Thèse

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE

UMR 7501

Strasbourg

présentée pour obtenir le grade de docteur de
l'Université de Strasbourg
Spécialité MATHÉMATIQUES APPLIQUÉES

Ranine TARABAY

Simulations des écoulements sanguins dans des réseaux vasculaires complexes

Soutenue le 26 septembre 2016
devant la commission d'examen

Christophe PRUD'HOMME, directeur de thèse
Nicolas PASSAT, co-directeur de thèse
Marcela SZOPOS, co-encadrante
Emmanuel MAITRE, rapporteur
Damien TROMEUR DERVOUT, rapporteur
Stéphanie SALMON, examinatrice

www-irma.u-strasbg.fr

Towards a large scale 3D computational model of physiological hemodynamics, remarkable progress has been made in simulating blood flow in realistic anatomical models constructed from three-dimensional medical imaging data in the past few decades. When accurate anatomic models are of primary importance in simulating blood flow, realistic boundary conditions are equally important in computing velocity and pressure fields. Thus, the first target of this thesis was to investigate the convergence analysis of the unknown fields for various types of boundary conditions allowing for a flexible framework with respect to the type of input data (velocity, pressure, flow rate, ...). In order to deal with the associated large computational cost, requiring high performance computing, we were interested in comparing the performance of two block preconditioners; the least-squared commutator preconditioner and the pressure convection diffusion preconditioner. We implemented the latter, in the context of this thesis, in the Feel++ library. With the purpose of handling the fluid-structure interaction, we focused of the approximation of the force exerted by the fluid on the structure, a field that is essential while setting the continuity condition to ensure the coupling of the fluid model with the structure model. Finally, in order to assess our numerical choices, two benchmarks (the FDA benchmark and the Phantom benchmark) were carried out, and a comparison with respect to experimental and numerical data was established and validated.

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE
UMR 7501
Université de Strasbourg et CNRS
7 Rue René Descartes
67 084 STRASBOURG CEDEX

Tél. 03 68 85 01 29
Fax 03 68 85 03 28

www-irma.u-strasbg.fr
irma@math.unistra.fr

Institut de Recherche Mathématique Avancée

IRMA 2010/01

ISSN 0755-3390

http://tel.archives-ouvertes.fr/tel-00468343

Ce qui mérite d'être fait, mérite d'être bien fait.

Nicolas POUSSIN

À toi papa

Contents

Notations

The fluid and flow related quantities

Symbol		Description
Re	:	the Reynolds number $Re = \frac{\rho DU}{\mu}$
D	:	the characteristic length
μ	:	the viscosity of the fluid
ρ	:	the density of the fluid
$\mathbf{u}, p$	:	the velocity and pressure of the fluid
$\mathbf{u}_h, p_h$	:	the discrete versions of the velocity and pressure
$\mathbf{v}, q$	:	the velocity and the pressure test functions of the fluid
$\boldsymbol{\sigma}(\mathbf{u}, p)$	:	the stress tensor $\boldsymbol{\sigma}(\mathbf{u}, p) = -p\mathbf{I} + 2\mu\mathbf{D}(\mathbf{u})$
$\mathbf{D}(\mathbf{u})$	:	the strain tensor: $\mathbf{D}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T))$
λ	:	Lagrange multiplier insuring the inextensibility of the membrane
h	:	the mesh size
Δt	:	the time step
d	:	the space dimension, in this work, $d = 2$ or 3

Domains and spaces

Symbol		Description
Ω	:	the whole domain
$\partial\Omega$	:	the boundary of Ω
Γ_{in}	:	the *inlet* of the domain in the case of an inflow/outflow problem
Γ_{out}	:	the *outlet* of the domain in the case of an inflow/outflow problem
Γ_w	:	the *wall* of the domain in the case of an inflow/outflow problem
Γ_D	:	the part of the domain boundary where a Dirichlet boundary condition is set
Γ_N	:	the part of the domain boundary where a Neumann boundary condition is set
Γ_b	:	the bottom part of a rectangular domain
$L^2(\Omega)$	:	the square-integrable functions space, whose norm is: $\|v\|_{L^2(\Omega)} = \left(\int_\Omega v^2 \right)^{1/2}$
$L^2_0(\Omega)$	:	the square-integrable functions space with null mean pressure
$[L^2(\Omega)]^d$	:	the square-integrable scalar functions space of dimension d

$H^1(\Omega)$	:	the Sobolev scalar functions space, whose norm is: $\|v\|_{H^1(\Omega)} = \left(\|v\|^2_{L^2(\Omega)} + \|\nabla v\|^2_{L^2(\Omega)} \right)^{1/2}$
$H^1_0(\Omega)$	:	the Sobolev functions space with vanishing value at boundary
$[H^1(\Omega)]^d$	:	the Sobolev functions space of dimension d
$H^{\frac{1}{2}}(\partial\Omega)$	:	$\{\mathbf{u} \in L^2(\partial\Omega) \vert \exists \tilde{\mathbf{u}} \in H^1(\Omega), \mathbf{u} = tr(\tilde{\mathbf{u}})\}$
V_δ	:	a generic discrete velocity functions space
Q_δ	:	a generic discrete pressure functions space
N_{V_δ}	:	number of velocity degrees of freedom
N_{Q_δ}	:	number of pressure degrees of freedom
$\varphi_{i,i=1,...,N_{V_\delta}}$	:	the velocity basis functions
$\Psi_{i,\ i=1,...,N_{Q_\delta}}$	:	the pressure basis functions
$\xi_{i,\ i=1,...,N_{Q_\delta}}$	:	the Lagrange multiplier basis functions

Finite elements symbols

Symbol		Description
K	:	a finite element
$\hat{K}$	:	the reference element
$\mathbb{P}_N(K)$	:	a vectorial space of polynomials with a total degree less or equal to N
$(K,\ \mathbb{P}_N,\ \Sigma)$	:	the Lagrange finite elements triplet
$\Sigma = \{\sigma_i, i = 1, ..., T_N\}$	:	defined as follows: $\sigma_i : \mathbb{P}_N(K) \longrightarrow \mathbb{R},\quad p \longmapsto p(\boldsymbol{a}_i)$
$\boldsymbol{a}_i$	:	the interpolation points on K
$\hat{\Phi}^N_i$	:	the basis functions of a finite element $(K,\ \mathbb{P}_N,\ \Sigma)$ such that $\hat{\sigma}_j\left(\hat{\Phi}^N_i\right) = \delta_{ij}, \quad 1 \leq i, j \leq \hat{T}_N$, with $\hat{\Phi}^N_i(x) = \sum_{j=1}^{\hat{T}_N} \alpha_{ij}\psi_j(x), \quad i = 1, ..., \hat{T}_N$, ψ_j being the basis functions of the primal basis
$\boldsymbol{\varphi}^{\text{geo}}_K$	:	the geometric application that transforms $\hat{K}$ into K
$(\boldsymbol{\varphi}^{\text{geo}}_K)^{-1}$	:	the geometric application that transforms K into $\hat{K}$ $\boldsymbol{\varphi}^{\text{geo}}_K : \underset{\hat{\boldsymbol{x}}}{\hat{K} \in \mathbb{R}^d} \to \underset{\boldsymbol{x}}{K \in \mathbb{R}^d}$ and $(\boldsymbol{\varphi}^{\text{geo}}_K)^{-1} : \underset{\boldsymbol{x}}{K \in \mathbb{R}^d} \to \underset{\hat{\boldsymbol{x}}}{\hat{K} \in \mathbb{R}^d}$
$a(\mathbf{u}, \mathbf{v})$	:	the bilinear form defined as follows: $a(\mathbf{u}, \mathbf{v}) = 2\mu \int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v}) dx$
$b(\mathbf{v}, p)$	:	the bilinear form defined as follows: $b(\mathbf{v}, p) = -\int_\Omega p\ \mathbf{div}(\mathbf{v}) dx$
$c(\mathbf{w}, \mathbf{u}, \mathbf{v})$	:	the trilinear form defined as follows: $c(\mathbf{w}, \mathbf{u}, \mathbf{v}) = \rho \int_\Omega (\mathbf{w} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} dx$

Matrices and operators

Symbol		Description
$\mathcal{A}$	:	the Stokes or Navier-Stokes matrix
$\mathbf{A_u}$	:	the stiffness matrix defined on the velocity space
$\mathbf{A}_p$	:	the stiffness matrix defined on the pressure space
$\mathbf{B}$	:	the divergence matrix
$\mathbf{B^T}$	:	the transpose divergence matrix
$\mathbf{C}$	:	the matrix arising from the explicit treatment of the convective term
$\mathbf{Q_u}$	:	the velocity mass matrix
$\mathbf{Q}_p$	:	the pressure mass matrix
$\mathbf{F_u}$	:	matrix that coincides with: $\mathbf{A_u}$: in the case of a Stokes problem $\frac{1}{\Delta t}\mathbf{Q_u} + \mathbf{A_u}$: in case of an explicit treatment of the convective term $\frac{\mathbf{Q_u}}{\Delta t} + \mathbf{A_u} + \mathbf{C}$: in the case of a semi-implicit or an implicit treatment.
$\mathbf{S}$	:	$\mathbf{B}{\mathbf{F_u}}^{-1}\mathbf{B}^T$, the Schur complement
$\mathbf{S}^*$	:	the Schur complement approximation $\mathbf{S}^* = \mathbf{Q_p}\mathbf{F_p^{-1}}\mathbf{A_p}$
$\mathcal{L}$	:	the convection-diffusion operator: $\mathcal{L} = -\nu\nabla^2 + \omega_h \cdot \nabla$
$\mathcal{E}$	:	convection-diffusion operators with the divergence commutator defined as: $\mathcal{E} = \nabla \cdot (-\nu\nabla^2 + \omega_h \cdot \nabla) - (-\nu\nabla^2 + \omega_h \cdot \nabla)_p \nabla$
$\mathbf{F}_p$	:	the pressure convection diffusion matrix associated to the operator $\mathcal{L}$

Abreviations

Abreviation		Description
ALE	:	Arbitrary Lagrangian Eulerian method
AMG	:	Algebraic MultiGrid methods
ASM	:	Additive Schwarz methods methods
bvq	:	boundary value problem
BDF	:	Backward Differential Formula
bJacobi	:	Block Jacobi method
CEMRACS	:	Centre d'Été Mathématique de Recherche Avancée en Calcul Scientifique
CFD	:	Computational Fluid Dynamics
CRB	:	Certified Reduced Basis methods
CG	:	Continuous Galerkin
DD	:	Domain Decomposition method
DG	:	Disontinuous Galerkin
DOF	:	Degrees of Freedom
FD	:	Finite difference method
FDA	:	Food and Drugs Administration
FE	:	Finite elements method
Feel++	:	Finite Elements Embedded Library in C++
FV	:	Finite volumes method
GAMG	:	Geometric Agglomeration MultiGrid methods
GASM	:	Generalized Additive Schwarz Method

CG	:	Conjugate Gradient
GCR	:	Generalized Conjugate Residual method
GMRES	:	Generalised Minimal RESidual method
GPU	:	Graphics Processing Unit
LSC	:	Least Square Commutator preconditioner
ML	:	Multi-Levels algorithm of the Multigrid preconditioning
MINRES	:	Minimal RESidual method
PCD	:	Pressure Convection Diffusion preconditioner
PDE	:	Partial Differential Equations
PETSc	:	Portable, Extensible Toolkit for Scientific Computation
PMM	:	Pressure Mass Matrix preconditioner
SEM	:	Spectral Elements Methods
SIMPLE:	:	Semi-Implicit Method for Pressure Linked Equations
SOR	:	Successive Over Relaxation method
VIVABRAIN	:	VIrtual angiography simulation from 3D and 3D+t BRAIN VAscular models

Introduction

Selon l'Organisation Mondiale de la Santé (OMS), avec plus de 17.5 millions de décès par an, les cardiopathies ischémiques, ou maladies coronariennes, et les maladies cérébro-vasculaires sont la cause majeure de morbidité et de mortalité dans le monde (voir Figure 1). Ceci a motivé, il y a plusieurs années, la communauté mathématique à pousser ses recherches dans le domaine bio-médical afin de mettre à la disposition du monde de la santé des outils automatiques de simulation basés sur la reconstruction de la géométrie vasculaire à partir d'imagerie médicale [61]. Pendant longtemps, ces simulations n'étaient pas envisageables parce qu'elles nécessitaient des ressources de calcul très importantes. Il a fallu attendre les dernières décennies pour que l'application de modèles mathématiques pour les écoulements sanguins devienne monnaie courante au sein de la bio-ingénierie et de la communauté de recherche médicale. Les principales raisons de ce progrès ne sont pas seulement l'augmentation de la puissance des ordinateurs modernes, les progrès de l'imagerie et des techniques d'extraction de la géométrie, mais également le développement de meilleurs algorithmes numériques.

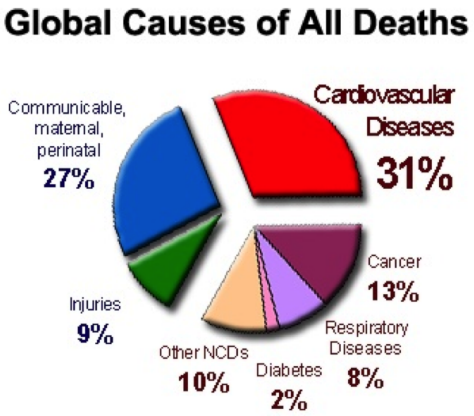


Figure 1: Taux global de mortalité dans le monde en 2012. (*https://heartnewslink.com*)

Les domaines d'application de la CFD sont multiples. Elle a été d'abord adaptée pour des études physiologiques et physiopathologiques du système cardio-vasculaire [139, 59, 62] mais aussi pour établir des modalités d'intervention médicales dans le cadre d'opérations de maladies vasculaires [72, 150, 97]. Cette technique a été également utilisée dans l'industrie d'outils médicaux pour prouver l'efficacité, développer et/ou améliorer les performances des prothèses de valves [101], des stents [7, 102], des filtres de sang [54], des outils d'assistance ventriculaires [153]. En outre, récemment, les méthodes de simulations sanguines ont été développées pour calculer les niveaux d'hémolyse

[109, 69, 25, 41, 154, 10, 15] et de thrombose [152, 67, 130, 53, 136].
L'atout principal des simulations sanguines est qu'elles peuvent être vues comme un *laboratoire virtuel*. Elles permettent en effet de tester différentes hypothèses, de faire varier plusieurs paramètres et tester plusieurs modèles de manière non-invasive et ceci en faisant des expérimentations *"in silico"*. Elles peuvent fournir des informations et un aperçu sur la performance d'un certain dispositif médical en complément des protocoles coûteux sur des animaux ou des tests cliniques. Elles peuvent également fournir des informations dans des régions où les données expérimentales seraient difficiles voire impossible à obtenir. Elles sont même capables de calculer des quantités physiques qu'on ne peut pas mesurer à l'aide d'appareils médicaux tels que le wall shear stress (WSS), grandeur qui donne une information sur la détérioration de la paroi vasculaire à l'origine d'anévrismes lorsqu'elle est élevée et le risque d'athérosclérose lorsqu'elle est faible.
Mon doctorat, financé par un contrat doctoral du Ministère de l'Enseignement Supérieur et de la Recherche, s'inscrit dans le cadre du projet ANR VIVABRAIN dont je suis membre (Virtual angiography simulation from 3D and 3D+t brain vascular models) [1]. L'objectif final de ce projet est la génération d'angiographies virtuelles par résonance magnétique du cerveau humain à partir de modèles anatomiques (3D) et hémodynamiques (3D + t), permettant ainsi à la communauté médicale de mener des expérimentations *in silico* et d'apporter des informations complémentaires, voire impossibles à obtenir, sur des patients (voir Figure 2). Ainsi, ma thèse se situe au niveau de l'étape de simulation numérique d'écoulements sanguins dans les géométries reconstruites à partir de volumes vasculaires extraits de ces images médicales. Plusieurs niveaux de complexité rentrent en jeu dans

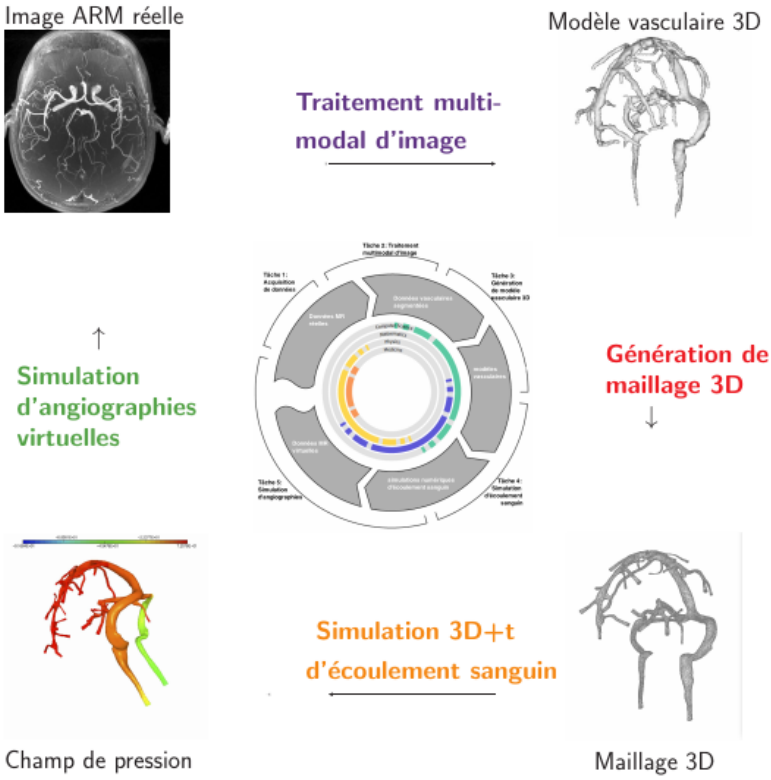


Figure 2: Les différentes tâches du projet ANR VIVABRAIN.

le cadre de la fiabilité de l'approche numérique pour la modélisation des écoulements

sanguins. En effet, il ne s'agit pas seulement de vérifier les méthodes numériques utilisées et de valider le modèle mathématique sur des cas tests académiques, il faut surtout faire une validation sur des géométries réalistes, dans notre cas obtenues par imagerie médicale. Pour arriver à cette fin, il faut que le modèle *i*) gère le cadre multiphysique de l'écoulement, *ii*) se confronte au caractère multi-échelles, *iii*) prenne en compte la variabilité cérébro-vasculaire inter-individuelle pour la génération de modèle anatomique.

Cadre multi-echelles

- En temps:
  Dans le cadre de la modélisation de phénomènes biologiques, une simulation peut aller d'une seconde (un battement cardiaque) à quelques minutes (temps d'acquisition d'une IRM) jusqu'à quelques années (rupture d'un anévrisme).

- En espace:
  Le système circulatoire sanguin est un circuit fermé qui assure le transport du sang du coeur vers les extrémités et les divers organes, et en retour de ceux-ci vers le coeur. Trois types de vaisseaux assurent le transport du sang : les artères, les capillaires et les veines. Les artères sont des gros vaisseaux sanguins (4 à 25 mm) qui assurent le transport du sang du coeur vers les organes. Les veines (5 à 30 mm) assurent le transport du sang des organes vers le coeur et les capillaires sont des vaisseaux sanguins de très petit diamètre (5 μm), constituant une véritable surface d'échange au sein des organes. 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, (voir la table 6).
  Par conséquent, pour pouvoir modéliser l'écoulement sanguin dans tout le système circulatoire, il faut considérer toute l'arborescence de vaisseaux sanguins, de la plus grosse (30 mm) jusqu'à la plus fine (5 μm), ce qui rend le domaine de calcul grand et à échelles variées. Pour remédier à cela, dans la suite des travaux de N. Poussineau [110], nous avons travaillé pendant une partie de cette thèse sur une méthode de réduction variationnelle dont le principe est de tronquer le domaine de calcul et de coupler une zone de calcul 3D d'intérêt (*e.g.*, présentant une pathologie) avec des modèles réduits 1D dont le coût de simulation associé est accessible, (voir Figure 3). Cette partie ne figure pas dans ce manuscrit, les résultats étant incomplets.

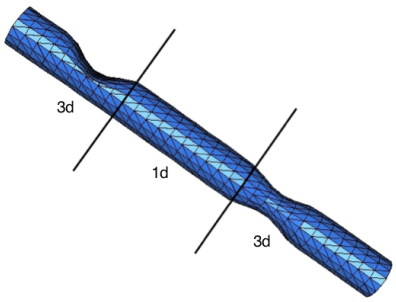


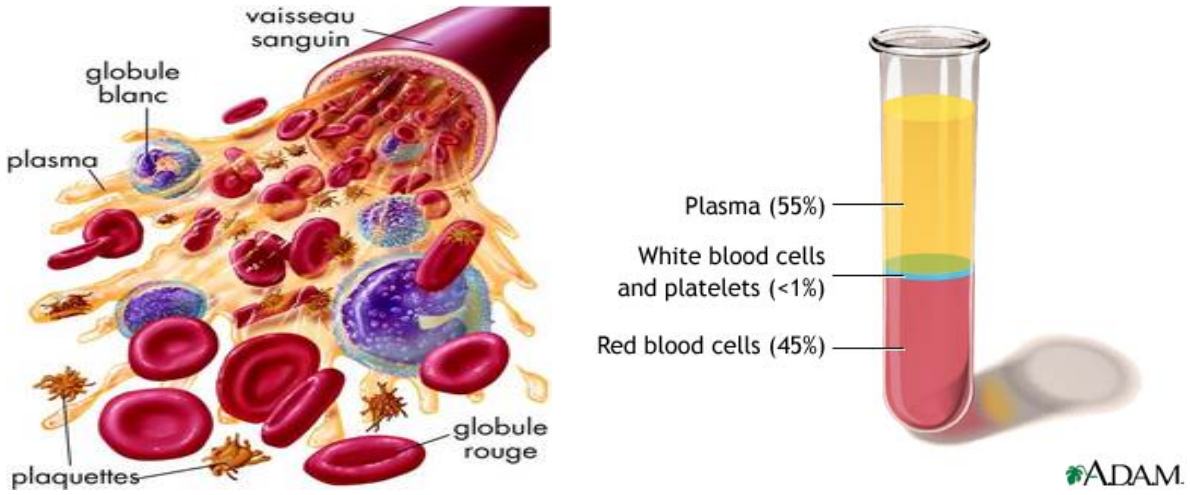
Figure 3: Domaine de calcul partagé en 3 sous-domaines dont les latéraux comprennent chacune une sténose et sont donc traités en 3D, et un domaine intermédiaire droit qui sera traité grâce à un modèle réduit 1D [110].

Vaisseau	Diamètre moyen	Epaisseur de la paroi
Aorte	25	2
Artere	4	1
Arteriole	$3 \cdot 10^{-2}$	$2 \cdot 10^{-2}$
capillaire	$5 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
veinule	$2 \cdot 10^{-2}$	$2 \cdot 10^{-3}$
Veine	5	5
VeineCave	30	1.5

Table 6: Tailles charactéristiques des différents types de vaisseaux sanguins (mm).http://collettemathieu.blog.lemonde.fr/category/cours-sur-la-rigidite-arterielle/

Cadre multi-physique Plusieurs types d'interaction rentrent en jeu dans le cadre de la modélisation des écoulements sanguins:

- Interaction particules-particules et fluide-particule:
  Le sang, est un fluide complexe constitué à 55% de plasma, liquide visqueux de couleur jaunâtre et qui est constitué à 90% d'eau, dans lequel baignent les cellules sanguines : les globules rouges (45%), les globules blancs et les plaquettes (moins de 1%). Pour pouvoir modéliser un comportement physiologique du sang il est alors



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

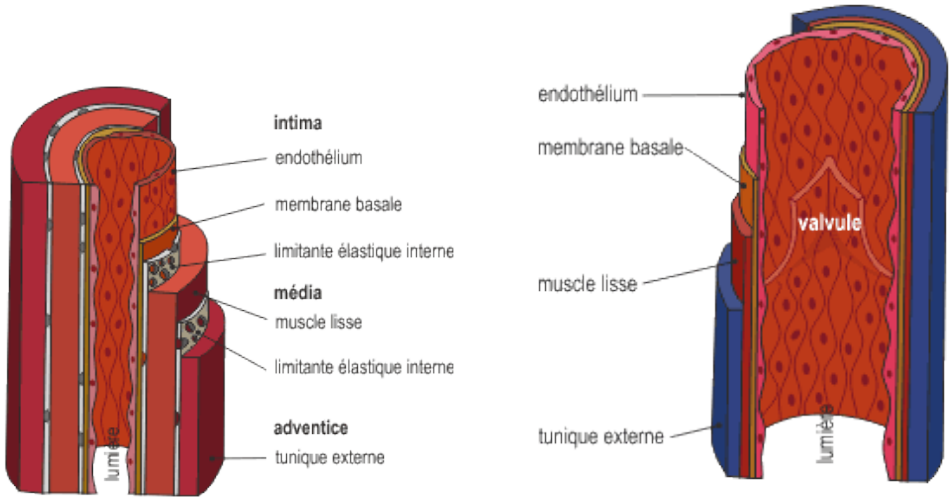
(b) Composition quantitative
(image issue de http://www.pennmedicine.org).

Figure 4: Schémas représentant la composition du sang.

impossible de négliger sa rhéologie et par conséquent les interactions particules-particules et fluide-particule provenant de la constitution même du sang.

Dans le cadre de cette thèse, nous nous sommes intéressés à l'écoulement d'un point de vue macroscopique, la rhéologie constituant à elle seule un domaine de recherche assez vaste.

- Interaction fluide-structure:
  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



(a) Artère (b) Veine

Figure 5: Schéma d'une artère et d'une veine. (image issue de http://www.fedecardio.org/).

(artères et veines) est donnée dans les schémas de la Figure 5. 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 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).

On ne peut donc pas négliger le rôle que joue la paroi sur l'écoulement sanguin. Par conséquent, un autre type d'interaction à prendre en compte est l'interaction entre le fluide et la structure. Dans ce contexte, nous nous sommes intéressés dans ce manuscrit à l'évaluation des contraintes qu'exerce le fluide sur la paroi pour évaluer d'un point de vue numérique et théorique la précision de l'approximation de cette quantité. L'importance d'une bonne approximation du tenseur de déformation vient du fait que cette quantité intervient au moment du couplage entre le modèle pour

le fluide et le modèle pour la structure. À l'interface entre le fluide et la structure, une continuité des contraintes s'impose, assurant ainsi, avec la contrainte de la continuité géométrique et la continuité de vitesse, un couplage numérique entre les deux modèles mathématiques.

Variabilité géométrique La géométrie d'un vaisseau a une forte influence sur l'hémodynamique [134], par consequent, dans le cadre d'une application clinique on ne peut pas envisager de travailler sur une arborescence modèle de la zone vasculaire concernée. Afin d'avoir une description plus précise de l'écoulement, il faut partir de l'imagerie médicale du patient en question pour pouvoir générer le maillage volumique correspondant. Des atlas vasculaires statistiques [45] peuvent aider à l'identification d'anormalités vasculaires et à l'étiquetage de veines.

Malgré ces difficultés numériques, physiologiques et mécaniques la modélisation des écoulements sanguins s'est imposée comme un outil de simulation de phénomènes biologiques, de pronostique et parfois de diagnostic. Les modèles mathématiques pour la simulation d'écoulements sanguins sont présentés dans le paragraphe suivant.

Modèle mathématique pour le écoulements sanguins Puisque la recherche scientifique est avant tout un travail congruent d'enrichissement et de reflexion contemporaine sur des connaissances et des théories de nos prédécesseurs, il est indispensable, en homage à ces gens, et pour la continuité de l'histoire, de faire un brève aperçu historique sur la genèse des équations de Navier-Stokes — le modèle mathématique reproduisant le comportement d'un fluide Newtonien incompressible.
L'histoire de la mécanique des fluides a commencé en 1738 lorsque Daniel Bernoulli s'intéressa sur l'étude les fluides non visqueux, fondant son analyse sur la conservation de l'énergie. La révolution de la compréhension mathématique du mouvement des corps, solides et liquides se poursuivi avec le développement de la théorie du calcul différentiel avec d'abord Leibniz, mais aussi Clairaut, Jean, Jacques et Nicolas Bernoulli puis Newton. Mais il fallait attendre jusqu'en 1750 pour que Jean d'Alembert soumette un manuscrit de 137 pages à l'Académie des sciences de Berlin proposant une nouvelle vision de l'hydrodynamique basée sur l'introduction des notions suivantes: *les dérivées partielles, un champ de vitesses et la pression interne d'un fluide*. L'analyse amenée par d'Alembert sur ce dernier point n'était pas complète, c'est ainsi qu'à Euler que l'on doit l'écriture finale des équations de la dynamique des fluides incompressibles en 1755. Soit $\mathbf{u}$ et p le champ de vitesse et la pression d'un fluide, respectivement, on peut écrire:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla p, \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

Euler parvient à dégager la notion de *gradient de pression*, notion qui avait échappé à d'Alembert. Cependant les équations d'Euler ne permettaient pas de comprendre pourquoi un solide plongé dans un liquide va subir en général une force de résistance, tendant à le freiner. D'Alembert s'en était aperçu mais il fallait Navier en 1820 pour comprendre que, lors de son évolution, un fluide va en effet avoir tendance à dissiper de l'énergie, sous forme de chaleur, et ce simplement par le frottement d'une couche de

fluide sur l'autre. Il introduisit ainsi avec Stokes en 1845 un terme qui permet la dissipation d'énergie sous forme de chaleur, proposant ainsi le modèle de Navier-Stokes pour l'évolution d'un fluide Newtonien incompressible (voir chapitre 2).

$$\rho\frac{\partial \mathbf{u}}{\partial t} - 2\mathbf{div}(\mu\mathbf{D}(\mathbf{u})) + \rho(\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u} + \nabla p = \mathbf{0}, \quad \text{in } \Omega\times I \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \text{in } \Omega\times I \tag{2}$$

Un modèle stationnaire simplifié des équations de Navier-Stokes, qui s'applique lorsque les effets visqueux dominent sur les effets inertiels est le système d'équations de Stokes qui s'écrit:

$$-2\mu\mathbf{div}(\mathbf{D}(\mathbf{u})) + \nabla p = \mathbf{0}, \quad \text{sur } \Omega \tag{3}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \text{sur } \Omega \tag{4}$$

où $\mathbf{u}$ et p correspondent à la vitesse et la pression du fluide, respectivement. μ représente la viscosité dynamique du fluide et $\mathbf{D}(\mathbf{u})$ le tenseur linéaire de déformation, donné par l'expression $\frac{1}{2}(\nabla\mathbf{u}+\nabla\mathbf{u}^T)$. Ces notations permettent de définir le tenseur des contraintes $\boldsymbol{\sigma}(\mathbf{u},p) = -p\mathbf{I} + 2\mu\mathbf{D}(\mathbf{u})$, où $\mathbf{I}$ est le tenseur identité. Le systeme (3)-(4) est ensuite complété par des conditions aux limites appropriées.
Dans ce contexte je me suis également intéressée dans ma thèse au couplage des équations de Stokes à différentes conditions aux limites permettant de couvrir un cadre flexible tenant compte du type de données en entrée (vitesse, pression, débit...) [26]. Nous avons étudié trois manières d'imposer les conditions aux limites pour les sections d'entrée et de sortie: *i*) la condition aux limites la plus classique, le cas où nous connaissons le profil de vitesse en entrèe et à la sortie *ii*) le cas où nous connaissons le profil de vitesse en entrée et la pression en sortie (condition de sortie libre) *iii*) le cas où l'on connait l'expression du tenseur des contraintes en entrée et en sortie. Nous avons montré numériquement que les différentes formulations discrètes associées aux différentes conditions aux limites convergent avec l'ordre prédit par la théorie, non seulement en contrôlant l'erreur de discrétisation par rapport à la vitesse et à la pression mais aussi par rapport à la géométrie, ce dernier point n'étant pas classique et présentant des ouvertures pour de nouveaux résultats théoriques.

La présence de termes non linéaires dans les équations de Navier-Stokes les rend difficiles à résoudre analytiquement . Il est donc nécessaire d'utiliser des méthodes numériques telles que les éléments finis [27, 107, 127, 35, 24], les volumes finis [146], ou encore les différences finies [6] pour trouver des solutions. L'utilisation de telles méthodes nécessite un conditionnement bien adapté au type de problème afin d'éviter les instabilités numériques dues à la propagation d'erreurs. Dans cette thèse la méthode numérique utilisée est la méthode des éléments finis.

Librairie Feel++

Les simulations dans le cadre de cette thèse ont été faites en utilisant Feel++, *Finite Elements Embedded Library in C++*. Cette dernière appartenant à la classe des DSEL, *Domain Specific Embedded Language*, le langage hôte ici enfoui étant le C++. Les DSEL

permettent à chaque utilisateur de pouvoir s'exprimer dans un langage très proche de son langage technique de tous les jours, ici les mathématiques associées à la discrétisation et la résolution des EDPs et ils permettent également à chaque contributeur aux différents niveaux du logiciel de contribuer sur des aspects spécifiques sans se soucier des éventuelles interactions avec les autres niveaux. La figure 6 illustre ces derniers points.

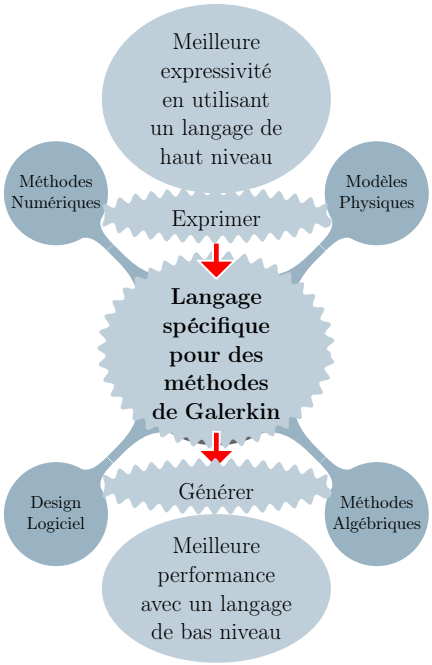


Figure 6: Les langages spécifiques permettent de réduire la complexité.

Feel++ permet la résolution d'EDP en 1D, 2D, 3D par des méthodes de Galerkin (FEM, SEM, CG, DG, CRB) d'ordre arbitraire (y compris en géométrie) en formulation continue ou discontinue sur des maillages de simplexes ou d'hypercubes. Les ingrédients de base de cette librairie sont l'adaptation de maillage, l'interpolation et le parallélisme, outils transparents dans un langage informatique expressif, proche des mathématiques. Les domaines d'utilisation de cette librairie sont multiples, en partant du développement et/ou vérification de (nouvelles) méthodes numériques mathématiques, au développement d'applications multi-physiques à grandes échelles, jusqu'à l'intérêt académique et pédagogique.

Feel++ profite des avantages de la version la plus récente de C++ (C++14) tels que l'interférence de type et s'appuie sur diverses librairies et logiciels telles que

- Boost [91], un ensemble de librairies C++ avancée dont Feel++ utilise un nombre important dont BOOST.PARAMETER, BOOST.FUSION ou BOOST.MPL et autres;
- Gmsh [64] pour la génération et l'adaptation de maillage, et la visualisation;
- PETSc[11, 12] pour les structures de données de matrices et vecteurs ainsi que les solveurs (non)-linéaires et les préconditionneurs;
- SLEPc [74] pour la résolution de problèmes aux valeurs propres standards et généralisés;

- Eigen [52] une librairie d'algèbre linéaire en C++;
- GiNaC, pour le calcul symbolique et formel [14];
- Paraview [73] et Ensight [2] pour la visualisation et le post processing.

Ces différentes bibliothèques ainsi que les améliorations de langage ont permis d'écrire des codes C ++ très concis, robustes et expressifs permettant d'implémenter et de développer des méthodes numériques avancées dont voici une liste non exhaustive ainsi que des publications associées:

- Interaction fluide-structure: formulation ALE ordre élévé [31], formulation level-set [43], formulation ALE-méthode frontières élargies [18];
- Méthodes des domaines fictifs, méthodes frontières élargies [19, 20], pénalisation [111];
- Méthodes de décomposition de domaines: méthodes de Schwarz avec et sans recouvrement, méthodes des joints [123, 122];
- Méthodes des bases réduites [125, 40].

qui ont été appliquées à différents domaines, tels que l'ingénierie mécanique classique, les écoulements sanguins et rhéologie sanguine, la simulation d'électro-aimants à hauts champs ($> 24T$), la tomographie optique, et l'aérothermie.

La dernière version de Feel++ (v 0.100.0) fournit, parmi d'autres fonctionnalités, la possibilité d'utiliser des éléments H_{div} et H_{curl}, ainsi qu'une stratégie de préconditionnement "faite maison" basée sur un framework d'opérateurs et de preconditionneurs par blocs pour des problèmes d'écoulement fluide ou de magnétostatique. Ça repose essentiellement sur trois couches principales: *(i)* une interface d'opérateur , *(ii)* une interface d'extraction de blocs de matrices qui dépend de *Fieldsplit* couplé à *bJacobi* ainsi qu'une *(iii)* interface PETSc pour l'appel de solveurs et préconditionneurs pour les sous-problèmes qui surgissent d'une méthode de préconditionnement par blocs [29]. Plusieurs choix peuvent alors être faits à ce niveau. Feel++ gère par conséquent cette variété de choix et d'options à travers un système de fichiers de configuration où il est possible de: *(i)* définir la géométrie et des paramètres géométriques, *(ii)* assigner des valeurs à des paramètres, *(iii)* choisir la méthode numérique, les solveurs, préconditioneurs et les options pour chaque backend, *(iv)* définir le type de conditions aux limites, la partie du domaine où elles seront appliquées ainsi que leurs expressions, etc.

Pour illustrer le tout, le script suivant présente un code complet Feel++ pour résoudre un problème de Stokes en 3D dans une géométrie d'ordre 2 téléchargée par l'utilisateur pour des éléments finis $\mathbb{P}_2\mathbb{P}_1$.

Listing 1: Code Stokes.

```
int main(int argc, char**argv )
{
    Environment env( _argc=argc, _argv=argv);

    auto mesh = loadMesh(_mesh=new Mesh<Simplex< Dim,GeoDim,Dim > > );
    auto Vh = THchP<U_order>( mesh );
    auto U = Vh->element();
```

```
    auto u = U.element<0>();
    auto p = U.element<1>();
    auto v = U.element<0>();
    auto q = U.element<1>();

    auto g = expr<3,1>( soption(_name="functions.g"), "g" );

    auto a = form2( _trial=Vh, _test=Vh );
    a = integrate(_range=elements(mesh),_expr=trace(gradt(u)*trans(grad(
        ↪ v))) );
    a+= integrate(_range=elements(mesh),_expr=-div(v)*idt(p)-divt(u)*id(
        ↪ q));
    auto l = form1( _test=Vh );
    l = integrate(_range=elements(mesh),_expr=trans(f)*id(u));
    //setting boundary conditions
    a+=on(_range=boundaryfaces(mesh), _rhs=l,_element=u,_expr=g);
    //solve
    a.solve(_rhs=l,_solution=U);
}
```

Ci-dessous quelques remarques concernant le code:

- L'implémentation C++ est très proche de la formulation variationnelle;
- Les détails d'implémentations concernant la parallélisation et la représentation algébrique sont cachés à l'utilisateur, en particulier le code peut être executé de manière transparente aussi bien sur un coeur de calcul que des milliers de coeurs de calcul[1];
- Les formules de quadrature sont déduites automatiquement des expressions, cependant l'utilisateur peut les adapter si nécessaire à travers l'interface d'`integrate(.)`.
- Le code est générique. On peut facilement passer du 2D au 3D, monter en ordre géométrique et en ordre éléments finis.

D'autres exemples ainsi que la documentation de la librairie et du langage sont disponibles dans [37].
Dans ce qui suit, les simulations on été réalisée sur *(i)* Curie au TGCC France grâce à des allocations Prace et Genci qui ont mis à nos dispositions 80.000 coeurs de calcul et des performances de $3\,\text{Pflop}\cdot\text{s}^{-1}$, ainsi qu'un *(ii)* cluster local avec 96 coeurs de calcul d'une performance de $0.44\,\text{Tflop}\cdot\text{s}^{-1}$ mis à notre disposition par Cemosis [3], et *(iii)* sur mesocentre de Strasbourg[2];.

Plan de la thèse

Ce manuscrit est organisé en trois parties.

La première constitue le cadre théorique de mes travaux de thèse et comprend quatre chapitres. Nous commençons dans le Chapitre 1 par introduire quelques notions préliminaires sur les espaces de fonctions qui constitueront le cadre mathématique de la méthode des éléments finis. Nous présenterons les ingrédients de base de cette méthode ainsi que

[1] Feel++ est au coeur de projets PRACE HP-Feel++ (supermuc@lrz) et HP-PDE (curie@tgcc) et a été lauréat des meso-challenges de Strasbourg et Grenoble.

[2] https://hpc.unistra.fr/

les notations correspondantes. Nous finissons ce chapitre par une définition de la notion de scalabilité faible et forte et des métriques d'évaluation de scalabilité. Dans le deuxième chapitre, nous introduisons le système d'équations de Navier-Stokes pour la simulation de l'écoulement d'un fluide Newtonien incompressible dans un domaine fixe. Nous nous attarderons sur les différents types de conditions aux limites qui pourraient compléter ce système ainsi que sur la discrétisation en espace et les multiples techniques de discrétisation en temps. Un panorama sur les méthodes de résolution (directes et itératives) est présenté dans le Chapitre 3. Dans le Chapitre 4 nous faisons un aperçu sur les méthodes de préconditionnement par blocs.

La seconde partie est dédiée aux applications et cas tests expérimentaux et est constituée de même de cinq chapitres. Dans le Chapitre 5 nous montrons les résultats de tests de convergence pour les équations de Stokes, pour les différents types de conditions aux limites détaillées dans la deuxième section du Chapitre 2 ainsi que des résultats de scalabilité pour différents solveurs, travail réalisé pendant le CEMRACS 2012. Les résultats d'une étude de convergence pour deux approches différentes pour le calcul du tenseur des contraintes sont présentés dans la deuxième partie du Chapitre 5. Dans le Chapitre 6, nous décrirons, dans un premier temps, le cas test de la marche dont le but est la validation du préconditionneur PCD à travers une étude de scalabilité et dans un second temps, un cas test proposé par la FDA, *Food ans Drugs Administration*, dont le but est la validation de la CFD et ceci en mettant en ligne, à la disposition de la communauté scientifique, des résultats d'expériences hydrodynamiques effectuées dans des dispositifs qui reproduisent le comportement de dispositifs médicaux. Dans le Chapitre 7, nous décrivons, dans le cadre du projet ANR VIVABRAIN, une chaine logiciel complète qui commence par l'acquisition d'IRM jusqu'à la simulation d'IRM via des simulations d'écoulements sanguins, travail effectué pendant le CEMRACS 2015.

La troisième partie de ce manuscrit est consacrée aux détails de l'implémentation du préconditionneur PCD dans le Chapitre 8.

Publications

- N. Passat, S. Salmon, J.-P. Armspach, B. Naegel, C. Prud'homme, H. Talbot, A. Fortin, S. Garnotel, O. Merveille, O. Miraucourt, R. Tarabay, V. Chabannes, A. Dufour, A. Jezierska, O. Balédent, E. Durand, L. Najman, M. Szopos, A. Ancel, J. Baruthio, M. Delbany, S. Fall, G. Pagé, O. Génevaux, M. Ismail, P. Loureiro de Sousa, M. Thiriet, J. Jomier. From Real MRA to Virtual MRA: Towards an Open-Source Framework. International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2016), Lecture Notes in Computer Science, vol. 9902, 335–343, Springer, Athens, Greece, 2016.

- R. Tarabay, V. Chabannes, M. Szopos, C. Prud'homme, Validation of the open source framework Feel++ for the simulation of blood flow in rigid vessels, under preparation.

- R. Tarabay, A. Ancel, A. Fortin, S. Garnotel, O. Miraucourt, Phantom project: Development and validation of the pipeline from MRA acquisition to MRA simulation, ESAIM: Proceedings, (2015), accepted.

- C. Caldini Queiros, V. Chabannes, M. Ismail, G. Pena, C. Prud'Homme, M. Szopos, R. Tarabay, Towards large-scale three-dimensional blood flow simulations in realistic geometries, ESAIM: Proceedings 43 (2013) 195-212.

Talks

- R. Tarabay, On the block-preconditioning technics for the Navier-Stokes equations, Young researcher seminar, Strasbourg 2016.

- R. Tarabay, Efficient solving strategies for incompressible Navier-Stokes equations for large scale simulations using the open source software Feel++, SimRace 2015 – Numerical methods and high performance computing for industrial fluid flows, Paris, December 2015.

- R. Tarabay, Scalable Navier-Stokes solution strategies, 3rd Feel++ User Day, Strasbourg 2015.

- R. Tarabay, Blood flow simulation in complex vascual networks, Young researcher seminar, Reims 2014.

- R. Tarabay, Vers les méthodes de réduction d'ordre pour l'interaction fluide-structure, 3rd Feel++ User Day, Strasbourg 2014.

- R. Tarabay, Blood flow simulation in complex vascular networks, Young researcher seminar, Strasbourg 2014.

- R. Tarabay, Blood flow simulation in complex vascular networks, 2nd Feel++ User Day, Strasbourg 2013.

Awards

- SimRace simulation challenge award winner :
  (www.rssimrace.com/Projet/jcms/c_880221/fr/simraceaward)
- First price poster winner at the Strasbourg Doctoral School poster journey, 2013.

Part I

Mathematical model

Chapter 1

Preliminaries

For the resolution of the Navier-Stokes equations for fluid flow, several numerical methods have been studied, such as i) the finite volume method (FVM) [146] which has an advantage in memory usage and solution speed, especially for large problems, high Reynolds number turbulent flows, and source term dominated flows. It is based on "physical" conservation properties and hence is widely used in the numerical solution of conservation laws, ii) the finite elements method (FEM), wildly used is the cardiovascular community and well suited to elliptic or parabolic i.e. diffusive problems. A combination of the FEM and the FVM in computational fluid dynamics was used in [103, 104], iii) the finite difference methods (FDM) [6], which are easy to implement, but have difficulties when it comes to curved boundaries, mesh adaptation.

In the context of this thesis, we chose the FE discretisation method. By this approach the discretization inherits most of the rich structure of the continuous problem, which, on the one hand provides a high computational flexibility and on the other hand facilitates a rigorous mathematical error analysis. This chapter provides some preliminary notions on functional analysis, and explains the fundamentals of the FE method and the high order geometry approximation. Some preliminaries on scalability analysis are also introduced in the last section of this chapter.

Contents

1 Function spaces

Let us note Ω a regular bounded domain in $\mathbb{R}^d$ (pour $d = 1, ..., 3$) and $\partial\Omega$ its local Lipschitz boundary.

1.1 Lebesgue function spaces

The $L^2(\Omega)$ function space is the space of square-integrable functions for which the integral of the square of the absolute value is finite in Ω, we have

$$L^2(\Omega) = \Big\{ v : \Omega \to \mathbb{R} \text{ such that } \|v\|_{L^2(\Omega)} < \infty \Big\}, \tag{1.1}$$

where

$$\|v\|_{L^2(\Omega)} = \left(\int_\Omega v^2 \right)^{1/2}. \tag{1.2}$$

The corresponding $L^2(\Omega)$ inner product of two functions f and g, which we denote (f, g) takes the integral of the pointwise product of the two functions over the entire domain Ω:

$$(f, g) := \int_\Omega f(x)g(x)dx.$$

It measures the degree to which the two functions overlap. For instance, in Figure 1.1, the top two functions have a large inner product; the bottom two have a smaller inner product (as indicated by the dark blue regions).

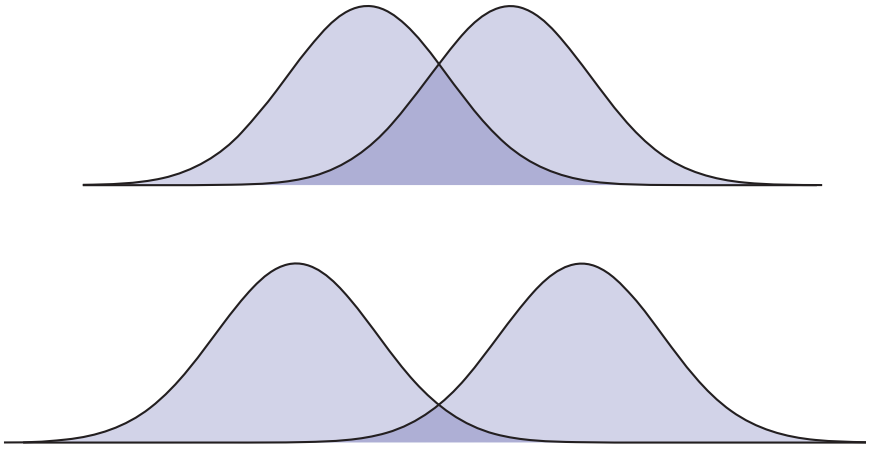


Figure 1.1: $L^2(\Omega)$ scalar product of two functions illustrated with respect to the overlap of the two functions area. (http://brickisland.net/cs177/?p=309)

More generally, a Lebesgue space $L^p(\Omega)$, with $p \geq 1$, is defined as follows:

$$L^p(\Omega) = \Big\{ v : \Omega \to \mathbb{R} \text{ such that } \|v\|_{L^p(\Omega)} < \infty \Big\}, \tag{1.3}$$

with

$$\|v\|_{L^p(\Omega)} = \left(\int_\Omega \big|v\big|^p \right)^{1/p}, \ 1 \leq p \leq \infty \text{ et } \|v\|_{L^\infty(\Omega)} = \operatorname*{ess\,sup}_{\mathbf{x}\in\Omega} \big|v(\mathbf{x})\big|. \tag{1.4}$$

1.2 Hilbert function space

We denote $H^k(\Omega)$, with $k \in \mathbb{N}^+$, the Hilbert function spaces. We have

$$H^k(\Omega) \ = \Big\{ v : \Omega \to \mathbb{R} \text{ such that } D^\alpha v \in L^2(\Omega) \ , \forall \alpha \text{ such that } |\alpha| \leq k \Big\}, \tag{1.5}$$

where $D^\alpha v$ is the general distributional partial derivative of v. For all the indices $\alpha = (\alpha^1, ..., \alpha^l)$ such that $\alpha^i \geq 0$ for $i = 1, ..., l$, and for all $\mathbf{x} = (x^1, ..., x^l) \in \Omega$, we have

$$D^\alpha \equiv \frac{\partial^\alpha}{\partial_{x^1}^{\alpha^1} ... \partial_{x^l}^{\alpha^l}}, \text{ and } |\alpha| = \sum_{i=1}^{l} \alpha_i. \tag{1.6}$$

Let us denote $\big(u, v\big)_{H^k(\Omega)}$ the scalar product associated to $H^k(\Omega)$. It is defined as follows,

$$\big(u, v\big)_{H^k(\Omega)} \ = \sum_{|\alpha| \leq k} \int_\Omega D^\alpha u \ D^\alpha v, \tag{1.7}$$

and the induced norm,

$$\|u\|_{H^k(\Omega)} \ = \left(\sum_{|\alpha| \leq k} \int_\Omega \big| D^\alpha u \big|^2 \right)^{1/2} . \tag{1.8}$$

We can notice that the $L^2(\Omega)$ space, which corresponds to $H^0(\Omega)$, is the only Lebesgue space that is an Hilbert space too.

1.3 Sobolev spaces

Let us now introduce the spaces that we need to define the variational formulation of the partial differential equations, PDEs. This formulation, also called weak formulation, is the basis of the finite elements method. Let $W^{k,p}(\Omega)$, with $k \geq 0$ and $p \geq 1$, denote the Sobolev space defined as follows,

$$W^{k,p}(\Omega) = \Big\{ v : \Omega \to \mathbb{R} \text{ such that } D^\alpha v \in L^p(\Omega) \ , \forall \alpha \text{ such that } |\alpha| \leq k \Big\}. \tag{1.9}$$

The Sobolev spaces are Banach spaces for the following norm:

$$\big|\big|v\big|\big|_{W^{k,p}(\Omega)} = \left(\sum_{|\alpha| \leq k} \int_\Omega \big| D^\alpha v \big|^p \right)^{1/p} , \text{ for } 1 \leq p < \infty, \tag{1.10}$$

and

$$\big|\big|v\big|\big|_{W^{k,\infty}(\Omega)} = \max_{|\alpha| \leq k} \operatorname*{ess\,sup}_{\mathbf{x} \in \Omega} \big| D^\alpha v(\mathbf{x}) \big|. \tag{1.11}$$

The case $p = 2$ is particularly interesting. In fact, the spaces $W^{k,2}(\Omega)$, with $k \geq 0$, have a Hilbert space structure ($W^{k,2}(\Omega) = H^k(\Omega)$). Let us also note that when we set $k = 0$, the spaces $W^{0,p}(\Omega)$ are the $L^p(\Omega)$ spaces. Thus, it is clear that the Lebesgue spaces are particular cases of the Sobolev spaces.

2 Finite elements approximation

In this section, we will first start by describing the finite elements method. We explain the construction of this method for arbitrary polynomial approximation order in space, but also for an arbitrary polynomial approximation order of the geometry. This description is taken from [107, 28]. The reader may refer to these two theses for more details. In this thesis, the elementary elements used in the meshes are mainly simplexes segments (1D), triangles (2D) or tetrahedrons (3D). In the following, the descriptions are done using simplexes, but it is also possible to use hypercubes quadrangles (2D) or hexahedrons (3D) with a similar treatment. A complete description may be found in [27, 107, 127, 35, 24].

2.1 Finite elements notations

In the context of finite elements methods and spectral elements, we will rely on the definition of Ciarlet [35]. This formalism introduces the notion of finite elements, an elementary brick in functions interpolation. To construct a finite element, we need to define the triplet $(K,\ P,\ \Sigma\ )$, with :

- $K \in \mathbb{R}^d$, a compact geometrical element, connected with nonempty interior ($\overset{\circ}{K} \neq \varnothing$) with Lipschitz boundary;
- P, a finite-dimensional vector space of functions $p : K \to \mathbb{R}^\mu$ with μ a positive integer;
- Σ, a set of T_N linear forms σ_i defined on P such that the linear application:

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

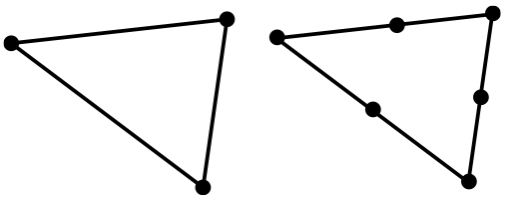
  is bijective. The linear forms $(\sigma_1, ..., \sigma_{T_N})$ are called the degrees of freedom of the finite element.

There are many finite elements, we can cite, for example, the Lagrange finite elements, the Raviart-Thomas finite elements, the Crouzeix-Raviart finite elements, the Nedelec finite elements, ... A complete description of those elements is done in [35, 50, 88]. In the following, we will use the simplest and commonly used finite element, the Lagrange finite elements described as follows: $(K,\ \mathbb{P}_N,\ \Sigma)$. The set of linear forms $\Sigma = \{\sigma_i, i = 1, ..., T_N\}$ is defined as follows:

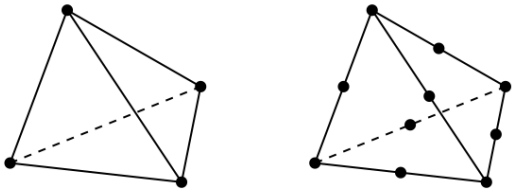
$$\begin{aligned} \sigma_i : \mathbb{P}_N(K) &\longrightarrow \mathbb{R} \\ p &\longmapsto p\left(\boldsymbol{a}_i\right) \end{aligned}$$

where the $\boldsymbol{a}_i$ are the interpolation points on K and $\mathbb{P}_N(K)$ is a vectorial space of polynomials with a total degree less or equal to N. The vectorial space is defined on the element K and has a value in $\mathbb{R}$. To construct the vectorial space $\mathbb{P}_N(K)$, we will have to choose a basis of this space called the primal basis. An illustration of a 2D and a 3D $\mathbb{P}_1$ and $\mathbb{P}_2$ representation of a finite element is shown in figure 1.2.

Definition 1. *Let $(K, \mathbb{P}_N, \Sigma)$ be a Lagrange finite element, $\{\Sigma_i, i = 1, ..., T_N\}$ the linear forms associated to Σ, and T_N the dimension of $\hat{\mathbb{P}}_N(K)$. The basis functions of a finite*



(a) 2D representation of P1 (left) and P2 (right) elements.



(b) 3D representation of P1 (left) and P2 (right) elements.

Figure 1.2: Element representation in 2D and 3D. [49]

element $(K, \mathbb{P}_N, \Sigma)$ *see figure 1.3, are the functions* $\left\{\Phi_i^N \in \mathbb{P}_N(K), i = 1, ..., T_N\right\}$ *such that:*

$$\Sigma_j\left(\Phi_i^N\right) = \delta_{ij}, \quad 1 \leq i, j \leq T_N \tag{1.12}$$

With this choice of finite element construction, the primal basis is only constructed on the reference element K. We denote $\mathcal{B} = \{\psi_j\}_{j=1}^{T_N}$ the set of basis functions of the primal basis. Recalling the Lagrange finite elements example, each of the basis functions Φ_i^N is determined by the T_N^2 coefficients α_{ij} such that:

$$\Phi_i^N(x) = \sum_{j=1}^{T_N^2} \alpha_{ij}\psi_j(x), \quad i = 1, ..., T_N^2 \tag{1.13}$$

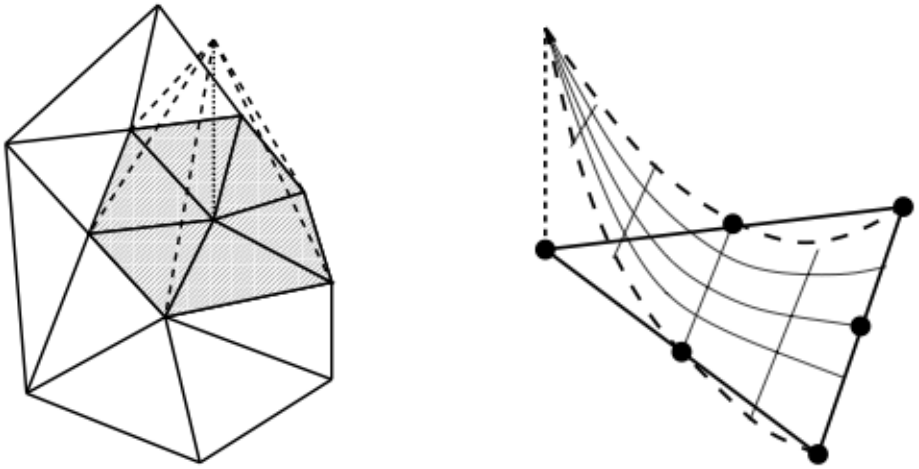
2.2 Geometric transformation

There are two techniques in order to define a finite element K. We can either apply the previously defined formalism on K, or we can bring back all the calculus to the reference element. In the following, we will use the latter for it has many advantages explained in the sequel. Let us recall the basis functions $\left\{\Phi_i^N\right\}_{i=1}^{T_N}$ associated to the Lagrange finite element $(K, \mathbb{P}_N, \Sigma)$. In order to define those basis functions, we will first define the $\hat{\Phi}_i^N$ basis functions defined on the reference element $\hat{K}$ and then obtain the basis functions defined on the real element K, via a geometric transformation that we denote $\boldsymbol{\varphi}_K^{\text{geo}}$.

Thus, many elementary calculus will be done on the element $\hat{K}$, such as the evaluation or the derivative of the basis functions on the quadrature points.

The geometric transformation is an invertible application from $\hat{K}$ to K, whenever the element is not degenerate. We thus define $\boldsymbol{\varphi}_K^{\text{geo}}$ and its inverse $\left(\boldsymbol{\varphi}_K^{\text{geo}}\right)^{-1}$:

$$\begin{array}{ccccccccccc} \boldsymbol{\varphi}_K^{\text{geo}}: & \hat{K} \in \mathbb{R}^d & \to & K \in \mathbb{R}^d & \text{and} & \left(\boldsymbol{\varphi}_K^{\text{geo}}\right)^{-1}: & K \in \mathbb{R}^d & \to & \hat{K} \in \mathbb{R}^d \\ & \hat{\boldsymbol{x}} & \mapsto & \boldsymbol{x} & & & \boldsymbol{x} & \mapsto & \hat{\boldsymbol{x}} \end{array} \tag{1.14}$$



(a) P1 basis function (b) P2 basis function

Figure 1.3: First and second order basis function representation. [49]

The reference element $\hat{K}$ may be freely chosen as long as it respects:

- the element type (segment, triangle, quadrangle, tetrahedron, ...);
- the number of geometric nodes (order of the geometric transformation);
- the compatibility of the nodes numbering in K and $\hat{K}$, in order to guarantee the C^1-diffeomorphism property of $\boldsymbol{\varphi}_K^{\text{geo}}$.

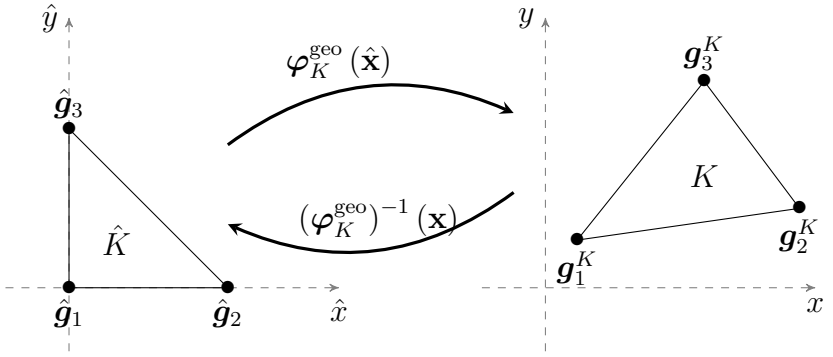
However, its choice may be motivated by the domain definition of the primal basis.

We will now use the Lagrange finite element to define the geometric transformation $\boldsymbol{\varphi}_K^{\text{geo}}$. This formalism allows us to generalise the geometric transformation definition for an arbitrary approximation order. In figure 1.4, we show two types of geometric transformation.

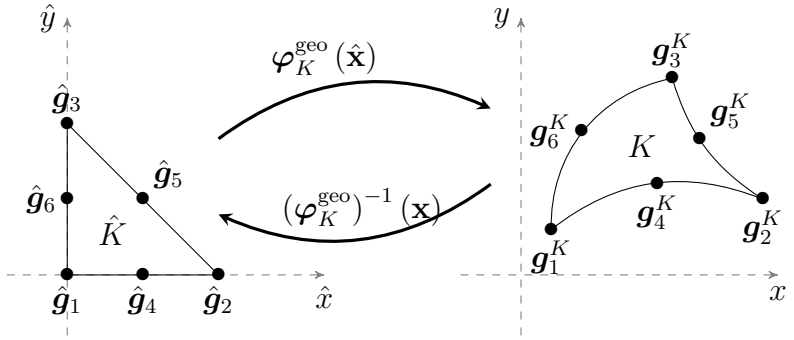
Definition 2. *We say that* $(\hat{K}, \hat{\mathbb{P}}_k^{geo}, \hat{\Sigma}^{geo})$ *is the geometric finite element of order k. We set* $N_g = card(\hat{\Sigma}^{geo})$. *We denote* $\left\{\hat{\boldsymbol{g}}_1, ..., \hat{\boldsymbol{g}}_{N_g}\right\}$ *the set of geometric nodes of* $\hat{K}$. *The later are the equidistributed points on this element. The set of basis functions of this element is denoted by* $\left\{\psi_1, ..., \psi_{N_g}\right\}$.

The element K is composed of N_g geometric nodes $\left\{\boldsymbol{g}_1^K, ..., \boldsymbol{g}_{N_g}^K\right\}$. Each point $\boldsymbol{g}_i^K$ is characterised by its coordinates $(g_{i,x}^K, g_{i,y}^K, g_{i,z}^K)$ in the cartesian coordinate system. We can write $\boldsymbol{\varphi}_K^{\text{geo}}$ as a linear combination of basis functions $\psi_i, i = 1, ..., N_g$:

$$\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}) = \sum_{i=1}^{N_g} \boldsymbol{g}_i^K \psi_i(\hat{\mathbf{x}}) \tag{1.15}$$



(a) Linear transformation on a triangle.



(b) Quadratic transformation on a triangle.

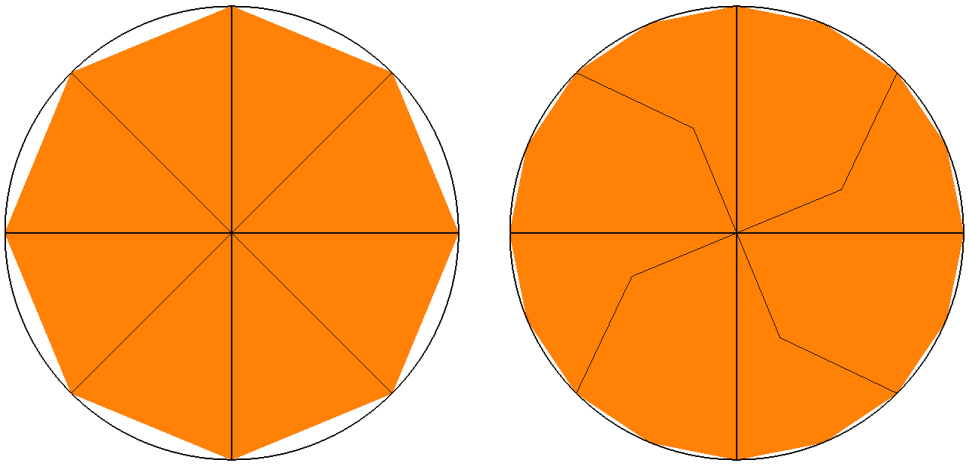
Figure 1.4: Geometric transformations of order 1 et 2 on a triangle. [28]

We will also need to define the Jacobian matrix of $\boldsymbol{\varphi}_K^{\text{geo}}$, that we will denote $J(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}})$, its determinant $|J(\hat{\mathbf{x}})| = det\,(J(\hat{\mathbf{x}}))$, and the transpose of the inverse of $J(\hat{\mathbf{x}})$ that we will denote $J(\hat{\mathbf{x}})^{-T}$. Those expressions are essential when we need to do a substitution $\boldsymbol{\varphi}_K^{\text{geo}}$ in the integral on K or on a face of K. That is, instead of doing an integral over an element K we will bring the integral on the reference element $\hat{K}$.

Let $f : K \to \mathbb{R}$ be a function. The following substitution formulas will be used in the integral calculus over an element:

$$\begin{aligned}
\int_K f(\mathbf{x})\, d\mathbf{x} &= \int_{\hat{K}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}))\ |J(\hat{\mathbf{x}})|\ d\hat{\mathbf{x}} \\
\int_K \nabla_{\mathbf{x}} f(\mathbf{x})\, d\mathbf{x} &= \int_{\hat{K}} \nabla_{\hat{\mathbf{x}}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}))\ J(\hat{\mathbf{x}})^{-T}\ |J(\hat{\mathbf{x}})|\ d\hat{\mathbf{x}} \\
\int_K \nabla_{\mathbf{x}} f(\mathbf{x}) \cdot \nabla_{\mathbf{x}} g(\mathbf{x})\, d\mathbf{x} &= \int_{\hat{K}} \nabla_{\hat{\mathbf{x}}} f(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}))\ (J(\hat{\mathbf{x}})^{-T}\ J(\hat{\mathbf{x}})^{-1}) \nabla_{\hat{\mathbf{x}}} g(\boldsymbol{\varphi}_K^{\text{geo}}(\hat{\mathbf{x}}))\ |J(\hat{\mathbf{x}})|\ d\hat{\mathbf{x}}
\end{aligned}$$

In order to illustrate the importance of a high order geometry approximation, a convergence analysis was carried on the approximation of a unit disk surface (2D) and the approximation of a unit sphere volume(3D). The error plots for $|\int_\Omega 1 - \int_{\Omega_h} 1|$, Ω being the sphere in the graph in the left and the disk in the graph of the right are reported in Figure 1.6. An order of 2 is retrieved for the approximation error of the volume of the sphere for a geometric transformation of first order $\mathbb{G}_1$, while an order of 4 is retrieved for a geometric transformation of second order $\mathbb{G}_2$. The same orders are obtained for the



(a) $\mathbb{G}_1$ mesh (b) $\mathbb{G}_2$ mesh

Figure 1.5: $\mathbb{G}_N$ geometry approximation, $N = 1, 2$.

approximation error of the surface of the unit disk for a geometric transformation of order $\mathbb{G}_1$ and $\mathbb{G}_2$.

3 Scalability Analysis

In the context of High-Performance Computing (HPC), the scalability expresses the ability of a given parallel algorithm to best exploit a parallel computer architecture. The notion of scalability is related to the notions of speedup and efficiency. We introduce below the notion of strong scalability (speedup) and the weak scalability (efficiency).

Definition 3. *Let T_1 be the sequential run time of a problem of size n on one processor and T_p be the parallel run time of the same problem on p processors. Let us denote S_p the speedup of an application, it is given by the relation:*

$$S_p = T_1/T_p.$$

The measure of speedup is used to determine the quality of parallel algorithm running on a parallel computational platform.

Definition 4. *The strong scalability determines how the computational time varies with the number of cores for a fixed overall problem size. An application is said to be strongly scalable if the time to solution would decrease in inverse proportion to the number of cores employed (so-called linear speedup).*

Definition 5. *The weak scalability study consists of increasing the problem size n and the number of processing units p such as the problem size per processing unit n/p remains constant throughout all computations. Let Tp be the time to solution of a problem of size $p \times n$ on p processors, and Tq be the time to solution of a problem of size $q \times n$ on q processors. An application is said to be weakly scalable if $Tp \approx Tq$.*

Definition 6. *The efficiency is defined by*

$$E_p = S_p/p.$$

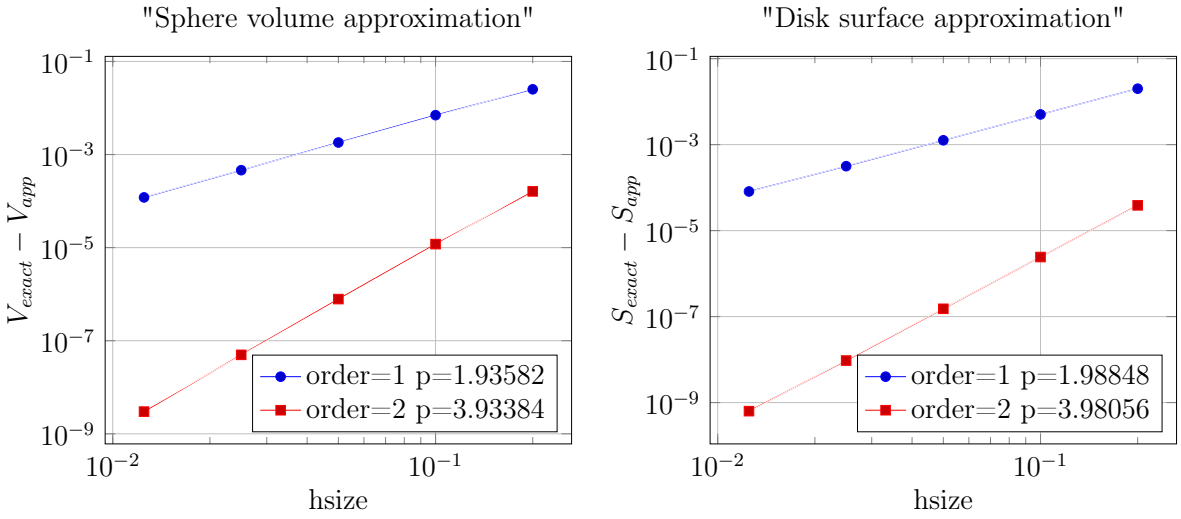


Figure 1.6: High order geometry approximation: error plots for $|\int_\Omega 1 - \int_{\Omega_h} 1|$, Ω being a sphere (left) and a disk (right).

The best possible efficiency is $E_p = 1$. It is reached when the speedup is linear, *i.e.* $S_p = p$. Note that in efficiency analysis, the problem size assigned to each core remains constant and additional processing units are used for solving a larger problem. Thus, the efficiency is directly related to the weak scalability.

The performance of a parallel computer depends on a wide number of factors [81] affecting the scalability of a parallel algorithm.

From [81], some basic metrics affecting the scalability of a parallel computer for a parallel algorithm are given by:

- Machine size — the number of processing units employed in a parallel computer. The computational power is a function of machine size.
- Clock rate — the clock rate refers to the frequency of a CPU.
- Problem size — the amount of computational workload used for solving a given problem. The problem size is directly proportional to the *sequential execution time*.
- CPU time — the elapsed CPU time (in seconds) while running a given program on a parallel computer with n processing units. It is the *parallel execution time*.
- I/O demand — the input/output demands when running the program.
- Memory capacity — the amount of main memory (in bytes) used in the execution of a program. The memory demand is affected by the problem size, the algorithms and the data structures used.
- Communication overhead — the amount of time elapsed in interprocess communications, synchronization and remote memory access.
- Computer cost — the total cost of hardware and software resources required to carry out the execution of a program.

Chapter 2

Fluid model

This chapter is dedicated to the mathematical model devoted for the simulation of a Newtonian, incompressible fluid flow, the Navier-Stokes equations system. We are interested in particular, in the Galerkin finite element method for the numerical resolution. In this chapter, we explicit the different variational formulations retrieved by applying i) different time discretisations, and ii) different boundary conditions.We study the stability of the scheme, and give the corresponding error estimates. The numerical convergence analysis with respect to the various variational formulation presented in this chapter is reported in chapter 5.

Contents

1 Introduction

In order to perform a rigorous hemodynamics modelling, since blood is not just a fluid but a suspension of particles in a fluid mainly made of water, the plasma, we can not but mention the two models depending on the blood flow behaviour. On one side the Newtonian model which neglects shear thinning and viscoelastic effects and is suitable in larger vessels or when we are not interested in the finer details of the flow, as non-Newtonian behaviour may affect. On the other side, in vessels of diameter, say, less than 1 mm, the use of Newtonian models is hardly justifiable. The small velocities and shear stress here involved call for the use of one of the non-Newtonian models.
In this thesis, we focus our investigations on flow in large and medium sized vessels. The flow is hence governed by the unsteady incompressible Navier-Stokes equations.

2 The Navier-Stokes equations

Let $\Omega \subset \mathbf{R}^d$, $d \geq 1$, denote the bounded connected domain under investigation, fixed in time. The Navier-Stokes equations can be written as:

$$\rho \frac{\partial \mathbf{u}}{\partial t} - 2\mathbf{div}(\mu \mathbf{D}(\mathbf{u})) + \rho(\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} + \nabla p = \mathbf{0}, \quad \text{in } \Omega \times I \tag{2.1}$$

$$\text{div}(\mathbf{u}) = 0, \quad \text{in } \Omega \times I \tag{2.2}$$

where $I = (0, T]$ is the time interval, $\mathbf{u}$ and p are the velocity and pressure of the fluid, respectively, ρ and μ are the density and the dynamic viscosity of the fluid, respectively, and $\mathbf{D}(\mathbf{u})$ is the linear fluid deformation tensor (given by the expression $\frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T)$). These notations allow us to define the stress tensor $\boldsymbol{\sigma}(\mathbf{u}, p) = -p\mathbf{I} + 2\mu \mathbf{D}(\mathbf{u})$, where $\mathbf{I}$ is the identity tensor. System (2.1) - (2.2) is completed with appropriate initial and boundary conditions that will be detailed later. The first equation expresses the conservation of linear momentum. It is a vector equation formed by three scalar equations, one for each component of the velocity. The second equation expresses the conservation of mass.

The flow is characterised by the Reynolds number:

$$Re = \frac{\rho D U}{\mu}, \tag{2.3}$$

a dimensionless number that identifies the transition of the flow to turbulence. It depends on the diameter of the vessel D (or radius R), the mean blood velocity U, the density and viscosity of the blood. The higher the Reynolds number gets, the more turbulent the flow becomes, and vice-versa.

2.1 Variational form of the Navier-Stokes equations

To write formally a variational formulation of the problem (2.1) - (2.2), let us denote by $\mathbb{V}$ and $\mathbb{M}$ the functional spaces for the velocity and pressure fields, respectively. These spaces will be set later on according to the specific choices of boundary conditions. We will take, for the moment, $\mathbb{V} = [H^1(\Omega)]^d = [\{f \in L^2(\Omega) / \forall i = 1 \dots n, \frac{\partial f}{\partial x_i} \in L^2(\Omega)\}]^d$ and $\mathbb{M} = L^2(\Omega)$.

Taking the scalar product of equation (2.1) by a test function $\mathbf{v} \in \mathbb{V}$, multiplying equation (2.2) by a test function $q \in \mathbb{M}$ and integrating the resulting equalities over Ω, we are led to the following weak formulation: for every t>0, find $(\mathbf{u}(t), p(t)) \in \mathbb{V} \times \mathbb{M}$ such that $\forall \mathbf{v} \in \mathbb{V}$, $\forall q \in \mathbb{M}$,

$$\int_\Omega \rho \frac{\partial \mathbf{u}(t)}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}(t)\cdot\nabla\mathbf{u}(t))\cdot\mathbf{v} - 2\mu\int_\Omega \mathbf{div}(\mathbf{D}(\mathbf{u}(\mathbf{t})))\cdot\mathbf{v}\ d\mathbf{x} \tag{2.4}$$

$$+\int_\Omega \nabla p(t)\cdot\mathbf{v}\ d\mathbf{x} = 0, \tag{2.5}$$

$$\int_\Omega q\ \mathrm{div}(\mathbf{u}(t))\ d\mathbf{x} = 0 \tag{2.6}$$

Remark 1. *In the following we will use* $\mathbf{u}$ *and* p *instead of* $\mathbf{u}(t)$ *and* $p(t)$*, for simplicity reasons.*

We integrate by parts the third and forth integrals of equation (2.4). We obtain

$$\int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}) : \boldsymbol{\nabla}\mathbf{v}\ d\mathbf{x} - 2\mu\int_{\partial\Omega}\mathbf{D}(\mathbf{u})\boldsymbol{n}\cdot\mathbf{v}\ ds$$
$$+\int_{\partial\Omega} p\ \mathbf{v}\cdot\boldsymbol{n}\ ds - \int_\Omega p\ \mathrm{div}(\mathbf{v})\ d\mathbf{x} = 0.$$

Note that, for symmetry reasons, the equality $\mathbf{D}(\mathbf{u}) : \boldsymbol{\nabla}\mathbf{v} = \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v})$ holds. Thus, the variational formulation of (2.1) - (2.2) can be written as: find $(\mathbf{u}, p) \in \mathbb{V} \times \mathbb{M}$ such that $\forall \mathbf{v} \in \mathbb{V}$, $\forall q \in \mathbb{M}$ we have:

$$\int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v})\ d\mathbf{x} - 2\mu\int_{\partial\Omega}\mathbf{D}(\mathbf{u})\boldsymbol{n}\cdot\mathbf{v}\ ds$$
$$+\int_{\partial\Omega} p\ \mathbf{v}\cdot\boldsymbol{n}\ ds - \int_\Omega p\ \mathrm{div}(\mathbf{v})\ d\mathbf{x} = 0$$
$$\int_\Omega q\ \mathrm{div}(\mathbf{u})\ d\mathbf{x} = 0$$

or, equivalently: find $(\mathbf{u}, p) \in \mathbb{V} \times \mathbb{M}$ such that $\forall \mathbf{v} \in \mathbb{V}$, $\forall q \in \mathbb{M}$

$$\int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v})\ d\mathbf{x} - \int_{\partial\Omega}\boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n}\cdot\mathbf{v}\ ds$$
$$-\int_\Omega p\ \mathrm{div}(\mathbf{v})\ d\mathbf{x} = 0 \tag{2.7}$$
$$\int_\Omega q\ \mathrm{div}(\mathbf{u})\ d\mathbf{x} = 0 \tag{2.8}$$

We have not yet incorporated the boundary conditions in the weak formulation. To do so, let us start with denoting $\partial\Omega = \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_w$ the local Lipschitz boundary of the domain Ω where Γ_w is the wall where we will consider an adherence boundary condition, Γ_{in} the inlet and Γ_{out} outlet of a chanel. Note that inlet and outlet can have several locally connected components (from the topological point of view). The common boundary condition to all our following simulations which is the *no slip* condition on Γ_w:

$$\mathbf{u} = \mathbf{0} \text{ on } \Gamma_w.$$

The standard manner to deal with this essential boundary condition is to choose the functional space $\mathbb{V}$ as

$$\mathbb{V} = \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_w\}. \tag{2.9}$$

Within this functional setting for the velocity field, equations (2.7) - (2.8) are rewritten as: find $(\mathbf{u}, p) \in \mathbb{V} \times \mathbb{M}$ such that $\forall \mathbf{v} \in \mathbb{V}$, $\forall q \in \mathbb{M}$,

$$\begin{aligned} \int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}):\mathbf{D}(\mathbf{v})\ d\mathbf{x} & \\ - \int_{\Gamma_{in}\cup\Gamma_{out}} \boldsymbol{\sigma}(\mathbf{u},p)\boldsymbol{n}\cdot\mathbf{v}\ ds - \int_\Omega p\ \mathrm{div}(\mathbf{v})\ d\mathbf{x} &= 0, \qquad (2.10) \\ \int_\Omega q\ \mathrm{div}(\mathbf{u})\ d\mathbf{x} &= 0, \qquad (2.11) \end{aligned}$$

In the weak formulation (2.10) - (2.11), we have to take into account the boundary conditions on the inlet and outlet parts of the boundary. Recall that in this thesis we are interested in simulating fluid flows in straight and curved pipes and in realistic geometries of blood vessels. In all these cases, the computational domain is only a part of the physical one. Therefore, inlets and outlets are in fact artificial sections that separate the computational and physical domains.

Consequently, in order not to change the physics of the problem, special attention needs to be given on the boundary conditions imposed at the inlets or outlets.

In the following, let us consider different types of boundary conditions for the inlet and outlet sections. We start by the most classical boundary condition, the case where we know the velocity profiles at inlet and outlet (for example, Poiseuille profiles). The second case we consider is the free outlet condition. Finally, we focus on less classical boundary conditions that correspond to the case where we know only the pressure at the inlet and outlet sections. These nonstandard boundary conditions are very useful if we want to make a comparison between the simulations performed and the experimental data. Indeed, in physical experiments, it might be convenient to impose pressure than velocity on both inlet and outlet.

2.2 Boundary conditions

2.2.1 Dirichlet-Dirichlet boundary conditions

Let us suppose that we know the velocity profiles at the inlets and outlets and that they are described by two functions $\mathbf{u}_{in} \in [H^{\frac{1}{2}}(\Gamma_{in})]^d$ and $\mathbf{u}_{out} \in [H^{\frac{1}{2}}(\Gamma_{out})]^d$ such that

$$\mathbf{u} = \mathbf{u}_{in} \quad \text{on } \Gamma_{in}, \tag{2.12}$$

$$\mathbf{u} = \mathbf{u}_{out} \quad \text{on } \Gamma_{out}. \tag{2.13}$$

In this case, in order to have a well posed problem, it is sufficient to choose

$$\mathbb{V} = \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_w,\ \mathbf{v} = \mathbf{u}_{in} \text{ on } \Gamma_{in},\ \mathbf{v} = \mathbf{u}_{out} \text{ on } \Gamma_{out}\} \tag{2.14}$$

and $\mathbb{M} = L^2_0(\Omega)$, where $L^2_0(\Omega)$ denotes the set of functions in $L^2(\Omega)$ with zero mean value.

With this choice of functional spaces, problem (2.10) - (2.11) becomes: find $(\mathbf{u}, p) \in \mathbb{V} \times \mathbb{M}$ such that $\forall \mathbf{v} \in [H_0^1(\Omega)]^d, \forall q \in L_0^2(\Omega)$

$$\int_\Omega \rho \frac{\mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v})\ d\mathbf{x} - \int_\Omega p\ \mathbf{div}(\mathbf{v})\ d\mathbf{x} = 0, \quad (2.15)$$

$$\int_\Omega q\ \mathrm{div}(\mathbf{u})\ d\mathbf{x} = 0, \quad (2.16)$$

The additional restriction of zero mean value to $L^2(\Omega)$ function allows to uniquely define the pressure in $\mathbb{M}$ and may be integrated in the variational formulation (2.15) - (2.16) by adding a suitable Lagrange multiplier. The final variational formulation, with the Lagrange multiplier, reads as: find $(\mathbf{u}, p, \zeta) \in \mathbb{V} \times \mathbb{M} \times \mathbb{R}$ such that $\forall \mathbf{v} \in [H_0^1(\Omega)]^d$, $\forall q \in \mathbb{M}$, $\forall \xi \in \mathbb{R}$

$$\int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v})\ d\mathbf{x} - \int_\Omega p\ \mathbf{div}(\mathbf{v})\ d\mathbf{x} = 0, \quad (2.17)$$

$$\int_\Omega (q\ \mathrm{div}(\mathbf{u}) + \xi q)\ d\mathbf{x} = 0, \quad (2.18)$$

$$\int_\Omega p\xi\ d\mathbf{x} = 0. \quad (2.19)$$

Remark 2. *In the last formulation, the Dirichlet boundary conditions were set strongly. There are various strategies for this operation. The most coherent approach with the theory consists of eliminating the rows and columns that belong to the nodes associated to the Dirichlet boundary, by setting them to zero except on the diagonal which will be set to 1, the corresponding entry in the right-hand side will be set to* $\mathbf{u_{in}}$*. Notice that this operation keeps the symmetric aspect of the matrix but changes its pattern. To avoid this inconvenient, we can act on only the corresponding row by annihilating the extra-diagonal row elements and setting the diagonal one to 1 without modifying the column elements. In the following we will denote the first operation by "***elimination_symmetric***".*
*Note that the value on the diagonal element corresponding to a Dirichlet node can be left unchanged in some cases. We will refer to this option by the suffix "***keep-diagonal***" e.g.* **–on.type=elimination_symmetric_keep_diagonal***".*

Remark 3. *Another way of setting the Dirichlet boundary conditions is using a weak treatment. It consists of writing the equations in the conservative form, adding terms to ensure consistency, symetrizing to ensure adjoint consistency and adding a penalization term with factor* $\gamma(\mathbf{u} - \mathbf{u_{in}})/h$ *that ensures that the solution will be set to the proper value sat the boundary.*
The main advantages of this treatment are that first, it is uniform for all boundary conditions type, and second if the boundary conditions are time independent the terms are assembled once for all unlike the strong treatment that have to be reapplied on the matrix each time we have a term that is time dependent. The main disadvantage is that we will be introducing the penalization parameter γ *that needs to be tweaked so that the coercivity property is still satisfied.*

2.2.2 Dirichlet-Neumann boundary conditions

The previous Dirichlet-Dirichlet boundary conditions do not correspond to the most common real situation because, in general, we do not know exactly the velocity profile at

the outlet sections, even if we assume that we know it at the inlets. Indeed, it is difficult to predict the velocity profile at outlets since it depends on the channel geometry or the number of outlets sections in a vessel network, for example. Therefore, we can consider the so-called *non-homogenous Neumann* boundary conditions. These boundary conditions can be formalized as

$$\mathbf{u} = \mathbf{u}_{in} \qquad \text{on } \Gamma_{in}, \tag{2.20}$$
$$\boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} = \mathbf{g}_N \quad \text{on } \Gamma_{out}. \tag{2.21}$$

In this case, we retrieve the following variational formulation: find $(\mathbf{u}, p) \in \mathbb{V} \times \mathbb{M}$ such that $\forall \mathbf{v} \in \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_w \cup \Gamma_{in}\}$, $\forall q \in \mathbb{M}$

$$\int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v})\, d\mathbf{x} - \int_{\Gamma_{out}} \mathbf{g}_N\boldsymbol{n}\cdot\mathbf{v}\, ds - \int_\Omega p\ \mathrm{div}(\mathbf{v})\, d\mathbf{x} = 0 \tag{2.22}$$
$$\int_\Omega q\ \mathrm{div}(\mathbf{u})\, d\mathbf{x} = 0 \tag{2.23}$$

with:

$$\mathbb{V} = \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_w,\ \mathbf{v} = \mathbf{u}_{in} \text{ on } \Gamma_{in}\} \quad \text{and} \quad \mathbb{M} = L^2_0(\Omega). \tag{2.24}$$

It appears that this choice of boundary conditions is not in agreement with our problem since the numerically computed velocity field is not necessarily orthogonal to the outlet section. See [98] for some studies on these boundary conditions in the framework of human lung modeling and some numerical experiments showing this defect or [121] for some numerical examples in the case of blood flow simulation.

2.2.3 Neumann-Neumann boundary conditions

Boundary conditions involving the pressure and the stress tensor To overcome the difficulties related to the free outlet boundary conditions, we assume now that we know the exact value of the normal stress tensor at inlets and outlets. Due to the definition of the stress tensor, it is sufficient to know both the pressure and the velocity at inlets and outlets to enforce such boundary conditions. They read as

$$\boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} = \boldsymbol{\sigma}_{in}\boldsymbol{n} = -p_{in}\boldsymbol{n} + 2\mu\mathbf{D}(\mathbf{u_{in}})\boldsymbol{n}, \qquad \text{on } \Gamma_{in}, \tag{2.25}$$
$$\boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} = \boldsymbol{\sigma}_{out}\boldsymbol{n} = -p_{out}\boldsymbol{n} + 2\mu\mathbf{D}(\mathbf{u_{out}})\boldsymbol{n}, \quad \text{on } \Gamma_{out}. \tag{2.26}$$

The corresponding variational formulation is then written as: find $(\mathbf{u}, p) \in \mathbb{V} \times \mathbb{M}$ such that

$$\int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{v} + 2\mu\int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v})\, d\mathbf{x} - \int_\Omega p\ \mathbf{div}(\mathbf{v})\, d\mathbf{x} = \int_{\Gamma_{in}} \boldsymbol{\sigma}_{in}\boldsymbol{n}\cdot\mathbf{v}\, ds + \int_{\Gamma_{out}} \boldsymbol{\sigma}_{out}\boldsymbol{n}\cdot\mathbf{v}\, ds, \quad \forall \mathbf{v} \in \mathbb{V} \tag{2.27}$$
$$\int_\Omega q\ \mathrm{div}(\mathbf{u})\, d\mathbf{x} = 0, \quad \forall q \in \mathbb{M} \tag{2.28}$$

where

$$\mathbb{M} = L^2(\Omega) \quad \text{and} \quad \mathbb{V} = \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_w\} \tag{2.29}$$

Boundary conditions involving the pressure without the stress tensor It is obvious that in the most common situations, the normal stress tensor is not known at the inlets or outlets. Nevertheless, in some physical experiments, we have direct access to the pressure at inlets and outlets because it is imposed by the experience. Therefore, we suppose the existence of two functions $p_{in} \in H^{-\frac{1}{2}}(\Gamma_{in})$ and $p_{out} \in H^{-\frac{1}{2}}(\Gamma_{out})$ such that

$$\boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} = -p_{in}\boldsymbol{n} \quad \text{on } \Gamma_{in} \tag{2.30}$$

$$\boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} = -p_{out}\boldsymbol{n} \quad \text{on } \Gamma_{out} \tag{2.31}$$

The corresponding variational formulation is: find $(\mathbf{u}, p) \in \mathbb{V} \times \mathbb{M}$ such that $\forall \mathbf{v} \in \mathbb{V}$, $\forall q \in \mathbb{M}$,

$$\int_\Omega \rho \frac{\partial \mathbf{u}}{\partial t}\mathbf{v} + \int_\Omega \rho(\mathbf{u} \cdot \nabla \mathbf{u}) \cdot \mathbf{v} + 2\mu \int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v}) \, d\mathbf{x} - \int_\Omega p \, \mathbf{div}(\mathbf{v}) \, d\mathbf{x} = - \int_{\Gamma_{in}} p_{in}\boldsymbol{n} \cdot \mathbf{v} \, ds - \int_{\Gamma_{out}} p_{out}\boldsymbol{n} \cdot \mathbf{v} \, ds \tag{2.32}$$

$$\int_\Omega q \, \mathrm{div}(\mathbf{u}) \, d\mathbf{x} = 0, \tag{2.33}$$

where $\mathbb{M} = L^2(\Omega)$ and $\mathbb{V} = \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_w\}$.

The latest variational formulation gives an appropriate framework for imposing boundary conditions on the pressure. However, this option may only make sense if the stress tensor is diagonal at inlets and outlets, meaning that the computational domain Ω is separated by inlets and outlets from a perfect gas-like media at pressure p_{in} and p_{out} respectively. However, this is obviously not the case for blood flow simulations, where Ω is the computational domain, corresponding to a part of the circulatory system and this is why this formulation cannot be used in our case. For example, the formulation (2.32) - (2.33) cannot even recover the Poiseuille flow in a straight pipe.

Remark 4. *We must mention that the weak formulation of a Stokes problem with the same type of boundary conditions is retrieved in the same way as described above for the Navier-Stokes problem.*

2.2.4 Other types of boundary conditions

In the following, we do a brief overview on other types of boundary conditions for hemodynamic flow that we did not use in the context of this thesis.
Significant progress has been made in terms of physiological boundary conditions. For instance, Stergiopulos et al., in 1992, used a lumped parameter model of the vasculature downstream of each branch in his numerical model [132]. When zero mean pressure or equal pressures or tractions are the common choice of the pressure boundary at the

outlets, the resulting flow split will be dictated solely by the resistance to flow in the branches of the domain of interest, neglecting the dominant effect of the resistance of the downstream vascular beds. An alternative approach is to use 3D models for the major arteries where high-fidelity information is needed, and reduced order models for the rest of the system. While closed-loop models are optimal, a simpler approach is to directly represent the vasculature of the small arteries and arterioles using 0D or 1D models. Several groups [137, 132, 58, 149, 39, 126, 128, 112, 93, 114] have successfully coupled 3D models to either resistances or more sophisticated 0D models (lumped models), first introduced by Formaggia et al. in [56, 57]. Another method, well suited for handling complex geometries and boundary conditions inherent in modeling blood flow was introduced in [138] and extended in [147]. It is based on the Dirichlet-to-Neumann (DtN) [66] and the variational multiscale [80] methods and is an extension of the 1D coupled multidomain approach and can be applied with a variety of models of the downstream domain. In [36, 68] a natural normal velocity boundary condition formulation was introduced. These boundary conditions force the velocity to be normal to the outlet since a zero Dirichlet velocity is imposed for the tangential directions. It has the disadvantage of directly modify the local flow fields, in particular when there are eddies which cross the boundary. For more information the reader may refer to [147, 60].

2.3 Space discretization

Remark 5. *We must mention that, although the existence of a unique 3D strong solution on* $(0,T)$ *is proven for sufficiently small data, e.g.,* $\|\nabla \mathbf{u}_0\|_{L^2(\Omega)}$ *small enough (global-in-time unique solution), or on sufficiently short intervals of time,* $0 \leq t \leq T$ *because of a good energy balance due to the conservation property of the nonlinear term, the uniqueness of a weak solution on* $(0,T)$ *is still an open problem. For 2D problems with essential boundary conditions, the control of the inertial term guarantees the uniqueness of a weak solution on any time interval* $(0,T)$*. Besides, it is also a strong solution if the data of the problem are sufficiently smooth. For more details the reader may refer to [94, 96, 140].*

Remark 6. *In the following, we will consider the Dirichlet-Neuman formulation* (2.22) - (2.23) *in order to show the treatment of all the terms that appear in these equations. We denote by* $\Gamma_D = \Gamma_w \cup \Gamma_{in}$ *the portions of the boundary where the Dirichlet condition* ($\mathbf{u} = \mathbf{g}_D$) *are applied and* $\Gamma_N = \Gamma_{out}$ *the portions of the boundary where the Neumann condition* ($\boldsymbol{\sigma}(\mathbf{u},p)\boldsymbol{n} = \mathbf{g}_N$) *are applied.*

Remark 7. *Let us denote:*

$$a(\mathbf{u},\mathbf{v}) = 2\mu \int_\Omega \mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{v}), \tag{2.34}$$

$$c(\mathbf{w},\mathbf{u},\mathbf{v}) = \rho \int_\Omega (\mathbf{w}\cdot\nabla\mathbf{u})\cdot\mathbf{v}, \tag{2.35}$$

$$b(\mathbf{v},p) = -\int_\Omega p\ \mathbf{div}(\mathbf{v}) \tag{2.36}$$

The weak formulation (2.22) - (2.23) *can be written as: find* $(\mathbf{u},p) \in \mathbb{V}\times\mathbb{M}$ *such that* $\forall \mathbf{v} \in \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0}$ *on* $\Gamma_D\}$, $\forall q \in \mathbb{M}$

$$\rho(\frac{\partial \mathbf{u}}{\partial t}, \mathbf{v}) + c(\mathbf{u}, \mathbf{u}, \mathbf{v}) + a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) = \int_{\Gamma_N} \mathbf{g} \cdot \mathbf{v} \quad (2.37)$$

$$b(\mathbf{u}, q) = 0 \quad (2.38)$$

with:

$$\mathbb{V} = \{\mathbf{v} \in [H^1(\Omega)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_w,\ \mathbf{v} = \mathbf{u}_{in} \text{ on } \Gamma_{in}\} \quad \text{and} \quad \mathbb{M} = L^2_0(\Omega). \quad (2.39)$$

We present in the sequel the discretisation strategy, following the framework presented in [30] and introduced in the preliminaries section of this thesis. Let δ be a discretization parameter. We define $\hat{K} \subset \mathbb{R}^d$ $(d = 1, 2, 3)$ a reference elementary convex, *e.g.* a simplex or a hypercube. We denote by $\mathcal{T}_\delta$ a finite collection of nonempty, disjoint open simplices or hypercubes $\mathcal{T}_\delta \equiv \mathcal{T}_{(h,k)} = \{K = \boldsymbol{\varphi}^{\text{geo}}_{K,k}(\hat{K})\}$, forming a partition of Ω such that $h = \max_{K \in \mathcal{T}_\delta} h_K$, with h_K denoting the diameter of the element $K \in \mathcal{T}_\delta$ and $\boldsymbol{\varphi}^{\text{geo}}_{K,k}$ is the polynomial of degree k that maps $\hat{K}$ to K which is also called the geometric transformation. The partition $\mathcal{T}_\delta$ induces a discretization of Ω, denoted Ω_δ, defined as the union of the closure of all elements in this partition. Note that if Ω is a polyhedral domain, then $\Omega_\delta = \Omega$. Following these notations, we denote $\Gamma_{in,\delta}$, $\Gamma_{out,\delta}$, $\Gamma_{w,\delta}$ the discretization of $\Gamma_{in}, \Gamma_{out}, \Gamma_w$, respectively.

We say that a hyperplanar closed subset F of $\overline{\Omega_\delta}$ is a *mesh face* if it has positive $(d-1)$-dimensional measure and if either there exist K_1, $K_2 \in \mathcal{T}_\delta$ such that $F = \partial K_1 \cap \partial K_2$ (in this case F is called an *internal face*) or there exists $K \in \mathcal{T}_\delta$ such that $F = \partial K \cap \partial \Omega_\delta$ (and F is called a *boundary face*). Internal faces are collected in the set $\mathcal{F}^i_\delta$, boundary faces in $\mathcal{F}^b_\delta$ and we let $\mathcal{F}_\delta := \mathcal{F}^i_\delta \cup \mathcal{F}^b_\delta$. For all $F \in \mathcal{F}_\delta$, we define $\mathcal{T}_F := \{K \in \mathcal{T}_\delta \mid F \subset \partial K\}$. For every interface $F \in \mathcal{F}^i_\delta$ we introduce two associated normals to the elements in $\mathcal{T}_F$ and we have $\boldsymbol{n}_{K_1,F} = -\boldsymbol{n}_{K_2,F}$, where $\boldsymbol{n}_{K_i,F}$, $i \in \{1, 2\}$, denotes the unit normal to F pointing out of $K_i \in \mathcal{T}_F$. On a boundary face $F \in \mathcal{F}^b_\delta$, $\boldsymbol{n}_F = \boldsymbol{n}_{K,F}$ denotes the unit normal pointing out of Ω_δ.

Without loss of generality we suppose from now on that we work with simplicial elements. Given a positive integer N, we denote by $\mathbb{P}^N(\hat{K})$ and $\mathbb{P}^N(K)$ the spaces of polynomials of total degree $\leq N$ defined in $\hat{K}$ and K, respectively. We define $P^N_c(\Omega_\delta \equiv \Omega_{(h,k)})$ and $[P^N_c(\Omega_\delta \equiv \Omega_{(h,k)})]^d$ with $k \geqslant 1$:

$$P^N_c(\Omega_\delta) = \{v \in C^0(\Omega_\delta) \mid v \circ \boldsymbol{\varphi}^{\text{geo}}_{K,k} \in \mathbb{P}^N(\hat{K})\ \forall K \in \mathcal{T}_\delta\}, \quad [P^N_c(\Omega_\delta)]^d = \prod_1^d P^N_c(\Omega_\delta). \quad (2.40)$$

Let us denote:

$$\begin{aligned} H^1_{(\mathbf{g}_D, \Gamma_{D,\delta})} &= \{f \in H^1(\Omega_\delta) / f|_{\Gamma_{D,\delta}} = \mathbf{g}_D\} \\ V_\delta &= \{v \in H^1_{(\mathbf{g}_D, \Gamma_{D,\delta})}(\Omega_\delta) \cap [P^M_c(\Omega_\delta)]^d\} \\ V_{\delta,0} &= \{v \in H^1_{(\mathbf{0}, \Gamma_{D,\delta})}(\Omega_\delta) \cap [P^M_c(\Omega_\delta)]^d\} \\ Q_\delta &= \{v \in P^N_c(\Omega_\delta)\} \end{aligned}$$

the discrete spaces associated to the velocity and the pressure, respectively.

Remark 8. *In the case of a Stokes problem, in order to ensure the existence, uniqueness and the stability of a solution of the abstract saddle point problem, the spaces V_δ and Q_δ must satisfy the so called inf-sup condition:*

$$\exists \beta_\delta > 0 \mid \quad \inf_{q_\delta \in Q_\delta} \sup_{\mathbf{v}_\delta \in V_\delta} \frac{\int_{\Omega_\delta} q_\delta \nabla \cdot \mathbf{v}_\delta}{\|q_\delta\|_{0,\Omega_\delta} \|\mathbf{v}_\delta\|_{1,\Omega_\delta}} \geq \beta_\delta \tag{2.41}$$

Besides, we have the following error estimations:

$$\|\mathbf{u} - \mathbf{u}_\delta\|_{1,\Omega} \leq c_{1\delta} \inf_{\mathbf{v}_\delta \in V_\delta} \|\mathbf{u} - \mathbf{v}_\delta\|_{1,\Omega} + c_{2\delta} \inf_{q_\delta \in Q_\delta} \|p - q_\delta\|_{0,\Omega}$$

$$\|p - p_\delta\|_{0,\Omega} \leq c_{3\delta} \inf_{\mathbf{v}_\delta \in V_\delta} \|\mathbf{u} - \mathbf{v}_\delta\|_{1,\Omega} + c_{4\delta} \inf_{q_\delta \in Q_\delta} \|p - q_\delta\|_{0,\Omega}$$

with

$$\begin{aligned}
c_{1\delta} &= (1 + \frac{\|a\|_{V_\delta,V_\delta}}{\alpha})(1 + \frac{\|b\|_{V_\delta,Q_\delta}}{\beta_\delta}), \alpha \text{ being the coercivity constant for } a \\
c_{2\delta} &= \frac{\|b\|_{V_\delta,Q_\delta}}{\alpha} \\
c_{3\delta} &= c_{1\delta} \frac{\|a\|_{V_\delta,V_\delta}}{\beta_\delta} \\
c_{4\delta} &= 1 + \frac{\|b\|_{V_\delta,Q_\delta}}{\beta_\delta} + c_{2\delta} \frac{\|a\|_{V_\delta,V_\delta}}{\beta_\delta}
\end{aligned}$$

When the inf-sup condition is violated, for example when $M = N$, the spaces are said to be unstable or incompatible. There are many ways to resolve this issue; one of them is to add stabilization terms by SUPG Streamline Upwind Petrov Galerkin or GLS Generalized Least Squares techniques. For general discussion on stabilization techniques, the reader can refer e.g. to [24].

We will consider, in the following, the stable generalized Taylor-Hood finite elements for the velocity-pressure discretisation, that is to say, we look for the velocity in $[P_c^{N+1}(\Omega_{(h,k_{\text{geo}})})]^d$ and the pressure in $P_c^N(\Omega_{(h,k_{\text{geo}})})$ [131]. We shall use from now on the notation $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_{k_{\text{geo}}}$ to specify exactly the discretisation spaces used for the velocity, pressure and geometry, respectively. The resulting approximate velocity and pressure fields are denoted by $\mathbf{u}_\delta$ and p_δ, respectively.

Remark 9. *In the case of a discrete Stokes problem, for $N \geq 2$, the $\mathbb{P}_N\mathbb{P}_{N-1}\mathbb{G}_1$ finite elements ($\mathbb{P}_N$ for velocity and $\mathbb{P}_{N-1}$ for pressure) as well as the $\mathbb{Q}_N/\mathbb{Q}_{N-1}$ finite elements ($\mathbb{Q}_N$ for velocity and $\mathbb{Q}_{N-1}$ for pressure) are compatible in two and three dimensions. When the exact solution is smooth enough, these elements yield the errors estimates in the case of a Stokes problem:*

$$\|\mathbf{u} - \mathbf{u}_\delta\|_{0,\Omega} + \|\mathbf{u} - \mathbf{u}_\delta\|_{1,\Omega} + \|p - p_\delta\|_{0,\Omega} \leq h^{N+1}(\|\mathbf{u}\|_{N+1,\Omega} + \|p\|_{N,\Omega}) \tag{2.42}$$

Proofs and further insight can be found in A. Ern and J.-L. Guermond [[50], Chapter 4], and F. Brezzi and M. Fortin [[24], Chapter 4] or V. Girault and P.A. Raviart [[144], Chapter 2].

Let us return to the Navier-Stokes discrete problem, the weak formulation then reads to: find $(\mathbf{u}_\delta, p_\delta) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H^1(\Omega_\delta)]^d \cap [P_c^M(\Omega_\delta)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{D,\delta}\}$, $\forall q_\delta \in Q_\delta$

$$\begin{aligned}\int_{\Omega_\delta} \rho \frac{\partial \mathbf{u}_\delta}{\partial t} \mathbf{v}_\delta \, d\mathbf{x} + \int_{\Omega_\delta} \rho(\mathbf{u}_\delta \cdot \nabla \mathbf{u}_\delta) \cdot \mathbf{v}_\delta \, d\mathbf{x} + 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta) : \mathbf{D}(\mathbf{v}_\delta) \, d\mathbf{x} \\ - \int_{\Gamma_{N,\delta}} \boldsymbol{\sigma}(\mathbf{u}_\delta, p_\delta) \boldsymbol{n}_\delta \cdot \mathbf{v}_\delta \, ds - \int_{\Omega_\delta} p_\delta \, \mathrm{div}(\mathbf{v}_\delta) \, d\mathbf{x} &= 0 \\ \int_{\Omega_\delta} q_\delta \, \mathrm{div}(\mathbf{u}_\delta) \, d\mathbf{x} &= 0\end{aligned}$$

2.4 Time discretization

The simulation will run up to a time T and so will cover the $I = [0, T]$ interval which we subdivide into N sub-intervals $I^k = (t^k, t^{k+1})$ with $k = 1, ..., N$ and where $t^{k+1} - t^k = \Delta t$ the time step assumed constant over time.
We denote by $(\mathbf{u}_\delta^k, p_\delta^k)$ the approximate solution at time t^k. Then the weak formulation can be written as: find $(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H^1(\Omega_\delta)]^d \cap [P_c^M(\Omega_\delta)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{D,\delta}\}$, $\forall q_\delta \in Q_\delta$

$$\begin{aligned}\int_{\Omega_\delta} \rho \frac{\partial \mathbf{u}_\delta^{k+1}}{\partial t} \mathbf{v}_\delta \, d\mathbf{x} + \int_{\Omega_\delta} \rho(\mathbf{u}_\delta^* \cdot \nabla \mathbf{u}_\delta^{**}) \cdot \mathbf{v}_\delta \, d\mathbf{x} + 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta^{\mathbf{k+1}}) : \mathbf{D}(\mathbf{v}_\delta) \, d\mathbf{x} \\ - \int_{\Gamma_{N,\delta}} \boldsymbol{\sigma}(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \boldsymbol{n}_\delta \cdot \mathbf{v}_\delta \, ds - \int_{\Omega_\delta} p_\delta^{k+1} \, \mathrm{div}(\mathbf{v}_\delta) \, d\mathbf{x} &= 0 \\ \int_{\Omega_\delta} q_\delta \, \mathrm{div}(\mathbf{u}_\delta^{k+1}) \, d\mathbf{x} &= 0\end{aligned} \tag{2.43}$$

We first start by discretizing the time derivative of the velocity by choosing an implicit scheme, the so-called *backward differentiation formulation* (BDF) [115]. The choice of this scheme is motivated by the fact that the BDF discretisation methods are zero-stable (a perturbation in the starting values of size ϵ causes the numerical solution over that time interval to change by no more than $K\epsilon$ for some value of K which does not depend on the step size), and the regions of absolute stability are unbounded (see Figure 2.1).

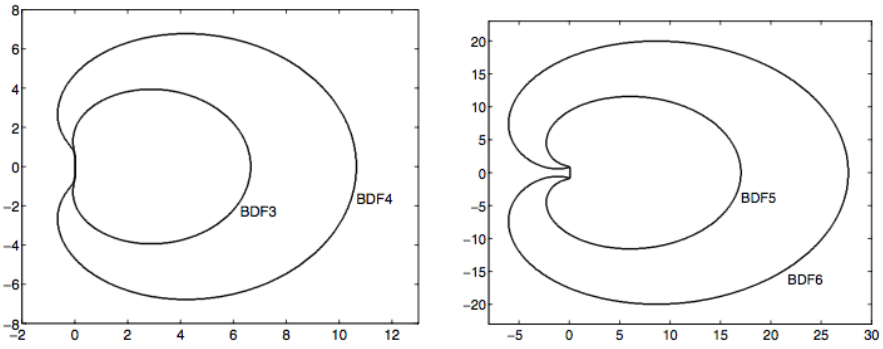


Figure 2.1: BDF absolute stability contours.[115]

These schemes can be written up to an arbitrary order q, thus we can denote them by BDF_q.

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 2.1: BDF_q coefficients up to $q = 4$.

The following formula describe the approximation of the time derivative of the velocity for an arbitrary order q using the β_j coefficients of table 2.1:

$$\frac{\partial \mathbf{u}_\delta^{k+1}}{\partial t} \approx \frac{\beta_{-1}}{\Delta t}\boldsymbol{u}_\delta^{k+1} - \sum_{j=0}^{q-1} \frac{\beta_j}{\Delta t}\boldsymbol{u}_\delta^{k-j} \tag{2.44}$$

As for the expression of $\mathbf{u}_\delta^*$ and $\mathbf{u}_\delta^{**}$ in the non-linear convective term, they can be chosen as follows:

$$\mathbf{u}_\delta^* \cdot \nabla \mathbf{u}_\delta^{**} = \begin{cases} \mathbf{u}_\delta^k \cdot \nabla \mathbf{u}_\delta^K, & \text{fully explicit treatment} \\ \mathbf{u}_\delta^k \cdot \nabla \mathbf{u}_\delta^{k+1}, & \text{standard semi-implicit treatment} \\ \mathbf{u}_\delta^{k+1} \cdot \nabla \mathbf{u}_\delta^{k+1}, & \text{fully implicit treatment (non-linear system)} \end{cases} \tag{2.45}$$

Remark 10. *In the following, we will use the first order BDF for simplicity reasons, to reduce the length of the equations.The details about the different convective term treatments are shown in the sequel.*

2.4.1 Fully explicit treatment

The weak formulation corresponding to the fully explicit treatment with homogeneous Dirichlet boundary condition reads to: find $(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H^1(\Omega_\delta)]^d \cap [P_c^N(\Omega_\delta)]^d\}$, $\forall q_\delta \in Q_\delta$

$$\begin{aligned} \rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^{k+1}}{\Delta t}\mathbf{v}_\delta \, d\mathbf{x} = -\rho \int_{\Omega_\delta} (\mathbf{u}_\delta^k \cdot \nabla \mathbf{u}_\delta^k) \cdot \mathbf{v}_\delta \, d\mathbf{x} - 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta^{\mathbf{k}}) : \mathbf{D}(\mathbf{v}_\delta) \, d\mathbf{x} \\ + \int_{\Omega_\delta} p_\delta^k \, \mathrm{div}(\mathbf{v}_\delta) \, d\mathbf{x} + \rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^k}{\Delta t} \cdot \mathbf{v}_\delta \, d\mathbf{x} \end{aligned} \tag{2.46}$$

$$\int_{\Omega_\delta} q_\delta \mathrm{div}(\mathbf{u}_\delta^{k+1}) \, d\mathbf{x} = 0 \tag{2.47}$$

The first equation can be trivially solved for $\mathbf{u}_\delta^{k+1}$, however, the fundamental problem with this approach is that the new velocity $\mathbf{u}_\delta^{k+1}$ does not, in general, satisfy the mass conservation equation. Moreover, there is no natural computation of p_δ^{k+1}. A possible

remedy is to replace the pressure p_δ^k with p_δ^{k+1}, in the conservation of the momentum equation, which leaves two unknowns, $\mathbf{u}_\delta^{k+1}$ and p_δ^{k+1}, and hence requires a simultaneous solution of the two equations.

$$\begin{aligned}\rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^{k+1}}{\Delta t}\mathbf{v}_\delta \, d\mathbf{x} - \int_{\Omega_\delta} p_\delta^{k+1} \, \mathrm{div}(\mathbf{v}_\delta) \, d\mathbf{x} &= -\rho \int_{\Omega_\delta} (\mathbf{u}_\delta^k \cdot \nabla \mathbf{u}_\delta^k) \cdot \mathbf{v}_\delta \, d\mathbf{x} \\ - 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta^{\mathbf{k}}) : \mathbf{D}(\mathbf{v}_\delta) \, d\mathbf{x} + \rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^k}{\Delta t} \cdot \mathbf{v}_\delta \, d\mathbf{x} & \qquad (2.48) \\ \int_{\Omega_\delta} q_\delta \mathrm{div}(\mathbf{u}_\delta^{k+1}) \, d\mathbf{x} &= 0\end{aligned}$$

We can eliminate $\mathbf{u}_\delta^{k+1}$ by taking the divergence of (2.48) to obtain a Poisson equation for the pressure. This method corresponds to the semi-explicit time discretisation. However, there are no natural boundary conditions for p_δ^k. Hence, solving the so obtained Poisson equation for the pressure, and then finding $\mathbf{u}_\delta^{k+1}$ trivially from (2.48) is therefore not in itself a sufficient solution strategy. This new scheme is temporally stable provided the time step satisfies the following limitation:

$$\Delta t \leq C min(\frac{h^2}{\mu}, \frac{h}{max_{\mathbf{x}\in\Omega}|\mathbf{u}^k(\mathbf{x})|})$$

2.4.2 Fully implicit treatment

Rewriting the weak formulation corresponding to the fully implicit treatment with homogeneous Dirichlet boundary condition, it reads to: find $(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H_0^1(\Omega_\delta)]^d \cap [P_c^N(\Omega_\delta)]^d\}$, $\forall q_\delta \in Q_\delta$

$$\begin{aligned}\rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^{k+1}}{\Delta t}\mathbf{v}_\delta \, d\mathbf{x} + \rho \int_{\Omega_\delta} (\mathbf{u}_\delta^{k+1} \cdot \nabla \mathbf{u}_\delta^{k+1}) \cdot \mathbf{v}_\delta \, d\mathbf{x} + 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta^{\mathbf{k+1}}) : \mathbf{D}(\mathbf{v}_\delta) \, d\mathbf{x} & \\ - \int_{\Omega_\delta} p_\delta^{k+1} \, \mathrm{div}(\mathbf{v}_\delta) \, d\mathbf{x} = \rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^k}{\Delta t} \cdot \mathbf{v}_\delta \, d\mathbf{x} & \qquad (2.49) \\ \int_{\Omega_\delta} q_\delta \mathrm{div}(\mathbf{u}_\delta^{k+1}) \, d\mathbf{x} = 0 & \qquad (2.50)\end{aligned}$$

Two methods can be used in order to solve this type of non linear system; either a Newton-type method based on the computation of the Jacobian matrix, or a Picard-type method based on a fixed-point strategy.

Newton method for non-linear system solve A natural approach for solving such a non-linear algebraic system can be achieved by using the *Newton-Krylov* technique, that is by using a Krylov method (e.g. GMRES or BiCGStab) for solving the linear system that is obtained at each Newton iteration step. This approach entails three nested cycles:

- temporal iteration: $t^k \longrightarrow t^{k+1}$
- Newton iteration: $X_n^{k+1} \longrightarrow X_{n+1}^{k+1}$

$$DF(X_n^{k+1})(X_{n+1}^{k+1} - X_n^{k+1}) = -\mathcal{F}(X_n^{k+1})$$

With: $\mathcal{F}(X_n^{k+1})$ the residual resulting from (2.49), $X_n^{k+1} = \begin{pmatrix} (u_i)_{i=1..N_{V_\delta}}^{k+1} \\ (p_j)_{j=1..N_{Q_\delta}}^{k+1} \end{pmatrix}$ and $DF(X) = \dfrac{\partial F}{\partial X_i}$

- Krylov iteration $[X_n^{k+1}]_j \longrightarrow [X_n^{k+1}]_{j+1}$

We recall that, when using the Newton method, a full linearisation of the convective term is done.
In its algebraic form the problem would be written as follows:

$$\underbrace{\begin{pmatrix} \mathbf{F_u} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix}}_{\mathcal{A}} \begin{pmatrix} \tilde{\mathbf{U}} \\ \tilde{\mathbf{P}} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{G}} \\ \mathbf{0} \end{pmatrix} \tag{2.51}$$

where $\mathbf{F_u}$ is a function of the unknown velocity, and has the form $\mathbf{F_u} = \frac{\mathbf{Q_u}}{\Delta t} + \mathbf{A} + \mathbf{C}$, where $\mathbf{Q_u}$ is the mass matrix, $\mathbf{A}$ the stiffness matrix and $\mathbf{C}$ the matrix arising from the explicit treatment of the convective term.

Remark 11. *Note that during at the first Newton iteration ($n = 0$), the unknown that we are seeking to evaluate is $(\delta\mathbf{U}^{k+1}, \delta P^{k+1}) = (\mathbf{U}_{n+1}^{k+1} - \mathbf{U}_n^{k+1}, P_{n+1}^{k+1} - P_n^{k+1})$. Since $(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \in V_\delta \times Q_\delta$, then $(\delta\mathbf{U}^{k+1}, \delta P^{k+1}) \in V_{\delta,0} \times Q_\delta$. Thus the contribution of the Dirichlet boundary condition on the Dirichlet nodes is vanished. So, in order to take into consideration the Dirichlet boundary condition, we have to carefully choose the "initial guess". Ideally, the initial iterate is obtained by solving the corresponding discrete Stokes problem with the same initial Dirichlet conditions.*

Picard method for non-linear system solve: At each time step $k+1$, a Picard iteration for the nonlinear system is derived by lagging the convection coefficient in the quadratic term $(\mathbf{u}^{k+1} \cdot \nabla)\mathbf{u}^{k+1}$. For the steady-state problem, this lagging procedure starts with some initial guess $\mathbf{u}_{(0)}^{k+1}$ (satisfying the discrete incompressibility constraint) for the velocities and then constructs a sequence of approximate solutions $(\mathbf{u}_{(n)}^{k+1}, p^{k+1})$ by solving the corresponding linear Oseen problem:

$$-\mu\Delta\mathbf{u}_{(n+1)}^{k+1} + (\mathbf{u}_{(n)}^{k+1} \cdot \nabla)\mathbf{u}_{(n+1)}^{k+1}) + \nabla p^{k+1} = \mathbf{f} \text{ in } \Omega, \tag{2.52}$$

$$\nabla \cdot \mathbf{u}_{(n+1)}^{k+1} = 0 \text{ in } \Omega \tag{2.53}$$

If we use $\mathbf{u}^0 = \mathbf{0}$, the first iteration corresponds to the Stokes problem.
The solution $\mathbf{u}_{(n+1)}^{k+1}$ at time $k+1$ is chosen when $\|\delta\mathbf{u}^{k+1}\|_{L^2} = \|\mathbf{u}_{(n+1)}^{k+1} - \mathbf{u}_{(n)}^{k+1}\|_{L^2}$ reaches a certain given tolerance.

The main disadvantage of Newton's method is that its convergence depends on the viscosity parameter. For high Reynolds number, the initial guess needs to be carefully chosen, unlike the Picard method where the radius of the ball of convergence is higher. Another advantage of the Picard method is that it is easier to settle compared to the Newton method. However, the number of iterations needed to achieve a certain accuracy is lower in the case of the Newton method, a plus that can play for this method.

2.4.3 Semi-implicit treatment

The semi-implicit treatment is retrieved when choosing $\mathbf{u}_\delta^* \cdot \nabla \mathbf{u}_\delta^{**} = \mathbf{u}_\delta^k \cdot \nabla \mathbf{u}_\delta^{k+1}$ in (2.43). The weak formulation then reads: find $(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H^1(\Omega_\delta)]^d \cap [P_c^M(\Omega_\delta)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{D,\delta}\}$, $\forall q_\delta \in Q_\delta$

$$\rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^{k+1}}{\Delta t} \mathbf{v}_\delta \, d\mathbf{x} + \int_{\Omega_\delta} \rho(\mathbf{u}_\delta^k \cdot \nabla \mathbf{u}_\delta^{k+1}) \cdot \mathbf{v}_\delta \, d\mathbf{x} + 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta^{\mathbf{k+1}}) : \mathbf{D}(\mathbf{v}_\delta) \, d\mathbf{x} - \int_{\Omega_\delta} p_\delta^{k+1} \, \mathrm{div}(\mathbf{v}_\delta) \, d\mathbf{x} - \int_{\Gamma_{N,\delta}} \boldsymbol{\sigma}(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \boldsymbol{n}_\delta \cdot \mathbf{v}_\delta \, ds = \rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^k}{\Delta t} \cdot \mathbf{v}_\delta \, d\mathbf{x} \tag{2.54}$$

$$\int_{\Omega_\delta} q_\delta \, \mathrm{div}(\mathbf{u}_\delta^{k+1}) \, d\mathbf{x} = 0 \tag{2.55}$$

This is the standard semi-implicit treatment of the convective term. A more general semi-implicit treatment is the so called Oseen scheme where the convective term is extrapolated usually by choosing an extrapolation order equal to the BDF order to guarantee a proper convergence order.
In the following, let us consider a second order backward differential formula for the time derivative, and an Oseen scheme of second order for the convective term treatment. The weak formulation then reads: find $(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H^1(\Omega_\delta)]^d \cap [P_c^M(\Omega_\delta)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{D,\delta}\}$, $\forall q_\delta \in Q_\delta$

$$\rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^{k+1}}{\Delta t} \mathbf{v}_\delta \, d\mathbf{x} + \rho \int_{\Omega_\delta} ((2\mathbf{u}_\delta^k - \mathbf{u}_\delta^{k-1}) \cdot \nabla \mathbf{u}_\delta^{k+1}) \cdot \mathbf{v}_\delta \, d\mathbf{x} + 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta^{\mathbf{k+1}}) : \mathbf{D}(\mathbf{v}_\delta) \, d\mathbf{x} - \int_{\Gamma_{N,\delta}} \boldsymbol{\sigma}(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \boldsymbol{n}_\delta \cdot \mathbf{v}_\delta \, ds - \int_{\Omega_\delta} p_\delta^{k+1} \, \mathrm{div}(\mathbf{v}_\delta) \, d\mathbf{x} = \int_{\Omega_\delta} \mathbf{f}_\delta^{k+1} \cdot \mathbf{v}_\delta \, d\mathbf{x} + \rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^k}{\Delta t} \cdot \mathbf{v}_\delta \, d\mathbf{x}$$

$$\int_{\Omega_\delta} q_\delta \, \mathrm{div}(\mathbf{u}_\delta^{k+1}) \, d\mathbf{x} = 0$$

While the fully implicit scheme is unconditionally stable, the semi-implicit scheme has a stability restriction on the time step:

$$\Delta t \leq C \frac{h}{max_{\mathbf{x} \in \Omega} |\mathbf{u}^n(\mathbf{x})|}$$

2.4.4 Characteristics method

This method [27] is not implemented in Feel++, library that we use to perform our simulations in the context of this thesis. However, it was used in the context of the CEMRACS 2015 when we compared Feel++ CFD outputs to the Freefem++ CFD outputs (see Chapter 7).
In the previous section we have applied an implicit finite difference scheme in order to discretise the velocity time derivative term of the momentum equation of the Navier-Stokes problem. In this section, we will apply the same finite difference scheme but to

the so-called *material derivative* or the *Lagrangian derivative* of the velocity vector field $\left(\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right)$.
The backward Euler formula of the above quantity is given by:

$$\frac{D\mathbf{u}}{Dt}(\mathbf{x}) \approx \frac{\mathbf{u}^{k+1}(\mathbf{x}) - \mathbf{u}^k(\mathbf{x}_p)}{\Delta t}$$

where $\mathbf{x}_p = \mathbf{x} - \mathbf{u}^k(\mathbf{x})\Delta t + \mathcal{O}(\Delta t^2)$ is the *foot* at time t^k of the characteristic issuing from $\mathbf{x}$ at time t^{k+1}.
The momentum equation then reads:

$$\rho\frac{\mathbf{u}^{k+1}(\mathbf{x}) - \mathbf{u}^k(\mathbf{x}_p)}{\Delta t} - 2\mathbf{div}(\mu\mathbf{D}(\mathbf{u}^{k+1}(\mathbf{x}))) + \nabla p^{k+1}(\mathbf{x}) = \mathbf{f}^{k+1}(\mathbf{x}).$$

To follow backwards the trajectory $\mathbf{X}$ (or characteristics) under the action of the flow $\mathbf{u}(t)$ of a particle of fluid which is at a point $\mathbf{x}$ at time s, we will be led to solve, for $s = t^{k+1}$, the following system of ordinary differential equations:

$$\begin{aligned}\frac{d\mathbf{X}}{dt}(t;s,\mathbf{x}) &= \mathbf{u}(t,\mathbf{X}(t;s,\mathbf{x})), \quad t\in[t^k,t^{k+1}]\\ \mathbf{X}(t;s,\mathbf{x}) &= \mathbf{x}\end{aligned}$$

The momentum equation retrieved from the previous discretisation of the material derivative becomes:

$$\frac{\mathbf{u}^{k+1}(\mathbf{x}) - \mathbf{u}^k(\mathbf{x}_p)}{\Delta t} - \mu\Delta\mathbf{u}^{k+1}(\mathbf{x}) + \nabla p^{k+1}(\mathbf{x}) = \mathbf{f}^{k+1}(\mathbf{x})$$

That gives:

$$\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}.\nabla)\mathbf{u}\right)(t^n,x) \sim \frac{\mathbf{u}(t^{n+1},x) - \mathbf{u}(t^n, X^n(x))}{\Delta t} \tag{2.56}$$

with $X^n(x) = x - \mathbf{u}(t^n,x)\Delta t + \mathcal{O}(\Delta t^2)$.

We finally have:

$$\frac{\rho}{\Delta t}(\mathbf{u}^{n+1} - \mathbf{u}^n\circ X^n) - \mu\Delta\mathbf{u}^{n+1} + \nabla p^{n+1} = 0 \tag{2.57a}$$

$$\mathrm{div}(\mathbf{u}^{n+1}) = 0 \tag{2.57b}$$

The weak formulation can be written as: find $(\mathbf{u}_\delta^{n+1}, p_\delta^{n+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v}\in[H^1(\Omega_\delta)]^d \cap [P_c^{N+1}(\Omega_\delta)]^d \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{D,\delta}\}$, $\forall q_\delta \in Q_\delta$

$$\begin{aligned}\int_{\Omega_\delta}\frac{\rho}{\Delta t}\mathbf{u}_\delta^{k+1}\mathbf{v}_\delta - \int_{\Omega_\delta}\frac{\rho}{\Delta t}(\mathbf{u}_\delta^k\circ\mathbf{X}_\delta^k)\cdot\mathbf{v}_\delta + 2\mu\int_{\Omega_\delta} D(\mathbf{u}_\delta^{k+1}) : D(\mathbf{v}_\delta)&\\ - \int_{\Gamma_{N,\delta}}\boldsymbol{\sigma}(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1})\boldsymbol{n}_\delta\cdot\mathbf{v}_\delta\, ds - \int_{\Omega_\delta} p_\delta^{k+1}\,\mathrm{div}(\mathbf{v}_\delta) &= 0 \qquad (2.58)\\ \int_{\Omega_\delta} q_\delta\,\mathrm{div}(\mathbf{u}_\delta^{k+1}) &= 0 \qquad (2.59)\end{aligned}$$

This scheme is inconditionnally stable. Moreover, for a positive constant C independant of Δt , it satisfies the error estimate

$$\|\mathbf{u}(t^k) - \mathbf{u}^k\|_{L^2(\Omega)} \le C(h + \Delta t + \frac{h^2}{\Delta t}) \quad \forall k \ge 1$$

For more information about the characteristic-based time discretisation strategies for spectral methods, the reader can refer to [27].

Conclusion

Through this chapter, we reminded the governing equations of an incompressible fluid flow in a general setting, and explained in more details the difficulties related to the treatment of the boundary conditions. We explicitly wrote the different weak formulations corresponding to the different boundary conditions and different time discretisation schemes. We gave the pros and the cons of each of the schemes. The numerical convergence analysis with respect to the various variational formulation presented in this chapter is reported in chapter 5.

Chapter 3

An overview of the resolution methods

In this chapter, in order to fix terminology and notations, we provide a brief review of the resolution methods of linear systems of equations that are of generic form $\mathcal{A}\mathbf{x} = \mathbf{b}$ *. For more details, the reader may refer to the standard textbooks [49, 95, 118]. Our discussion is restricted to the methods we used in our numerical computational simulations, and mainly available in* PETSc*, such as i) the direct solver LU, ii) the relaxation iterative methods, iii) the Krylov methods, in particular the conjugate gradient method, the GMRES method and its variants, the GCR method, iv) the Multigrid methods, and v) the domain decomposition methods.We introduce, at the end of this chapter, some definitions and preliminaries on the preconditioning principle.*

Contents

1 Resolution methods

Let us point out that the algebraic system obtained in each of the previous time discretization treatment can be written as follows,

$$\underbrace{\begin{pmatrix} \mathbf{F_u} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix}}_{\mathcal{A}} \begin{pmatrix} \mathbf{U} \\ \mathbf{P} \end{pmatrix} = \begin{pmatrix} \mathbf{G} \\ \mathbf{0} \end{pmatrix} \tag{3.1}$$

where $\mathbf{F_u}$ coincides with the stiffness matrix $\mathbf{A}$ in the case of a Stokes problem, with $\frac{1}{\Delta t}\mathbf{Q_u} + \mathbf{A}$ in case of an explicit treatment of the convective term, and with $\frac{\mathbf{Q_u}}{\Delta t} + \mathbf{A} + \mathbf{C}$ in the case of a semi-implicit or an implicit treatment.
In this chapter, we review different classes of solution methods that can be used to solve the linear system (3.1).

1.1 Direct method

Direct methods [44] are based on the factorization of the coefficient matrix $\mathbf{A}$ into easily invertible matrices. They are widely used and are the solver of choice in many industrial codes, especially where reliability is the primary concern. The idea behind these methods started from the fact that each non singular matrix $\mathcal{A}$ can be written as an LU product with proper row and/or column orderings or permutations, where L is a lower triangular matrix ans U is an upper triangular matrix.
We recall theorem 6.2.1 in [5]:

Theorem 1. *Let $A = (a_{i,j})_{1 \leq i,j \leq n}$ be a matrix of order n all of whose diagonal submatrices of order k are nonsingular. There exists a unique pair of matrices (L, U), with U upper triangular and L lower triangular with a unit diagonal (i.e., $l_{i,i} = 1$), such that $A = LU$.*

Direct methods make use of Gaussian elimination to construct L and U matrices. Once constructed, solving the system $\mathcal{A}\mathbf{x} = b$, equivalently $LU\mathbf{x} = b$, involves two logical steps:

1. Solve the equation $L\mathbf{y} = b$ for $\mathbf{y}$;

2. Solve the equation $U\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$;

This process is called the forward and backward substitution, and requires $O(N^2)flops$, where N is the number of unknowns. The cost of the Gaussian elimination algorithm is $O(N^3)$.
A direct method is usually used when the matrix is dense because it leads to a more accurate solution with a fixed number of work compared to iterative methods and for systems with thousands of degrees of freedom.
Although they are very robust, they tend to require a predictable amount of resources in terms of time and storage [63], and scale poorly with problem size in terms of operation counts and memory requirements, especially on problems arising from the discretization of PDEs in 3D.

1.2 Iterative method

Let us suppose we need to solve a linear system $\mathcal{A}\mathbf{x} = b$.
We assume that $\mathcal{A}$ is a non singular sparse matrix, and b is a given vector. An iterative method is based on the construction of a sequence of vectors $(\mathbf{x})_{k,k=0,1,...}$ with ($\mathbf{x}_0$ given), which is expected to converge towards $\mathbf{x}$, when $k \to \infty$.
The advantages of iterative methods are *i*) the matrix $\mathcal{A}$ is not modified, no fill-in is generated and there is no need for additional space for more elements, *ii*) for large problems, iterative methods are faster then direct methods, *iii*) iterative methods are easy to implement. However, the convergence of iterative methods is not guaranteed for general matrices and they may require a lot of time if we are seeking a high accuracy.
There are four big families of iterative methods: *i*) the classical methods (SOR, Gauss Seidel, Jacobi) *ii*) the Krylov subspace methods (CG, Bi-CGSTAB, GMRES *etc.*), *iii*) multigrid method, and *iv*) domain decomposition methods.

1.2.1 Classical iterative methods: Relaxation methods

The relaxation methods are based on a splitting of the matrix $\mathcal{A}$ into a sum of three matrices: $\mathcal{A} = L + D + U$, where L the lower triangular part of the matrix $\mathcal{A}$, D the diagonal component of $\mathcal{A}$, and U the upper part of the matrix $\mathcal{A}$. The solution is then obtained iteratively via:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + P^{-1}(b - \mathcal{A}\mathbf{x}^{(k)}) \tag{3.2}$$

where P^{-1} is equal to D^{-1} in the case of the Jacobi method, and to $(D+L)^{-1}$ in the case of Gauss-Seidel method.
The element-wise formula for the Gauss–Seidel method is extremely similar to that of the Jacobi method. An advantage of the Gauss-Seidel method is that, unlike the Jacobi method, only one storage vector is required as elements can be overwritten as they are computed. However, also unlike the Jacobi method, the computations for each element cannot be done in parallel. Furthermore, the values at each iteration are dependent on the order of the original equations.
As for the SOR (Successive Over Relaxation) method, the linear system may be rewritten as:

$$(D + \omega L)\mathbf{x} = \omega\mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}$$

for a constant $\omega > 1$, called the relaxation factor.
Analytically, this may be written as:

$$\mathbf{x}^{(k+1)} = (D + \omega L)^{-1}\big(\omega\mathbf{b} - [\omega U + (\omega - 1)D]\mathbf{x}^{(k)}\big) = L_w\mathbf{x}^{(k)} + \mathbf{c}$$

where $\mathbf{x}^{(k)}$ is the k^{th} approximation or iteration of $\mathbf{x}$ and $\mathbf{x}^{(k+1)}$ is the next or $k+1$ iteration of $\mathbf{x}$.
We can notice that this method is a variant of the Gauss–Seidel method for $\omega = 1$. However it performs a faster convergence than the latter.
Iterative methods based on relaxation are generally used as smoothers in the multigrid method to damp high frequencies error.

1.2.2 Krylov subspace

We now turn to the most frequent iterative methods, the Krylov subspace methods.
Let us recall the iterative method (3.2), and denote $\mathbf{r_k} = b - \mathcal{A}\mathbf{x_k}$ the residual. If we start

with $\mathbf{x_0}$, the next steps can be written such as:

$$\begin{aligned} \mathbf{x_1} &= \mathbf{x_0} + P^{-1}\mathbf{r_0}, \\ \mathbf{x_2} &= \mathbf{x_1} + P^{-1}\mathbf{r_1} \\ &... \end{aligned}$$

Substituting $\mathbf{x_1}$ from the previous step and using $\mathbf{r_1} = b - \mathcal{A}\mathbf{x_1}$, leads to:

$$\mathbf{x_2} = \mathbf{x_0} + 2P^{-1}\mathbf{r_0} - P^{-1}\mathcal{A}P^{-1}\mathbf{r_0}$$

This is equivalent to :

$$\mathbf{x_k} \in \mathbf{x_0} + span\{P^{-1}\mathbf{r_0}, P^{-1}\mathcal{A}(P^{-1}\mathbf{r_0}), ..., (P^{-1}\mathcal{A})^{k-1}(P^{-1}\mathbf{r_0})\}$$

The Krylov subspace of dimension k, corresponding to the matrix $\mathcal{A}$ and initial residual r_0 is therefore defined as$K_k(\mathcal{A}; r_0) := span\{r_0, \mathcal{A}r_0, ..., \mathcal{A}^{k-1}r_0\}$.

Conjugate gradient method One of the most popular and academic Krylov methods is the conjugate gradient (CG) method, developed by Hestenes and Stiefel [79] for systems that are symmetric positive definite or Hermitian positive definite.

Definition 7. *Two non-zero vectors* $\mathbf{u}$ *and* $\mathbf{v}$ *are conjugate (with respect to* $\mathcal{A}$*) if*

$$\mathbf{u}^\mathsf{T}\mathcal{A}\mathbf{v} = 0.$$

Since $\mathcal{A}$ *is symmetric and positive definite, the left-hand side defines an inner product* $\langle \mathbf{u}, \mathbf{v} \rangle_\mathcal{A} := \langle \mathcal{A}\mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{u}, \mathcal{A}^\mathsf{T}\mathbf{v} \rangle = \langle \mathbf{u}, \mathcal{A}\mathbf{v} \rangle = \mathbf{u}^\mathsf{T}\mathcal{A}\mathbf{v}$. *Two vectors are conjugate if and only if they are orthogonal with respect to this inner product.*

Let $\mathbf{x}^*$ be the solution of the system $\mathcal{A}\mathbf{x} = \mathbf{b}$, the CG method is an orthogonal projection method that satisfies a minimality condition: the error is minimal in the so-called energy norm or $\mathcal{A}$-norm defined by:

$$\|\mathbf{d}\|_\mathcal{A} = \sqrt{\mathbf{d}^T\mathcal{A}\mathbf{d}}.$$

where $\mathbf{d} := \mathbf{x} - \mathbf{x}^*$ is the error vector. Note that the initial problem $\mathcal{A}\mathbf{x} = \mathbf{b}$ can be recasted as a minimization of a quadratic function:

$$f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T\mathcal{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

with a symmetric positive definite matrix $\mathcal{A}$. f is convex and has a unique minimum. Its gradient is

$$\nabla f(\mathbf{x}) = \mathcal{A}\mathbf{x} - \mathbf{b} = -\mathbf{r}$$

where $\mathbf{r}$ is the residual corresponding to $\mathbf{x}$. Hence,

$$\mathbf{x} \text{ minimizer of f} \iff \nabla f(\mathbf{x}) = \mathbf{0} \iff \mathcal{A}\mathbf{x} = \mathbf{b}$$

This means we are looking at the minimizer $\mathbf{x}^*$ of the energy norm of the error vector of the linear system $\mathcal{A}\mathbf{x} = \mathbf{b}$, that is the solution of the system $\mathcal{A}\mathbf{x} = \mathbf{b}$. The idea is to find this minimiser by descending on the surface representing f by following the direction of

steepest descent. Let us take the first basis vector $\mathbf{p_0}$ to be the negative of the gradient of f at $\mathbf{x} = \mathbf{x_0}$. Starting with a "guessed solution" $\mathbf{x_0}$, this means we take $\mathbf{p_0} = \mathbf{b} - \mathcal{A}\mathbf{x_0}$. The other vectors in the basis will be conjugate to the gradient, hence the name conjugate gradient method. Let $\mathbf{r}_k$ be the residual at the k^{th} step: $\mathbf{r}_k = \mathbf{r}_{k-1} - \mathbf{A}\mathbf{x}_k$. Note that $\mathbf{r}_k$ is the negative gradient of f at $\mathbf{x} = \mathbf{x_k}$, so the gradient descent method would be to move in the direction $\mathbf{r}_k$. Here, we insist that the directions $\mathbf{p}_k$ be conjugate to each other. This gives the following expression:

$$\mathbf{p}_k = \mathbf{r}_{k-1} + \beta_k \mathbf{p}_{k-1}$$

Following this direction, the next optimal location is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_{k+1}\mathbf{p}_{k+1}$$

with

$$\alpha_k = \frac{\mathbf{r}_{k-1}^T \mathbf{r}_{k-1}}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}.$$

The conjugate gradient algorithm can thus be written as follows:

Algorithm 1 Conjugate gradient method

Input $k = 0$, $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{r}_0$
while $\mathbf{r}_k \neq \mathbf{0}$ **do**
  $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
  $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$
  if $\mathbf{r}_{k+1}$ *is sufficiently small* **then**
    exit
  else
    $\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$
    $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k \quad k = k + 1$
  end
end

The CG method can theoretically be viewed as a direct method, as it produces the exact solution after a finite number of iterations, which is not larger than n, the size of the matrix, in the absence of round-off error. However, the CG method is unstable with respect to even small perturbations, e.g., most directions are not in practice conjugate, and the exact solution is never obtained. Fortunately, the CG method can be used as an iterative method as it provides monotonically improving approximations $\mathbf{x}_k$ to the exact solution, which may reach the required tolerance after a relatively small (compared to the problem size) number of iterations. For this we recall the convergence property of the CG method from [5] (*th.* 9.4.1 and *remark* 9.4.1).

Theorem 2. *Let $\mathcal{A}$ be a symmetric positive definite matrix. There exists indeed a unique vector $\mathbf{x}_{k+1} \in [\mathbf{x}_0 + K_k]$. Furthermore, this algorithm converges to the solution of the linear system $\mathcal{A}\mathbf{x} = \mathbf{b}$ in at most n iterations.*

Remark 12. *The conjugate gradient algorithm that we have devised as an iterative method is in fact a direct method, since it converges in a finite number of iterations (precisely $k_0 + 1$, where k_0 is the critical Krylov dimension defined in Lemma 9.3.2). However, in practice, we use it like an iterative method that (hopefully) converges numerically in fewer than $k_0 + 1 \leq n$ iterations.*

The biconjugate gradient (biCG) method [55] provides a generalization to non-symmetric matrices.

Generalised Minimal RESidual (GMRES) method: When no indication is given about the system's matrix, a classical choice of solver is the GMRES methods. The generalized minimal residual algorithm, developed by Saad and Schultz [120], is a Krylov subspace method based on long recurrences. It computes an approximation of the minimal of the residual and thus satisfies an optimality property. This method is used for non-symmetric (non)singular matrices. Despite being stable, an inconvenient, of this method is that the work per iteration and memory requirements increases for an increasing number of iterations. In order to avoid the problem of excessive storage requirements and computational costs for the orthogonalization, GMRES is usually restarted after m iterations, which uses the last iteration as starting vector for the next restart. The restarted GMRES is denoted as GMRES(m). Consequently, another problem seems to arise when using this method, namely that the convergence depends on the choice of m.The property of superlinear convergence is lost by throwing away all the previous information of the Krylov subspace. If no restart is used, GMRES (like any orthogonalizing Krylov subspace method) will converge in no more than N steps.
There are few variants of the GMRES method such as the FGMRES and GMRESR.
The Flexible GMRES, (FGMRES) method, proposed by Saad [119], is a generalization of GMRES that allows greater flexibility in the choice of solution subspace than GMRES specially for ill-posed problems. This method provides a unified approach to include user-specified vectors in the solution subspace that may determine approximate solutions of higher quality than commonly applied iterative methods. It has also a considerable impact on the choice of the preconditioner that can vary from one Krylov iteration to another.

GMRESR is developed by Vuik and van der Vorst in [145]. The idea is that the GMRES method can be effectively combined with other iterative schemes. The outer iteration steps are performed by GCR, an iterative method explained thereafter, while the inner iteration steps can be performed by GMRES or with any other iterative method.

Generalized Conjugate Residual (GCR) method: The GCR is a Krylov method dedicated to non-symmetric matrices [46]. With the Conjugate Residual method (CR) it is a variant of the (CG) method. In the Conjugate Gradient algorithm, the $\mathbf{p}_i$'s are $\mathbf{A}$-orthogonal, *i.e.*, conjugate, while in the CR method the residual vectors are $\mathbf{A}$-orthogonal, *i.e.*, conjugate. In addition, the vectors $\mathbf{A}\mathbf{p}_i$'s $i = 0, 1, ...,$ are orthogonal, *i.e.*, the $\mathbf{p}_i$'s are $\mathbf{A}^T\mathbf{A}$-orthogonal. The difference in the GCR method is that the next basis vector $\mathbf{p}_{i+1}$ is computed as a linear combination of the current residual $\mathbf{r}_{i+1}$ and all previous $\mathbf{p}_j$'s, $j = 1..i$.
An advantage of the GCR method is that, unlike GMRES and FGMRES, when using

GCR, the solution and residual vector can be directly accessed at any iterate, with zero computational cost.

1.2.3 Multigrid method

The multigrid approach is initially specifically designed for the solution of discretized elliptic PDEs and exploiting more information on the problem. The idea behind the multigrid solver is to approximate the original PDE problem of interest on a hierarchy of grids and use solutions from coarse grids to accelerate the convergence on the finest grid. This principle is based on a combination of two processes. First, we apply a smoother such as the classical iterative method like a Jacobi or a Gauss Seidel scheme to reduce the high frequency components of the error. Next, low frequency error components are reduced by a coarse grid correction procedure. The smooth error components are represented as a solution of an appropriate coarser system. After solving the coarser problem, the solution is interpolated back to the fine grid to correct the fine grid approximation for its low frequency errors.

Let P be an interpolation (prolongation) operator that transfers solutions from coarse grids to finer grids, and R be a restriction operator that transfers solutions from fine grids to the coarse grid, the following algorithm 2 describes briefly the different basic steps of the multigrid method.

Algorithm 2 2-grid cycle:

1: Subscript h is used for the fine grid and H for the coarse grid.
2: Perform smoothing by using k iterations of an iterative method (Jacobi, Gauss Seidel, etc) on the problem $\mathbf{A}_h\mathbf{u}_h = \mathbf{b}_h$
3: Compute the residual $\mathbf{r}_h = \mathbf{b}_h - \mathbf{A}_h\mathbf{u}_h$
4: Restrict the residual $\mathbf{r}_H = R\mathbf{r}_h$
5: Solve for the coarse grid correction, $\mathbf{A}_H\mathbf{e}_H = \mathbf{r}_H$
6: Prolongate and update $\mathbf{u}_h = \mathbf{u}_h + P\mathbf{e}_H$
7: Perform smoothing by using iterations of an iterative method (Jacobi, Gauss Seidel, etc) on the problem $\mathbf{A}_h\mathbf{u}_h = \mathbf{b}_h$

The solve for the coarse grid correction can be optimised by employing recursive calls carried out in a loop, allowing different numbers of iterative sweeps on different coarse grids. Thus we obtain the different V, W and F cycles.

We distinguish two multigrid approaches: the geometric multigrid methods, where the restriction and prolongation operators, and coarse grids are chosen based on the geometric information [143] and the algebraic multigrid (AMG) methods that construct their hierarchy of operators directly from the system matrix in the construction of the next coarse grid space without any geometrical problem background or interpretation [23, 117, 116, 4]. In the following, we will describe the outlines of algebraic treatment.

In order to solve the PDE on a coarser grid we need to perform a smooth aggregation. The matrix graph is first coarsened (actually vertices are aggregated together) and then a grid transfer operator is constructed.

To generate the matrix graph for the coarsened grid, the idea is to associate a vertex with each equation and add an edge between two vertices i and j if there is a nonzero in the $(i,j)^{th}$ or $(j,i)^{th}$ entry. It is this matrix graph whose vertices are aggregated together that

effectively determines the next coarser mesh. The aggregation process can be achieved in two ways. First, a block matrix graph can be constructed instead of a point matrix graph. In particular, all the DOFs at a grid point can be collapsed into a single vertex of the matrix graph. The resulting block matrix graph is significantly smaller than the point matrix graph, and hence all unknowns at a grid point are kept together. This usually results in better convergence rates, and the coarsening is actually less expensive to compute. The second way is to omit the small values, *i.e.* ignoring weak coupling during coarsening. In general, a drop tolerance, ϵ, can be set such that an individual matrix entry $A(i,j)$ is dropped in the coarsening phase if $|A(i,j)| \leq \epsilon\sqrt{|A(i,i)A(j,j)|}$. This drop tolerance ϵ (whose default value is zero) is also called threshold.

A significant inconvenient of the multigrid methods is that they may require implementations that are specific to the physical problem at hand, in contrast with preconditioned Krylov subspace methods which attempt to be general-purpose, using no other information than the problem matrix and the corresponding right-hand side vector.

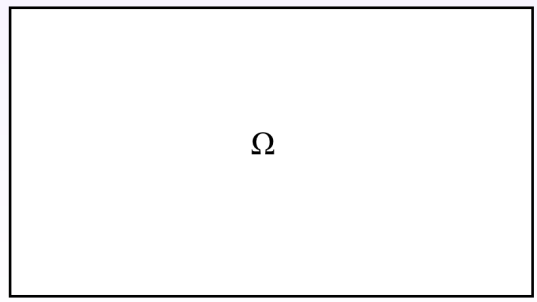
The algebraic multigrid techniques, although applicable to a wider range of linear systems, are often less robust, more complex to implement, and less efficient than the geometric counterpart. The geometric multigrid methods, on the other hand, are well suited for applications with simple geometry that are large enough in size to justify the added complexity.

1.2.4 Domain decomposition method

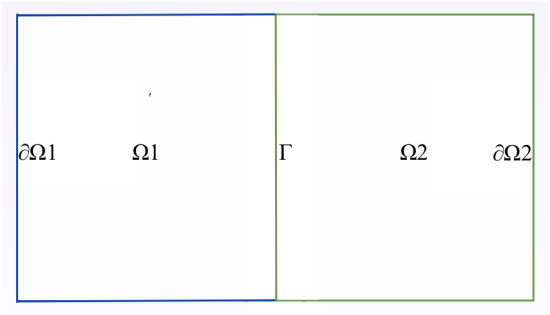
In this section, the methods of concern are based on a physical decomposition of a global solution domain. The global solution to a PDE is then sought by solving the smaller subdomain problems collaboratively and "patching together" the subdomain solutions. These numerical methods are therefore termed as domain decomposition (DD) methods. The DD methods have established themselves as very efficient PDE solution methods (FEM, FV, FD, SEM). Although sequential DD methods already have superior efficiency compared with many other numerical methods, their most distinguished advantage is the straightforward applicability for parallel computing. Other advantages include easy handling of global solution domains of irregular shape and the possibility of using different numerical techniques in different subdomains, *e.g.*, special treatment of singularities.

The DD methods allow the reformulation of a boundary-value problem (BVP) on a partition of the computational domain into subdomains. Thus, it is a very convenient framework for the solution of heterogeneous or multiphysics problems, *i.e.* those that are governed by differential equations of different kinds in different subregions of the computational domain.

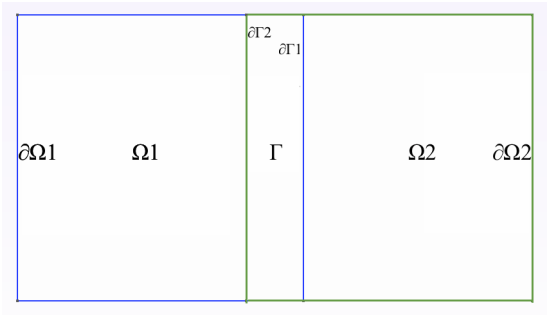
There are two ways of subdividing the computational domain: i) with disjoint subdomains, ii) with overlapping subdomains. Correspondingly, different DD algorithms will be set up such as Jacobi Schwarz Method (JSM), Additive Schwarz Method (ASM), Multiplicative Schwarz Methods (MSM), and Restricted Additive Schwarz (RAS) which is the default parallel solver in PETSc, Neumann-Neumann/FETI and Optimized Schwarz. In the case of overlapping subdomains, we will use the following notations. $\partial\Omega_1$ and $\partial\Omega_2$ will denote the boundary of Ω_1 and Ω_2, respectivelyy. Γ will denote the overlapping region of Ω_1 and Ω_2, $\Gamma = \Omega_1 \cap \Omega_2$. The boundary of Γ will be denoted by $\partial\Gamma = \partial\Gamma_1 \cup \partial\Gamma_2 \cup (\partial\Omega_1 \cap \partial\Omega_2)$, where $\partial\Gamma_1$ and $\partial\Gamma_1$ are the parts of Γ's boundary included



(a) Original domain



(b) Domain decomposition without overlap.



(c) Domain decomposition with overlap.

Figure 3.1: Decomposition of a 2D domain.

strictly in Ω_1 and Ω_2, respectively.

Consider the model problem: find $\mathbf{u}$ such that:

$$\begin{aligned} \mathbf{Lu} &= \mathbf{f} \quad \text{in } \Omega \\ \mathbf{u} &= \mathbf{0} \quad \text{on } \partial\Omega \end{aligned}$$

where $\mathbf{L}$ is a generic second order elliptic operator. The weak formulation reads:

$$\text{find } \mathbf{u} \in \mathbb{V} = H_0^1(\Omega) \quad \text{such that} \quad a(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \quad \forall \mathbf{v} \in \mathbb{V}$$

where $a(.,.)$ is the bilinear form associated to $\mathbf{L}$.

Case with overlapping sub-domains: Let us now consider the case where the domain Ω is decomposed into two sub-domains Ω_1 and Ω_2 that overlap. The iterative Schwarz method reads: given u_2^0 on $\partial\Gamma_1$, for $k \geq 1$,
solve:

$$\begin{aligned} \mathbf{Lu}_1^k &= \mathbf{f} \quad \text{in } \Omega_1 \\ \mathbf{u}_1^k &= \mathbf{u}_2^{k-1} \quad \text{on } \partial\Gamma_1 \\ \mathbf{u}_1^k &= \mathbf{0} \quad \text{on } \partial\Omega_1 \backslash \partial\Gamma_1 \end{aligned}$$

solve:

$$\begin{aligned} \mathbf{L}\mathbf{u}_2^k &= \mathbf{f} && \text{in } \Omega_2 \\ \mathbf{u}_2^k &= \begin{cases} \mathbf{u}_1^k \\ \mathbf{u}_1^{k-1} \end{cases} && \text{on } \partial\Gamma_2 \\ \mathbf{u}_2^k &= \mathbf{0} && \text{on } \partial\Omega_2 \backslash \partial\Gamma_2 \end{aligned} \tag{3.3}$$

We have two elliptic boundary-values problems with Dirichlet conditions in Ω_1 and Ω_2, and we wish the two sequences $\mathbf{u}_1^{(k)}$ and $\mathbf{u}_2^{(k)}$ to converge to the restrictions of the solution $\mathbf{u}$ of the original problem: $\lim_{k\to\infty} \mathbf{u}_1^{(k)} = \mathbf{u}_{|\Omega_1}$ and $\lim_{k\to\infty} \mathbf{u}_2^{(k)} = \mathbf{u}_{|\Omega_2}$ The Schwarz method applied to the model problem always converges, with a rate that increases as the measure $|\Gamma|$ of the overlapping region Γ increases.

In system (3.3), $\mathbf{u}_2^k = \mathbf{u}_1^k$ stands for the multiplicative Schwarz method while $\mathbf{u}_2^k = \mathbf{u}_1^{k-1}$ stands for the additive Schwarz method. It is worth pointing out that, although the additive Schwarz method suits well for parallel computing, its convergence property is inferior to that of the multiplicative Schwarz method. The additive Schwarz method uses roughly twice as many iterations as that of the standard multiplicative Schwarz method, in case of convergence. This is not surprising when the Schwarz methods are compared with their linear system solver analogues; multiplicative Schwarz is a block Gauss-Seidel approach, whereas additive Schwarz is a block Jacobi approach.

Case with non-overlapping sub-domains: We partition now the domain Ω in two disjoint subdomains. The following equivalence result holds.

Theorem 3. *The solution* $\mathbf{u}$ *of the model problem is such that* $\mathbf{u}_{|\Omega_i} = \mathbf{u}_i$ *for* $i = 1, 2$*, where* $\mathbf{u}_i$ *is the solution to the problem:*

$$\begin{aligned} \mathbf{L}\mathbf{u}_i &= \mathbf{f} \quad \textit{in } \Omega_i \\ \mathbf{u}_i &= \mathbf{0} \quad \textit{on } \partial\Omega_i \backslash \Gamma \\ \mathbf{u}_1 &= \mathbf{u}_2 \quad \textit{on } \partial\Gamma \\ \frac{\partial \mathbf{u}_1}{\partial n} &= \frac{\partial \mathbf{u}_2}{\partial n} \quad \textit{on } \Gamma \end{aligned}$$

For more information about the boundary conditions, the reader can refer to [113]. To solve each of the subdomain, a direct or an iterative solver can be called.

1.3 Preconditioning

For solving very large sparse linear systems, direct methods or iterative methods can be used. Although direct methods are robust and accurate for general nonsingular problems, they scale poorly when the problem size increases, in terms of time and memory complexity, especially for problems resulting from the discretization of PDEs in 3D space. On the other hand, iterative methods require less storage and fewer operations than direct methods. However, their performance depends strongly on the spectral properties of the linear system. The preconditioning techniques can improve the efficiency and the robustness of these methods. The term preconditioning refers to a transformation of the original linear system $\mathcal{A}\mathbf{x} = \mathbf{b}$ into another linear system with the same solution, but with more favourable properties for iterative solver.

Definition 8. *A condition number $\kappa(\mathcal{A})$ associated to the linear equation $\mathcal{A}\mathbf{x} = \mathbf{b}$ is a constant that gives a bound on how inaccurate the solution $\mathbf{x}$ will be after approximation. The condition number relative to a subordinate matrix norm $\|\|$ is defined by*

$$\kappa(\mathcal{A}) = \|\mathcal{A}\|\|\mathcal{A}^{-1}\|.$$

Note that we always have $\kappa(\mathcal{A}) \geq 1$, since $1 = \|\mathcal{I}\| = \|\mathcal{A}\mathcal{A}^{-1}\| \leq \|\mathcal{A}\|\|\mathcal{A}^{-1}\|$.
In practice, the most frequently used conditionings are $\kappa_p(\mathcal{A}) = \|\mathcal{A}\|_p\|\mathcal{A}^{-1}\|_p$ for $p = 1, 2, +\infty$.

Proposition 1. *Consider a matrix $\mathcal{A} \in \mathcal{GL}_n(\mathbb{C})$, we have:*

- *$\kappa(\mathcal{A}) = \kappa(\mathcal{A}^{-1})$ and $\kappa(\alpha\mathcal{A}) = \kappa(\mathcal{A})\forall\alpha \neq 0$.*
- *For any matrix $\mathcal{A}$,*

  $$\kappa_2(\mathcal{A}) = \frac{\sigma_{\max}(\mathcal{A})}{\sigma_{\min}(\mathcal{A})}$$

  where $\sigma_{\min}(\mathcal{A})$ and $\sigma_{\max}(\mathcal{A})$ are respectively the smallest and the largest singular values of $\mathcal{A}$.
- *For a normal matrix $\mathcal{A}$,*

  $$\kappa_2(\mathcal{A}) = \frac{|\lambda_{\max}(\mathcal{A})|}{|\lambda_{\min}(\mathcal{A})|}$$

  where $\lambda_{\min}(\mathcal{A})$ and $\lambda_{\max}(\mathcal{A})$ are respectively the modulus of the smallest and largest eigenvalues of A.
- *For any unitary matrix $\mathbf{U}$, $\kappa_2(\mathbf{U}) = 1$.*
- *For any unitary matrix $\mathbf{U}$, $\kappa_2(\mathcal{A}\mathbf{U}) = \kappa_2(\mathbf{U}\mathcal{A}) = \kappa_2(\mathcal{A})$.*

Definition 9. *A problem with a low condition number is said to be well-conditioned, while a problem with a high condition number is said to be ill-conditioned.*

Definition 10. *A preconditioner P of a matrix $\mathcal{A}$ is a matrix that aims at improving the spectral properties of the matrix $\mathcal{A}$ associated with the linear system such that $P^{-1}\mathcal{A}$ has a smaller condition number than $\mathcal{A}$.*

The idea of the preconditioning method is that, instead of solving the original linear system $\mathcal{A}\mathbf{x} = \mathbf{b}$, one may solve, either the right preconditioned system:

$$\mathcal{A}P^{-1}P\mathbf{x} = \mathbf{b}$$

via solving $AP^{-1}\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$ and $P\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$; or the left preconditioned system:

$$P^{-1}(\mathcal{A}\mathbf{x} - \mathbf{b}) = \mathbf{0}$$

or a symmetric preconditionning:

$$P_L^{-1}\mathcal{A}P_R^{-1} = P_L^{-1}\mathbf{b}$$

with $\mathbf{x} = P_R^{-1}\mathbf{y}$.

When Krylov subspace methods are used, it is not necessary to form the preconditioned matrices $P^{-1}\mathcal{A}$ or $\mathcal{A}P^{-1}$ explicitly (this would be too expensive, and we would loose sparsity). Instead, matrix-vector products with $\mathcal{A}$ and solutions of linear systems of the form $P\mathbf{z} = \mathbf{r}$ are performed (or matrix-vector products with P^{-1} if this is explicitly known). Besides, split preconditioning is also possible, *i.e.*, $P_1^{-1}\mathcal{A}P_2^{-1}\mathbf{y} = P_1^{-1}\mathbf{b}$, $\mathbf{x} = P_2^{-1}\mathbf{y}$, where the preconditioner is now $P = P_1P_2$. The choice of the preconditioning type depends on the choice of the iterative method, problem characteristics, and so forth. For instance, with residual minimizing methods, like GMRES, right preconditioning is often used. In exact arithmetic, the residuals for the right-preconditioned system are identical to the true residuals $\mathbf{r}_k = \mathbf{b} - \mathcal{A}\mathbf{x}_k$.

To illustrate the action of a preconditioner, let us take, for sake of simplicity, the example of the conjugate gradient algorithm. The following algorithm is the corresponding preconditioned algorithm.

Algorithm 3 Preconditioned conjugate gradient method

Input $k = 0$
$\mathbf{x}_0 = \mathbf{0}$
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{z}_0 = \mathbf{P}^{-1}\mathbf{r_0}$
$\mathbf{p}_0 = \mathbf{z}_0$
while $\mathbf{r}_k \neq \mathbf{0}$ **do**
  $\alpha_k = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$
  $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
  if $\mathbf{r}_{k+1}$ *is sufficiently small* **then**
    exit
  else
    $\mathbf{z}_{k+1} = P^{-1}\mathbf{r}_{k+1}$
    $\beta_k = \frac{\mathbf{z}_{k+1}^T\mathbf{r}_{k+1}}{\mathbf{z}_k^T\mathbf{r}_k}$
    $\mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
    $k = k + 1$
  end
end

In general, a good preconditioner P should verify the following requirements. First, the preconditioned system must be easy to solve, and second, the preconditioner should be easy to construct and apply. The first property means that the preconditioned iteration should converge rapidly, while the second ensures that each iteration is not too expensive. It is necessary to strike a balance between the two needs. What constitutes an acceptable cost of constructing the preconditioner, or setup time, will typically depend on whether the preconditioner can be reused or not. In the common situation where a sequence of linear systems with the same coefficient matrix (or a slowly varying one) and different right-hand sides has to be solved, it may pay to spend some time computing a powerful preconditioner, since its cost can be amortized over repeated solves. This is can be the case, for instance, when solving nonlinear problems by some variant of Newton's method.

We must distinguish at this level, two approaches for constructing a preconditioner. The first one is the application-specific approach which can be very successful, but it may

require complete knowledge of the problem at hand, including the original (continuous) equations, the domain of integration, the boundary conditions, details of the discretization, and so forth. The geometric multigrid methods, previously mentioned, are often of this kind. The second approach is the purely algebraic methods that use only information contained in the coefficient matrix **A**. These techniques, while not optimal for any particular problem, achieve reasonable efficiency on a wide range of problems.

In order to solve problem (3.1), we developed in the sequel, a preconditioning strategy based on the first approach, where the knowledge we have about the fluid flow is incorporated in the construction of the preconditioner.

Remark 13. *The convergence of preconditioned Krylov subspace methods for solving linear systems arising from discretized PDE tends to slow down considerably as these systems become larger and deteriorate the efficiency. Multigrid methods are however a class of methods capable of achieving convergence rates independently of the mesh size. The main difference with the preconditioned Krylov subspace approach is being initially specifically designed for the solution of discretized elliptic PDEs and exploiting more information on the problem.*

Feel++ relies on the PETSc library that provides a wide range of solvers and preconditioners. Let us do a quick enumeration on some solvers and preconditoners that we will use un the sequel. We will first start with *i*) `LU` preconditioner provided by the library `MUMPS` accessible from PETSc. With this preconditioner, a Krylov method is no longer necessary since it can be seen as a direct solver. This preconditioner is the most competitive for 2D problems, however, for 3D problems or when the size of the problem becomes considerably important, it scales less due to the computational cost and memory consumption. *ii*) `Additive Schwarz` methods which belong to the class of domain decomposition methods such as Block Jacobi (`b-Jacobi`) which is an additive Schwarz algorithm without overlapping, `ASM` preconditioner, which is also an additive Schwarz method with algebraic overlapping. The classical additive Schwarz preconditioner P applied on p subdomains is written as follows:

$$P = \sum_{i=1}^{p} R_i^T \mathcal{A}_i^{-1} R_i \tag{3.4}$$

with R_i the restriction operator and $\mathcal{A}_i = R_i \mathcal{A} R_i^T$ the restriction of $\mathcal{A}$ on the i^{th} subdomain. Again, the $\mathcal{A}_i^{-1}$ are not explicitly calculated, a Krylov method is however used with a preconditioner. Hence, for each subdomain problem, different sequential solvers and preconditioners can be used, such as `LU`, `ILU` (Incomplete `LU`), `Cholesky`, `ICC` (Incomplete Cholesky), `Jacobi`, `ML` (algebric multigrid). PETSc also provides the `GASM` preconditioner which is a variant of `ASM` method extended for a high number of subdomains. The latter will be the most used domain decomposition method in this thesis. *iii*) `Fieldsplit` is also a preconditioner class for block-matrices provided by PETSc. It includes `b-Jacobi`, `Gauss-Seidel` and symmetric block Gauss-Seidel. The blocks can be bounded by the lines and columns indices. In the following, we will make use of this family of preconditioners for block-preconditioners. Finally, PETSc provides the access to `ML`, `hyper` and `GAMG` multigrid preconditioners. The latter is designed to be modular, extensible, and to accept small contributions from users as easily as possible, which made

it our Multgrid preconditioner choice in the sequel.

While separately Krylov subspace and Multigrid methods are widely used to develop fast solvers, a new trend consists in designing hybrid methods gathering both these strategies. It aims to join, in a unified framework, the main features of these methods, that are the easy computation, the parameter free, the high possibility of vectorization and parallelization for the Krylov subspace methods, the grid size independent convergence rates for the multigrid method. The efficiency of the multigrid method is due to the fact that the spectral radius of the iteration matrix is uniformly bounded by a constant smaller than one. However, for difficult model problems, the poor convergence rate is due to the slow convergence of some eigenmodes of the iteration matrix. One idea is to use Krylov subspace methods to capture these eigenmodes and therefore to improve the convergence properties. The multigrid is then seen as a preconditioner in a Krylov subspace method. It is found that multigrid preconditioning involves a favourable spectrum and speeds up the convergence rate also in the case of good convergence properties. We must mention that the iteration matrix of a general multigrid method is nonsymmetric. Moreover, the smoothing step is also nonsymmetric. Therefore, the preconditioning matrix is nonsymmetric, which requires a Krylov subspace method adapted to non-symmetric problems. Domain decomposition schemes used as preconditioners are also an efficient strategy, specially for high number of domains, they require less tuning when they are employed as a preconditioner to accelerate the convergence of a Krylov method. Since Schwarz methods represent fixed point iterations applied to preconditioned global problems, and consequently do not provide the fastest convergence, it is natural to apply Krylov methods instead. This provides the justification of using Schwarz methods as preconditioners rather than solvers.

Conclusion

Through this chapter, we have briefly explained the main resolutions and preconditioning strategies towards a high precision computing (HPC) simulations of problem like $\mathcal{A}\mathbf{x} = \mathbf{b}$. *We also gave the pros and cons of each strategy depending on our observations and experiments. Besides, we have introduced the preconditioners that we will use in Chapter 6 for our benchmarks.*

Chapter 4

An overview of block preconditioners

In this chapter, we give a survey of the most efficient block-preconditioner for the incompressible Navier-Stokes equations. We enter into the details of three block-preconditioners, the physics-based SIMPLE preconditioner, first introduced in [106], the pressure convection-diffusion preconditioner (PCD) proposed in [90, 129] and the least squares commutator (LSC) [47]. The latters are obtained using a purely algebraic manipulation of the matrix blocks while SIMPLE can be written as an approximate factorization of the system's matrix. The efficiency of the three aforementioned preconditioners is evaluated in Section 1 of Chapter 6. For a more global overview on preconditioning technics, the reader may refer to [16].

Contents

1 Introduction

We start by observing that the matrix of discrete system that we denote by $\mathcal{A}$, can be written as a LDU product:

$$\mathcal{A} = \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B}\mathbf{F_u}^{-1} & \mathbf{I} \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \mathbf{F_u} & \mathbf{0} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{F_u}^{-1}B^T \\ \mathbf{0} & -\mathbf{S} \end{pmatrix}}_{U} \tag{4.1}$$

where $\mathbf{S} = \mathbf{B}\mathbf{F_u}^{-1}\mathbf{B^T}$ is the Schur complement matrix. Most block-type preconditioners are based on a combination of those three matrices and a suitable approximation of the Schur complement matrix. We distinguish two main categories of block preconditioners, left preconditioners, based on the LD product, such as the SIMPLE preconditioner family, and right preconditioners, based on the DU product, such as the PCD preconditioner, the LSC preconditioner and the augmented Lagrange preconditioner. The latter is not discussed in this thesis.

Remark 14. *The choice of the global iterative method is crucial for applying each of the above preconditioners. A Krylov method should be suitable with the type of the preconditioner, either right or left. For example the GCR and fGMRESR methods are appropriate for right preconditioner while GMRES is appropriate for left preconditioners.*

2 LSC preconditioner

The least squares preconditioner (LSC) [47] makes use of the following approximation of $\mathbf{S}$

$$\hat{\mathbf{S}}_{\text{LSC}} = (\mathbf{B}\mathbf{Q_u^{-1}}\mathbf{B^T})(\mathbf{B}\mathbf{Q_u^{-1}}\mathbf{F_p}\mathbf{Q_u^{-1}}\mathbf{B^T})^{-1}(\mathbf{B}\mathbf{Q_u^{-1}}\mathbf{B^T}) \tag{4.2}$$

where $\mathbf{F_p}$ is the matrix that corresponds to the convection diffusion operator defined on the pressure space.
In practice, the mass matrix $\mathbf{Q_u}$ may be approximated by its diagonal $diag(\mathbf{Q_u})$ to avoid the dense inverse. The convergence rate of the method is improved since this matrix acts like a scaling matrix. Let us denote $\mathbf{S_p r}_1 = \mathbf{B}\mathbf{Q_u^{-1}}\mathbf{B^T}$, solving $P_{LSC}\mathbf{z} = \mathbf{r}$ with $\mathbf{z} = \begin{pmatrix} \mathbf{z_u} \\ \mathbf{z}_p \end{pmatrix}$ and $\mathbf{r} = \begin{pmatrix} \mathbf{r_u} \\ \mathbf{r}_p \end{pmatrix}$ then reads:

Algorithm 4 LSC algorithm

1: Solve $\mathbf{S_p z}_p = \mathbf{r}_p$
2: Update $\mathbf{r}_p = \mathbf{B}\mathbf{Q_u^{-1}}\mathbf{F_p}\mathbf{Q_u^{-1}}\mathbf{B^T}\mathbf{z}_p$
3: Solve $\mathbf{S_p z}_p = -\mathbf{r}_p$
4: Update $\mathbf{r_u} = \mathbf{r_u} - \mathbf{B^T z}_p$
5: Solve $\mathbf{F_u z_u} = \mathbf{r_u}$

Applying this algorithm involves two Poisson-type solves for the pressure subsystem and one velocity solve. These subproblems need to be supplemented with appropriate boundary conditions in order to be homogeneous with the initial boundary conditions

applied on the global problem. A new choice of boundary conditions was proposed in [49], but in the following, we will be testing the old version of this preconditioner introduced in [129].

3 SIMPLE preconditioner

Another LU factorisation of A,

$$\mathcal{A} = \underbrace{\begin{pmatrix} \mathbf{F_u} & \mathbf{0} \\ \mathbf{B} & -\mathbf{S} \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{F_u}^{-1}\mathbf{B^T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}}_{U} \tag{4.3}$$

stands at the base of the so-called SIMPLE preconditioner [106], obtained by replacing $\mathbf{F_u}^{-1}$ in both the factors L and U by a diagonal matrix $\mathbf{D^{-1}}$ (for instance, $\mathbf{D}$ could be the diagonal of $\mathbf{F_u}$). More precisely,

$$P_{SIMPLE} = \begin{pmatrix} \mathbf{F_u} & \mathbf{0} \\ \mathbf{B} & -\hat{\mathbf{S}} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{D^{-1}B^T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = \hat{L}\hat{U}, \tag{4.4}$$

with $\hat{S} = BD^{-1}B^T$. Note that when the mesh size h decreases and/or the Reynolds number increases, the convergence of the preconditioned iterative methods deteriorates. Many generalisations of the $SIMPLE$ preconditioner have been proposed such as $SIMPLZE$, $h - SIMPLE$ and $MSIMPLER$.
The P_{SIMPLE} uses an $\hat{L}\hat{U}$ factorisation that can be regarded as a special case of a more general family of inexact or algebraic factorisations that read as follows:

$$\hat{\mathcal{A}} = \hat{L}\hat{U} = \begin{pmatrix} \mathbf{F_u} & \mathbf{0} \\ \mathbf{B} & -\mathbf{B}\mathcal{L}\mathbf{B^T} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathcal{U}\mathbf{B^T} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{4.5}$$

where $\mathcal{L}$ and $\mathcal{U}$ are two approximations of $\mathbf{F_u}^{-1}$. If the time dependent Navier-Stokes equations are discretized in time by high-order (≥ 2) temporal scheme, inexact factors are chosen so that the time discretization order is maintained. When used in connection with time-dependent (either Stokes or Navier-Stokes) problem *e.g.* (2.54), two different possibilities stand out:

- $\mathcal{L} = \mathcal{U} = (\frac{1}{\Delta t}\mathbf{Q_u})^{-1}$: *Chorin-Temam algebraic approximation*,
- $\mathcal{L} = (\frac{1}{\Delta t}\mathbf{Q_u})^{-1}$ and $\mathcal{U} = \mathbf{F_u}^{-1}$: *Yosida approximation* as it can be interpreted as a Yosida regularisation of the Schur complement.

The fact that the Yosida approximation does not require any special care about boundary conditions which are implicitly accounted for the algebraic formulation, is an advantage of this strategy.

4 The Pressure Convection Diffusion (PCD)

Let us recall the weak formulation corresponding to the fully implicit treatment with homogeneous Dirichlet boundary conditions on Γ_D and a stress free condition on Γ_N,

with $\partial\Omega = \Gamma_D \cup \Gamma_N$. It reads to: find $(\mathbf{u}_\delta^{k+1}, p_\delta^{k+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H_0^1(\Omega_\delta)]^d \cap [P_c^M(\Omega_\delta)]^d\}$, $\forall q_\delta \in Q_\delta$

$$\rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^{k+1}}{\Delta t}\mathbf{v}_\delta + \rho \int_{\Omega_\delta} (\mathbf{u}_\delta^{k+1} \cdot \nabla \mathbf{u}_\delta^{k+1}) \cdot \mathbf{v}_\delta + 2\mu \int_{\Omega_\delta} \mathbf{D}(\mathbf{u}_\delta^{\mathbf{k+1}}) : \mathbf{D}(\mathbf{v}_\delta)\ d\mathbf{x}$$
$$- \int_{\Omega_\delta} p_\delta^{k+1}\ \mathrm{div}(\mathbf{v}_\delta)\ d\mathbf{x} = \int_{\Omega_\delta} \mathbf{f}_\delta^{k+1} \cdot \mathbf{v}_\delta + \rho \int_{\Omega_\delta} \frac{\mathbf{u}_\delta^k}{\Delta t} \cdot \mathbf{v}_\delta$$
$$\int_{\Omega_\delta} q_\delta\ \mathrm{div}(\mathbf{u}_\delta^{k+1})\ d\mathbf{x} = 0$$

In its algebraic form, the problem using Picard linearisation method is written as follows:

$$\underbrace{\begin{pmatrix} \mathbf{F_u} & \mathbf{B^T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}}_{\mathcal{A}} \begin{pmatrix} \mathbf{U^{n+1}} \\ \mathbf{P^{n+1}} \end{pmatrix} = \begin{pmatrix} \mathbf{G} \\ \mathbf{0} \end{pmatrix} \tag{4.6}$$

where $\mathbf{F_u}$ is a function of the unknown velocity,

$$(\mathbf{F_u})_{i,j} = \rho \frac{1}{\Delta t} \underbrace{< \varphi_i, \psi_j >}_{Qu_{i,j}} + \underbrace{a(\varphi_i, \psi_j)}_{A_{i,j}} + \underbrace{c(\mathbf{u}^{k+1}, \varphi_i, \psi_j)}_{C_{i,j}} \tag{4.7}$$

where $\mathbf{Q_u}$ is the fluid mass matrix, $\mathbf{D}$ the stiffness matrix, and $\mathbf{C}$ the convection terms of the momentum equation. $\mathbf{B}$ and $\mathbf{B^T}$ are the discretized counterparts of the divergence operator and the gradient operator, respectively. $\mathbf{U^{n+1}}$ is the vector of the velocity unknowns at time $t = t^{n+1}$ and $\mathbf{P^{n+1}}$the one of the pressure unknowns. $\mathbf{G}$ is a known vector depending on the discretized source force, on $\mathbf{U^n}$, and on $\mathbf{f}$, $\mathbf{g_N}$, and $\mathbf{g_D}$, the boundary condition expressions on the Dirichlet and the Neumann boundaries, respectively.

The PCD preconditioned, proposed in [90, 129], is based on the following factorization of the matrix $\mathcal{A}$:

$$\mathcal{A} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B F_u^{-1}} & \mathbf{I} \end{pmatrix} \underbrace{\begin{pmatrix} \mathbf{F_u} & \mathbf{B^T} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix}}_{P_{PCD}} \tag{4.8}$$

where $\mathbf{S} = \mathbf{B F_u^{-1} B^T}$ is the so-called Schur complement. However getting the inverse of the Schur complement written this way is a costly operation. We will consequently replace the Schur complement S by an appropriate approximation.
For that purpose, let us consider the convection-diffusion operator:

$$\mathcal{F} = -\nu\nabla^2 + \omega_h \cdot \nabla$$

where the $\mathbb{R}^d$ vector field ω_h denotes the approximation of the discrete velocity at the previous nonlinear iteration.
Let us also consider the analogous convection-diffusion operator $\mathcal{F}_p$ defined on the pressure space and the following commutator of the convection-diffusion operators with the divergence operator,

$$\mathcal{E} = \nabla \cdot (-\nu\nabla^2 + \omega_h \cdot \nabla) - (-\nu\nabla^2 + \omega_h \cdot \nabla)_p \nabla, \tag{4.9}$$

The matrix representation of the discrete divergence operator is $\mathbf{Q_p^{-1}B}$, where $\mathbf{Q_p}$ is the mass matrix associated to the pressure space, and the one associated to the discrete gradient operator is $\mathbf{Q_p^{-1}B^T}$. Thus the discrete commutator leads to:

$$\mathcal{E}_h = (\mathbf{Q_p^{-1}B})(\mathbf{Q_u^{-1}F_u}) - (\mathbf{Q_p^{-1}F_p})(\mathbf{Q_p^{-1}B}). \tag{4.10}$$

Under the assumption that the commutator is small, we have:

$$(\mathbf{Q_p^{-1}B})(\mathbf{Q_u^{-1}F_u}) \approx (\mathbf{Q_p^{-1}F_p})(\mathbf{Q_p^{-1}B}). \tag{4.11}$$

Let us premultiply by $\mathbf{Q_pF_p^{-1}Q_p}$ and post-multiply by $\mathbf{F_u^{-1}B^T}$, we obtain:

$$\begin{aligned} \mathbf{Q_pF_p^{-1}Q_p(Q_p^{-1}B)(Q_u^{-1}F_u)F_u^{-1}B^T} &\approx \mathbf{Q_pF_p^{-1}Q_p(Q_p^{-1}F_p)(Q_p^{-1}B)F_u^{-1}B^T} \\ \mathbf{Q_pF_p^{-1}}\underbrace{\mathbf{BQ_u^{-1}B^T}}_{\mathbf{A_p}} &\approx \underbrace{\mathbf{BF_u^{-1}B^T}}_{\mathbf{S}} \end{aligned}$$

In the following, we will consequently replace the Schur complement $\mathbf{S}$ by $\mathbf{S^* = Q_pF_p^{-1}A_p}$, $\mathbf{F_p}$ is the convection-diffusion operator defined on the discrete pressure space of the operator F (*i.e.*, the discretized version of the operator $D_t - \nu\Delta + \mathbf{u}\cdot\nabla$ for the pressure space with D_t being the contribution of the time discretization), and $\mathbf{A_p = BQ_u^{-1}B^T}$ is a discrete weighted Laplacian operator for the pressure space.
In order to have a sparse discrete Laplacian, a more direct way is to replace $\mathbf{Q_u}$ by its diagonal approximation $\mathbf{T} = diag(\mathbf{Q_u})$.
A clear advantage of defining $\mathbf{A_p}$ as a discrete Laplacian assembled matrix is that it is easily adapted to handle stabilised elements, and thus is a more general option. As for the choice of $\mathbf{BT^{-1}B^T}$ it is more appropriate for evolutionary problems

Having all the ingredients of the preconditioner well defined, then solving solving $P_{PCD}\mathbf{z} = \mathbf{r}$ with $\mathbf{z} = \begin{pmatrix}\mathbf{z_u}\\ \mathbf{z}_p\end{pmatrix}$ and $\mathbf{r} = \begin{pmatrix}\mathbf{r_u}\\ \mathbf{r}_p\end{pmatrix}$ requires the following steps:

1. Solve $\mathbf{S^*}\mathbf{z}_p = \mathbf{r}_p$
2. Update $\mathbf{r_u} = \mathbf{r_u} - \mathbf{B^T}\mathbf{z}_p$
3. Solve $\mathbf{F_u}\mathbf{z_u} = \mathbf{r_u}$

Let us note that the preconditioning strategy requires the action of $\mathbf{S}*^{-1} = \mathbf{A_p^{-1}F_pQ_p^{-1}}$ thus, in order to approximate the inverse of the Schur complement, we will need to do a Poisson solve, a mass matrix solve and a matrix-vector product. These subproblems need to be completed with appropriate boundary conditions in order to be homogeneous with the initial boundary conditions applied on the global problem. The details are in section 4.1. Note that to do the inverse $\mathbf{A_p^{-1}}$ and $\mathbf{Q_p^{-1}}$, any iterative or direct solver may be choose. The details about our numerical choices for solvers and sub-solvers for the different sub-problems are reported and discussed in Chapter 6.

Remark 15. *If the Schur complement was applied exactly, the resulting GMRES method would converge in at most two iterations, while the convergence rate when using* $\mathbf{M_pF_p^{-1}A_p}$ *depends on the boundary conditions used to form* $\mathbf{F_p}$.

Remark 16. *In the previous version of the preconditioner, the commutator was defined using the gradient operator rather than the divergence operator, giving* $(-\nu\nabla^2+\omega_h\cdot\nabla)\nabla-\nabla(\nu\nabla^2+\omega_h\cdot\nabla)_p$. *The latest approach was designed so that boundary conditions are incorporated more effectively in the construction of the preconditioner. Actually, the divergence operator operates on the velocity space, where boundary conditions are defined, whereas the gradient operator operates on the pressure space, where no boundary conditions are specified [48].*

Remark 17. *In the case of a Stokes problem, a more appropriate preconditioned is the pressure mass matrix (PMM) preconditioner. In this case the Schur complement is approximated by the pressure mass matrix scaled with the inverse of the viscosity* $\mathbf{S}^* = \frac{1}{\nu}\mathbf{Q_p}$. *It is obvious that with this choice of preconditioner, the solve of the pressure step is cheaper than having to do a Poisson solve, a mass matrix solve and two matrix-vector products.*

4.1 Boundary conditions for the subproblems

Boundary condition for the Laplacian operator A_p

- On Γ_D: Let us rewrite the weak form of the Laplacian operator:

$$[A_p\mathbf{p}]_i = \Sigma_{j=1}^{n_p}[A_p]_{ij}\mathbf{p}_j = \Sigma_{j=1}^{n_p}(\int_\Omega \nabla\varphi_i\cdot\nabla\varphi_j)\mathbf{p}_j = \int_\Omega \nabla p_h\cdot\nabla\varphi_i$$
$$= \int_\Omega(-\nabla^2 p_h)\cdot\varphi_i - \int_{\partial\Omega}(\nabla p_h\cdot\mathbf{n})\varphi_i$$

  In order to have a consistent approximation of the Laplacian, the boundary integral in the construction of the corresponding matrix operator must vanish. Since $\varphi_i \neq 0$ on $\partial\Omega$, p_h has to satisfy an homogeneous Neumann condition on $\partial\Omega_D$.

- On Γ_N: Let us consider a velocity test function $v_h = \phi_l$, and let G denote a discrete gradient operator. Then its weak form leads to:

$$\begin{aligned}[G\mathbf{p}]_l &= \int_\Omega \nabla p_h\cdot\phi_l \\ &= -\int_\Omega p_h\nabla\cdot\phi_l + \int_{\partial\Omega_N} p_h(\phi_l\cdot\mathbf{n}) \\ &= \Sigma_{m=1}^{n_p}(-\int_\Omega \varphi_m\nabla\cdot\phi_l)\mathbf{p}_m + \Sigma_{m=1}^{n_p}(\int_{\partial\Omega_N}\varphi_m(\phi_l\cdot\mathbf{n})\mathbf{p}_m \\ &= [B^T\mathbf{p}]_l + [R\mathbf{p}]_l\end{aligned}$$

  Thus, we we have the following system $\begin{pmatrix}\mathbf{A} & G\\ B & 0\end{pmatrix}\begin{pmatrix}\mathbf{u}\\ \mathbf{p}\end{pmatrix} = \begin{pmatrix}\mathbf{f}\\ \mathbf{g}\end{pmatrix}$, where $G = B^T + R$. In order to have an homogeneous system with the one corresponding to a Poisson problem, R must be replaced with the zero matrix, that is we will have to force the discrete pressure to satisfy an homogeneous Dirichlet condition on the outflow boundary $\partial\Omega_N$.

Boundary condition for the convection diffusion operator F_p Let us first consider the 1D version of the commutator $\mathcal{E}$ applied on the interval $\Omega_1 = (0, 1)$ with the following operators,

$$\mathcal{F} = \mathcal{F}_p := -\nu \frac{d^2}{dx^2} + \omega \frac{d}{dx}, \quad \mathcal{B} = \frac{d}{dx} \tag{4.12}$$

In order to be uniquely defined, the operator must be coupled with appropriate boundary conditions. It leads to:

$$\mathcal{B}\mathcal{F} = \mathcal{F}_p\mathcal{B} = -\nu \frac{d^3}{dx^3} + \omega \frac{d^2}{dx^2} \tag{4.13}$$

Let us denote $v = \mathcal{F}u$, u defined on Ω_1, we then have,

$$v = \mathcal{F}u = (-\nu \frac{d}{dx} + \omega)\frac{du}{dx} = (-\nu \frac{d}{dx} + \omega)p = -\nu p' + \omega p \qquad \text{with } p = \frac{du}{dx} \tag{4.14}$$

Rewriting (4.13) using the expression below, we obtain $\mathcal{B}v = \mathcal{B}\mathcal{F}u = \mathcal{F}_p\mathcal{B}u$. We can notice that, in order for v to be appropriate as an argument of $\mathcal{B}$, we require $v(0) = 0$. So recalling (4.14) that depends on p, a Dirichlet condition $v(0) = 0$ for $\mathcal{B}$ leads to a Robin boundary condition for $\mathcal{F}_p$, $-\nu p' + \omega p = 0$ at the inflow boundary. Note that on the wall, where $\mathbf{u} = \mathbf{0}$, a Robin boundary condition reverts to a Neumann condition. To extend this discussion on higher dimension, the reader may refer to [48].

To sum up, when we have Dirichlet boundary conditions for the main problem, it leads to a Neumann boundary conditions on the $\mathbf{A}_p$ problem, and when we have Neumann boundary conditions for the main problem, it leads to a Dirichlet boundary conditions on the $\mathbf{A}_p$ problem. While for the $\mathbf{F}_p$ problem, a Dirichlet boundary condition on the main problem leads to a Robin boundary condition.

Remark 18. *In the case of a Stokes problem, a more appropriate preconditioned is the pressure mass matrix (PMM) preconditioner. In this case the Schur complement is approximated by the pressure mass matrix scaled with the inverse of the viscosity* $\mathbf{S}^* = \frac{1}{\nu}\mathbf{Q}_p$. *It is obvious that with this choice of preconditioner, the solve of the pressure step is cheaper than having to do a Poisson solve, a mass matrix solve and two matrix-vector products.*

4.2 Sub-problems solve

To solve the different sub-problems arising from the use of the PCD preconditioner, we will apply the following strategy.
For the Laplacian problem, Multigrid is the most appropriate choice, for it provides a good efficiency and strong scalability results. For the mass-matrix-problem inverse, different strategies may be used. We can either choose to replace $\mathbf{Q}_p^{-1}$ by the inverse of the lumped diagonal of $\mathbf{Q}_p$, or use a Chebychev polynomial preconditioner that converges faster with a minimal cost. The advantage of the first method is that the approximation is fast to build, and applying it is a embarrassingly parallel task. Actually we will not exactly inverse the $\mathbf{Q}_p$ matrix, instead we can use the `preonly` option and apply B-Jacobi as a preconditioner to extract the diagonal. A default choice also could be to use the Multigrid which also gives a fast approximation of the inverse of this matrix. For the inverse of the convection-diffusion matrix $\mathcal{F}_u$, we have chosen to extract the diagonal components block of this matrix using B-Jacobi and apply a new solving strategy for each block. Depending on the problem size, an accurate but heavy choice could be to use of GASM with LU

in the subdomains. An alternative could be to use Multigrid. However, as we will see in Chapter 6, GAMG does not converges for high Reynolds number, which plays in the advantage of GASM with LU in the sub-domains.

Conclusion

In this chapter, we did a review over the most popular block-preconditiones for the Navier-Stokes equations system. For the evaluation of the performance of the LSC and PCD preconditioners we rely on the PETSc *version of the first preconditioner, and we implemented the latter in* Feel++ *library. The details about the implementation are explained in Chapter 8. The efficiency of the LSC preconditioner and in-house PCD preconditioner is shown in Section 1 of Chapter 6.*

Part II

Numerical experiments and applications

Chapter 5

Convergence analysis of the Stokes formulations and stress tensor computation

The results presented in the first section of this chapter are a joint work with V. Chabannes and C.Caldini, M. Ismail, G. Pena, M. Szopos and C. Prud'homme, in the context of the CEMRACS 2012, published in [26]. In the first part of this chapter, we focus on the simulation step, using for now the standard Newtonian incompressible Navier-Stokes equations, and we are interested in particular in i) handling various boundary conditions settings allowing for a flexible framework with respect to the type of input data (velocity, pressure, flow rate, ...); ii) handling of the discretization errors not only with respect to the physical fields (velocity and pressure) but also with respect to the geometry. We emphasize that the accuracy with which the boundary of the physical domain is approximated has a considerable impact on the quality of the numerical approximation, see for instance [108]. In particular, in the context of blood flow approximation, this takes particular relevance since the boundary might also be an unknown of the problem and needs to be accurately approximated.The aim of the second part is to present and compare two different methods for the computation of the stress tensor for low and high order finite element and geometry approximation on a straight and curved boundary.

Contents

1 Numerical study of the Stokes Navier-Stokes problem with different boundary conditions

1.1 Benchmark setup

Our goal is to impose different kinds of boundary conditions (for both velocity and pressure) to the Navier-Stokes system (2.1)-(2.2). Since these considerations are independent of the presence of a convective term or a time derivative, in the following we focus only on the Stokes equations.

The differential problem is then written as: find $(\boldsymbol{u}, p)$ such that

$$-2\mu \mathbf{div}(\mathbf{D}(\mathbf{u})) + \nabla p = \mathbf{0} \text{ in } \Omega, \tag{5.1}$$

$$\mathrm{div}(\mathbf{u}) = 0 \text{ in } \Omega, \tag{5.2}$$

Equations (5.1) (5.2) will be supplemented with boundary conditions like those detailed in Section 2.2, and the corresponding variational formulations are retrieved following the same steps as described in Section 2.1.

In order to verify that the various formulations converge with the theoretically expected rates (see error estimates (2.42)), we consider the following benchmark that consists of the simulation of a 3D Poiseuille flow in a cylinder with a base of radius $r = 1$ and length $L = 5$ centered at $(2.5, 0, 0)$, see Figure 5.1. The exact solution for this benchmark is

$$\boldsymbol{u}_{ex}(x, y, z) = \left(\frac{(p_{in} - p_{out})r^2}{4\mu L} \left(1 - \frac{y^2 + z^2}{r^2} \right), 0, 0 \right), \tag{5.3}$$

$$p_{ex}(x, y, z) = \frac{p_{out} - p_{in}}{L} x + p_{in}. \tag{5.4}$$

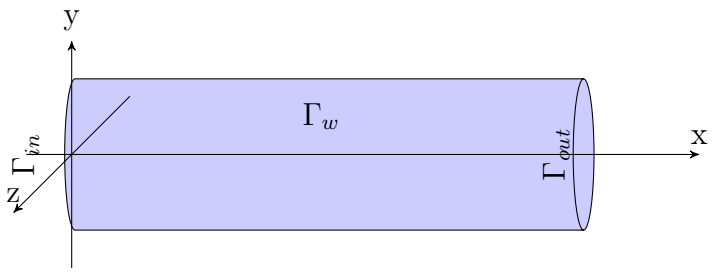


Figure 5.1: Geometry for convergence rates verification Ω

We note that the discrete geometry, Ω_δ, will be different from the exact one, Ω. Hence, the geometry approximation will play a crucial role in the verification process. To measure the geometric approximation error, we not only measure the error in the standard $L^2(\Omega_\delta)$ and $H^1(\Omega_\delta)$ norms, but also the applied forces by integrating the stress tensor on one of the domain boundaries, here Γ_{in}. That is to say, we compute

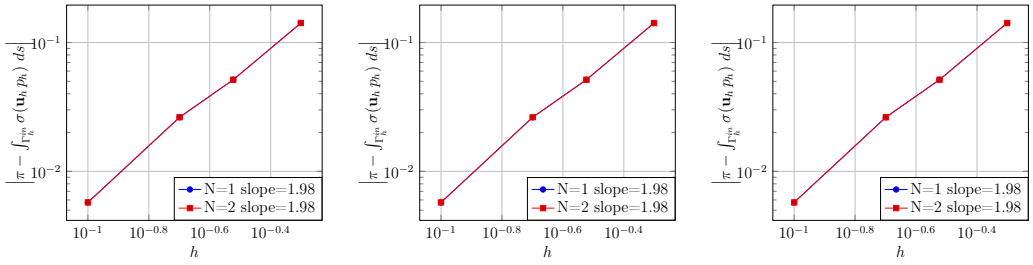
$$\boldsymbol{F}_\delta = \int_{\Gamma_{in,\delta}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \, ds. \tag{5.5}$$

Thanks to (5.3), we can compute by hand the exact counterpart $\boldsymbol{F}_{ex} = \int_{\Gamma_{in}} \boldsymbol{\sigma}(\mathbf{u_{ex}}, p_{ex})\boldsymbol{n} \, ds$ and hence $\|\boldsymbol{F}_{ex} - \boldsymbol{F}_\delta\|_2$ is evaluated. This is the simplest way to take properly into account

the error in the geometry as well as in velocity and pressure. Indeed we have otherwise only access to $L^2(\Omega_\delta)$ and $H^1(\Omega_\delta)$ norms and not $L^2(\Omega)$ and $H^1(\Omega)$. We remark that a similar test case was used in [108] to check for the dependence of the geometry approximation in the numerical convergence verification process.

1.2 Convergence analysis results

We first start with the standard finite element approximations using first order geometry, namely $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ and $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_1$, applied to the Dirichlet-Dirichlet (see Section 2.2.1), Dirichlet-Neumann (see Section 2.2.2) and Neumann-Neumann (see Section 2.2.3) boundary conditions. The results are displayed in Table 5.1 and are quite interesting: even though the geometry is not properly approximated, velocity and pressure error norms are 0 up to machine precision in all cases. This can be explained by the facts that *(i)* the exact velocity is quadratic and the exact pressure linear, see (5.3), *(ii)* the geometric transformation is first order and all volume and surface numerical integrals are computed exactly thanks to FEEL++, hence the only possible solution in the velocity and pressure spaces are the exact velocity and exact pressure, respectively, which we see in the $L^2(\Omega_\delta)$ and $H^1(\Omega_\delta)$ error norms. These results are however somewhat deceptive as they are not taking into account the geometrical approximation as mentioned above. Indeed, if we now look at the error in $\|\boldsymbol{F}_{ex} - \boldsymbol{F}_\delta\|$, it displays only an order 2 convergence rate with respect to h — as expected when using order 1 geometry — not withstanding the polynomial order of the velocity and pressure, see Table 5.1 and Figure 5.2.



(a) Dirichlet-Dirichlet (b) Dirichlet-Neumann (c) Neumann-Neumann

Figure 5.2: Output convergence using $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_1$ approximation spaces, $N = 1, 2$.

In order to improve the approximation properties of $\boldsymbol{F}_\delta$, we now turn to the second order geometry approximation. The results are displayed in Table 5.2. Again the results are interesting: even though the exact velocity and pressure are quadratic and linear, respectively, the finite element approximations are not exact. First recall that, see (2.40), it is not the finite element approximations in the real element that are polynomials of degree N, but the finite element approximations in the reference element $\hat{K}$, and second that the geometric transformation is no longer linear. Hence the numerical integrations are not exact as the integrands are no longer polynomial when derivatives are involved — thanks to the chain rule — . We recover however very good convergence rates and in fact we have super convergence — one order more than expected — . This is due to the symmetries of the cylinder. Finally, we plot the $\boldsymbol{F}_\delta$ convergence in Figure 5.3 for $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_N$, $N = 1, 2$ as well as $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_2$. The later case shows that increasing just the geometrical approximation improves already $\boldsymbol{F}_\delta$ tremendously. To summarize, to handle non linear

(a) Dirichlet-Dirichlet: $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$2.0\cdot10^{-16}$	$2.1\cdot10^{-15}$	$4.9\cdot10^{-16}$	$1.4\cdot10^{-1}$	–
0.3	$1.2\cdot10^{-16}$	$3.6\cdot10^{-15}$	$5.5\cdot10^{-16}$	$5.1\cdot10^{-2}$	1.98
0.2	$3.3\cdot10^{-16}$	$1.3\cdot10^{-14}$	$8.4\cdot10^{-16}$	$2.6\cdot10^{-2}$	1.65
0.1	$1.3\cdot10^{-16}$	$1.5\cdot10^{-14}$	$1.7\cdot10^{-15}$	$5.7\cdot10^{-3}$	2.2

(b) Dirichlet-Dirichlet: $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_1$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$1.5\cdot10^{-16}$	$8.4\cdot10^{-15}$	$1.1\cdot10^{-15}$	$1.4\cdot10^{-1}$	–
0.3	$1.8\cdot10^{-16}$	$4.9\cdot10^{-15}$	$1.2\cdot10^{-15}$	$5.1\cdot10^{-2}$	1.98
0.2	$1.9\cdot10^{-16}$	$9.2\cdot10^{-15}$	$1.8\cdot10^{-15}$	$2.6\cdot10^{-2}$	1.65
0.1	$2.1\cdot10^{-15}$	$4.6\cdot10^{-14}$	$4.3\cdot10^{-15}$	$5.7\cdot10^{-3}$	2.2

(c) Dirichlet-Neumann: $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$8.1\cdot10^{-17}$	$2.3\cdot10^{-15}$	$3.9\cdot10^{-16}$	$1.4\cdot10^{-1}$	–
0.3	$8.4\cdot10^{-17}$	$1.4\cdot10^{-15}$	$5.1\cdot10^{-16}$	$5.1\cdot10^{-2}$	1.98
0.2	$1.7\cdot10^{-16}$	$2.0\cdot10^{-15}$	$8.5\cdot10^{-16}$	$2.6\cdot10^{-2}$	1.65
0.1	$1.6\cdot10^{-16}$	$3.9\cdot10^{-15}$	$1.7\cdot10^{-15}$	$5.7\cdot10^{-3}$	2.2

(d) Dirichlet-Neumann: $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_1$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$2.0\cdot10^{-16}$	$5.4\cdot10^{-15}$	$1.1\cdot10^{-15}$	$1.4\cdot10^{-1}$	–
0.3	$2.1\cdot10^{-16}$	$6.7\cdot10^{-15}$	$1.4\cdot10^{-15}$	$5.1\cdot10^{-2}$	1.98
0.2	$2.1\cdot10^{-16}$	$7.1\cdot10^{-15}$	$1.9\cdot10^{-15}$	$2.6\cdot10^{-2}$	1.65
0.1	$4.9\cdot10^{-16}$	$1.3\cdot10^{-14}$	$4.2\cdot10^{-15}$	$5.7\cdot10^{-3}$	2.2

(e) Neumann-Neumann: $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$2.3\cdot10^{-16}$	$3.2\cdot10^{-15}$	$4.6\cdot10^{-16}$	$1.4\cdot10^{-1}$	–
0.3	$1.8\cdot10^{-16}$	$3.1\cdot10^{-15}$	$5.5\cdot10^{-16}$	$5.1\cdot10^{-2}$	1.98
0.2	$1.9\cdot10^{-16}$	$2.7\cdot10^{-15}$	$8.3\cdot10^{-16}$	$2.6\cdot10^{-2}$	1.65
0.1	$1.7\cdot10^{-16}$	$3.6\cdot10^{-15}$	$1.5\cdot10^{-15}$	$5.7\cdot10^{-3}$	2.2

(f) Neumann-Neumann: $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_1$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$2.2\cdot10^{-16}$	$4.2\cdot10^{-15}$	$1.0\cdot10^{-15}$	$1.4\cdot10^{-1}$	–
0.3	$1.8\cdot10^{-16}$	$4.5\cdot10^{-15}$	$1.3\cdot10^{-15}$	$5.1\cdot10^{-2}$	1.98
0.2	$2.0\cdot10^{-16}$	$6.3\cdot10^{-15}$	$1.8\cdot10^{-15}$	$2.6\cdot10^{-2}$	1.65
0.1	$8.0\cdot10^{-16}$	$1.8\cdot10^{-14}$	$3.9\cdot10^{-15}$	$5.7\cdot10^{-3}$	2.2

Table 5.1: Dirichlet-Dirichlet, Dirichlet-Neumann and Neumann-Neumann formulations using first order geometry approximation.

geometries, *i.e.* $\Omega_\delta \neq \Omega$, we not only need to increase the order of approximations in velocity and pressure if we want to improve the accuracy of our simulations, but we need also to increase the order of approximation of the geometry. This, of course, comes at a cost which the Feel++ framework allows to alleviate to find a good balance between h, N and k_{geo}.

(a) Dirichlet-Dirichlet $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_2$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	SlopeP	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	SlopeU	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$6.3\cdot10^{-6}$	$5.8\cdot10^{-5}$	–	$2.2\cdot10^{-5}$	–	$4.6\cdot10^{-4}$	–
0.3	$6.2\cdot10^{-7}$	$8.5\cdot10^{-6}$	3.76	$2.5\cdot10^{-6}$	4.28	$5.7\cdot10^{-5}$	4.1
0.2	$1.2\cdot10^{-7}$	$2.1\cdot10^{-6}$	3.48	$8.9\cdot10^{-7}$	2.52	$1.5\cdot10^{-5}$	3.31
0.1	$6.4\cdot10^{-9}$	$1.4\cdot10^{-7}$	3.88	$7.0\cdot10^{-8}$	3.67	$6.2\cdot10^{-7}$	4.59

(b) Dirichlet-Neumann $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_2$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	SlopeP	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	SlopeU	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$6.3\cdot10^{-6}$	$1.0\cdot10^{-4}$	–	$2.2\cdot10^{-5}$	–	$4.0\cdot10^{-4}$	–
0.3	$6.4\cdot10^{-7}$	$1.7\cdot10^{-5}$	3.55	$2.3\cdot10^{-6}$	4.36	$4.6\cdot10^{-5}$	4.23
0.2	$1.3\cdot10^{-7}$	$3.9\cdot10^{-6}$	3.6	$8.8\cdot10^{-7}$	2.4	$1.2\cdot10^{-5}$	3.21
0.1	$6.8\cdot10^{-9}$	$2.6\cdot10^{-7}$	3.91	$7.1\cdot10^{-8}$	3.62	$4.6\cdot10^{-7}$	4.77

(c) Neumann-Neumann $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_2$

h	$\|\mathbf{u}-\mathbf{u}_\delta\|_{0,\Omega_\delta}$	$\|p-p_\delta\|_{0,\Omega_\delta}$	SlopeP	$\|\mathbf{u}-\mathbf{u}_\delta\|_{1,\Omega_\delta}$	SlopeU	$\|\|\boldsymbol{F}_{ex}-\boldsymbol{F}_\delta\|\|$	Slope
0.5	$8.9\cdot10^{-6}$	$5.2\cdot10^{-5}$	–	$2.3\cdot10^{-5}$	–	$5.1\cdot10^{-4}$	–
0.3	$1.1\cdot10^{-6}$	$4.3\cdot10^{-6}$	4.85	$2.4\cdot10^{-6}$	4.45	$6.7\cdot10^{-5}$	3.98
0.2	$2.3\cdot10^{-7}$	$1.3\cdot10^{-6}$	2.92	$8.9\cdot10^{-7}$	2.45	$1.8\cdot10^{-5}$	3.29
0.1	$1.2\cdot10^{-8}$	$7.0\cdot10^{-8}$	4.25	$7.1\cdot10^{-8}$	3.65	$8.4\cdot10^{-7}$	4.4

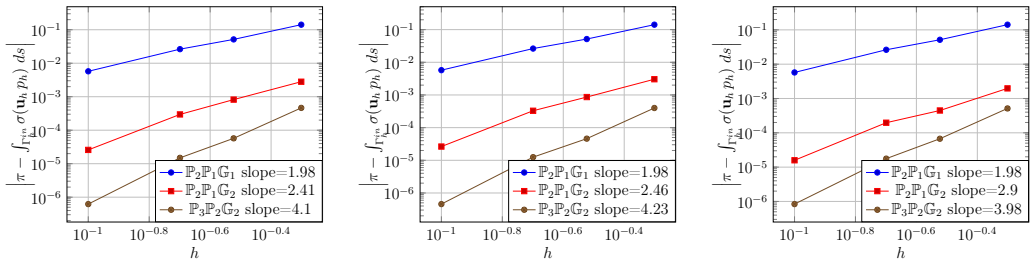
Table 5.2: Dirichlet-Dirichlet, Dirichlet-Neumann and Neumann-Neumann formulations using second order geometry approximation.

1.3 Flow simulation on cerebrovenous system

To exercise our full fledged framework, we perfom now a (incompressible) Navier-Stokes simulation in a realistic geometry, the cerebrovenous system. This geometry represented by Figure 5.4(a) has 29 inlets and 2 outlets. The boundary conditions imposed in this application are not physiological but only used for testing purposes: on each inlet, we impose $\boldsymbol{\sigma}(\mathbf{u},p)\boldsymbol{n} = g_{in}\boldsymbol{n}$ with

$$g_{in} = -0.5\cdot10^5\left(1-\cos\left(\frac{\pi t}{0.0015}\right)\right), \tag{5.6}$$

and at the outlets, we set: $\boldsymbol{\sigma}(\mathbf{u},p)\boldsymbol{n} = \mathbf{0}$ which corresponds to the Neumann-Neumann formulation described in Section 2.2.3. Regarding the physical parameters, we take the density ρ equal to 1kg/m^3 and the dynamic viscosity μ equal $0.003\text{N}\cdot\text{s/m}^2$. The time



(a) Dirichlet-Dirichlet test case (b) Dirichlet-Neumann test case (c) Neumann-Neumann test case

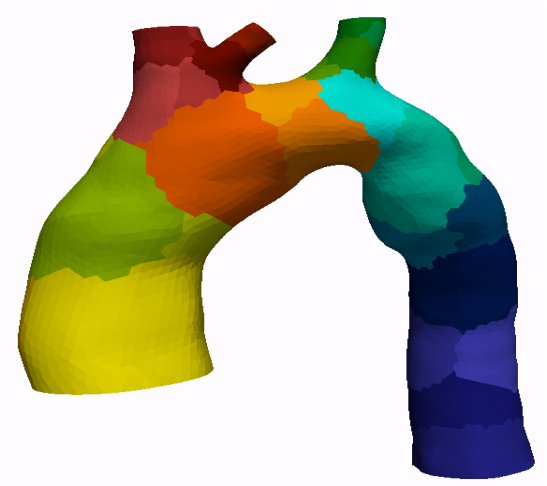
Figure 5.3: $\boldsymbol{F}_\delta$ convergence using $\mathbb{P}_{N+1}\mathbb{P}_N G_N, N = 1, 2$ approximations and $\mathbb{P}_2\mathbb{P}_1 G_2$ approximation.

step is $\Delta t = 10^{-5}$s and we perform a simulation up to $t = 0.003$s. The approximation used is $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$. Table 5.3 shows the number of tetrahedrons in the mesh as well as the number of degrees of freedom associated to the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ approximation.

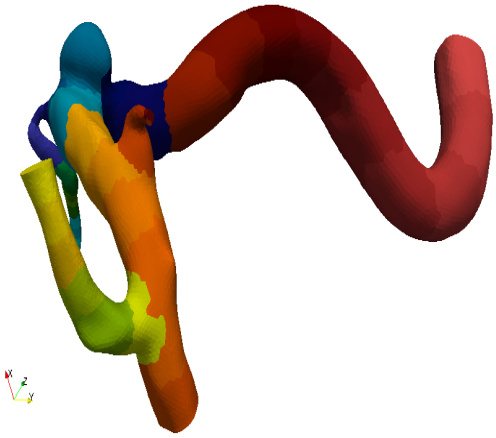
Tetrahedrons	Dof($\mathbf{u}_\delta$)	Dof(p_δ)	Dof(Total)
237,438	1,119,411	54,183	1,173,594

Table 5.3: Number of elements and degrees of freedom using $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ approximations.

(a) A mesh partition (32 partitions) of the cerebrovenous system



(b) Mesh partitioning of an aorta



(c) Mesh partitioning of an artery with an aneurysm

Figure 5.4: Computational meshes of the cerebrovenous system, of a realistic aorta and of an artery with aneurysm geometries.

1.3.1 Scalability analysis results

We now turn to a scalability analysis applied to problems in fluid mechanics using FEEL++. We propose here to study two configurations: *(i)* a Stokes model with a simple geometry (a cylinder), *(ii)* a Navier-Stokes model with complex and realistic geometry (an aorta), presented in Figure 5.4(b), in order to check the strong scalability — increase processing units for a fixed size problem — and weak scalability — increase processing units along with the problem size. Each test will measure *(i)* the assembly CPU time for matrices and vectors as well as the necessary MPI communications, *(ii)* the linear and/or nonlinear algebraic system CPU time.

Regarding the strong scalability, the problem size stays fixed while the number of processing elements increases from $N^{\text{core}} = N^{\text{core}}_{\text{min}}, \ldots, N^{\text{core}}_{\text{max}}$ and we are interested in the

speed-up $= t_{N^{\text{core}}_{\text{min}}}/t_{N^{\text{core}}}$ obtained with N^{core} cores where $t_{N^{\text{core}}_{\text{min}}}$ is the elapsed wall-clock time with $N^{\text{core}}_{\text{min}}$ cores and $t_{N^{\text{core}}}$ the elapsed wall-clock time with N^{core} processors.

As to the weak scalability, the problem size increases with the number of processing units such that the problem size assigned to each processing element remains constant throughout all computations and we measure the *efficiency* $= 100(t_{N^{\text{core}}_{\text{min}}}/t_{N^{\text{core}}})$.

Stokes model:
To perform the scalability test, we consider a cylinder of length L_C and radius 0.5. We impose Dirichlet-Neumann boundary conditions, *i.e.*, a parabolic known velocity profile at the inlet and a zero normal stress at the outlet, see Section 2.2.2.

(a) Strong scalability settings

Mesh	Tetrahedrons	Dof
M8	111,475	530,275
M9	151,443	712,545
M10	210,526	980,105

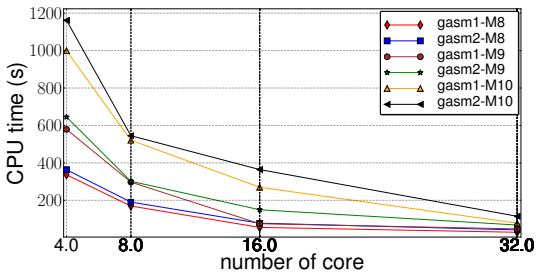
(b) Weak scalability settings

N^{core}	L_C	Dof(M4)	Dof(M5)	Dof(M6)
32	16.0	297,372	398,181	563,574
16	8.0	146,094	197,262	290,423
8	4.0	77,255	105,296	148,027
4	2.0	39,834	56,075	75,005
2	1.0	23,950	31,971	41,869
1	0.5	15,705	19,207	26,523

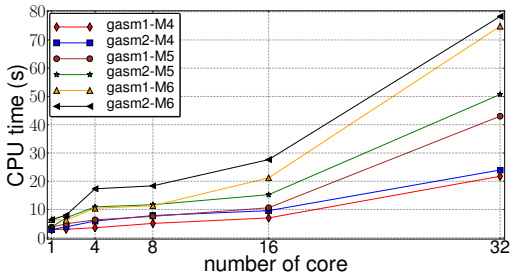
Table 5.4: Configurations used for the scalability tests with the Stokes model and approximation spaces $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$.

The scalability tests of Stokes model were made using the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ approximation spaces. The different configurations used are displayed in Table 1(a) for the strong scalability and in Table 1(b) for the weak scalability. Figure 5.5 displays all the results obtained. We use several meshes associated to a characteristic mesh size which we denote by M4, M5, M6, etc. The larger the index, the finer the mesh. We also compare two parallel preconditioners GASM1 and GASM2. Both precondtioners use a direct subsolver (LU with MUMPS) within the subdomains.

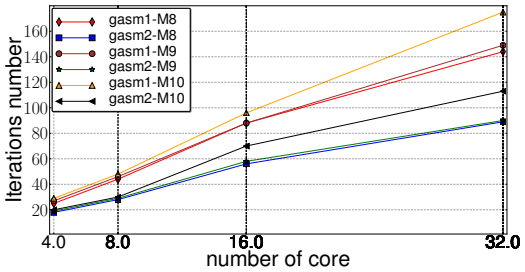
Figures 5.5(a) and 5.5(b) display the elapsed wall-clock CPU time to solve the linear system for strong and weak scalability. We see that the preconditioner GASM1 is more efficient than GASM2 for each scalability test. Figure 5.5(c) plots the number of solver iterations associated to the strong scalability test: the preconditioner GASM2 reduces the number of iteration which is due to the overlap. However, in both cases, the number of iterations increases significantly as we increase the number of cores. A similar behaviour for weak scalability is observed in Figure 5.5(c). In particular when using 32 cores the number of solver iterations increases a lot and we have slow convergence. Figures 5.5(e) and 5.5(f) plot the assembly time of the algebraic structures. The performances are perfect



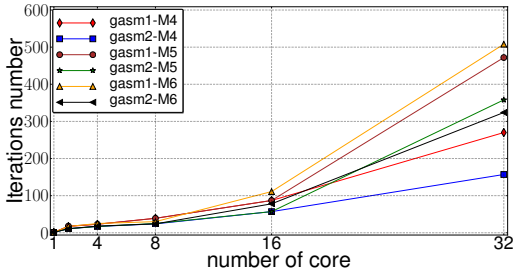
(a) CPU time to solve the algebraic system (strong scalability)



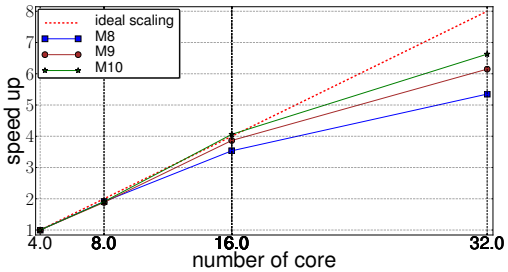
(b) CPU time to solve the algebraic system (weak scalability)



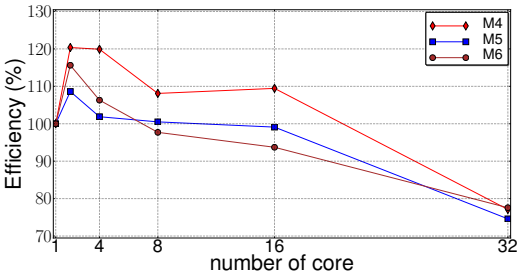
(c) Number of solver iterations (strong scalability)



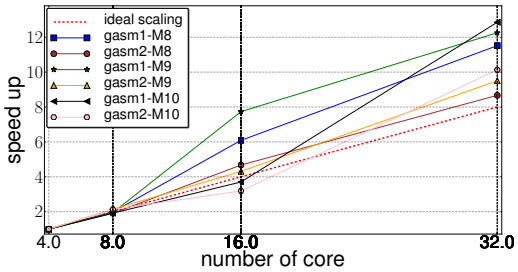
(d) Number of solver iterations (weak scalability)



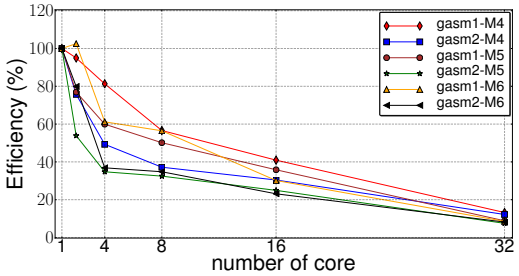
(e) Assembly speed-up (strong scalability)



(f) Assembly efficiency (weak scalability)



(g) Global speed-up (strong scalability)



(h) Global efficiency (weak scalability)

Figure 5.5: Results for the scalability tests with the Stokes model.

with for both scalability tests up to 16 processors and it deteriorates at 32 cores as the communication become dominant. We note however that the speed-up is improving with the finest meshes.

To conclude the overall — the sum of assembly and solves times — measured speed-up is very good, even above an ideal speed-up, see Figure 5.5(g). As to weak scalability, see Figure 5.5(h), performance decreases with increasing the number of cores which is explained by the strong increase in the number of iterations.

Navier-Stokes model

We now turn to the strong scalability test for the Navier-Stokes strategy on a realistic geometry — an aorta, — see Figure 5.4(b). The formulation of this nonlinear model is described with equations (2.1)-(2.2). Regarding the problem setting, we have a nonzero value of normal stress tensor at the inlet, a zero normal stress is set on the four outlets and a no-slip condition on the wall, see Section 2.2.3. We measure the computational time only during the first time step.

Mesh	Tetrahedrons	Dof
M2	42,744	199,371
M3	109,997	497,525
M4	223,120	989,237

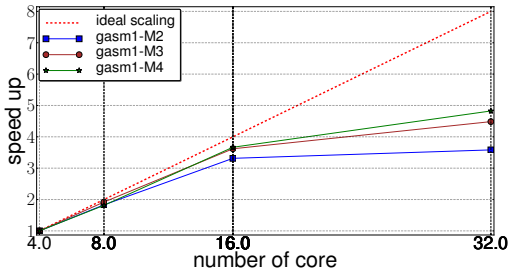
Table 5.5: Configurations used for the Navier-Stokes model with $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ approximations.

The strong scalability tests are obtained using $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ approximations. The number of mesh elements and degrees of freedom are reported in Table 5.5 and the scalability results in Figure 5.6. The speed-up for assembly of the algebraic structures is ideal up to 16 processors but at 32 cores performance stagnates because communications become dominant compared to the cost of local assembly. Indeed we note that speed-up improves with the finest meshes, *i.e.* the local assembly cost is still important. As to the speed-up of the solver strategy, it is excellent, see Figure 5.6(b). Indeed, the number of iterations of the linear solves remains relatively low, see Figure 5.6(d). Also note that the scalability of this application is dominated by the solver, see Figure 5.6(d), and not by assembly: we obtain, in overall, an excellent global strong scalability.

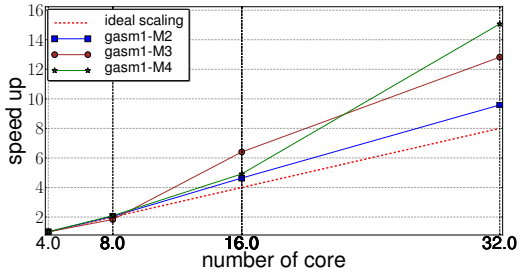
The simulations were carried out on the IRMA-cluster, the HPC cluster of Aix-Marseille University.

More recently, an analysis of the boundary conditions was performed in [32], and the authors found that the effect of setting the inflow boundary condition on the forces created by blood flow, is likely greater than for other modeling assumptions, the other important factor being the blood viscosity model, especially in wall shear stress computations.

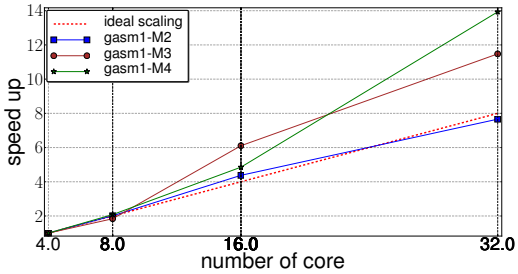
Conclusion: In this first section of this chapter, we have proposed a flexible framework to answer some modeling and computational issues in order to perform large-scale three-dimensional blood flow simulations in realistic geometries. In particular, we have presented different strategies to handle boundary conditions appropriate to blood flow simulation as well as their variational formulation. We have also displayed initial numerical



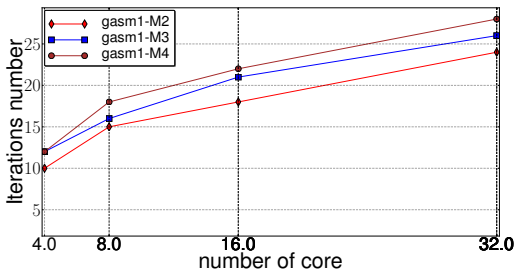
(a) Assembly speed-up



(b) Non-linear solves speed-up



(c) Global speed-up



(d) Number of iterations needed by the first linear solve

Figure 5.6: Results for the strong scalability tests with the Navier-Stokes model.

results on a realistic geometry which now need to be backed up by further mathematical and bio-mechanical modeling.

2 Stress tensor computation methods

2.1 Computational methods

Many investigations have been done in the context of the computation of the drag and the lift, see, for instance, Gresho and Sani [68] in 1998, and Tezduyar et al. in[141]. Recently, some error estimates in stationary flows have been obtained for the drag and the lift, see for instance John, Tabata, and Tobiska in[148], Larson in [65], and Tabata and Itakura in [135]. Drag and lift coefficients of a 2D flow around a cylinder are computed very accurately in [99, 85] and for a 3D flow in [83]. In [84], a numerical study of a time-dependent 2D flow through a channel around a cylinder was carried. In the following, we present a slightly different context, namely the computation of the force exerted by a fluid on a boundary surface Γ_{bottom} with two different methods.
The first method consists of explicitly applying the definition of this force, given by the following formula:

$$\boldsymbol{F} = \int_{\Gamma_b} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \tag{5.7}$$

where $\boldsymbol{\sigma}(\mathbf{u}, p) = -p\mathbf{I} + \mu\nabla\mathbf{u}$.
This integral is on the surface thus, we will denote "surface method" the surface integral

computation of this force. We will also denote

$$\boldsymbol{F}_{ex} = \int_{\Gamma_{b,ex}} \boldsymbol{\sigma}(\mathbf{u_{ex}}, p_{ex})\boldsymbol{n},$$

the exact expression of the force, which can be exactly calculated via a symbolic calculation when the geometry is analytically well know. Finally, we will denote

$$\boldsymbol{F}_{\delta} = \int_{\Gamma_{b\delta}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n},$$

the computed force on the approximated boundary, and

$$\boldsymbol{F}_{\delta,ex} = \int_{\Gamma_{b\delta}} \boldsymbol{\sigma}(\mathbf{u_{ex}}, p_{ex})\boldsymbol{n},$$

the computed exact expression of the force on the approximated boundary.

The second way of obtaining this force is by applying the following steps when retrieving the variational formulation of the following Stokes problem.

Let us recall the Stokes problem: find (**u**,p) such that

$$\begin{aligned} -\nu\Delta\mathbf{u} + \nabla p &= \mathbf{f} \quad \Omega && (5.8)\\ \nabla\cdot\mathbf{u} &= 0 \quad \Omega && (5.9)\\ \mathbf{u} &= \mathbf{u_{exact}} \quad \Gamma_{bottom} && (5.10)\\ \mathbf{u} &= \mathbf{0} \quad \partial\Omega \setminus \Gamma_{bottom} && (5.11) \end{aligned}$$

The standard variational formulation for the stokes problem is retrieved by following the same steps as for the Navier-Stokes problem. It reads as: find $(\mathbf{u}, p)$ in $[H_0^1(\Omega)]^d \times L_0^2(\Omega)$ such that for all $\mathbf{v} \in [H_0^1(\Omega)]^d$, and $q \in L_0^2(\Omega)$ we have

$$\begin{aligned} \nu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} - \int_\Omega p\mathrm{div}(\mathbf{v})\ dx &= 0, && (5.12)\\ \int_\Omega q\ \mathrm{div}(\mathbf{u})\ dx &= 0. && (5.13) \end{aligned}$$

The additional restriction of zero mean value to $L^2(\Omega)$ function allows to uniquely define the pressure in $\mathbb{M} = L^2(\Omega)$ and may be integrated in the variational formulation (5.12)-(5.13) by adding a suitable Lagrange multiplier. The final variational formulation, with the Lagrange multiplier, reads as: find $(\mathbf{u}, p, \zeta) \in [H_0^1(\Omega)]^d \times L^2(\Omega) \times \mathbb{R}$ such that

$$\begin{aligned} \nu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v}\ dx - \int_\Omega p\ \mathbf{div}(\mathbf{v})\ dx &= 0, \quad \forall\mathbf{v} \in [H_0^1(\Omega)]^d, && (5.14)\\ \int_\Omega (q\ \mathrm{div}(\mathbf{u}) + \zeta q)\ dx &= 0, \quad \forall q \in L^2(\Omega), && (5.15)\\ \int_\Omega p\xi\ dx &= 0, \quad \forall\xi \in \mathbb{R}. && (5.16) \end{aligned}$$

Let us now, instead of choosing $\mathbf{v} \in [H_0^1(\Omega)]^d$, choose $\mathbf{v} \in \mathbb{W} = [H^1(\Omega)]^d$. The problem (5.8)-(5.9) leads to: find $(\mathbf{u}, p, \zeta) \in \mathbb{W} \times L_0^2(\Omega) \times \mathbb{R}$ such that $\forall (\mathbf{v}, q, \xi) \in \mathbb{W} \times \mathbb{M} \times \mathbb{R}$ we have:

$$\nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \; dx - \int_{\partial\Omega} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds - \int_\Omega p \; \mathrm{div}(\mathbf{v}) \; dx = \int_\Omega \mathbf{f} \cdot \mathbf{v} dx \quad (5.17)$$

$$\int_\Omega q \; \mathrm{div}(\mathbf{u}) + \zeta q \; dx = 0 \quad (5.18)$$

$$\int_\Omega p\xi \; dx = 0 \quad (5.19)$$

Rewriting 5.17 with $\mathbf{v} = \mathbf{1}$ over Γ_{bottom} and $\mathbf{0}$ elsewhere we obtain:

$$\int_{\Gamma_{bottom}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds = -\int_\Omega \mathbf{f} \cdot \mathbf{v} + \nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \; dx - \int_\Omega p \; \mathrm{div}(\mathbf{v}) \; dx - \int_{\Gamma_{in} \bigcup \Gamma_{out}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds$$

Let us denote

$$R = -\int_\Omega \mathbf{f} \cdot \mathbf{v} + \nu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \; dx - \int_\Omega p \; \mathrm{div}(\mathbf{v}) \; dx - \int_{\Gamma_{in} \bigcup \Gamma_{out}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds \quad (5.20)$$

This is a computation on all the domain, thus we will denote this method the "volume computation".

Remark 19. *The fact that* $\int_{\Gamma_{in} \bigcup \Gamma_{out}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds$ *has not vanished despite having* $\mathbf{v} = \mathbf{0}$ *on* $\partial\Omega \setminus \Gamma_{bottom}$ *is due to the fact that, when computing* $\int_{\Gamma_{bottom}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds$*, the stress tensor must be evaluated on the elements having a common edge with the boundary* Γ_{bottom} *(blue triangles in Figure 5.7), the red vertexes included. However, since we have taken* $\mathbf{v} = 0$ *on* $\Gamma_{in} \bigcup \Gamma_{out} \bigcup \Gamma_{top}$*, the red vertexes of the green triangles of the same Figure 5.7, are no longer considered when replacing* $\int_{\partial\Omega} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds$ *by* $\int_{\Gamma_{bottom}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds$*, while they should. Hence, the contribution of the integral* $\int_{\Gamma_{\Gamma_{in} \bigcup \Gamma_{out}}} \boldsymbol{\sigma}(\mathbf{u}, p)\boldsymbol{n} \cdot \mathbf{v} \; ds$ *needs to be taken into account in* (5.20)*.*

2.2 Benchmark setup

The aim of this section is to evaluate and compare the accuracy of the two aforementioned methods with respect to the exact solution for high order finite elements and high order approximation of the geometry. For that end, let us consider two computational domains:*(i)* a 2D rectangular domain $\Omega = (-0.5, 1) \times (-0.5, 1.5)$, and *(ii)* the same domain for whom we deform the $\partial\Omega_b$ edge according to the law:

$$\boldsymbol{\phi}(\mathbf{Y}) = [\mathbf{Y}, 0.08(\mathbf{Y} + 0.5)(\mathbf{Y} - 1)(\mathbf{Y}^2 - 1)]^T, \quad \mathbf{Y} \in (-0.5, 1)$$

by solving the Laplacian problem:

$$-\Delta \boldsymbol{\mu} = 0 \text{ in } \Omega, \quad (5.21)$$

$$\boldsymbol{\mu} = \boldsymbol{\phi} \text{ in } \Gamma_b, \quad (5.22)$$

$$\boldsymbol{\mu} = \mathbf{0} \text{ in } \partial\Omega - \Gamma_b \quad (5.23)$$

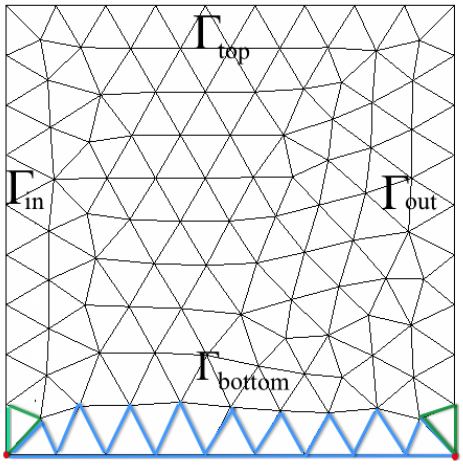


Figure 5.7: Maillage 2D

Then we affect this displacement to the initial mesh using the *meshMove* function of FEEL++.

Let us also consider the non-polynomial Kovaznay solution [92] of the Stokes problem:

$$\mathbf{u}(x,y) = [1 - e^{\lambda x}cos(2\pi y), \frac{\lambda}{2\pi}e^{\lambda x}sin(2\pi y)]^T \tag{5.24}$$

$$p(x,y) = -\frac{e^{2\lambda x}}{2} \tag{5.25}$$

with

$$\lambda = \frac{1}{2\nu} - \sqrt{\frac{1}{4\nu^2} + 4\pi^2} \tag{5.26}$$

$$\nu = 0.035 \tag{5.27}$$

and $\mathbf{f}(x,y) = [e^{\lambda x}\left(\left(\lambda^2 - 4\pi^2\right)\nu cos(2\pi y) - \lambda e^{\lambda x}\right), \frac{\lambda}{2\pi}e^{\lambda x}\nu sin(2\pi y)(-\lambda^2 + 4\pi^2)]^T$.
The importance of the Kovaznay solution comes from the fact that it is not polynomial, therefore the impact of the high order finite elements approximation will be enhanced. We recall that in the previous chapter, the analytical solution was polynomial of degree 2, thus, a $\mathbb{P}_2\mathbb{P}_1$ finite elements choice was already sufficiently accurate.

Choosing a curved domain aims to show the importance of the high-order geometry approximation. We expect, as we increase the order of the geometry approximation, that, when it reaches 4, the order of the polynomial displacement defining the curved boundary, the error produced by the geometry approximation will vanish and the only error remaining will be the one of the finite element approximation.
Thanks to (5.24) and (5.25), we can analytically compute the exact counterpart $\boldsymbol{F}_{ex} = \int_{\Gamma_{b,ex}} \boldsymbol{\sigma}(\mathbf{u_{ex}}, p_{ex})\boldsymbol{n}\ ds$ and hence evaluate the error of the two aforementioned methods with respect to the exact value. In the following notation $\|F_{ex} - stress\|_2$, *stress* will refer to either the surface method which we denote S or the volume method which we denote V.

2.2.1 Convergence analysis results on a straight boundary

Figure 5.8 shows the convergence plots for the volume and surface methods of computing the force applied by the fluid on the bottom boundary. The error are evaluated with respect to the Kovasnay solution on the straight boundary, for the $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_1$. The results show that the volume method gives more accurate results than the surface method and a higher convergence order for all the finite elements, at least half an order is gained with volume method.

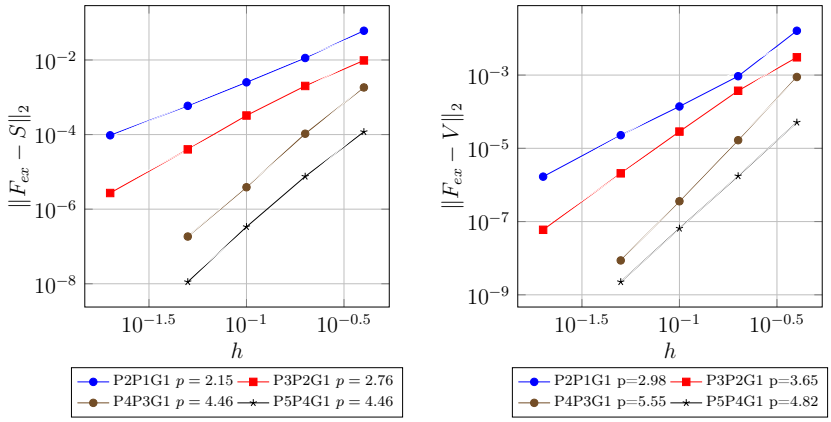


Figure 5.8: Convergence analysis for the $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_1$ configuration on a straight boundary.

We shall mention that, since the boundary where we are evaluating the force on is straight, a $\mathbb{G}_1$ geometric transformation order approximates exactly the boundary of the geometry, and the results of figure 5.9 confirm this. Keeping the same finite element order and increasing the geometry transformation order didn't improve the results for the two methods.

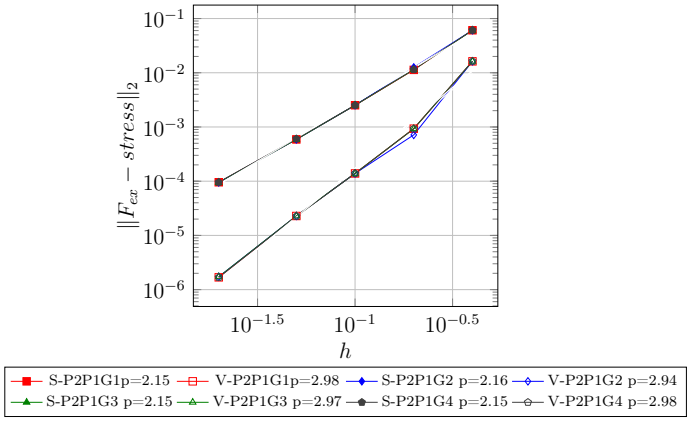
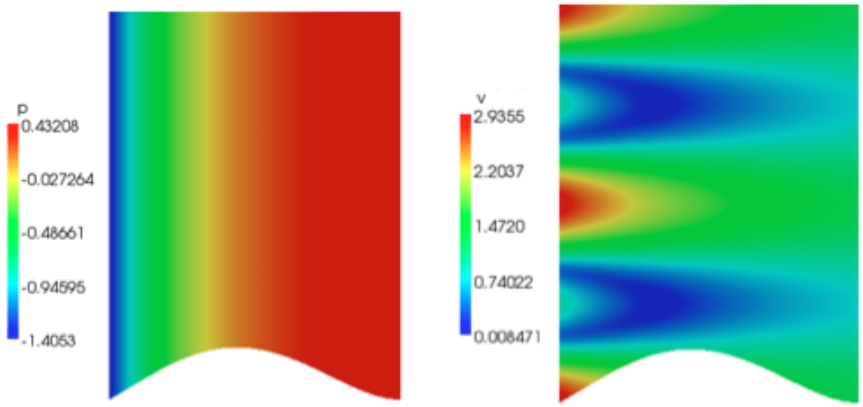


Figure 5.9: Convergence analysis for the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_N$ configuration on a straight boundary.

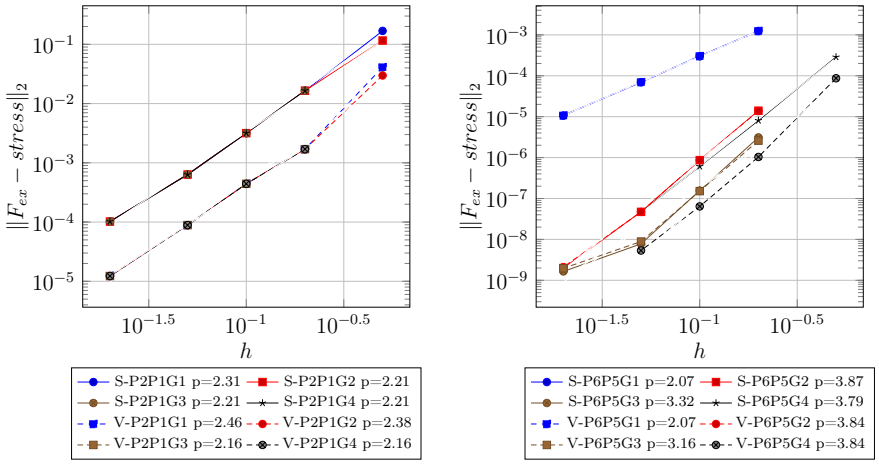
2.2.2 Convergence analysis results on a curved boundary

The plot of the domain's shape, the velocity magnitude and pressure profile is depicted in Figure 5.10.



(a) Pressure profile (b) Velocity magnitude profile

Figure 5.10: Velocity magnitude and pressure profile of the Kovaznay solution on the curved domain for a $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ element on a curved boundary.



(a) $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_N$ (b) $\mathbb{P}_6\mathbb{P}_5\mathbb{G}_N$

Figure 5.11: Convergence analysis for the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_N$ and $\mathbb{P}_6\mathbb{P}_5\mathbb{G}_N$ configurations on a curved boundary.

The results of Figure 5.11 show that, for the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_N$ configuration, for the same method, all the plots overlap. This means that the finite element approximation takes over the geometry approximation. For a low order finite element approximation, here $\mathbb{P}_2\mathbb{P}_1$, increasing the geometric transformation order is not improving the convergence nor decreasing the error. The results of the $\mathbb{P}_6\mathbb{P}_5\mathbb{G}_N$ configuration show a more interesting behaviour. The convergence plots of the two methods, for the same geometric transformation order overlap up to $N = 3$, however, for $N = 4$, which is polynomial order of the boundary curve, the volume method gives better convergence results than the surface method. Which first means that, when the finite element error vanishes due to a high order finite element approximation the geometry approximation takes over the accuracy of the two methods, and it is when the geometry is exactly approximated that we can see the difference between the accuracy of the two methods.

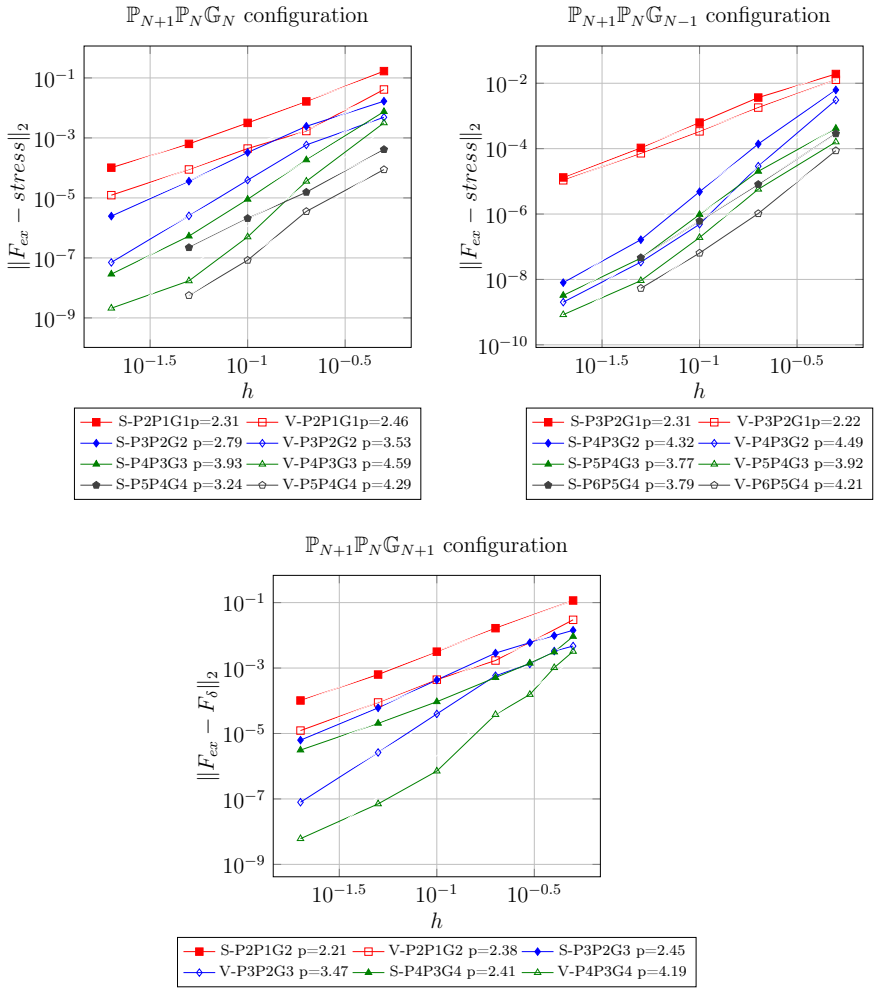


Figure 5.12: Convergence analysis for the $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_N$, $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_{N+1}$ and $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_{N-1}$ configurations on a curved boundary.

The results in Figure 5.12 show that for all the three configurations, $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_N$, $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_{N+1}$ and $\mathbb{P}_{N+1}\mathbb{P}_N\mathbb{G}_{N-1}$, the volume method is always more accurate than the surface method and gives better convergence order than the iso-parametric configuration
.

Conclusion

In the first section of this chapter, we did a convergence analysis for the different weak formulations presented in Chapter 2. In particular, we were interested in discretization errors, not only with respect to the physical fields (velocity and pressure), but also with respect to high order geometry approximations for various boundary conditions settings allowing for a flexible framework with respect to the type of input data (velocity, pressure, flow rate, ...). In the second part, we presented and compared two different methods for the computation of the stress tensor for low and high order finite element and geometry approximations on a straight and curved boundary. We showed that the volume method is more accurate than the surface method in all the above cited cases, a conclusion that should be further investigated from a theoretical point of view.

Chapter 6

Verification and validation of the numerical solution of the Navier Stokes system

The aim of this chapter is to assess to efficiency of the block preconditioner framework for the 3D steady and unsteady Navier-Stokes equations presented in Chapter 4. In particular, we are interested in preconditioners based on an algebraic factorization of the system's matrix which exploit its block structure, such as the Pressure Convection-Diffusion (PCD) preconditioner, the SIMPLE preconditioner or the LSC preconditioner. A comparison between the efficiency of the PCD and LSC preconditioners is ascertain by testing them over the 3D backward facing step benchmark described in the first section of this chapter. In the second section we describe a framework for the solution of flow problems relevant to biomechanics strongly supported by the aforementioned solving strategies. We assess the efficiency of this framework through experimental data for fluid flow in a nozzle model with rigid boundaries, a device designed to reproduce acceleration, deceleration and recirculation, features commonly encountered in medical devices. The flow rates were chosen to cover laminar, transient and turbulent regimes.

Contents

1 The backward facing step benchmark

1.1 Introduction

The backward-facing step (BFS) flow in a channel has been extensively studied in the last two decades, both numerically and experimentally. Despite its simple geometry, flow over the BFS shows some features of more complex geometry flows (i.e. separation, recirculation, reattachment), depending on Reynolds number and some geometrical parameters. Due to this fact, and also because the results of the numerical computations can be usefully compared with experimental data, it represents a good test case for any new numerical methodology. One of the first experimental investigations on this subject was published by Armaly et al. (1983) [8], providing experimental and numerical results for a backward-facing step flow being bounded in a nominally two-dimensional channel. The investigations covered a wide Reynolds number range of $70 \leq Re \leq 8000$, Up to $Re = 400$, good agreement between experimental and numerical results was found. For higher Re, the primary separation length was systematically under-predicted compared with the experiment. The authors explained this discrepancy by the occurrence of three-dimensional effects that could not be covered by their two-dimensional computations. Williams & Baker (1997) [151] have studied internal backward-facing step flows, they found that a transverse flow occurs immediately behind the step that flows from the sidewalls of the channel to its centre. This transverse flow increases in strength with increase in Reynolds number. These kinds of wall jets were confirmed by Chiang & Sheu (1999) [34], Nie & Armaly (2002) [100, 9] and Biswas, Breuer & Durst (2004) [21]. Instabilities of the step flow were investigated numerically by Kaiktsis, Karniadakis & Orszag (1991, 1996) [86, 87], using the same expansion ratio as Armaly et al. (1983). A three-dimensional stability analysis without sidewalls at an expansion ratio of $r = 2$ was performed by Barkley, Gomes & Henderson (2002) [13]. Recently, Blackburn, Barkley & Sherwin (2008) [22] carried out a detailed investigation of the convective instability and transient growth in flows over a backward facing step in the Reynolds number range $0 - 500$ (based on the step height and the peak inflow velocity). In the literature it is also possible to find studies that investigate the numerical stability of the flow over a backward facing step. For example, Fortin et al. and Barkley et al. [13] showed that their two-dimensional computational stability analysis show that the flow over a backward facing step is temporally stable at high Reynolds numbers, that is to say there exist steady solutions of the flow over a backward-facing step at high Reynolds numbers. Cruchaga et al. (1998) [38] solved the steady backwardf acing step flow using finite element method and obtained steady numerical solutions up to $Re = 5500$.
A thorough recent review on the investigations on the BFS flow may be found in the work of Erturk (2008) [51] or Schafer et al. (2009) [124].

1.2 Numerical benchmark setup

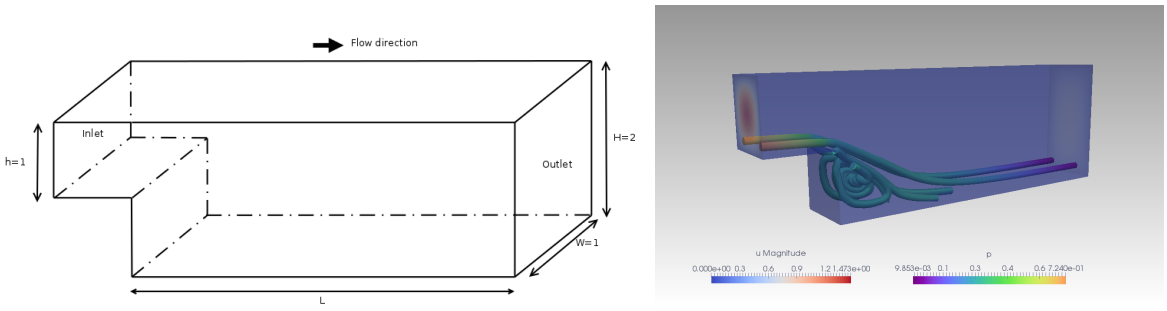
This section is dedicated to assess the performance of the block-type preconditioners, described in Chapter 4, by presenting the results of some computational experiments and convergence analysis over the backward facing step benchmark problem. Our main target is to show the non-dependance of the convergence on the mesh size, the Reynolds number and the polynomial order of the finite elements. We will also highlight the efficiency of

the iterative methods used to handle the Poisson equations and the convection-diffusion equations that arise as subproblems. In these tests we used the boundary conditions adjustments for PCD, presented in [49], the old version of these preconditioners being presented in [47].

Let us consider the backward-facing step geometry illustrated in Figure 1.2, which is an example of an inflow/outflow problem. The relative contribution of the convection and diffusion are defined by the Reynolds number:

$$Re = \frac{UD}{\nu} \tag{6.1}$$

where U is the mean of the velocity at the inflow, D is the characteristic length scale of the domain, here the width of the step ($D = 2$) and ν the kinematic viscosity.
The inflow is at $x = -1$ and the outflow is at $x = 5$ for $Re = 10$ and $Re = 100$, at $x = 10$ for $Re = 200$, and at $x = 20$ for $Re = 400$



(a) Computational domain.

(b) Velocity profile with streamlines colored with pressure

Figure 6.1: Computational domain and velocity profile with streamlines.

A Poiseuille flow profile is imposed on the inflow boundary, a no-flow condition is imposed on the wall and a Neumann condition is applied at the outflow boundary. The 2D Poiseuille profiles are respectively defined as follows:

$$\begin{aligned} u_x &= 6y(1-y) \\ u_y &= 0 \end{aligned} \qquad \begin{aligned} u_x &= 24y(1-y)z(1-z) \\ u_y &= 0 \\ u_z &= 0 \end{aligned}$$

We deal with the nonlinear system arising from the discrete Navier-Stokes equations by using Picard iterations. The initial iterate $(\mathbf{u}_0, p_0)$ is obtained by solving the corresponding discrete Stokes problem.
The stopping criterion of the nonlinear iteration is when the vector Euclidean norm of the nonlinear residual has a relative error of 10^{-5}, that is

$$\| \begin{pmatrix} \mathbf{f} - \left(\mathbf{F}(\mathbf{u}^m)\mathbf{u}^{(m)} + B^T p^{(m)}\right) \\ \mathbf{g} - B\mathbf{u}^{(m)} \end{pmatrix} \| \leq 10^{-5} \| \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix} \| \tag{6.2}$$

As for the starting vector for the linearized iteration it is set to zero and the stopping criterion is

$$\|\mathbf{r}^{(k)}\| \leq 10^{-6} \|\mathbf{s}^{(m)}\|, \tag{6.3}$$

where $\mathbf{r}^{(k)}$ is the residual of the linear system and $\mathbf{s}^{(m)}$ is the left-hand side residual in (6.2) associated with the final nonlinear system.

2D		$L = 5$			$L = 10$	
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	7,739	18,775	34,737	14,716	35,529	65,804
0.0625	30,172	73,209	135,932	57,617	139,959	260,017
0.03125	119,120	290,097	539,673	227,011	553,155	1,028,595
0.015625	472,615	1,152,829	2,146,860	900,446	2,197,573	4,092,058

2D	$L = 20$		
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	28,706	69,389	128,594
0.0625	112,084	272,249	506,260
0.03125	441,902	1,077,093	2,004,430
0.015625	1,756,774	4,912,653	9,149,642

Table 6.1: Total number of DOF for the 2D step geometry for $L = 5$, $L = 10$ and $L = 20$ with $\mathbb{P}_2\mathbb{P}_1$, $\mathbb{P}_3\mathbb{P}_2$ and a $\mathbb{P}_4\mathbb{P}_3$ configurations.

3D		$L = 5$			$L = 10$	
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	121,562	411,494	984,705	231,786	787,645	1,889,149
0.09375	281,943	966,615	2,329,328	535,256	1,841,941	4,447,255
0.0625	844,291	2,996,831	7,130,915	1,685,670	5,882,155	14,305,517

3D	$L = 20$		
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	454,492	1,548,313	3,718,399
0.09375	1,034,805	3,564,749	8,611,636
0.0625	3,281,688	11,459,239	27,879,297

Table 6.2: Total number of DOF for the 3D step geometry for

To solve the linear problem associated to each Picard iteration, we use the GCR algorithm described in Chapter 3, with $restart = 100$, a Krylov method which supports non-symmetric matrices and permits the use of a preconditioner which may vary from one iteration to the next [46].

1.3 Numerical results

The analyses presented in the sequel were partly performed on Curie at TGCC France thanks to a Prace and Genci allocations, which provide 80.000 cores and a peak performance of $3\,\mathrm{Pflop}\cdot\mathrm{s}^{-1}$, as well as a local cluster offering 96 cores with $0.44\,\mathrm{Tflop}\cdot\mathrm{s}^{-1}$

as peak performance respectively. The computations, including post-processing analysis, were distributed from 32 to 2048 cores depending on the cost of the study.

	Re10			Re100		
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	14[5]	13[5]	13[5]	17[14]	17[15]	16[15]
0.0625	13[5]	12[5]	12[5]	16[15]	15[15]	15[15]
0.03125	12[5]	11[5]	11[5]	15[15]	14[15]	13[15]
0.015625	11[5]	10[5]	9[5]	13[15]	12[15]	12[15]

	Re200			Re400		
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	21[26]	20[26]	19[26]	44[57]	23[57]	22[57]
0.0625	19[26]	18[26]	17[26]	22[57]	21[57]	20[57]
0.03125	17[26]	16[26]	15[26]	20[58]	19[57]	18[57]
0.015625	16[26]	14[26]	13[26]	19[57]	17[57]	16[58]

Table 6.3: Number of GCR linear iterations at the last nonlinear iteration of the Picard algorithm and [the total number of nonlinear iterations] for the 2D step geometry for $Re = 10, 100, 200$ and $Re = 400$.

	Re10			Re100		
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	22[4]	20[4]	18[4]	16[9]	14[9]	13[9]
0.09375	21[4]	18[4]	17[4]	15[9]	13[9]	12[9]
0.0625	19[4]	17[4]	16[4]	13[9]	12[9]	11[9]

	Re200			Re400		
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_4\mathbb{P}_3$
0.125	17[11]	15[11]	13[12]	21[14]	16[15]	15[15]
0.09375	16[11]	14[11]	12[11]	17[14]	15[15]	–
0.0625	14[12]	12[11]	–	16[15]	14[15]	–

Table 6.4: Number of GCR linear iterations at the last nonlinear iteration of the Picard algorithm and [the total number of nonlinear iterations] for the 3D step geometry for $Re = 10, 100, 200$ and $Re = 400$, non symmetric formulation.

In the following, we have chosen the *Multigrid* methods [70, 142] for each of the sub-problems arising from the PCD block-partitioning. This choice is motivated by the fact that *Multigrid* methods provides good efficiency and strong scalability results for the Laplacian problem on one hand, and on another hand they optimize work/memory complexity. They also explicitly decouples the problem into two parts: the coarse grid space for scaling and an iterative solver (the smoother) to solve the physics.

	Re10		Re100		Re200		Re400	
h	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$	$\mathbb{P}_2\mathbb{P}_1$	$\mathbb{P}_3\mathbb{P}_2$
0.125	19[4]	18[4]	18[9]	17[9]	19[11]	17[11]	20[14]	18[16]
0.09375	18[4]	17[4]	17[9]	15[9]	18[11]	16[11]	19[14]	17[16]
0.0625	17[4]	15[-]	16[9]	14[9]	16[11]	14[11]	17[16]	15[16]

Table 6.5: Number of GCR linear iterations at the last nonlinear iteration of the Picard algorithm and [the total number of nonlinear iterations] for the 3D step geometry for $Re = 10, 100, 200$ and $Re = 400$, symmetric formulation

A special treatment was applied on the convection-diffusion $\mathbf{F_u}$ sub-matrix. Since the main features are in the diagonal blocks, and the extra diagonal blocks contain only the coupling between the velocity derivatives from the diffusion term, we choose to extract the block diagonal part of the $\mathbf{F_u}$sub-matrix and construct a one big block diagonal matrix. To our best knowledge this technique was never used for the velocity block matrix. In [42] $\mathbf{F_u}$ was replaced by its diagonal in the first and third blocks.
We choose the use of additive *Fieldsplit* (*bJacobi*) with a relative tolerance of 10^{-9} to extract the diagonal blocks of $\mathbf{F_u}$, and we apply a geometric algebraic *Multigrid* preconditioner on each block with a relative threshold of 10^{-6} to drop the edges from the aggregation graph and one smoothing step.

h	$Re = 10$	$Re = 100$	$Re = 200$	$Re = 400$
0.125	33[5]	43[11]	50[13]	79[17]
0.09375	37[5]	50[11]	59[14]	84[17]
0.0625	39[5]	47[11]	57[14]	95[17]

Table 6.6: LSC: $\mathbb{P}_2\mathbb{P}_1$ elements using (GCR) GASM+LU for the sub-problems and a tolerance of 10^{-6}

A standard choice for the approximation of $\mathbf{A}_p$ is using *Multigrid*, for it provides a good efficiency and strong scalability results for pseudo-laplacian algebraic operators. For our simulations we used, for both $\mathbf{A}_p$ and $\mathbf{Q}_p$ operators, the multiplicative GAMG with one level of pre smoothing performed by a local symmetric SOR and a LU factorization for the solve on the coarse level with a tolerance of 10^{-5}.

In order to evaluate the performance of the preconditioner and its sensitivity with respect to the Reynolds number and to the mesh size h, we will compare the number of GCR iterations required to solve the linear system arising from the last step of the nonlinear iteration. In the literature, this study was only done for $\mathbb{P}_2\mathbb{P}_1$ and $\mathbb{Q}_2\mathbb{Q}_1$ finite elements. In the following, we extend this study to high order Taylor-Hood finite elements $\mathbb{P}_3\mathbb{P}_2$ and $\mathbb{P}_4\mathbb{P}_3$.

In Tables 6.1 and 6.2, we show the size of our 2D and 3D problems in term of degrees of freedom for different *i*) levels of refinement, *ii*) lengths of the step and *iii*) polynomial finite elements order.

Remark 20. *The absence of values in some cases of table 6.5 is due to the memory issues for a large size problem. The simulations were run on the IRMA-cluster on 96 processor units.*

The number of the GCR linear iterations at the last nonlinear iteration of the Picard algorithm and the total number of nonlinear iterations required to solve the Navier-Stokes problem are presented in Tables 6.3 and 6.5. As expected, it is independent on the variation of the mesh size and mildly dependent on the variation for the Reynolds number for 2D and 3D computational domain. We can also see that, for high order finite elements approximation, the number of GCR iterations, for the 2D and 3D problem, slightly varies, which allows to extend the independence of the convergence to high order finite elements too.
A comparison with the results of the LSC preconditioner, reported in Table 6.6, shows that the latter has the same behaviour as the PCD preconditioner but takes few more iterations to converge.

We must mention in this context that the LSC preconditioner here used is the PETSc version of this preconditioner that does not include yet the boundary conditions improvements of [49]. It is based on the first version of this preconditioner introduced in [129]. However, the PCD preconditioner here used is the in-house implemented preconditioner, and we took into consideration, while coding this preconditioner, the latest improvements brought in [49]. This difference may explain the difference between the performance of the two preconditioners, the literature predicts a mildly better performance for LSC for $\mathbb{Q}_N\mathbb{Q}_{N-1}$ finite elements type, for both inclosed or inflow-outflow problems [49].

Remark 21. *We are not showing results for SIMPLE preconditioner for the backward facing step problem for it doesn't give good convergence results for steady state problems.*

2 The FDA benchmark

2.1 Benchmark description

In the previous section, we focused on the verification of our numerical choices, looking only at the preconditioner performance in term of number of iterations required to achieve a certain relative tolerance without taking care of the precision of the retrieved solution. In this section, the reliability of CFD simulation and the validation of our numerical choices are now assessed by evaluating the precision of the computed solution regarding experimental data outputs available online to the scientific community for the FDA benchmark [1].

The goal of the FDA challenge is to assess the reliability of CFD simulation by comparing their output with experimental data available online to the scientific community. The benchmark consists of performing flow visualisation experiments of an incompressible Newtonian fluid with prescribed density and viscosity ($\rho_f = 1056kg/m^3$ and $\mu_f = 0.0035Pa\cdot s$) for three flow rates spanning laminar, transitional and turbulent regimes in a rigid domain representative of a medical device shaped like a nozzle. This idealised device was designed to feature accelerating, decelerating and recirculating flow, all of commonly encountered in real medical devices. Its geometry consists of a cylindrical channel with a diameter of $0.0012m$, a conical collector leading to a throat section of $0.004m$ of diameter ending with a sudden expansion in a cylindrical channel with the same diameter of the first one (see Figure 6.2). To validate the computational fluid dynamics simulations, three

[1] https://fdacfd.nci.nih.gov/interlab_study_1_nozzle/data

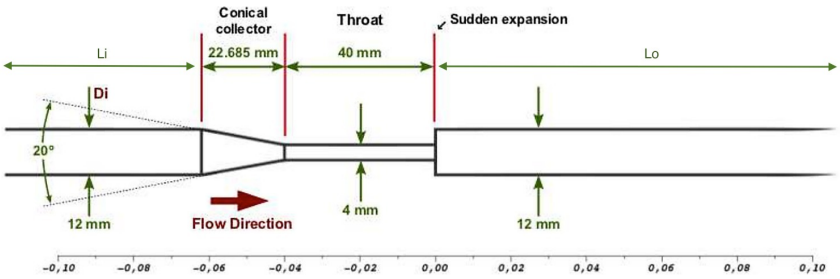


Figure 6.2: Geometry description.

laboratories have made, separately, data acquisitions on fabricated physical models using particle image velocimetry technique over a range of Reynolds numbers (see Table 6.7) on different sections carefully chosen in the geometry so that all the fluid's behavior can be detected. For comparison and analysis purposes, participants were requested to provide simulation data along the model centerline, at various radial cuts (defined in figure 6.3), and along the wall. Flow variables requested along the centerline and radial cuts were the velocity components, pressures (centerline only).

Participants were free to choose the solver, mesh density, element shape, inlet/outlet lengths, inlet/outlet boundary conditions, turbulence model (if needed), and all other parameters of their simulations, according to their individual preference and experience. Participants were free to perform simulations on as many grids as they thought necessary, and the production grid used for the other flow rates could be any one of the grids. More then 28 laboratories have accepted the challenge and have submitted their CFD results. Some were rejected because their simulations give a mass conservation error greater than 10% [71, 133].
The aim of the following work is to reproduce the nozzle benchmark and compare our numerical results to the experimental data for the validation of CFD simulations.
In this section, we provide a detailed report on the methodology we used for our work to be reproducible, and the validation metrics that allow the comparison with the experimental data.

Re_i	Re_t	FLOW RATE
167	500	$5.21 \cdot 10^{-6}$
667	2,000	$2.08 \cdot 10^{-5}$
1,167	3,500	$3.64 \cdot 10^{-5}$

Table 6.7: Reynolds number in the throat section Re_t, Reynolds number at the inlet section Re_i and flow rate for the flow regimes under consideration.

2.2 Validation metrics

First, we compare the numerical wall pressures difference with respect to the pressure at $z = 0$ to experimental data measurements.

$$\Delta p^{norm} = \frac{p_z - p_{z=0}}{\frac{1}{2}\rho_f \overline{u}_t^2} \qquad (6.4) \qquad \text{where} \quad \overline{u}_t^2 = \frac{4Q}{\pi D_t^2}$$

This pressure difference is normalized with respect to the average velocity at the throat $\overline{u}_t$, where Q is the volumetric flow rate retrieved from the Reynolds number at the throat. Then we will compare the normalized axial velocity along i)the centerline (z axis) and ii) at the various radial sections shown in Figure 6.3.

$$u_z^{norm} = \frac{u_z}{\overline{u}_i} \qquad (6.5) \qquad \text{where} \quad \overline{u}_i = \frac{4Q}{\pi D_i^2}$$

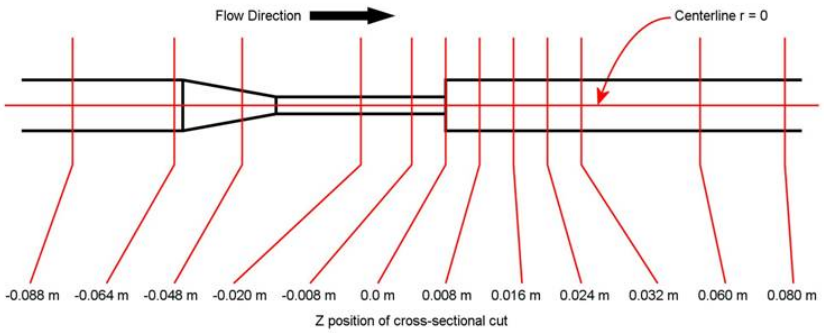


Figure 6.3: Radial sections at which the numerical results are compared with experimental data.

One of the fundamental laws that the simulations must obey to is the conservation of mass law. Since blood is an incompressible fluid, the volumetric flow rate should remain constant as a function of axial position. Which leads to the following conservation of mass error metric proposed in [71]:

$$E_Q = \frac{Q_{CFD} - Q_{theory}}{Q_{theory}}.100\%, \qquad (6.6)$$

where Q_{CFD} is the volumetric flow rate locally evaluated at each of the axial sections shown in Fig 6.3, by integrating the axial velocities along the radius. Q_{theory} is the theoretical volumetric flow rate calculated from the throat Reynolds number. High values of this metric indicate a problem with the numerical model performance. Note that simulations with conservation of mass errors $> 10\%$ were discarded fro the initial comparison [71].

A generic validation metric E_z was also defined on each of the radial sections to quantify the ability of fit between a simulation and the averaged experimental measurements:

$$E_{z,\theta} = \frac{1}{n}\sum_{i=1}^{N} \left| \frac{\overline{u}_{e,i} - u_{c,i}}{\overline{u}_{e,i}} \right| \qquad (6.7)$$

where $E_{z,\theta}$ is the validation metric with respect to the set of experimental and CFD data located on the line that forms a θ angle with the x axis. $\overline{u}_{e,i}$ is the average of the experimental velocity data at one discrete point i along the radial cuts, $\overline{u}_{c,i}$ is the CFD data at the same point i, and n is number of discrete points. In fact, three experimental velocity profiles are provided by each of the laboratories for a given z, for an angle θ equal

to 0°, 45° and 90° respectively. Each velocity profile is presented like a set of couples, where the first component represent the radius of the point and the second component represents the corresponding u_z. Consequently, to evaluate the E_z metric, the mesh size parameter h will separate our discretization points i, and each couple of experimental data, with a radius between $i + h$ and $i - h$ is considered as a neighbor of i, and its corresponding velocity component u_z is taken into consideration while evaluating the $\overline{u}_{e,i}$ quantity. For example, in Figure 6.16, in order to evaluate $E_{z,45°}$ we considered two discrete points (green dots), i and $i + 1$, separated by a space step h. We assume that the red and blue points represent two sets of experimental data. The experimental data points considered as neighbours of the discrete point i are those located at a distance smaller than h from i.

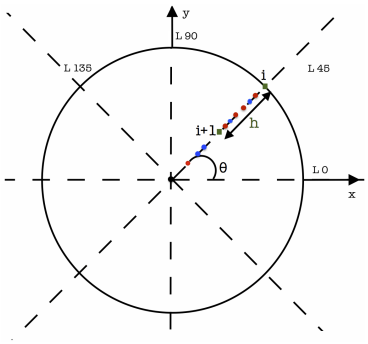
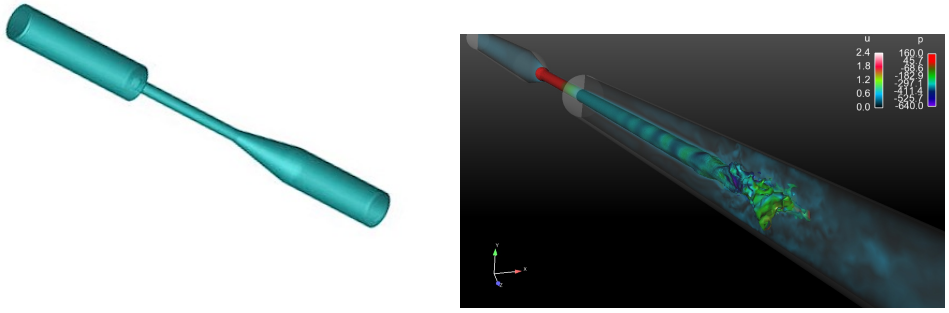


Figure 6.4: Ez metric computation illustration.

2.3 Numerical strategy

The geometry and the mesh were carried out using GMSH [64], and the partitioning was done using METIS [89]. The volume was obtained by, first constructing the 2D longitudinal cut across the centreline of the 3D geometry on which we applied four consecutive rotations of $\pi/4$.
A special characteristic length was attributed to each of the points defining the 2D longitudinal cut in order to have the desired spacial refinement on the regions of interests in the geometry. An "h_{min}" value was applied on the points which belong to the throat regions, and for $Re_t = 2000$, and $Re_t = 3500$, on the points which belong to the radial sections where the metrics are calculated. An "h_{max}" value was set at the *inlet* and *outlet* sections. And finally, an "h_{avg}" value was prescribed on the points belonging to the convergent region. For $Re_t = 500$ we consider four meshes with different refinement levels: M0, M1, M2 and M3. As for $Re_t = 2000$ and $Re_t = 3500$ we only took one mesh sufficiently fine, see Table 6.8 for the characteristic lengths of each mesh.The same meshing strategy was used in [105]. Table 6.9 shows the characteristics of the performed simulations in terms of time step used, the total simulation run time and the final time, the total number of degrees of freedom, the number of CPU units used, and the machine where the simulations were run.

For the following simulations we choose a non-symmetric formulation of the deformation tensor. At the inlet we subscribed a smoothed Poiseuille velocity profile so that it transits from a fluid at rest to the desired regime flow rate reported in Table (6.7). Therefore, we start our simulations with $p_0 = 0$ and $\mathbf{u_0} = \mathbf{0}$. At the outlet we prescribed a



(a) Computational domain

(b) Axial velocity isosurface colored by the pressure and the cross section by the velocity magnitude

Figure 6.5: Computational domain and velocity profile view.

stress-free boundary condition. For the resolution of our unsteady Navier-Stokes problem we chose the previously detailed *PCD* preconditioner coupled with the *GCR* global solver with a relative tolerance of 10^{-8}. For the $\mathbf{A}_p$, M_p sub-problems we used the *Multigrid* method with a relative tolerance of 10^{-6}. As for the $F_{\mathbf{u}}$ sub problem we used the *Fieldsplit* method, *bJacobi* coupled the *Multigrid* method on the spacial components of $F_{\mathbf{u}}$ with a relative tolerance of 10^{-6}, except for $Re_t = 3500$ where the *Multigrid* was replaced by the *GASM* domain decomposition method with *LU* on the sub-domains.

Geometry	h_{min}	h_{max}	h_{avrg}	Nodes	Tetrahedrons
M0-P2P1G2	$1.7 \cdot 10^{-4}$	$3.18 \cdot 10^{-3}$	$8.9 \cdot 10^{-4}$	787,204	496,129
M0-P2P1G1	$1.9 \cdot 10^{-4}$	$2.9 \cdot 10^{-3}$	$1.3 \cdot 10^{-3}$	108,051	412,575
M0-P3P2G1	$1.9 \cdot 10^{-4}$	$2.9 \cdot 10^{-3}$	$1.3 \cdot 10^{-3}$	108,051	412,575
M1	$1.6 \cdot 10^{-4}$	$1.8 \cdot 10^{-3}$	$7.6 \cdot 10^{-4}$	170,000	830,000
M2	$1.4 \cdot 10^{-4}$	$1.96 \cdot 10^{-3}$	$6 \cdot 10^{-4}$	630,000	3,400,000
M3	$8.5 \cdot 10^{-5}$	$1.7 \cdot 10^{-3}$	$3.5 \cdot 10^{-4}$	1,300,000	7,000,000
M2000	$2.04 \cdot 10^{-4}$	$5.4 \cdot 10^{-3}$	$5.4 \cdot 10^{-4}$	460,000	2,500,000
M2000-bis	$6.3 \cdot 10^{-5}$	$2.07 \cdot 10^{-3}$	$5.8 \cdot 10^{-4}$	557,536	2,879,365
M3500	$1.45 \cdot 10^{-4}$	$2.6 \cdot 10^{-3}$	$4.1 \cdot 10^{-4}$	560,000	3,200,000

Table 6.8: Characteristic lengths of the different meshes.

2.4 Results

In the sequel we show respectively the results for $Re_t = 500$, $Re_t = 2000$ and $Re_t = 3500$. The velocity profiles were obtained using Paraview. Physical surfaces, corresponding to each of the internal radial sections, were included inside the mesh so that the evaluation of the velocity profiles on the radial sections is more accurate.

GEOMETRY	TimeStep	T_f	DOF	CPU	TIME	MACHINE
M0-P2P1G2	$1 \cdot 10^{-3}$	3s	$2.25 \cdot 10^6$	96	27:31:44	Atlas
M0-P2P1G1	$1 \cdot 10^{-3}$	3s	$1.89 \cdot 10^6$	256	11:46:36	Curie
M0-P3P2G1	$1 \cdot 10^{-3}$	3s	$6.54 \cdot 10^6$	256	2-19:26:58	Curie
M1	$1 \cdot 10^{-3}$	3s	$3.93 \cdot 10^6$	32	2-10:03:36	Atlas
M2	$1 \cdot 10^{-3}$	3s	$1.54 \cdot 10^7$	128	3-00:43:38	Curie
M3	$1 \cdot 10^{-3}$	3s	$2.88 \cdot 10^7$	1,024	1-14:43:52	Curie
M2000	$1 \cdot 10^{-4}$	0.45s	$1.23 \cdot 10^7$	72	7-20:52:48	Atlas
M2000-bis	$1 \cdot 10^{-4}$	0.45s	$1.24 \cdot 10^7$	96	7-09:05:00	Atlas
M3500	$1 \cdot 10^{-4}$	0.4s	$1.39 \cdot 10^7$	1,024	6-09:36:00	Curie

Table 6.9: Simulations characteristics. Δt being the time step, T_f being the right bound of the time interval I, DOF being the number of degrees of freedom, CPU being the number of processors, TIME being the simulation run time and MACHINE being the cluster where we run the simulations.

Remark 22. *Since Paraview does a $\mathbb{P}_1$ visualisation of a $\mathbb{P}_2$ field and does not support a high order geometry approximation, and in order to improve the accuracy of the exported solution, we run the final time step of each of the performed simulations on the Lagrange $\mathbb{P}_1$ interpolation of the initial mesh.*

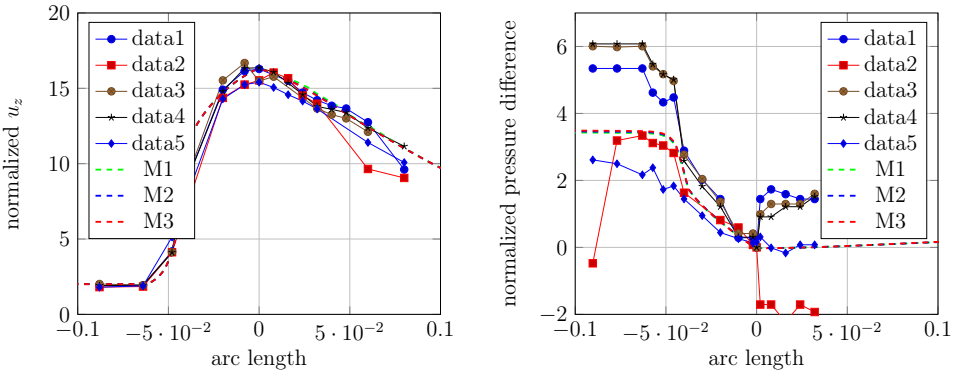
A script was also implemented to evaluate the E_z metric.

Remark 23. *In the following, remarkable differences between the experimental data may be observed. It is due to the fact that the accuracy of the PIV technique depends on the quality of the images, the spatial and temporal image resolutions, and the number of images used, as well as the particle size, particle seeding density, particle displacement, and the ability of the particles to follow the flow. Differences in the PIV algorithms can also significantly influence the accuracy of the predicted flow quantities. Furthermore, the accuracy and reproducibility of PIV experiments are affected not only by the algorithms used, but also by experimental factors such as fluctuations in the flow conditions and fluid properties, as well as uncertainties associated with measurements of fluid properties, all of which can vary from one laboratory to another. Controlling these variables to accurately generate the desired flow field is absolutely critical to obtaining accurate estimates of the velocity and stress fields [71].*

Results for $Re_t = 500$

The simulation at $Re_t = 500$ was carried out until $t = 3s$, time reasonably close to the steady state, and we set the time step to $\Delta t = 10^{-3}$. Figure 6.6 shows the results at $Re_t = 500$ for the normalized axial velocity along the z axis and the normalized pressure difference along the same axis for the M1, M2 and M3 levels of refinements. The magnitude of the velocity is in a perfect agreement with the experimental data (left panel). A very satisfactory agreement is also obtained for the pressure difference (right panel) . Similar conclusions can be made when comparing at the computated $\mathbb{P}_2$ velocity

profiles at sections $z = -0.064, -0.008, 0.016$ and $z = 0.06$ compared with the five sets of experimental data at the same sections, for the same mesh refinement levels, shown in figure 6.7.



(a) normalized axial velocity along z (b) normalized pressure difference along z

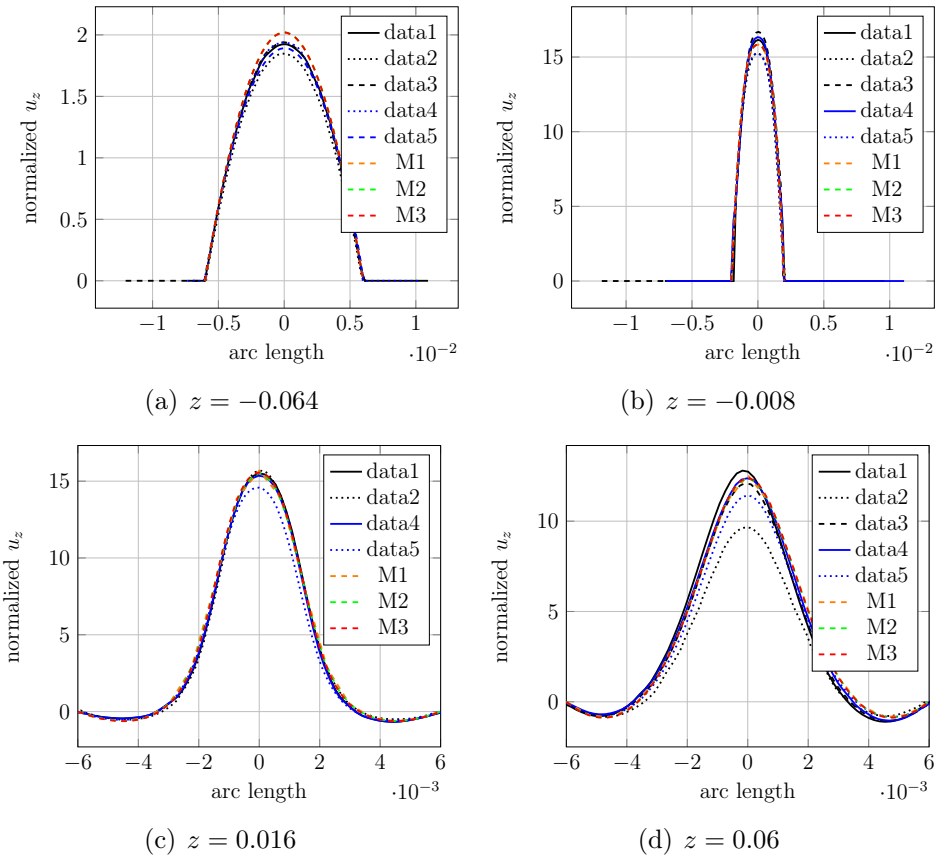
Figure 6.6: (a) Comparison between experimental data and numerical results for the normalized axial velocity along z (Eq 6.5) and (b) the normalized pressure difference along z (Eq 6.4), for $Re_t = 500$.

We next focus on the comparison between CFD profiles for $\mathbb{P}_2\mathbb{P}_1$ and $\mathbb{P}_3\mathbb{P}_2$ finite elements configurations. To our best knowledge, there are no high order finite elements results for the FDA benchmark in the literature, neither a high order approximation of the geometry. Figure 6.8 shows the results of the CFD profiles for $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$, $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_1$, and $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_2$ elements over the M0 mesh. We also compare those results to the one corresponding to the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ on the finest mesh M3. The results for the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ and $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_1$ configurations over the M0 mesh are quite similar to those for the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_1$ configuration over the mesh M3. However, although the results of the $\mathbb{P}_2\mathbb{P}_1\mathbb{G}_2$ elements over the M0 mesh capture well the profiles shape, they under estimate the normalized pressure difference along the centerline with respect to the finest mesh at the inlet of the flow. The results for the $\mathbb{P}_3\mathbb{P}_2\mathbb{G}_2$ configuration over the M0 mesh are on going. We expect them to be the closest to the results on the finest mesh M3.

Results for $Re_t = 2000$

The simulation at $Re_t = 2000$ was carried out until $t = 0.45s$, time when the regime was fully developed, and we set the time step to $\Delta t = 10^{-4}$. It is worth stressing that farther the downstream of the sudden expansion, the experimental velocity profiles are significantly different from one another. The jet breakdown point captured by laboratories vary remarkably.

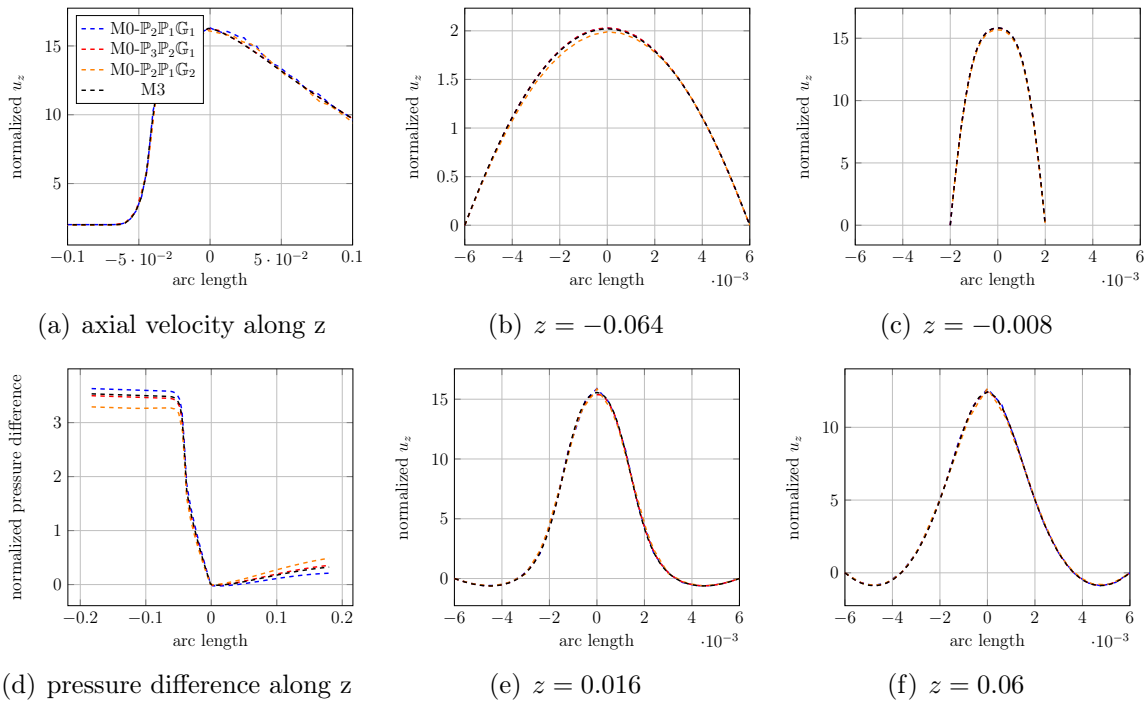
Numerically the results were very sensitive to the mesh refinement and to the time step. Figure 6.9 shows that the simulated pressure difference matches very well with the experimental data, while a numerical jet breakdown point was captured much farther downstream than the experimental observed breakdown point. A possible explanation of this mismatch may be the accuracy of the numerical integration. It is worth testing



(a) $z = -0.064$ (b) $z = -0.008$

(c) $z = 0.016$ (d) $z = 0.06$

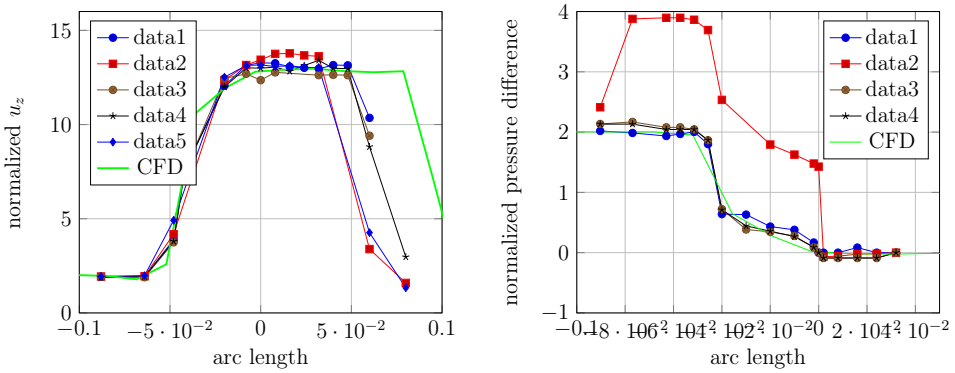
Figure 6.7: Comparison between experimental data profiles of the normalized axial velocity (Eq 6.5), for $Re_t = 500$.



(a) axial velocity along z (b) $z = -0.064$ (c) $z = -0.008$

(d) pressure difference along z (e) $z = 0.016$ (f) $z = 0.06$

Figure 6.8: Comparison between CFD profiles for $\mathbb{P}_2\mathbb{P}_1$ elements on the M1 mesh and $\mathbb{P}_2\mathbb{P}_1$ and $\mathbb{P}_3\mathbb{P}_2$ on the coarser mesh M0.



(a) normalized axial velocity along z (b) normalized pressure difference along z

Figure 6.9: (a) Comparison between experimental data and numerical results for the normalized axial velocity along z (Eq 6.5) and (b) the normalized pressure difference along z (Eq 6.4) for $Re_t = 2000$.

increasing the quadrature formulas order. This led us to do more investigations. The meshing strategy was readapted so that a special refinement was prescribed at the exit of the throat to better handle the recirculations, see Figure 6.10. Besides, we tested a symmetric formulation of the stress tensor. With that choice, the preconditioner, specially when solving the $\mathbf{F_u}$ subproblem, with the component block split strategy, must give more accurate results. Doing this operation we expect, for a symmetric formulation of the deformation tensor, that as the Reynolds number increases, that the convergence will improve. In fact, the extra diagonal blocks that we have left will tend to be null when the viscosity decreases. The new results after those adjustments are shown in figure 6.11.

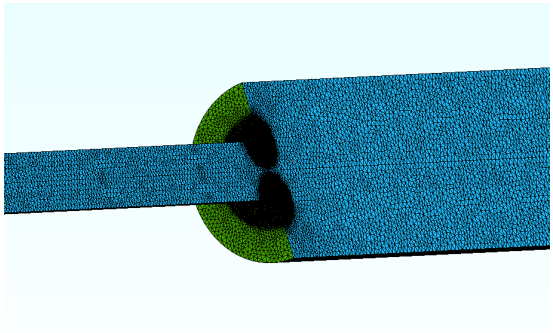
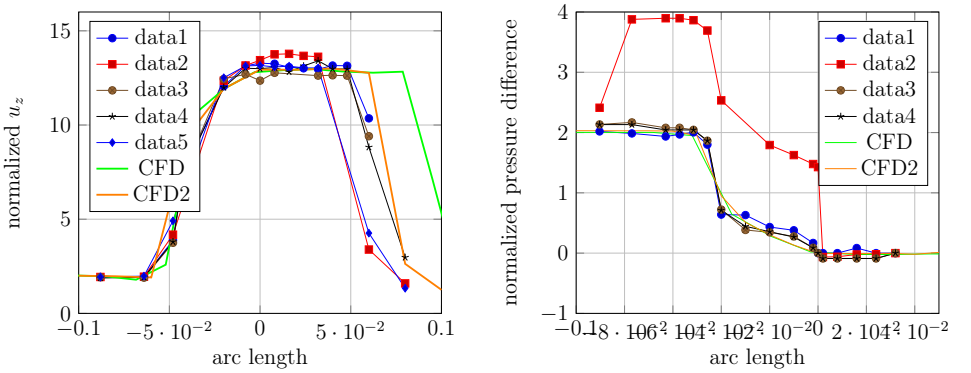


Figure 6.10: A torus of refinement around the throat exit.

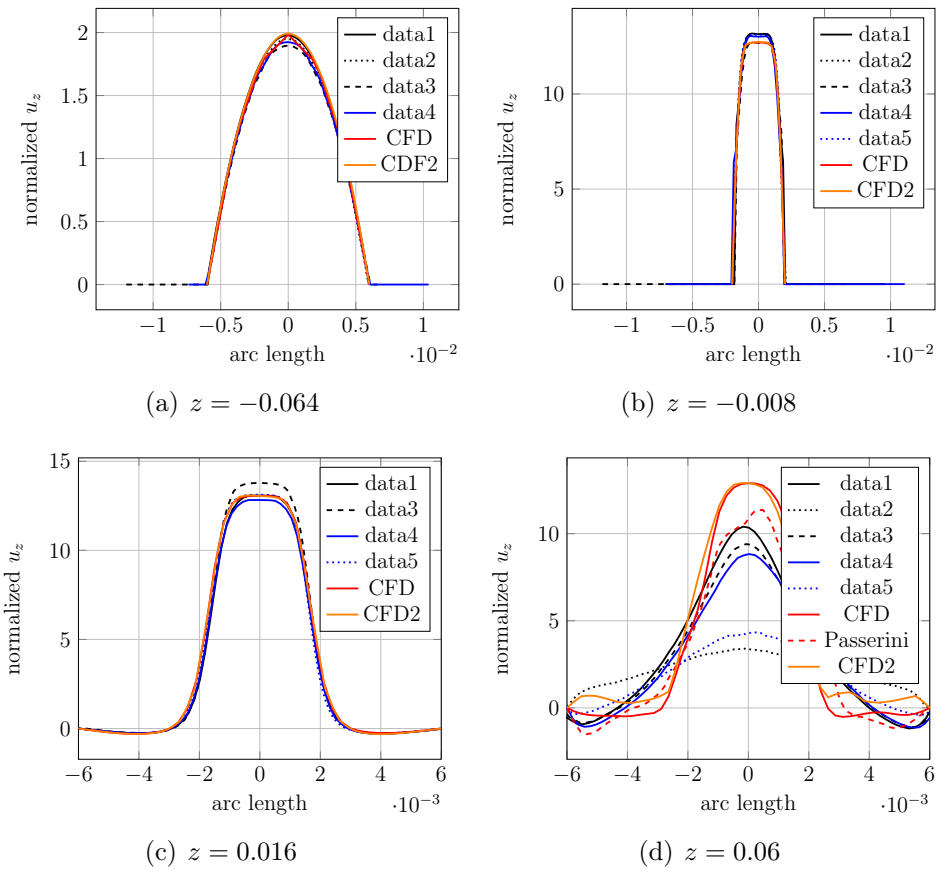


(a) normalized axial velocity along z (b) normalized pressure difference along z

Figure 6.11: (a) Comparison between experimental data and numerical results for the normalized axial velocity along z (Eq 6.5) and (b) the normalized pressure difference along z (Eq 6.4) for $Re_t = 2000$.

We can clearly see the difference between the old strategy (CFD) and the new strategy (CDF2) in Figure 6.11, where the jet breakdown point of the new strategy fit with the experimental jet breakdown point.

In figure 6.12, a very good agreement between the numerical data and the experimental data was observed at $z = -0.064, -0.008$ and $z = 0.016$, while a small mismatch was noticed at $z = 0.06$, section close to the jet breakdown point. The retrieved numerical profile over this section is the same as the one reported in [105] at this same section.



(a) $z = -0.064$ (b) $z = -0.008$

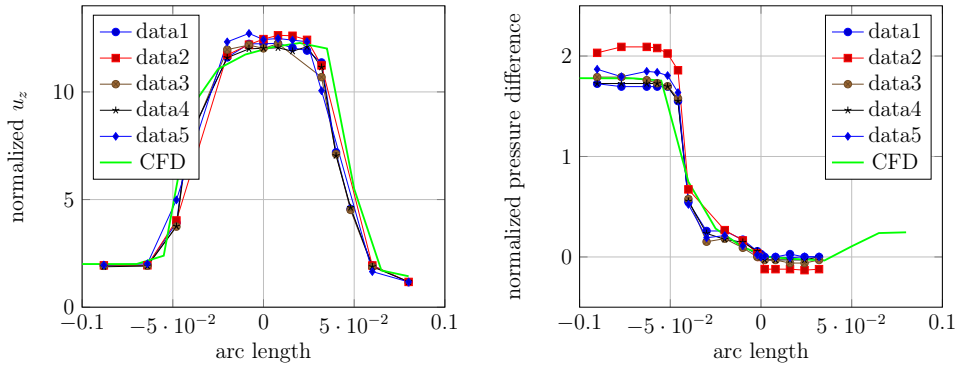
(c) $z = 0.016$ (d) $z = 0.06$

Figure 6.12: Experimental data profiles vs CFD data profiles of the normalized axial velocity (Eq 6.5), for $Re_t = 2000$.

Figure 2.3(b) shows the axial velocity isosurface coloured by the pressure and the cross section coloured by the velocity magnitude at $t = 0.45s$ for $Re_t = 500$.

Results for $Re_t = 3500$

The simulation at $Re_t = 3500$ was carried out until $t = 0.4s$, and the time step was set to $\Delta t = 10^{-4}$. The normalized velocity profile along the centerline and the pressure difference shown in Figure 6.13 show a good matching with the experimental data whereas the CFD results presented in [133] failed all to catch the jet breakdown point; the DNS results over predicted the jet length while the turbulence models predicted a shorter jet. The results of the normalized axial velocity profiles presented in Figure 6.14 are in a good agreement with the experimental data. They mildly underestimate the experimental velocity at $z = -0.008$, and pass the jet breakpoint, at $z = 0.06$ where the velocity is much reduced, the profile and the magnitude of the velocity were still respected.

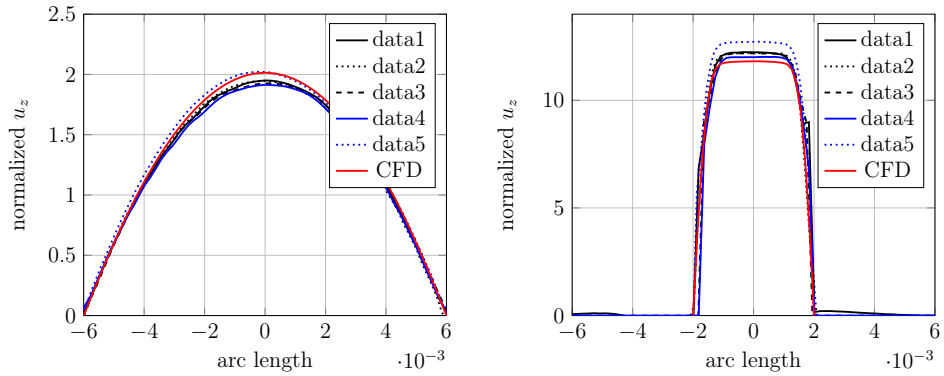


(a) normalized axial velocity along z (b) normalized axial pressure along z

Figure 6.13: (a) Experimental data vs numerical results for the normalized axial velocity along z (Eq 6.5) and (b) the normalized pressure difference along z (Eq 6.4) for $Re_t = 3500$.

Figure 6.15 shows the velocity profiles at the last time step for the different flow regimes coloured with the velocity magnitude. A laminar behaviour is clearly distinguished at $Re_t = 500$ while we can observe the fluctuations at $Re_t = 2000$ and $Re_t = 3500$, and a pulsatile jet behaviour at the expansion for $Re_t = 3500$.

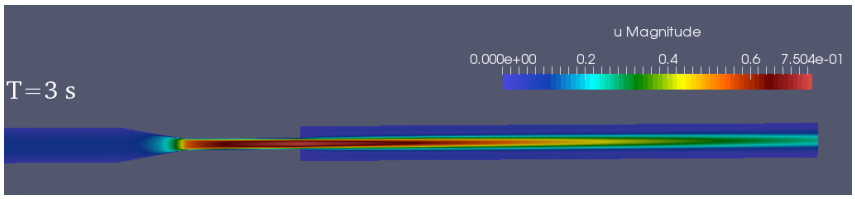
Remark 24. *Recently in [17] turbulent flow simulations for $Re_t = 3500$ and $Re_t = 5000$ were performed using the Leray turbulent model — a Large Eddy Simulation (LES) technique — to average in space the Navier-Stokes equations and handle the strong convective fields compared with viscous forces that may trigger flow disturbances up to turbulence. The same model was also used in [82] for $Re_t = 6500$. In [133] a comparison between the shear stress transport (SST) models the $k - \omega$ (KO) models and the $k - \epsilon$ models were done for $Re_t = 500, 2000, 3500, 5000$ and $Re_t = 6500$.*



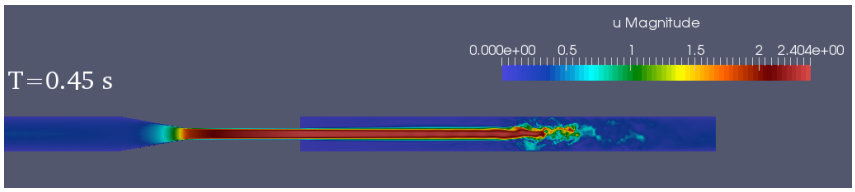
(a) $z = -0.064$ (b) $z = -0.008$

(c) $z = 0.016$ (d) $z = 0.06$

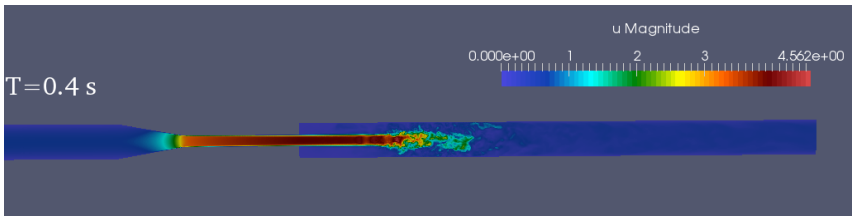
Figure 6.14: Comparison between CFD data and experimental data profiles of the normalized axial velocity (Eq 6.5), for $Re_t = 3500$.



(a) Re500



(b) Re2000



(c) Re3500

Figure 6.15: Velocity profiles at the last time step.

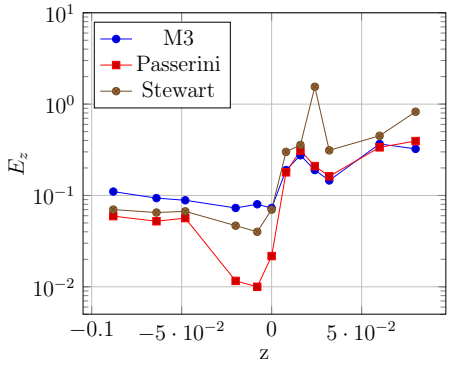
Results for E_z and E_Q metrics

The results of the E_z and E_Q metrics of all the simulations are presented in Figures 6.16 and 6.17, respectively for $Re_t = 500, 2000$ and $Re_t = 3500$, and in Figure 6.18 for the different mesh refinements and high order finite element approximation at $Re_t = 500$. In Figure 6.16 we show a comparison with the values of the E_z metrics reported in [105] and in [133] (simulation for the laminar model). We can notice the same behaviour at $Re_t = 500$ and $Re_t = 2000$ with respect to the *Passerini et al.* results, we are close to the error they reported, except in the throat where we suppose we have a coarser mesh with respect to theirs. The same error profile is however retrieved for $Re_t = 3500$. In figure 6.18(a) we plot the E_z metric for the different mesh refinements at $Re = 500$. The convergent behaviour is not clear since the refinement strategy isn't homogeneous in all parts of the geometry.

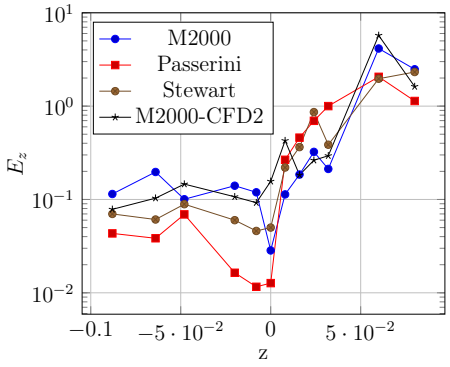
Remark 25. *It is worth pointing out on the fact that, since the E_z metric is a sum of normalized absolute values, a large value of E_z doesn't necessarily implies a disagreement between the experimental and computational data. In particular, small values of $\mathbf{u}_{e,i}$, specially close to the wall, increase the contribution of the error.*

As for the conservation of mass metric, Figure 6.17 show a comparison with the values of the E_z metrics reported in [105] and in [133] for the laminar model. (Note that the computational data for [133] are only available for $Re = 3500$), while Figure 6.18(b) show the E_q metric for the different mesh refinements at $Re = 500$. We can see that the error doesn't exceed the $\sim 2\%$ except for the coarser mesh M0 where, for two sections, the error

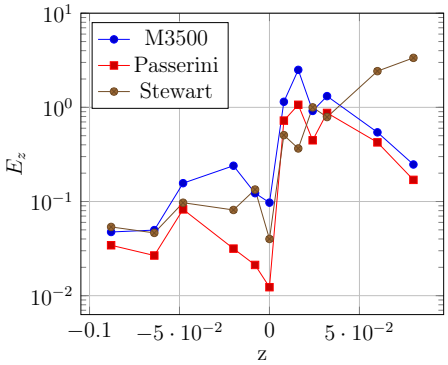
increases up to $\sim 10\%$. The mesh is clearly not enough fine in this case.



(a) Validation metric E_z, $Re_t = 500$



(b) Validation metric E_z, $Re_t = 2000$

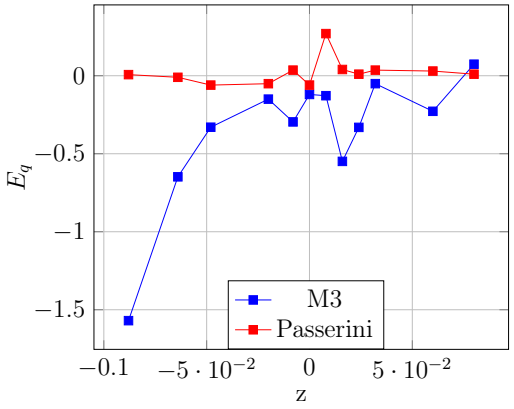


(c) Validation metric E_z, $Re_t = 3500$

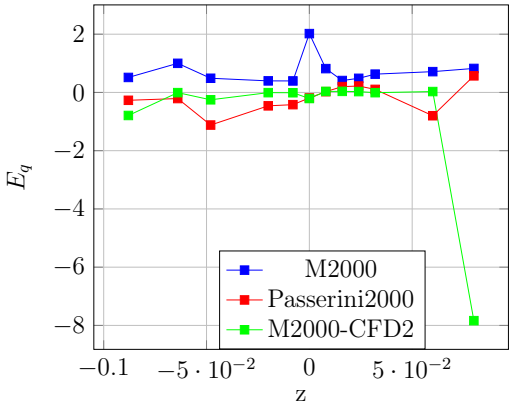
Figure 6.16: (a) Comparison of the Ez metrics (Eq (6.7)) with respect to Passerini et al. and Stewart et al. data in semi-logarithmic scale for $Re = 500$, (b,c) Comparison of the Ez metrics with respect to Passerini et al. and Stewart et al. data in semi-logarithmic scale for $Re = 2000$ and $Re = 3500$, respectively.

In Table 6.10, we report the number of iteration needed until achieving the desired accuracy for the global linearized Navier-Stokes problem, for the $\mathbf{F_u}$, $\mathbf{A}_p$ and $\mathbf{Q}_p$ sub-problems. The number of iterations for the global problem, for the Laplacian problem and the mass matrix problem is quasi constant with respect to the mesh refinement and the increasing of the Reynolds number, while the number of iteration for the $\mathbf{F_u}$ problem varies with the simulations. We recall that for $Re_t = 2000$ and $Re_t = 3500$ we used GASM method with a LU solver in the subdomains for the convection-diffusion problem $\mathbf{F_u}$.

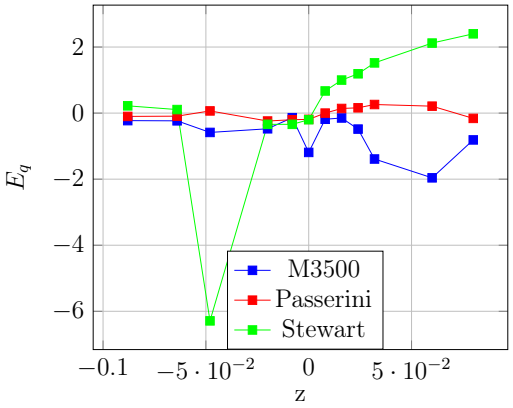
Surprisingly, the simulation at $Re_t = 2000$ was the most difficult, not the one at $Re_t = 3500$. Retrieving a jet point comparable to the experimental data was not an easy task. We had to try several meshing strategies and different simulation final steps to detect when the regime was fully developed.



(a) Validation metric E_q, $Re_t = 500$

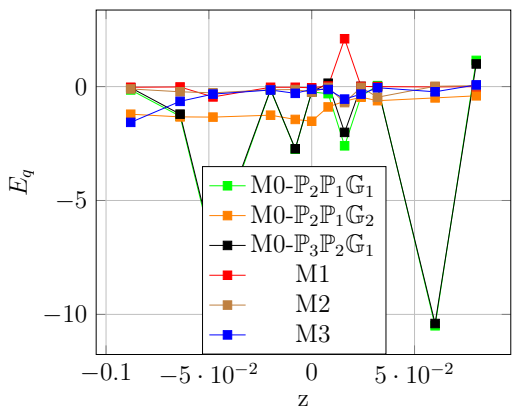


(b) Validation metric E_q, $Re_t = 2000$

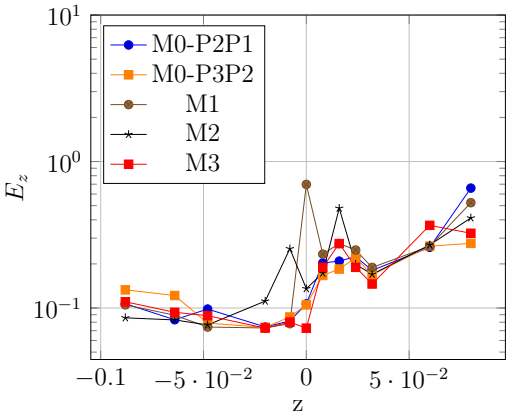


(c) Validation metric E_q, $Re_t = 3500$

Figure 6.17: Comparison of mass error metric E_Q (Eq 6.6) as a function of the position along the z axis with respect to Passerini et al. CFD data for (a) $Re_t = 500$, (b) $Re_t = 2000$ and (c) $Re_t = 3500$.



(a) Validation metric E_q



(b) Validation metric E_z

Figure 6.18: Validation metrics E_z (Eq (6.7)) and E_q ((6.6)) for $Re = 500$ for the different mesh refinements

Re	NS	Fu	Ap	Qp
M0-$\mathbb{P}_2\mathbb{P}_1$	2	50	28	4
M0-$\mathbb{P}_3\mathbb{P}_2$	2	53	75	12
M1	2	121	10	3
M3	2	7	82	4
M2000	10	37	88	3
M3500	7	115	89	3

Table 6.10: PCD: Global number of iteration/sub-problem, $rtol = 10^{-6}$ for the subproblems and $rtol = 10^{-8}$ for Navier-Stokes iteration.

Conclusion

Through this chapter, we have, in a first part, shown the scalability of the in-house implemented PCD preconditioner in terms of number of GCR iterations to achieve a certain tolerance. We have also validated the independency of the PCD preconditioner with respect to the mesh size, the Reynolds number. Moreover we have extended this independency to high order finite elements approximation, study that was never carried out before from a numerical point of view.
In the second part of this chapter, we used the so-validated in-house PCD preconditioner to perform the FDA benchmark. The importance of this benchmark rely on the fact that i) it covers different flow regimes, ii) the used device features phenomena that could be encountered in real medical devices, iii) experimental data from 5 difference laboratories are available online. We have thus validated and verified our numerical choices through a perfect agreement between our CFD output data and the experimental data for all the flow regimes. High order finite elements and high order geometry approximation were firstly used in our work for this benchmark, none of the numerical teams that have taken the FDA challenge have done this investigation before. We also compared our numerical results to the results of Passerini et al. [105] and Stewart et al. [133].

Chapter 7

A pipeline from medical images to simulated MRI

Angiographic imaging is a crucial domain of medical imaging. In particular, Magnetic Resonance Angiography (MRA) is used for both clinical and research purposes. This chapter presents the first framework geared toward the design of virtual MRA images from real MRA images. It relies on a pipeline that involves image processing, vascular modeling, computational fluid dynamics and MR image simulation, with several purposes. It aims to provide to the whole scientific community (1) software tools for MRA analysis and blood flow simulation; and (2) data (computational meshes, virtual MRAs with associated ground truth), in an open-source / open-data paradigm. Beyond these purposes, it constitutes a versatile tool for progressing in the understanding of vascular networks, especially in the brain, and the associated imaging technologies. The results shown in this chapter, are a joint work with PhD students from the VIVABRAIN ANR project, and were obtained during the CEMRACS 2015.

Contents

1 Introduction

In the context of the CEMRACS 2015 (Centre d'Été Mathématique de Recherche Avancée en Calcul Scientifique), devoted in its twentieth edition to the simulation of (coupled) multi-physics models involving fluids, I have participated as a member of the project "Phantom" — *A pipeline from medical images to simulated MRI images of phantoms*. The CEMRACS is a scientific event of the SMAI (the French Society of Applied and Industrial Mathematics), whose goal is to bring together scientists from both the academic and industrial communities and discuss over a specific topic. It is located in the CIRM (Centre International de Recherche Mathématique) in Marseille and is organised into a 6 weeks event, where, during the first week, a classical summer school is proposed. It consists of several lectures given by leading scientists and related to the topics of the research projects. The remaining 5 weeks are dedicated to working on the research projects, possibly after a morning seminar.

The purpose of the "Phantom" project is to develop a multidisciplinary pipeline for the generation of virtual (i.e., simulated) angiographic images (more precisely, Magnetic Resonance Angiographies, MRA) of the human brain, associated to their anatomical (3D) and hemodynamic (3D+t) models (providing ground-truths). These angiographic images have progressively proved their usefulness in various clinical applications, in particular for cerebrovascular issues (e.g., neurosurgery planning; stenoses, aneurysm or thrombosis quantification; arteriovenous malformation detection and follow-up, etc.), thanks to the progress in 3D medical imaging (such as Magnetic Resonance Imaging, MRI, and X-ray Computed Tomography, CT) that has led to the development of modalities devoted to visualise vascular structures. However, these cerebral angiographic data are generally complex to process and analyse due to their size and low amount of relevant (vascular) information versus noise, artifacts and other anatomical structures. This has motivated, since the mid 90's, the proposal of several image processing tools for vessel filtering, segmentation and quantification. Unfortunately, contrary to morphological brain image analysis, for which synthetic (i.e., virtual) images and associated ground-truths (segmented data) are now widely available (e.g., BrainWeb), there is no such data in the case of cerebrovascular images.

Simulated MRA and ground-truths are currently not available for complex vascular networks (and in particular cerebral ones). This results in a lack of common development, validation and comparison framework in the research fields related to vessel analysis. Making these data and ground-truths fully available for the whole medical image analysis community constitutes the very goal of the ANR project VIVABRAIN.

2 Scientific program of VIVABRAIN ANR project

The interdisciplinary program starts from real MR angiographic data to finally lead to the generation of virtual MR angiographic data. During this process, which shall lead to these simulated data, realistic 3D (anatomical) and 3D+t (hemodynamic) models –providing ground-truths for the virtual MRA images– are obtained. In order to do so, these successive steps are considered:

- Task 1: Data acquisition: it consists of planning the acquisition slots on the MRI

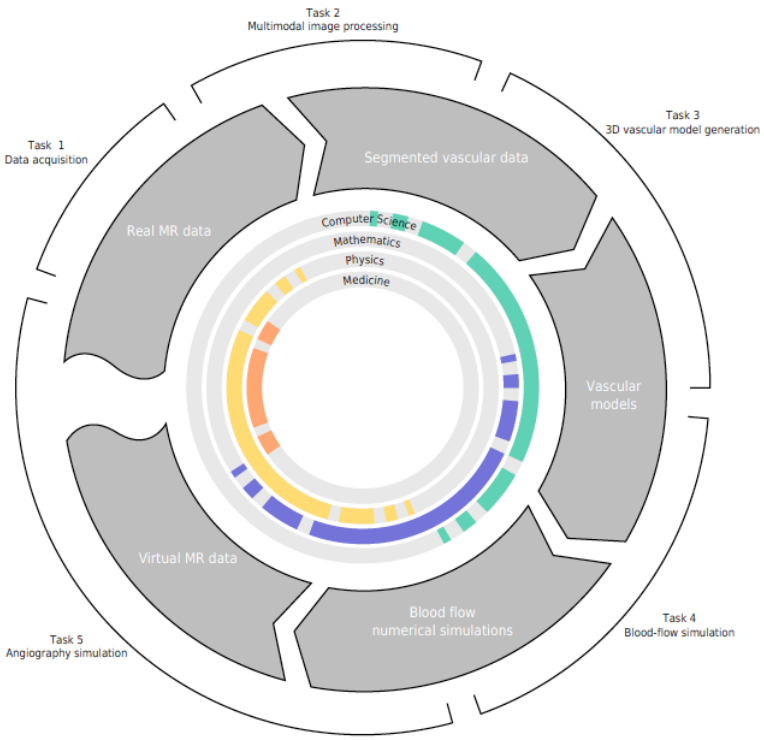
platform, acquiring and collecting MR images under the medical control of an hospital practitioner of the HUS, *Hôpital Universitaire de Strasbourg*.

- Task 2: Extraction of vascular volumes from real MRA images: this step mainly deals with image processing and analysis. This requires the development of methodological and applicative techniques, in the fields of filtering, segmentation and interactive correction, in particular in the context of mathematical morphology and discrete topology.

- Task 3: Generation of 3D vascular models from these data. This requires the development of methodological and applicative techniques in the fields of sparse image registration and knowledge fusion, in particular in the context of atlas generation and geometric/topological modelling.

- Task 4: 3D+t simulation of blood flow in complex (arterial and venous) models. First, this requires proper computational meshes which is currently a challenge. Then, it requires not only state of the art, but also novel numerical methods to process these problems that are large scale, coupled, highly non-linear, multi-physics and multi-scale models (in particular with respect to blood modelling and rheology). Finally, it requires validation steps allowing for calibration and uncertainty quantification.

- Task 5: Simulation of MR acquisition of angiographic data from these 3D+t models. This requires the modelling of physical phenomena related to the specific MR sequences devoted to visualise moving structures, in order to reproduce the signal and noise finally leading to the formation of MRA data, on the basis of the 3D and 3D+t models previously generated.



3 The CEMRACS project goals

Our target during the CEMRACS 2015 was to access the reliability and accuracy of the output data of tasks 3, 4 and 5. For that reason, we considered a physical phantom so that we have an exact description of its geometry. The validation process steps are the following:

- Access the reliability of the segmentation techniques: compare the realistic geometry, knowing exactly its shape, with MRI segmentations, obtained by a simple threshold and by the classical method of snake.
- Access the reliability of the CFD simulation: (i) compare the outputs of two identical simulations using Feel++ in the first and Freefem++ in the second, (ii) Compare the outputs of the Feel++ and Freefem++ simulations to the MRI measurements.
- Access the reliability of the MRI simulation: compare the simulated phase and magnitude images to (i) the realistic geometry and (ii) the MRI acquisitions.

In the context of CFD simulation, we also considered two different ways of imposing the boundary conditions at the inlet, the first by setting a constant academic Poiseuille flow, the second by imposing a pulsatile retrieved from velocity MRI measurements. Our study was also extended to the realistic geometry of the brain venous network. We compared the area and flow calculation at the different inlets and outlets sections.
Another accomplished task during the CEMRACS was to ensure the transmission of the data from one task to another. This was done by converting the format of the output of a task to the format of the input of the following task. The AngioTK software, developed in the context of the VIVABRAIN project was tested on the physical phantom and performed according to the required data formats.

ESAIM: PROCEEDINGS AND SURVEYS, Vol. ?, 2016, 1-10
Editors: Will be set by the publisher

PHANTOM PROJECT: DEVELOPMENT AND VALIDATION OF THE PIPELINE FROM MRA ACQUISITION TO MRA SIMULATIONS

Alexandre Ancel[1], Alexandre Fortin[2], Simon Garnotel[3], Olivia Miraucourt[2] and Ranine Tarabay[1]

Abstract. The aim of this project is to validate the *Vivabrain* pipeline with a physical phantom from real MRI acquisition to MRI simulations through image segmentation and computational fluid dynamics (CFD) simulations. For that purpose, we set up three comparison benchmarks. The first benchmark compares dimensions of the reconstructed geometry from real MRI acquisition to the physical phantom dimensions. The second aims to validate the CFD simulations by comparing the outputs of two simulations, one carried out Feel++ and the other using FreeFem++. The CFD outputs are also compared to MRI flow measurement data. The goal of the last comparison benchmark is to compare the MRI simulations outputs to the numerical fluid simulations.

Introduction

In the last 20 years, progress in medical imaging has led to the development of modalities devoted to visualizing vascular structures. These angiography images progressively proved their usefulness in various clinical applications, in particular for cerebrovascular issues. This project is within the context of the ANR project *Vivabrain* [2]. The goal of this project is to develop a pipeline for the generation of virtual Magnetic Resonance Angiography (MRA) of the human brain, associated to their anatomical (3D) and hemodynamic (3D+t) models (providing a ground-truth). The interdisciplinary program follows four steps, see Fig. 1. We first start from real MRA data from which we extract the vascular volumes. We then generate the 3D vascular mesh. We perform 3D+t simulations of blood flow in the complex (arterial and venous) mesh. Finally, we run the simulations of MR acquisition of angiography data from these 3D+t models.

AngioTK [1] is a software framework developed in this context, using open source softwares. This framework takes MRI data as input and produces virtual angiographies as output. AngioTK proceeds in several steps, see also Fig. 1:

(1) Filtering: Filter the initial MRI data, to ease the extraction of arterial/venous data;
(2) Segmentation: Separate the veins/arteries from the background;
(3) Mesh processing: Process the segmented data to prepare it for numerical simulations. This task involves several subtasks using centerlines notably;
(4) Numerical simulations: simulations of blood flow in the processed mesh (Feel++/FreeFem++);
(5) Particles tracing: Generates particles for virtual angiographies;
(6) Virtual angiography: Uses JEMRIS to simulate virtual angiographies.

[1] University of Strasbourg, IRMA / UMR 7501, Strasbourg, France
[2] University of Reims Champagne-Ardenne, LMR, Reims, France
[3] University of Picardie Jules Verne, BioFlowImage Laboratory, Amiens, France

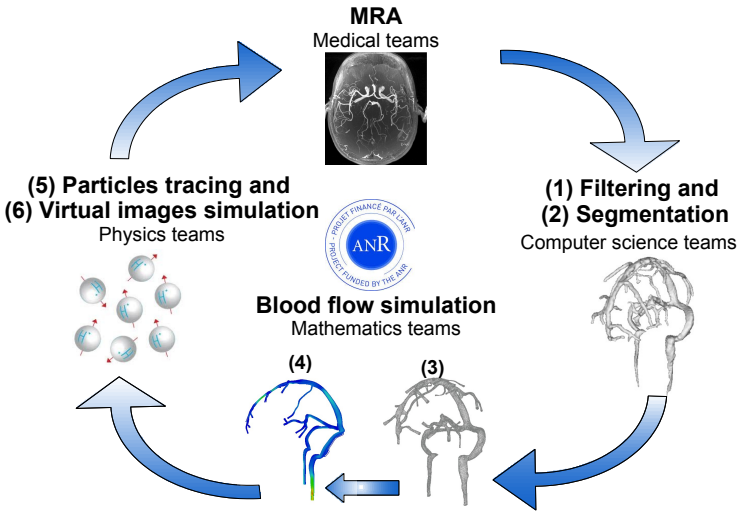


FIGURE 1. The *Vivabrain* project task loop.

In this paper, our purpose is to evaluate the accuracy of the image segmentation, the (3D+t) blood flow simulations and the MRI simulations. To this end, we started from a physical phantom, see Fig. 2, so that we have an exact description of its geometry that can be used to evaluate the error made when extracting the 3D volume from the MRI segmentation. For the computational fluid dynamic validation, we have chosen to run simulations using two finite elements libraries, FEEL++ [13] and FREEFEM++ [10]. A first level of validation of the computational fluid dynamics (CFD) simulations is to compare the numerical outputs obtained by using FEEL++ and FREEFEM++ libraries, running equivalent simulations on the same mesh. The second level of comparison is to compare the FEEL++ and FREEFEM++ numerical output data to the experimental MRI measurements. Lastly, the validation of the MRI simulations will be performed by comparing their outputs to the MRI acquisition and to the exact geometry dimensions.

The first part of this paper is dedicated to the description of the image segmentation, the numerical model and methods for the simulations of blood flow and MRI simulations. In the second part, we describe the benchmark setup used for the previously mentioned comparisons. The third part of this paper is dedicated to the image segmentation evaluation. Finally, we present the blood flow and MRI simulations results on the phantom. In particular, we run a cross-validation between the FEEL++ and FREEFEM++ libraries to estimate the precision of the step of blood flow simulations in the phantom and the complex cerebral venous network, which is studied in the *Vivabrain* project.

1. NUMERICAL METHODS

1.1. Image segmentation

In the first part of the pipeline, we need to extract the vascular volumes from the MRA images. Here, to segment the phantom, we use an active contour model called "snake" whose principle is to evolve a curve to detect the objects boundaries in a given image. Let Ω be a bounded open set of $\mathbb{R}^2$, with $\partial\Omega$ its boundary. Let $I : \bar{\Omega} \to \mathbb{R}^2$ be a given image of bounded variation and $\mathcal{C}(s) : [0,1] \to \mathbb{R}^2$ be a parameterized curve, at least of class C^2. The classical snake model consists of minimizing the energy functional $J_1(\mathcal{C})$ given by:

$$J_1(\mathcal{C}) = \alpha \int_0^1 |\mathcal{C}'(s)|^2 ds + \beta \int_0^1 |\mathcal{C}''(s)| ds - \lambda \int_0^1 |\nabla I(\mathcal{C}(s))|^2 ds \tag{1}$$

FIGURE 2. The physical phantom.

where α, β and λ are positive parameters. The first two terms control the smoothness of the contour (the internal energy), while the third term attracts the contour towards the object (the external energy). The snake minimizes the energy $J_1(\mathcal{C})$ by trying to locate the curve at the points of maxima $|\nabla I(\mathcal{C}(s))|$, acting as an edge-detector. In general, $-|\nabla I|$ can be replaced by $g(|\nabla I|)$ where g is a positive and decreasing function such as $\lim_{x\to\infty} = 0$ (e.g. $g(x) = \frac{1}{1+|x|}$ or $g(x) = \exp(-x)$). ITK-SNAP [1] implements the 3D geodesic active contour method developed by Caselles et al. [7], which allows topological changes of the curve, contrary to the classical snake. The new energy $J_2(\mathcal{C})$ to be minimized is given by:

$$J_2(\mathcal{C}) = 2\int_0^1 |\mathcal{C}'(s)| \cdot g(|\nabla I(\mathcal{C}(s))|)ds \tag{2}$$

This is a problem of geodesic computation in a Riemannian space, according to a metric induced by the image I. This minimization problem is solved by a gradient descent which follows an evolution equation and the curve changes over time as it can be seen in Fig. 3

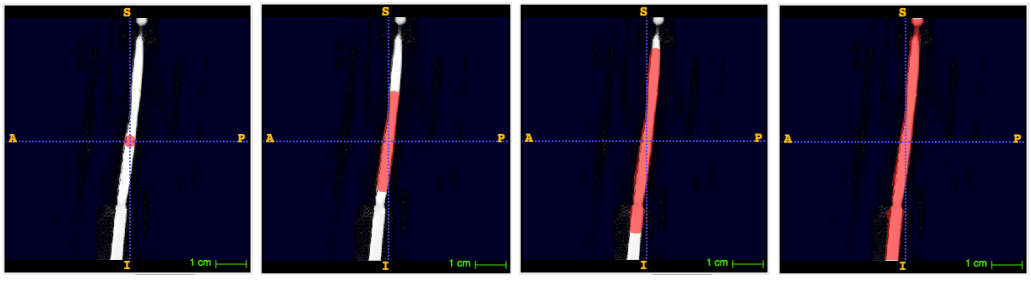


FIGURE 3. Evolution of the snake on the phantom at iterations $n = 0$, $n = 500$, $n = 1000$, $n = 1500$.

1.2. **Fluid simulations**

In this section, we introduce the equations of the fluid dynamics and the numerical methods used to compute the blood flow in the extracted volumes. Let $\Omega \subset \mathbb{R}^3$ denote the bounded connected domain under investigation and let $\partial\Omega = \Gamma_D \bigcup \Gamma_N$ be its boundary, where Γ_D and Γ_N are the sections of the boundary where we will respectively impose a Dirichlet condition and a Neumann condition. In the context of blood flow, it is assumed

[1] http://www.itksnap.org

that the unsteady incompressible Navier-Stokes equations Eq. (3,4) hold. They read as:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}.\nabla)\mathbf{u} + \nabla p - \mu\Delta\mathbf{u} = \mathbf{f} \quad \text{in } \Omega \tag{3}$$

$$\text{div}(\mathbf{u}) = 0 \quad \text{in } \Omega \tag{4}$$

where $\mathbf{u}$ and p are respectively the velocity and pressure of the fluid, ρ and μ are the density and the dynamic viscosity of the fluid, and $\mathbf{f}$ is an external force taken here equal to zero (gravity is neglected).

Let us consider a Dirichlet boundary condition at the inflow and a Neumann boundary condition at the outflow:

$$\mathbf{u} = \mathbf{u}_{in} \qquad \text{on } \Gamma_D \tag{5}$$

$$\boldsymbol{\sigma}(\mathbf{u}, p)\mathbf{n} = \mathbf{0} \quad \text{on } \Gamma_N \tag{6}$$

with the stress tensor that writes for a Newtonian fluid:

$$\boldsymbol{\sigma}(\mathbf{u}, p) = \mu\nabla\mathbf{u} - p\mathbf{I} \tag{7}$$

1.3. Numerical algorithms

Space discretization Let δ be a discretization parameter. We define $\hat{K} \subset \mathbb{R}^d$ $(d = 1, 2, 3)$ a reference elementary convex, *e.g.* a simplex or a hypercube. We denote by $\mathcal{T}_\delta$ a finite collection of nonempty, disjoint open simplices or hypercubes $\mathcal{T}_\delta = \{K = \boldsymbol{\varphi}^{\text{geo}}_{K,k}(\hat{K})\}$ forming a partition of Ω, where $\boldsymbol{\varphi}^{\text{geo}}_{K,k}$ is the polynomial of degree k that maps $\hat{K}$ to K, which is also called the geometric transformation. We denote by $\mathbb{P}^N(\hat{K})$ and $\mathbb{P}^N(K)$ the spaces of polynomials of total degree less or equal than N defined on $\hat{K}$ and K respectively. We denote Ω_δ the discrete domain, and $\Gamma_{D,\delta}$ and $\Gamma_{N,\delta}$ the discrete sections of the boundary where we set a Dirichlet and a Neumann boundary condition, respectively. Following these notations we can define $P^N_c(\Omega_\delta)$ and $[P^N_c(\Omega_\delta)]^d$:

$$P^N_c(\Omega_\delta) = \{v \in C^0(\Omega_\delta) \mid \ v \circ \boldsymbol{\varphi}^{\text{geo}}_{K,k} \in \mathbb{P}^N(\hat{K}) \ \forall K \in \mathcal{T}_\delta\}, \quad [P^N_c(\Omega_\delta)]^d = \prod_1^d P^N_c(\Omega_\delta). \tag{8}$$

In the following, we introduce the discrete spaces associated to the velocity and the pressure respectively:

$$\begin{aligned} H^1_{(\mathbf{u_{in}},\Gamma_{D,\delta})}(\Omega_\delta) &= \{f \in H^1(\Omega_\delta)/f|_{\Gamma_{D,\delta}} = \mathbf{u_{in}}\} \\ V_\delta &= \{v \in H^1_{(\mathbf{u_{in}},\Gamma_{D,\delta})}(\Omega_\delta) \cap [P^M_c(\Omega_\delta)]^d\} \\ V_{\delta,0} &= \{v \in H^1_{(\mathbf{0},\Gamma_{D,\delta})}(\Omega_\delta) \cap [P^M_c(\Omega_\delta)]^d\} \\ Q_\delta &= \{v \in P^N_c(\Omega_\delta)\} \end{aligned}$$

In order to ensure the existence, the uniqueness and the stability of the solution of the discrete problem, the spaces V_δ and Q_δ must satisfy the so called Inf-Sup condition:

$$\exists \beta_\delta > 0 \mid \quad \inf_{q_\delta \in Q_\delta} \sup_{\mathbf{v}_\delta \in V_\delta} \frac{\int_{\Omega_\delta} q_\delta \text{div}(\mathbf{v}_\delta)}{\|q_\delta\|_{0,\Omega_\delta}\|\mathbf{v}_\delta\|_{1,\Omega_\delta}} \geq \beta_\delta \tag{9}$$

We will consider the generalized Taylor-Hood finite element for the velocity-pressure discretization, that is to say we look for the velocity in $[P^{N+1}_c(\Omega_h)]^d, N \geq 1$ and the pressure in $P^N_c(\Omega_h)$ which satisfy the inf-sup condition. The resulting approximate velocity and pressure fields are respectively denoted by $\mathbf{u}_\delta$ and p_δ. Typically the Taylor-Hood finite element $(N = 1)$ is considered to solve this problem.

Time discretization Let T, be the timerun of the simulations and $I = [0, T]$, be the time interval. We subdivide I into $\mathcal{N}$ sub-intervals $I^n = (t^n, t^{n+1})$ with $n = 1, ..., \mathcal{N}$ and where $t^{n+1} - t^n = \Delta t$, the time step assumed constant over time.

1.3.1. FEEL++

We denote $(\mathbf{u}_\delta^n, p_\delta^n)$ the approximate discrete solution at time t^n. We first start by discretizing the time derivative of the velocity by choosing an implicit scheme, the so-called *backward differentiation formulation* (BDF) of order two. We then choose a semi-implicit scheme with a second order extrapolation in order to manage the non-linear term $\mathbf{u} \cdot \nabla \mathbf{u}$. The weak formulation then reads: find $(\mathbf{u}_\delta^{n+1}, p_\delta^{n+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H^1(\Omega_\delta)]^3 \cap [P_c^M(\Omega_\delta)]^3 \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{D,\delta}\}$, $\forall q_\delta \in Q_\delta$

$$\begin{aligned} \rho \int_{\Omega_\delta} \frac{\frac{3}{2}\mathbf{u}_\delta^{n+1} - 2\mathbf{u}_\delta^n - \frac{1}{2}\mathbf{u}_\delta^{n-1}}{\Delta t} \mathbf{v}_\delta + \rho \int_{\Omega_\delta} ((2\mathbf{u}_\delta^n - \mathbf{u}_\delta^{n-1}) \cdot \nabla \mathbf{u}_\delta^{n+1}) \cdot \mathbf{v}_\delta + \mu \int_{\Omega_\delta} \nabla \mathbf{u}_\delta^{n+1} : \nabla \mathbf{v}_\delta & \\ - \int_{\Omega_\delta} p_\delta^{n+1} \operatorname{div}(\mathbf{v}_\delta) &= 0 \\ \int_{\Omega_\delta} q_\delta \operatorname{div}(\mathbf{u}_\delta^{n+1}) &= 0 \end{aligned}$$

The choice of this discretisation is motivated by the fact that BDF schemes with extrapolation yield to stable time discretizations of the Navier-Stokes equations at the continuous level. The notation $\mathbf{A} : \mathbf{B}$ corresponds to the inner product of two tensor fields, it can be explicitly written as:

$$\mathbf{A} : \mathbf{B} = Tr(\mathbf{A}^\mathbf{T}\mathbf{B})$$

1.3.2. FREEFEM++

In this section, we describe the numerical method used in FREEFEM++ to solve the Navier-Stokes equations: the method of characteristics [6].

In order to manage the non-linearity, we consider the term $\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}.\nabla)\mathbf{u}$ as a particular derivative, so for every particle we can write:

$$\begin{aligned} \frac{dX}{\Delta t}(x, s; t) &= \mathbf{u}(t, X(x, s; t)) & (10) \\ X(x, s; s) &= x & (11) \end{aligned}$$

where $X(x, s; t)$ is the particle position at time t, which was in x at time s.

That gives:

$$\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}.\nabla)\mathbf{u}\right)(t^n, x) \sim \frac{\mathbf{u}(t^{n+1}, x) - \mathbf{u}(t^n, X^n(x))}{\Delta t} \quad (12)$$

with $X^n(x) = x - \mathbf{u}(t^n, x)\Delta t + \mathcal{O}(\Delta t^2)$.

We finally have:

$$\begin{aligned} \frac{\rho}{\Delta t}(\mathbf{u}^{n+1} - \mathbf{u}^n \circ X^n) - \mu \Delta \mathbf{u}^{n+1} + \nabla p^{n+1} = 0 & \quad (13a) \\ \operatorname{div}(\mathbf{u}^{n+1}) = 0 & \quad (13b) \end{aligned}$$

The weak formulation can be written as: find $(\mathbf{u}_\delta^{n+1}, p_\delta^{n+1}) \in V_\delta \times Q_\delta$ such that $\forall \mathbf{v}_\delta \in \{\mathbf{v} \in [H^1(\Omega_\delta)]^3 \cap [P_c^{N+1}(\Omega_\delta)]^3 \mid \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{D,\delta}\}$, $\forall q_\delta \in Q_\delta$

$$\int_{\Omega_\delta} \frac{\rho}{\Delta t}\mathbf{u}_\delta^{n+1}\mathbf{v}_\delta + \int_{\Omega_\delta} \frac{\rho}{\Delta t}(\mathbf{u}_\delta^n \circ \mathbf{X}_\delta^n)\cdot\mathbf{v}_\delta + \mu\int_{\Omega_\delta} \nabla\mathbf{u}_\delta^{n+1} : \nabla\mathbf{v}_\delta$$
$$-\int_{\Omega_\delta} p_\delta^{n+1}\ \mathrm{div}(\mathbf{v}_\delta) = 0 \quad (14)$$
$$\int_{\Omega_\delta} q_\delta\ \mathrm{div}(\mathbf{u}_\delta^{n+1}) = 0 \quad (15)$$

This method is implemented using the `convect` operator of FreeFem++.

The method of characteristics is unconditionally stable [**?**].

1.4. MRI simulations

MRI simulations was initially developed to optimize and test the MRI sequences. This method allows to predict easily, quickly and at a low cost the result of any complex experiment, with any physical parameters. Another motivation is educational, for basic understanding of MRI physics. Finally, this can be used for physical model validation, as it is the case in the *Vivabrain* project, where we test our Computational Fluid Dynamics models in the last part of the pipeline.

1.4.1. *JEMRIS MRI simulator*

JEMRIS is an advanced MRI simulator software written in C++, open-source and freely modifiable [16]. Several optional Matlab GUIs offer great freedom to design original sequences, including arbitrary pulse shapes and gradients, as well as management of various physical parameters. Many off-resonance factors are taken into account, such as chemical shift, concomitant gradient fields, magnetic susceptibility, etc., and possibly other phenomena accompanying the imaging process (molecular diffusion, Gaussian noise, patient move, ...).

1.4.2. *Principle: Isochromat summation*

In MRI, the signal collected over time is generated by the temporal variations of the macroscopic magnetization of tissues. This signal contains all the information needed to reconstruct the final image.

The most popular technique for MRI simulations is isochromat summation. The sample to be imaged is divided into equal subvolumes called isochromats (see Fig. 4). Those subvolumes are supposed to possess uniform physical properties: spin relaxation times $T1$, $T2$, $T2^*$[2], equilibrium magnetization M_0 and magnetic susceptibility χ.

The measurable signal emitted by one isochromat is obtained by numerically solving Bloch equations. The whole MR signal is then obtained by summing the contribution of each isochromat over the entire sample.

1.4.3. *Bloch equations: Temporal evolution of magnetization*

Bloch equation gives the temporal evolution of tissue magnetization for one isochromat:

$$\frac{d\mathbf{M}}{dt} = \gamma\mathbf{M}\times\mathbf{B} - \hat{\mathbf{R}}(\mathbf{M}-\mathbf{M_0}) \quad (16)$$

where $\mathbf{M}$ is the magnetization vector of the tissue, γ is the gyromagnetic ratio of hydrogen, $\mathbf{B}$ is the external magnetic field and $\hat{\mathbf{R}}$ the relaxation matrix containing $T1$ and $T2$.

The magnetic field term $\mathbf{B}(\mathbf{r},t)$ contains all the MR sequence elements (gradients and radio frequency pulses). Its expression is given by:

$$\mathbf{B}(\mathbf{r},t) = [\mathbf{G}(t).\mathbf{r} + \Delta B(\mathbf{r},t)].\mathbf{e_z} + \mathbf{B_1}(\mathbf{r},t) \quad (17)$$

[2] $T2^*$ time is due to local magnetic field inhomogeneities. It results in a shorter transversal relaxation time.

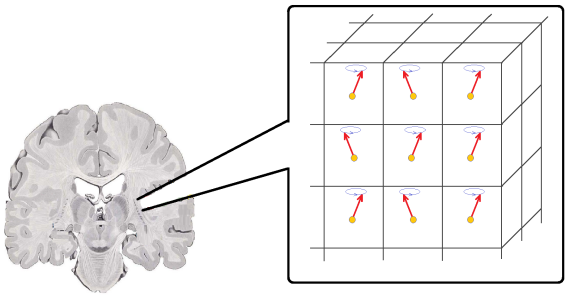


FIGURE 4. Cutting the sample into isochromats. A magnetization vector is associated to each isochromat and its evolution is monitored during the acquisition sequence. The collected MR signal corresponds to the transverse component of the magnetization.

where $\mathbf{G}(t)$ is the gradients sequence, $\mathbf{r}$ is the isochromat position, $\Delta B(\mathbf{r},t)$ corresponds to the field inhomogeneities due to off-resonance and non-uniform gradients, and $\mathbf{B_1}(\mathbf{r},t)$ the radio frequency pulses sequence.

In the Lagrangian version presented here, Bloch equations are a simple ordinary differential equation (ODE) system. The software we use for our simulations, JEMRIS, solves those equations with the ODE solver CVODE, which is part of the SUNDIALS [3] libraries. Numerical solving is based on Adams-Moulton linear multistep method.

1.4.4. *Flow motion simulations*

Natively, Bloch equation solving in JEMRIS is only dedicated to simulate static tissues. So, we provide an extension to the software in order to simulate fluid travelling [8]. Movements are taken into account with a Lagrangian approach, which requires to determine each individual spin trajectory. While solving Bloch equations, this approach avoids the need to use a different treatment for static tissues and flowing particles. We simply vary the position of the spin over time, which changes the field value seen by the particle:

$$\mathbf{r} = \mathbf{r}(t) \quad \Rightarrow \quad \mathbf{B}(\mathbf{r},t) = [\mathbf{G}(t).\mathbf{r}(\mathbf{t}) + \Delta B(\mathbf{r},t)].\mathbf{e_z} + \mathbf{B_1}(\mathbf{r},t)$$

By default, JEMRIS only allows to specify one trajectory for all spins, in order to simulate movement of a rigid sample. So we added a specific class to the C++ code to allow users to specify a different trajectory for each spin. A simple example of MRI simulations performed with our modified version of JEMRIS is shown on Fig. 5. As it appears on that image, we used four spins with specific individual behaviors: two static spins (at the top of the image), and two moving spins with diagonal trajectories (at the bottom).

With this new version, it thus becomes possible to describe flow phenomena, after converting flow data to the proper format (Fig. 6).

2. BENCHMARK SETUP

The aim of this benchmark is to validate the CFD simulations by comparing the numerical outputs with experimental data. For that purpose, a physical phantom was designed as a double bifurcation fluid circuit so that it can reproduce acceleration, deceleration and recirculation all of which can be encountered in real vascular networks. The characteristics dimensions of the model were carefully chosen so that it mimics human cerebral little vessels, like lingual emissary veins.

The inlet branch, the left channel, has a diameter of 5 mm and a length of 40 mm, the lower left and lower right branches have a diameter of 2 mm and a length of 30 mm each, the upper left and upper right branches have a diameter of 3 mm and a length of 30 mm, and finally the outlet branch, the right channel, has a diameter

[3] https://computation.llnl.gov/casc/sundials/main.html

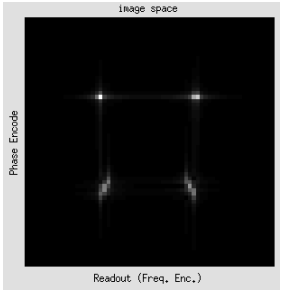


FIGURE 5. Our modifications allow JEMRIS to simulate individual trajectories for each spin. Here, an example of simple gradient echo MRI sequence with two static spins at the top and two moving spins at the bottom.

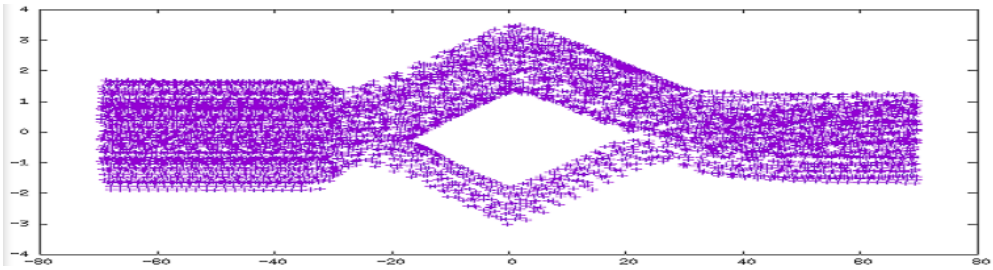


FIGURE 6. The Lagrangian approach we used to simulate flow motion requires to convert the velocity field determined with Navier-Stokes equations into individual particles trajectories, with a path tracer code.

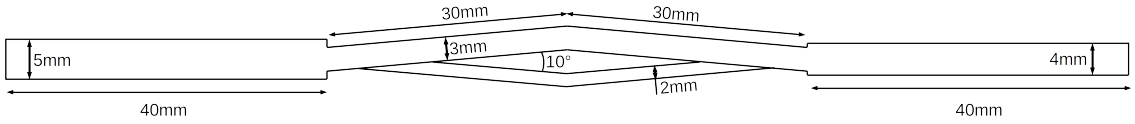


FIGURE 7. The physical phantom.

of 4 mm and a length of 40 mm. The angle between the two branches of the bifurcation is 10° (see Fig. 7). The physical phantom is machining in a rectangular block of Plexiglas that we consider as a non deformable material.

The fluid should have been chosen so that it satisfies isothermal and incompressible Newtonian blood properties, but instead, for the sake of simplicity, we used water to perform our experiments. The constant density and the dynamic viscosity are then $0.001 g/mm^3$ and $0.001 Pa \cdot s$, respectively.

For the MRI measurements, we considered the following setup. We used a Masterflex roller pump at a flow rate of 100 mL/min to inject fluid into the phantom, which is in the MRI. The fluid then goes to a reservoir crossing a heart rate getting which is used for the PC-MRI acquisition, see Fig. 8.

We used a 3T Philips Achieva dStream MRI machine for all acquisitions at the University Hospital of Amiens, Picardie.

Morphological MRI

In order to obtain the geometry of the phantom, we performed a morphological MRI.

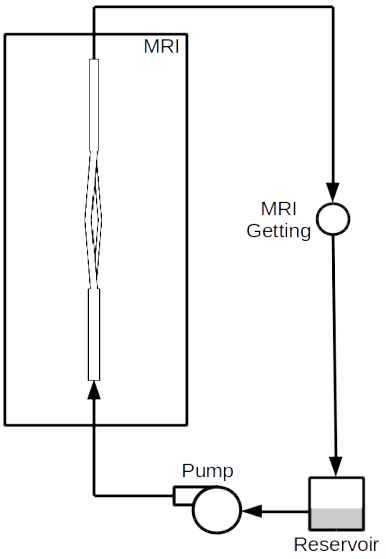


FIGURE 8. MRI benchmark.

The sequence used is a 3D phase contrast angiography with a flip angle of 12° for a pixel size of 0.5 mm × 0.5 mm and a slice thickness of 0.6 mm. The microscopic coils used for the MRI flow measurements have an inner diameter of 47 mm.

Flow MRI

To obtain the flow measurements in the different branches of the phantom, we performed a phase contrast MRI (PC-MRI). We used 2D QFlow sequences with an echo time of 11 ms, a repetition time of 18 ms, a flip angle of 30°, a slice thickness of 1 mm and finally a pixel size of 0.35 mm × 0.35 mm. The field of view is 30 mm × 30 mm, the number of slices by cycle is 32 and the acquisition time is around 2 minutes. Encoding velocities are summarized in Tab. 1. The microscopic coils used for the MRI flow measurements have an inner diameter of 47 mm.

SLICE	ENCODING VELOCITY
Inlet	400mm/s
Outlet	600mm/s
Other	550mm/s

TABLE 1. The characteristics of the flow MRI.

MR images were processed at the BioFlowImage laboratory with the home-made software Flow Analysis [3].

3. SEGMENTATION RESULTS

3.1. **Comparison MRI geometry – realistic geometry**

First, we compare the exact geometry with the MRI geometry to assess the accuracy of the MRI measurements. For this purpose, we need to segment the MRI images and we use two methods: the first is a simple thresholding (Fig. 9(a)) while the second is a snake included in the software ITK-SNAP and detailed in subsection 1.1 (Fig. 9(b)). To estimate the differences between the realistic geometry and the MRI geometry, we measure diameters at the radial slices given on Fig. 13 and we list results in Tab. 2.

We can notice that the thresholded MRI is thinner than the segmented MRI, while the segmented MRI is larger. This last issue is due to the snake which evolves outside the boundary of the desired object. In fact, the MRI geometry can not be larger than the realistic geometry, so the thresholded MRI is closer to the reality. Nevertheless a lot of information is lost, certainly because of the smoothness. Another issue is the difficulty to match the two geometries. Indeed, the plane of the MRI acquisition is slightly inclined to the horizontal plane and as a result, we have to perform a transformation to align the two geometries.

In both cases, the input of the MRI does not correspond to the exact input, as well as the output. Indeed, the MRI tends to smooth the straight forms. These differences can skew the results of the MRI measurements in the next parts and the MRI acquisition should be improved in the future.

(a) Thresholded MRI

(b) Segmented MRI

FIGURE 9. Comparison of the realistic geometry with MRI segmentations, the first is obtained by a simple threshold while the second by the classical snake approach.

RADIAL SLICES	IN	UL	LL	UC	LC	UR	LR	OUT
Realistic mesh	5.00	3.00	2.00	3.00	2.00	3.00	2.00	4.00
Thresholded MRI	3.36	2.70	1.97	2.45	1.73	2.70	1.97	3.12
Segmented MRI	5.10	5.78		3.60	2.30	5.27		4.45

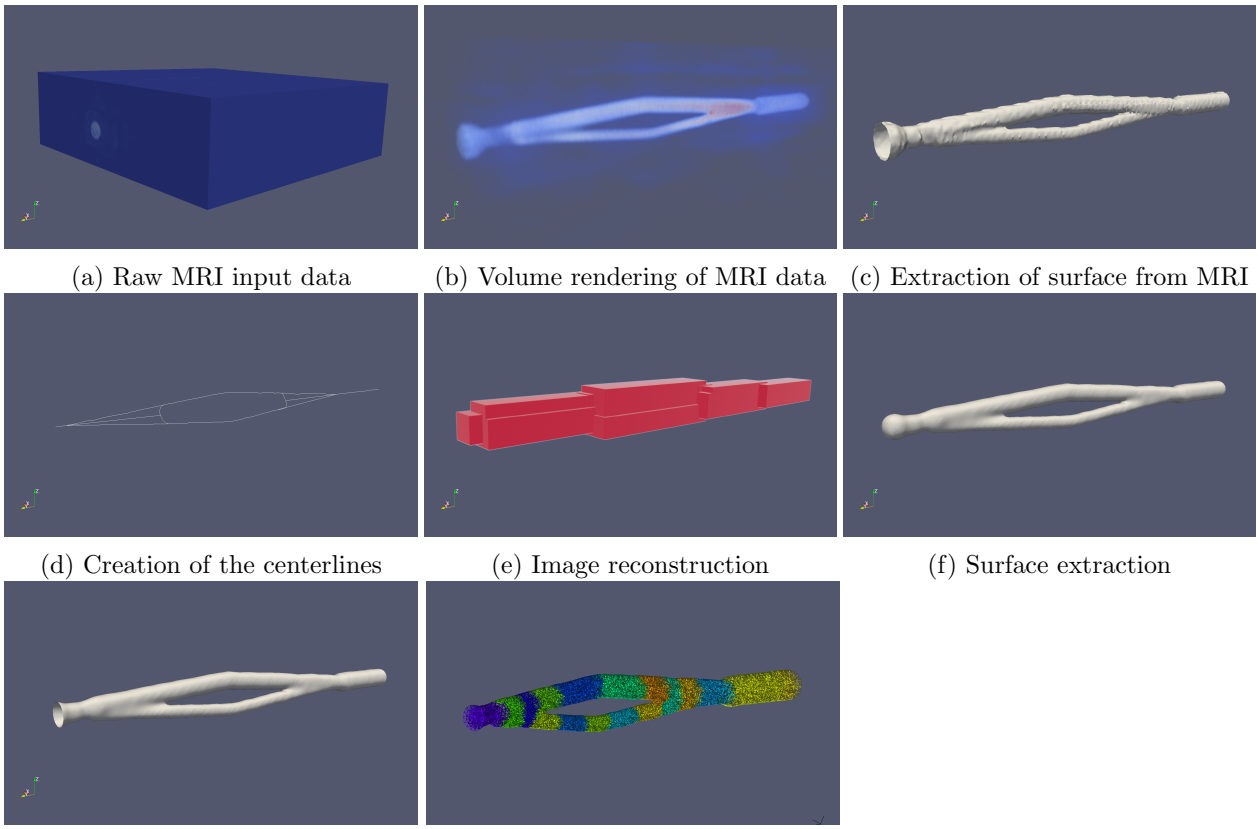
TABLE 2. Comparison of the diameters (mm) at the radial slices : IN=inlet, UP=uppper left, LL=lower left, UC=upper center, UL=upper left, UR=upper right, LR=lower right and OUT=outlet. As the UL and LL parts and the UR and LR parts respectively are sticked together in the segmented IRM, we compute the diameters on the entire left slice and on the entire right slice respectively which are theorically equal to 5.11mm.

3.2. **Comparison AngioTK geometry – realistic geometry**

We used AngioTK up until the numerical simulations step as a tool to compare ground truth geometries built from real measurements and geometries extracted from MRI. In Fig. 10, we illustrate the different steps of mesh processing that lead to a mesh suitable for numerical simulations.

From left to right and from top to bottom, we have:

- The original MRI data visualized as a block (a) and with volume rendering (b);
- The initial mesh extracted from MRI data (c). This extraction is performed based on the level-set method;
- We then build centerlines (d) with additional information, like the radius of the larger inscribed sphere at each centerline point;
- Based on the centerlines, we generate a new volume of data (e) and extract a proper mesh (f); this mesh is then opened in (g);



(a) Raw MRI input data (b) Volume rendering of MRI data (c) Extraction of surface from MRI

(d) Creation of the centerlines (e) Image reconstruction (f) Surface extraction

(g) Mesh opening (h) Tetrahedralization/partitioning

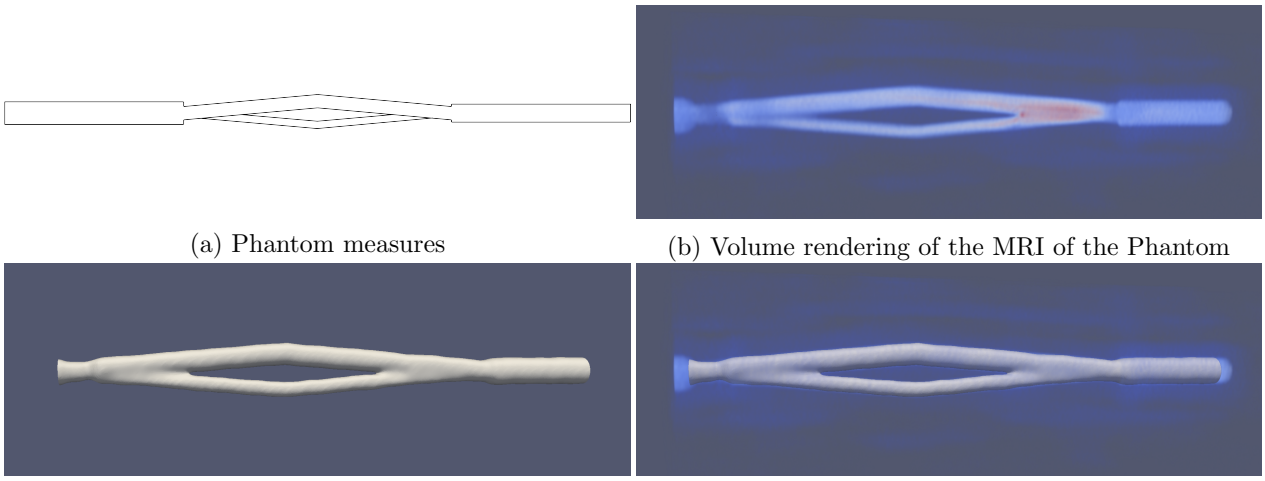
FIGURE 10. A sample execution of the AngioTK pipeline on the phantom dataset.

- We generate a volumic mesh using tetrahedra and partition the mesh for numerical simulations in (h) (here on 16 processors).

Let us first have a look at the difference between the original MRI data and the final mesh generated with AngioTK in Fig. 11

Fig. 11 (a) and (b) show that the MRI did not capture the entire Phantom during the acquisition. We are indeed losing a part of inlet and outlet tubes through which the water flowed. Fig. 11 (d) shows that we correctly reconstructed the shape of the Phantom through MRI processing and mesh generation. However, AngioTK did not accurately capture the very beginning of the inlet and the end of the outlet parts. This is mainly due to the parametrization of the algorithms we used. Thus capturing the required parts would require a finer tuning of the whole process. A second issue is located at the two bifurcations of the tube. In the real phantom, the tubes are cut though plexiglas, so as they converge/diverge, the separation is sharp. We can however notice with the reconstruction that the tubes are joined together for a certain distance between diverging/converging. This is due to the reconstruction of the phantom using spheres, as AngioTK is designed to be used with veins and arteries.

To measure the accuracy of the reconstruction, we illustrate in Tab. 3 the differences between real data (cf. Fig. 7) and measured data in term of tube width. We compare the tube width of internal section to avoid the parts influenced by the use of AngioTK as previously mentioned. The sections of the mesh from which we extract the data are described in Fig. 12.



(a) Phantom measures

(b) Volume rendering of the MRI of the Phantom

(c) Mesh generated by AngioTK

(d) Superimposition of volume rendering and mesh

FIGURE 11. Comparison between the original MRI data and the final mesh generated by AngioTK.

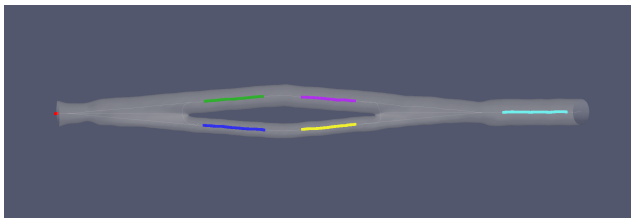


FIGURE 12. Sections on which we compute the minimal and maximal inscribed sphere radius. The different sections are the following: the inlet in red, the upper branch section 1 in green, the upper branch section 2 in purple, the lower branch section 1 in blue, the lower branch section 2 in yellow and the outlet in cyan.

For the size to be comparable, we decided to compute a ratio between the real phantom tube size and the size of the biggest inscribed sphere in the created mesh. Note that the largest inscribed sphere information is computed at each centerline point. If the reconstruction is performed correctly, we should get a similar ratio between the different section sizes.

As it can be seen in Tab. 3, the ratio are close to 2 for the different sections, so we get a good match of the original size. The only problematic case is at the inlet, where is strongly differs from the other ratios. This is again due to the fact that we were not able to grasp the real shape of the inlet.

4. SIMULATIONS RESULTS

The fluid flow simulations were performed using FEEL++ and FREEFEM++. The finite element spaces used to discretize the velocity and the pressure are the Inf-Sup stable Taylor-Hood finite elements $P2P1$.

For the FREEFEM++ simulations, we used the method of characteristics [4] described in section (1.2.2). In order to speed-up the resolution, this method is coupled with an iterative Uzawa Conjuguate Gradient algorithm [14] with a Cahouet-Chabart preconditionner [5]. As for the FEEL++ simulations, we used a second order finite difference scheme to approximate the time derivative and a second order extrapolation of the non-linear convective term [17], described in section (1.2.1). The resolution in the latter is made using the LU solver.

Section	Original size (mm)	Number of sample points	Size of inscribed sphere (no unit, min-max values)	Ratio (original size / inscribed sphere size)
Inlet branch (Fig. 12 in red)	5	2	[1.634, 1.770]	3.060, 2.824
Upper branch (Fig. 12 in green)	3	90	[1.489, 1.553]	2.014, 1.931
Upper branch (Fig. 12 in purple)	3	83	[1.552, 1.593]	1.932, 1.883
Lower branch (Fig. 12 in blue)	2	93	[0.985, 1.056]	2.03, 1.893
Lower branch (Fig. 12 in yellow)	2	83	[1.023, 1.079]	1.955, 1.853
Outlet branch (Fig. 12 in cyan	4	97	[1.981, 2.024]	2.019, 1.976

TABLE 3. Size comparisons between the size of the real phantom and the data extracted from MRI data.

The discretized Navier-Stokes equations were supplemented for both FEEL++ and FREEFEM++ simulations by a no-slip boundary conditions on the lateral surface of the computational domain, since we are dealing with viscous fluid. At the inflow and the outflow we imposed a Dirichlet and a Neumann boundary conditions, respectively. Although the choice of a Neumann-type boundary conditions at the outflow in physiological flows may lead to possible instabilities with recirculations because of the loss of energy estimates, we did not use stability technics and did not observe any instability at the chanel exit because the exit is sufficiently far, and the Reynolds number is in the lower laminar range. For the Dirichlet boundary conditions two flow profiles were implemented at the inlet of the channel: *i*) a Poiseuille profile with constant flow and *ii*) a pulsatile flow retrieved by using the experimental measurements of the flow at the inlet during a 5s MRI acquisition.

We start our simulations with a fluid at rest, that is, $p = 0$ and $\mathbf{u} = \mathbf{0}$ everywhere in Ω. In this part, in order to avoid additional geometric errors, the geometry was constructed using real dimensions described in Section (2). The meshing of the geometry was carried out using Gmsh 2.8.3 [9] and the partitioning using Metis [11]. An adapted mesh size was chosen depending on the diameter of the channel. A "h_{max}" was prescribed in the inlet and outlet channels, a "$h_{average}$" was prescribed on the upper channel of the bifurcation and a "h_{min}" was set in the lower channel of the bifurcation. Three levels of refinements were performed in order to make a mesh convergence study. The corresponding mesh characteristics are reported in Tab. 4.

GEOMETRY	h_{min}	h_{max}	$h_{average}$	TETRAHEDRA	DOF
M1	0.2	0.5	0.3	157245	769662
M2	0.25	0.625	0.375	93655	469008
M3	0.3	0.75	0.45	60349	307510

TABLE 4. The characteristics of the three types of geometries.

The comparison is made in terms of axial component of the velocity and pressure along the centerline and at various radial sections (see Fig. 13) and normalized wall pressure difference along the length of the domain.

4.1. **Comparison Feel++– FreeFem++ in the phantom**

The numerical fluid simulations performed in the phantom geometry using FEEL++ and FREEFEM++ are quantitatively of the same order of Fig. 14. But in order to carry out a cross validation of these two finite

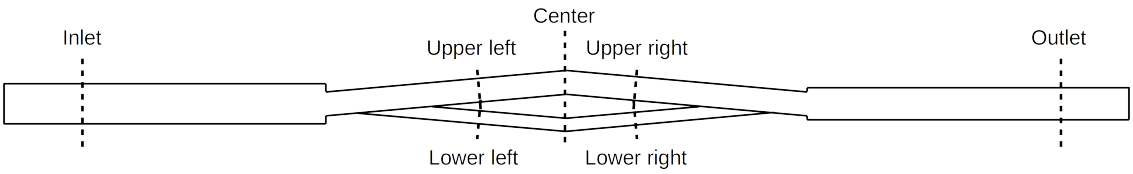


FIGURE 13. Radial slices where the velocity profiles are plotted.

elements method libraries, we compared pressure and velocity profiles at six different radial sections previously defined, and along the centerline.

A constant velocity of 41mm/s (corresponding to the lowest velocity of the MRI measurements that we describe later) is imposed at the inlet for one second. The results are shown for the final time step.

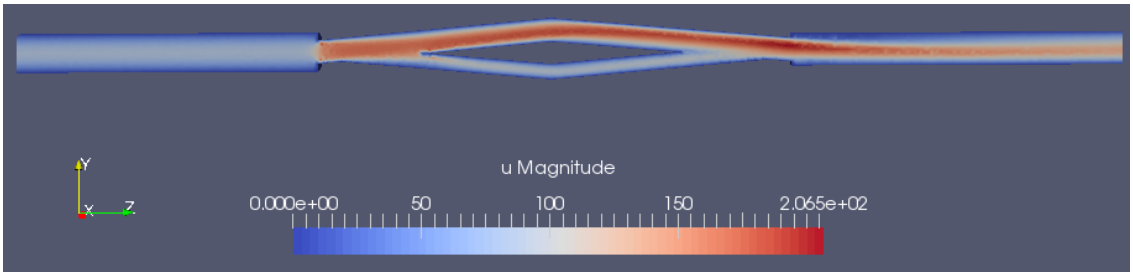


FIGURE 14. Velocity magnitude, M3 mesh, constant flow.

Pressure and velocity along the centerline are shown in Fig. 15 for FEEL++ and FREEFEM++ with a mesh convergence. We can see that FREEFEM++ gives a slightly higher pressure gradient than FEEL++, for the tree levels of refinement. However, velocity profiles are very similar.

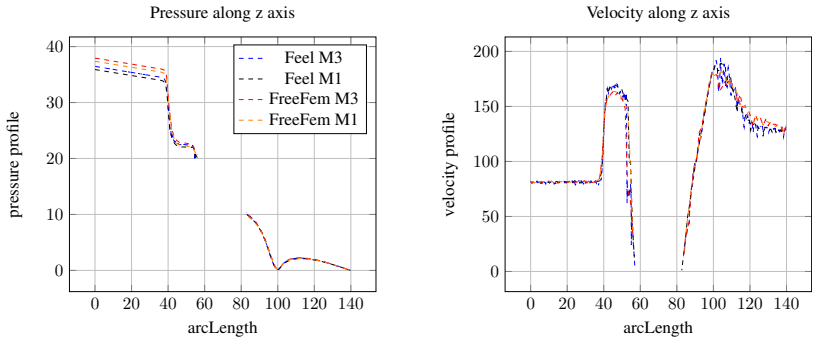


FIGURE 15. Comparison FEEL++ vs FREEFEM++ for a constant flow (V_{min}), $P2P1$: Velocity and pressure profiles along z axis.

At the inlet section, the profiles overlap, retrieving the Poiseuille profile boundary condition imposed at the inlet. As for the outlet, the velocity profiles show also a good agreement.

Velocity profiles at the lower and upper channels, left and right sections are shown in Fig. 17. Some differences can be observed with the coarse mesh M3, but the mesh convergence shows that the results tend towards the same solution.

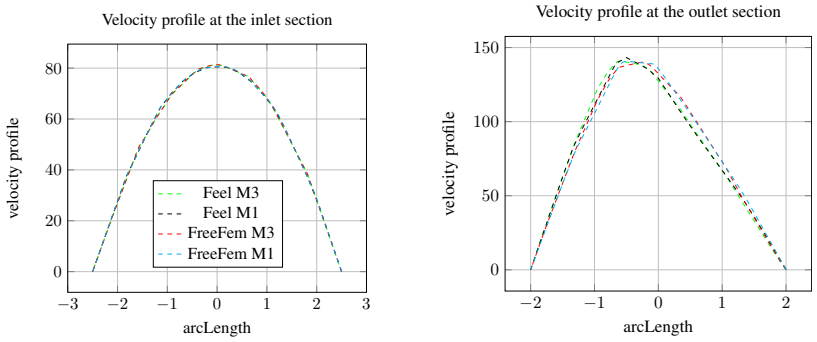


FIGURE 16. Comparison FEEL++ vs FREEFEM++ for a constant flow (V_{min}), $P2P1$: Velocity at the inlet and outlet sections.

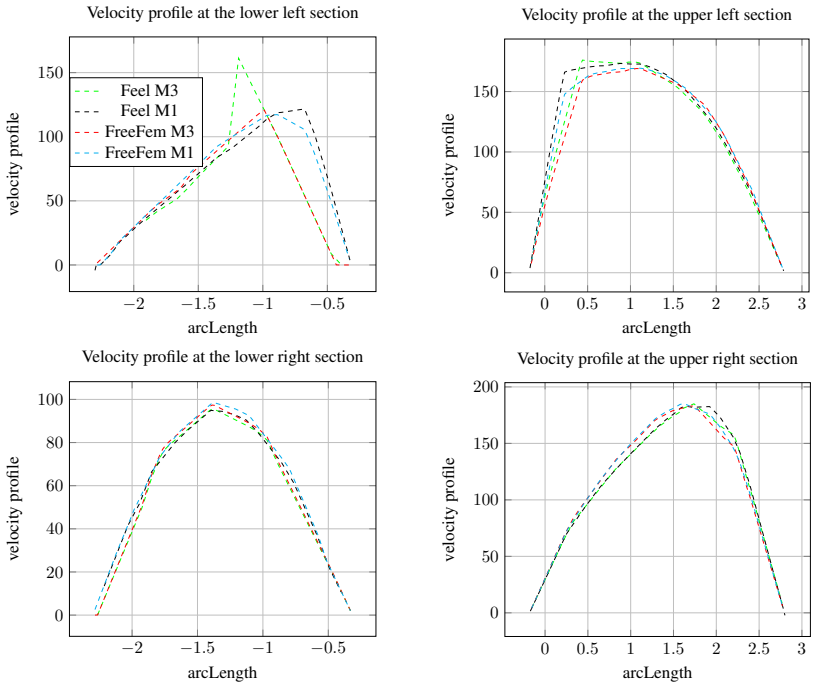


FIGURE 17. Comparison FEEL++ vs FREEFEM++ for a constant flow (V_{min}), $P2P1$: Velocity profile at the right and left sections in the upper and lower channels.

Most of the results are equivalent in this FEEL++–FREEFEM++ comparison. We believe that the observed differences can be due to the choice of algorithms for the two FEM libraries; one uses a second order time discretization while the other uses a first order one. FEEL++ algorithm equally uses a domain decomposition method that can impact the computed solutions. Even if these differences exist, they are quite small and we can see a large correspondence of the results for the two libraries.

The Reynolds number in the constant and pulsatile flow cases varies between 200 and 700, thus the inertial terms could not be neglected. However, no recirculation or instabilities have been observed.

4.2. **Comparison Numerical simulations – MRI measurements in the phantom**

Now a pulsatile flow, obtained by PC-MRI measurement, is used to impose the velocity at the inlet of the phantom.

Comparison between FEEL++ and FREEFEM++ simulations and PC-MRI measurement is shown in Fig. 18 for the input and the output, and in Fig. 19 for the two branches.
The input and output flows are very similar, both between the two simulations and between simulations and experiment. Branches flow are equally similar for the two simulations but MRI measurements are slightly higher. These differences can be explained by the fact that PC-MRI images have to be manually segmented to obtain velocity and flow informations, a process that usually overestimates the real conditions of the experiment: operator dependent, manually segmented on a screen and dependent of the slice angle during the MRI sequence. At the inlet and the outlet, the pipe is quite large and the segmentation is very accurate, but in the two small branches greater errors may occur.

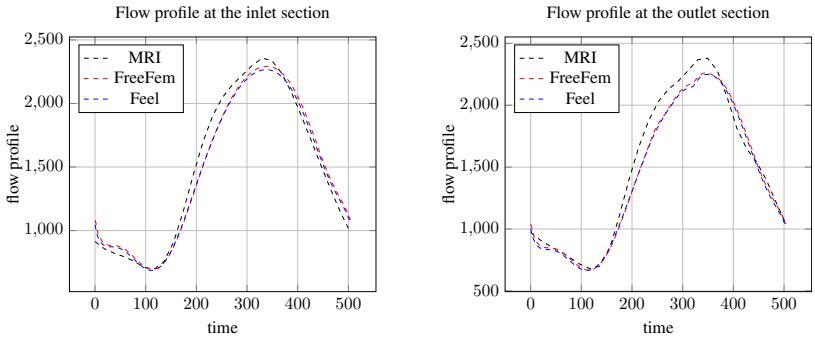


FIGURE 18. Comparison FEEL++ vs FREEFEM++ on the M3 mesh with a pulsatile flow, flow profile at the inlet and outlet sections during time.

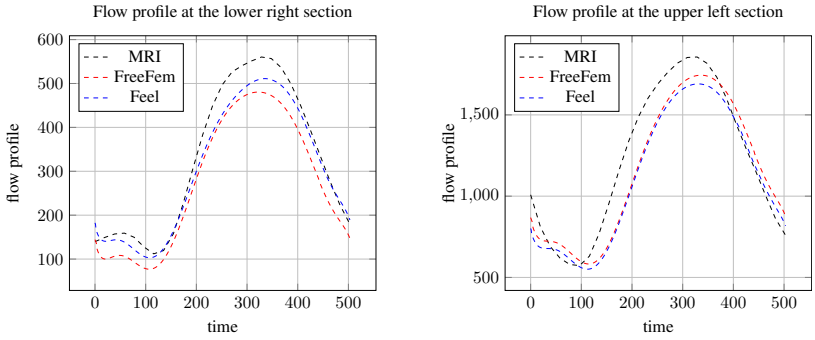


FIGURE 19. Comparison FEEL++ vs FREEFEM++ on the M3 mesh with a pulsatile flow, flow profile at the left sections in the upper and lower channels during time.

4.3. **Comparison Feel++– FreeFem++ in the cerebral venous network**

The final goal of the *Vivabrain* project is to compute virtual cerebral angiography images. It is still too early to perform MR acquisition of complete angiographic images, but we have already reached the stage of blood flow simulations in the complex cerebral venous network. In this type of realistic geometry, a main issue remains the experimental validation of the results. If the physical phantom can reach that goal in the case of simple

bifurcations, such approaches are hardly tractable for complex network. Therefore, finding a way to validate this key step is essential. In this part, we run a cross-validation between the FEEL++ and FREEFEM++ results in this realistic geometry.

First, we describe briefly an appropriate model for flow in the cerebral venous network (see [12] for more details). At the macroscopic scale, the brain venous network (see Fig. 20) is composed by – input – veins (7)–(11) draining the blood into the superior sagittal sinus (2) and the straight sinus (3), until their confluence (4). The blood then passes into the transverse (5) and sigmoid parts (6) of the lateral sinuses, and reaches an extracranial area, composed of the – output – internal jugular veins (1).

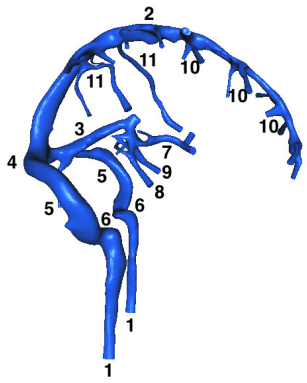


FIGURE 20. Cerebral venous network.

Concerning the flow in these vessels, we adopt standard assumptions: (i) the blood density is constant; (ii) the flow is assumed to be incompressible and isothermal; (iii) blood is supposed to be Newtonian. Another important issue is the relevance of using either a complex fluid-structure interaction model or a fluid model (hence neglecting the interaction with the vessel wall). Intracranial veins are quite constrained in the brain, thus they are considered as rigid.

The boundary conditions are: (i) inflow: a steady profile (constant velocity of small magnitude, due to microcirculation exit); (ii) outflow: homogeneous natural conditions; (iii) lateral boundary: no-slip condition, since walls are assumed to be rigid.

Taking into account all the previous hypotheses leads to consider the Navier-Stokes equations for modeling the blood flow dynamics. As a first approximation, we solve instead the corresponding steady Stokes problem with the same boundary conditions as those considered in the Navier-Stokes equations with a dynamic viscosity of $3.5 Kg.mm^{-3}$.

Concerning the Dirichlet boundary conditions on the inlet and the wall, there are two ways to impose them (see Fig. 21). The first way is to first impose a constant velocity at the inlet and then a no-slip condition (zero velocity) on the wall. In this case, the velocity is null on the wall and on the elements which are at the intersection between the inlet and the wall sections of the boundary. The second way is to first impose the no-slip condition on the wall and then to impose the constant velocity at the inlet, therefore the velocity is zero on the wall, except for the elements which are at the intersection between the inlet and the wall sections of the boundary where it is equal to the constant velocity imposed on the inlet section.

We run the Stokes problem on the realistic geometry and we take an initial inflow of 6000 mm^3/s which corresponds to an initial velocity of 28 mm/s. The areas of the input and output veins calculated by the two libraries are identical, contrary to the flows. These data are described in Tab. 5 and are given in percentage of the total area and total flow. On the one hand, we can notice for the two libraries that the total inflow is preserved in the Inlet/Wall case but is significantly lower than the initial flow (respectively 5164 mm^3/s in FREEFEM++ and 5406 mm^3/s in FEEL++). On the other hand, the total inflow is no longer preserved in the Wall/Inlet but the inflow (-5985 mm^3/s and -5987 mm^3/s) and the outflow (6227 mm^3/s and 6175

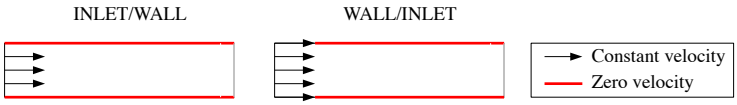


FIGURE 21. Different ways to impose the boundary conditions.

mm^3/s) are closer than the initial flow. Finally, the inflows and outflows are very close in FREEFEM++ and FEEL++ and regarding in percentage of the total flow, they are substantially equal. This would ensure a correct simulations.

4.4. **Comparison MRI simulation – Numerical simulations**

MRI simulations were performed using JEMRIS [16] with constant flow computed from numerical simulations. The input required in order to perform the simulations are the trajectories of a fixed number of particles, that are the positions along the flow at each time step, for the whole MRI sequence duration. For constant and pulsatile flow, a VTK [15] script was implemented to save the *particle tracer* of a sphere of point sources with its corresponding computed velocity field.

We show below a simulated phase contrast image of constant flow obtained with JEMRIS (Fig. 22). About 19 000 spins trajectories were calculated on a time interval of 5s (700 MB of data) from numerical data calculated in amount for constant flow with FEEL++ and FREEFEM++. We used a phase contrast sequence with resolution 128, matrix 180×30, TE of 8 ms, TR of 100 ms, Venc of 400 mm/s and Nex=1. Calculations took only 15 minutes with 20 CPU on Romeo supercomputer from Reims[4].

This velocity image was then filtered with its corresponding signal magnitude image (Fig. 23). To do this, we applied a mask to set to zero the pixels of the velocity map when the corresponding signal intensity in that pixel was inferior to a fixed threshold value. Thus, only the significant values of velocity present in the phantom geometry appear on the final image. We can notice an important lack of spins in the lower branch because of the low number of flow particles traveling along that path.

The velocities measured in each branch give right order of magnitude compared to initial data calculated with FEEL++ and FREEFEM++ (Tab. 6 and Fig. 24). However, the low resolution used for MRI simulation lead to an averaging of the velocities in the voxels[5] (partial volume effect). This phenomenon lead to an underestimation of the peak value in each branch of the phantom, as the values measured are averaged on a wide range of velocities. This effect is particularly severe in the thinnest branch, where the gradients of velocities are very high inside a voxel, leading to an important underestimation, even more increased by the lack of particles.

Those problems should be solved, in the future, by increasing the total number of particles (to increase the number of spins in the lower branch) and by enhancing image resolution (in order to limit partial volume effects). This will lead, on the other hand, to an increase of data files volume and of total computation time. Simulations of pulsatile flow will also be performed in short term works, but it will probably need to reuse the same trajectories of a few number of cardiac cycles translated in time, in order to cover the whole MRI sequence duration. Otherwise, the trajectories data volume could rapidly become prohibitive. Another step to improve our simulations in the future will be to check the effect of spatial and temporal spins density variations on the quality of the velocity measured.

[4] https://romeo.univ-reims.fr/
[5] Voxel: 3D pixel.

		Flow Inlet/Wall		Flow Wall/Inlet	
Label	Area (%)	FreeFem++ (%)	Feel++ (%)	FreeFem++ (%)	Feel++ (%)
Input					
1	5.9	5.12	5.11	5.09	5.09
2	1.98	1.96	1.97	1.98	1.98
3	1.85	1.83	1.84	1.86	1.86
4	1.26	1.23	1.24	1.26	1.26
5	1.45	1.43	1.44	1.42	1.45
6	4.94	4.98	4.97	4.94	4.94
8	1.25	1.23	1.24	1.25	1.25
9	2.52	2.52	2.52	2.52	2.52
10	6.05	6.09	6.07	6.05	6.05
11	5.23	5.24	5.24	5.21	5.22
12	6.22	6.28	6.26	6.22	6.22
13	1.56	1.54	1.55	1.56	1.56
14	0.85	0.83	0.84	0.85	0.847
15	6.42	6.41	6.41	6.42	6.42
16	7.28	7.29	7.29	7.28	7.28
17	1.70	1.68	1.68	1.70	1.70
18	1.89	1.87	1.88	1.90	1.90
19	0.96	0.93	0.94	0.96	0.96
20	2.43	2.40	2.41	2.43	2.43
21	0.52	0.51	0.51	0.52	0.52
22	2.64	2.61	2.62	2.64	2.64
23	6.30	6.34	6.33	6.30	6.31
25	3.54	3.54	3.54	3.54	3.54
26	5.43	5.46	5.45	5.43	5.43
27	3.92	3.92	3.92	3.93	3.92
28	5.78	5.83	5.81	5.78	5.78
29	6.34	6.36	6.35	6.34	6.34
30	1.18	1.16	1.17	1.18	1.18
31	3.40	3.39	3.39	3.40	3.40
Total	214.85 mm^2	-5163.93 mm^3/s	-5405.517 mm^3/s	-5984.6 mm^3/s	-5987.07 mm^3/s
Output					
7	45.08	69.18	69.1	69.19	69.2
24	54.92	30.82	30.9	30.81	30.8
Total	37.64 mm^2	5163.78 mm^3/s	5405.51 mm^3/s	6227.29 mm^3/s	6175.05 mm^3/s

Table 5. Comparison between flows in cerebral veins, calculated by FreeFem++ and Feel++. These flows are given in percentage of total inflow and outflow.

5. Conclusion

The MRI–realistic geometry comparison shows qualitative differences between real and MRI-measured diameters of each part of the phantom. This study emphasizes the necessity of segmentation or threshold procedure improvement for a physiological use.

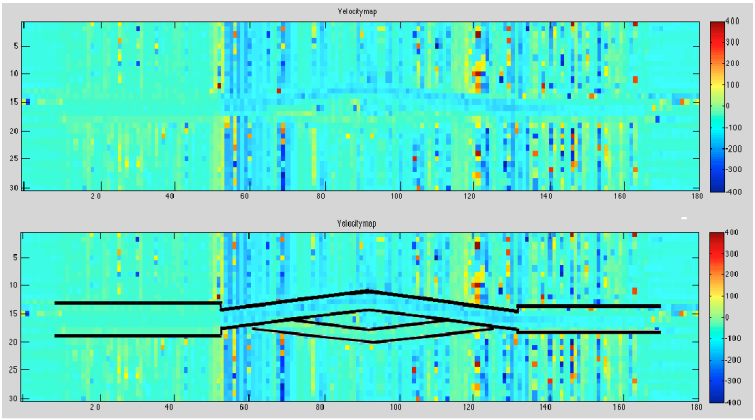


FIGURE 22. Above: Velocity map simulated with phase contrast sequence in JEMRIS. Below: Superimposition of the simulated image with the phantom geometry. The remainder of the image is noise due to air.

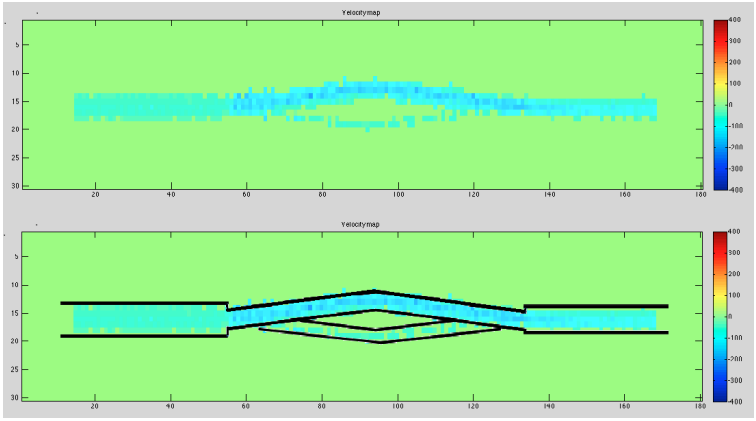


FIGURE 23. Above: Same image as Fig. 22, filtered with its corresponding magnitude image. Below: The filtered velocity map matches well with the initial phantom geometry, but we can notice a lack of particles in the lower branch.

BRANCH	FEEL++/ FREEFEM++	JEMRIS
Inlet	80mm/s	70mm/s
Outlet	140mm/s	110mm/s
Upper branch	170mm/s	150mm/s
Lower branch	110mm/s	60mm/s

TABLE 6. Comparison of velocity peaks measured in the center of each branch with JEMRIS vs Numerical simulations. In each branch, the low resolution used for MRI simulations lead to an averaging and an underestimation of the peak velocity (type of partial volume effect), particularly severe in the thinnest branch. This could be corrected with more particles trajectories and higher resolution.

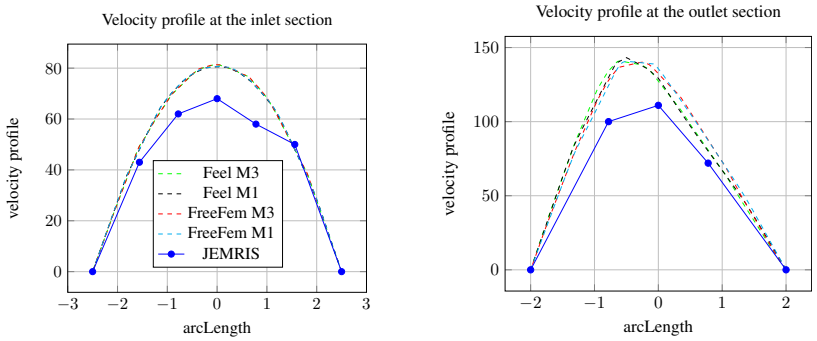


FIGURE 24. Velocity profiles at the inlet and outlet sections measured with JEMRIS. Velocities measured give a good order of magnitude compared to initial numerical simulations data but we can observe an underestimation of velocity peak, due to low resolution and partial volume effects.

The FEEL++–FREEFEM++ comparison in the phantom shows a huge correlation between the computed solutions. These kinds of comparison allow us to confirm the good agreement of the solution regarding to a physiological use. The FEEL++–FREEFEM++–PC-MRI comparison shows qualitative agreement of the flow in all the phantom. In the realistic geometry of the cerebral venous network, a good agreement of the solutions is equally found between FEEL++ and FREEFEM++ results. However, geometrical errors shown in the first part of the pipeline should not be neglected. For example, if a lower section is used to perform simulations, a higher pressure will be calculated than the physiological one.

The JEMRIS – FEEL++ FREEFEM++ comparison with constant flow showed a good qualitative agreement for the velocity profiles, but an underestimation of peak velocities. This should easily be corrected with more particles trajectories and higher image resolution in the virtual MRI simulations.

To conclude, all the steps of the pipeline are developed and now, we have to adapt them for a realistic geometry.

6. ACKNOWLEDGMENTS

Thanks to Stéphanie Salmon for the supervision of this project. Thanks to Stéphanie Salmon, Marcela Szopos, Christophe Prud'homme, Nicolas Passat, Hugues Talbot, Olivier Balédent, Emmanuel Durand for the supervision of the authors' PhD theses. Thanks to Vincent Chabannes and Odyssée Merveille for their help during the CEMRACS. Thanks to Gwenaël Pagé for the MRI acquisitions. Thanks to PFT Innovaltech for machining the phantom. This project was funded by Agence Nationale de la Recherche (Grant Agreement ANR-12-MONU-0010 (VIVABRAIN)). This work was granted access to the HPC resources of Aix-Marseille Université financed by the project Equip@Meso (ANR-10-EQPX-29-01) of the program "Investissements d'Avenir" supervised by the Agence Nationale de la Recherche. We would like to thanks the IRMIA Labex at IRMA, Strasbourg for funding and Cemosis for the support and development environment provided for this project.

REFERENCES

[1] AngioTK. http://www.cemosis.fr/interdisciplinary-bio-project/2015/09/01/angiotk/.
[2] VIVABRAIN project. http://vivabrain.fr. Accessed: 2014-05-14.
[3] O. Balédent, I. Idy-Peretti, et al. Cerebrospinal fluid dynamics and relation with blood flow: A magnetic resonance study with semiautomated cerebrospinal fluid segmentation. *Invest. Radiol.*, 36(7):368–377, 2001.
[4] J.P. Benque, B. Ibler, and G. Labadie. A finite element method for Navier-Stokes equations. In *Numerical Methods for Non-Linear Problems*, volume 1, pages 709–720, 1980.

[5] J. Cahouet and J.P. Chabard. Some fast 3D finite element solvers for the generalized Stokes problem. *Int. J. Numer. Methods Fluids*, 8:869–895, 1988.
[6] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods - Fundamentals in Single Domains*. Springer-Verlag, Berlin Heidelberg, 2006.
[7] V. Caselles, R. Kimmel, and G. Sapiro. On geodesic active contours. *Int. J. Comput. Vis.*, 22(1):61–79, 1997.
[8] A. Fortin, S. Salmon, and E. Durand. Extension of an mri simulator software for phase contrast angiography experiments. In *Biomedical Simulation*, volume 8789 of *Lecture Notes in Computer Science*, pages 150–154, 2014.
[9] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.
[10] F. Hecht. New development in Freefem++. *J. Numer. Math.*, 20:251–265, 2012.
[11] George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.
[12] O. Miraucourt, O. Génevaux, M. Szopos, M. Thiriet, H. Talbot, S. Salmon, and N. Passat. 3D CFD in complex vascular systems: A case study. In *Biomedical Simulation*, volume 8789 of *Lecture Notes in Computer Science*, pages 86–94, 2014.
[13] C. Prudhomme, V. Chabannes, G. Pena, and S. Veys. 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.*
[14] J.E. Roberts and J.M. Thomas. Mixed and hybrid methods. *Handb. Numer. Anal.*, 2:523–639, 1991.
[15] W.J. Schroeder, B. Lorensen, and K. Martin. *The visualization toolkit*. Kitware, 2004.
[16] T. Stocker, K. Vahedipour, D.I. Pflugfelder, and N.J. Shah. High-performance computing MRI simulations. *Magnetic Resonance in Medicine*, 64:186–193, 2010.
[17] S. Turek. A comparative study of some time-stepping techniques for the incompressible Navier-Stokes equations: From fully implicit nonlinear schemes to semi-implicit projection methods. *Int. J. Numer. Methods Fluids*, 22(2):987–1011, 1996.

Part III

Implementation

Chapter 8

PCD preconditioner implementation

This chapter presents the implementation details of the pressure convection diffusion preconditioner introduced in Section 4 of Chapter 4 in the Feel++ *library. An operator framework was set up to handle the operations that we can apply on the different operators retrieved from the block-structure of this preconditioner. The call of the preconditioner as well as an example of a configuration file are presented in the last section of this chapter.*
The fundamental motivation of this implementation is that the PCD preconditioner is not implemented in PETSc*, and the only LSC version provided by* PETSc *does not support the latest improvements of this preconditioners in terms of boundary conditions brought in [49].*
Feel++ *relies on the* PETSc*, Portable, Extensible Toolkit for Scientific Computation [11]. It is a large suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the MPI standard for parallelism. All of its features are directly usable with the programming language C. The* PETSc *package is designed around two main concepts, namely data encapsulation and software layering. Feel++ supports the PETSc framework, the Environment takes care of initializing the associated PETSc environment.*

Contents

1 Agebraic context

Our purpose is to solve the Navier-Stokes system (2.10), (2.11) preconditioned by the PCD block preconditioner presented in chapter 4. Let us recall the algebraic form of the problem written as follows:

$$\underbrace{\begin{pmatrix} \mathbf{F_u} & \mathbf{B^T} \\ \mathbf{B} & \mathbf{0} \end{pmatrix}}_{\mathcal{A}} \begin{pmatrix} \mathbf{U^{n+1}} \\ \mathbf{P^{n+1}} \end{pmatrix} = \begin{pmatrix} \mathbf{G} \\ \mathbf{0} \end{pmatrix} \tag{8.1}$$

The preconditioned system to solve is $\mathcal{A}P^{-1}P\mathbf{x} = \mathbf{b}$, or equivalently

$$\mathcal{A}P^{-1}\mathbf{y} = \mathbf{b} \tag{8.2}$$
$$P\mathbf{x} = \mathbf{y} \tag{8.3}$$

where

$$P = P_{PCD} = \begin{pmatrix} \mathbf{F_u} & \mathbf{B^T} \\ \mathbf{0} & -\mathbf{S} \end{pmatrix}$$

with $\mathbf{S} = \mathbf{B}{\mathbf{F_u}}^{-1}\mathbf{B^T}$, the Schur complement that we replace by $\mathbf{S}* = \mathbf{Q_p}\mathbf{F_p^{-1}}\mathbf{A_p}$, with $\mathbf{A_p}$ a discrete weighted Laplacian operator for the pressure space.

The action of the preconditioner P_{PCD} on $\mathbf{x} = \begin{pmatrix} \mathbf{x_u} \\ \mathbf{x}_p \end{pmatrix}$ in equation (8.3) requires:

Algorithm 5 Application of the preconditioner

1: Solve $\mathbf{Q_p}\mathbf{F_p^{-1}}\mathbf{A_p}\mathbf{x}_p = \mathbf{y}_p$
2: Update $\mathbf{y_u} = \mathbf{y_u} - \mathbf{B^T}\mathbf{x}_p$
3: Solve $\mathbf{F_u}\mathbf{x_u} = \mathbf{y_u}$
4: **return** $\begin{pmatrix} \mathbf{x_u} \\ \mathbf{x}_p \end{pmatrix}$

The solve in the first step of Algorithm 5 requires *(i)* the inverse of the mass matrix $\mathbf{Q_p}$, *(ii)* he product of the retrieved vector with the matrix $\mathbf{F_p}$, *(iii)* applying the inverse of the stiffness $\mathbf{A_p}$ to the vector retrieved from the previous step.

This led us to develop an operator framework to handle the different operators involved in the structure of the PCD preconditioner. We henceforth are able to *(i)* define an operator, *(ii)* create the inverse operator, *(iii)* create a transpose operator, and *(iv)* to be able to compose operators. Hence, the approximation of the Shur complement can be easily written as: `S = compose( Ap, compose(inv(op(G,"Fp")),Qp) );`

2 Implementation

2.1 Backend

The Algebraic representations are handled in Feel++ using a so-called *backend* which is a wrapper class that encapsulates several algorithms as well as data structures like

vectors and matrices. It provides all the algebraic data structures behind function spaces, operators and forms. In the case of linear functionals, the representation is a vector and, in the case of linear operators and bilinear forms, the representation is a matrix. The backend abstraction allows to write code that is independent of the libraries used in the assembly process or to solve the linear systems involved, thus hiding all the details of that algebraic part under the hood of the backend. Once a backend is set, the user can create in a transparent way vectors and matrices to be used during the assembly process, as is it shown is the following listing.

```
// allocates a sparse matrix based on the degrees of freedom of the
// trial functions space X_h and the test functions space V_h
auto A = backend->newMatrix( _trial=Xh, _test=Vh );

// allocates a vector based on the degrees of freedom of the
// test functions space V_h
auto b = backend->newVector( _test=Vh );
```

There are two available backends that provide an interface to `PETSc/SLEPc`, see [11, 12, 74], and `Trilinos`, see [76, 77, 78, 75]. The user can choose any of them, bearing in mind the tools and features available in each, such as parallelism, direct or iterative linear system solvers or preconditioners for these systems. Once a backend is defined, say `Backend <Trilinos>`, the user is free to manipulate objects such as vectors and matrices (from the `Epetra` class, in this case), using most of the algebraic operations attainable from the original library. The backend also provides interfaces to linear system solvers (direct or iterative, with or without a preconditioner). The syntax for this function is illustrated in the next listing. Similar interfaces exist for non-linear and standard/generalized eigenvalue solvers.

```
// solve the linear system Au = F
backend -> solve( _matrix=A , _solution=u , _rhs=F );
```

2.2 Operator framework

The main ingredient of the operator framework is the `OperatorMatrix` class explained in Listing 8.2. This class inherits from the `OperatorBase` parent class. In the following Listing 8.1, we show a reduced version of this class, only the public interface. FEEL++ provide a interface for the creation of new operators. The principle consists of loading a parent class (here `OperatorBase`) that contains a certain number of pure virtual methods that we will define in the child class.

Listing 8.1: Define the base operators class.

```
template<typename T>
class OperatorBase
{
  public:
    typedef T value_type;
    typedef typename Backend<value_type>::solve_return_type
       ↪ solve_return_type;
    typedef DataMap datamap_type;
    typedef boost::shared_ptr<DataMap> datamap_ptrtype;
    typedef boost::shared_ptr<Preconditioner<T» pc_ptrtype;
```

```
        OperatorBase( datamap_ptrtype const& map, std::string label,
            ↪ bool use_transpose, bool has_norminf  )
        :
        M_domain_map( map ),
        M_image_map( map ),
        M_label( label ),
        M_use_transpose( use_transpose ) ,
        M_has_norminf( has_norminf )
        {}
        virtual ~OperatorBase() {};
    //other constructors are also available
    virtual void setUseTranspose(bool UseTranspose)  { M_use_transpose =
        ↪  UseTranspose; }

    virtual int apply(const vector_type& X, vector_type& Y) const = 0;
    virtual int applyInverse(const vector_type& X, vector_type& Y) const
        ↪  = 0;

     // \return true is transposed should be used, false otherwise

    virtual bool useTranspose() const { return M_use_transpose; }

    // more methods are also available in this class, we restrict the
        ↪ list to those used in the sequel
};
```

Let us first start by describing the `OperatorMatrix` class that allows to define an operator, as seen in the constructor in Listing 8.3. This class (see Listing 8.2) inherits from an `OperatorBase` class definedpreviously in Feel++, and is defined as follows:

Listing 8.2: Define the class of operators associated to matrix.

```
template<typename T>
class OperatorMatrix : public OperatorBase<T>
{
public:
   //Create operator
   OperatorMatrix(sparse_matrix_ptrtype const& F,std::string _label,
                bool transpose = 0 )
      :
      OperatorBase<T>( F->mapColPtr(), F->mapRowPtr(), _label,
            transpose, true ),
      M_F( F ),
      M_xx(backend(_name=this->label())->newVector(F->mapRowPtr())),
      M_yy(backend(_name=this->label())->newVector(F->mapColPtr()) ,
      M_hasInverse( 1 ),
      M_hasApply( 1 ),
      M_closeMatrixRhs( true )
  {
      auto b = backend(_name=this->label(),_rebuild=boption(
                _name="backend.rebuild_op",_prefix=this->label()));
  }
    //apply(X,Y)
    int apply( const vector_type& X, vector_type& Y ) const
    {
      M_F->multVector( X, Y );
      Y.close();
      return !hasApply();
    }
    // apply the inverse of an
    int applyInverse ( const vector_type& X, vector_type& Y ) const
    {
      *M_xx = X;
      M_xx->close();
```

```
        bool cv;
        if(!this->M_pc)
        {
            this->M_return = backend(_name=this->label())->solve(
                _matrix=M_F, _rhs=M_xx, _solution=M_yy,
                _close=M_closeMatrixRhs );
            cv = this->M_return.isConverged();
        }
        else
        {
            this->M_return = backend(_name=this->label())->solve(
               _matrix=M_F,_rhs=M_xx, _solution=M_yy,_prec=this->M_pc,
               _close=M_closeMatrixRhs );
            cv = this->M_return.isConverged();
        }
        Y=*M_yy;
        Y.close();
        return cv;
    }
};
```

The following constructor allows to create an operator associated to a given matrix. The Boolean passed in the arguments define whether we want to create the operator associated to the matrix or to its transpose.

Listing 8.3: Create an operator.

```
template<typename MatrixType>
boost::shared_ptr<OperatorMatrix<typename MatrixType::value_type> >
op( boost::shared_ptr<MatrixType> M, std::string label,
                                     bool transpose = false )
{
    return boost::make_shared<OperatorMatrix<typename
                  MatrixType::value_type> >(M,label,transpose) ;
}
```

The following Listing 8.4 defines the `OperatorCompose` class and its constructor. As its name indicates, this class allows to define the $(F \circ G)$ operator from the F and G operators.

Listing 8.4: Operator class to model the $(F \circ G)$ operator.

```
template< typename op1_type, typename op2_type >
class OperatorCompose : public OperatorBase<typename op1_type::
   ↪ value_type>
{
    typedef OperatorBase<typename op1_type::value_type> super;
public:

    typedef boost::shared_ptr<op1_type> op1_ptrtype;
    typedef boost::shared_ptr<op2_type> op2_ptrtype;
    typedef typename op2_type::value_type value_type;

    // This constructor implements the (F o G) operator
OperatorCompose( op1_ptrtype F, op2_ptrtype G )
        :
        super(G->domainMapPtr(),F->imageMapPtr(), F->label(),
                  F->useTransposeVector(),false),
        M_F( F ),
        M_G( G ),
        M_ZG ( backend()->newVector( M_G->imageMapPtr() ) ),
        M_ZF ( backend()->newVector( M_F->domainMapPtr() ) )
    {
        std::string t( F->label() );
        std::string u( G->label() );
```

```
        t.append( "*" );
        t.append( u );
        this->setLabel(t);
        LOG(INFO) << "Create operator " << this->label() << " ...\n";
    }

template<typename Op1Type, typename Op2Type>
boost::shared_ptr<OperatorCompose<Op1Type,Op2Type> >
compose( boost::shared_ptr<Op1Type>  op1,  boost::shared_ptr<Op2Type>
    ↪ op2  )
{
 return boost::make_shared<OperatorCompose<Op1Type,Op2Type> >(op1,op2);
}
```

Finally, Listing 8.8 corresponds to the `OperatorInverse` class that inherits from the `OperatorBase` mother class. This class allows the creation of an inverse operator.

Listing 8.5: Operator class to model an inverse operato.r

```
template< typename operator_type >
class OperatorInverse : public OperatorBase<typename operator_type::
    ↪ value_type>
{
    typedef OperatorBase<typename operator_type::value_type> super;
public:

    typedef boost::shared_ptr<operator_type> operator_ptrtype;
    typedef typename operator_type::value_type value_type;

    // This constructor implements the F^-1 operator
    OperatorInverse( operator_ptrtype& F )
        :
        super( F->imageMapPtr(), F->domainMapPtr(), F->label(),
                                 F->useTranspose(), false ),
        M_F( F )
    {
        this->setName();
    }

    OperatorInverse( const OperatorInverse& tc )
        :
        super(tc),
        Copy inverse operator
        M_F( tc.M_F )
    {}

    bool hasInverse() const { return M_F->hasApply();  }

    bool hasApply() const
    {
        return M_F->hasInverse();
    }

    int apply( const vector_type & X, vector_type & Y ) const
    {
      CHECK(hasApply())<<"Operator"<<this->label()<<"not applied";
      M_F->applyInverse( X,Y );
      return !hasApply();
    }
    //apply Inverse matrix
    int applyInverse ( const vector_type& X, vector_type& Y ) const
    {
      CHECK(hasInverse())<<"Operator"<<this->label()<<"not inverted";
      CHECK(M_F)<<"Invalid operator "<< this->label()<<"to inverse";
```

```
      M_F->apply( X,Y );

      return !hasInverse();
    }
private:

    operator_ptrtype M_F;

    void setName()
    {
      std::string L = M_F->label();
      L.append( ")" );
      std::string temp( "inv(" );
      temp.append( L );
      this->setLabel(temp);
    }
};
```

2.3 Sub-matrix extraction

Now that we have all the operator ingredients well defined, we can start with extracting the matrix blocks $\mathbf{B^T}$ and $\mathbf{F_u}$ from the global matrix $\mathcal{A}$ needed in Steps 2. and 3. of Algorithm 5, respectively. Then define the corresponding operators in a second time. These submatrices are extracted by using the Feel++ function, the so-called `createSubmatrix`, an interface for the `MatCreateSubMatrix` function of PETSc. The input parameters required for this function include the index sets for rows and columns corresponding to the submatrices in the initial matrix. The Listing 8.6 illustrates the examples of submatrices extraction in Feel++.

Listing 8.6: Extraction of submatrices.

```
// create submatrices Fu and B^T from the global matrix A
M_F = this->matrix()->createSubMatrix(M_Vh_indices,M_Vh_indices,true );
M_F->mapRowPtr()->setIndexSplit( M_Vh->dof()->indexSplit() );
M_Bt = this->matrix()->createSubMatrix( M_Vh_indices, M_Qh_indices );
// define the corresponding operators
helmOp = op( M_F, "Fu" );
divOp = op( M_Bt, "Bt");
```

Remark 26. *Note that we chose to have a special treatment to the $\mathbf{F_u}$, that is extracting the x, y and z components of the matrix and apply a special preconditioner for each component-block, that is why we can see the IF block in the previous listing.*

2.4 Block preconditioners framework

Listing 8.7 is the constructor of the class `PreconditionerBlockNS`. The first argument defines the block preconditioner type we want to use. Note that this framework handles the PCD preconditioner, the PMM preconditioner as well as the SIMPLE preconditioner. The constructor also needs to initialise the function spaces, the boundary conditions of the global problem in order to associate the corresponding boundary conditions for the $\mathbf{A}_p$ and $\mathbf{F_u}$ sub-problems (see section 4.1). It also requires the global matrix from which we are going to extract the $\mathbf{F_u}$ and $\mathbf{B}^T$ blocks, and the density and viscosity of the fluid. The *alpha* parameter is a Boolean that defines whether the problem is time dependent or independent, or in other terms if there is or not a mass matrix.

Listing 8.7: PreconditionerBlockNS constructor.

```
template<typename MuExprT, typename RhoExprT, typename AlphaExprT>
PreconditionerBlockNS( std::string t, //PCD, PMM, SIMPLE
                       space_ptrtype Xh,
                       properties_space_ptrtype Ph,
                       BoundaryConditions bcFlags,
                       std::string const& s,
                       sparse_matrix_ptrtype A,
                       MuExprT mu,
                       RhoExprT rho,
                       AlphaExprT alpha );
```

The following Listing 8.8 shows the `update` function of the `PreconditionerBlockNS` class. Besides setting the *mu*, *rho* and *alpha* parameter values, the main ingredients of this function are the call of the `createSubMatrices` function and the `update` of the PCD operator. The later is detailed in Listing 8.9.

Listing 8.8: The preconditioner update function.

```
template < typename SpaceType, typename PropertiesSpaceType >
template< typename Expr_convection, typename Expr_bc,
                  typename MuExprT, typename RhoExprT,
                  typename AlphaExprT >
void PreconditionerBlockNS<SpaceType,PropertiesSpaceType>::
update( sparse_matrix_ptrtype A, Expr_convection const& expr_b,
        Expr_bc const& g,
        MuExprT mu,
        RhoExprT rho,
        AlphaExprT alpha,
        bool hasConvection,
        double tn, double tn1 )
{
    this->setMatrix( A );
    this->createSubMatrices();

    this->setMu(mu);
    this->setRho(rho);
    this->setAlpha(alpha);

    pcdOp->update( expr_b, g, hasConvection, tn, tn1 );

}
```

Listing 8.9 is the `update` PCD operator function. In this function we update the bilinear form associated to the $\mathbf{F_p}$ problem and we set the appropriate boundary conditions for this sub-problem depending on which boundary conditions are set for the global problem (see section 4.1).
The PCD operator is then updated using the `compose`, and `inv` features of the *operator framework*.
Note that for setting the boundary conditions for the $\mathbf{F_p}$ problem we chose to act directly on the matrix (strong Dirichlet conditions). We made the choice of doing a symmetric elimination and keeping the value on the diagonal instead of replacing it with 1 so that the spectrum of the matrix doesn't change.

Listing 8.9: The update function of a PCD type operator.

```
template < typename space_type, typename PropertiesSpaceType>
template < typename ExprConvection, typename ExprBC >
void
OperatorPCD<space_type, PropertiesSpaceType>::
            update( ExprConvection const& expr_b, ExprBC const& ebc,
```

```
                          bool hasConvection, double tn,double tn1 )
{
    double time_step = M_accel?tn1-tn:1; // if time adaptation
    // OperatorPCD::update apply diffusion
    form2_conv = integrate( _range=elements(M_Qh->mesh()),
                        _expr=idv(*M_mu)*gradt(p)*trans(grad(q)));
    // OperatorPCD::update apply convection
    form2_conv += integrate( _range=elements(M_Qh->mesh()),
                       _expr=(trans(expr_b)*trans(gradt(p)))*id(q));
    // Setting Robin condition at the inlet
    for( auto dir : M_bcFlags[M_prefix]["Dirichlet"])
        if ( ebc.find( dir.marker() ) != ebc.end() )
           form2_conv += integrate(
                _range=markedfaces(M_Qh->mesh(), dir.meshMarkers()),
                _expr=-idv(*M_rho)*trans(ebc.find(dir.marker())->
                           second)*N()*idt(p)*id(q));

   // if neumann bc for velocity problem at the outlet, then apply
   // Dirichlet condition for pressure problem at the outlet
    if ( soption("blockns.pcd.outflow") == "Dirichlet" )
        for( auto cond : M_bcFlags[M_prefix]["Neumann"])
        {
            form2_conv += on( markedfaces(M_Qh->mesh(),
            cond.meshMarkers()), _element=p, _rhs=rhs, _expr=cst(0.),
            _type="elimination_keep_diagonal" );
        }
    this->applyBC(G);
    static bool init_G = false;
    // setting pcd operator
    if ( !init_G )
    {   //                  Ap       Fp^-1              Qp
       precOp = compose( diffOp, compose(inv(op(G,"Fp")),massOp) );
       init_G = true;
    }
}
```

The `diffOp` and `massOp` are defined in Listings 8.10 and 8.11, respectively. Note that for the diffusion operator construction, we can either choose the explicit definition $\mathbf{A_p} = \mathbf{BQ_u^{-1}B^T}$ (`blockns.pcd.diffusion == "BTBt"` option) or consider it as a discrete weighted Laplacian operator on the pressure space, and hence assemble the matrix (`blockns.pcd.diffusion == "Laplacian"`) option.

Listing 8.10: Assemble diffusion function.

```
template < typename space_type, typename PropertiesSpaceType>
void
OperatorPCD<space_type, PropertiesSpaceType>::assembleDiffusion()
{
    if ( soption("blockns.pcd.diffusion") == "Laplacian" )
    {
        auto d = form2( _test=M_Qh, _trial=M_Qh, _matrix=M_diff );
        d = integrate( _range=elements(M_Qh->mesh()), _expr=gradt(p)*
            trans(grad(q)));
        for( auto cond : M_bcFlags[M_prefix]["Neumann"])
        {
                d += on( markedfaces(M_Qh->mesh(),cond.meshMarkers()),
                    _element=p, _rhs=rhs,
                        _expr=cst(0.), _type="elimination_keep_diagonal
                            " );
        }
        M_diff->close();
     }
    if ( soption("blockns.pcd.diffusion") == "BTBt" )
    {
```

```
        std::vector<size_type>M_Vh_indices(M_Vh->nLocalDofWithGhost());
        std::vector<size_type>M_Qh_indices(M_Qh->nLocalDofWithGhost());
        std::iota( M_Vh_indices.begin(), M_Vh_indices.end(), 0 );
        auto m = form2( _test=M_Vh, _trial=M_Vh );
        m = integrate( elements(M_Vh->mesh()), trans(idt(u))*id(v) );
        m.matrixPtr()->close();
        auto d = M_b->newVector( M_Vh );
        M_b->diag( m.matrixPtr(), d );
        d->reciprocal();
        M_b->diag( d, M_massv_inv );
        M_massv_inv->close();
        M_b->PtAP( M_massv_inv, M_Bt, M_diff );
        M_diff->close();
        if ( Environment::numberOfProcessors() == 1 )
            M_diff->printMatlab( "BTBt.m" );
    }

    diffOp = op( M_diff, "Ap" );
}
```

Listing 8.11: Assemble diffusion function.

```
template < typename space_type, typename PropertiesSpaceType>
void
OperatorPCD<space_type, PropertiesSpaceType>::assembleMass()
{
    auto m = form2( _test=M_Qh, _trial=M_Qh, _matrix=M_mass );
    m = integrate( elements(M_Qh->mesh()), idt(p)*id(q) );
    M_mass->close();
    massOp = op( M_mass, "Mp" );
}
```

The application of the so-constructed preconditioner is done during the call of the `applyInverse` function of the `PreconditionerBlockNS` class detailed in Listing 8.12. We first apply the inverse of the Schur complement, (Step 1 of Algorithm 1), the retrieved pressure is stored in vector M_{pout}. We then apply the divergence operator $\mathbf{B^T}$ on the computed pressure M_{pout}, the resulted vector M_{vout} is stored in an auxiliary vector M_{aux} (Step 2 of Algorithm 1). We finally apply the inverse of the $\mathbf{F_u}$ matrix-operator on the M_{aux} vector to retrieve the velocity unknown vector stored in M_{vout}. We end up with updating the velocity and pressure elements.

Listing 8.12: ApplyInverse function.

```
template < typename SpaceType, typename PropertiesSpaceType >
int PreconditionerBlockNS<SpaceType,PropertiesSpaceType>::
     applyInverse ( const vector_type& X, vector_type& Y ) const
{
    U = X;
    U.close();
    *M_vin = U.template element<0>();
    M_vin->close();
    *M_pin = U.template element<1>();
    M_pin->close();
    *M_aux = *M_vin;
    M_aux->close();
// Apply S*^-1
    pcdOp->applyInverse( *M_pin, *M_pout );
    M_pout->scale(-1);
    M_pout->close();

// Apply B^T
    divOp->apply( *M_pout, *M_vout );
```

```
    M_aux->add( -1.0, *M_vout );
    M_aux->close();
// Apply Fu^-1
    helmOp->applyInverse(*M_aux, *M_vout);
// Update output velocity/pressure
    U.template element<0>() = *M_vout;
    U.template element<1>() = *M_pout;
    U.close();
    Y=U;
    Y.close();
    return 0;
}
```

In the following listing 8.13, we show an example of the PCD preconditioner call. The *blockns* constructor is called after assembling the bilinear and the linear form and is given during the call of the *solve* function under the argument *_prec*.

Listing 8.13: Example of a call for the PCD preconditioner.

```
constexpr int dim = FEELPP_DIM;
constexpr int order_p= FEELPP_ORDER_P;
auto mesh = loadMesh(_mesh=new Mesh<Simplex<dim> >);
auto Vh = THch<order_p>( mesh );
auto U = Vh->element();
auto V = Vh->element();
auto u = U.element<0>();
auto v = V.element<0>();
auto p = U.element<1>();
auto q = V.element<1>();
BoundaryConditions bcs;
//define a map for the boundary condition
map_vector_field<dim,1,2> m_dirichlet {
        bcs.getVectorFields<dim> ( "velocity", "Dirichlet" ) };
//define the bilinear form
auto at = form2( _trial=Vh, _test=Vh);
        //Assemble the bilinear and the linear forms and set BC
           ...
//define the linear form
auto l = form1( _test=Vh );
//setting up preconditioner Blockns
auto a_blockns = blockns( _space=Vh,
                          _properties_space=Pdh<0>(Vh->mesh()),
                          _type=soption("PCD"),
                          _bc=bcs, _matrix= at.matrixPtr(),
                          _prefix="velocity" );

a_blockns->update( at.matrixPtr(), zero<dim,1>(), m_dirichlet );
//solve
at.solveb(_rhs=l,_solution=U,_backend=backend(_name="stokes"),
               _prec=a_blockns);
```

2.5 Definition of the problem and configuration file

The problem physical parameters, the backends options as well as the geometry details, and the boundary conditions are handled and set in a configuration file. Listing 8.14 shows an example of a configuration file for a 2D Navier-Stokes problem using the Feel++ in-house PCD preconditioner. The Navier-Stokes problem is initialised by doing a Stokes solve, hence the Stokes backend. For this problem, a suitable preconditioner is the PMM preconditioner. As for the Navier-Stokes solve (ns backend) we again used the in-house PCD preconditioner that have the advantage of handling the new boundary conditions

improvements [49] with respect to the PETSc version of the PCD preconditioner. In this example, we used the multigrid GAMG for the $\mathbf{A_p}$ and $\mathbf{Q_p}$ subproblems. As for $\mathbf{F_u}$ we used Fieldsplit coupled with BJacobi (additive option) to extract the component of the $\mathbf{F_u}$ matrix and be able to apply a special treatment on each component problem. Here again we used GAMG. We chose a 1 level of smoothing and set to off the `gamg-set-sym-graph` option because sometimes not having a symmetric graph can accelerate the convergence. As for the `pc-gamg-threshold` option, it determines the amount of edges that are going to be dropped when constructing the aggregation graph. If it is set to 0, no edge is dropped. The higher the value of this option gets, the more edges are going to be dropped, the coarser the graph is going to be. Hence, the construction of the graph will be faster but the convergence may deteriorate.

Listing 8.14: Example of a configuration file.

```
mu=1
rho=1
alpha=1

bc-file=bcFile.bc

[functions]
g={y*(1-y),0}:y
h={0,0}

[blockns.pcd]
//CL at inflow of pressure
inflow=Robin
//CL at outflow of pressure
outflow=Dirichlet

[Ap]  //diffusion operator
reuse-prec=true
pc-type=gamg
ksp-monitor=1
pc-gamg-type=agg
pc-gamg-agg-nsmooths=1
pc-gamg-threshold=1e-4
pc-gamg-set-sym-graph=false
ksp-rtol=1e-9
ksp-maxit=300

[Fu]  //velocity convection diffusion operator
pc-type=fieldsplit
ksp-rtol=1e-9
ksp-monitor=1
fieldsplit-use-components=1
fieldsplit-type=additive

[Fu.fieldsplit-0]   //x component
pc-type=gamg
ksp-type=preonly
pc-gamg-type=agg
pc-gamg-agg-nsmooths=1
pc-gamg-threshold=1e-4
pc-gamg-set-sym-graph=false
ksp-rtol=1e-9

[Fu.fieldsplit-1]  //y component
pc-type=gamg
ksp-type=preonly
pc-gamg-type=agg
pc-gamg-agg-nsmooths=1
```

```
pc-gamg-threshold=1e-4
pc-gamg-set-sym-graph=false
ksp-rtol=1e-9

[Mp] //pressure mass matrix
reuse-prec=true
pc-type=gamg
ksp-monitor=1
pc-gamg-type=agg
pc-gamg-agg-nsmooths=1
pc-gamg-threshold=1e-4
pc-gamg-set-sym-graph=false
ksp-rtol=1e-9
ksp-maxit=300

[stokes] //stokes problem backend
preconditioner=PMM
ksp-rtol=1e-6#1e-10
ksp-monitor=1
ksp-type=gcr
ksp-use-initial-guess-nonzero=false
gcr-restart=100

[ns] //NAvier-Stokes problem backend
preconditioner=PCD
ksp-rtol=1e-6#1e-10
ksp-monitor=1
ksp-type=gcr
ksp-use-initial-guess-nonzero=false
gcr-restart=100

[gmsh]
hsize=0.1
filename=file.geo
```

Finally, the boundary conditions are handled in a *json* (JavaScript Object Notation) file described in Listing 8.15. *json* is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs.

Listing 8.15: Example of a json file defining boundary conditions.

```
{
    "velocity":
    {

        "Dirichlet":
        {
            "inlet":
            {
                "expr":"{3*16*y*(1-y)*z*(1-z)/2,0,0}:y:z"
            },
            "wall":
            {
                "expr":"{0,0,0}"
            }
        },
        "Neumann":
        {
            "outlet":
            {}
        }
    }
}
```

As we can see, the use of the configuration files and json files makes it easy for a user to

change the problem dimension, *i)* the finite elements order, *ii)* the geometry approximation order, *iii)* the geometry, *iv)* the boundary conditions expressions, as well as *v)* the physical parameters values, and *vi)* the backends options without having to modify the code and thus recompiling it. Hence, the code can be generic and easily adapted to any kind of problem. Besides, the parallelism is hidden, the user have only to precise the CPU units needed while lunching the application, and it will run on the desired number of processors.

Conclusion

As seen in Chapter 6, the Pressure Convection Diffusion preconditioner is one of the best suited block-preconditioners for the Navier-Stokes equations. As it is not implemented in PETSc *on which* Feel++ *relies for the systems resolution, we have implemented this preconditioner in* Feel++*. In this chapter we have seen the implementation details of this preconditioner, and how it can be easily called by the user. We described the operator framework implementation, the matrix extraction feature, and the block-preconditioners framework. We also emphasized the seamless parallelism, the code genericity via an example of a configuration file handling all the options, parameters and linked files.*

Conclusions and outlook

At the beginning of this thesis, the main objective was to develop numerical methods for the 3D simulation of blood flow in real vascular networks. To reach this end, we needed first to validate and verify our numerical model, in terms of i) convergence analysis with respect to different kinds of boundary conditions, and ii) comparison with respect to experimental data.
The FDA benchmark provided a well detailed test case, and experimental data were available online for CFD validation. This benchmark was however very challenging to perform. It required a high accuracy and hence a heavy cost in terms memory and run time due to the increasing size of the problem with higher Reynolds numbers. The basic iterative solvers were no longer suitable, for they do not scale with such big problems. This pushed us to investigate on a suitable preconditioning strategy that takes into consideration the physiology of the flow and can scale with a HPC.
Block-preconditioners such as PCD, LSC and SIMPLE were therefore our choice of preconditioning strategy. They were tested over the backward facing step as shown in Chapter 6. While SIMPLE did not converge because of the steady state of the problem, LSC and PCD showed an independence to the mesh size and the Reynolds number. We therefore implemented the PCD preconditioner in the Feel++ library because it is not provided by PETSc. We also made use of the high order finite element approximation — feature available in Feel++— to also prove the independency of LSC and the in-house implemented PCD preconditioners from the increase of the finite elements order, point that was never studied before in the literature. The in-house PCD preconditioner was then used to perform the transitional and turbulent regimes of the FDA benchmark. An agreement between the experimental data and the CFD output data was shown in the second section of Chapter 6. This work on the scalability analysis of the PCD and LSC preconditioner and its application to the FDA benchmark was presented during the SIMRACE 2015 conference in Paris [33]. It is the object of a submitted publication to *Computational and mathematical Biomedical Engineering* CMBE 2017.

Interesting ideas about improving the PCD preconditioner could be tested in the future.
Instead of replacing the exact Schur complement $\mathbf{S} = \mathbf{B}\mathbf{F}_{\mathbf{u}}^{-1}\mathbf{B}^{\mathbf{T}}$ in PCD preconditioner by its approximation $\mathbf{S}^* = \mathbf{Q}_p\mathbf{F}_p^{-1}\mathbf{A}_p$ to reduce the cost of inverting $\mathbf{F}_{\mathbf{u}}$, an idea could be to keep the exact Schur complement $\mathbf{S}$ and invert using $\mathbf{S}^*$ as a preconditioner. Another idea could be to precondition $\mathbf{S}^*$ using another approximation of the Schur complement approximation $\mathbf{S}^{**} = approx(\mathbf{Q}_p)\mathbf{F}_p^{-1}approx(\mathbf{A}_p)$. This second method could accelerate the convergence of the preconditioner.The advantage of this two new approaches could be to use a low order approximations $\mathbf{S}^*$ or $\mathbf{S}^{**}$ to precondition a high order exact Schur complement.

As for the convergence analysis with respect to different kind of boundary conditions, the results shown in Chapter 5 were published in the proceeding of the CEMRACS 2012 [30]. More physiological boundary conditions needs to be also studied in the future, especially the 3D-1D or 3D-0D coupling [56, 57, 137, 132, 58, 149, 39, 126, 128, 112, 93, 114].

Concerning the approximation of the stress tensor, to the best of our knowledge, no estimates on the high order geometry approximation are available in the literature. A theoretical investigation must be carried on to confirm or reject the numerical results

presented in Chapter 5. A natural outlook is to test the volumic method on the coupling of the fluid and the structure problems in a FSI simulation.

In a different context, I participated during the CEMRACS 2015, with my colleagues in the ANR project VIVABRAIN, to the validation of a pipeline that goes from real MRI data acquisition to simulated MRI. The validation process rely on i) accessing the reliability of the segmentation techniques: compare the realistic geometry, knowing exactly its shape, with MRI segmentations, obtained by a simple threshold and by the classical method of snake, ii) accessing the reliability of the CFD simulation by first comparing the outputs of two identical simulations using Feel++ in the first and Freefem++ in the second, then comparing the outputs of the Feel++ and Freefem++ simulations to the MRI measurements on an idealised device (the phantom device) and on the cerebrovenous network, iii) accessing the reliability of the MRI simulation by comparing the simulated phase and magnitude images to the realistic geometry on one hand and to the MRI acquisitions on the other hand.
The MRI–realistic geometry comparison shows qualitative differences between real and MRI-measured diameters of each part of the phantom. This study emphasizes the necessity of segmentation or threshold procedure improvement for a physiological use.
The Feel++/ FreeFem++ comparison in the phantom shows a good agreement between the numerical libraries outputs. Such a comparison allows us to confirm the good correlation between the solution regarding to a physiological use. While a Feel++/ FreeFem++/ PC-MRI comparison shows a sufficiently fair agreement in all the phantom, a dissimilar distribution between the two libraries of the flow in the branches is however observed and deserves to be investigated. In the realistic geometry of the cerebral venous network, a good agreement is equally found between Feel++ and FreeFem++ results. We are however conscious that the boundary conditions used, although the closer to the physiological reality, don't reproduce the desired flow behaviour, the mismatch with the experimental flow rate being an argument in this direction.
As for the JEMRIS / Feel++ FreeFem++ comparison with constant flow showed a good qualitative agreement for the velocity profiles, but an underestimation of peak velocities. This can be easily corrected with more particles trajectories and higher image resolution in the virtual MRI simulations.
Few more points need to be checked in order to complete all the validation process. A comparison of the output data of the CFD simulation in the exact geometry need to be compared with the output of the CFD simulation in the 3D mesh retrieved from the segmented geometry obtained from the MRI acquisition of the physical phantom. Another track that can be interesting to investigate on is the accuracy of the MRI acquisition, task that was not studied in our validation process. In fact, in order for our work to be relevant, it is important to establish an uncertainty quantification of our pipeline framework. This point is particularly relevant to the physical phantom since in this case we possess exact data in terms of geometry details and physical flow magnitudes. A special care needs to be given while acquiring experimental data

Going back to the main objective of this thesis, one of the numerical methods that we were interested in developing was, in the continuity of the work of N. Poussineau [110], a reduction method for the 3D Navier-Stokes equations, aiming at decreasing the heavy cost that usually arise when trying to perform a 3D simulation of the circulatory system

that takes into account the interaction between the fluid and the structure. We had a lot of ideas that we didn't have the time to check due to lack of time. The previously detailed issues took more time than expected.
The idea behind this reduction method is that, instead of performing a 3D simulation in all the computational domain, one may truncate the computational domain and couple a 3D region of interest (presenting a pathology) with a reduced model whose computational cost is accessible. The level of simplification can be easily selected. This method uses the variational formulation, thus the reduction concerns both unknown and test functions. Given the geometry of an artery, we can hope that the variations in the direction of the flow would be greater than in the radial direction. For this, the idea is to consider a coordinate system whose first components are in the direction of a radial section orthogonal to the axis of the artery and the third component would be following this same axis. (See Figure 8.1).

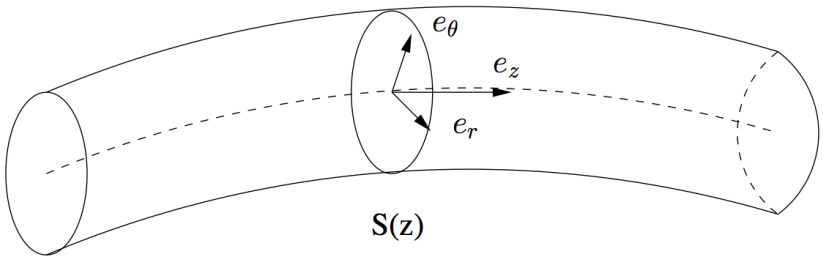


Figure 8.1: The coordinate system adjusted with respect to the centreline.[110]

Reducing the fluid model is assuming that the velocity and the pressure on each section $S(z)$ follow some known profiles. Thus, the variational space of the unknowns and the test functions will be reduced.
For instance, let us assume that the fluid follows a Poiseuille flow. On each of the radial sections the velocity could then be written as:

$$\mathbf{u}(x,y,z) = u_1(z)\phi_1(x,y,z) + u_2(z)\phi_2(x,y,z)$$

with $\phi_1(x,y,z) = \begin{pmatrix} \frac{x}{R} \\ \frac{y}{R} \\ 0 \end{pmatrix}$ and $\phi_2(x,y,z) = \begin{pmatrix} 0 \\ 0 \\ 1 - \frac{x^2+y^2}{R^2} \end{pmatrix}$.
Considering that the pressure could be written as $p(x,y,z) = p_1(z)\eta_1(x,y,z)$ with $\eta_1(x,y,z) = 1$ we will then be led to search for two scalar unknowns for the velocity and for one scalar unknown for the pressure on each section, and hence reducing the global number of unknowns with respect to a 3D model.
After reproducing the results of N. Poussineau in [110], the idea that we were last working on was to use a reduce basis framework to predict offline the velocity and pressure profiles on the radial section, and then perform the mono-dimensional model using the so retrieved profiles to enrich the velocity and pressure basis of the reduced problem. With this strategy more complex geometries could be used in the framework introduced in [110] whose limitations were that in complex geometries it is difficult to predict the velocity profile.

Bibliography

[1] http://vivabrain.fr.

[2] https://www.ensight.com/ensight-hpc/.

[3] http://www.cemosis.fr.

[4] M. F. Adams and J. Demmel. Parallel multigrid solver algorithms and implementations for 3D unstructured finite element problem. In *ACM/IEEE Proceedings of SC99: High Performance Networking and Computing*, Portland, Oregon, November 1999.

[5] Grégoire Allaire and Sidi Mahmoud Kaber. *Algebre linéaire numérique*, volume 512. Ellipses, 2002.

[6] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher. Computational fluid mechanics and heat transfer. 1984.

[7] S. Appanaboyina, F. Mut, R. Löhner, C. M. Putman, and J. R. Cebral. Computational fluid dynamics of stented intracranial aneurysms using adaptive embedded unstructured grids. *International Journal for Numerical Methods in Fluids*, 57(5):475–493, 2008.

[8] B. F. Armaly, F. Durst, J. C. F. Pereira, and B. Schönung. Experimental and theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics*, 127:473–496, 1983.

[9] BF Armaly, A Li, and JH Nie. Three-dimensional forced convection flow adjacent to backward-facing step. *Journal of thermophysics and heat transfer*, 16(2):222–227, 2002.

[10] A. Arvand, M. Hormes, and H. Reul. A validated computational fluid dynamics model to estimate hemolysis in a rotary blood pump. *Artificial organs*, 29(7):531–540, 2005.

[11] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[12] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

[13] Dwight Barkley, M Gabriela M Gomes, and Ronald D Henderson. Three-dimensional instability in flow over a backward-facing step. *Journal of Fluid Mechanics*, 473:167–190, 2002.

[14] C. Bauer, A. Frink, and R. Kreckel. Introduction to the ginac framework for symbolic computation within the c++ programming language. *J. Symbolic Computation*, 33:2002, 2002.

[15] M. Behbahani, M. Behr, M. Hormes, U. Steinseifer, D. Arora, O. Coronado, and M. Pasquali. A review of computational fluid dynamics analysis of blood pumps. *European Journal of Applied Mathematics*, 20(04):363–397, 2009.

[16] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418 – 477, 2002.

[17] L Bertagna, A Quaini, and A Veneziani. Deconvolution-based nonlinear filtering for incompressible flows at moderately large reynolds numbers. *International Journal for Numerical Methods in Fluids*, 2015.

[18] S. Bertoluzza, M. Ismail, V. Chabannes, and C. Prud'homme. Saddle point formulation for the fat boundary method : fluid structure interaction case. 2013.

[19] S. Bertoluzza, M. Ismail, V. Chabannes, and C. Prud'homme. Saddle point formulation for the fat boundary method : laplacian case. 2013.

[20] S. Bertoluzza, M. Ismail, V. Chabannes, and C. Prud'homme. Saddle point formulation for the fat boundary method : Stokes case. 2013.

[21] G Biswas, M Breuer, and F Durst. Backward-facing step flows for various expansion ratios at low and moderate reynolds numbers. *Journal of fluids engineering*, 126(3):362–374, 2004.

[22] HM Blackburn, Dwight Barkley, and Spencer J Sherwin. Convective instability and transient growth in flow over a backward-facing step. *Journal of Fluid Mechanics*, 603:271–304, 2008.

[23] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (amg) for automatic multigrid solutions with application to geodetic computations. *Report, Inst. for computational Studies, Fort collins, colo*, 1982.

[24] F. Brezzi and M. Fortin. *Mixed and hybrid finite elements methods*. Springer series in computational mathematics. Springer-Verlag, 1991.

[25] G. W. Burgreen, J. F. Antaki, Z. J. Wu, and A. J. Holmes. Computational fluid dynamics as a development tool for rotary blood pumps. *Artificial Organs*, 25(5):336–340, 2001.

[26] C. Caldini Queiros, V. Chabannes, M. Ismail, G. Pena, C. Prud'homme, M. Szopos, and R. Tarabay. Towards large-scale three-dimensional blood flow simulations in realistic geometries. *ESAIM: Proceedings*, 43:195–212, December 2013. CEMRACS 2012.

[27] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods - Fundamentals in Single Domains*. Springer-Verlag, Berlin Heidelberg, 2006.

[28] V. Chabannes. *Towards Blood Flow Simulation*. Thesis, Université de Grenoble, July 2013.

[29] V. Chabannes, C. Daversin, V. Huber, C. Prudhomme, and C. Trophime. Recent advances in feel++: a framework and language for solving partial differential equations in c++. In *ESAIM: Proceedings*, 2016.

[30] V. Chabannes, G. Pena, and C. Prud'homme. High-order fluid-structure interaction in 2D and 3D: Application to blood flow in arteries. *Journal of Computational and Applied Mathematics*, 2012.

[31] V. Chabannes, C. Prud'homme, and G. Pena. High order fluid structure interaction in 2D and 3D: Application to blood flow in arteries. *Journal of Computational and Applied Mathematics (Accepted)*, 2012.

[32] Vincent Chabannes, Mourad Ismail, Christophe Prud'Homme, and Marcela Szopos. Hemodynamic simulations in the cerebral venous network: A study on the influence of different modeling assumptions. *Journal of Coupled Systems and Multiscale Dynamics*, 3(1):23–37, 2015.

[33] Vincent Chabannes, Christophe Prud'Homme, Marcela Szopos, and Ranine Tarabay. Numerical methods and high performance computing for industrial fluid flows efficient solving strategies for incompressible navier-stokes equations for large scale simulations using the open source software feel++. In *SimRace-2015*, 2015.

[34] T. P. Chiang and T. W. H. Sheu. Time evolution of laminar flow over a three-dimensional backward-facing step. *International journal for numerical methods in fluids*, 31(4):721–745, 1999.

[35] P. G. Ciarlet. *The finite element method for elliptic problems*, volume 40. Siam, 2002.

[36] C. CONCA, F. Murat, and O. Pironneau. The stokes and navier-stokes equations with boundary conditions involving the pressure. *Japanese journal of mathematics. New series*, 20(2):279–318, 1994.

[37] Feel++ Consortium. Feel++ online documentation. http://feelpp.github.io/feelpp/.

[38] Marcela A Cruchaga. A study of the backward-facing step problem using a generalized streamline formulation. *Communications in Numerical Methods in Engineering*, 14(8):697–708, 1998.

[39] A. Curcio, M. E. Clark, M. Zhao, and W. Ruan. A hyperbolic system of equations of blood flow in an arterial network. *SIAM Journal on Applied Mathematics*, 64(2):637–667, 2004.

[40] C. Daversin, S. Veys, C. Trophime, and C. Prud'homme. A Reduced Basis Framework: Application to large scale non-linear multi-physics problems. pp 26, Accepted in ESAIM: Proc, CEMRACS 2012, 2012.

[41] D. De Wachter and P. Verdonck. Numerical calculation of hemolysis levels in peripheral hemodialysis cannulas. *Artificial organs*, 26(7):576–582, 2002.

[42] Simone Deparis, Gwenol Grandperrin, and Alfio Quarteroni. Parallel preconditioners for the unsteady navier–stokes equations and applications to hemodynamics simulations. *Computers & Fluids*, 92:253–273, 2014.

[43] V. Doyeux, V. Chabannes, C. Prud'homme, and M. Ismail. Simulation of two fluid flow using a level set method application to bubbles and vesicle dynamics. *Journal of Computational and Applied Mathematics (Accepted)*, 2012.

[44] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. Clarendon press Oxford, 1986.

[45] A. Dufour, C. Ronse, J. Baruthio, O. Tankyevych, H. Talbot, and N. Passat. Morphology-based cerebrovascular atlas. In *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*, pages 1210–1214. IEEE, 2013.

[46] S.C. Eisenstat, H.C. Elman, and M.H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, 1983.

[47] 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.

[48] H. Elman, D. Silverster, and A. Wathen. *Finite elements and fast iterative solvers with applications in incompressible fluid dynamics*. Oxford science publications, Oxford, 2014.

[49] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers with applications in incompressible fluid dynamics*. Oxford University Press, Oxford, 2005.

[50] A. Ern and JL Guermond. *Theory and practice of finite elements*, volume 159. Springer Science & Business Media, 2013.

[51] E. Erturk. Numerical solutions of 2-d steady incompressible flow over a backward-facing step, part i: High reynolds number solutions. *Computers & Fluids*, 37(6):633–655, 2008.

[52] Jacob B. et al. Eigen web site. http://eigen.tuxfamily.org/.

[53] A. M. Fallon, L. P. Dasi, U. M. Marzec, S. R. Hanson, and A. P. Yoganathan. Procoagulant properties of flow fields in stenotic and expansive orifices. *Annals of biomedical engineering*, 36(1):1–13, 2008.

[54] G. B. Fiore, U. Morbiducci, R. Ponzini, and A. Redaelli. Bubble tracking through computational fluid dynamics in arterial line filters for cardiopulmonary bypass. *ASAIO Journal*, 55(5):438–444, 2009.

[55] Roger Fletcher. Conjugate gradient methods for indefinite systems. In *Numerical analysis*, pages 73–89. Springer, 1976.

[56] L. Formaggia, J-F Gerbeau, F. Nobile, and A. 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.

[57] L. Formaggia, J-F Gerbeau, F. Nobile, and A. Quarteroni. Numerical treatment of defective boundary conditions for the navier–stokes equations. *SIAM Journal on Numerical Analysis*, 40(1):376–401, 2002.

[58] L. Formaggia, F. Nobile, A. Quarteroni, and A. Veneziani. Multiscale modelling of the circulatory system: a preliminary analysis. *Computing and visualization in science*, 2(2-3):75–83, 1999.

[59] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*, volume 1. Springer Science & Business Media, 2010.

[60] J. Fouchet-Incaux. Artificial boundaries and formulations for the incompressible navier–stokes equations: applications to air and blood flows. *SeMA Journal*, 64(1):1–40, 2014.

[61] I T Gabe. Arterial blood flow by analogue solution of the navier-stokes equation. *Physics in Medicine and Biology*, 10(2):271, 1965.

[62] G. P. Galdi, R. Rannacher, A. M. Robertson, and S. Turek. Hemodynamical flows. *Delhi Book Store*, 2008.

[63] A. George and J. W. Liu. Computer solution of large sparse positive definite. 1981.

[64] 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.

[65] M. B. Giles, M. Larson, M. Levenstam, and E. Suli. Adaptive error control for finite element approximations of the lift and drag coefficients in viscous flow. 1997.

[66] D. Givoli and J. B. Keller. A finite element method for large domains. *Computer Methods in Applied Mechanics and Engineering*, 76(1):41–66, 1989.

[67] P. D. Goodman, E. T. Barlow, P. M. Crapo, S. F. Mohammad, and K. A. Solen. Computational model of device-induced thrombosis and thromboembolism. *Annals of biomedical engineering*, 33(6):780–797, 2005.

[68] P. M. Gresho and R. L. Sani. Incompressible flow and the finite element method. volume 1: Advection-diffusion and isothermal laminar flow. 1998.

[69] M. Grigioni, C. Daniele, G. D'Avenio, and V. Barbaro. A discussion on the threshold limit for hemolysis related to reynolds shear stress. *Journal of biomechanics*, 32(10):1107–1112, 1999.

[70] Wolfgang Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[71] P. Hariharan, M. Giarra, V. Reddy, S. W. Day, K. B. Manning, S. Deutsch, S. F. C. Stewart, M. R. Myers, M. R. Berman, G. W. Burgreen, E. G. Paterson, and R. A. Malinauskas. Multilaboratory particle image velocimetry analysis of the fda benchmark nozzle model to support validation of computational fluid dynamics simulations. *Journal of Biomechanical Engineering*, 133, 02 2011.

[72] T. Hassan, M. Ezura, E. V. Timofeev, T. Tominaga, T. Saito, A. Takahashi, K. Takayama, and T. Yoshimoto. Computational simulation of therapeutic parent artery occlusion to treat giant vertebrobasilar aneurysm. *American journal of neuroradiology*, 25(1):63–68, 2004.

[73] A. Henderson, J. Ahrens, and C. Law. *The ParaView Guide*. Kitware Clifton Park, NY, 2004.

[74] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3):351–362, 2005.

[75] Michael Heroux, Roscoe Bartlett, Vicki Howle Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[76] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.

[77] Michael A. Heroux and James M. Willenbring. Trilinos Users Guide. Technical Report SAND2003-2952, Sandia National Laboratories, 2003.

[78] Michael A. Heroux, James M. Willenbring, and Robert Heaphy. Trilinos Developers Guide. Technical Report SAND2003-1898, Sandia National Laboratories, 2003.

[79] Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS, 1952.

[80] T. J-R Hughes. Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer methods in applied mechanics and engineering*, 127(1):387–401, 1995.

[81] K. Hwang, A. Ramachandran, and R. Purushothaman. *Advanced computer architecture: parallelism, scalability, programmability*, volume 199. McGraw-Hill New York, 1993.

[82] Gábor Janiga. Large eddy simulation of the fda benchmark nozzle for a reynolds number of 6500. *Computers in biology and medicine*, 47:113–119, 2014.

[83] Volker John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3d navier–stokes equations. *International Journal for Numerical Methods in Fluids*, 40(6):775–798, 2002.

[84] Volker John. Reference values for drag and lift of a two-dimensional time-dependent flow around a cylinder. *International Journal for Numerical Methods in Fluids*, 44(7):777–788, 2004.

[85] Volker John and Gunar 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.

[86] Lambros Kaiktsis, George Em Karniadakis, and Steven A Orszag. Onset of three-dimensionality, equilibria, and early transition in flow over a backward-facing step. *Journal of Fluid Mechanics*, 231:501–528, 1991.

[87] Lambros Kaiktsis, George Em Karniadakis, and Steven A Orszag. Unsteadiness and convective instabilities in two-dimensional flow over a backward-facing step. *Journal of Fluid Mechanics*, 321:157–187, 1996.

[88] G. Karniadakis and S. J. Sherwin. *Spectral/hp element methods for CFD*. Oxford University Press, USA, 1999.

[89] G. Karypis and V. Kumar. Metis ver. 4.0: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *Department of Computer Science, University of Minnesota*, 1995.

[90] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state navier–stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, 2002.

[91] S. Koranne. Boost c++ libraries. In *Handbook of Open Source Tools*, pages 127–143. Springer, 2011.

[92] L. Kovasznay. Laminar flow behind a two-dimensional grid. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 44, pages 58–62. Cambridge Univ Press, 1948.

[93] K. Lagana, G. Dubini, F. Migliavacca, R. Pietrabissa, G. Pennati, A. Veneziani, and A. Quarteroni. Multiscale modelling as a tool to prescribe realistic boundary conditions for the study of surgical procedures. *Biorheology*, 39(3, 4):359–364, 2002.

[94] Jean Leray. Etude de diverses équations intégrales non linéaires et de quelques problèmes que pose l'hydrodynamique. *Thèses françaises de l'entre-deux-guerres*, 142:1–82, 1933.

[95] Jörg Liesen and Zdenek Strakos. *Krylov subspace methods: principles and analysis*. OUP Oxford, 2012.

[96] Jacques-Louis Lions, Jacques-Louis Lions, Jacques-Louis Lions, and Jacques-Louis Lions. *Quelques méthodes de résolution des problemes aux limites non linéaires*, volume 31. Dunod Paris, 1969.

[97] A. L. Marsden, I. E. Vignon-Clementel, F. P. Chan, J. A. Feinstein, and C. A. Taylor. Effects of exercise and respiration on hemodynamic efficiency in cfd simulations of the total cavopulmonary connection. *Annals of biomedical engineering*, 35(2):250–263, 2007.

[98] B. Maury. *The Respiratory System in Equations*. Springer, 2013.

[99] Guido Nabh. *On high order methods for the stationary incompressible Navier-Stokes equations*. Interdisziplinäres Zentrum für Wiss. Rechnen der Univ. Heidelberg, 1998.

[100] JH Nie and Bassem F Armaly. Three-dimensional convective flow adjacent to backward-facing step-effects of step height. *International journal of heat and mass transfer*, 45(12):2431–2438, 2002.

[101] M. Nobili, U. Morbiducci, R. Ponzini, C. Del Gaudio, A. Balducci, M. Grigioni, F. M. Montevecchi, and A. Redaelli. Numerical simulation of the dynamics of a bileaflet prosthetic heart valve using a fluid–structure interaction approach. *Journal of biomechanics*, 41(11):2539–2550, 2008.

[102] S. Pant, N. W. Bressloff, A. I. J. Forrester, and N. Curzen. The influence of strut-connectors in stented vessels: a comparison of pulsatile flow through five coronary stents. *Annals of biomedical engineering*, 38(5):1893–1907, 2010.

[103] F. Pascal. On combining finite element methods and finite volume methods in computational fluid dynamics. *Progress in analysis*, 1:1205–1213, 2001.

[104] F. Pascal and J-M Ghidaglia. Footbridge between finite volumes and finite elements with applications to cfd. *International journal for numerical methods in fluids*, 37(8):951–986, 2001.

[105] T. Passerini, A. Quaini, U. Villa, A. Veneziani, and S. Canic. Validation of an open source framework for the simulation of blood flow in rigid and deformable vessels. *International Journal for Numerical Methods in Biomedical Engineering*, 29:1192 – 1213, 2013.

[106] S. Patankar. *Numerical heat transfer and fluid flow*. CRC Press, 1980.

[107] G. Pena. *Spectral element approximation of the incompressible Navier-Stokes equations in a moving domain and applications*. Theses, Ecole Polytechnique Fédérale de Lausanne (EPFL), October 2009.

[108] G. Pena, C. Prud'homme, and A. Quarteroni. High Order Methods for the Approximation of the Incompressible Navier-Stokes Equations in a Moving Domain. *Computer Methods in Applied Mechanics and Engineering*, 209-212:197–211, 2012.

[109] M. Pinotti and E. S. Rosa. Computational prediction of hemolysis in a centrifugal ventricular assist device. *Artificial Organs*, 19(3):267–273, 1995.

[110] N. Poussineau. *Réduction variationnelle d'un couplage fluide-structure. Application à l'hémodynamique*. PhD thesis, Université Paris 6, 2007.

[111] C. Prud'homme, V. Chabannes, V. Doyeux, M. Ismail, A. Samake, and G. Pena. Feel++: A computational framework for galerkin methods and advanced numerical methods. *ESAIM Proc.*, 2012.

[112] A. Quarteroni, S. Ragni, and A. Veneziani. Coupling between lumped and distributed models for blood flow problems. *Computing and Visualization in Science*, 4(2):111–124, 2001.

[113] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical mathematics and scientific computation. Clarendon Press, 1999.

[114] A. Quarteroni and A. Veneziani. Analysis of a geometrical multiscale model based on the coupling of ode and pde for blood flow simulations. *Multiscale Modeling & Simulation*, 1(2):173–195, 2003.

[115] Alfio Maria Quarteroni, Riccardo Sacco, and Fausto Saleri. *Méthodes numériques: algorithmes, analyse et applications*. Springer Science & Business Media, 2008.

[116] J. Ruge. Algebraic multigrid (amg) for geodetic survey problems. In *Prelimary Proc. Internat. Multigrid Conference, Fort Collins, CO*, 1983.

[117] J. Ruge and K. Stüben. *Efficient solution of finite difference and finite element equations by algebraic multigrid AMG*. Gesellschaft f. Mathematik u. Datenverarbeitung, 1984.

[118] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2003.

[119] Youcef Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[120] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986.

[121] S. Salmon, S. Sy, and M. Szopos. Cerebral blood flow simulations in realistic geometries. *ESAIM: Proc.*, 35:281–286, 2012.

[122] A. Samake, S. Bertoluzza, M. Pennacchio, C. Prud'homme, and C. Zaza. A Parallel Implementation of the Mortar Element Method in 2D and 3D. pp 12, Accepted in ESAIM: Proc, CEMRACS 2012, 2012.

[123] A. Samake, V. Chabannes, C. Picard, and C. Prud'homme. Domain decomposition methods in Feel++. pp 8, Accepted in DD21 proceedings, 2012.

[124] F Schäfer, M Breuer, and F Durst. The dynamics of the transitional flow over a backward-facing step. *Journal of Fluid Mechanics*, 623:85–119, 2009.

[125] E. Schenone, S. Veys, and C. Prud'homme. High Performance Computing for the Reduced Basis Method. Application to Natural Convection. pp 19, Accepted in ESAIM: Proc, CEMRACS 2012, 2012.

[126] S. J. Sherwin, L. Formaggia, J. Peiro, and V. Franke. Computational modelling of 1d blood flow with variable mechanical properties and its application to the simulation of wave propagation in the human arterial system. *International Journal for Numerical Methods in Fluids*, 43(6-7):673–700, 2003.

[127] SJ Sherwin and G. E. Karniadakis. Spectral/hp element methods for cfd, 1999.

[128] E. B. Shim, R. D. Kamm, T. Heldt, and R. G. Mark. Numerical analysis of blood flow through a stenosed artery using a coupled multiscale simulation method. In *Computers in Cardiology 2000*, pages 219–222. IEEE, 2000.

[129] D. Silvester, H. Elman, D. Kay, and A. Wathen. Efficient preconditioning of the linearized navier–stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics*, 128(1):261–279, 2001.

[130] E. N. Sorensen, G. W. Burgreen, W. R. Wagner, and J. F. Antaki. Computational simulation of platelet deposition and activation: I. model development and properties. *Annals of biomedical engineering*, 27(4):436–448, 1999.

[131] Rolf Stenberg and Manil Suri. Mixed hp finite element methods for problems in elasticity and stokes flow. *Numerische Mathematik*, 72(3):367–389, 1996.

[132] N. Stergiopulos, D. F. Young, and T. R. Rogge. Computer simulation of arterial flow with applications to arterial and aortic stenoses. *Journal of biomechanics*, 25(12):1477–1488, 1992.

[133] S. F. C. Stewart, E. G. Paterson, G. W. Burgreen, P. Hariharan, M. Giarra, V. Reddy, S. W. Day, K. B. Manning, S Deutsch, M. R. Berman, M. R. Myers, and R. A. Malinauskas. Assessment of cfd performance in simulations of an idealized medical device: Results of fda's first computational interlaboratory study. *Cardiovascular Engineering and Technology*, 3(2):139–160, 2012.

[134] Marcela Szopos, Nicole Poussineau, Yvon Maday, Carla Canniffe, David S Celermajer, Damien Bonnet, and Phalla Ou. Computational modeling of blood flow in the aorta,Äîinsights into eccentric dilatation of the ascending aorta after surgery for coarctation. *The Journal of thoracic and cardiovascular surgery*, 148(4):1572–1582, 2014.

[135] M. Tabata and K. Itakura. A precise computation of drag coefficients of a sphere. *International Journal of Computational Fluid Dynamics*, 9(3-4):303–311, 1998.

[136] M. Tamagawa, H. Kaneda, M. Hiramoto, and S. Nagahama. Simulation of thrombus formation in shear flows using lattice boltzmann method. *Artificial organs*, 33(8):604–610, 2009.

[137] C. A. Taylor, M. T. Draney, J. P. Ku, D. Parker, B. Steele, K. Wang, and C. K. Zarins. Predictive medicine: computational techniques in therapeutic decision-making. *Computer aided surgery*, 4(5):231–247, 1999.

[138] C. A. Taylor, T. J-R Hughes, and C. K. Zarins. Finite element modeling of blood flow in arteries. *Computer methods in applied mechanics and engineering*, 158(1):155–196, 1998.

[139] C. A. Taylor and D. A. Steinman. Image-based modeling of blood flow and vessel wall dynamics: applications, methods and future directions. *Annals of biomedical engineering*, 38(3):1188–1203, 2010.

[140] Roger Temam. *Navier-Stokes equations: theory and numerical analysis*, volume 343. American Mathematical Soc., 2001.

[141] T. E. Tezduyar, S. Mittal, SE Ray, and R. Shih. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering*, 95(2):221–242, 1992.

[142] U Trottenberg and T Clees. Multigrid software for industrial applications-from mg00 to samg. In *100 Volumes of ‚ÄòNotes on Numerical Fluid Mechanics,Äô*, pages 423–436. Springer, 2009.

[143] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic press, 2000.

[144] V V. Girault and PA Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media, 2012.

[145] Henk A Van der Vorst and Cornelis Vuik. Gmresr: a family of nested gmres methods. *Numerical linear algebra with applications*, 1(4):369–386, 1994.

[146] H. K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education, 2007.

[147] I. E. Vignon-Clementel, C. A. Figueroa, K. E. Jansen, and C. A. Taylor. Outflow boundary conditions for three-dimensional finite element modeling of blood flow and pressure in arteries. *Computer methods in applied mechanics and engineering*, 195(29):3776–3796, 2006.

[148] J. Volker, M. Tabata, and L. Tobiska. *Error Estimates for Nonconforming Finite Element Approximations of Drag and Lift in Channel Flows*. Citeseer, 1998.

[149] J. Wan, B. Steele, S. A Spicer, S. Strohband, G. R. Feijo´ o, T. J-R Hughes, and C. A. Taylor. A one-dimensional finite element method for simulation-based medical planning for cardiovascular disease. *Computer Methods in Biomechanics & Biomedical Engineering*, 5(3):195–206, 2002.

[150] K. K. Whitehead, K. Pekkan, H. D. Kitajima, S. M. Paridon, A. P. Yoganathan, and M. A. Fogel. Nonlinear power loss during exercise in single-ventricle patients after the fontan insights from computational fluid dynamics. *Circulation*, 116(11 suppl):I–165, 2007.

[151] P. T. Williams and A. J. Baker. Numerical simulations of laminar flow over a 3d backward-facing step. *International Journal for Numerical Methods in Fluids*, 24(11):1159–1183, 1997.

[152] D. M. Wootton, C. P. Markou, S. R. Hanson, and D. N. Ku. A mechanistic model of acute platelet accumulation in thrombogenic stenoses. *Annals of biomedical engineering*, 29(4):321–329, 2001.

[153] J. Wu, B. E. Paden, H. S. Borovetz, and J. F. Antaki. Computational fluid dynamics analysis of blade tip clearances on hemodynamic performance and blood damage in a centrifugal ventricular assist device. *Artificial organs*, 34(5):402–411, 2010.

[154] T. Yano, K. Sekine, A. Mitoh, Y. Mitamura, E. Okamoto, D. W. Kim, I. Nishimura, S. Murabayashi, and R. Yozu. An estimation method of hemolysis within an axial flow blood pump by computational fluid dynamics analysis. *Artificial organs*, 27(10):920–925, 2003.