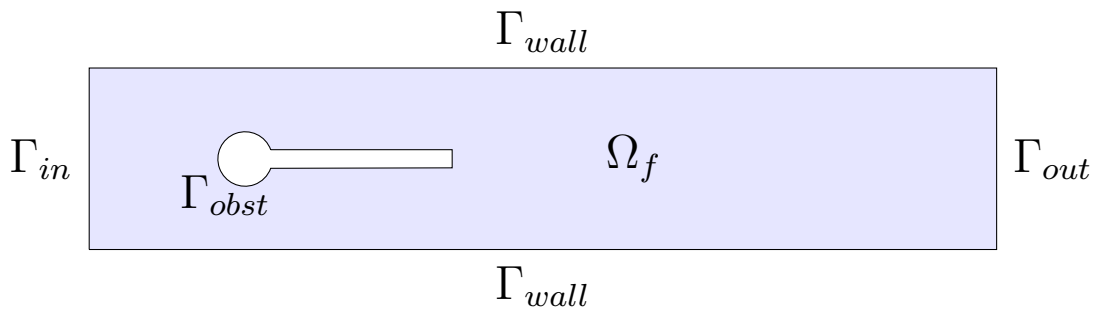


FIGURE 9.4 – Géométrie du benchmark CFD.

Le modèle utilisé pour décrire l'écoulement fluide est le modèle de Navier-Stokes incompressible. Il est caractérisé par une viscosité dynamique $\mu_f = 1\ kg/ms$ et une masse volumique $\rho_f = 1000\ kg/m^3$. Nous n'avons pas de terme source pour ce benchmark, c'est-à-dire $\boldsymbol{f}_f \equiv \mathbf{0}$. Pour décrire l'entrée de l'écoulement sur la frontière Γ_{in}, nous imposons un profil de vitesse parabolique défini par :

$$v_{cst} = 1.5\bar{U}\frac{4}{0.1681}y\,(0.41 - y) \tag{9.1}$$

avec $\bar{U}$ le débit moyen que l'on choisit égal à 2. Toutefois, ce profil de vitesse est imposé progressivement. Nous définissons alors :

$$v_{in} = \begin{cases} v_{cst}\dfrac{1 - \cos\left(\frac{\pi}{2}t\right)}{2} & \text{si } t < 2 \\ v_{cst} & \text{sinon} \end{cases} \tag{9.2}$$

À l'aide de ces données, nous pouvons ensuite exprimer les conditions aux limites du benchmark **CFD3** :

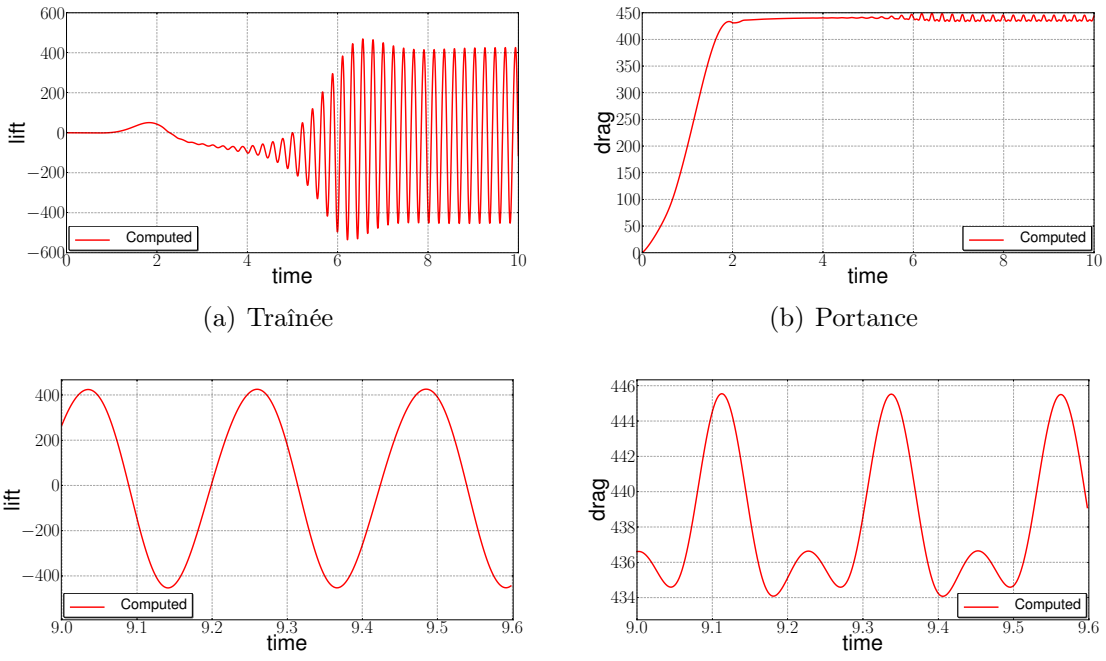
- $\boldsymbol{u}_f = (v_{in}, 0)$ sur Γ_{in}
- $\boldsymbol{u}_f = \mathbf{0}$ sur Γ_{wall}
- $\boldsymbol{u}_f = \mathbf{0}$ sur Γ_{obst}
- $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0}$ sur Γ_{out}

Les mesures effectuées pour contrôler le comportement du solveur fluide sont les forces de traînée F_D et les forces de portance F_L. Ces forces sont calculées autour de l'obstacle par la formule suivante :

$$(F_D, F_L) = \int_{\Gamma_{obst}} \boldsymbol{\sigma}_f \boldsymbol{n}_f \tag{9.3}$$

N_{geo}	N_{elt}	N_{dof}	N_{bdf}	Traînée	Portance
[154]				$439.45 \pm 5.6183\,[4.3956]$	$-11.893 \pm 437.81\,[4.3956]$
1	8042	$37514(P_2/P_1)$	2	$437.47 \pm 5.3750\,[4.3457]$	$-9.786 \pm 437.54\,[4.3457]$
2	2334	$26706(P_3/P_2)$	2	$439.27 \pm 5.1620\,[4.3457]$	$-8.887 \pm 429.06\,[4.3457]$
2	7970	$89790(P_3/P_2)$	2	$439.56 \pm 5.2335\,[4.3457]$	$-11.719 \pm 425.81\,[4.3457]$
1	3509	$39843(P_3/P_2)$	2	$438.24 \pm 5.5375\,[4.3945]$	$-11.024 \pm 433.90\,[4.3945]$
1	8042	$90582(P_3/P_2)$	2	$439.25 \pm 5.6130\,[4.3945]$	$-10.988 \pm 437.70\,[4.3945]$
2	2334	$26706(P_3/P_2)$	2	$439.49 \pm 5.5985\,[4.3945]$	$-10.534 \pm 441.02\,[4.3945]$
2	7970	$89790(P_3/P_2)$	2	$439.71 \pm 5.6410\,[4.3945]$	$-11.375 \pm 438.37\,[4.3945]$
3	3499	$73440(P_4/P_3)$	3	$439.93 \pm 5.8072\,[4.4921]$	$-14.511 \pm 440.96\,[4.3945]$
4	2314	$78168(P_5/P_4)$	2	$439.66 \pm 5.6412\,[4.3945]$	$-11.329 \pm 438.93\,[4.3945]$
2	7942	$89482(P_3/P_2)$	2	$439.81 \pm 5.7370\,[4.3945]$	$-13.730 \pm 439.30\,[4.3945]$
3	2340	$49389(P_4/P_3)$	2	$440.03 \pm 5.7321\,[4.3945]$	$-13.250 \pm 439.64\,[4.3945]$
3	2334	$49266(P_4/P_3)$	3	$440.06 \pm 5.7773\,[4.3945]$	$-14.092 \pm 440.07\,[4.3945]$

TABLE 9.2 – Résultats obtenus pour **CFD3** avec $\Delta t = 0.01, 0.005, 0.002$. La première ligne montre les valeurs de référence obtenues dans [154] pour $\Delta t = 0.005$.



(a) Traînée (b) Portance

(c) Traînée (d) Portance

FIGURE 9.5 – Resultats du benchmark **CFD3**.

Les résultats que nous avons mesurés sont affichés dans la table 2. Ils contiennent plusieurs types de configurations qui se distinguent avec l'ordre géométrique (N_{geo}), les approximations polynomiales vitesse/pression (P_M/P_N), l'ordre des schémas BDF (N_{bdf}) et le nombre d'éléments du maillage (N_{elt}). Nos résultats utilisant BDF_2 et $\Delta t = 0.005$ sont en très bon accord avec ceux présentés dans [154]. Ces derniers ont également été

calculés avec $\Delta t = 0.005$. Cependant, nous avons effectué des calculs supplémentaires en utilisant un pas de temps plus petit, $\Delta t = 0.002$. Nous observons alors un décalage significatif des valeurs mesurées, en particulier pour la traînée. On remarque que ce décalage était présent avec BDF_3 et $\Delta t = 0.005$. Cela semble signifier que le schéma BDF_2 avec $\Delta t = 0.005$ n'est pas assez précis. Nous pouvons aussi voir l'intérêt de l'ordre géométrique qui permet d'obtenir de très bons résultats sans avoir à beaucoup raffiner le maillage. Enfin, nous avons illustré les courbes des forces de traînée et de portance avec la figure 9.5.

3 Benchmark FSI

Enfin, nous présentons le benchmark d'interaction fluide structure proposée dans [154] et nommé par **FSI3**. Cette application modélise l'écoulement d'un fluide autour d'un obstacle rigide circulaire et d'une barre élastique accrochée à ce solide rigide. La figure 9.6 illustre cette géométrie qui se base sur les géométries de benchmarks précédents. Elle représente les états initiaux des domaines fluides et structures.

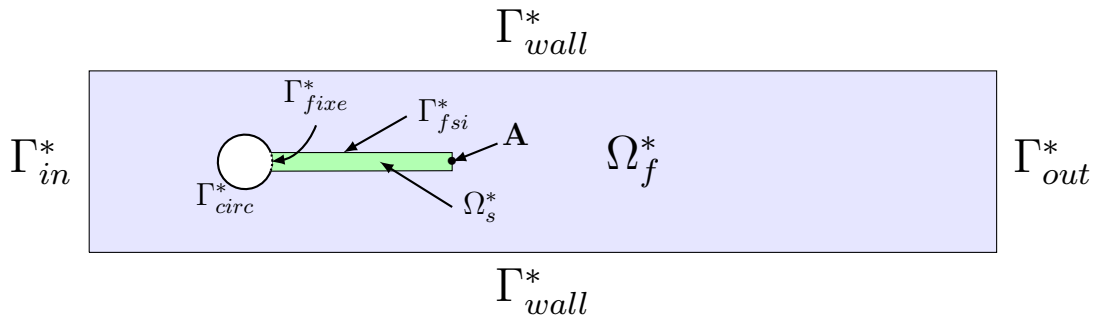


FIGURE 9.6 – Géométrie du benchmark FSI.

Le modèle utilisé pour décrire l'écoulement fluide est toujours le modèle de Navier-Stokes incompressible. Il est caractérisé par une viscosité dynamique $\mu_f = 1\ kg/(m \cdot s)$ et une masse volumique $\rho_f = 1000\ kg/m^3$. La barre élastique est considérée comme un matériau hyperélastique avec une loi de comportement Saint-Venant-Kirchhoff compressible. Il est caractérisé par son module de Young $E_s = 5.6 \times 10^6\ kg/ms^2$, son coefficient de Poisson $\nu_s = 0.4$ et sa masse volumique $\rho^*_s = 1000\ kg/m^3$. Pour les conditions aux limites, nous considérons :

- $\boldsymbol{u}_f = (v_{in}, 0)$ sur Γ^*_{in}
- $\boldsymbol{u}_f = \boldsymbol{0}$ sur Γ^*_{wall}
- $\boldsymbol{u}_f = \boldsymbol{0}$ sur Γ^*_{circ}
- $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \boldsymbol{0}$ sur Γ^*_{out}
- $\boldsymbol{\eta}_s = \boldsymbol{0}$ sur Γ^*_{fixe}

avec v_{in} le même profil parabolique qui a été décrit dans le benchmark **CFD3**. Pour le couplage fluide structure sur Γ^*_{fsi}, nous utilisons les mêmes conditions standards que nous avons présentées au chapitre 6. Aussi, nous n'avons pas de terme source présent pour ce benchmark.

Pour vérifier nos simulations, nous calculons la force de portance et de traînée sur l'obstacle circulaire et élastique (i.e sur $\Gamma^*_{circ} \cup \Gamma^*_{fsi}$). Nous mesurons aussi le déplacement du point **A**. Plusieurs configurations ont été utilisées. Elles sont énumérées dans la table 9.3 avec aussi quelques informations sur l'un des résultats de référence. De plus, chacune de nos configurations utilisent un schéma BDF d'ordre 2 pour le fluide. Pour le solveur fluide structure, nous utilisons un schéma implicite avec une tolérance pour l'algorithme de point de 10^{-6} (voir chapitre 6).

	N_{elt}	N_{dof}	$[P_c^N(\Omega_{f,\delta})]^2 \times P_c^{N-1}(\Omega_{f,\delta}) \times V_{s,\delta}^N$	Δt
[155]	15872	304128		0.00025
(1)	1284	27400	$[P_c^4(\Omega_{f,(h,3)})]^2 \times P_c^3(\Omega_{f,(h,3)}) \times V_{s,(h,3)}^3$	0.005
(2)	2117	44834	$[P_c^4(\Omega_{f,(h,3)})]^2 \times P_c^3(\Omega_{f,(h,3)}) \times V_{s,(h,3)}^3$	0.005
(3)	4549	95427	$[P_c^4(\Omega_{f,(h,3)})]^2 \times P_c^3(\Omega_{f,(h,3)}) \times V_{s,(h,3)}^3$	0.005
(4)	17702	81654	$[P_c^2(\Omega_{f,(h,1)})]^2 \times P_c^1(\Omega_{f,(h,1)}) \times V_{s,(h,1)}^1$	0.0005

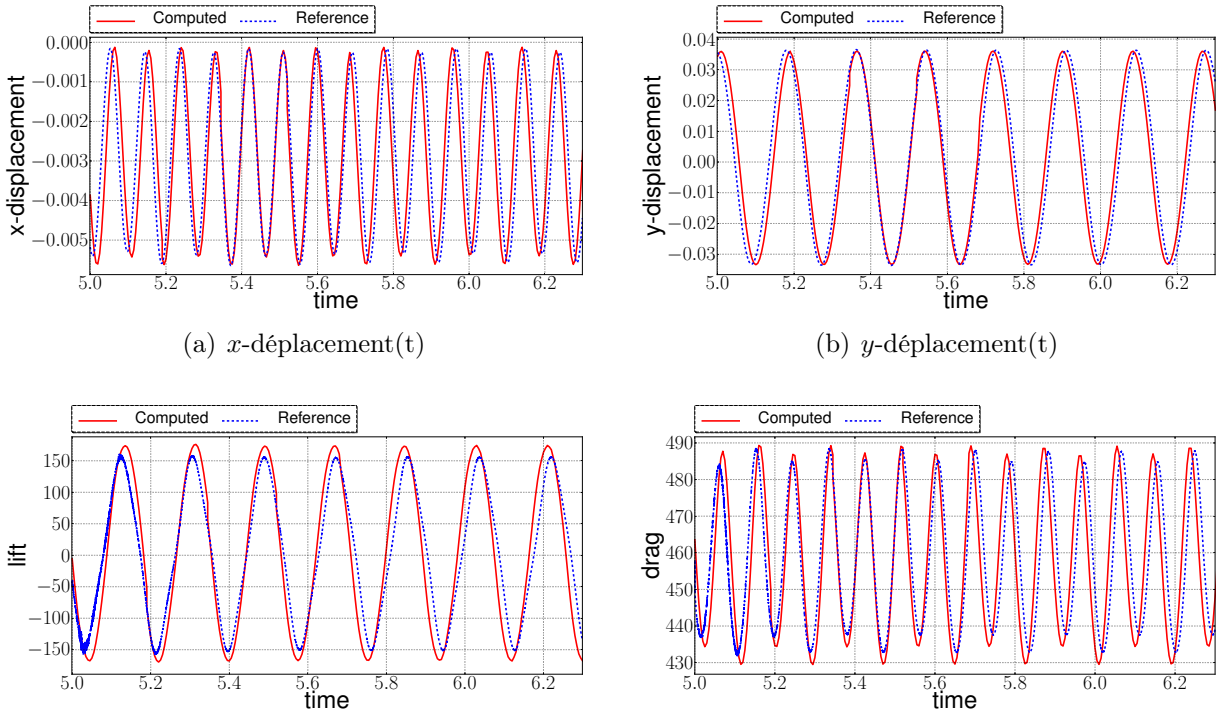
TABLE 9.3 – Configurations utilisées pour le benchmark **FSI3**.

Les résultats calculés par nos simulations se trouvent dans la table 9.4. Ils sont comparées à plusieurs autres codes dont les résultats sont fournis dans [154, 30, 155, 122, 64, 143]. Avec la figure 9.7, nous avons également tracé les quantités du benchmark mesurées avec la configuration **(3)** sur un petit intervalle de temps. Nous trouvons aussi les résultats obtenus avec [155].

	x $[\times 10^{-3}]$	y $[\times 10^{-3}]$	Traînée	Portance
[154]	-2.69 ± 2.53 [10.9]	1.48 ± 34.38 [5.3]	457.3 ± 22.66 [10.9]	2.22 ± 149.78 [5.3]
[30]			464.5 ± 40.50	6.00 ± 166.00 [5.5]
[155]	-2.88 ± 2.72 [10.9]	1.47 ± 34.99 [5.5]	460.5 ± 27.74 [10.9]	2.50 ± 153.91 [5.5]
[122]	-4.54 ± 4.34 [10.1]	1.50 ± 42.50 [5.1]	467.5 ± 39.50 [10.1]	16.2 ± 188.70 [5.1]
[64]			474.9 ± 28.10	3.90 ± 165.90 [5.5]
[143]	-2.83 ± 2.78 [10.8]	1.35 ± 34.75 [5.4]	458.5 ± 24.00 [10.8]	2.50 ± 147.50 [5.4]
(1)	-2.86 ± 2.74 [10.9]	1.31 ± 34.71 [5.4]	459.7 ± 29.97 [10.9]	4.46 ± 172.53 [5.4]
(2)	-2.85 ± 2.72 [10.9]	1.35 ± 34.62 [5.4]	459.2 ± 29.62 [10.9]	3.53 ± 172.73 [5.4]
(3)	-2.88 ± 2.75 [10.9]	1.35 ± 34.72 [5.4]	459.3 ± 29.84 [10.9]	3.19 ± 171.20 [5.4]
(4)	-2.90 ± 2.77 [11.0]	1.33 ± 34.90 [5.5]	457.9 ± 31.79 [11.0]	8.93 ± 216.21 [5.5]

TABLE 9.4 – Résultats pour **FSI3**

Les configurations **(1)**, **(2)** et **(3)** donnent des résultats qui sont en assez bon accord avec les valeurs de référence. Ces résultats soulignent l'intérêt de l'approximation d'ordre élevé en espace et géométrie. Cette méthode permet de réduire le nombre de degrés de liberté en gardant des résultats assez précis (voir par exemple la configuration [155], table 9.3). Pour illustrer cette propriété, nous montrons avec la figure 9.9 le maillage d'ordre 3 utilisé dans la configuration **(1)**. On remarque cependant que la portance n'est pas en parfait accord avec les autres valeurs de référence. Des perturbations apparaissent dans cette quantité. Elles sont encore plus importantes lorsque l'on baisse le pas de temps. On



(a) x-déplacement(t)

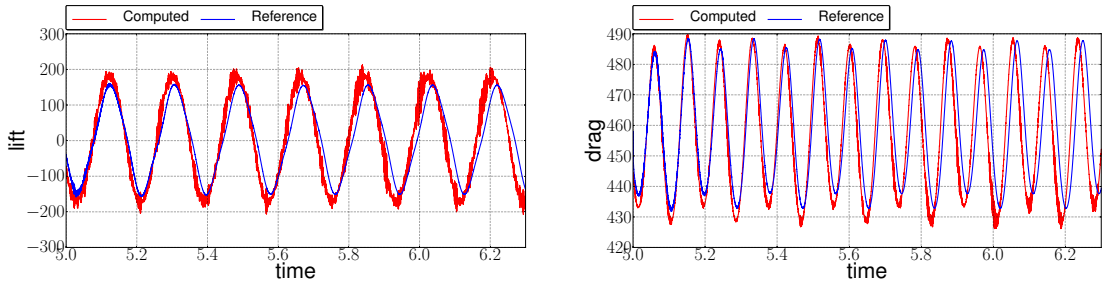
(b) y-déplacement(t)

(c) Portance(t)

(d) Traînée(t)

FIGURE 9.7 – Resultats pour **FSI3** avec la configuration **(3)** de la table 9.3

remarque ainsi que les résultats de la configuration **(4)** se détériorent fortement pour la portance. Nous pouvons aussi illustrer ces perturbations à l'aide de la figure 9.8(a) pour la portance. On voit aussi apparaitre quelques oscillations plus légères pour la traînée avec la figure 9.8(b). Une description de ce phénomène est aussi faite dans [30]. Les auteurs expliquent que l'apparition de ces oscillations est due à la relaxation dynamique d'Aitken utilisée dans l'algorithme de point fixe pour le fluide structure. Pour éviter cette instabilité, il propose de baisser la tolérance du point fixe, mais cela détériore les performances. Une autre solution décrite consiste à utiliser un paramètre de relaxation θ fixe. Pour cette application, le paramètre fixe optimal semble être $\theta = 0.5$.



(a) Portance

(b) Traînée

FIGURE 9.8 – Perturbations obtenues avec la configuration **(4)** pour la portance et la traînée.

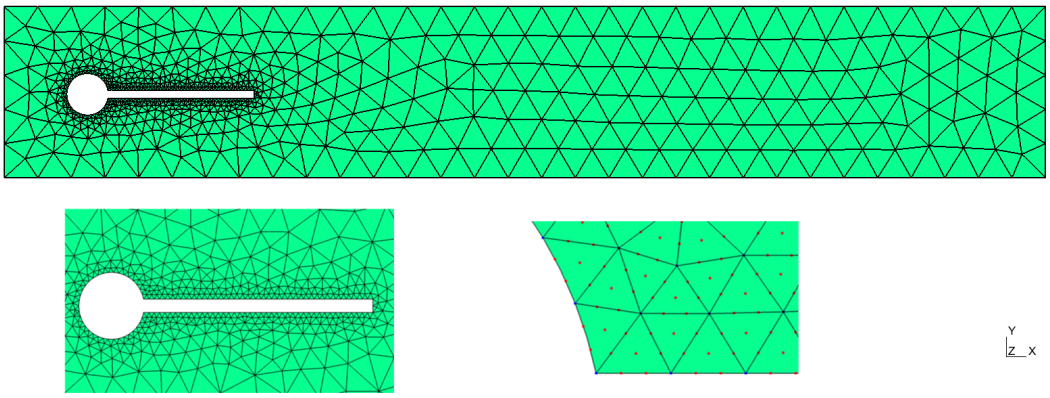
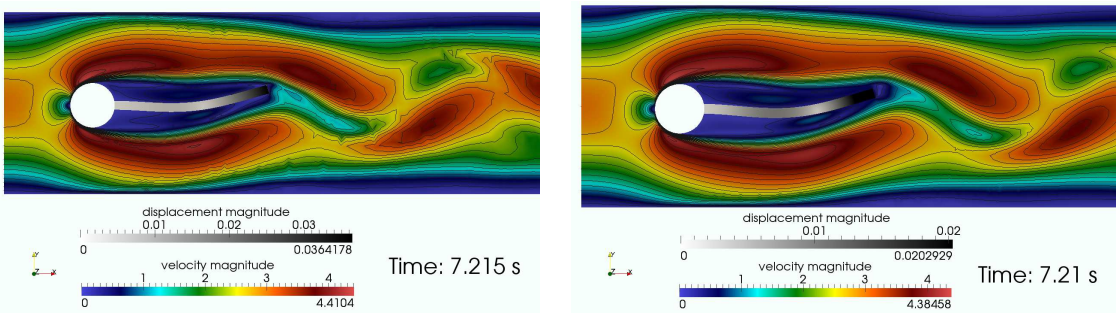


FIGURE 9.9 – Maillage de référence utilisé dans la configuration **(1)** de la table 9.3

Finalement, nous concluons ce chapitre en montrant les figures 9.10(a) et 9.10(b) qui représentent le champ de vitesse obtenu pour deux configurations différentes.



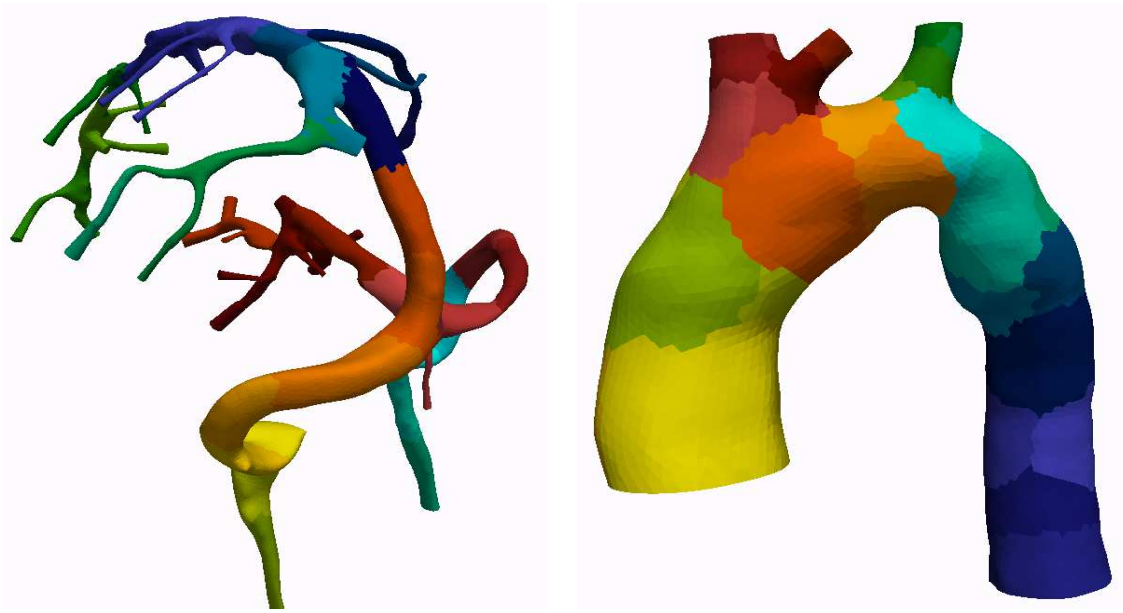
(a) Configuration **(1)** (b) Configuration **(3)**

FIGURE 9.10 – Champ de vitesse obtenu pour **FSI3** avec les configurations **(1)** et **(3)**

Chapitre 10

Simulation des écoulements sanguins

Pour ce dernier chapitre, nous allons présenter les simulations numériques que nous avons réalisées dans le sens du titre de la thèse, vers les écoulements sanguins. Et comme le titre l'indique, les simulations effectuées ne représenteront pas des modélisations qui pourront prétendre être réaliste. En effet, de nombreuses hypothèses physiologiques ont été négligées comme le caractère non newtonien des écoulements, l'anisotropie et la visco-élasticité des parois artérielles. Les conditions aux limites et les paramètres physiologiques seront aussi simplifiés. Toutefois, nous avons eu l'opportunité de pouvoir réaliser certaines simulations sur des géométries réalistes (deux exemples avec la figure 10.1). Elles étaient créées à partir de l'imagerie médicale et les maillages étaient ensuite générés à l'aide d'un processus complexe [110, 141].



(a) système veineux cérébral [141] (b) aorte (issue du site internet de GMSH [67])

FIGURE 10.1 – Exemples de géométries réalistes (coloriées avec un partitionnement).

Le but de ce chapitre est d'expérimenter les outils qui ont été développés dans cette thèse dans l'ambition de réaliser des simulations proches des écoulements sanguins. Nous modéliserons principalement des applications représentant la propagation d'une onde de

pression dans un vaisseau sanguin. Ce phénomène est possible grâce à l'interaction fluide structure entre le plasma et la paroi artérielle. L'outil d'interpolation sera l'une des principales clés de la réussite de cette modélisation. Pour des résultats plus précis, nous utiliserons également notre méthode d'interaction fluide structure pour des déplacements d'ordre élevé. De plus, la taille importante des géométries et la précision des calculs nécessiteront l'utilisation de l'environnement parallèle dans ces domaines complexes. Le modèle FBM-FSI qui a pour but de modéliser les globules rouges n'est pas encore assez mûr pour obtenir des résultats de rhéologie sanguine. Cette simulation ne sera donc pas présente dans ce chapitre.

Nous commencerons cette partie par la réalisation de « benchmarks classiques » avec interaction fluide structure en 2D et 3D. Ces applications modéliseront la propagation d'une onde de pression dans un vaisseau sanguin idéalisé. Ensuite, nous montrerons quelques résultats numériques obtenus avec des géométries réalistes issues de l'imagerie médicale. Tout d'abord, nous montrerons une simulation d'écoulement dans une partie du réseau veineux du cerveau. Nous terminerons avec deux simulations FSI dans l'aorte et dans une artère contenant un anévrisme. La simulation dans ces domaines complexes a été possible grâce à l'implémentation parallèle qui a été mise en place dans cette thèse.

1 Applications 2D

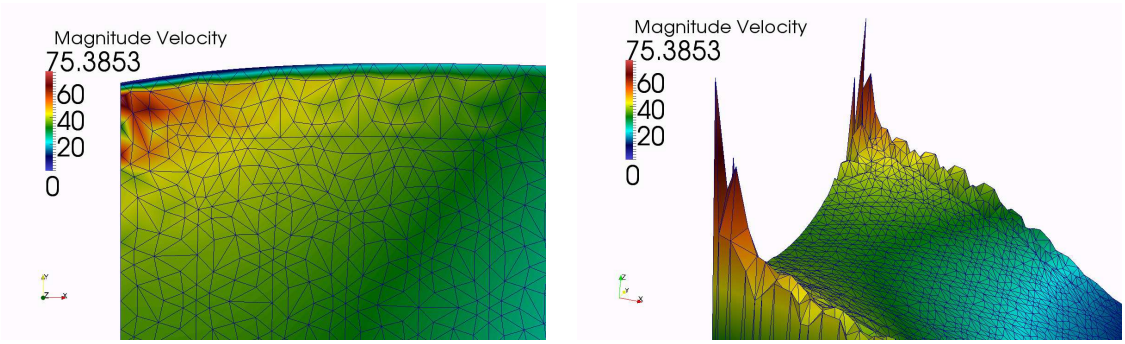
Nous présentons maintenant les résultats numériques obtenus pour la modélisation des écoulements sanguins bidimensionnelle. Il s'agit de reproduire un phénomène de propagation d'onde de pression dans un canal droit à l'aide d'un modèle d'interaction fluide structure. Ce type d'application a aussi été expérimenté dans [124, 123, 66].

FIGURE 10.2 – Géométrie de l'application 2D.

Comme le montre la figure 10.2, le canal où le fluide s'écoule est un rectangle de longueur 6 cm et de largeur 1 cm. Les parois du haut et du bas sont mobiles et interagissent donc avec l'écoulement. Nous n'avons pas représenté le domaine élastique représentant la paroi vasculaire, car nous utilisons le modèle réduit 1D que nous avons décrit à la section 5 du chapitre 5. Ainsi, nous représentons le domaine structure par $\Omega_s^* = \Gamma_{fsi}^*$. Le modèle utilisé pour le fluide est celui de Navier-Stokes en domaine mobile. Les constantes matérielles de l'écoulement sont : $\rho_f = 1\ gr/cm^3$ et $\mu_f = 0.003\ poise$. Pour la structure, nous avons utilisé le modèle *generalized string* qui est caractérisé par : $\rho_s^* = 1.1\ gr/cm^3$, $h = 0.1\ cm$, $E_s = 0.75 \cdot 10^6\ dynes/cm^2$, $R_0 = 0.5\ cm$, $G_s = 10^5\ Pa$, $\nu_s = 0.5$, $k = 2.5$ et $\gamma_v = 0.01$. Pour la condition aux limites en entrée du fluide, nous imposons l'impulsion de l'onde de pression sur $\Gamma_f^{i,t}$ par :

$$\boldsymbol{\sigma}_f \boldsymbol{n}_f = \begin{cases} \left(-\dfrac{2 \cdot 10^4}{2} \left(1 - \cos\left(\dfrac{\pi t}{2.5 \cdot 10^{-3}} \right) \right), 0 \right)^T & \text{si } t < 0.005 \\ \mathbf{0} & \text{sinon} \end{cases} \tag{10.1}$$

Pour modéliser la sortie de l'écoulement sur $\Gamma_f^{o,t}$, deux types de conditions vont être expérimentés. Nous avons la condition « classique » $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0}$ qui sera aussi appelée sortie libre ($f$ree outlet). Nous utiliserons également le modèle de Windkessel (décrit à la section 5.1 du chapitre 4) comme condition aux limites de sortie. Ainsi, nous imposons $\boldsymbol{\sigma}_f \boldsymbol{n}_f = -P_0 \boldsymbol{n}_f$ avec P_0 la pression proximale calculée par ce modèle. Les paramètres utilisés pour ce modèle de conditions aux limites sont tirés de [21] pour une application 3D dans un tube cylindrique. Nous avons ainsi pris $R_p = 400$, $R_d = 6.2 \cdot 10^3$ et $C_d = 2.72 \cdot 10^{-4}$. Au niveau de la structure élastique, nous choisissons d'imposer $\eta_s = 0$ à chaque extrémité. Nous imposons également un déplacement nul sur les bords $\Gamma_f^{i,t}$ et $\Gamma_f^{o,t}$. Enfin pour le couplage fluide structure, nous utilisons les équations (5.46)-(5.48) que nous avons présentées lors de la description du modèle de structure réduit. Au niveau du critère d'arrêt de la convergence de l'algorithme FSI, nous choisissons de prendre une tolérance relative égale à 10^{-6}. Pour la discrétisation en temps, nous utilisons un schéma BDF_2 pour le fluide et un schéma de Newmark pour la structure. Le pas de temps utilisé dans toutes nos simulations sera $\Delta t = 0.0001\ s$.



(a) Magnitude de la vitesse (t=0.0027s)

(b) Élévation de la magnitude de la vitesse (t=0.0027s)

FIGURE 10.3 – Illustration de l'instabilité numérique apparaissant dans l'application 2D sans la stabilisation CIP.

Après les premières tentatives de simulation de cette application, nous nous sommes heurtés à de fortes instabilités numériques situées au niveau des coins de l'entrée du fluide. Ces instabilités sont illustrées avec les figures 10.3(a) et 10.3(b). On y voit une forte augmentation de la magnitude de la vitesse dans ces zones. Cela a pour conséquence de faire diverger rapidement notre solveur fluide structure. Nous avons remarqué que ces instabilités étaient d'autant plus fortes que l'intensité de l'onde de pression était importante. En réduisant cette dernière, il était alors possible de calculer une solution numérique qui modélisait la propagation de l'onde de pression. Cette instabilité semble être causée par la condition de Neumann imposée en entrée. Ce problème a déjà été évoqué dans la littérature [117, 95]. Ces auteurs proposent d'ajouter un terme de stabilisation pour l'advection. Dans cette thèse, nous avons choisi d'appliquer une stabilisation de la convection à l'aide de la formulation CIP stabilisée présentée au chapitre 6. Cette technique nous a aussi permis de résoudre le problème d'instabilité. Cependant, l'utilisation de cette technique est assez coûteuse dans la pratique. En effet, elle a pour conséquence d'augmenter considérablement le nombre de non-zéros dans la matrice. Cela induit un coût important

pour l'assemblage des structures algébriques, mais aussi pour la résolution des systèmes linéaires. C'est pourquoi, nous avons effectué cette stabilisation uniquement dans la zone d'instabilité. Cela permet de résoudre le problème d'instabilité tout en gardant des performances proches de la version non stabilisée. Au niveau de la zone de stabilisation, plusieurs choix ont été testés. En rappelant que $\mathcal{F}$ désigne l'ensemble des faces du maillage, nous désignerons par $\mathcal{F}_{stab}$ l'ensemble des faces qui sont stabilisées. Pour les résultats numériques qui vont suivre, nous avons utilisé :

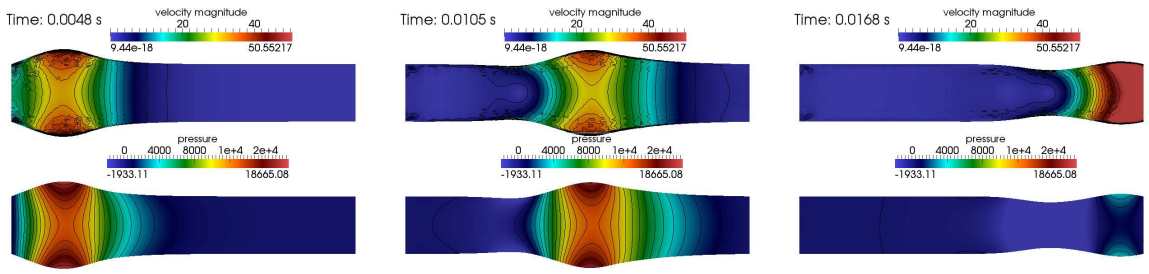
$$\mathcal{F}_{stab} = \mathcal{F} \bigcap \left(\left\{ (x,y) \in \mathbb{R}^2 : (x - 0.3) \leqslant 0 \right\} \cup \left\{ (x,y) \in \mathbb{R}^2 : (x - 5.7) \geqslant 0 \right\} \right) \tag{10.2}$$

Nous illustrons maintenant les résultats obtenus à l'aide de la stabilisation CIP. Tout d'abord, nous présentons les configurations utilisées pour cette étude avec la table 10.1. Elles illustrent notre intérêt pour les approximations d'ordre élevé en espace et en géométrie. On remarque que le nombre d'éléments utilisé pour la discrétisation géométrique du fluide est faible. La structure quant à elle est composée uniquement d'éléments correspondant aux arêtes du fluide formant l'interface fluide structure.

Config	fluide			structure		
	N_{elt}	N_{geo}	N_{dof}	N_{elt}	N_{geo}	N_{dof}
(1)	342	3 (P4P3)	7377	58	1	176 (P3)
(2)	342	4 (P5P4)	11751	58	1	234 (P4)

TABLE 10.1 – Configurations utilisées pour l'application 2D.

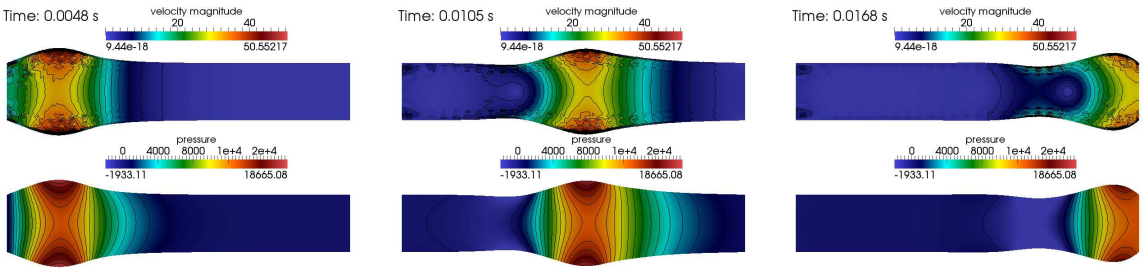
Nous commençons par l'application où l'on a utilisé la condition de sortie libre. Nous avons avec la figure 10.4 trois captures d'écran représentant trois instants de la simulation avec la configuration (2). Avec ces figures, nous pouvons voir la propagation de l'onde de pression. On remarque alors l'atténuation de l'onde de pression à la sortie du domaine, figure 10.4(c). On constate également dans cette zone une forte augmentation de la vitesse du fluide. Cela aura pour conséquence d'engendrer des phénomènes non physiologiques que nous décrirons dans la suite.



(a) t=0.0048 s (b) t=0.0105 s (c) t=0.0168 s

FIGURE 10.4 – Résultats obtenus à trois instants différents avec la condition de sortie libre. Pour chaque figure, nous avons affiché la magnitude de la vitesse et la pression avec une déformation amplifiée par 5.

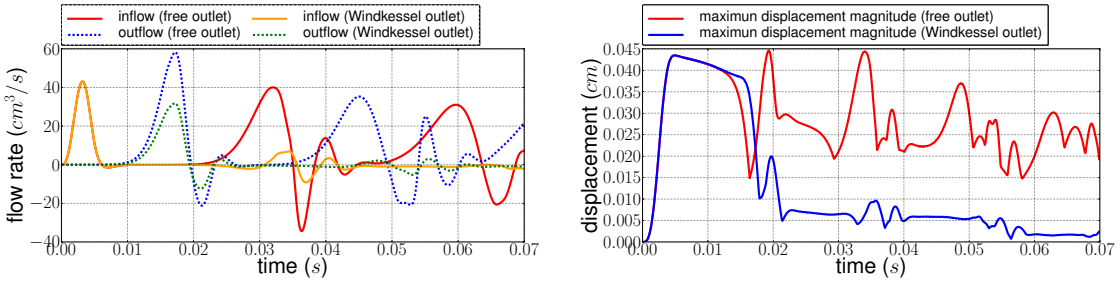
Nous avons ensuite la figure 10.5 qui représente la même simulation, mais avec les conditions aux limites de sortie basées sur le modèle de Windkessel. Ces résultats utilisent la configuration (1). Nous constatons que le comportement de la simulation est assez similaire pour le début de la propagation de l'onde de pression, figures 10.5(a) et 10.5(b). Par contre, la sortie de l'écoulement est beaucoup plus réaliste, figure 10.5(c). L'onde de pression se propage beaucoup plus naturellement à l'aide de cette condition aux limites. Nous montrerons par la suite des résultats quantitatifs de cette illustration qualitative.



(a) t=0.0048 s (b) t=0.0105 s (c) t=0.0168 s

FIGURE 10.5 – Résultats obtenus à trois instants différents avec la condition de sortie utilisant le modèle de Windkessel. Pour chaque figure, nous avons affiché la magnitude de la vitesse et la pression avec une déformation amplifiée par 5.

Dans ces deux simulations, on remarque cependant que le champ de vitesse est assez perturbé proche de l'interface fluide structure. Ces phénomènes n'ont pas une importance cruciale dans la modélisation de cet écoulement sanguin. Toutefois, le raffinement du maillage aux alentours de cette interface permet d'améliorer cette quantité. On remarque également qu'avec le même maillage, l'approximation spatiale et géométrique d'ordre plus élevé (figures 10.4(a) et 10.4(b)) améliore la précision de ce phénomène comparée à une approximation plus faible (figure 10.5(a) et 10.5(b)).

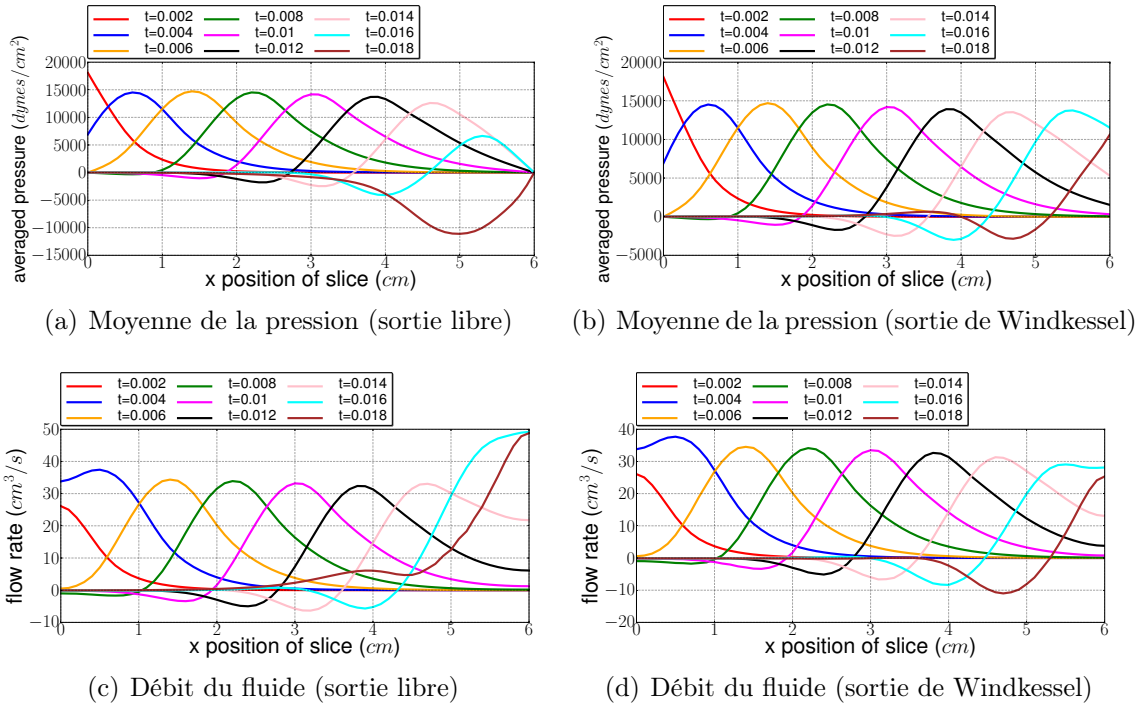


(a) Débits en entrée et sortie (b) Maximum de la magnitude du déplacement

FIGURE 10.6 – Comparaison entre la condition de sortie libre et la condition de sortie utilisant le modèle de Windkessel.

Nous passons à présent à la présentation de résultats quantitatifs de cette simulation. Nous trouvons tout d'abord avec la figure 10.6 une comparaison entre les conditions aux limites de sortie libre et les conditions aux limites de sortie couplée au modèle de Windkessel. Nous avons représenté l'évolution des débits en entrée et sortie avec la figure 10.6(a). On voit dans un premier temps la similitude entre les deux tests pour l'entrée

de l'écoulement au début de la simulation. Lorsque l'onde de pression arrive à la sortie du tube, on constate une différence avec l'augmentation du débit pour le test avec la sortie libre. La condition de sortie libre et la structure fixée à ces extrémités entrainent alors l'apparition de phénomènes non physiologiques. L'onde de pression est réfléchie sur le bord $\Gamma_f^{o,t}$, ce qui cause la propagation d'une onde de pression dans le sens inverse de l'écoulement. Puis l'onde arrive de nouveau à l'entrée et elle est réfléchie et ainsi de suite. Ces phénomènes sont illustrés sur la figure 10.6(a) avec de fortes variations du débit après le passage de la première onde de pression ($t > 0.02$). Par contre, on voit que le modèle de Windkessel permet de réduire considérablement ce phénomène. Il reste encore quelques résidus non physiologiques qui sont principalement dus à la fixation de la structure et du couplage réalisé avec le modèle 0D. Nous avons tracé sur la figure 10.6 le maximum du déplacement sur l'interface fluide structure. On constate les mêmes phénomènes évoqués précédemment. La simulation avec la sortie libre entraine un mouvement de la structure qui se prolonge après le passage de l'onde de pression initiale. Le modèle 0D réduit fortement ces perturbations. On conclut que cette condition aux limites est importante.



(a) Moyenne de la pression (sortie libre)

(b) Moyenne de la pression (sortie de Windkessel)

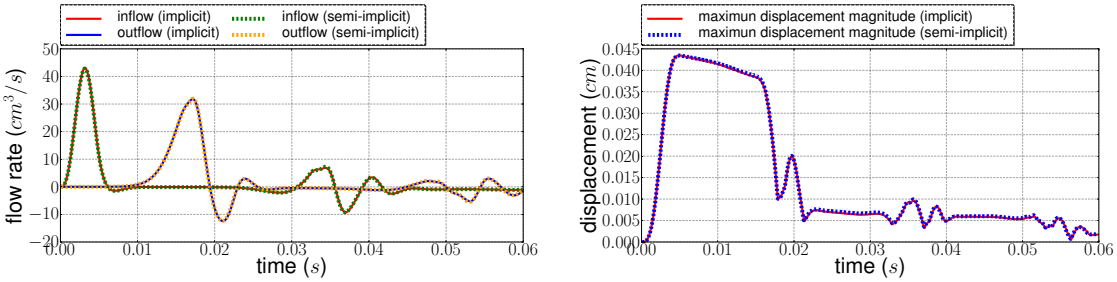
(c) Débit du fluide (sortie libre)

(d) Débit du fluide (sortie de Windkessel)

FIGURE 10.7 – Mesure de la moyenne de la pression et du débit pour plusieurs sections verticales (figures de gauche pour la sortie libre et figures de droite pour la sortie de Windkessel).

Nous avons ensuite mesuré la moyenne de la pression et le débit sur des tranches longitudinales de la géométrie fluide. Nous avons utilisé 60 sections $\{x_i\}_{i=1}^{60}$ définies par $x_i = 0.1i$. Nous avons tracé ces mesures pour plusieurs instants de la simulation avec la figure 10.7. Ces mesures ont également été faites dans [123] et nos résultats sont en très bon accord. On constate également les phénomènes non physiologiques en sortie de l'écoulement que nous avons décrits précédemment.

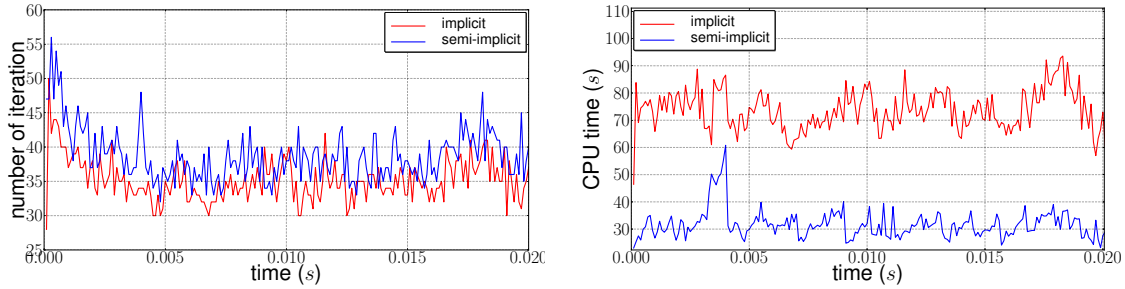
La figure 10.8 montre une comparaison entre les deux algorithmes de résolution du modèle d'interaction fluide structure. Il s'agit de l'algorithme implicite et de l'algorithme



(a) Débits en entrée et sortie (b) Maximum de la magnitude du déplacement

FIGURE 10.8 – Comparaison entre le couplage FSI implicite et semi-implicite avec utilisation du modèle de Windkessel.

semi-implicite. Ce test comparatif a été effectué avec la configuration (2) et la condition de sortie du fluide utilisant le modèle Windkessel. On constate alors que les courbes de débits (figure 10.8(a)) et de déplacement (figure 10.8(b)) se superposent parfaitement. On montre ainsi que la précision obtenue avec l'algorithme semi-implicite est très bonne pour cette application.



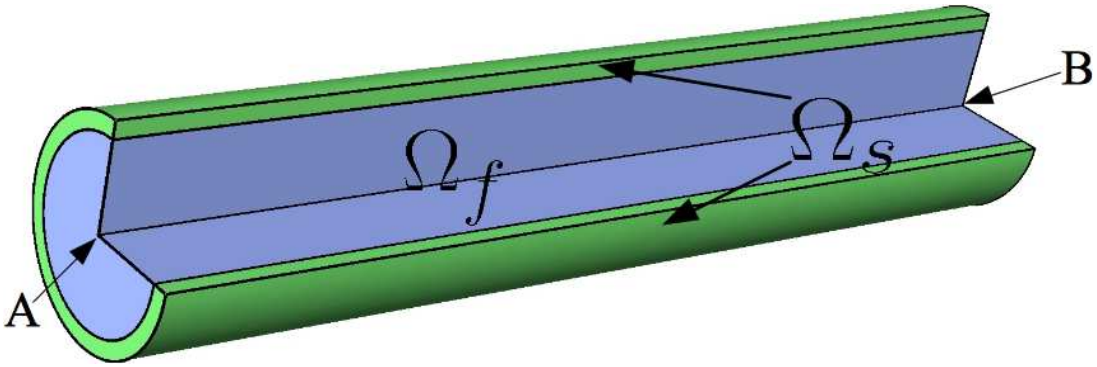
(a) Nombre d'itérations FSI (b) Temps CPU à chaque pas de temps

FIGURE 10.9 – Comparaison entre le couplage FSI implicite et semi-implicite avec utilisation du modèle de Windkessel.

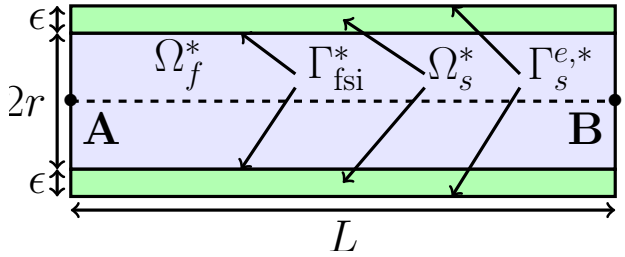
Enfin, nous montrons les performances de l'algorithme semi-implicite avec la figure 10.9. La configuration utilisée est la même que celle du test précédent. Nous avons d'abord illustré le nombre d'itérations nécessaire pour la convergence des deux algorithmes FSI à chaque pas de temps avec la figure 10.9(a). Nous voyons que la méthode semi-implicite requiert un peu plus d'itération que le cas implicite. Cependant, on s'aperçoit que cet algorithme est deux à trois fois plus performant que l'algorithme implicite. C'est ce que nous avons illustré avec la figure 10.9(b). Cette performance s'explique par les optimisations possibles avec cet algorithme. Tout d'abord, le calcul de la carte ALE est fait une seule fois par pas de temps. Comme le domaine est fixe dans les itérations FSI, nous pouvons stocker tous les termes linéaires de la matrice jacobienne et certains termes du résidu et les réutiliser à chaque itération. On stocke également la partie dépendante de la carte ALE dans la relation de contrainte (5.47). Ce terme assez complexe a un coût de calcul non négligeable.

2 Applications 3D

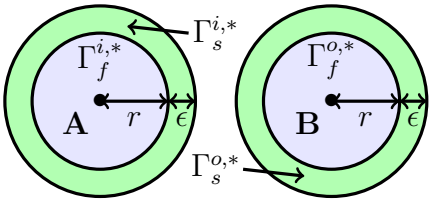
Nous passons à présent à la réalisation de simulations 3D dans les écoulements sanguins. Elle consiste toujours à représenter la propagation d'une onde de pression dans un vaisseau, mais cette fois-ci nous utiliserons une conduite cylindrique comme domaine d'étude. Cette application a aussi souvent été utilisée dans la littérature, voir par exemple [66, 156, 59, 33]. Elle peut être considérée comme un cas test pour l'interaction fluide structure. Nous avons détaillé la géométrie du problème avec la figure 10.10 en utilisant toujours les mêmes notations. Cette illustration représente les domaines initiaux (et de référence) pour le fluide et la structure. Les paramètres géométriques sont donnés par $r = 0.5\ cm$, $L = 5\ cm$, $\epsilon = 0.1\ cm$, $\mathbf{A} = (0, 0, 0)$ et $\mathbf{B} = (5, 0, 0)$.



(a) Géométrie globale de la conduite cylindrique



(b) Section longitudinale



(c) Sections entrée et sortie

FIGURE 10.10 – Géométrie de l'application 3D.

Aucune réduction de modèle n'a été faite pour cette application. Le fluide et la structure seront représentés par les modèles 3D que nous avons présentés tout au long de cet ouvrage. Les caractéristiques des matériaux sont : $\rho_f = 1\ g/cm^3$, $\mu_f = 0.03\ poise$, $\rho_f^* = 1.2\ g/cm^3$, $E_s = 3 \cdot 10^6\ dynes/cm^2$ et $\nu_s = 0.3$. Pour la condition en entrée du fluide défini sur $\Gamma_f^{i,t}$, nous considérons dans toutes nos applications la relation suivante :

$$\boldsymbol{\sigma}_f \boldsymbol{n}_f = \begin{cases} \left(-\dfrac{1.3332 \cdot 10^4}{2} \left(1 - \cos\left(\dfrac{\pi t}{1.5 \cdot 10^{-3}} \right) \right), 0 \right)^T & \text{si } t < 0.003 \\ \mathbf{0} & \text{sinon} \end{cases} \tag{10.3}$$

Cette expression est très similaire à celle présentée dans l'application 2D sauf que l'intensité est légèrement moins forte. Pour le bord de la structure en entrée, nous imposons une contrainte de non-mobilité. Cette condition n'est pas très réaliste dans notre contexte,

mais l'intérêt de cette étude s'axe plutôt sur le comportement de la déformation de la paroi vasculaire au milieu et en sortie d'écoulement. Au niveau du couplage FSI, nous utilisons les conditions standards présentées au chapitre 6. La tolérance utilisée pour l'algorithme de couplage est égale à 10^{-5}. Pour les autres conditions aux limites, nous allons expérimenter deux types de modélisations. La première, que l'on notera **BC-1**, impose les relations suivantes :

- $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0}$ sur $\Gamma_f^{o,t}$
- $\boldsymbol{\eta}_s = \mathbf{0}$ sur $\Gamma_s^{i,*} \cup \Gamma_s^{o,*}$
- $\boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s^* = \mathbf{0}$ sur $\Gamma_s^{e,*}$
- $\mathcal{A}_f^t = \mathbf{x}^*$ sur $\Gamma_f^{i,*} \cup \Gamma_f^{o,*}$

Nous montrerons dans la suite que ces types de conditions ne nous permettront pas d'obtenir des résultats réalistes de simulation d'écoulement sanguin. C'est pourquoi, nous avons utilisé des conditions plus physiologiques proposées dans [42, 118]. On les note par **BC-2** et elles s'expriment par :

- $\boldsymbol{\sigma}_f \boldsymbol{n}_f = -P_0 \boldsymbol{n}_f$ sur $\Gamma_f^{o,t}$ (Windkessel)
- $\boldsymbol{\eta}_s = \mathbf{0}$ sur $\Gamma_s^{i,*}$
- $\boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s^* + \alpha \boldsymbol{\eta}_s = \mathbf{0}$ sur $\Gamma_s^{e,*}$
- $\boldsymbol{F}_s \boldsymbol{\Sigma}_s \boldsymbol{n}_s^* = \mathbf{0}$ sur $\Gamma_s^{o,*}$
- $\mathcal{A}_f^t = \mathbf{x}^*$ sur $\Gamma_f^{i,*}$
- $\nabla \mathcal{A}_f^t \boldsymbol{n}_f^* = \boldsymbol{n}_f^*$ sur $\Gamma_f^{o,*}$

Pour les paramètres du modèle de Windkessel, nous avons pris les mêmes que pour l'applicatition 2D, c'est-à-dire $R_p = 400$, $R_d = 6.2 \cdot 10^3$ et $C_d = 2.72 \cdot 10^{-4}$. La stabilisation CIP utilisée dans l'application précédente est aussi nécessaire pour la stabilité de cette application. Par contre, nous avons utilisé cette technique uniquement sur la frontière entrée $\Gamma_f^{i,t}$ et la frontière sortie $\Gamma_f^{o,t}$. Cette restriction est suffisante ici, contrairement à l'application 2D, car l'intensité de l'onde de pression est moins forte.

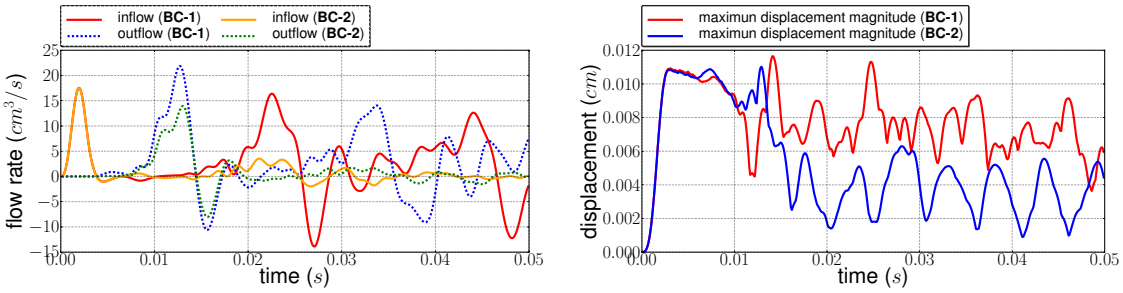
Nous avons utilisé plusieurs configurations pour cette application, elles sont décrites dans la table 10.2. La colonne Incomp désigne l'utilisation ou pas de la contrainte d'incompressibilité pour le modèle de structure. Pour toutes nos simulations, nous utilisons le pas de temps $\Delta t = 0.0001s$ et un schéma BDF_2 pour le fluide. Tous nos calculs ont été faits en parallèle (entre 4 et 16 proc selon les cas) avec un préconditionneur gasm1 combiné à un préconditionneur LU dans chaque bloc (voir chapitre 7).

Config	fluide			structure			
	N_{elt}	N_{geo}	N_{dof}	N_{elt}	N_{geo}	N_{dof}	Incomp
(1)	13625	1	69836(P2P1)	12961	1	12876(P1)	non
(2)	13625	1	69836(P2P1)	12961	1	81536 (P2P1)	oui
(3)	1609	2	30744(P3P2)	3361	2	19878 (P2)	non

TABLE 10.2 – Configurations utilisées pour l'application 3D.

Nous commençons par effectuer une comparaison entre les conditions aux limites **BC-1** et **BC-2** avec la figure 10.11. Pour cela, nous avons mesuré les débits en entrée et en sortie ainsi que le maximum de la magnitude du déplacement de la structure. Pour les deux simulations présentes, nous utilisons la configuration (1) avec un modèle de Navier-

Stokes pour le fluide et un modèle hyperélastique compressible de Saint-Venant-Kirchhoff pour la structure. On retrouve alors les mêmes phénomènes non physiologiques que nous avons évoqués pour l'application 2D avec l'utilisation de **BC-1**. La réflexion de l'onde de pression lorsqu'elle arrive en sortie d'écoulement est représentée par l'augmentation du débit, figure 10.11(a). Nous constatons ensuite de fortes variations périodiques du débit en entrée et sortie signifiant que l'écoulement est encore actif, ce qui n'est pas réaliste. Cela a pour conséquence le déplacement important de la structure après le passage de l'onde de pression initial, figure 10.11(b). Ces résultats sont en accord avec [66, 59, 33] qui ont utilisé des conditions aux limites similaires. Grâce à l'utilisation des conditions **BC-2**, nous arrivons à réduire fortement ces perturbations. Il reste toutefois quelques résidus non physiologiques dus à l'imperfection de nos conditions.



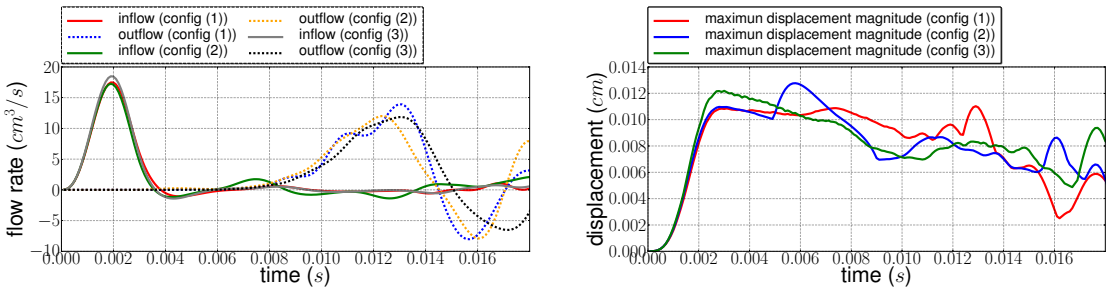
(a) Débit en entrée et sortie

(b) Maximun de la magnitude du déplacement

FIGURE 10.11 – Comparaison entre les conditions aux limites **BC-1** et **BC-2** en utilisant la configuration (1) avec Navier-Stokes pour le fluide et Saint-Venant-Kirchhoff compressible pour la structure.

Nous trouvons maintenant avec la figure 10.12 un test de comparaison entre les trois configurations de la table 10.2 utilisant les conditions aux limites **BC-2**. L'intervalle de temps d'étude est plus court que le test précédent, il correspond au temps nécessaire pour que l'onde de pression propagée dans le tube s'évacue. Pour toutes les configurations, nous avons les équations de Navier-Stokes comme modèle pour le fluide. Pour la structure, les configurations (1) et (3) utilisent un modèle de Saint-Venant-Kirchhoff compressible et la configuration (2) un modèle de Mooney-Rivlin incompressible. Le choix des coefficients du modèle de Mooney-Rivlin est pris de la même manière qu'au chapitre 5 section 4.2, c'est-à-dire qu'ils peuvent être exprimés en fonction des coefficients de Lamé (et donc avec le module de Young et le coefficient de Poisson). On remarque que la configuration (3) utilise l'ordre 2 en géométrie et contient très peu d'éléments dans les maillages fluides et structures comparé aux autres configurations. Comme pour le test précédent, nous avons mesuré les débits en entrée et sortie (figure 10.12(a)) et le maximum de la magnitude du déplacement de la structure (figure 10.12(b)). On remarque tout d'abord que les trois configurations donnent les mêmes résultats en début de simulation, c'est-à-dire lors de la création de l'onde de pression. Une légère différence est à noter avec la configuration (3) causée par l'approximation géométrique d'ordre 2. En effet, cette discrétisation géométrique diffère des deux autres configurations. Elle respecte a priori mieux la géométrie du cylindre. On constate ensuite qu'il y a une sorte de « cassure » avec le modèle incompressible de structure vers t=0.005 s. Avec la visualisation de la solution, on voit clairement que ce phénomène est dû à la condition aux limites en sortie de la structure.

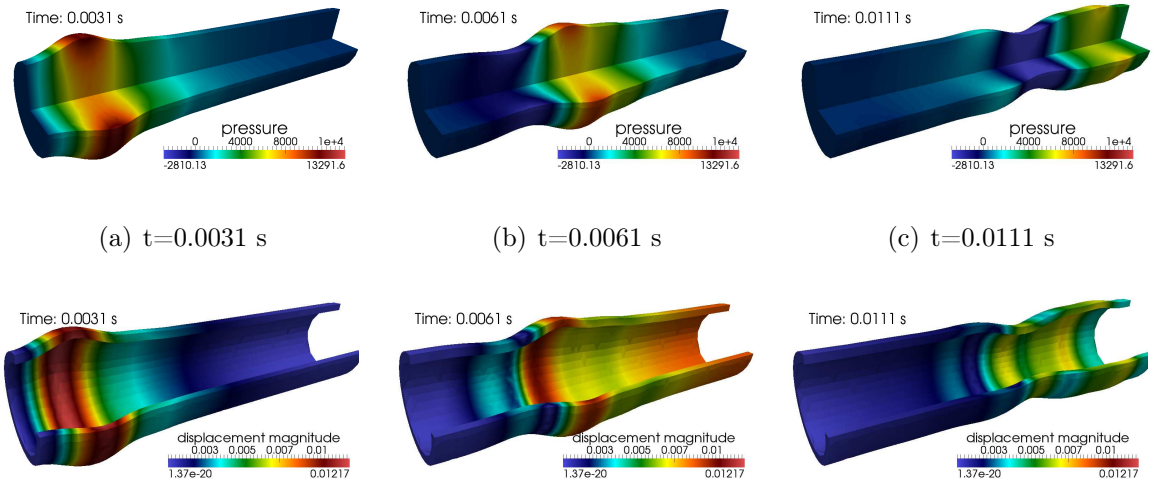
La combinaison de l'incompressibilité et de ces conditions aux limites provoque donc ce phénomène et montre l'imperfection des conditions aux limites. Pour le reste de la simulation, le comportement global des trois configurations est assez similaire. Les principaux contrastes visibles se produisent au moment de l'évacuation de l'onde de pression, avec une différence dans l'amplitude et un léger décalage dans les mesures. Pour mieux caractériser le comportement de cette simulation et l'intérêt de l'ordre élevé, une simulation sur maillage très fin devrait être effectuée. Malheureusement, les ressources informatiques dont nous disposons à l'heure actuelle ne nous ont pas permis de réaliser ce calcul (mais cela devrait être possible dans un futur proche).



(a) Débits en entrée et sortie (b) Maximun de la magnitude du déplacement

FIGURE 10.12 – Comparaison entre plusieurs configurations utilisant les conditions aux limites **BC-2**.

Pour illustrer visuellement cette application, nous avons affiché avec la figure 10.13 quelques captures d'écran de la simulation utilisant la configuration (3) et **BC-2**. Le modèle pour le fluide est toujours celui de Navier-Stokes. Pour la structure, c'est un modèle hyperélastique de Saint-Venant-Kirchhoff compressible. Nous montrons la position de l'onde de pression à plusieurs instants ainsi que le déplacement de la structure.

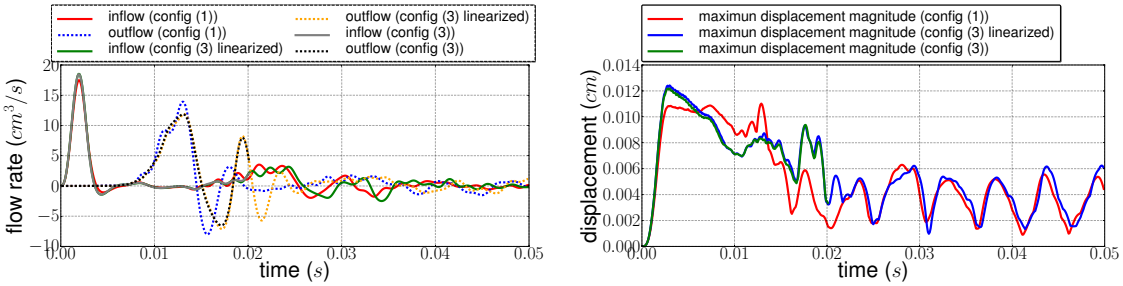


(a) t=0.0031 s (b) t=0.0061 s (c) t=0.0111 s

(d) t=0.0031 s (e) t=0.0061 s (f) t=0.0111 s

FIGURE 10.13 – Résultats obtenus à trois instants différents avec la configuration (2) et les conditions aux limites **BC-2**. Nous avons en haut la pression du fluide et en bas la magnitude du déplacement de la structure. Pour toutes les figures, la déformation du domaine est amplifiée par 15.

Nous avons ensuite effectué un autre test comparatif avec la figure 10.14. Celui-ci vise à confronter des modèles non linéaires (Navier-Stokes et hyperélasticité compressible avec Saint-Venant-Kirchhoff) combinés avec un solveur FSI implicite et des modèles linéaires simplifiés (Oseen et élasticité linéaire) combinés avec un solveur FSI semi-implicite. Pour cela, nous avons utilisé la configuration (1) et (3) pour le cas des modèles non linéaires. Nous avons noté par configuration (3) linearized le cas linéaire simplifié basé sur la configuration (3). On constate que la différence des mesures entre le modèle non linéaire et le modèle simplifié utilisant la configuration (3) est très faible, aussi bien pour les débits (figure 10.14(a)) que pour le déplacement (figure 10.14(b)). Cette comparaison est faite jusqu'à t=0.02 s. Nous montrons aussi que pour t>0.02s, le comportement est très similaire avec la configuration (1) dans les effets résiduels du passage de l'onde de pression. Ainsi, on peut conclure que pour cette application, l'utilisation des modèles complexes peut être inutile. De plus, grâce à l'utilisation de l'ordre élevé géométrique et d'un maillage « grossier » (configuration (3)), nous arrivons à obtenir des résultats intéressants.

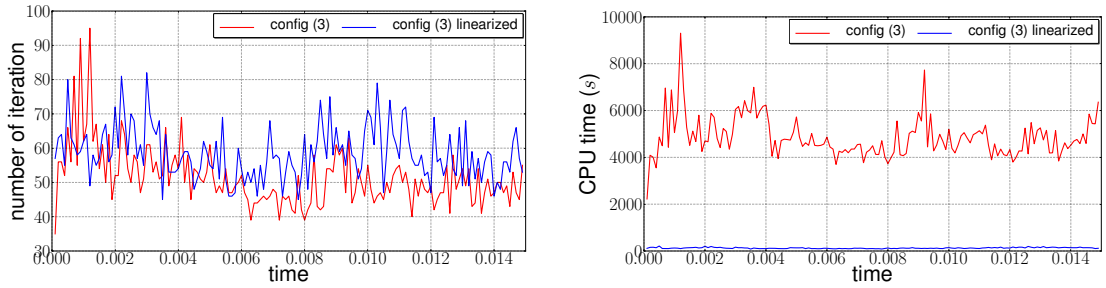


(a) Débits en entrée et sortie

(b) Maximun de la magnitude du déplacement

FIGURE 10.14 – Comparaison (mesures physiques) entre des modèles non linéaires avec un algorithme FSI implicite et un modèle linéaire avec un algorithme FSI semi-implicite.

Finalement, nous montrons avec la figure 10.15 une comparaison des performances entre une version non linéaire implicite et une version linéaire semi-implicite. Pour chaque cas, c'est la configuration (3) qui est utilisée avec les conditions **BC-2**. Ces deux calculs ont été lancés en parallèle sur la même machine avec 10 processeurs. Nous présentons d'abord avec la figure 10.15(a) le nombre d'itérations de l'algorithme FSI à chaque pas de temps. On peut voir que le schéma semi-implicite requiert un peu plus d'itération pour converger que le schéma implicite. Nous avons avec la figure 10.15(b) une comparaison du temps CPU écoulé à chaque pas de temps pour ces deux configurations. On constate alors un temps de calcul nettement inférieur avec la version simplifiée. En moyenne, le cas non linéaire implicite met 5763s contre 150s pour le cas simplifié, ce qui nous fait une accélération d'un facteur de presque 40 pour ce modèle simplifié. Dans chacun des cas, nous avons appliqué des optimisations très efficaces comme la réutilisation du préconditionneur et le stockage de certaines parties des structures algébriques. L'efficacité du modèle simplifié est liée principalement aux stockages des structures algébriques dans toutes les itérations de l'algorithme FSI semi-implicite, car le domaine de calcul ne se déplace pas, ce qui n'est pas possible avec le schéma implicite.



(a) Nombre d'itérations FSI

(b) Temps CPU à chaque pas de temps

FIGURE 10.15 – Comparaison (mesures performances) entre un modèle non linéaire avec un algorithme FSI implicite et un modèle linéaire avec un algorithme FSI semi-implicite.

3 Simulations dans des géométries réalistes

Nous présentons dans cette dernière partie quelques résultats numériques obtenus sur des géométries réalistes d'écoulements sanguins. À partir des images médicales, ces maillages tridimensionnels sont obtenus à l'aide de processus complexes. Ces applications ne prétendront pas reproduire des cas physiologiques à cause des conditions initiales, des conditions aux limites et des simplifications des modèles physiques que nous considérerons. Elles vont nous permettre de tester les solveurs numériques qui ont été développés dans cette thèse sur des géométries complexes. Nous commencerons par un écoulement de Navier-Stokes dans un réseau veineux, puis nous passerons à la modélisation de la propagation d'une onde de pression à l'aide d'un modèle d'interaction fluide structure.

3.1 Écoulement dans le réseau veineux du cerveau

Le rôle du système veineux cérébral est de drainer le sang de tous les sinus crâniens vers les veines jugulaires [76]. La géométrie complexe de ce réseau veineux est représentée avec la figure 3.1. Elle comporte 29 entrées et 2 sorties. Le maillage a été obtenu à partir de l'imagerie médicale par un procédé d'angiographies à résonnance magnétique, voir [141] pour plus de détails.

Pour cette application, nous considérons que la paroi vasculaire est rigide, nous ne modélisons donc pas l'interaction fluide structure mais uniquement l'écoulement de sang. Le modèle utilisé pour cette simulation est celui de Navier-Stokes incompressible. Les conditions aux limites imposées ne sont pas physiologiques. Sur chacune des entrées, nous imposons une impulsion qui est définie par $\boldsymbol{\sigma}_f \boldsymbol{n}_f = g_{in} \boldsymbol{n}$ avec

$$g_{in} = -0.5 \cdot 10^5 \left(1 - \cos\left(\frac{\pi t}{0.0015}\right)\right), \tag{10.4}$$

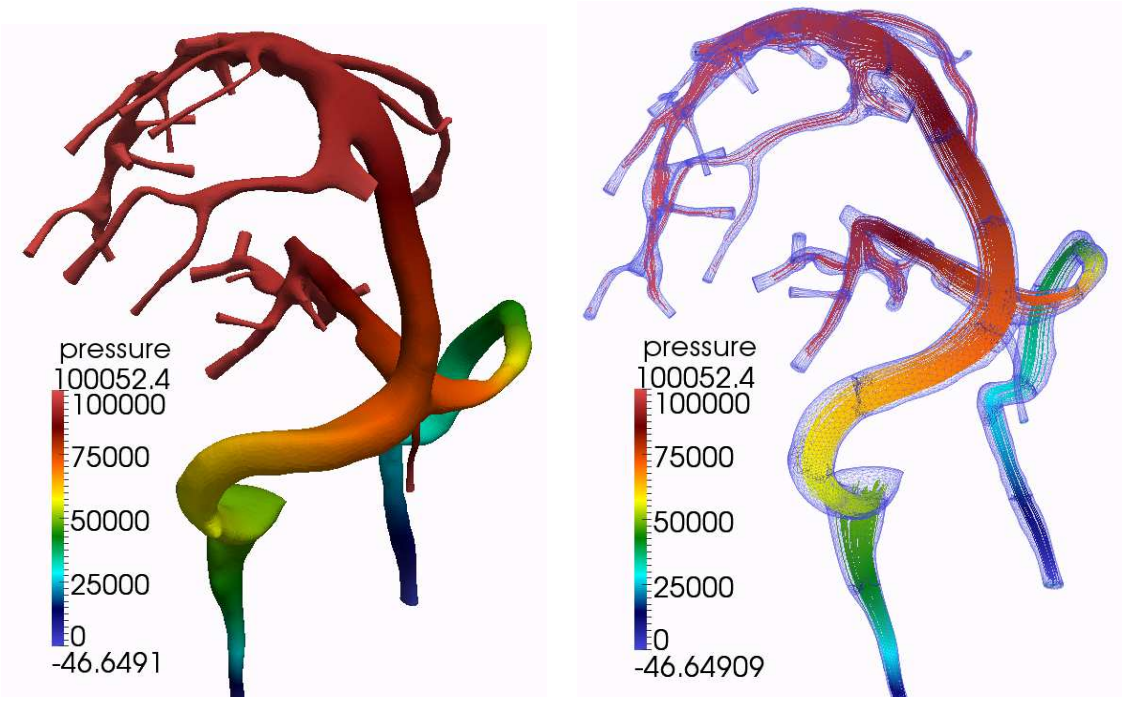
Pour les deux sorties, nous choisissons d'utiliser une condition de sortie libre, c'est-à-dire que $\boldsymbol{\sigma}_f \boldsymbol{n}_f = \mathbf{0}$. Au niveau des paramètres physiques, nous prenons une densité ρ_f égale à 1 kg/m^3 et une viscosité dynamique μ_f égale à 0.003 *poise*. Le pas de temps utilisé est $\Delta t = 10^{-5}$ s et nous effectuons la simulation jusqu'à $t = 0.003$ s. Nous avons utilisé une approximation P2P1 pour le couple vitesse-pression. La table 10.3 nous donne le nombre de tétraèdres présents dans le maillage ainsi que le nombre de degrés de liberté de notre approximation numérique. Cette simulation a été effectuée sur 32 processeurs et le

préconditionneur parallèle utilisé est GASM1 avec un préconditionneur LU dans chaque bloc (voir chapitre 7).

N_{elt}	N_{dof} (vitesse)	N_{dof} (pression)	N_{dof} (total)
237438	1119411	54183	1173594

TABLE 10.3 – Configuration utilisée pour la simulation dans le réseau veineux.

Pour vérifier le bon comportement de notre simulation, nous avons mesuré la somme des débits D_{in} en entrée et la somme des débits D_{out} en sortie. Comme le fluide est incompressible et que le domaine de calcul n'est pas mobile, la quantité de fluide entrant doit être égale à la quantité de fluide sortant, c'est-à-dire que $|D_{in} - D_{out}|$ doit être zéro. Dans notre simulation, cette estimation varie entre 10^{-10} et 10^{-13}. Finalement, la figure 3.1 illustre deux captures d'écran de la solution numérique au temps $t = 0.00151$s. Nous trouvons le champ de pression (figure 10.16(a)) et les lignes de courant (figure 10.16(b)).



(a) Champ de pression (b) Lignes de courant colorées avec la pression

FIGURE 10.16 – Solution numérique obtenue sur le réseau veineux du cerveau au temps $t = 0.00151$s.

3.2 Modèle FSI dans les artères

Nous terminons nos expérimentations sur les écoulements sanguins avec deux applications utilisant un modèle d'interaction fluide structure. Les géométries utilisées proviennent du site internet de GMSH [67]. Nous avons un maillage d'aorte (figure 10.17) et un anévrisme cérébral (figure 10.18). La génération des maillages est faite avec des fonctionnalités disponibles dans GMSH [110].

Pour ces deux simulations, nous voulons modéliser la propagation d'une onde de pression dans ces géométries complexes. Chacune d'elles est composée d'une seule entrée que nous distinguerons par la suite. La condition aux limites imposée pour le fluide en entrée est identique à celle décrite précédemment pour l'application FSI 3D dans le tube cylindrique (section 2). En entrée, nous utilisons donc la relation 10.3. Les autres conditions sont décrites par celles que nous avions notées **BC-2**. Les paramètres physiques (μ_f, ρ_f,...) sont également similaires. Seuls les paramètres du modèle de Windkessel vont changer, ils seront exprimés lors de la présentation des résultats. De plus, les conditions initiales utilisées sont une vitesse et une pression nulles.

Pour chacune de ces applications, nous avons utilisé les modèles simplifiés que nous avons évoqués dans la section 2. Ainsi, nous choisissons le modèle d'Oseen pour le sang et l'élasticité linéaire pour la paroi vasculaire. Au niveau de l'algorithme FSI, c'est le schéma semi-implicite qui sera appliqué. Nous avons également dû utiliser la stabilisation CIP sur les entrées et sorties pour assurer une stabilité numérique. L'approximation numérique employée pour ces calculs est P2P1 pour le fluide et P1 pour le déplacement de la structure. L'ordre géométrique des maillages est d'ordre 1. Ces choix sont motivés par leur performance en contrepartie de la précision. Les ressources informatiques dont nous disposions n'étaient pas suffisantes pour des modèles plus complexes. L'accès à un supercalculateur devrait nous permettre de franchir cette limitation.

Simulation dans l'aorte

Ce premier exemple est basé sur une géométrie de l'aorte, figure 10.17. Nous avons une entrée qui se trouve en bas à gauche et quatre sorties que nous numérotons par ordre « d'apparition » dans l'écoulement (0,1,2 pour les trois sorties en haut de gauche à droite et 3 pour la sortie en bas à droite). Grâce à cette numérotation, nous définissons les paramètres du modèle de Windkessel de chaque sortie avec la table 10.4. Ils ont été pris dans [21]. Toutefois, ces paramètres ne sont pas parfaitement adaptés à notre cas, ils doivent être calibrés à partir de données physiologiques [21].

	Sortie 0	Sortie 1	Sortie 2	Sortie 3
R_p	250	683	615	94
R_d	10^4	$1.296 \cdot 10^4$	$1.1664 \cdot 10^4$	$0.1794 \cdot 10^4$
C_d	$4 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$14 \cdot 10^{-4}$

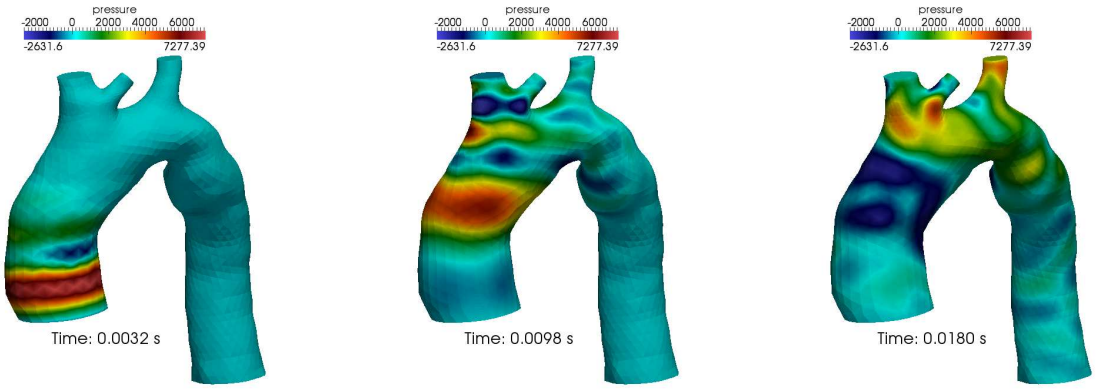
TABLE 10.4 – Paramètres du modèle Windkessel pour la simulation FSI dans l'aorte.

Avec la table 10.5, nous avons des informations sur le nombre d'éléments et de degrés de liberté utilisé pour cette simulation.

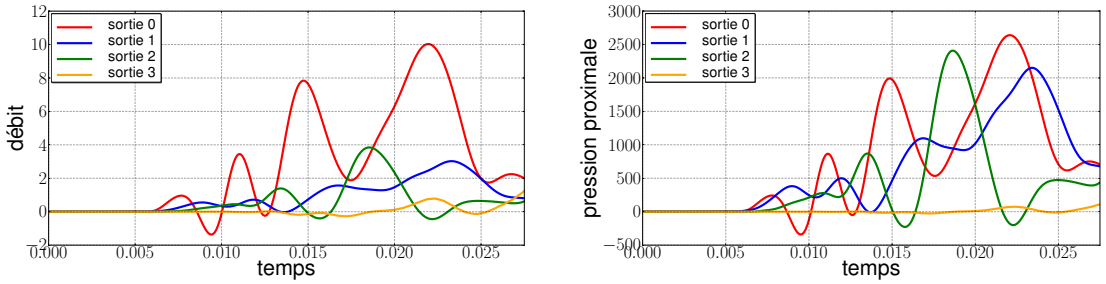
fluide				structure	
N_{elt}	N_{dof} (vitesse)	N_{dof} (pression)	N_{dof} (total)	N_{elt}	N_{dof}
43249	192687	8702	201389	38838	29664

TABLE 10.5 – Configuration utilisée pour la simulation FSI dans l'aorte.

Les résultats numériques obtenus pour cette simulation se trouvent dans la figure 10.17. Nous avons illustré la propagation de l'onde de pression à trois instants avec les figures 10.17(a), 10.17(b) et 10.17(c). Ensuite, nous avons la figure 10.17(d) qui représente la variation du débit de chacune des sorties. Enfin, la figure 10.17(e) montre l'évolution de la valeur de la pression proximale (P_l) à chaque sortie.



(a) t=0.0032 s (b) t=0.0098 s (c) t=0.0180 s



(d) Débits (e) Pression proximale

FIGURE 10.17 – Solutions numériques obtenues pour la simulation FSI dans l'aorte.

Simulation dans une artère contenant un anévrisme cérébral

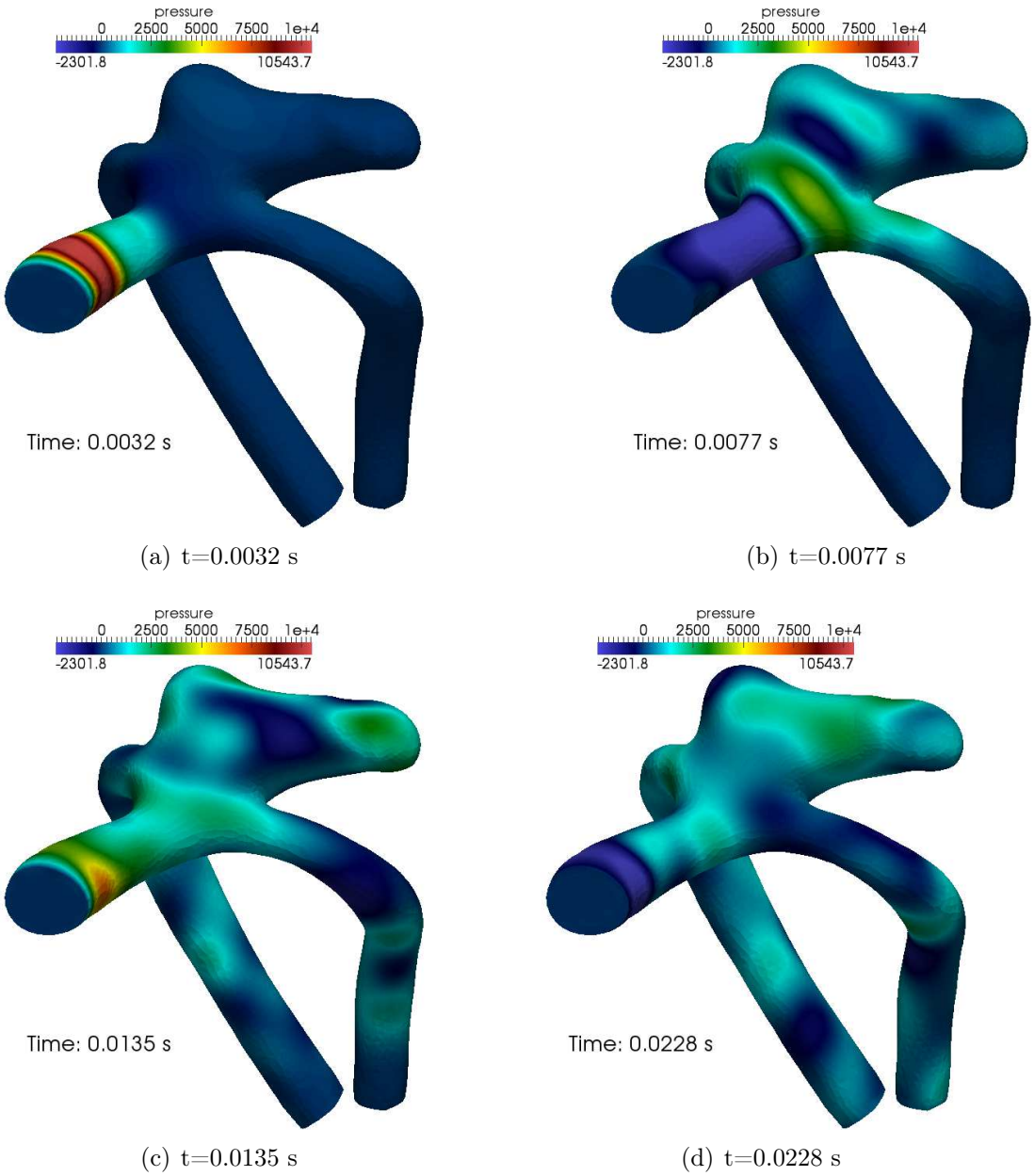
Notre dernière simulation représente la propagation d'une onde de pression dans un anévrisme cérébral. Nous avons avec cette géométrie (figure 10.18) une entrée (à gauche) et deux sorties (en bas à droite). Pour les paramètres du modèle de Windkessel, nous avons pris $R_p = 400$, $R_d = 6.2 \cdot 10^3$ et $C_d = 2.72 \cdot 10^{-4}$. Nous montrons avec la table 10.6 quelques informations sur la configuration utilisée.

fluide				structure	
N_{elt}	N_{dof} (vitesse)	N_{dof} (pression)	N_{dof} (total)	N_{elt}	N_{dof}
43278	217848	11111	228959	41751	41946

TABLE 10.6 – Configuration utilisée pour la simulation FSI dans l'anévrisme cérébral.

Finalement, nous avons la figure 10.18 comme résultats numériques. Elle illustre l'évolution de l'onde de pression à plusieurs instants. Cet exemple permet de mettre en évidence

un comportement non physiologique apparaissant à l'entrée (figures 10.18(c) et 10.18(d)). En effet, on constate que l'onde de pression est réfléchie par l'anévrisme et alors une grande partie de celle-ci revient vers l'entrée de l'écoulement. Il faut donc trouver une condition d'entrée plus élaborée pour éviter d'avoir ces perturbations. On peut également dire que le phénomène de masse ajoutée est très présent dans cette application (voir chapitre 6). Il faut en moyenne 200 itérations FSI pour converger.



(a) t=0.0032 s (b) t=0.0077 s

(c) t=0.0135 s (d) t=0.0228 s

FIGURE 10.18 – Solution numérique du champ de pression obtenue à différents instants pour le maillage de l'anévrisme cérébral.

Conclusions et perspectives

Nous venons de présenter dans cette thèse les principaux ingrédients nécessaires pour l'approche de la simulation numérique des écoulements sanguins. À partir de la description de l'hémodynamique et en négligeant certains comportements, nous avons considéré un modèle newtonien pour le sang et un modèle hyperélastique isotrope pour la paroi vasculaire. La prise en compte des globules rouges n'a pas été faite dans nos applications d'hémodynamique. Nous avons toutefois avancé sur cette interaction en proposant une nouvelle approche pour le transport de particules dans un fluide qui est basée sur la méthode de la frontière élargie. Pour la résolution des équations aux dérivées partielles, nous avons utilisé la méthode des éléments finis. L'implémentation informatique de nos simulations a été faite avec la librairie Feel++.

Tout d'abord, nous avons décrit les méthodes numériques de base utilisées pour résoudre les problèmes physiques (fluide ou structure). Ces méthodes ont été choisies pour leurs optimalités théoriques au sens de la méthode des éléments finis. Nous avons décrit l'implémentation de l'opérateur d'interpolation de Lagrange pour le transfert d'informations entre des domaines de calcul. Nous avons aussi retrouvé numériquement les propriétés théoriques de cet outil. Pour employer des équations aux dérivées partielles dans des domaines mobiles, nous avons choisi d'utiliser la méthode ALE. Cette méthode précise permet de déplacer un maillage par le mouvement de ces bords. La représentation du domaine mobile est alors faite par le calcul de la carte ALE. Nous avons proposé une nouvelle construction de la carte ALE pour des géométries d'ordre élevé dans le but d'obtenir une meilleure précision du déplacement aux bords. Nous avons montré théoriquement et numériquement que cette construction vérifiait des propriétés d'optimalité sur le déplacement du bord d'ordre élevé. Nous avons ensuite étudié deux formulations de la méthode de la frontière élargie. Elles permettent la résolution efficace et précise de problèmes définis dans des domaines perforés. On transforme alors ce problème en plusieurs sous-problèmes définis sur des géométries plus simples. Pour la première formulation, l'optimalité de la méthode a pu être retrouvée, sauf pour les approximations polynomiales d'ordre élevé où la convergence semble atteindre un seuil. La nouvelle formulation proposée dans cette thèse contient des termes non standards et cela a nécessité un conséquent développement dans la librairie Feel++ pour leur prise en compte. L'avantage de cette nouvelle formulation est qu'elle s'applique plus naturellement aux modèles de mécanique des fluides. Par contre, l'analyse du comportement numérique de cette nouvelle méthode n'a pas abouti à une vérification de l'optimalité que suppose la théorie (en cours). Nous avons toutefois montré une validation du modèle à l'aide d'une simulation d'écoulement d'un fluide autour d'un obstacle sphérique.

Dans la deuxième partie, nous avons présenté en détail les modèles d'interaction fluide structure. Nous avons commencé par la description des équations de Navier-Stokes incom-

pressibles dans un domaine mobile. À partir des équations classiques des fluides newtoniens, nous avons obtenu un modèle de fluide en domaine mobile en utilisant la méthode ALE. Après la discrétisation en espace et en temps, nous avons pu vérifier notre solveur en effectuant un test de *Geometric Conservation Law* (GCL). Ensuite, nous avons exposé le modèle des dynamiques des solides. Nous avons présenté les modèles hyperélastiques isotropes compressible et incompressible. La discrétisation en temps a été réalisée avec un schéma de Newmark. Pour la vérification de notre solveur structure, nous avons effectué des applications classiques en mécanique des solides. Cependant, le contrôle de nos résultats a été fait uniquement de manière qualitative. Nous avons ensuite présenté notre stratégie d'interaction fluide structure. Celle-ci est basée sur une méthode partitionnée, c'est-à-dire que les problèmes fluides et structures sont résolus séparément à l'aide d'une méthode itérative appelée Dirichlet-Neumann. Plusieurs « benchmarks » ont été réalisés pour vérifier le comportement de ce modèle complexe. Avec ces applications, nous avons pu montrer l'intérêt de l'utilisation de l'ordre élevé en espace et géométrie dans la précision obtenue sur des maillages relativement « grossiers ». Nous avons également présenté un nouveau modèle pour le déplacement d'une particule déformable dans un fluide. Cette méthode est basée sur la deuxième formulation de la méthode de la frontière élargie. Nous avons ensuite illustré les premières expérimentations de ce modèle d'interaction fluide-particules.

Nous avons ensuite présenté une troisième partie orientée dans le développement informatique. En effet, une étape importante de cette thèse a été la parallélisation de la librairie Feel++. Cette étape a nécessité un important travail de développement, car cet outil était totalement séquentiel au début de la thèse. Pour pouvoir réaliser une parallélisation « assez » rapide et efficace, nous avons utilisé deux librairies, Gmsh pour le partitionnement et PETSc pour les structures et solveurs algébriques parallèles. La principale tâche de notre travail a consisté à la création d'une table des degrés de liberté parallèle à partir du partitionnement du maillage. Une fois construite, cette table permet de construire les structures algébriques gérées par PETSc. Nous avons ensuite effectué plusieurs tests de scalabilité sur des problèmes de mécanique des fluides et des solides avec un nombre de coeurs de calcul relativement petit (≤ 32). Les résultats obtenus pour la scalabilité dite forte sont assez bons et encourageants pour le passage à un grand nombre de coeurs. Par contre, la scalabilité dite faible a montré les faiblesses de notre solveur algébrique parallèle qui est basé sur une méthode de décomposition de domaine algébrique de type Schwarz additif avec recouvrement. L'augmentation du nombre de coeurs entraine une augmentation conséquente du nombre d'itérations du solveur et donc une chute de l'efficacité mesurée. Nous avons ensuite montré une utilisation possible du nouveau paradigme de calcul parallèle, le GPGPU, dans notre librairie de calcul. L'accélération potentielle que nous avons présentée est située dans nos méthodes numériques au niveau de l'assemblage local. On a alors montré que l'utilisation du GPGPU devenait intéressante pour des approximations polynomiales d'ordre élevé.

La dernière partie de cette thèse a été consacrée à la simulation numérique des écoulements sanguins. Ces applications modélisaient principalement la propagation d'une onde de pression dans des artères. Toutefois, le réalisme de ces simulations restait limité à cause des conditions aux limites et des données physiologiques utilisées. Ces simulations nous ont permis d'expérimenter presque tous les ingrédients qui ont été développés dans cette thèse : interpolation, modèles physiques, algorithmes d'interaction fluide structure et calcul parallèle. Nous avons pu montrer également les capacités de nos solveurs en trai-

tant des géométries complexes. Notamment, la simulation dans le réseau veineux cérébral a été possible grâce au parallélisme de FEEL++. Nous avons aussi montré des applications d'interaction fluide structure sur des géométries réalistes. Elles ont permis de vérifier l'implémentation non triviale de l'opérateur d'interpolation parallèle.

Pour la suite de ce projet, de nombreuses perspectives peuvent être envisagées. Tout d'abord, l'algorithme d'interaction fluide structure basé sur une méthode partitionnée à montrer ces faiblesses pour les géométries réalistes. Ce phénomène connu pour les applications d'hémodynamique est appelé l'effet de masse ajouté (chapitre 6). Il cause une forte augmentation du nombre d'itérations nécessaires à la convergence de l'algorithme itératif de résolution. L'utilisation d'un algorithme monolithique [83, 66, 57] semble être un choix judicieux.

Nous avons présenté les premiers résultats d'une nouvelle formulation de la frontière élargie pour la modélisation de l'interaction fluide-particules. De nombreux aspects théoriques et numériques sont à voir, à revoir et à étendre pour pouvoir tenter de modéliser la présence de globules rouges dans le sang. Ce travail en cours devrait permettre de publier prochainement les articles suivants [23, 24, 22]. La réussite de ce travail devrait permettre d'envisager la simulation d'un écoulement contenant une suspension de particule. De plus, nous voulons étendre notre travail à l'élaboration des modèles plus complexes pour représenter les globules rouges (membrane élastique + cytoplasme). Une comparaison avec d'autres méthodes comme les méthodes *level-set* [41] pourrait être ensuite effectuée.

L'utilisation de nouvelles conditions aux limites pour les écoulements sanguins est un travail indispensable à faire pour le réalisme de nos simulations. Par exemple, l'utilisation d'un couplage fort 3D-1D [121, 59, 123, 60] pourrait être une solution. Une autre approche [127] peut aussi être envisagée en se basant sur un couplage faible à l'aide d'une réduction variationnelle des modèles physiques. L'utilisation des modèles de sang non newtoniens est aussi une perspective majeure qu'il faudra considérée par la suite. Le recueil de géométries et de données physiologiques réelles est également un besoin important qui doit être envisagé pour la modélisation réaliste des écoulements sanguins. La thèse [149] qui est en cours s'intéresse particulièrement à ces aspects en se basant sur les outils numériques qui ont été développés dans ce manuscrit.

Enfin, nous devons expérimenter nos solveurs parallèles sur des machines disposant d'une grande puissance de calcul (>1000 coeurs). Ces tests devraient ainsi mettre en évidence un certain nombre de faiblesses à corriger ou à optimiser. Le calcul haute performance est un ingrédient indispensable à la réussite de ce projet. À l'aide des avancées que nous avons réalisées en matière de HPC, nous avons alors proposé un projet nommé HP-Feel++ pour le 6$^{\text{ème}}$ appel *Prace* (http ://www.prace-ri.eu/). Nous venons d'apprendre très récemment que ce projet a été sélectionné et qu'il a obtenu 60 millions d'heures de calcul sur le supercalculateur SUPERMUC en Allemagne.

Annexes

Annexe A

Scalabilités en mécanique des fluides

Dans cette annexe, nous allons compléter les résultats de scalabilité du modèle de Stokes que nous avons présentés dans la section 2.3 du chapitre 7. Nous rappelons que cette application consiste à simuler un écoulement de fluide dans un cylindre de longueur L_c et de rayon 0.5. Nous effectuons alors dans cette partie des tests de scalabilité forte et faible basés cette fois-ci sur une approximation P3P2 et une géométrie d'ordre 2. Pour chacune des scalabilités fortes et faibles, nous effectuons les mêmes types de mesure : temps d'assemblage, temps de résolution du système algébrique et nombre d'itérations du solveur.

Nous commençons par présenter les configurations utilisées avec la table A.1 pour la scalabilité forte et la table A.2 pour la scalabilité faible.

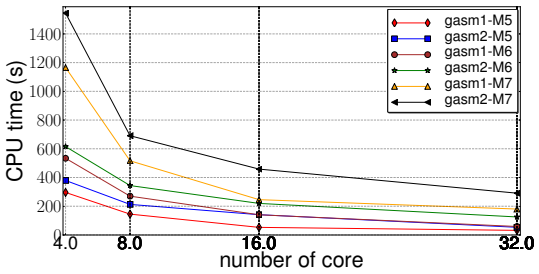
config	L_c	N_{elt}	N_{dof}
M5	6	30702	515938
M6	6	44227	736002
M7	6	67566	1110569

TABLE A.1 – Configurations utilisées pour la scalabilité forte du modèle de Stokes avec une approximation P3P2 et une géométrie d'ordre 2.

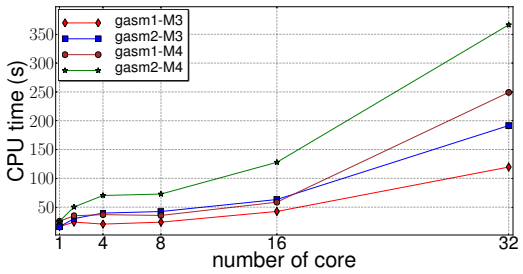
nProc	L_c	config	N_{dof}	config	N_{dof}
32	16	M3	754015	M4	984581
16	8	M3	383423	M4	489214
8	4	M3	197431	M4	252665
4	2	M3	103055	M4	141432
2	1	M3	60303	M4	77396
1	0.5	M3	40444	M4	50568

TABLE A.2 – Configurations utilisées pour la scalabilité faible du modèle de Stokes avec une approximation P3P2 et une géométrie d'ordre 2.

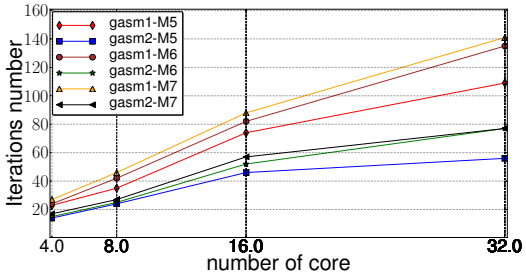
Nous trouvons ensuite les résultats obtenus pour ces deux types de scalabilité avec les figures qui suivent. Globalement, nous avons un comportement qui est un peu moins bon que pour le cas P2P1G1, mais les performances restent toutefois correctes.



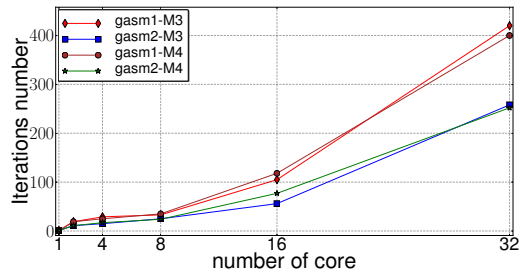
(a) Temps CPU pour la résolution du système algébrique pour la scalabilité forte



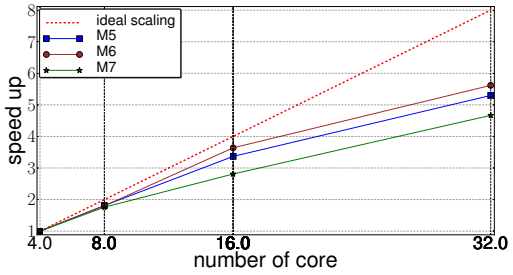
(b) Temps CPU pour la résolution du système algébrique pour la scalabilité faible



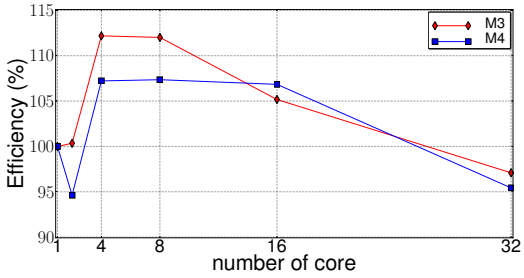
(c) Nombre d'itération du solveur pour la scalabilité forte



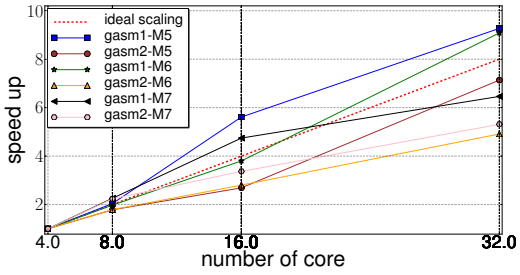
(d) Nombre d'itération du solveur pour la scalabilité faible



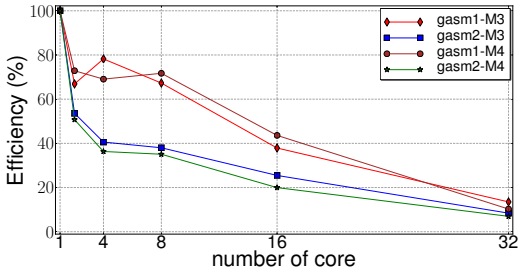
(e) Speed-up du temps d'assemblage pour la scalabilité forte



(f) Efficacité du temps d'assemblage pour la scalabilité faible

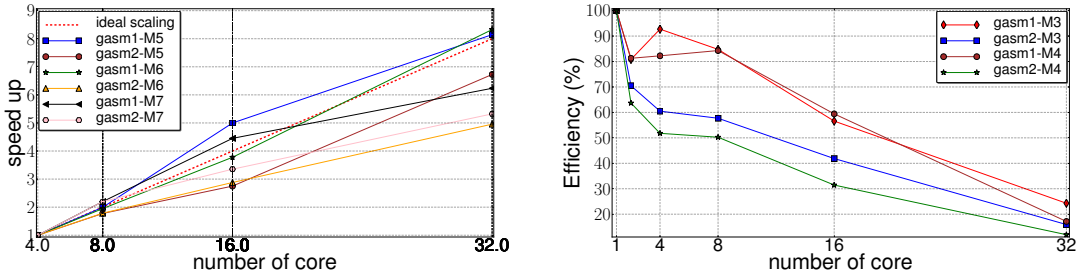


(g) Speed-up du demps de résolution pour la scalabilité forte



(h) Efficacité du temps de résolution pour la scalabilité faible

FIGURE A.1 – Tests de scalabilité pour P3P2G2



(a) Speed-up du temps d'assemblage + résolution pour la scalabilité forte

(b) Efficacité du temps d'assemblage + résolution pour la scalabilité faible

FIGURE A.2 – Tests de scalabilité pour P3P2G2

Annexe B

Scalabilités sur GPU

Nous allons dans cette annexe compléter les résultats de scalabilité de la partie appelée assemblage local sur GPU (section 3.1 du chapitre 7). Avant de donner les résultats, nous rappelons brièvement les principaux aspects de l'assemblage local. Nous considérons les deux intégrales suivantes qui vont respectivement représenter la matrice de masse et de raideur :

$$\int_{\Omega} u\left(\mathbf{x}\right) v\left(\mathbf{x}\right) \quad , \quad \int_{\Omega} \nabla u\left(\mathbf{x}\right) \cdot \nabla v\left(\mathbf{x}\right) \tag{B.1}$$

avec $u \in H^1\left(\Omega\right)$ et $v \in H^1\left(\Omega\right)$. Après discrétisation, nous pouvons décomposer ces intégrales sur tous les éléments du maillage. Le calcul des matrices élémentaires permet alors d'obtenir les contributions apportées par chaque élément. On a donc N_{elt} matrices élémentaires à construire. Chacune de ces matrices est de taille $(T_N \times T_N)$, où T_N est le nombre de degrés de liberté de l'élément fini.

Calcul de la matrice de masse élémentaire On pose $\left\{(\hat{\mathbf{x}}_q^m, w_q^m), 1 \leq \ q \leq N_q^m\right\}$ l'ensemble définissant une formule de quadrature pour le terme de masse. Il contient les couples points de quadrature et poids de quadrature. L'expression des entrées $(i,j)_{1 \leq i,j \leq T_N}$ de la la matrice élémentaire M_K, $\forall K \in \mathcal{T}_\delta$ est donnée par :

$$M_K\left(i,j\right) \ = \ \sum_{q=1}^{N_q} w_q^m \ \hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q^m\right) \hat{\Phi}_j^N\left(\hat{\mathbf{x}}_q^m\right) \left|J_K\left(\hat{\mathbf{x}}_q^m\right)\right|$$

Calcul de la matrice de raideur élémentaire On pose $\left\{(\hat{\mathbf{x}}_q^r, w_q^r), 1 \leq \ q \leq N_q^r\right\}$ l'ensemble définissant une formule de quadrature pour le terme de raideur. L'expression des entrées $(i,j)_{1 \leq i,j \leq T_N}$ de la la matrice élémentaire R_K, $\forall K \in \mathcal{T}_\delta$ est donnée par :

$$R_K\left(i,j\right) \ = \ \sum_{q=1}^{\tilde{N}_q} w_q^r \ \left[\boldsymbol{B}_K\left(\hat{\mathbf{x}}_q^r\right) \ \nabla_{\hat{\mathbf{x}}} \hat{\Phi}_i^N\left(\hat{\mathbf{x}}_q\right)\right] \cdot \left[\boldsymbol{B}_K\left(\hat{\mathbf{x}}_q^r\right) \ \nabla_{\hat{\mathbf{x}}} \hat{\Phi}_j^N\left(\hat{\mathbf{x}}_q^r\right)\right] \left|J_K\left(\hat{\mathbf{x}}_q^r\right)\right|$$

Matrice de Masse :

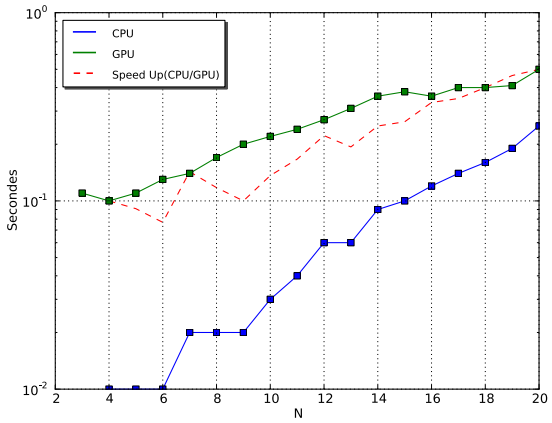


FIGURE B.1 – Influence du degré d'interpolation avec d=1, Nel=10000

N	CPU	GPU	Speed Up
3	0	0.11	0
4	0.01	0.1	0.1
5	0.01	0.11	0.0909091
6	0.01	0.13	0.0769231
7	0.02	0.14	0.142857
8	0.02	0.17	0.117647
9	0.02	0.2	0.1
10	0.03	0.22	0.136364
11	0.04	0.24	0.166667
12	0.06	0.27	0.222222
13	0.06	0.31	0.193548
14	0.09	0.36	0.25
15	0.1	0.38	0.263158
16	0.12	0.36	0.333333
17	0.14	0.4	0.35
18	0.16	0.4	0.4
19	0.19	0.41	0.463415
20	0.25	0.5	0.5

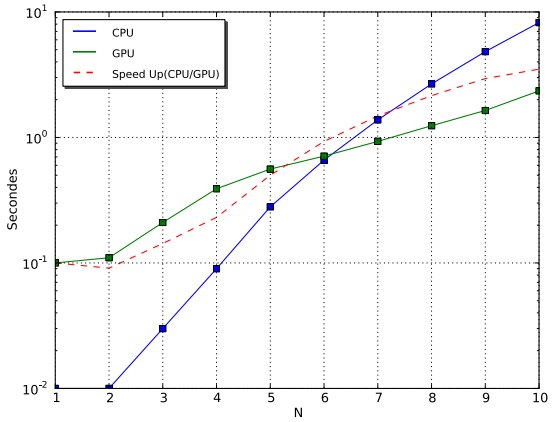


FIGURE B.2 – Influence du degré d'interpolation avec d=2, Nel=10000

N	CPU	GPU	Speed Up
1	0.01	0.1	0.1
2	0.01	0.11	0.0909091
3	0.03	0.21	0.142857
4	0.09	0.39	0.230769
5	0.28	0.56	0.5
6	0.66	0.71	0.929577
7	1.38	0.93	1.48387
8	2.67	1.24	2.15323
9	4.83	1.64	2.94512
10	8.23	2.35	3.50213

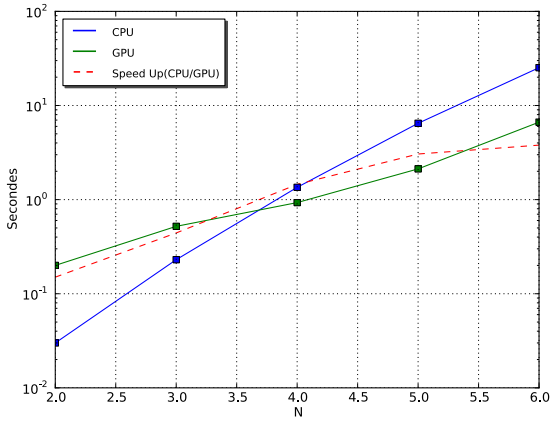


FIGURE B.3 – Influence du degré d'interpolation avec d=3, Nel=10000

N	CPU	GPU	Speed Up
2	0.03	0.2	0.15
3	0.23	0.52	0.442308
4	1.35	0.93	1.45161
5	6.47	2.12	3.05189
6	25.22	6.65	3.79248

Matrice de Raideur :

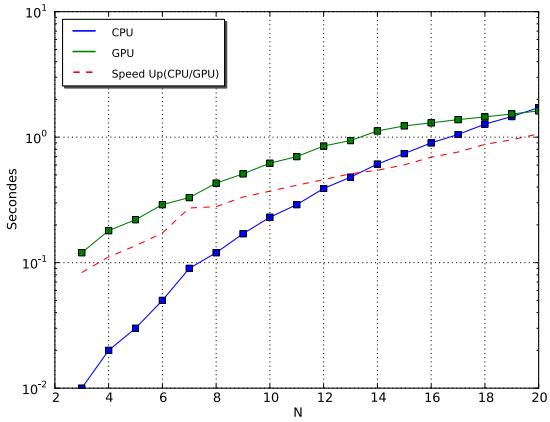


FIGURE B.4 – Influence du degré d'interpolation avec d=1, Nel=10000

N	CPU	GPU	Speed Up
3	0.01	0.12	0.0833333
4	0.02	0.18	0.111111
5	0.03	0.22	0.136364
6	0.05	0.29	0.172414
7	0.09	0.33	0.272727
8	0.12	0.43	0.27907
9	0.17	0.51	0.333333
10	0.23	0.62	0.370968
11	0.29	0.7	0.414286
12	0.39	0.85	0.458824
13	0.48	0.94	0.510638
14	0.61	1.12	0.544643
15	0.74	1.23	0.601626
16	0.9	1.3	0.692308
17	1.05	1.38	0.76087
18	1.27	1.45	0.875862
19	1.46	1.53	0.954248
20	1.72	1.62	1.06173

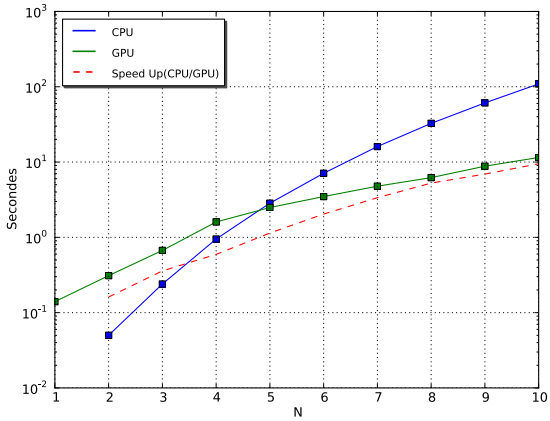


FIGURE B.5 – Influence du degré d'interpolation avec d=2, Nel=10000

N	CPU	GPU	Speed Up
1	0	0.14	0
2	0.05	0.31	0.16129
3	0.24	0.67	0.358209
4	0.95	1.6	0.59375
5	2.84	2.49	1.14056
6	7.09	3.48	2.03736
7	16.07	4.78	3.36192
8	32.63	6.21	5.25443
9	61	8.81	6.92395
10	109.09	11.5	9.48609

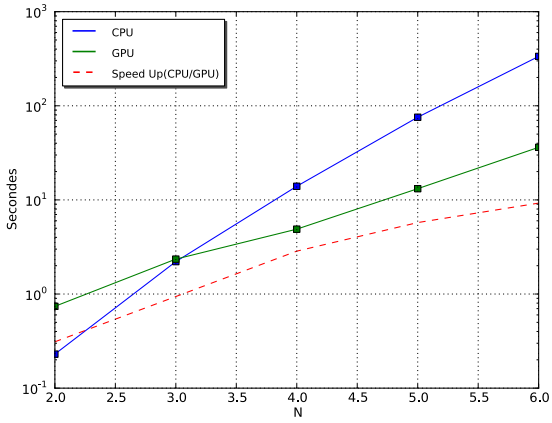


FIGURE B.6 – Influence du degré d'interpolation avec d=3, Nel=10000

N	CPU	GPU	Speed Up
2	0.23	0.74	0.310811
3	2.21	2.35	0.940426
4	13.96	4.89	2.85481
5	75.59	13.16	5.74392
6	334.79	36.2	9.24834

Matrice de Masse et Raideur :

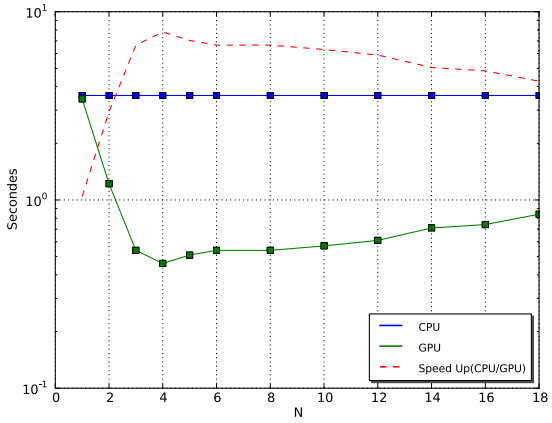


FIGURE B.7 – Influence de la taille des blocs avec d=2, N=8, Nel=1000

Taille	CPU	GPU	Speed Up
2	3.59	1.22	2.94262
3	3.59	0.54	6.64815
4	3.59	0.46	7.80435
5	3.59	0.51	7.03922
6	3.59	0.54	6.6481
8	3.59	0.54	6.6481
10	3.59	0.57	6.2982
12	3.59	0.61	5.88525
14	3.59	0.71	5.05634
16	3.59	0.74	4.85135
18	3.59	0.84	4.27381

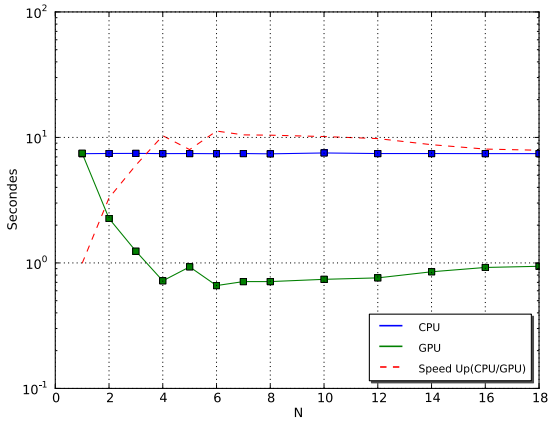


FIGURE B.8 – Influence de la taille des blocs avec d=3, N=6, Nel=200

Taille	CPU	GPU	Speed Up
2	7.45	2.26	3.2964
3	7.46	1.24	6.01613
4	7.43	0.72	10.3194
5	7.44	0.93	8.0
6	7.42	0.66	11.2424
7	7.44	0.71	10.4789
8	7.4	0.71	10.4225
10	7.52	0.74	10.1622
12	7.44	0.76	9.7894
14	7.44	0.85	8.75294
16	7.43	0.92	8.07609
18	7.43	0.94	7.90426

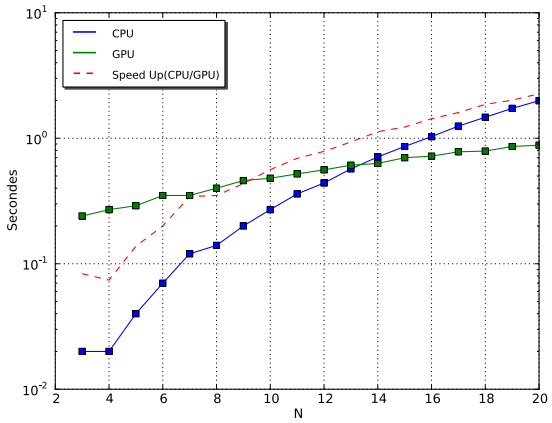


FIGURE B.9 – Influence du degré d'interpolation avec d=1, Nel=10000, TailleBloc=3

N	CPU	GPU	Speed Up
3	0.02	0.24	0.0833333
4	0.02	0.27	0.0740741
5	0.04	0.29	0.137931
6	0.07	0.35	0.2
7	0.12	0.35	0.342857
8	0.14	0.4	0.35
9	0.2	0.46	0.434783
10	0.27	0.48	0.5625
11	0.36	0.52	0.692308
12	0.44	0.56	0.785714
13	0.57	0.61	0.934426
14	0.71	0.63	1.12698
15	0.86	0.7	1.22857
16	1.03	0.72	1.43056
17	1.25	0.78	1.60256
18	1.47	0.79	1.86076
19	1.73	0.86	2.01163
20	1.99	0.88	2.26136

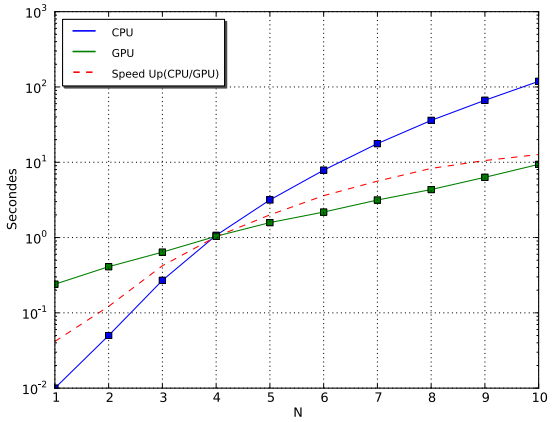


FIGURE B.10 – Influence du degré d'interpolation avec d=2, Nel=10000, TailleBloc=6

N	CPU	GPU	Speed Up
1	0.01	0.24	0.0416667
2	0.05	0.41	0.121951
3	0.27	0.64	0.421875
4	1.07	1.04	1.02885
5	3.16	1.58	2
6	7.85	2.18	3.60092
7	17.68	3.15	5.6127
8	35.95	4.34	8.28341
9	66.53	6.31	10.5436
10	118.71	9.37	12.6692

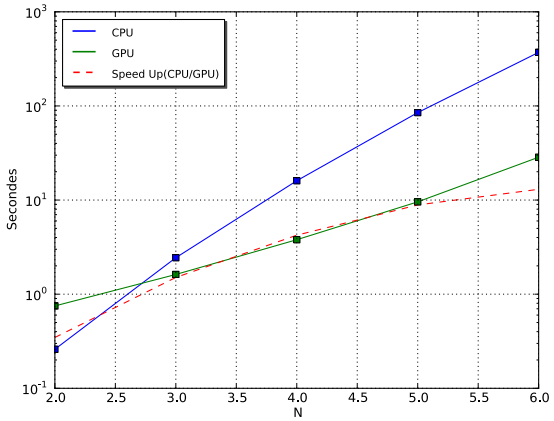


FIGURE B.11 – Influence du degré d'interpolation avec d=3, Nel=10000, TailleBloc=6

N	CPU	GPU	Speed Up
2	0.26	0.75	0.346667
3	2.44	1.62	1.50617
4	16.04	3.79	4.23219
5	84.97	9.57	8.87879
6	371.06	28.52	13.0105

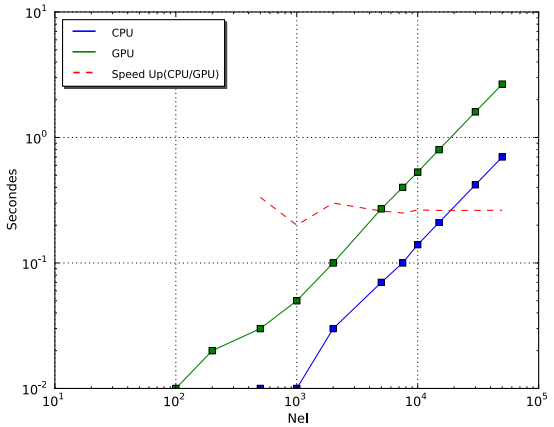
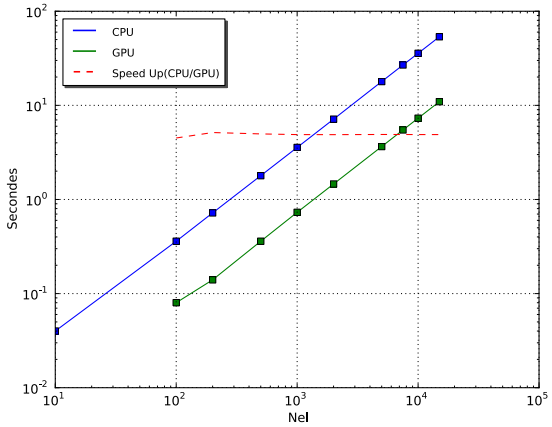


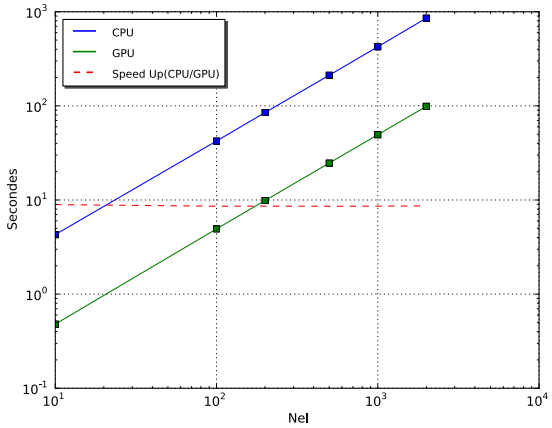
FIGURE B.12 – Influence du nombre d'élément avec d=1, N=8

Nel	CPU	GPU	Speed Up
10	0	0	nan
100	0	0.01	0
200	0	0.02	0
500	0.01	0.03	0.333333
1000	0.01	0.05	0.2
2000	0.03	0.1	0.3
5000	0.07	0.27	0.259259
7500	0.1	0.4	0.25
10000	0.14	0.53	0.264151
15000	0.21	0.8	0.2625
30000	0.42	1.6	0.2625
50000	0.7	2.66	0.263158



Nel	CPU	GPU	Speed Up
10	0.04	0	inf
100	0.36	0.08	4.5
200	0.72	0.14	5.14286
500	1.79	0.36	4.97222
1000	3.57	0.73	4.89041
2000	7.13	1.46	4.88356
5000	17.9	3.65	4.90411
7500	26.86	5.48	4.90146
10000	35.72	7.3	4.89315
15000	53.57	10.95	4.89224

FIGURE B.13 – Influence du nombre d'élément avec d=2, N=8



Nel	CPU	GPU	Speed Up
10	4.29	0.48	8.9375
100	42.33	4.93	8.58621
200	85	9.88	8.60324
500	211.62	24.67	8.57803
1000	424.52	49.38	8.597
2000	855.91	98.8	8.66306

FIGURE B.14 – Influence du nombre d'élément avec d=3, N=8

Bibliographie

[1] *Encyclopédie familiale de la santé : Comprendre, prévenir, soigner*. Québec Amérique, 2010.

[2] D. Abrahams and D. Wallin. Boost.parameter library, 2005.

[3] Yves Achdou and Jean-Luc Guermond. Convergence analysis of a finite element projection/lagrange–galerkin method for the incompressible navier–stokes equations. *SIAM Journal on Numerical Analysis*, 37(3) :799–826, 2000.

[4] Nazmiye Acikgoz and Carlo L. Bottasso. A unified approach to the deformation of simplicial and non-simplicial meshes in two and three dimensions with guaranteed validity. *Computers & Structures*, 85(11–14) :944 – 954, 2007. <ce :title>Fourth {MIT} Conference on Computational Fluid and Solid Mechanics</ce :title>.

[5] P.R. Amestoy, I.S. Duff, J.Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1) :15–41, 2001.

[6] P.R. Amestoy, A. Guermouche, J.Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel computing*, 32(2) :136–156, 2006.

[7] Harm Askes, Ellen Kuhl, and Paul Steinmann. An ale formulation based on spatial and material settings of continuum mechanics. part 2 : Classification and applications. *Computer Methods in Applied Mechanics and Engineering*, 193(39â€“41) :4223 – 4245, 2004.

[8] G. Astarita and G. Marrucci. *Principles of non-Newtonian fluid mechanics*, volume 28. McGraw-Hill New York, 1974.

[9] Matteo Astorino and Céline Grandmont. Convergence analysis of a projection semi-implicit coupling scheme for fluid–structure interaction problems. *Numerische Mathematik*, 116(4) :721–767, 2010.

[10] Santiago Badia, Fabio Nobile, and Christian Vergara. Robin–robin preconditioned krylov methods for fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 198(33–36) :2768 – 2784, 2009.

[11] Santiago Badia, Annalisa Quaini, and Alfio Quarteroni. Modular vs. non-modular preconditioners for fluid-structure systems with large added-mass effect. *Computer Methods in Applied Mechanics and Engineering*, 197(49â€“50) :4216 – 4232, 2008.

[12] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L.C. McInnes, B. Smith, and H. Zhang. Petsc users manual revision 3.3. 2012.

[13] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2011. http ://www.mcs.anl.gov/petsc.

[14] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[15] Galina Baltgaile. Arterial wall dynamics. *Perspectives in Medicine*, 1(1-12) :146 – 151, 2012.

[16] M. Bathe, RD Kamm, et al. A fluid–structure interaction finite element analysis of pulsatile blood flow through a compliant stenotic artery. *Journal of biomechanical engineering*, 121(4) :361, 1999.

[17] A.J. Bax and R.A. Schachar. Maximisation of the design of a scleral expansion band segment for the reversal of presbyopia. In *ANSYS Conference and Exhibition*, 2000.

[18] GustavG. Belz. Elastic properties and windkessel function of the human aorta. *Cardiovascular Drugs and Therapy*, 9 :73–83, 1995.

[19] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9) :509–517, 1975.

[20] Jon Louis Bentley. Multidimensional binary search trees in database applications. *Software Engineering, IEEE Transactions on*, (4) :333–340, 1979.

[21] C. Bertoglio. *Problèmes Directs et Inverses en Interaction Fluide-Structure. Application à l'hémodynamique*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2012.

[22] S. Bertoluzza, M. Ismail, V. Chabannes, and C. Prud'homme. Saddle point formulation for the fat boundary method : fluid structure interaction case. 2013.

[23] S. Bertoluzza, M. Ismail, V. Chabannes, and C. Prud'homme. Saddle point formulation for the fat boundary method : laplacian case. 2013.

[24] S. Bertoluzza, M. Ismail, V. Chabannes, and C. Prud'homme. Saddle point formulation for the fat boundary method : Stokes case. 2013.

[25] Silvia Bertoluzza, Mourad Ismail, and Bertrand Maury. The fat boundary method : Semi-discrete scheme and some numerical experiments. In *Domain Decomposition Methods in Science and Engineering*, volume 40 of *Lecture Notes in Computational Science and Engineering*, pages 513–520. Springer Berlin Heidelberg, 2005.

[26] Silvia Bertoluzza, Mourad Ismail, and Bertrand Maury. Analysis of the fully discrete fat boundary method. *Numerische Mathematik*, 118 :49–77, 2011.

[27] J. Bonet and R.D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.

[28] C++ Boost. Libraries, 2008. http ://www.boost.org.

[29] Carlo L. Bottasso, Davide Detomi, and Roberto Serra. The ball-vertex method : a new simple spring analogy method for unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering*, 194(39–41) :4244 – 4264, 2005.

[30] M. Breuer, G. De Nayer, M. Münsch, T. Gallinger, and R. Wüchner. Fluid–structure interaction using a partitioned semi-implicit predictor–corrector coupling scheme for the application of large-eddy simulation. *Journal of Fluids and Structures*, 2012.

[31] F.(Franco) Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, 1991.

[32] E. Burman and M.A. Fernández. Continuous interior penalty finite element method for the time-dependent Navier–Stokes equations : space discretization and convergence. *Numerische Mathematik*, 107(1) :39–77, 2007.

[33] E. Burman and M.A. Fernández. Stabilization of explicit coupling in fluid–structure interaction involving fluid incompressibility. *Computer Methods in Applied Mechanics and Engineering*, 198(5) :766–784, 2009.

[34] Céline Caldini Queiros, Vincent Chabannes, Mourad Ismail, Gonçalo Pena, Christophe Prud'Homme, Marcela SZOPOS, and Ranine Tarabay. Towards large-scale three-dimensional blood flow simulations in realistic geometries. pp. 17, Submitted to ESAIM : Proc, CEMRACS 2012, 2012-12.

[35] C. Canuto, MY Hussaini, A. Quarteroni, and TA Zang. *Spectral methods*. Springer, 2006.

[36] C.J. Carmody, G. Burriesci, I.C. Howard, and E.A. Patterson. An approach to the simulation of fluid-structure interaction in the aortic valve. *Journal of Biomechanics*, 39(1) :158 – 169, 2006.

[37] P. Causin, J.F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid–structure problems. *Computer methods in applied mechanics and engineering*, 194(42) :4506–4527, 2005.

[38] Vincent Chabannes, Gonçalo Pena, and Christophe Prud'homme. High-order fluide-structure interaction in 2d and 3d application to blood flow in arteries. *Journal of Computational and Applied Mathematics*, 246(0) :1 – 9, 2013.

[39] P.G. Ciarlet. The finite element method for elliptic problems, vol. 40 of classics in applied mathematics, society for industrial and applied mathematics (siam), philadelphia, pa, 2002. reprint of the 1978 original. *Reprint of the 1978 original*, 1(1) :2–4.

[40] R. Codina, G. Houzeaux, H. Coppola-Owen, and J. Baiges. The fixed-mesh ale approach for the numerical approximation of flows in moving domains. *Journal of Computational Physics*, 228(5) :1591–1611, 2009.

[41] Georges-Henri Cottet and Emmanuel Maitre. A level set method for fluid-structure interactions with immersed surfaces. *Mathematical models and methods in applied sciences*, 16(03) :415–438, 2006.

[42] Paolo Crosetto, Philippe Reymond, Simone Deparis, Dimitrios Kontaxakis, Nikolaos Stergiopulos, and Alfio Quarteroni. Fluid structure interaction simulation of aortic blood flow. *Computers & Fluids*, 43(1) :46 – 57, 2011.

[43] A. Curnier. *Méthodes numériques en mécanique des solides*. PPUR presses polytechniques, 1993.

[44] Cécile Daversin, Stéphane Veys, Christophe Trophime, and Christophe Prud'Homme. A Reduced Basis Framework : Application to large scale non-linear multi-physics problems. pp 26, Submitted to ESAIM : Proc, CEMRACS 2012, 2012.

[45] V. Doyeux, Y. Guyot, V. Chabannes, C. Prud'homme, and M. Ismail. Simulation of two-fluid flows using a finite element/level set method. application to bubbles and

vesicle dynamics. *Journal of Computational and Applied Mathematics*, 246(0) :251 – 259, 2013.

[46] Vincent Doyeux, Vincent Chabannes, Christophe Prud'Homme, and Mourad Ismail. Simulation of vesicle using level set method solved by high order finite element. *ESAIM : Proceedings*, 38 :335–347, August 2012.

[47] A. DROCHON and R. CHOTARD-GHODSNIA. Le sang et les cellules circulantes : de la rhéologie aux enjeux cliniques : L'eau et le monde vivant. *Houille blanche(Grenoble)*, (4), 2005.

[48] T. Dunne and R. Rannacher. Adaptive finite element approximation of fluid-structure interaction based on an eulerian variational formulation. *Fluid-structure interaction*, pages 110–145, 2006.

[49] G. Duvaut and H. Dumontet. *Mécanique des milieux continus*, volume 34. Masson Paris etc, 1990.

[50] N. El Khatib. *Modélisation mathématique de l'athérosclérose*. PhD thesis, Université Claude Bernard-Lyon I, 2009.

[51] H. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing*, 27(5) :1651–1668, 2006.

[52] H.C. Elman. Multigrid and krylov subspace methods for the discrete stokes equations. 1998.

[53] A. Ern and J.L. Guermond. *Theory and practice of finite elements*, volume 159. Springer Verlag, 2004.

[54] C.R. Ethier and DA Steinman. Exact fully 3d navier–stokes solutions for benchmarking. *International Journal for Numerical Methods in Fluids*, 19(5) :369–375, 1994.

[55] M. Faivre. *Gouttes, Vésicules Et Globules Rouges*. PhD thesis, 2006.

[56] MA Fernández, J-F Gerbeau, and Celine Grandmont. A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid. *International Journal for Numerical Methods in Engineering*, 69(4) :794–821, 2006.

[57] M.Á. Fernández and M. Moubachir. A Newton method using exact jacobians for solving fluid-structure coupling. *Computers & Structures*, 83(2-3) :127–142, 2005.

[58] H. Finney. Finite element analysis. *Engineering with Rubber : How to Design Rubber Components (2nd ed.), Hanser Gardner, Cincinnati, OH*, pages 257–304, 2001.

[59] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics : Modeling and simulation of the circulatory system*. Springer Verlag, 2009.

[60] Luca Formaggia, Jean-Frédéric Gerbeau, Fabio Nobile, and Alfio Quarteroni. On the coupling of 3d and 1d navier–stokes equations for flow problems in compliant vessels. *Computer Methods in Applied Mechanics and Engineering*, 191(6) :561–582, 2001.

[61] Luca Formaggia and Fabio Nobile. A stability analysis for the arbitrary lagrangian eulerian formulation with finite elements. *East West Journal of Numerical Mathematics*, 7 :105–132, 1999.

[62] Luca Formaggia and Fabio Nobile. Stability analysis of second-order time accurate schemes for ale–fem. *Computer methods in applied mechanics and engineering*, 193(39) :4097–4116, 2004.

[63] Luca Formaggia and Christian Vergara. Prescription of general defective boundary conditions in fluid-dynamics. *Milan Journal of Mathematics*, 80(2) :333–350, 2012.

[64] T.G. Gallinger. *Effiziente Algorithmen zur partitionierten Lösung stark gekoppelter Probleme der Fluid-Struktur-Wechselwirkung*. Shaker, 2010.

[65] William Ganong. *Physiologie médicale*. De Boeck Supérieur, 2005.

[66] J.F. Gerbeau, M. Vidrascu, et al. A quasi-newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows. 2003.

[67] C. Geuzaine and J.-F. Remacle. Gmsh : a three-dimensional finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11) :1309–1331, 2009.

[68] Fabian Gieseke, Gabriel Moruz, and Jan Vahrenhold. Resilient kd trees : K-means in space revisited. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 815–820. IEEE, 2010.

[69] F.J.H. Gijsen, E. Allanic, F.N. van de Vosse, and J.D. Janssen. The influence of the non-newtonian properties of blood on the flow in large arteries : unsteady flow in a 90° curved tube. *Journal of Biomechanics*, 32(7) :705 – 713, 1999.

[70] Roland Glowinski, T-W Pan, Todd I Hesla, and Daniel D Joseph. A distributed lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25(5) :755–794, 1999.

[71] Roland Glowinski, Tsorng-Whay Pan, and Jacques Periaux. A fictitious domain method for dirichlet problem and applications. *Computer Methods in Applied Mechanics and Engineering*, 111(3) :283–303, 1994.

[72] Roland Glowinski, Tsorng-Whay Pan, and Jacques Periaux. A fictitious domain method for external incompressible viscous flow modeled by navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 112(1) :133–148, 1994.

[73] W. J. Gordon and C. A. Hall. Construction of curvilinear coordinate systems and their applications. *Int. J. Numer. Meth. Eng.*, 7 :461–477, 1973.

[74] W. J. Gordon and C. A. Hall. Transfinite element methods : blending-function interpolation over arbitrary curved element domains. *Int. J. Numer. Meth. Eng.*, 21 :109–129, 1973.

[75] D. Gregor and M. Troyer. Boost.mpi, 2006.

[76] V Grevy and E Escuret. Le retour veineux sanguin cérébral. *Annales Françaises d'Anesthésie et de Réanimation*, 17(2) :144 – 148, 1998.

[77] JL Guermond and Jie Shen. A new class of truly consistent splitting schemes for incompressible flows. *Journal of computational physics*, 192(1) :262–276, 2003.

[78] J. De Hart, G.W.M. Peters, P.J.G. Schreurs, and F.P.T. Baaijens. A three-dimensional computational analysis of fluid-structure interaction in the aortic valve. *Journal of Biomechanics*, 36(1) :103 – 112, 2003.

[79] F. Hecht, O. Pironneau, J. Morice, A. Le Hyaric, and K. Ohtsuka. Freefem++ documentation, version 3.19-1, 2012. *Webpage : http ://www. freefem. org/ff+.*

[80] Bruce Hendrickson and Robert Leland. The chaco user's guide version 2.0. Technical report, Technical Report SAND95-2344, Sandia National Laboratories, 1995.

[81] C.O. Horgan and G. Saccomandi. Constitutive models for compressible nonlinearly elastic materials with limiting chain extensibility. *Journal of Elasticity*, 77(2) :123–138, 2004.

[82] J. Hron and S. Turek. A monolithic FEM solver for an ALE formulation of fluid–structure interaction with configuration for numerical benchmarking. *Computational Methods for Coupled Problems in Science and Engineering, volume First Edition*, page 148, 2005.

[83] J. Hron and S. Turek. A monolithic FEM/multigrid solver for an ALE formulation of fluid-structure interaction with applications in biomechanics. *Fluid-Structure Interaction*, pages 146–170, 2006.

[84] Thomas JR Hughes and Alec Brooks. A multidimensional upwind scheme with no crosswind diffusion. *Finite element methods for convection dominated flows, AMD*, 34 :19–35, 1979.

[85] Thomas JR Hughes, Leopoldo P Franca, and Gregory M Hulbert. A new finite element formulation for computational fluid dynamics : Viii. the galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, 73(2) :173–189, 1989.

[86] M. Ismail. Méthode de la frontière élargie pour la résolution de problèmes elliptiques dans des domaines perforés. Application aux écoulements fuides tridimensionnels. PhD thesis, Laboratoire Jacques-Louis Lions,Université Pierre et Marie Curie, 2004.

[87] V. John and G. Matthies. Higher-order finite element discretizations in a benchmark problem for incompressible flows. *International Journal for Numerical Methods in Fluids*, 37(8) :885–903, 2001.

[88] Piotr Kalita and Robert Schaefer. Mechanical models of artery walls. *Archives of Computational Methods in Engineering*, 15 :1–36, 2008.

[89] H. Kanchi and A. Masud. A 3D adaptive mesh moving scheme. *International Journal for Numerical Methods in Fluids*, 54(6-8) :923–944, 2007.

[90] S.L. Karman. Virtual control volumes for two-dimensional unstructured elliptic smoothing. *Proceedings of the 19th International Meshing Roundtable*, pages 121–142, 2010.

[91] G. Karniadakis and S.J. Sherwin. *Spectral/hp element methods for CFD*. Oxford University Press, USA, 1999.

[92] V. Karvonen and P. Mensonides. The boost preprocessor library, 2001.

[93] George Karypis and Vipin Kumar. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN*, 1998.

[94] C. Kassiotis, A. Ibrahimbegovic, R. Niekamp, and H.G. Matthies. Nonlinear fluid–structure interaction problem. part i : implicit partitioned algorithm, nonlinear stability proof and validation examples. *Computational Mechanics*, 47(3) :305–323, 2011.

[95] H.J. Kim, I.E. Vignon-Clementel, C.A. Figueroa, K.E. Jansen, and C.A. Taylor. Developing computational methods for three-dimensional finite element simulations of coronary blood flow. *Finite Elements in Analysis and Design*, 46(6) :514 – 525, 2010. <ce :title>The Twenty-First Annual Robert J. Melosh Competition</ce :title>.

[96] Bhaskar Chandra Konala, Ashish Das, and Rupak K. Banerjee. Influence of arterial wall-stenosis compliance on the coronary diagnostic parameters. *Journal of Biomechanics*, 44(5) :842 – 847, 2011.

[97] Steen Krenk. Energy conservation in newmark based time integration algorithms. *Computer Methods in Applied Mechanics and Engineering*, 195(44-47) :6110 – 6124, 2006.

[98] Ellen Kuhl, Harm Askes, and Paul Steinmann. An ale formulation based on spatial and material settings of continuum mechanics. part 1 : Generic hyperelastic formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(39â€“41) :4207 – 4222, 2004.

[99] U. Kuttler and W.A. Wall. Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1) :61–72, 2008.

[100] M.R. Labrosse. Structure and Mechanics of the Artery.

[101] Patrick Lacolley. *Biologie et pathologie du coeur et des vaisseaux*. John Libbey Eurotext, 2007.

[102] Patrick Le Tallec, Jean-Frédéric Gerbeau, Patrice Hauret, and Marina Vidrascu. Fluid structure interaction problems in large deformation. *Comptes Rendus Mecanique*, 333(12) :910–922, 2005.

[103] Patrick Le Tallec and Jean Mouro. Fluid structure interaction with large structural displacements. *Computer Methods in Applied Mechanics and Engineering*, 190(24) :3039–3067, 2001.

[104] S. Lejeunes. *Modélisation de structures lamifiées élastomère-métal à l'aide d'une méthode de réduction de modèles*. PhD thesis, Université de la Méditerranée-Aix-Marseille II, 2006.

[105] G. Lenormand. *Élasticité du squelette du globule rouge humain-une étude par pinces optiques*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2001.

[106] J.H. Leung, A.R. Wright, N. Cheshire, J. Crane, S.A. Thom, A.D. Hughes, and Y. Xu. Fluid structure interaction of patient specific abdominal aortic aneurysms : a comparison with solid stress models. *BioMedical Engineering OnLine*, 5(1) :33, 2006.

[107] Baihua Li, Qinggang Meng, and Horst Holstein. Similarity k-d tree method for sparse point pattern matching with underlying non-rigidity. *Pattern Recognition*, 38(12) :2391 – 2399, 2005.

[108] VD Liseĭkin. *Grid generation methods*, volume 1434. Springer Verlag, 1999.

[109] Renate Lüllmann-Rauch. *Histologie*. De Boeck Supérieur, 2008.

[110] E. Marchandise, C. Geuzaine, and J.F. Remacle. Cardiovascular and lung mesh generation based on centerlines. *International Journal for Numerical Methods in Biomedical Engineering*. submitted in 2013.

[111] Claude Martin, Bruno Riou, and Benoît Vallet. *Physiologie humaine appliquée*. Arnette Blackwell, 2006.

[112] K. Martin and B. Hoffman. An open source approach to developing software in a small organization. *Software, IEEE*, 24(1) :46–53, 2007.

[113] K. Martin and B. Hoffman. *Mastering CMake 4th Edition*. Kitware, Inc., 2008.

[114] I. Masson. *Contribution à la modélisation mécanique du comportement dynamique, hyperélastique et anisotrope de la paroi artérielle*. PhD thesis, Université Paris-Est, 2008.

[115] Bertrand Maury. A fat boundary method for the poisson problem in a domain with holes. *Journal of Scientific Computing*, 16 :319–339, 2001.

[116] D.A. May and L. Moresi. Preconditioned iterative methods for stokes flow problems arising in computational geodynamics. *Physics of the Earth and Planetary Interiors*, 171(1) :33–47, 2008.

[117] Mahdi Esmaily Moghadam, Irene E. Vignon-Clementel, Richard Figliola, and Alison L. Marsden. A modular numerical method for implicit 0d/3d coupling in cardiovascular finite element simulations. *Journal of Computational Physics*, 244(0) :63 – 79, 2013.

[118] P. Moireau, N. Xiao, M. Astorino, C.A. Figueroa, D. Chapelle, C.A. Taylor, and J.F. Gerbeau. External tissue support and fluid–structure simulation in blood flows. *Biomechanics and modeling in mechanobiology*, 11(1) :1–18, 2012.

[119] DP Mok and WA Wall. Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures. *Trends in computational structural mechanics, Barcelona*, 2001.

[120] D.S. Molony, A. Callanan, E.G. Kavanagh, M.T. Walsh, T.M. McGloughlin, et al. Fluid-structure interaction of a patient-specific abdominal aortic aneurysm treated with an endovascular stent-graft. *Biomed Eng Online*, 8 :24, 2009.

[121] A. Moura. *The Geometrical Multiscale Modelling of the Cardiovascular System : Coupling 3D and 1D FSI models*. PhD thesis, Ph. D. thesis, Politecnico di Milano, 2007.

[122] M. Münsch and M. Breuer. Numerical simulation of fluid–structure interaction using eddy–resolving schemes. *Fluid Structure Interaction II*, pages 221–253, 2010.

[123] F. Nobile. Numerical approximation of fluid-structure interaction problems with application to haemodynamics. *PhD, Ecole Polytechnique Féderale de Lausanne, Switzerland*, 2001.

[124] G. Pena. *Spectral element approximation of the incompressible Navier-Stokes equations evolving in a moving domain and applications*. PhD thesis, École Polytechnique Fédérale de Lausanne, November 2009. n^{o}4529.

[125] Gonçalo Pena and Christophe Prud'homme. Construction of a high order fluid-structure interaction solver. *J. Comput. Appl. Math.*, 234(7) :2358–2365, August 2010.

[126] K. Perktold and G. Rappitsch. Computer simulation of local blood flow and vessel mechanics in a compliant carotid artery bifurcation model. *Journal of biomechanics*, 28(7) :845–856, 1995.

[127] Nicole Poussineau. *Réduction variationnelle d'un couplage fluide-structure : application à l'hémodynamique*. PhD thesis, 2007.

[128] C. Prud'homme. A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations. *Scientific Programming*, 14(2) :81–110, 2006.

[129] C. Prud'homme, V. Chabannes, and G. Pena. Feel++ : Finite Element Embedded Language in C++. Free Software available at http ://www.feelpp.org. Contributions from A. Samake, V. Doyeux, M. Ismail and S. Veys.

[130] Christophe Prud'homme. A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations. 2006.

[131] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Gonçalo Pena. Feel++ : A Computational Framework for Galerkin Methods and Advanced Numerical Methods. *ESAIM : Proc.*, 38 :429 – 455, December 2012.

[132] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, Gonçalo Pena, Cécile Daversin, and Christophe Trophime. Advances in Feel++ : a domain specific embedded language in C++ for partial differential equations. In *Eccomas'12 - European Congress on Computational Methods in Applied Sciences and Engineering*, Vienna, Autriche, September 2012.

[133] Christophe Prud'Homme, Gonçalo Pena, and Vincent Chabannes. High order fluid structure interaction in 2D and 3D : Application to blood flow in arteries. In *ACOMEN 11 - 5th International Conference on Advanced COmputational Methods in ENgineering*, pages 1–10, Liège, Belgium, October 2011.

[134] Christophe Prud'homme, Gonçalo Pena, and Alfio Quarteroni. High order methods for the approximation of the incompressible Navier-Stokes equations in a moving domain. *Comput. Meth. Appl. Mech. Eng.*, 2011. to appear.

[135] Christophe Prud'homme. Life : Overview of a unified c++ implementation of the finite and spectral element methods in 1d, 2d and 3d. In Bo Kågström, Erik Elmroth, Jack Dongarra, and Jerzy Waśniewski, editors, *Applied Parallel Computing. State of the Art in Scientific Computing*, volume 4699 of *Lecture Notes in Computer Science*, pages 712–721. Springer Berlin Heidelberg, 2007.

[136] D Quemada. Hydrodynamique sanguine : hémorhéologie et écoulement du sang dans les petits vaisseaux. *Le Journal de Physique Colloques*, 37(C1) :1–1, 1976.

[137] T. Richter. Numerical methods for fluid-structure interaction problems. *Course of lectures*, 2010.

[138] Th. Richter and Th. Wick. Finite elements for fluidâ€“structure interaction in ale and fully eulerian coordinates. *Computer Methods in Applied Mechanics and Engineering*, 199(41â€“44) :2633 – 2642, 2010.

[139] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2003.

[140] M. Safar. Paroi artérielle et vieillissement vasculaire. *Éditions scientifiques et médicales Elsevier SAS*, 2002.

[141] Salmon, Stéphanie, Sy, Soyibou, and Szopos, Marcela. Cerebral blood flow simulations in realistic geometries. *ESAIM : Proc.*, 35 :281–286, 2012.

[142] Abdoulaye Samake, Vincent Chabannes, Christophe Picard, and Christophe Prud'Homme. Domain decomposition methods in Feel++. pp 8, Accepted in DD21 proceedings, 2012.

[143] R. Sandboge. Fluid-structure interaction with openfsitm and md nastrantm structural solver. *Ann Arbor*, 1001 :48105, 2010.

[144] M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher. Benchmark computations of laminar flow around a cylinder. *Notes on numerical fluid mechanics*, 52 :547–566, 1996.

[145] Elisa Schenone, Stéphane Veys, and Christophe Prud'Homme. High Performance Computing for the Reduced Basis Method. Application to Natural Convection. pp 19, Submitted to ESAIM : Proc, CEMRACS 2012, 2012.

[146] B. Smith. Petsc introductory tutorial. 2006.

[147] B. Smith, P. Bjorstad, and W. Gropp. *Domain decomposition : parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press, 2004.

[148] S.Y. Soyibou. *Algorithmes semi-implicites pour des problèmes d'interaction fluide-structure : approches procédures partagées et monolithiques*. PhD thesis, Universite' de Haute Alsace, 2009.

[149] Ranine Tarabay. *Simulation d écoulements sanguins dans des modeles vasculaires complexes*. PhD thesis, 2015.

[150] M.A. Taylor, BA Wingate, and RE Vincent. An algorithm for computing fekete points in the triangle. *SIAM Journal on Numerical Analysis*, 38(5) :1707–1720, 2000.

[151] Tayfun E Tezduyar. Stabilized finite element formulations for incompressible flow computations. *Advances in applied mechanics*, 28(1) :1–44, 1992.

[152] Ryo Torii, Marie Oshima, Toshio Kobayashi, Kiyoshi Takagi, and TayfunE. Tezduyar. Fluid–structure interaction modeling of a patient-specific cerebral aneurysm : influence of structural modeling. *Computational Mechanics*, 43 :151–159, 2008.

[153] Mamadou Toungara. *Contribution à la Prédiction de la Rupture des Anévrismes de l'Aorte Abdominale (AAA)*. PhD thesis, Université de Grenoble, 2011.

[154] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. *Lecture Notes in Computational Science and Engineering*, 53 :371, 2006.

[155] S. Turek, J. Hron, M. Madlik, M. Razzaq, H. Wobker, and JF Acker. Numerical simulation and benchmarking of a monolithic multigrid solver for fluid-structure interaction problems with application to hemodynamics. *Fluid Structure Interaction II*, pages 193–220, 2010.

[156] J.G. Valdés Vázquez et al. Nonlinear analysis of orthotropic membrane and shell structures including fluid-structure interaction. 2007.

[157] A. Valencia, D. Ledermann, R. Rivera, E. Bravo, and M. Galvez. Blood flow dynamics and fluid–structure interaction in patient-specific bifurcating cerebral aneurysms. *International Journal for Numerical Methods in Fluids*, 58(10) :1081–1100, 2008.

[158] Marc J van Kreveld and Mark H Overmars. Divided kd trees. *Algorithmica*, 6(1) :840–858, 1991.

[159] R. Van Loon, PD Anderson, FN Van de Vosse, and SJ Sherwin. Comparison of various fluid–structure interaction methods for deformable bodies. *Computers & structures*, 85(11) :833–843, 2007.

[160] Erwan Verron. *Contribution expérimentale et numérique aux procédés de moulage par soufflage et de thermoformage*. PhD thesis, 1997.

[161] Xiaohong Wang and Xiaoyang Li. Fluid-structure interaction based study on the physiological factors affecting the behaviors of stented and non-stented thoracic aortic aneurysms. *Journal of Biomechanics*, 44(12) :2177 – 2184, 2011.

[162] T. Warburton. An explicit construction of interpolation nodes on the simplex. *Journal of engineering mathematics*, 56(3) :247–262, 2006.

[163] Thomas Wick. Fluid-structure interactions using different mesh motion techniques. *Computers & Structures*, 89(13) :1456–1467, 2011.

[164] A. Winslow. Numerical solution of the quasilinear poisson equations in a nonuniform triangle mesh. *J. Comp. Phys.*, 2 :149–172, 1967.

[165] B.J.B.M. Wolters, M.C.M. Rutten, G.W.H. Schurink, U. Kose, J. de Hart, and F.N. van de Vosse. A patient-specific computational model of fluid-structure interaction in abdominal aortic aneurysms. *Medical Engineering & Physics*, 27(10) :871 – 883, 2005.

[166] P. Zunino, C. D Angelo, L. Petrini, C. Vergara, C. Capelli, and F. Migliavacca. Numerical simulation of drug eluting coronary stents : Mechanics, fluid dynamics and drug release. *Computer Methods in Applied Mechanics and Engineering*, 198(45-46) :3633 – 3644, 2009.

Résumé

Contrairement aux liquides ordinaires, les fluides complexes comme le sang exhibent des comportements étranges qui dépendent essentiellement des structures sous-jacentes qui les composent. La simulation des écoulements sanguins continue de poser un formidable défi pour les modélisations théoriques et numériques dont l'intérêt est de développer des méthodes et des outils de simulation pour la communauté médicale. Nous proposons dans cette thèse une contribution à ce projet qui sera majoritairement centré sur les aspects numériques et informatiques. Nous nous sommes particulièrement intéressés à l'interaction entre le sang et la paroi vasculaire, qui joue un rôle important dans les grandes artères comme l'aorte. Nous nous sommes aussi investis dans la simulation du transport des cellules sanguines dans le sang.

Pour la résolution des équations aux dérivées partielles décrivant nos modèles d'hémodynamique, nous avons choisi d'utiliser des méthodes numériques dont la précision pourra être accrue de manière arbitraire. Dans ce but, les principaux ingrédients qui ont été mis en oeuvre sont *(i)* la méthode des éléments finis basée sur des approximations de Galerkin d'ordre arbitraire en espace et géométrie, *(ii)* la méthode ALE pour la prise en compte de la mobilité des domaines pour des déplacements d'ordre arbitraire, *(iii)* les couplages implicites et semi-implicites pour l'interaction fluide-structure. Nous proposons également une nouvelle formulation de la méthode de la frontière élargie visant à modéliser le transport de particules déformables immergées dans un fluide.

Nos simulations numériques se sont appuyées sur la librairie de calcul Feel++, spécialisée dans la résolution d'EDP. Outre l'implémentation des modèles physiques, nous y avons développé diverses fonctionnalités nécessaires à la mise en oeuvre de nos méthodes : interpolation, méthode de Galerkin non standard, méthode ALE, environnement pour l'interaction fluide-structure. De plus, de par la taille des géométries et la complexité des modèles mis en jeu, le passage au calcul parallèle a été indispensable pour pouvoir réaliser nos simulations. Ainsi, nous avons décrit le développement qui a été effectué dans cette librairie pour permettre le déploiement de nos programmes sur des architectures parallèles.

Mots-clés : simulation des écoulements sanguins, méthode des éléments finis, méthode ALE, interaction fluide-structure, méthode de la frontière élargie, calcul parallèle