Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Umberto Villa Date

Scalable Efficient Methods for Incompressible Fluid-dynamics
in Engineering Problems

by

Umberto Villa
Doctor of Philosophy

Mathematics and Computer Science

Alessandro Veneziani, Ph.D.
Advisor

Michele Benzi, Ph.D.
Committee Member

James Nagy, Ph.D.
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Scalable Efficient Methods for Incompressible Fluid-dynamics in Engineering Problems

By

Umberto Villa
M.S. in Mathematical Engineering, Politecnico di Milano, 2007
B.S. in Mathematical Engineering, Politecnico di Milano, 2005

Advisor: Alessandro Veneziani, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics and Computer Science
2012

Abstract

Scalable Efficient Methods for Incompressible Fluid-dynamics in Engineering Problems
By Umberto Villa

Accurate and effective methods for the numerical solution of incompressible fluid dynamics is an old but still important challenging problem, as more and more complex problems in engineering biology, ecology, medicine, sport are tackled with computational methods.

In this thesis, we investigate efficient solvers for two important models that governs the motion of a fluid, the *incompressible Navier-Stokes* and the *Brinkman* equations. The former describes the motion of an incompressible fluid in either an open or closed domain. The latter is used for describing the dynamics in a matrix of an inhomogeneous porous media, alternating bubbles and open channels.

For the solution of the unsteady Navier-Stokes Equations, we move from the pressure correction algebraic factorization formerly proposed by Saleri, Veneziani (2005), and we introduce the incremental formulation of pressure corrected schemes. These schemes feature an intrinsic hierarchical nature, such that an accurate approximation of the pressure Schur complement is obtained by computing intermediate low-order guesses. When used as a splitting method instead of a preconditioner, the difference between the pressure at two successive correction steps provides a natural a-posteriori estimator with no additional computational cost. We consider the basics settings of the method and its more stable variants; we also discuss implementation details that make the method competitive for real interest problems.

For the solution of the Brinkman Equations, we follow the approach presented in Mardal, Winther (2011) to precondition symmetric saddle point problems in a Hilbert settings. More specifically, we first present a novel mixed formulation of the Brinkman problem, with improved stability properties, in which we introduce the flow's vorticity as additional unknown. Based on stability analysis of the problem in the $H(\text{curl}) - H(\text{div}) - L^2$ norms, we derive a scalable block diagonal preconditioner which is optimal in the constant coefficient case.

Algorithms and preconditioners analysed in this thesis have been implemented in a parallel C++ code, using the finite element libraries LifeV and MFEM, and the linear algebra libraries Trilinos and HYPRE. We emphasize the performance of the proposed algorithms in solving problems of practical interest, involving complex geometries and realistic flow conditions. Numerical experiments in 2D and 3D confirm the effectiveness of our approach showing good efficiency and parallel scalability properties of the solvers proposed.

Scalable Efficient Methods for Incompressible Fluid-dynamics in Engineering Problems

By

Umberto Villa
M.S. in Mathematical Engineering, Politecnico di Milano, 2007
B.S. in Mathematical Engineering, Politecnico di Milano, 2005

Advisor: Alessandro Veneziani, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics and Computer Science
2012

Acknowledgements

I begin by thanking my advisor, Prof. **Alessandro Veneziani**, for leading me and the rest of the Italian crew towards the conquering of a "*brave new* (scientific research) *world*". As we usually say in Italy "*non c'e' il due senza il tre*", here I am defending my third thesis with Alessandro. I'd like to express my sincere gratitude to him for the trust he always granted me, the enthusiasm he was able to transmit, the freedom he gave me in pursing my scientific interests, and the encouragements with which he thrust me toward more and more ambitious challenges.

I also thank the members of my thesis committee, Prof. **Michele Benzi** and Prof. **James Nagy**, for their helpful suggestions and challenging questions. Their expert guidance in the new American academic system was of utmost importance to arrive where I am now.

A special thanks goes to **Tiziano**, **Lucia**, **Luca**, **Marina**, **Marta**, **Mauro**: their presence has been of great support but professionally and personally. They have been real friends to me, always willing to help, joke, and cheer me up in difficult moments. Also, their comments and feedback highly contributed to the quality of my work. I really enjoyed the long and productive discussions and I thank them for the fresh ideas and stimulating challenges they shared with me.

I am also indebted to Prof. **Vaidy Sunderam** and **Jaroslaw Slawinski**, their experience in parallel computing was an inestimable asset in building, executing, and tuning my applications on the HPC clusters and clouds. I acknowledge also Edgar Leon, our system administrator, for maintaining the local computational resources and supporting me with his invaluable know-how.

My sincere thanks goes also to the other faculty, staff, and graduate students for creating a friendly and stimulating environment. In particular, I am grateful to my teaching mentors Prof. **Dwight Duffus**, Prof **Skip Garibaldi**, and Prof. **Ron Gould**, who shared with me their passion, enthusiasm, and experience (sometimes also patience!) in teaching to undergrads. They helped me understanding the needs of my students and building a easygoing, professional and constructive relationship with them.

I owe special gratitude to Michele who introduced my to the National Labs world. I wish to thank Dr. **Judith Hill** (Oak Ridge Nat. Lab.) and Dr.

Panayot Vassilevski (Lawrence Livermore Nat. Lab.) for offering me the opportunity to work with them during my summer internships. It was a highly rewarding experience from which I learnt a lot and that helped my professional growth. The new connections I made with them and thanks to them will highly benefit the quality of my scientific research.

Last but not least, I would like to thank my family and my friends for their love and support. To my parents and sister, **Roberto**, **Ornella** and **Federica**, in Italy ("*le radici*", the roots) and to my wife **Deborah** ("*le ali*", the wings) I dedicate my thesis.

... and now, dear Debbie, it is time to fly west. Our journey together is waiting us!

Access to some computational resources was partially supported by US National Science Foundation Grant OCI-1124418, and used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575. We also thank Consorzio Interuniversitario Lombardo per L'Elaborazione Automatica (CILEA) for providing free access to their HPC clusters and for their assistance in building and executing our applications (Paride Dagna).

le radici e le ali

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

In this thesis, we are interested in efficient methods for the numerical solution of two important models that governs the motion of a fluid, the *incompressible Navier-Stokes* and the *Brinkman* equations. The first set of equations describes the flow of a Newtonian fluid in an either open or closed domain, while the second is commonly used to describe the flow in an inhomogeneous medium, where bubbles or channels alternate inside of a porous matrix. These equations are fundamental for many problems of practical interest, ranging from mechanical engineering to geophysics to biomedical to thermodynamics applications. Fast and accurate numerical solution of these problems is still a formidable challenge, as a consequence of the saddle point nature of the algebraic systems arising from finite element discretization of such equations. As it is well known [23, 107], the role of the pressure as Lagrange multiplier of the incompressibility constraint requires special numerical techniques, leading in general to large algebraic systems to be solved after space and time discretization. A further challenge is represented by the non linearity of the convective term (Navier-Stokes equations) or the extreme variations in the porous medium permeability (Brinkman equations).

Many different options are available for improving efficiency of numerical simulations, ranging from parallelization to identification of appropriate preconditioners [16, 45] and, in the unsteady case, to velocity-pressure splittings (see for example [106, 112, 125]) and time adaptivity [60, 61, 128]. In this thesis, we explore all these different acceleration techniques and we integrate them together in order to maximize the efficiency of the solver.

In the next sections we will describe in more details the two models of incompressible fluid that are studied in the thesis.

1.1 The incompressible Navier-Stokes equations

The incompressible Navier-Stokes equations (INS) govern the flow of a viscous Newtonian fluid in a generic domain Ω. Letting the velocity and kinetic pressure of the fluid be denoted by $\mathbf{u}(\mathbf{x}, t)$ and $p(\mathbf{x}, t)$ ($\mathbf{x} \in \Omega$, $t \in (t_0, T]$) respectively, the unsteady INS equations read

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} - \nabla \cdot \left(\nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)\right) + (\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega \times (t_0, T] \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega \times (t_0, T] \\ \mathcal{B}\,\mathbf{u} = \mathbf{g} & \text{on } \partial\Omega \times (t_0, T] \\ \mathbf{u}(\mathbf{x}, t_0) = \mathbf{u}_0 & \text{in } \Omega. \end{cases} \tag{1.1}$$

Here Ω is an open bounded domain in $\mathbb{R}^d$ where $d = 2, 3$, $\nu > 0$ is the kinetic viscosity, $\mathbf{f}$ an external force field, and $\mathcal{B}$ a boundary (trace) operator, including Dirichlet, Neumann or Robin conditions on different portions of the boundary $\partial\Omega$.

The Reynolds number $\mathrm{Re} = \frac{\|\mathbf{u}\| L}{\nu}$ represents the ratio between the viscous and the inertial forces in the fluid. Here L represents a characteristic length of the geometry. For example a common choice for L in the contest of blood flow problems is the average diameter of the vessel. In a straight channel with constant cross-sectional area, a Reynolds number of 4000 represents the transition between laminar and turbulent flows (see for example [110]). Given the kind of applications of interest for this thesis, we consider Reynolds number up to the order of few hundreds, which is typical in hemodynamic problems. In such conditions the flow is mostly in the laminar regime and shows disturbances or vortexes only in few localized regions. Such instabilities

occur in the presence of particular features of the geometry, such as obstacles, bifurcations, or steps. A well-known example is the flow past a cylinder benchmark proposed in [121]. In Fig. 1.1 we show the streamlines of few particles of fluids for a simulation with Reynolds number of 100, computed with the solvers implemented in this thesis. Small vortexes are visible in the region immediately downstream of the cylinder, while in the rest of the domain the flow is still in the laminar regime. Another example, this one of medical relevance, is shown in Fig. 1.2, where we compare the blood flow in the aorta of a healthy individual (right) and in a patient with implanted a left ventricular assistant device[1] (left). The left ventricular assistant device is a pump that helps the heart in pushing the blood from the left ventricular to the aorta. Notice how the device induces vortexes and recirculation in the blood flow that are not present in the healthy case.

The flow disturbances described above are usually captured by local mesh refinements, and do not require sophisticated turbulence models to be simulated. On the contrary, to accurately simulate fully developed turbulence, direct Navier-Stokes methods (DNS) would require highly refined meshes and they may not be a viable approach due to the extremely high computational cost. For example, in Fig. 1.3 we show a snapshot of a DNS simulation[2] of a fully developed turbulent flow (Reynolds number 3500) in a nozzle with a conical concentrator and a sudden expansion. It was necessary to compute more than 16 million of unknowns in order to capture a location of the jet rupture that was consistent with the experimental results presented in [65]. The simulation of even higher Reynolds numbers is beyond the scope of this thesis. For those problems, specialized techniques such as subgrid modeling of unresolved velocity and pressure modes [95] or deconvolution models [83] should be adopted to balance between accuracy and computational cost.

[1] Simulation prepared and executed by T. Passerini using the solvers presented in this thesis. Project in collaboration with Dr. D. Gupta, Department of Cardiology, Emory Hospital

[2] Courtesy of T. Passerini. Project in collaboration with A. Quaini, Dept. of Mathematics, Houston University

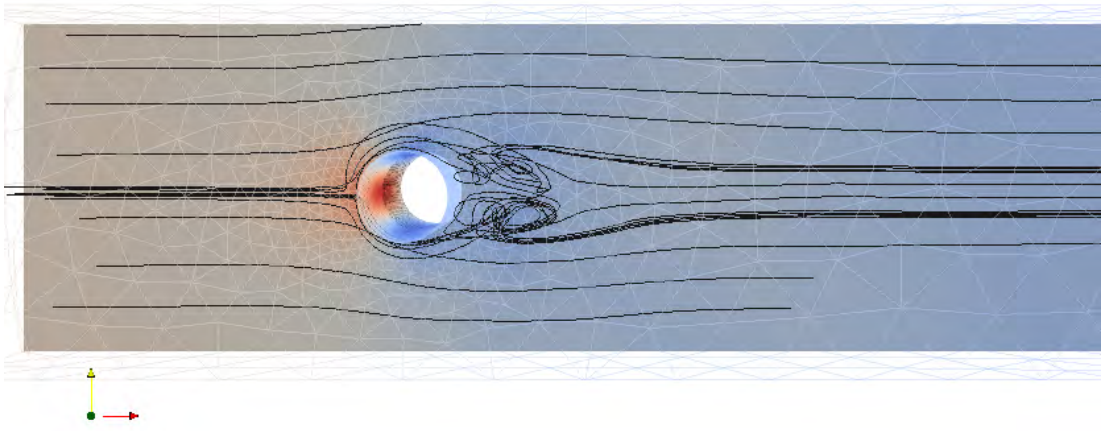


Figure 1.1: Flow past a cylinder for a Reynolds number of 100. At this Reynolds number the flow is mostly in the laminar regime, however observe the local formation flow disturbances downstream the cylinder.

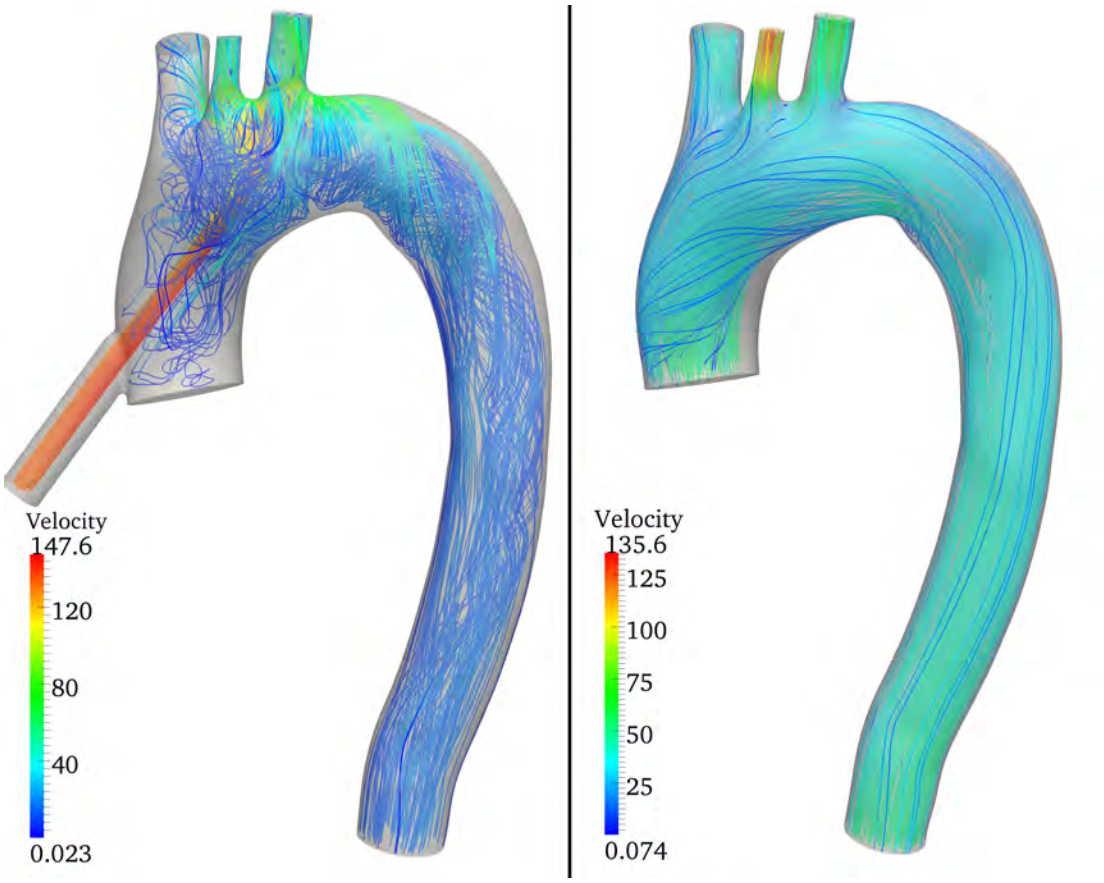


Figure 1.2: Localized turbulent flow in a blood flow simulation. The Left Ventricular Assistant Device (LVAD) is a pump that helps the heart in pushing the blood from the left ventricular to the aorta. Notice the formation vortexes in the blood flow where the LVAD hooks up to the aorta (left). These flow disturbances are not present in the case of a healthy person (right).

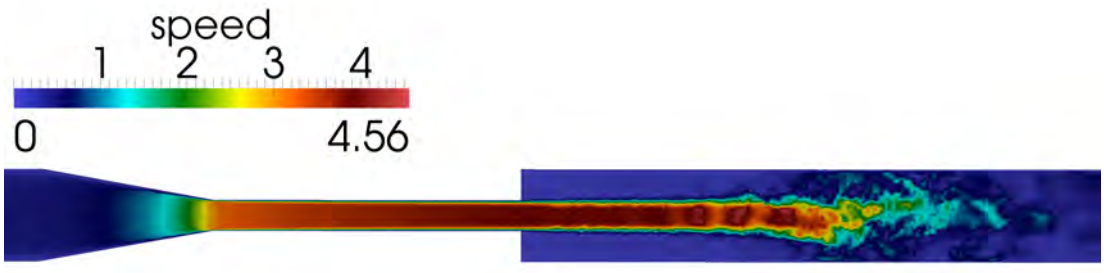


Figure 1.3: Direct Navier-Stokes simulation of turbulent flow past a step

The Reynolds number can therefore be interpreted as a measure of the difficulty of the problem: at high Reynolds number the convective term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ dominates the viscous stresses term, increasing the non-linearity and the non-symmetry of the problem. In the unsteady case, time lagging or extrapolation of the velocity field from the solution at the previous time steps (cfr. Section 2.4.1) is usually preferred to Newton or Picard iterations in order to cope with the non-linearity induced by the convective term. Overall, the computational advantages of this approach respect to non-linear iterations overcome the loss of accuracy in time (if low order extrapolation is used) or the conditional stability of the time advancing algorithm (if higher order extrapolation is used). Nevertheless convection dominated problems requires special stabilization techniques (streamline diffusion, streamline upwind Petrov Galerkin, least squares Galerkin) [45, 104] or discontinuous Galerkin (DG) discretizations [33, 85]. Moreover, the presence of a strong convection term represents a challenge also for the preconditioning of the saddle point problem.

1.2 The Brinkman Equations

The Brinkman equations describe the flow of a viscous fluid in cavity and porous media. It was initially proposed in [2, 3] as a homogenization technique for the Navier-Stokes equations. Since in real applications the number and the locations of the Stokes-Darcy interfaces might not be known *a priori*, the unified equations in the Brinkman model represent an advantage over the domain decomposition methods coupling the Darcy and the Stokes equations.

Mathematically speaking the Brinkman model is a parameter-dependent combination of the Darcy and Stokes models. Letting Ω be a bounded domain in $\mathbb{R}^d$ with a

regular enough boundary $\partial\Omega$, the steady Brinkman equations read

$$\begin{cases} -\nu\,\Delta\mathbf{u} + k(\mathbf{x})\mathbf{u} + \nabla p = \mathbf{f}(\mathbf{x}) & \forall \mathbf{x} \in \Omega \\ \text{div}\,\mathbf{u} = 0 & \forall \mathbf{x} \in \Omega \\ \mathcal{B}\mathbf{u} = \mathbf{g} & \text{on } \partial\Omega \end{cases} \tag{1.2}$$

where $\nu \geq 0$ is the fluid viscosity and $k(\mathbf{x})$ is the inverse permeability of the medium. If $\nu = 0$ the Brinkman equations reduce to the Darcy ones. The challenge of this problem is when the coefficient $k = k(\mathbf{x})$ admits extreme large variations between different parts of Ω. In Fig. 1.4 we show, as an example, an actual permeability field used in a oil reservoir simulation benchmark (`www.spe10.org`). Note that the permeability field admits jumps of several orders of magnitude between distinct layers of soil. The upper part is a Tarbet formation consisting of massive homogeneous highly permeable sandstones, while the lower part is a Ness formation, a heterogeneous mixture of sandstones, mudrocks, and coal.

In the part of the domain where k is small, the PDE behaves like a Stokes problem, whereas in the rest of the domain, it behaves like Darcy equation (Fig. 1.5). This double nature of the problem (Darcy or Stokes) represents a difficulty for the numerical discretization, since the construction of a finite element space which is uniformly well behaved with respect to both the Darcy and Stokes limits is still an area of active research [64, 130]. In addition, the high variability in the PDE coefficients negatively affects also the conditioning of the discrete problem which poses a substantial challenge for developing efficient preconditioners.

1.3 Thesis outline

In this thesis we investigate efficient solvers for the saddle point systems arising from finite element discretization of the two important classes of fluid dynamic problems

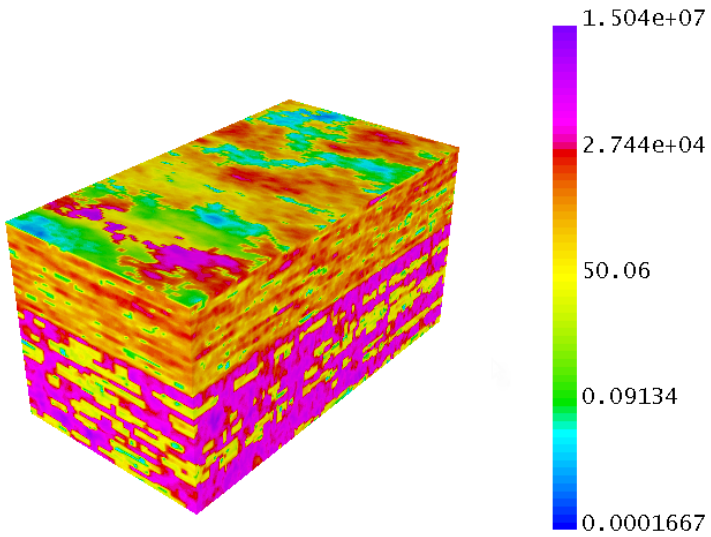


Figure 1.4: Logarithmic plot of the inverse permeability coefficient in the SPE10 problem, a challenging benchmark for reservoirs simulation codes. Note the two distinct soil layers and the large jumps in the coefficient between them.

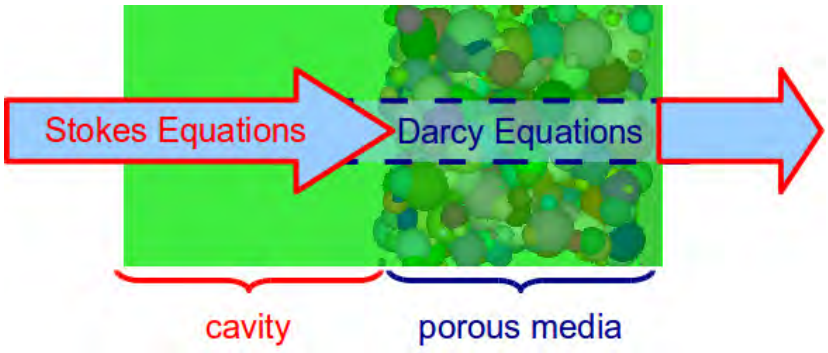


Figure 1.5: The Brinkman model is a unified law governing the flow of a viscous fluid in cavity (Stokes equations) and in porous media (Darcy Equations). In real applications, the number and the locations of the Stokes-Darcy interfaces might be not known *a priori*.

described in the previous sections.

For the solution of the **unsteady incompressible Navier-Stokes Equations**, we move from the pressure correction algebraic splittings formerly proposed by [112], and we introduce the incremental formulation of such schemes. These schemes feature an intrinsic hierarchical nature, such that an accurate approximation of the pressure Schur complement is obtained by computing intermediate low-order guesses [55, 126]. Such schemes prove to be efficient both when used as either a solver or a preconditioner. Additionally the difference between the pressure at two successive correction steps provides a natural a-posteriori estimator with no additional computational cost [128]. We consider the basic settings of the method and its more stable variants; we also discuss implementation details that make the method competitive for real interest problems.

For the solution of the **Brinkman Equations**, we follow the approach presented in [87] to precondition symmetric saddle point problems in a Hilbert complex. More specifically, we first present a novel mixed formulation of the Brinkman problem, with improved stability properties, in which we introduce the flow's vorticity as additional unknown. Such formulation extends to the Brinkman problem the mixed formulation of the Stokes problem in [12, 13]. Differently from other vorticity-velocity-pressure formulations (see e.g. [103], [96]) where the equation for the vorticity is obtained by taking the curl of the momentum equation, our formulation exploits the structure of the de Rham complex

$$H(\text{curl}) \xrightarrow{\text{curl}} H(\text{div}) \xrightarrow{\text{div}} L^2$$

to translate the governing equations into a mixed finite element settings. Indeed, this formulation leads to a uniformly stable and conforming discretization by standard finite elements (Nédélec, Raviart-Thomas, piecewise discontinuous). Based on the stability analysis of the problem in the $H(\text{curl}) - H(\text{div}) - L^2$ norms, we derive a scalable block diagonal preconditioner which is optimal in the constant coefficient case. Such pre-

conditioner is based on the auxiliary space algebraic multigrid solvers for $H(\text{curl})$ and $H(\text{div})$ (see e.g. [10, 11, 70, 80, 81]).

Algorithms and preconditioners analyzed in this thesis have been implemented in a parallel C++ code, using the finite element libraries LifeV and MFEM, and the linear algebra libraries Trilinos and HYPRE. Particular emphasis is posed in measuring the performance of the proposed algorithms in solving problems of practical interest, involving complex geometries and realistic flow conditions, as idealized case tests and simple geometries might be not fully reliable estimators of the performances of the algorithms. Numerical experiments in 2D and 3D support the validity of our approach showing both good efficiency and parallel scalability properties of the solvers proposed.

The thesis is organized as follows.

In **Chapter** 2 we provide some notations and basic settings used through this thesis. We first briefly derive the incompressible Navier-Stokes equations from the physical principles of mass, momentum, and energy conservation. We next introduce the weak formulation of the problem and we outline the well-posedness of the semidiscrete problem obtained with Galerkin Projection. Particular emphasis is posed on the importance of the compatibility between the velocity and the pressure spaces (*inf-sup condition*). We then address the space and time discretization of the Navier-Stokes equations by, respectively, Finite Element Method (FEM) and Backward Differential Formulas (BDF). We conclude the chapter with an original analysis of mass lumping techniques for high order finite elements.

In **Chapter** 3 we review, in a unified framework, the different velocity-pressure splitting methods based on an inexact block factorization of the fully coupled Navier-Stokes equations. We also introduce the incremental formulation, a popular approach to improve the accuracy of the splitting method. We next analyze in details the pressure

corrected Yosida schemes and we provide stability and convergence results. Finally we demonstrate how velocity and pressure splitting methods can be efficiently used as preconditioners for the coupled saddle point system.

In **Chapter** 4 we explain how the algebraic splitting methods introduced in the previous chapter can be successfully utilized to derive a posteriori error estimator for time adaptivity. As a matter of fact, the hierarchical structure of the pressure corrected schemes provides, as a by product of the computations, an estimate of the local splitting error that can be exploited to adaptively choose the time step. Numerical results suggest that time adaptivity can sensibly reduce the computational time in hemodynamic applications.

In **Chapter** 5 we provide implementation details and performance measures for the algebraic splitting schemes and preconditioners presented in Chapter 3. After a brief introduction of the scientific computing libraries LifeV, Trilinos, and SuiteSparseQR, we discuss efficient solvers targeting (1) small-medium scale applications that can easily run on personal laptops, and (2) large scale applications on massively parallel architectures. In particular, for small-medium scale applications we show how direct solvers for the pressure system can drastically improve performances of the pressure corrected schemes. For large scale applications, we present weak and strong scalability results of the block factorization preconditioner framework implemented in this thesis work.

In **Chapter** 6 we study a stable discretization and a robust preconditioner for the Brinkman problem. We first propose a novel mixed discretization with improved stability properties, and then we derive a scalable block diagonal preconditioner that is optimal in the constant coefficient case. The methods and numerical experiments presented in this chapter are the result of two internships (summer 2011, summer 2012) spent working at Lawrence Livermore National Laboratory (LLNL, Livermore, CA) under the supervision of Dr. P. Vassilevski.

Concluding remarks, future work guidelines and insights are reported in **Chapter**

7.

2 Discretization of the unsteady Navier-Stokes equations

In this chapter, we start by providing a brief derivation of the equations governing the motion of a fluid starting from the general physical principles of mass, momentum, and energy conservation and then we introduce the simplifying assumptions that lead to the well-known Navier-Stokes equations (Section 2.1). We then define the functional spaces and the weak formulation of the Navier-Stokes equations that represents the starting point of many space discretization methods based on Galerkin projection, such as Finite Elements, Spectral Methods, Discontinuous Galerkin (Section 2.2). We conclude by addressing the numerical discretization in space and time of the mixed formulation. To this aim, we initially discuss the finite element discretization of the Oseen problem (an auxiliary problem resulting from linearization of the Navier-Stokes equations), with particular emphasis on the choice of the discrete finite element spaces and on the stabilization techniques for convection dominated problems (Sections 2.3). Then, we present time-stepping techniques based on finite differences in time (BDF formulas) and different treatments (fully-implicit, semi-implicit, fully-explicit) of the non-linear convective term (Section 2.4)). We conclude the chapter with a note on mass lumping techniques for high order finite elements (Section 2.5). In particular, we discuss how the use of inexact quadrature rules in the assembly of the finite element mass matrices affects the overall accuracy of the space discretization. The analysis of these effects based on the Strang Lemma is an original results in this thesis work.

2.1 Governing equations

The equations describing the motion of a general fluid are derived from three conservation laws (mass, momentum, energy). Let us denote by $\mathbf{u}$ the velocity of the fluid, $\rho > 0$ its density, and $\hat{e} = e + \frac{1}{2}|\mathbf{u}|^2$ its total (thermal and kinetic) internal energy per unit mass. For a quantity of interest G ($G = \rho, \rho\mathbf{u}, \rho\hat{e}$), let us define the material derivative

$$\frac{DG}{Dt} = \frac{\partial G}{\partial t} + \text{div}(G\mathbf{u}).$$

Then the governing equations, in conservative forms, reads

$$\begin{cases} \dfrac{D\rho}{Dt} = \dfrac{\partial \rho}{\partial t} + \text{div}(\rho\mathbf{u}) = 0 & \text{(mass conservation)} \\ \dfrac{D(\rho\mathbf{u})}{Dt} = \dfrac{\partial(\rho\mathbf{u})}{\partial t} + \text{div}(\rho\mathbf{u}\otimes\mathbf{u}) = \text{div}\, T + \rho\mathbf{f} & \text{(momentum conservation)} \\ \dfrac{D(\rho\hat{e})}{Dt} = \dfrac{\partial\,(\rho\hat{e})}{\partial t} + \text{div}\,[(\rho\hat{e})\,\mathbf{u}] = \text{div}(T\ \mathbf{u}) - \text{div}\mathbf{q} + \rho\mathbf{f}\cdot\mathbf{u} + \rho r & \text{(energy conservation).} \end{cases} \tag{2.1}$$

Here we have denoted by $T = T(\mathbf{u}, \hat{p})$ the stress tensor, $\hat{p}$ the pressure, $\mathbf{q} = \mathbf{q}(\theta)$ the heat flux, $\theta > 0$ the (absolute) temperature, and finally by $\mathbf{f}$ and r the external force field per unit mass and heat source per unit mass per unit time, respectively. Moreover we have used the notation

$$(\mathbf{u}\otimes\mathbf{u})_{ij} := u_i u_j, \quad (\text{div}\, T)_i := \sum_{j=1}^{d} \frac{\partial T_{ij}}{\partial x_j}, \quad (T\ \mathbf{u})_i := \sum_{j=1}^{d} T_{ij} u_j, \quad i = 1, \ldots, d.$$

With standard simplifications and by exploiting the conservation of mass, the non-conservative form of (2.1) reads

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \text{div}(\rho\mathbf{u}) = 0 \\ \rho\dfrac{\partial(\mathbf{u})}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = \text{div}\, T + \rho\mathbf{f} \\ \rho\dfrac{\partial e}{\partial t} + \rho\mathbf{u}\cdot\nabla\ e = T : \varepsilon(\mathbf{u}) - \text{div}\mathbf{q} + \rho\mathbf{f}\cdot\mathbf{u} + \rho r, \end{cases} \tag{2.2}$$

where $\varepsilon(\mathbf{u})$ denotes the symmetric deformation tensor those components are given by

$$\varepsilon_{ij} = \left[\frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\right]_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{2.3}$$

and

$$T : \varepsilon(\mathbf{u}) = \sum_{i,j} T_{ij}\ \varepsilon_{ij}.$$

To close system (2.2) we need to introduce the constitutive equations which characterize the motion and thermodynamics of the fluid considered, and the state equations that relate pressure $\hat{p}$, internal energy e, density ρ and temperature θ. Constitutive equations and state equations are specific of the fluid and type of flow considered. In the following we will restrict ourself to the analysis of the incompressible, isothermal flow of a Newtonian fluid.

For incompressible flows, that are of main interest in this thesis, the pressure $\hat{p}$ is no longer related to the thermodynamic variables e, ρ, θ, but it is determined solely through the momentum equation $(2.2)_2$. In particular, the state equation for the pressure is replaced by the incompressibility constraint

$$\text{div}\ \mathbf{u} = 0. \tag{2.4}$$

The stress tensor is now independent of the thermodynamic variables and linear with respect to the strain tensor $\varepsilon(\mathbf{u})$, according to the constitutive equation

$$T = -\hat{p}I + 2\ \mu\ \varepsilon(\mathbf{u}) = -\hat{p}I + \mu\ \left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right), \tag{2.5}$$

where μ is the dynamic viscosity of the fluid, and I the identity tensor.

Thanks to the hypothesis of isothermal flow, we also assume that density ρ and dynamic viscosity μ are independent of the fluid temperature θ, so that we can decouple

the motion equations from the thermodynamics.

Under the above assumptions, we drop the energy equation and then rewrite system (2.2) in the simplified form

$$\begin{cases} \rho\left(\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla\hat{p} + \text{div}\left[\mu\ \left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)\right] + \rho\mathbf{f} \\ \dfrac{\partial \rho}{\partial t} + \mathbf{u}\nabla\rho = 0. \end{cases} \tag{2.6}$$

Finally, if we make the assumption of homogeneous flow, the density ρ does not depend of the space coordinate $\mathbf{x}$. Equation $(2.6)_2$ implies that for an incompressible and homogeneous fluid the density is equal to a constant ρ_0 at every point $\mathbf{x}$ in the domain Ω and instant t in $(0,\ T]$. Under these assumptions, and by letting $p = \frac{\hat{p}}{\rho_0}$, $\nu = \frac{\mu}{\rho_0}$ be the kinematic pressure and viscosity, the Navier-Stokes system takes the well known form

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} - \text{div}\left[\nu\ \left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)\right] + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \nabla p = \mathbf{f} & \text{in } \Omega\times(t_0,T] \\ \text{div}\,\mathbf{u} = \mathbf{0} & \text{in } \Omega\times(t_0,T]. \end{cases} \tag{2.7}$$

This system is then completed by prescribing an initial condition $\mathbf{u}_0$ on the fluid velocity,

$$\mathbf{u}(t=0,\mathbf{x}) = \mathbf{u}_0(\mathbf{x}), \quad \forall\,\mathbf{x}\in\Omega,$$

and proper boundary conditions. Generally the surface stress $\mathbf{h} = T\,\mathbf{n}$ is prescribed on a portion of the boundary Γ_n and a velocity profile $\mathbf{g}$ on the remaining part, called Γ_e. A possible set of boundary conditions for problem (2.7) is therefore

$$\begin{aligned} -p\,\mathbf{n} + \nu\ \left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)\ \mathbf{n} &= \mathbf{h} \quad \text{on } \Gamma_n\ \subset\ \partial\Omega \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \Gamma_e\ \subset\ \partial\Omega \end{aligned} \tag{2.8}$$

with $\Gamma_n \cup \Gamma_e = \partial\Omega$ and $\Gamma_n \cap \Gamma_e = \emptyset$. In case $\Gamma_n = \emptyset$ the boundary velocity $\mathbf{g}$ must

satisfy the condition $\int_{\partial\Omega} \mathbf{g} \cdot \mathbf{n} = 0$ to ensure compatibility with div $\mathbf{u} = 0$. For this set of boundary conditions the pressure p is defined up to constant.

Boundary conditions on Γ_e are called *essential* boundary conditions, since they will be imposed strongly in the variational formulation of (2.7) by an opportune choice of the functional spaces, and boundary conditions on Γ_n are called *natural* because they originate by integration by parts of the non-conforming terms.

In the following, for easiness of notation, we will take $\mathbf{g} = \mathbf{0}$, as the extension to non-homogeneous essential boundary conditions is rather standard. To this aim, one writes $\mathbf{u} = \overset{\circ}{\mathbf{u}} + \mathbf{u}_g$ where $\mathbf{u}_g$ is any function (*lifting*) such that $\mathbf{u}_g = \mathbf{g}$ on Γ_e and the new unknown $\overset{\circ}{\mathbf{u}}$ satisfies $\overset{\circ}{\mathbf{u}} = \mathbf{0}$ on Γ_e. The incompressibility constraint then reads div $\overset{\circ}{\mathbf{u}} = -\text{div}\,\mathbf{u}_g$, resulting in general in a non-homogeneous right end side for equation $(2.7)_2$. We refer e.g. to [52, Chapter 7] for the details.

To conclude this section, we observe that, by assuming constant viscosity and thanks to the incompressibility constraint (2.4), we can simplify the divergence of the viscous stress tensor as follows

$$-\nabla \cdot \left(\nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)\right) = -\nu\left(\Delta \mathbf{u} + \nabla(\text{div}\,\mathbf{u})\right) = -\nu \Delta \mathbf{u}. \tag{2.9}$$

However it is worth to notice that the equality above does not usually hold at the discrete level, leading to different sparsity patterns in the finite element matrices and in differences in the numerical solution, as shown in Section 2.3.

2.2 Weak formulation and Galerkin Projection

Let X denote a scalar Hilbert space. Then $\mathbf{X} = X^d$ is a vectorial Hilbert space, such that each component u_i ($i = 1, \ldots, d$) of a function $\mathbf{u} \in \mathbf{X}$ belongs to X. If $p, q \in X$ we denote with the symbol $(p, q)_X$ the standard inner product in X, and with $(\mathbf{u}, \mathbf{v})_{\mathbf{X}} := \sum_{i=1}^{d} (u_i, v_i)_X$ its vectorial counterpart.

For example, if Ω is a bounded regular domain in $\mathbb{R}^d$ ($d = 2, 3$), $L^2(\Omega)$ is the space of square-integrable functions and $\mathbf{L}^2(\Omega) := [L^2(\Omega)]^d$ is the space of the vectorial functions $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$, such that $\mathbf{f}_i$ is in $L^2(\Omega)$. The inner products in such spaces read

$$(p, q)_{L^2(\Omega)} = \int_\Omega p\ q\ d\Omega, \quad (\mathbf{u}, \mathbf{v})_{\mathbf{L}^2(\Omega)} = \int_\Omega \mathbf{u} \cdot \mathbf{v}\ d\Omega.$$

We also introduce the spaces

$$H^1(\Omega) := \left\{ u \in L^2(\Omega) \mid \nabla u \in \mathbf{L}^2(\Omega) \right\}, \quad \mathbf{H}^1(\Omega) = \left\{ \mathbf{u} \in \mathbf{L}^2(\Omega) \mid u_i = (\mathbf{u})_i \in H^1(\Omega) \right\},$$

equipped with the inner products

$$\begin{aligned} (u, v)_{H^1(\Omega)} &= \int_\Omega u\ v\ d\Omega + \int_\Omega \nabla u \cdot \nabla v\ d\Omega, \\ \text{and} \quad (\mathbf{u}, \mathbf{v})_{\mathbf{H}^1(\Omega)} &= \int_\Omega \mathbf{u} \cdot \mathbf{v}\ d\Omega + \int_\Omega \nabla \mathbf{u}\ :\ \nabla \mathbf{v}\ d\Omega. \end{aligned}$$

In the following we will omit the subscript in the inner product symbol for the spaces $L^2(\Omega)$ or $\mathbf{L}^2(\Omega)$.

To write the weak formulation of the Navier-Stokes problem, we introduce the spaces $\mathbf{V} \subseteq \mathbf{H}^1(\Omega)$ and $W \subseteq L^2(\Omega)$ for the velocity $\mathbf{u}(\mathbf{x}, t)$ and pressure $p(\mathbf{x}, t)$, respectively. The actual definition of the functional spaces $\mathbf{V}$ and W depends on the boundary conditions. For example, if we consider the boundary conditions in (2.8), we have

$$\mathbf{V} := \left\{ \mathbf{u} \in \mathbf{H}^1(\Omega) \mid \mathbf{u} = \mathbf{0} \text{ on } \Gamma_e \right\}, \text{ and } W := L^2(\Omega),$$

while in the case of only essential boundary conditions ($\Gamma_e = \partial\Omega$), we have

$$\mathbf{V} = \mathbf{H}_0^1(\Omega) = \left\{ \mathbf{u} \in \mathbf{H}^1(\Omega) \mid \mathbf{u} = \mathbf{0} \text{ on } \partial\Omega \right\}, \text{ and } W := L^2(\Omega) \backslash \mathbb{R}.$$

To obtain the weak formulation, we multiply the system (2.7) by arbitrary functions

$\mathbf{v} \in \mathbf{V}$ and $q \in W$ and integrate over the domain Ω,

$$\begin{cases} \displaystyle\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v}\, d\Omega + \int_\Omega (\mathbf{u} \cdot \nabla)\, \mathbf{u} \cdot \mathbf{v}\, d\Omega - \\ \qquad \displaystyle - \int_\Omega \operatorname{div}\left(\nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)\right) \cdot \mathbf{v}\, d\Omega + \int_\Omega \nabla p \cdot \mathbf{v}\, d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v}\, d\Omega \quad \forall \mathbf{v} \in \mathbf{V} \\ \displaystyle\int_\Omega (\operatorname{div} \mathbf{u})\, q\, d\Omega = 0 \qquad \forall q \in W. \end{cases}$$

Exploiting some basic integration-by-parts identities and the boundary conditions (2.8), the non-conforming terms become

$$\begin{aligned} &- \int_\Omega \operatorname{div}\left(\nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)\right) \cdot \mathbf{v}\, d\Omega + \int_\Omega \nabla p \cdot \mathbf{v}\, d\Omega = \\ &\int_\Omega \nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) : \left(\nabla \mathbf{v} + \nabla \mathbf{v}^T\right)\, d\Omega - \int_\Omega p\, (\operatorname{div} \mathbf{v})\, d\Omega - \int_{\partial\Omega} \left(-p\, I + \nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)\right) \mathbf{n} \cdot \mathbf{v}\, dS = \\ &\qquad \int_\Omega \nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) : \left(\nabla \mathbf{v} + \nabla \mathbf{v}^T\right)\, d\Omega - \int_\Omega p\, (\operatorname{div} \mathbf{v})\, d\Omega - \int_{\Gamma_n} \mathbf{h} \cdot \mathbf{v}\, dS. \end{aligned}$$

Therefore a weak solution $\mathbf{u}(t) \in \mathbf{V}$, $p(t) \in W$ of (2.7) satisfies the variational problem

$$\begin{cases} \displaystyle\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v}\, d\Omega + \int_\Omega (\mathbf{u} \cdot \nabla)\, \mathbf{u} \cdot \mathbf{v}\, d\Omega + \int_\Omega \nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) : \left(\nabla \mathbf{v} + \nabla \mathbf{v}^T\right)\, d\Omega - \\ \qquad \displaystyle\int_\Omega p \operatorname{div} \mathbf{v}\, d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v}\, d\Omega + \int_{\Gamma_n} \mathbf{h} \cdot \mathbf{v}\, dS \qquad \forall \mathbf{v} \in \mathbf{V} \\ \displaystyle - \int_\Omega (\operatorname{div} \mathbf{u})\, q\, d\Omega = 0 \qquad \forall q \in W. \end{cases}$$

Introducing the variational forms

- $F(\mathbf{v}) = \displaystyle\int_\Omega \mathbf{f} \cdot \mathbf{v}\, d\Omega + \int_{\Gamma_n} \mathbf{h} \cdot \mathbf{v}\, dS, \quad \mathbf{v} \in \mathbf{V}$,
- $m(\mathbf{u}, \mathbf{v}) = \displaystyle\int_\Omega \mathbf{u} \cdot \mathbf{v}\, d\Omega, \quad \mathbf{u}, \mathbf{v} \in \mathbf{V}$,
- $k(\mathbf{u}, \mathbf{v}) = \displaystyle\int_\Omega \nu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) : \left(\nabla \mathbf{v} + \nabla \mathbf{v}^T\right)\, d\Omega, \quad \mathbf{u}, \mathbf{v} \in \mathbf{V}$,
- $c(\mathbf{u}, \mathbf{v}; \mathbf{w}) = \displaystyle\int_\Omega (\mathbf{w} \cdot \nabla)\, \mathbf{u} \cdot \mathbf{v}\, d\Omega, \quad \mathbf{u}, \mathbf{v} \in \mathbf{V}$,

- $b(\mathbf{u}, q) = -\int_\Omega (\text{div } \mathbf{u})\ q\ d\Omega, \quad \mathbf{u} \in \mathbf{V},\ p \in W,$

we write the non-linear saddle-point problem

$$\begin{cases} \text{find } (\mathbf{u}(t), p(t)) \in \mathbf{V} \times W : \\ \frac{d}{dt}[m(\mathbf{u}(t), \mathbf{v})] + k(\mathbf{u}(t), \mathbf{v}) + c(\mathbf{u}(t), \mathbf{v}; \mathbf{u}(t)) + b^*(p(t), \mathbf{v}) = F(\mathbf{v}) & \forall \mathbf{v} \in \mathbf{V} \\ b(\mathbf{u}(t), q) = 0 & \forall q \in W. \end{cases} \tag{2.10}$$

In the above formulation, the bilinear form $b(\mathbf{u}, q)$ represents the incompressibility constraint (2.4). Functions in $\mathbf{V} \times W$ are such that the following *inf-sup condition* (Ladyzhenskaja-Babuska-Brezzi condition) is verified

$$\inf_{\mathbf{u} \in \mathbf{V}} \sup_{q \in W} \frac{b(\mathbf{u}, q)}{\|\mathbf{u}\|_{\mathbf{V}} \|q\|_W} \geq \beta_\Omega, \tag{2.11}$$

where β_Ω is a positive constant that depends only on the domain Ω.

A discrete in space approximation of (2.10) can be devised by applying the Galerkin method. With this aim, let $\{\mathbf{V}_h \mid h > 0\}$ be a family of finite dimensional subspaces of $\mathbf{V}$, and $\{W_h \mid h > 0\}$ a family of finite element subspaces of W. We assume that the spaces $\mathbf{V}_h$ and W_h are compatible in the sense that the inf-sup condition (2.11) holds also in the discrete case uniformly respect to discretization parameter h. In other words there exists a constant $\beta = \beta(\Omega)$ independent from h such that

$$\inf_{\mathbf{u}_h \in \mathbf{V}_h} \sup_{q_h \in W_h} \frac{b(\mathbf{u}_h, q_h)}{\|\mathbf{u}_h\|_{\mathbf{V}} \|q_h\|_W} \geq \beta. \tag{2.12}$$

Additionally we require the following approximation property to hold: there exist two operators $\Pi_h^{\mathbf{V}} : \mathbf{V} \to \mathbf{V}_h$ and $\Pi_h^W : W \to W_h$ such that

$$\|\mathbf{v} - \Pi_h^{\mathbf{V}}(\mathbf{v})\|_{\mathbf{V}} \to 0, \quad \|q - \Pi_h^W(q)\|_W \to 0 \quad \text{as } h \to 0.$$

Then, for each $h > 0$ we consider the non-linear semi-discrete in space problem

$$\begin{cases} \text{find} (\mathbf{u}_h(t), p_h(t)) \in \mathbf{V}_h \times W_h : \\ \frac{d}{dt}(\mathbf{u}_h(t), \mathbf{v}) + k(\mathbf{u}_h(t), \mathbf{v}_h) + c(\mathbf{u}_h(t), \mathbf{v}_h; \mathbf{u}_h(t)) + b^*(p_h(t), \mathbf{v}_h) = F(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in \mathbf{V}_h \\ b(\mathbf{u}_h(t), q_h) = 0 \qquad \forall q_h \in W_h. \end{cases} \tag{2.13}$$

For the convergence of the projected solution $(\mathbf{u}_h, p_h)$ of the Galerkin projection to the weak solution of (2.10) we refer, for example to [117].

Time discretization and Picard-like linearization of (2.13) (see Section 2.4) results in a sequence of (generalized) Oseen problems of the form

$$\begin{cases} \text{find } (\mathbf{u}_h, p_h) \in \mathbf{V}_h \times W_h : \\ \sigma[m(\mathbf{u}_h, \mathbf{v}_h)] + a(\mathbf{u}_h, \mathbf{v}_h) + b^*(p_h, \mathbf{v}_h) = F(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in \mathbf{V}_h \\ b(\mathbf{u}_h, q_h) = 0 \qquad \forall q_h \in W_h. \end{cases} \tag{2.14}$$

Here σ is a scalar value proportional to the inverse of the time step, and the variational form a is defined as

$$a(\mathbf{u}_h, \mathbf{v}_h) = k(\mathbf{u}_h, \mathbf{v}_h) + c(\mathbf{u}_h, \mathbf{v}_h; \mathbf{w}_h), \tag{2.15}$$

with $\mathbf{w}_h \in \mathbf{V}_h$ being a given velocity field (*wind velocity*). When $\mathbf{w}_h = \mathbf{0}$, the variational form a is symmetric positive definite, and we obtain the generalized Stokes problem.

In Section 2.3 we will discuss the Finite Element Method for problem (2.14), and in Section 2.4 we will describe time advancing and linearization techniques for problem (2.10).

2.3 Space discretization of the generalized Oseen Problem

We now proceed to the numerical approximation of the generalized Oseen problem (2.14) by using the Finite Element Method (FEM).

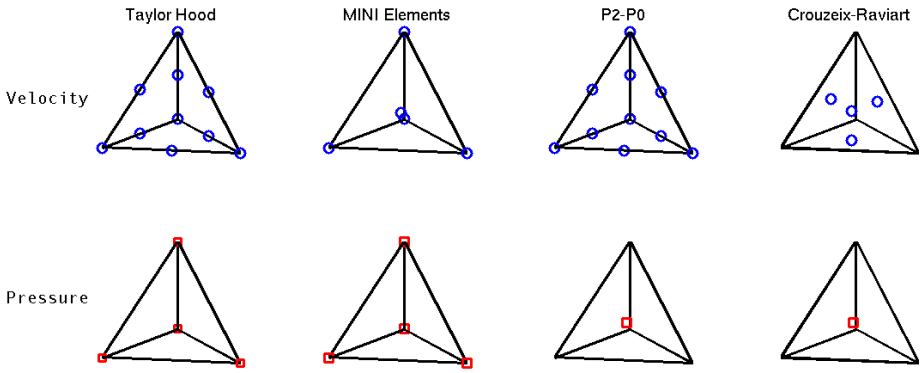


Figure 2.1: Inf-Sup compatible velocity (top) and pressure (bottom) spaces. Continuous pressure spaces (Taylor Hood, MINI Elements) on the left, discontinuous pressure elements (P2-P0, Crouzeix-Raviart) on the right.

We introduce a conformal, regular, and quasi-uniform partition $\mathcal{T}_h$ of the domain Ω in N_e polygonal (2D) or polyhedral (3D) elements T_k such that

$$\Omega \approx \Omega_h = \bigcup T_k, \text{ with } h = \max_{T_k \in \mathcal{T}_h} h_k, \quad h_k = \text{diam}(T_k), \ k = 1, \ldots, N_e.$$

Then we assume that $\mathbf{V}_h \subset \mathbf{V}$ is the finite dimensional space ($\dim(\mathbf{V}_h) = N_\mathbf{u}$) of vectorial piecewise-polynomial functions on every element of $\mathcal{T}_h$, and globally continuous in Ω. Similarly, $W_h \subset W$ ($\dim(W_h) = N_p$) is the space of piecewise polynomial functions on every element of the decomposition, not necessarily continuous. To guarantee the stability of the mixed formulation (2.14) we also assume that the spaces $\mathbf{V}_h$ and W_h satisfy the discrete *inf-sup condition* (2.12). Examples of *inf-sup compatible* spaces are shown in Fig. 2.1.

By denoting with $\{\varphi_j\}$, $j = 0, \ldots, N_\mathbf{u} - 1$ a base of the space $\mathbf{V}_h$ (*shape functions*), the finite element approximation $\mathbf{u}_h \in \mathbf{V}_h$ is uniquely associated with the vector $\mathbf{U} = \left[U_0, U_1, \ldots, U_{N_\mathbf{u}-1}\right]^T$ in the expansion

$$\mathbf{u}_h(\mathbf{x}) = \sum_{j=0}^{N_\mathbf{u}-1} U_j \boldsymbol{\varphi}_j(\mathbf{x}).$$

In a similar way, denoting with $\{\psi_j\}$, $j = 0, \dots, N_p - 1$, the shape functions of the space W_h, we have

$$P = \Big[P_0, P_1, \cdots, P_{N_p-1}\Big]^T, \quad p_h(\mathbf{x}) = \sum_{j=0}^{N_p-1} P_j \psi_j(\mathbf{x}).$$

By taking $\mathbf{v}_h = \boldsymbol{\varphi}_i$ $(0 \le i \le N_\mathbf{u} - 1)$ and $q_h = \psi_i$ $(0 \le i \le N_p - 1)$ in (2.14), we finally obtain the linear system

$$\begin{cases} \displaystyle\sum_{j=0}^{N_\mathbf{u}-1} \left(\sigma[m(\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i)] + a(\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i)\right) U_j + \sum_{j=0}^{N_p-1} b^*(\psi_j, \boldsymbol{\varphi}_i) P_j = F(\boldsymbol{\varphi}_i) & i = 0, \cdots, N_\mathbf{u} - 1 \\ \displaystyle\sum_{j=0}^{N_\mathbf{u}-1} b(\boldsymbol{\varphi}_j, \psi_i) U_j = 0 & i = 0, \cdots, N_p - 1. \end{cases}$$

This can be written in matrix form as

$$\begin{bmatrix} \sigma M + A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix}, \tag{2.16}$$

with

- $M = [m_{ij}] \in \mathbb{R}^{N_\mathbf{u} \times N_\mathbf{u}}, \quad m_{ij} = m(\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i), \quad 0 \le i, j \le N_\mathbf{u} - 1,$
- $A = [a_{ij}] \in \mathbb{R}^{N_\mathbf{u} \times N_\mathbf{u}}, \quad a_{ij} = a(\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i), \quad 0 \le i, j \le N_\mathbf{u} - 1,$
- $B = [b_{ij}] \in \mathbb{R}^{N_p \times N_\mathbf{u}}, \quad b_{ij} = b(\boldsymbol{\varphi}_j, \psi_i), \quad 0 \le i \le N_p - 1,\ 0 \le j \le N_\mathbf{u} - 1,$
- $\mathbf{f} = [f_i] \in \mathbb{R}^{N_\mathbf{u}}, \quad [f_i] = F(\boldsymbol{\varphi}_i), \quad 0 \le i \le N_\mathbf{u} - 1.$ In the following we will denote with $\mathcal{A}$ the block matrix in (2.16).

The inf-sup condition (2.12) guarantees that the pressure gradient matrix B^T has

full column rank. Indeed, for each $\mathbf{v}_h = \sum_{j=0}^{N_\mathbf{u}-1} V_j \boldsymbol{\varphi}_j$, $\|\mathbf{v}_h\|_\mathbf{V} = 1$, there exists $q_h = \sum_{j=0}^{N_\mathbf{u}-1} Q_j \boldsymbol{\varphi}_j$, $\|q_h\|_W = 1$, such that

$$QB\mathbf{V} = b(\mathbf{v}_h, q_h) \geq \beta.$$

The above inequality ensures the absence of *spurious* modes in the numerical approximations, that is oscillatory modes $\widehat{P}$ such that $B^T\widehat{P} = 0$. We refer to [21, 23] for more details on how to ensure the inf-sup compatibility of the spaces $\mathbf{V}_h$, W_h, and to [45, 107] for stabilization techniques to circumvent the inf-sup condition.

In Fig. 2.2 we show the typical sparsity pattern of the block matrix $\mathcal{A}$, after finite element discretization with Taylor-Hood elements. Note that, when the consistent viscous stress tensor $2\,\nu\,\varepsilon(\mathbf{u})$ is used, all the velocity components are coupled together (left). On the contrary, if one uses the simplification (2.9), it is possible to decouple the velocity components inducing a block diagonal structure in the matrix $\mathcal{A}$ (right). However in many applications, even if more computational expensive, the first approach is preferred because of its stabilizing effect (described in the next section) and the more physical natural boundary conditions associated to it. Moreover, to exploit (2.9) in a fluid-structure interaction problem can compromise the fidelity of the simulation by underestimating the stresses at the fluid-solid interface.

Stabilization of convection dominates problem

Another important aspect to be addressed in the discretization of the Oseen Problem is that severe oscillations may arise in the numerical approximation $\mathbf{u}_h$ of the velocity field if the viscosity ν is small with respect to the norm of the wind velocity $\|\mathbf{w}_h\|_{L^\infty(\Omega)}$ (convection dominated problem). To be more specific, let us

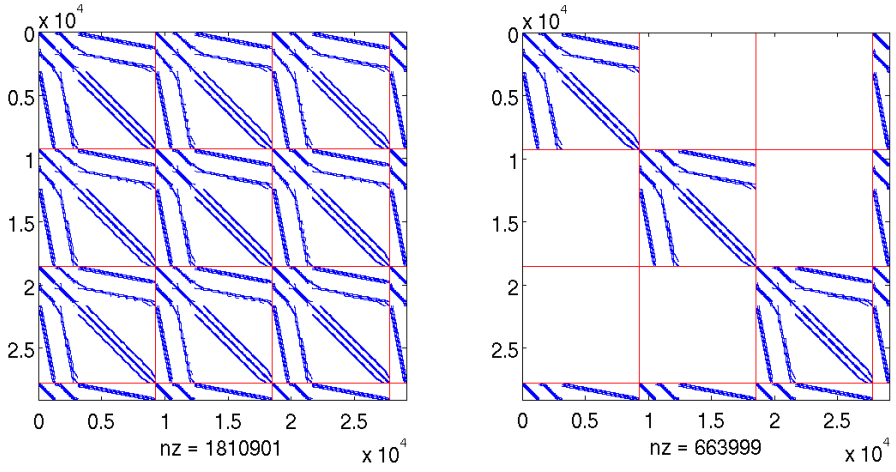


Figure 2.2: Sparsity pattern of Taylor-Hood finite element matrices for the Oseen Problem: consistent viscous stress tensor (left), consistent viscous stress tensor (right).

introduce the local *Péclet number* defined as

$$Pe_k := \frac{\|\mathbf{w}_h\|_{L^\infty(T_k)} h_k}{2\,\nu}, \quad k = 1, \ldots, N_e.$$

In the regions of the domain where Pe_k is small (less than 1) the viscous stresses dominate the convective term and the finite element methods provides a satisfactory approximation of the solution; while in the rest of the domain (large Pe_k) numerical viscosity should be added to guarantee stability [45, 107].

Many different stabilization techniques for advection dominated problems are known in literature: artificial diffusion, streamline upwind, SUPG (streamline upwind Petrov Galerkin), GLS (Galerkin Least Squares) to name some (see, for example, [45, 107]). In particular, for the Navier-Stokes equations, the *grad-div* and SUPG stabilizations are the most popular, due to their strong consistency properties. As a matter of fact (see [95]), such stabilizations can be derived from two different viewpoints: (i) as a variational multiscale approach for pressure

subgrid models, (ii) a stabilization procedure of least squares type. In this logic, the choice of the stabilization parameters can be informed by physical considerations, such as viscous dissipation of energy. In particular, for some parameters $\delta_1, \delta_2 > 0$, the *grad-div* and SUPG stabilizations result in adding to the variational form in (2.14) the extra terms

$$s_{\mathrm{grad-div}}(\mathbf{u}_h, \mathbf{v}_h) = \delta_1 \, (\mathrm{div}\, \mathbf{u}_h, \mathrm{div}\, \mathbf{v}_h)$$

and

$$s_{\mathrm{SUPG}}([\mathbf{u}_h, p_h], [\mathbf{v}_h, q_h]) = \delta_2 \sum_{T_k \in \mathcal{T}_h} (\mathcal{L}(\mathbf{u}_h, p_h) - \mathbf{f}, \frac{h_k}{|\mathbf{w}_h|} \mathcal{L}_{ss}(\mathbf{v}_h, q_h)),$$

where $\mathcal{L}_{ss} = (\mathcal{L} - \mathcal{L}^*)/2$ denotes the skew-symmetric part (with respect to the L^2 inner product) of the Oseen differential operator

$$\mathcal{L}(\mathbf{u}_h, p_h) = \begin{bmatrix} \sigma \mathbf{u}_h - \mathrm{div}(2\, \nu\, \varepsilon(\mathbf{u}_h)) + (\mathbf{w}_h \cdot \nabla)\, \mathbf{u}_h + \nabla p_h \\ \mathrm{div}\, \mathbf{u}_h \end{bmatrix}.$$

2.4 Time discretization

For the time discretization of the semidiscrete problem (2.13), we consider the backward differences formulas (BDF) method. BDF are among the most used schemes for the discretization of parabolic problems, due to their θ-stability properties (see e.g. [35], pages 618-619). BDF is a family of implicit multistep methods. The convergence rate is $\mathcal{O}(\Delta t^p)$, where p denotes the number of steps. Only formulas with $p \leq 6$ are zero-stable.

We summarize here the main ideas of the method in the case of constant and variable step size. Let us consider the following system of ordinary differential equations

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}),\ t \in (0, T]. \tag{2.17}$$

In the constant step size case, the time interval $(0, T]$ is subdivided into N subintervals with a positive time step $\Delta t = T/N$. Then the equations (2.17) are collocated at the time level $t_{n+1} = (n+1)\Delta t$, and the time derivative $\frac{d\mathbf{y}}{dt}$ is approximated by a linear combination of the solution $\mathbf{y}_i = \mathbf{y}(t_i)$ at the previous time levels. So we have

$$\left.\frac{d\mathbf{y}}{dt}\right|_{t=t_{n+1}} \approx \frac{1}{\Delta t}\sum_{i=0}^{p+1} \alpha_i \mathbf{y}_{n+1-i}, \tag{2.18}$$

where α_i are computed from the first derivative of the polynomial interpolator of $\mathbf{y}(t)$ at the $p+1$ time levels t_{n+1}, t_n, ..., t_{n-p+1}.

This leads, at each time level, to the iterative solution of the (possibly non-linear) system

$$\frac{\alpha_0}{\Delta t}\mathbf{y}_{n+1} - f(t, \mathbf{y}_{n+1}) = -\frac{1}{\Delta t}\sum_{i=1}^{p+1} \alpha_i \mathbf{y}_{n+1-i}.$$

A good initial guess $\mathbf{y}^*_{n+1}$ is therefore mandatory to guarantee (or to accelerate) the convergence of the method.

Such initial guess can be provided by using an explicit method to compute $\mathbf{y}^*_{n+1}$ (predictor-corrector approach), or simply by time lagging ($\mathbf{y}^*_{n+1} = \mathbf{y}_n$). A predictor-corrector approach computes a more accurate guess, but incurs in computational and memory allocations overhead.

Polynomial in time extrapolation of the solution $\mathbf{y}(t)$ combines the benefits of the two approaches. In fact, one can use the values of the solution $\mathbf{y}(t)$ at the previous p time levels t_n, ..., t_{n-p+1} (already stored by the BDF method) to extrapolate the solution $\mathbf{y}(t)$ at time t_{n+1}, that is

$$\mathbf{y}^*_{n+1} = \sum_{i=1}^{p} \beta_i \mathbf{y}_{n+1-i}. \tag{2.19}$$

	first derivative coeff.					extrapolation coeff.			
p	α_0	α_1	α_2	α_3	α_4	β_1	β_2	β_3	β_4
1	1	1	–	–	–	1	–	–	–
2	$\frac{3}{2}$	-2	$\frac{1}{2}$	–	–	2	-1	–	–
3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$	–	3	-3	1	–
4	$\frac{25}{12}$	-4	3	$-\frac{4}{3}$	$\frac{1}{4}$	4	-6	4	-1

Table 2.1: Coefficients of the zero-stable BDF formulas for $p = 1, \ldots, 4$ in the case of constant step-size.

In Tab. 2.1 we write the coefficients α_i and β_i for the zero-stable BDF formulas up to the 4^{th} order of accuracy ($p = 1, 2, 3, 4$). We also recall that BDF-1 and BDF-2 are unconditionally stable, while higher order BDF formulas are conditionally stable (θ-stable).

For variable step size BDF formulas we rely on the so called *variable coefficient technique* (see e.g. [28, 29]). Such technique extends the polynomial interpolation approach to the case of non-constant time step sizes. The time interval $(0, T]$ is subdivided into subintervals with a positive time step Δt_k ($k \geq 1$), which may be different in size for each time level or selected adaptively during the simulation on the basis of some *a posteriori* error estimator. The time derivative and the extrapolation of the solution $\mathbf{y}$ at the time level $t_{n+1} = \sum_{k=1}^{n+1} \Delta t_k$ are approximated by

$$\left.\frac{d\mathbf{y}}{dt}\right|_{t=t_{n+1}} \approx \frac{1}{\Delta t_{n+1}} \sum_{i=0}^{p+1} \alpha_i^{(n+1)} \mathbf{y}_{n+1-i}, \quad \mathbf{y}_{n+1}^{*} = \sum_{i=1}^{p} \beta_i^{(n+1)} \mathbf{y}_{n+1-i}$$

where the coefficients $\alpha_i^{(n+1)}$ and $\beta_i^{(n+1)}$ are recomputed at each time level by Algorithm 2.1.

Stability bounds for these time-advancing methods as function of the step-size ratios $\Delta t_{j+1}/\Delta t_j$ are available in literature (see e.g. [28, 29]).

Input: $p \longrightarrow$ order of the BDF discretization
Input: $\Delta t^{(j)} = \Delta t_{n+1-j},\ j = 0, \ldots, p-1 \longrightarrow$ current and previous time step sizes
Output: BDF coefficients α_i, β_i

for $j = 0, \ldots, p-1$ **do**

$$\rho^{(j)} = \frac{\Delta t^{(0)}}{\sum_{i=0}^{j} \Delta t^{(i)}};$$

end

$$\alpha_0 = \sum_j \rho^{(j)};$$

// β_0 is not used for the extrapolation

for $j = 0, \ldots, p-1$ **do**

$$\mathcal{I}_j = \{i \in N \mid 0 \leq i \leq p-1,\ i \neq j\};$$

$$\beta_{j+1} = \left(1 - \frac{\prod_{i \in \mathcal{I}_j} \rho^{(i)}}{\rho^{(j)}}\right)^{-1};$$

$$\alpha_j = \rho^{(j)} \beta^{(j)};$$

end

Algorithm 2.1: Arbitrary order, variable time step BDF coefficients

2.4.1 Treatment of the non-linear convective term

Different choices are available for the treatment of the non-linear convective term in (2.13), namely the fully-implicit, semi-implicit, and fully-explicit treatments. The fully-implicit treatment of the convective term requires the solution at each time level of a non-linear system of equations by using Newton or Picard fixed point iterations. This approach does not affect the stability and accuracy of the time discretization, but the cost for time-step might be prohibitive in real applications [107]. On the contrary, fully-explicit treatments of the convective term, provides an approximation of the convective term by exploiting the values of the solution computed at the previous time levels, that is

$$\mathbf{u}_{n+1} \cdot \nabla \mathbf{u}_{n+1} \approx F(\mathbf{u}_n, \mathbf{u}_{n-1}, \ldots, \mathbf{u}_{n+1-p}).$$

Two common choices for $F(\mathbf{u}_n, \mathbf{u}_{n-1}, \ldots, \mathbf{u}_{n+1-p})$ are given by

$$F_1(\mathbf{u}_n, \mathbf{u}_{n-1}, \ldots, \mathbf{u}_{n+1-p}) = \mathbf{u}^*_{n+1} \cdot \nabla \mathbf{u}^*_{n+1} = \left(\sum_{i=1}^{p} \beta_i \mathbf{u}_{n+1-i} \right) \cdot \nabla \left(\sum_{i=1}^{p} \beta_i \mathbf{u}_{n+1-i} \right),$$

or

$$F_2(\mathbf{u}_n, \mathbf{u}_{n-1}, \ldots, \mathbf{u}_{n+1-p}) = (\mathbf{u} \cdot \nabla \mathbf{u})^*_{n+1} = \sum_{i=1}^{p} \beta_i \left(\mathbf{u}_{n+1-i} \cdot \nabla \mathbf{u}_{n+1-i} \right).$$

The latter expression was proposed in [75], and it provides a more accurate estimate of the convective term. It is worth noticing that fully-explicit methods, independently of the particular choice of F, reduce the stability region of the time discretization, by requiring the time step to satisfy the CFL condition

$$\Delta t \leq C \frac{h_k}{\|\mathbf{u}\|_{L^\infty(T_k)}}, \ \forall T_k \in \mathcal{T}_h, \tag{2.20}$$

being h_k diameter of the element T_k in the triangulation $\mathcal{T}_h$, and C a constant (not necessarily smaller or equal to 1). However, the conditional stability of the method

is balanced by the fact that the problem to be solved at each time step has a moderate computational cost. Indeed the discretized Navier-Stokes equations reduce to a symmetric generalized Stokes system of the form

$$\begin{cases} \frac{\alpha_0}{\Delta t} M\mathbf{U_{n+1}} + K\mathbf{U_{n+1}} + B^T P_{n+1} = \mathbf{f}_{n+1} - F(\mathbf{u}_n, \dots, \mathbf{u}_{n+1-p}) - \frac{1}{\Delta t}\sum_{i=1}^{p} \alpha_i \mathbf{U}_{n+1-i} \\ B\mathbf{U}_{n+1} = 0, \end{cases} \tag{2.21}$$

for which optimal preconditioners, such as the Cahouet-Chabard preconditioner [26], are known in literature.

In order to balance between stability and cost per time-step, we consider a semi-implicit approach, amounting to the following approximation:

$$\mathbf{u}_{n+1} \cdot \nabla \mathbf{u}_{n+1} \approx \mathbf{u}^*_{n+1} \cdot \nabla \mathbf{u}_{n+1} = \sum_{i=1}^{p} \beta_i (\mathbf{u}_{n+1-i} \cdot \nabla)\mathbf{u}_{n+1}. \tag{2.22}$$

For each time level t_{n+1}, we then solve an Oseen problem (2.14) of the form

$$\begin{cases} \frac{\alpha_0}{\Delta t} M\mathbf{U_{n+1}} + A_{n+1}\mathbf{U_{n+1}} + B^T P_{n+1} = \mathbf{f}_{n+1} - \frac{1}{\Delta t}\sum_{i=1}^{p} \alpha_i \mathbf{U}_{n+1-i} \\ B\mathbf{U}_{n+1} = 0, \end{cases} \tag{2.23}$$

where A_{n+1} is the advection-diffusion matrix relative to the variational form

$$a_{n+1}(\mathbf{u}_h, \mathbf{v}_h) = (\varepsilon(\mathbf{u}_h), \varepsilon(\mathbf{v}_h)) + \left(\mathbf{u}^*_{n+1} \cdot \nabla \mathbf{u}_h, \mathbf{v}_h\right),$$

and $\mathbf{u}^*_{n+1}$ is the wind velocity.

It is worth noticing that the semi-implicit approach does not affect the global accuracy of the time discretization even if it introduces an upper bound on the allowable time step size Δt for high order in time schemes. However this bound is less restrictive than (2.20), and in many applications it does not represent a real limitation since accuracy considerations require smaller Δt.

Efficient numerical methods for the solution of the linear system (2.23), which is of main interest for this thesis, will be addressed in Chapter 3.

2.5 A note on mass lumping for high order finite element

Mass lumping is a technique that consists in replacing the consistent finite element mass matrix M with a diagonal approximation $\hat{M}$. This technique has been often advocated for the stabilization of reaction dominated problems and for efficient solution of time dependent problems (see e.g [52, 107]). In addition, as we will emphasize in the following chapters, mass lumping is highly desirable also for efficient implementation of the algebraic splitting solvers and preconditioners presented in this thesis.

However, mass lumping techniques commonly used for linear or bilinear finite elements are not trivially extended to the higher order *inf-sup* compatible finite element discretizations (P_1^b, $P2$, or $Q2$) for the velocity field. As well known, the simple row condensation approach

$$\hat{M}_{ii} = \sum_j M_{ij}, \quad \hat{M}_{ij} = 0 \text{ if } i \neq j,$$

does not lead necessarily to a non-singular (nor positive definite) matrix.

Two main different approaches have been proposed in literature to obtain positive definite lumped mass matrices $\hat{M}$ with higher order finite elements (see [132]),

1. normalize the mass matrix diagonal so that mass conservation is guaranteed, e.g.

$$\hat{M}_{ii} = M_{ii} \frac{\sum_{i,j} M_{ij}}{\sum_j M_{jj}} \quad \hat{M}_{ij} = 0 \text{ if } i \neq j; \tag{2.24}$$

2. enforce the orthogonality among the finite element basis functions by using a discrete scalar product induced by *ad-hoc* quadrature rules.

Here we follow the latter approach where the error induced by the lumping can be rigorously estimated in terms of the quadrature error by applying the *Strang Lemma*.

2.5.1 Mass lumping and orthogonal finite element basis

Let us consider Lagrangian finite elements. $\{a_i\}$ will denote the set of degrees of freedom of the finite element space V_h, and $\{\phi_i\}$ will indicate the corresponding shape functions.

When assembling the finite element mass matrix, we assume the $L^2(\Omega)$ scalar product (ϕ_i, ϕ_j) to be replaced by the discrete inner product $(\phi_i, \phi_j)_h$,

$$(\phi_i, \phi_j)_h \equiv \sum_{K \in \mathcal{T}_h} \sum_{l} \omega_{K,l} \phi_i(\hat{a}_{K,l}) \phi_j(\hat{a}_{K,l}) \approx (\phi_i, \phi_j), \tag{2.25}$$

where $\omega_{K,l}$ and $\hat{a}_{K,l}$ are respectively the l-th weight and quadrature node for the element K.

Mass lumping is achieved if the basis functions ϕ_i and ϕ_j ($i \neq j$) are orthogonal for the discrete inner product $(\cdot, \cdot)_h$. The orthogonality condition for the discrete inner product is stated in [34], as recalled by the following well known result.

Lemma 2.5.1 A sufficient condition for the orthogonality of the basis function respect to the discrete scalar product (2.25) is that the degrees of freedom of V_h coincide with the quadrature nodes.

Indeed, if $i \neq j$, the product $\phi_i \phi_j$ necessarily vanishes at each quadrature point. For linear Lagrangian finite elements this condition is promptly obtained by taking the vertices of the unit simplex as quadrature nodes (trapezoidal formula). For higher order elements, stability and accuracy issues should be taken into account while enforcing condition of Lemma 2.5.1. In particular, the quadrature weights should be strictly positive to guarantee the spectral equivalence between the consistent (symmetric positive

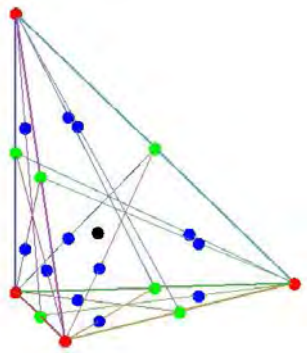


Figure 2.3: Degree of freedom of the modified second order finite element proposed in [92].

definite) mass matrix and the lumped one [34]. Moreover, when using P_k (respectively Q_k) finite element, the quadrature rule (2.25) should have degree of exactness at least $2k$ (respectively $2k$ for each space coordinate) to preserve the uniform V_h ellipticity and the accuracy of the space discretization [32, 50].

For example, in the case of P_2 finite elements no quadrature simultaneously satisfies both the orthogonality condition of Lemma 2.5.1 and the positivity constraint on the quadrature weights. In [92], a modified second order finite element has been proposed for the solution of the wave equations (see Fig. 2.3). The finite element and the associated quadrature rule satisfy both the orthogonality condition and stability constraints. Moreover, it does not affect the accuracy of the computation. However the high connectivity and number of degree of freedom of the element (23 in total, one per vertex, one per edge, three for face, and one barycentric) entail a huge cost both in terms of computational effort and memory allocation.

A possible trade-off is to apply a low order quadrature formula fulfilling Lemma 2.5.1 only to the mass matrix. Even though this may affect the order of the convergence of the discretization as stated by the Strang Lemma, the V_h ellipticity and stability of the discretization are preserved as long as the other terms of the problem are exactly integrated.

We investigate the accuracy of this approach for inf-sup compatible finite elements in the next subsection.

2.5.2 Accuracy of mass-lumped finite elements

Let us recall a well known result in the numerical integration literature (see e.g. [46], Lemma 8.4).

Lemma 2.5.2 Assume that $\{\mathcal{T}_h\}_{h>0}$ is a shape regular family of affine mesh which approximates a non empty, Lipschitz, compact subset Ω of $\mathbb{R}^d$. We consider a quadrature rule with degree of exactness $q > \frac{d}{2} - 1$ (which is trivially true for $d = 2, 3$). Let $\{\hat{a}_{K,l}\}$ and $\{\omega_{K,l}\}$ be the nodes and the weights of the quadrature formula on each element K. Let f be a L^1 function in Ω such that it belongs to $H^{q+1}(K)$ for each element K. Then, there exists a constant c independent of h, such that

$$\left| \int_\Omega f d\Omega - \sum_{K\in\mathcal{T}_h} \sum_l \omega_{K,l} f(\hat{a}_{K,l}) \right| \leq c h^{q+1} \left(\sum_{K\in\mathcal{T}_h} |f|^2_{H^{q+1}(K)} \right)^{\frac{1}{2}}. \tag{2.26}$$

In the sequel, we set $r \equiv q + 1$. Let $V = H^1(\Omega)$ and V_h a suitable finite dimension subspace of V, with $\|v_h\|_{V_h} = \|v_h\|_V$ for every $v_h \in V_h$. In the particular case of geometry discretization with tetrahedral meshes, we choose V_h to be the space of piecewise polynomial functions added by a bubble function vanishing on the boundaries. More precisely, we set for $k' > k$

$$V_h = \left\{ v_h \in H^1(\Omega) \mid \forall K \in \mathcal{T}_h \;\; v_h|_K \in \mathbb{P}_k + \mathbb{B}_{k'} \right\},$$

where $\mathbb{B}_{k'}$ is the set of bubble polynomial functions of degree k'.

Consider the generic problem: for a given f regular enough, find $u \in V$ solution to

$$a(u, v) \equiv (u, v) + b(u, v) = (f, v) \quad \forall v \in V \tag{2.27}$$

where $b(\cdot,\cdot)$ is a coercive bilinear form. We assume to integrate exactly the terms $b(u,v)$ and (f,v) with proper Gaussian quadratures. We replace the L^2 product with a quadrature formula $(\cdot,\cdot)_h$ fulfilling the assumptions of Lemma 2.5.2. This implies that the bilinear form $a(\cdot,\cdot)$ is uniformly coercive in $V_h \times V_h$ and the problem: find $u_h \in V_h$ s.t.

$$(u_h, v_h)_h + b(u_h, v_h) = (f, v_h) \quad \forall v_h \in V_h \tag{2.28}$$

is well posed.

By a direct application of the Strang Lemma (see e.g. [107]), we have the following Proposition. Set $m = \max(k', r)$.

Proposition 2.5.3 If $u \in H^{m+1}(\Omega)$, then the solution u_h of (2.28) fulfills the error estimate

$$\|u - u_h\|_V = \mathcal{O}(h^s),$$

being $s = \min(k, r + 1 - \min(r, k'))$.

Proof

Let us consider the following error term coming from the application of the Strang Lemma.

$$E_q = \inf_{v_h \in V_h} \sup_{w_h \in V_h} \frac{|a(v_h, w_h) - a_h(v_h, w_h)|}{\|w_h\|_V}.$$

In particular we select $v_h = \Pi_{V_h} u$ the polynomial of order k' projection of u on the V_h space. We have

$$E_q \leq \sup_{w_h \in V_h} \frac{|(\Pi_{V_h} u, w_h)_h - (\Pi_{V_h} u, w_h)|}{\|w_h\|_V}.$$

By exploiting the quadrature error formula given in Lemma 2.5.2 and the Cauchy-Schwarz inequality, we have

$$E_q \leq C h^r \|\Pi_{V_h} u\|_{H^r} \sum_{K \in \mathcal{T}_h} \frac{\|w_h\|_{H^r(K)}}{\|w_h\|_V}.$$

Since w_k is a polynomial of degree k', we have that $\|w_h\|_{H^r(K)} = \|w_h\|_{H^{\min(r,k')}(K)}$. Moreover, by the inverse inequality we have

$$\sum_{K\in\mathcal{T}_h} \frac{\|w_h\|_{H^{\min r,k'}(K)}}{\|w_h\|_V} \leq C_2 h^{-(\min(r,k')-1)} \sum_{K\in\mathcal{T}_h} \frac{\|w_h\|_{H^1(K)}}{\|w_h\|_V} \leq C_2 h^{-(\min r,k'-1)},$$

where C_2 is independent of h. We conclude therefore that the error induced by the quadrature is $\mathcal{O}(h^{r+1-\min(r,k')})$, so that the thesis follows from the Strang Lemma.

$\diamond$

From the previous result we conclude that the quadrature error is not affecting the finite element accuracy r for $r - \min(r, k') \geq k - 1$.

In Tab. 2.3, we particularize the theoretical convergence estimated rates in Proposition 2.5.3 to the case of P_1^b ($k = 2$, $k' = 4$, $r = 3$) and P_2^b ($k = 2$, $k' = 4$, $r = 4$) finite element discretizations. Notice that with similar arguments we can prove for hexahedral elements Q_k that the accuracy is in this case $\min(k, r + 1 - \min(r, k))$. Numerical results in Fig. 2.4 confirm the theoretical estimates in Tab. 2.3, in the case of a simple diffusion reaction problem. With elements P_1^b and the 5 nodes formula of Tab. 2.2, the linear accuracy is preserved, while P_2^b elements with the 11 nodes formula have only order 1. For Hexahedral elements, Q_2 with the quadrature formula with 27 nodes reported in Tab. 2.2 preserve second order accuracy (the quadrature error being of order $r + 1 - \min(r, k) = 3$).

number of points	degree of exactness z	nodes and weights
5	2	$(1,0,0,0)$ [4] mes$(K)\frac{1}{20}$ $(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ [1] mes$(K)\frac{4}{5}$
11	3	$(1,0,0,0)$ [4] mes$(K)\frac{1}{60}$ $(\frac{1}{2},\frac{1}{2},0,0)$ [6] mes$(K)\frac{1}{15}$ $(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ [1] mes$(K)\frac{8}{15}$
27	3	vertexes [8] mes$(K)\frac{1}{6^3}$ edges mid points [12] mes$(K)\frac{4}{6^3}$ faces barycenters [6] mes$(K)\frac{4^2}{6^3}$ element barycenter [1] mes$(K)\frac{4^3}{6^3}$

Table 2.2: Properties of the quadrature rules used to perform mass lumping with P_1^b and P_2^b finite elements on a tetrahedron (rows 1 and 2) and with Q_2 elements on a hexahedron (row 3). The quadrature nodes are given in barycentric coordinates (for quadrature rules on a tetrahedron), and the number of possible permutations are inside the square brackets ($[\cdot]$). The weights ω include the measure of the element K ($\mathrm{mes}(K)$).

FE Space	Quadrature rule	$\inf_{v_h \in V_h} \|u - v_h\|_{V_h}$	$\sup_{w_h \in V_h} \frac{\lvert a(v_h,w_h) - a_h(v_h,w_h)\rvert}{\|w_h\|_{V_h}}$
P_1^b	5v in Tab. 2.2	$\mathcal{O}(h)$	$\mathcal{O}(h)$
P_2^b	11v in Tab. 2.2	$\mathcal{O}(h^2)$	$\mathcal{O}(h)$
Q_2	27v in Tab. 2.2	$\mathcal{O}(h^2)$	$\mathcal{O}(h^3)$

Table 2.3: Comparison between the best subspace approximation error and the error introduced by the mass lumping. Proposition 2.5.3 shows that the convergence rate of finite elements without mass lumping is not affected for P_1^b and Q_2 elements, while it is reduced for P_2^b elements.

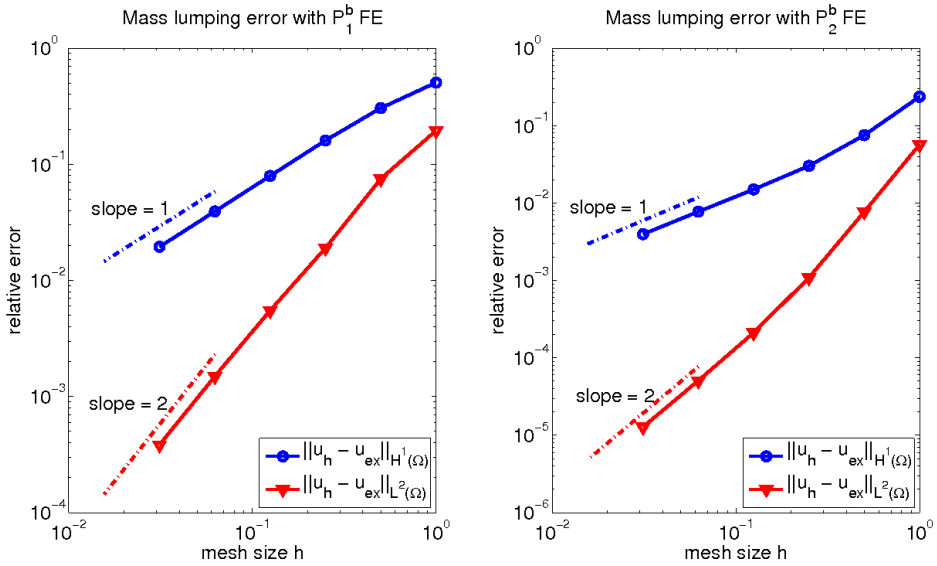


Figure 2.4: Relative error of the numerical solution of a diffusion reaction problem in the $H^1(\Omega)$ (blue circles) and $L^2(\Omega)$ (red triangles) norm as a function of the mesh size h. For P_1^b elements (on the left) the subspace approximation errors dominates the mass lumping error and we observe optimal rate of convergence ($\mathcal{O}(h)$ in the $H^1(\Omega)$ norm and $\mathcal{O}(h^2)$ in the $L^2(\Omega)$ norm). For P_2^b elements (on the right) the mass lumping error leads to suboptimal rate of convergence ($\mathcal{O}(h)$ in the $H^1(\Omega)$ norm and $\mathcal{O}(h^2)$ in the $L^2(\Omega)$ norm).

3 Algebraic splittings and block preconditioners

In this Chapter we address the numerical solution of the non-symmetric saddle-point problem (2.23), arising after space and time discretization and linearization of the unsteady Navier-Stokes equations. Two different, but related, approaches are described in this chapter: (i) fractional steps methods that decouple the computation of velocity and pressure, (ii) block preconditioners for the coupled velocity and pressure saddle point problem. We first provide a general review of the algebraic splitting methods in the framework of inexact LU factorizations of the saddle-point matrix in (2.23) (Section 3.1), then we focus on the construction and analysis of a class of pressure-corrected algebraic splittings that guarantee arbitrary high order accuracy in time (Section 3.2, 3.3 and 3.4). Such schemes, called in the following High Order Yosida (HOY), fulfil exactly the discrete momentum equation while introduce a consistency error in the conservation of mass that decays polynomially with respect to the time-step. However, such schemes are only conditionally stable, possibly reducing the absolute stability region of the BDF formulas. We conclude the chapter by investigating the use of such algebraic splittings as preconditioners of the saddle-point problem and comparing them with state of the art preconditioners, such as the Cahouet-Chabard and Least Squares Commutator preconditioners (Section 3.5).

3.1 Velocity-pressure splittings methods

Discretization (in time and space) and the semi-implicit linearization (2.22) of the unsteady Navier-Stokes (2.10) yield to the following algebraic system to be solved at time level $n+1$:

$$\mathcal{A}\mathbf{y}_{n+1} = \mathbf{b}_{n+1}, \quad n = 0, \ldots, N-1, \tag{3.1}$$

where

$$\mathcal{A} = \begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix}, \ \mathbf{y}_{n+1} = \begin{bmatrix} \mathbf{U}_{n+1} \\ P_{n+1} \end{bmatrix} \ \mathbf{b}_{n+1} = \begin{bmatrix} \widehat{\mathbf{f}}_{n+1} \\ \mathbf{0} \end{bmatrix}. \tag{3.2}$$

Here the momentum matrix $C \in \mathbb{R}^{N_\mathbf{u} \times N_\mathbf{u}}$ has the form

$$C = \frac{\alpha_0}{\Delta t} M + A, \ A = K + N, \tag{3.3}$$

where M is the velocity mass matrix, K is the discretization of the viscous terms, N stems from the Picard-linearization and discretization of the convective term. If $N = 0$, then matrix $A \equiv K$ is symmetric positive definite, and (3.1) reduces to the solution of a generalized Stokes problem.

The right hand side

$$\widehat{\mathbf{f}}_{n+1} = \mathbf{f}_{n+1} + \frac{M}{\Delta t} \sum_{i=1}^{p} \alpha_i \mathbf{U}_{n+1-i}$$

collects the contributions of fluid body forces (gravity, for example), boundary conditions, and discretization of the time derivative. Here p denotes the order of the BDF formula used for the time discretization.

System (3.1) typically features large dimensions and bad conditioning properties. Some popular strategies are based in subdividing the computation of velocity and pressure into successive solution of smaller problems. Such strategies consist in using fractional-step methods of differential or algebraic type. In the former, the splitting

is based either on physical considerations (see e.g. [58]), or on the Helmholtz decomposition of any vector field into a solenoidal and an irrotational part. These methods are called *projection methods*, and the most famous one is the Chorin-Temam scheme [30, 116]. The algorithm consists of two stages in which the computation of velocity and pressure are split at the level of the differential equation. In the first stage, an intermediate velocity $\widehat{\mathbf{u}}$ that does not satisfy the incompressibility constraint is computed at each time step. In the second, the pressure is used to project $\widehat{\mathbf{u}}$ onto a space of divergence-free velocity field to get the next update of velocity $\mathbf{u}$ and pressure p (see e.g. [107], [103]). More specifically, the Chorin-Temam scheme reads

$$\begin{cases} \frac{\alpha_0 \widehat{\mathbf{u}}_{n+1}}{\Delta t} - \nu \Delta \mathbf{u}_{n+1} + \mathbf{u}_{n+1} \nabla \mathbf{u}_{n+1} = \widehat{\mathbf{f}}_{n+1} \\ \Delta p_{n+1} = \frac{1}{\Delta t} \text{div } \widehat{\mathbf{u}}_{n+1} \\ \mathbf{u}_{n+1} = \widehat{\mathbf{u}}_{n+1} - \Delta t \ \nabla p_{n+1}. \end{cases} \tag{3.4}$$

We remark that the accuracy of the projection depends strongly on the boundary conditions chosen for the pressure differential problem $(3.4)_2$. In particular, if Dirichlet boundary conditions are prescribed on the whole boundary $\partial\Omega$ for the velocity unknown, as it was assumed in the original derivation of the scheme, Neumann boundary conditions should be imposed for $(3.4)_2$. However, if different sets of velocity boundary conditions are considered, it is not evident which kind of conditions should be imposed for the pressure. The main drawback of the *Chorin-Temam scheme* is therefore that pressure boundary conditions might not be in accordance with the fluid-dynamics, leading to inaccurate solutions at the boundary of the domain.

To overcome this problem, algebraic splittings methods decouple the velocity and pressure computation at the linear system level (3.1) instead that at the differential level. Following this approach, boundary conditions are directly incorporated in the discretized operator and no additional boundary conditions have to be selected. Such algebraic decomposition (or *splitting*) could be performed either by a sum of simpler

matrices (see e.g. [86, 131]) or by a product of block-triangular matrices. In this prospective, in [100] the Chorin-Temam method is revisited as an approximate (or *inexact*) block LU factorization of the matrix $\mathcal{A}$. Various works followed [100] in which new families of (algebraic) fractional step methods, with good stability and accuracy properties but not necessarily provided with a differential counterpart, were proposed and investigated (for e.g. [36, 75, 105, 106, 125]).

We now provide a unified derivation of those methods. Let us consider the block LU factorization

$$\mathcal{A} = \begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} C & 0 \\ B & \Sigma \end{bmatrix} \begin{bmatrix} I & C^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{3.5}$$

Here $\Sigma = -BC^{-1}B^T$ denotes the pressure Schur complement matrix that needs to be solved in order to compute the pressure unknown. While for the Steady Stokes problem Σ is spectrally equivalent to the pressure mass matrix [45], in the unsteady Navier-Stokes problem Σ is badly conditioned, the condition number increasing when the time step is decreased (see e.g. [22]). Also, the presence of conjugate pairs of complex eigenvalues in the spectrum of Σ due to the non-symmetry of the convective term represents an additional challenge for the iterative solution of linear system involving Σ. Moreover, Σ is not explicitly computable, due to the presence of the C^{-1} factor as well as to the *fill-in* associated with the matrix products.

Since the computational cost of solving the pressure Schur complement Σ can be very high, an exact block LU factorization may not be a viable approach for problems of practical interest. To avoid this computational bottleneck, practical splitting schemes therefore consider an inexact factorization

$$\widehat{\mathcal{A}} = \begin{bmatrix} C & 0 \\ B & -BFB^T \end{bmatrix} \begin{bmatrix} I & GB^T \\ 0 & I \end{bmatrix}, \tag{3.6}$$

being F and G suitable approximations of C^{-1}.

The solution of the problem at each time step then consists of the following linear systems

$$\mathrm{L}-\mathrm{step}: \begin{cases} C\widehat{\mathbf{U}} = \mathbf{f}_1 \\ -BFB^T\widehat{P} = \mathbf{f}_2 - B\widehat{\mathbf{U}} \end{cases}, \qquad \mathrm{U}-\mathrm{step}: \begin{cases} P = \widehat{P} \\ \mathbf{U} = \widehat{\mathbf{U}} - GB^T P. \end{cases} \tag{3.7}$$

Different methods can be derived by an opportune choice of the matrices F and G. We refer to [106] for additional details and for the relationships between inexact factorization and differential splittings. Here we limit to observe that the splitting error is given by

$$\mathcal{A} - \widehat{\mathcal{A}} = \begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} - \begin{bmatrix} C & CGB^T \\ B & -BFB^T + BGB^T \end{bmatrix} = \begin{bmatrix} 0 & (I - CG)B^T \\ 0 & B(G - F)B^T \end{bmatrix}. \tag{3.8}$$

In [125] it is suggested that possible choices for F and G can be obtained by truncating the Neumann expansion of the matrix C. Letting I be the $N_{\mathbf{u}} \times N_{\mathbf{u}}$ identity matrix, we formally write

$$C^{-1} = \frac{\Delta t}{\alpha_0}\left(I + \frac{\alpha_0}{\Delta t}M^{-1}A\right)^{-1} M^{-1} = \frac{\Delta t}{\alpha_0}\sum_{k=0}^{+\infty}(-\frac{\Delta T}{\alpha_0}M^{-1}A)^{k-1}M^{-1}. \tag{3.9}$$

If we denote with H the first order truncation of the Neumann expansion,

$$H = \frac{\Delta t}{\alpha_0}M^{-1}, \tag{3.10}$$

then the algebraic Chorin-Temam scheme (ACT) in [100] is obtained by selecting $F = G = H$ in (3.6), while the Yosida scheme (YOS) in [105, 106] is obtained by taking $F = H$ and $G = C^{-1}$. The two schemes feature the same L-step in (3.7) and approximate

the pressure Schur complement Σ with the discrete laplacian matrix

$$S = -BHB^T. \tag{3.11}$$

The YOS method differs from ACT basically by fulfilling the discrete momentum equation, while the ACT method guarantees discrete conservation of mass (as can be immediately observed by direct substitution of the matrices F and G defined above in (3.8)). For this reason, the YOS method is usually preferred in fluid-structure-interaction problems, where an accurate approximation of the fluid stresses at the interface with the solid is mandatory.

As a matter of fact, such splittings achieve a consistent reduction in the computational effort without affecting the stability of the algorithm. Moreover, if a first order time discretization scheme is adopted, the splitting error does not reduce the overall accuracy (see [105]).

The crucial point of extending the above result to more accurate in time discretizations was addressed for the first time in [66] and [125], using different but complementary approaches. While in [66] the authors advocate the use incremental schemes, based on numerical extrapolation of the pressure unknown, to increase the order of the splitting (cfr. Section 3.1.1), in [125] the author proposes unconditionally stable high order splitting by accounting for more terms in the Neumann expansion of C^{-1}. More specifically, instead of using the first order truncation H in the ACT or YOS schemes, the author defines the matrices $H_j = \sum_{k=0}^{j}(-\frac{\Delta T}{\alpha_0}M^{-1}A)^k M^{-1}$, being $j+1$ the number of terms retained in the expansion (3.9). Clearly $H_0 \equiv H$.

The following results in [125] generalize the stability analysis of the ACT and HOY schemes applied to the generalized Stokes problem ($A \equiv K$) to higher order splittings in which H_j ($j \geq 0$) is used instead of H.

Proposition 3.1.1 If the momentum matrix C is symmetric positive definite (s.p.d.),

then the matrices $S_j = -BH_jB^T$ are unconditionally s.p.d. for j even (i.e. s.p.d. for all Δt), and conditionally s.p.d. for j odd (i.e. s.p.d. for Δt sufficiently small).

Proposition 3.1.2 Suppose that the momentum matrix C is symmetric positive definite, and consider the inexact block LU factorization (3.6) built according to the ACT or YOS strategies (i.e. $F = G = H_j$ and $F = H_j,\ G = C^{-1}$ respectively). Then the splitting scheme induced by the factorization is unconditionally stable (in time) if j is even.

The computational bottleneck in the high order schemes so constructed is the solution of the pressure approximated Schur complements S_j. Indeed for problems of practical interest, the numerical solution of linear systems involving the high order approximated Schur complements S_j is not a viable approach. On the one hand S_j may not be explicitly available due to memory constraint (fill-in in the matrix-matrix multiplications), and on the other hand the construction of efficient preconditioners for matrix-free Krylov methods involving S_j may be cumbersome.

Even if the practical relevance of these schemes is limited, they represent an important step towards the construction of the more computationally efficient pressure corrected schemes [54, 55, 56, 112].

Pressure corrected schemes are based on the observation that at each time-step, the final pressure P of the inexact splitting scheme in (3.7) coincides with the intermediate pressure $\widehat{P}$ and it therefore induces a difference in the treatment of velocity and pressure deteriorating the accuracy of the scheme.

In [112] the authors consider the following *modified* inexact factorization of $\mathcal{A}$,

$$\widehat{\mathcal{A}} = \begin{bmatrix} C & 0 \\ B & -BFB^T \end{bmatrix} \begin{bmatrix} I & GB^TZ \\ 0 & Q \end{bmatrix} = \begin{bmatrix} C & CGB^TZ \\ B & B\left(GB^TR - FB^TQ\right) \end{bmatrix}, \tag{3.12}$$

where the matrices Z and Q are square $N_p \times N_p$ matrices chosen to reduce in some

sense the splitting error $\mathcal{A} - \widehat{\mathcal{A}}$.

Two new splittings (ACT-PC and YOS-PC) are proposed in [112] extending, respectively, the ACT and YOS approaches to the modified factorization (3.12).

More specifically, in the ACT-PC scheme ($F = G = H$) the discrete mass conservation is enforced by selecting $R = Q$, while in the YOS-PC scheme ($F = H$, $G = C^{-1}$) the discrete momentum equation is fulfilled exactly with the choice $R = I_{N_p}$ (see Algorithm 3.1 and 3.2 for their precise definition).

In fact, a simple computation gives

$$\mathcal{A} - \widehat{\mathcal{A}}_{\mathrm{ACT-PC}} = \begin{bmatrix} 0 & B^T - CHB^TQ \\ 0 & 0 \end{bmatrix}$$

and

$$\mathcal{A} - \widehat{\mathcal{A}}_{\mathrm{YOS-PC}} = \begin{bmatrix} 0 & 0 \\ 0 & SQ - \Sigma \end{bmatrix}.$$

It is worth to note that the correction matrix Q proposed in [112] is the same for both the ACT-PC and YOS-PC schemes, and in particular we have

$$Q = (BHCHB^T)^{-1}S.$$

The application of the pressure correction step $QP = \hat{P}$ therefore entails a linear solve with the scaled discrete laplacian matrix S and a few other matrix-vector multiplications.

Let us summarize here the main results concerning the stability and convergence of those methods, referring to [112] for the detailed analysis.

Proposition 3.1.3 Consider the Stokes problem discretized with BDF formulas. Then

1. the ACT-PC method is unconditionally stable and it introduces a first order in time

splitting error, that is $\|\mathcal{A} - \widehat{\mathcal{A}}_{\mathrm{ACT-PC}}\| = \mathcal{O}(\Delta t)$;

2. the YOS-PC method is conditionally stable (i.e. for Δt small enough) and it introduces a third order in time splitting error, that is $\|\mathcal{A} - \widehat{\mathcal{A}}_{\mathrm{YOS-PC}}\| = \mathcal{O}(\Delta t^3)$.

It is worth to notice that on the one hand pressure correction does not improve the order of accuracy of the ACT schemes, even if the magnitude of the error is reduced. This is mostly due by the fact that ACT-PC still uses the intermediate pressure in the computation of the end-of-step velocity (see Algorithm 3.1). On the other hand, the YOS-PC method trades the unconditional stability of the YOS scheme for a higher order of accuracy. In the next section we will present a possible approach to obtain arbitrary order pressure corrections for the Yosida scheme.

Input: $\mathbf{b}_1$, $\mathbf{b}_2$ the right hand sides for the momentum and continuity equation, respectively
Output: $\mathbf{U}$, P the end of step velocity and pressure

$C\widehat{\mathbf{U}} = \mathbf{b}_1$;
$S\widehat{P} = B\widehat{\mathbf{U}} - \mathbf{b}_2$;
$QP = \widehat{P}$;
$\mathbf{U} = \widehat{\mathbf{U}} - HB^T\widehat{P}$;

Algorithm 3.1: Pressure corrected algebraic Chorin-Temam (ACT-PC) splitting method

Input: $\mathbf{b}_1$, $\mathbf{b}_2$ the right hand sides for the momentum and continuity equation, respectively
Output: $\mathbf{U}$, P the end of step velocity and pressure

$C\widehat{\mathbf{U}} = \mathbf{b}_1$;
$S\widehat{P} = B\widehat{\mathbf{U}} - \mathbf{b}_2$;
$QP = \widehat{P}$;
$C\mathbf{U} = \mathbf{b}_1 - B^T P$;

Algorithm 3.2: Pressure corrected Yosida (YOS-PC) splitting method

3.1.1 Incremental formulation of splitting methods

A popular variant to the splitting methods presented above (both at the differential and algebraic level) often advocated is the so-called *incremental* formulation (see [62, 66, 102]). Let us write at each time step

$$P_n = \delta P_n + P_n^*$$

where P_n^* is an extrapolation of P_n based on the previous time steps, such that

$$P_n - P_n^* = \mathcal{O}(\Delta t^s), \mathbb{N} \ni s \geq 1. \tag{3.13}$$

More explicitly, we have

$$P_n^* = \sum_{i=1}^{s} \beta_i P_{n-i},$$

where the coefficients β_i are given in Tab. 2.1 (for the constant time step case) or recomputed at each time level by Algorithm 2.1 (for the variable time step case). For $s = 0$ (non incremental case) we keep the same notation for simplicity, with the assumption that $P_n^* = 0$. Then, we write

$$\begin{bmatrix} C & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \mathbf{U}_n \\ P_n \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{f}}_n \\ \mathbf{0} \end{bmatrix} \Rightarrow \begin{bmatrix} C & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \mathbf{U}_n \\ \delta P_n \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{f}}_n - B^T P_n^* \\ \mathbf{0} \end{bmatrix}. \tag{3.14}$$

Previous works on splitting methods for the solution of Navier-Stokes equations argue that solving for incremental pressure has better convergence properties than those of the non-incremental one, and that it does not affect the overall stability of the method. In particular, in [63] the authors note that the incremental method gives stable velocity and pressure fields under minimal hypotheses, provided that the finite element discretization fulfils the *inf-sup* condition. In [112], the authors provide numerical evidence of good stability and accuracy properties of the ACT-PC splitting (Algorithm

3.1 in Section 3.1) when coupled with a second order in time incremental approach. Finally, in [66], the authors suggest the incremental method as a valid and computationally convenient alternative to using high order matrix factorizations. However, they notice that incremental methods may accumulate round-off errors, in the case of long time simulations. For this reason they provide an alternative formulation in which they correct the end of step pressure to take into account the accumulation errors.

In Chapter 4 we will provide a detailed analysis of the convergence properties of the incremental pressure corrected algebraic splittings, and we will propose a time adaptive procedure based on these methods.

3.2 The high order Yosida splitting

A general framework for computing arbitrary order pressure correction for the Yosida scheme was proposed in [56] and analyzed in [55, 126].

As noted above, when we replace $\mathcal{A}$ with $\widehat{\mathcal{A}}$, we introduce a splitting error that is however confined to the continuity equation, as it is highlighted by the explicit form of $\widehat{\mathcal{A}}$,

$$\widehat{\mathcal{A}} = \begin{bmatrix} C & O \\ B & S \end{bmatrix} \begin{bmatrix} I_{N_u} & C^{-1}B^T \\ O & Q \end{bmatrix} = \begin{bmatrix} C & B^T \\ B & SQ - \Sigma \end{bmatrix}. \tag{3.15}$$

An exact factorization would be yielded by the choice $Q_{\text{ex}} = S^{-1}\Sigma$. This choice is however unfeasible, since it still requires the inversion of the Schur complement Σ. The high order Yosida schemes (HOY in the following) stem from the choice $Q = Q_q$ where q is a parameter related to the time accuracy of the method, such that

$$\|\Sigma - SQ_q\| = \mathcal{O}(\Delta t^{f(q)}) \tag{3.16}$$

where $f(q)$ is an appropriate (linear) function of q.

More specifically, two approaches are possible to build the matrices Q_q:

1. to construct a sequence of matrices $\{Q_q\}_q$ to approximate the exact correction matrix $Q_{\rm ex}$ by exploiting the Neumann expansion of Σ;

2. to consider the sequence of their inverses $\{Q_q^{-1}\}_q$ to directly approximate $Q_{\rm ex}^{-1}$ by exploiting the Neumann expansion of Σ^{-1}.

The second approach is to be preferred since it leads to more accurate and efficient algorithms [55].

By recalling the definition of the momentum matrix $C = H^{-1} + A$ in (3.3), we formally write the Neumann expansion of the pressure Schur complement as follows:

$$\Sigma = -BC^{-1}B^T = -B(I + HA)^{-1}HB^T = -\sum_{k\geq 0} B(-HA)^k HB^T. \tag{3.17}$$

We remark that such expansion is convergent only if the spectral radius $\rho(HA)$ is less than 1, implying the condition $\nu\Delta t = \mathcal{O}(h^2)$. However, such condition is not strictly required for the stability of the pressure corrected splitting schemes [55].

By setting

$$D_k = B(-HA)^k HB^T = \mathcal{O}(\Delta t^{k+1}),\ k \geq 0 \tag{3.18}$$

we have $\Sigma = -\sum_{k\geq 0} D_k$.

Next, noticing that $S = -BHB^T = -D_0$, we write

$$\Sigma^{-1} = \left(-\sum_{k\geq 0} D_k\right)^{-1} = \left(S - \sum_{k\geq 1} D_k\right)^{-1} = \left(I - S^{-1}\sum_{k\geq 1} D_k\right)^{-1} S^{-1}. \tag{3.19}$$

Now, let us denote $R = S^{-1}\sum_{k\geq 1} D_k$, for which $||R|| = \mathcal{O}(\Delta t)$. For Δt small enough such that the spectral radius $\rho(R) < 1$, we have

$$\Sigma^{-1} = (I - R)\, S^{-1} = \sum_{k\geq 0} R^k S^{-1}. \tag{3.20}$$

Consequently, we have $Q_{ex}^{-1} = \Sigma^{-1}S = \sum_{k=0}^{q} R^k + \sum_{k\geq q+1} R^k = I + \sum_{k=1}^{q} R^k + \mathcal{O}(\Delta t^{q+1})$.

A natural approximation of Q_{ex}^{-1} arises by neglecting the higher order term (h.o.t.) on the right hand side and taking $Q_{ex}^{-1} \approx \sum_{k=0}^{q} R^k$. However, as pointed out in [126], this is still unfeasible since R and its powers are series. It is also non-optimal for computational purposes, since R^k even for $k \leq q$ still retains some terms of order higher than q. In fact, terms of R^k can be re-arranged on the basis of their dependence on Δt [126], as we explicitly illustrate (for the case $q = 3$) in the table below.

	$\mathcal{O}(\Delta t)$	$\mathcal{O}(\Delta t^2)$	$\mathcal{O}(\Delta t^3)$	h.o.t.
R	$S^{-1}D_1$	$S^{-1}D_2$	$S^{-1}D_3$	$\mathcal{O}(\Delta t^4)$
R^2		$(S^{-1}D_1)^2$	$S^{-1}D_1S^{-1}D_2 + S^{-1}D_2S^{-1}D_1$	$\mathcal{O}(\Delta t^4)$
R^3			$(S^{-1}D_1)^3$	$\mathcal{O}(\Delta t^4)$
	$\widehat{R}_1$	$\widehat{R}_2$	$\widehat{R}_3$	

Then, following the columnwise notation introduced in the table and setting $\widehat{R}_0 = R^0 = I$, we have $\sum_{k=0}^{q} \widehat{R}_k = \sum_{k=0}^{q} R^k + \text{h.o.t.}$

A viable approximation of Q_{ex} is given therefore by

$$Q_{ex}^{-1} \approx Q_q^{-1} = \sum_{k=0}^{q} \widehat{R}_k. \tag{3.21}$$

Denoting by z_k the vectors such that $\widehat{R}_k z_k = \widehat{P}$, we have $P = \sum_{k=0}^{q} z_k$. Application of the sequence of matrices $\widehat{R}_k$ can be rearranged in a hierarchical way. Indeed, let us

consider, as before, the case $q = 3$. Set $z_0 = \widehat{P}$, then

$$\begin{aligned} z_1 &= S^{-1}D_1\widehat{P} \qquad \Rightarrow \qquad Sz_1 = D_1 z_0 \\ z_2 &= \left(S^{-1}D_2 + (S^{-1}D_1)^2\right)\widehat{P} \qquad \Rightarrow \qquad Sz_2 = D_2 z_0 + D_1 z_1 \\ z_3 &= \left(S^{-1}D_3 + S^{-1}D_2S^{-1}D_1 + S^{-1}D_1S^{-1}D_2 + (S^{-1}D_1)^3\right)\widehat{P} \quad \Rightarrow \\ & \qquad\qquad\qquad\qquad\qquad\qquad Sz_3 = D_3 z_0 + D_2 z_1 + D_1 z_2. \end{aligned} \tag{3.22}$$

More accuracy is obtained by solving more systems which, however, all entail the same matrix S. The efficiency of the method then strongly relies upon the easiness of solution for the matrix S. In order to preserve sparsity of S we resort to the mass lumping techniques introduced in Section 2.5 to approximate the velocity mass matrix M. For some finite elements (e.g. P1b on tetrahedra and Q2 on hexahedra) we proved, in fact, that mass lumping can be obtained by using apposite inexact quadrature rules for the mass matrix assembly, without jeopardizing the overall accuracy of the spacial discretization. Whenever possible, for small scale problems, direct methods for solving S should then be recommended (for instance in 2D, by exploiting a QR factorization). For large scale methods, recycling Krylov subspace solvers and algebraic multigrid preconditioners can be used for fast and efficient solution of linear systems involving S. More details are provided in Chapter 5.

3.3 Algorithmic form of High Order Yosida schemes

The practical implementation of the high order pressure corrected Yosida method requires well designed data structures in order to efficiently compute the pressure corrections for a generic order q. We address here some possible options that allow a compact implementation of the method. Let us introduce the following structures. Let Z be the matrix with N_p rows and $q + 1$ columns, such that the generic column i (where i starts

from 0 as in C/C++ syntax) corresponds to the vector $\mathbf{z}_i$,

$$\mathtt{Z} = [\mathbf{z}_0, \mathbf{z}_1, \ldots,]$$

Let moreover Z3 be the three index array (where the first index ranges from 0 to $N_u - 1$, the second and third ones from 0 to $q - 1$) such that

$$\begin{aligned}
&\mathtt{Z3}(0 : N_u - 1, 0, i) = -HAHB^T\mathbf{z}_i, \\
&\mathtt{Z3}(0 : N_u - 1, j, i) = -HA \times \mathtt{Z3}(0 : N_u - 1, j - 1, i), \quad j \geq 1
\end{aligned}$$

Finally, we introduce the three index array BZ3 such that $\mathtt{BZ3} = B \times \mathtt{Z3}$. A visual representation of BZ3 is given by the following $q \times q$ matrix,

$$\mathtt{BZ3} = \begin{bmatrix}
D_1\mathbf{z}_0 & D_2\mathbf{z}_0 & D_3\mathbf{z}_0 & \ldots & D_q\mathbf{z}_0 \\
D_1\mathbf{z}_1 & D_2\mathbf{z}_1 & D_3\mathbf{z}_1 & \ldots & D_q\mathbf{z}_1 \\
D_1\mathbf{z}_2 & D_2\mathbf{z}_2 & D_3\mathbf{z}_2 & \ldots & D_q\mathbf{z}_2 \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
D_1\mathbf{z}_{q-1} & D_2\mathbf{z}_{q-1} & D_3\mathbf{z}_{q-1} & \ldots & D_q\mathbf{z}_{q-1}
\end{bmatrix}, \tag{3.23}$$

where each entry is a vector (associated with the first index of the array).

With this data structure, the implementation of the pressure correction steps is given by Algorithm 3.3.

3.4 Analysis of the pressure corrected splittings

In this section we will recall the main results concerning the consistency, stability and convergence of the pressure corrected schemes. We will consider here mostly the non incremental formulation of these schemes, referring to Chapter 4 and [112, 128] for the incremental one. The analysis of the non incremental formulation has been carried

Input: u intermediate (non-divergence free) velocity
Output: pressure corrected computed pressure

```
Z[0] = solve(S, B * u);
for i = 0; i < q; ++i do
    Z3[0, i] = (−H * A) * H * B^T * Z[i];
    BZ3[0, i] = B * Z3[0, i];
    rhs = BZ3[0, i];
    for j = 1; j < i + 1; ++j do
        Z3[j, i − j] = (−H * A) * ZZ[j − 1, i − j];
        BZ3[j, i − j] = B * Z3[j, i − j];
        rhs += BZ3[j, i − j];
    end
    Z[i + 1] = solve(S, rhs);
end
pressure = sum(Z);
```

Algorithm 3.3: Implementation of the arbitrary order pressure correction algorithm. `solve(Matrix,RightHandSide)` gives the solution `x` of the system `Matrix*x=RightHandSide`.

out in [55] limited to the case $q = 0, 1, 2$, and extended to the case of arbitrary pressure corrections q in [126].

3.4.1 Non-singularity and consistency

The complete consistency analysis for the arbitrary order pressure correction schemes was given in [126] and it is summarized in the proposition below.

Proposition 3.4.1 For any $q \in \mathbb{N}$

1. the matrices Q_q are non-singular for Δt small enough;

2. the consistency error is such that

$$\|\Sigma - SQ_q\| = \mathcal{O}\left(\Delta t^{q+2}\right), \ \forall q \geq 0. \tag{3.24}$$

For the sake of completeness, let us provide a sketch of the prove of the proposition above. Since $Q_q = \left(\sum_{k=0}^{q} \widehat{R}_k\right)^{-1}$, a sufficient condition for the non-singularity of

Q_q is that the eigenvalues of $\sum_{k=0}^{q} \widehat{R}_k$ are bounded away from 0 and ∞. By letting $r_k = \|\widehat{R}_k\| = \mathcal{O}(\Delta t^k)$ for $k > 0$, a straightforward computation gives

$$\rho\left(\sum_{k=1}^{q} \widehat{R}^k\right) \leq \left\|\sum_{k=1}^{q} \widehat{R}^k\right\| \leq \sum_{k=1}^{q} \left\|\widehat{R}^k\right\| = \frac{r_1 - r_{q+1}}{1 - r_1} = \mathcal{O}(\Delta t).$$

Recalling that $\widehat{R}_0 = I$, the eigenvalues of $\sum_{k=0}^{q} \widehat{R}_k$ are in the form

$$\lambda_i = 1 + c_i \Delta t,$$

and consequently Q_q is not singular for Δt small enough.

For the second part, proceeding as in [55], let us introduce the matrices $\hat{Q}_q = S^{-1}\left(-\sum_{k=0}^{p} D_k\right)$. Such matrices provide a different approximation of $Q_{\text{ex}} = S^{-1}\Sigma$ directly based on the Neumann expansion of Σ. By recalling that $\|D_k\| = \mathcal{O}(\Delta t^{k+1})$, we have

$$\|S\hat{Q}_q - \Sigma\| = \left\|\sum_{k \geq q+1} D_k\right\| \leq \sum_{k \geq q+1} \|D_k\| = \mathcal{O}(\Delta t^{p+2}). \tag{3.25}$$

It is also possible to prove (see [126]) that

$$\|SQ_q - S\hat{Q}_q\| = \|SQ_q(I - Q_q^{-1}\hat{Q}_q)\| = \mathcal{O}(\Delta t^{p+2}). \tag{3.26}$$

The thesis then follows from (3.25) and (3.26) by triangular inequality.

The consistency analysis is confirmed by the numerical results on a unit square (2D lid-driven cavity) reported in Fig. 3.1 [1].

[1] All the 2-D results presented in this section were obtained by the thesis advisor using the Fortran finite element library LIFE-II and presented for the first time at the 5^{th} European Congress on Computational Methods in Applied Science and Engineering ECCOMAS 2008, Venice, Italy

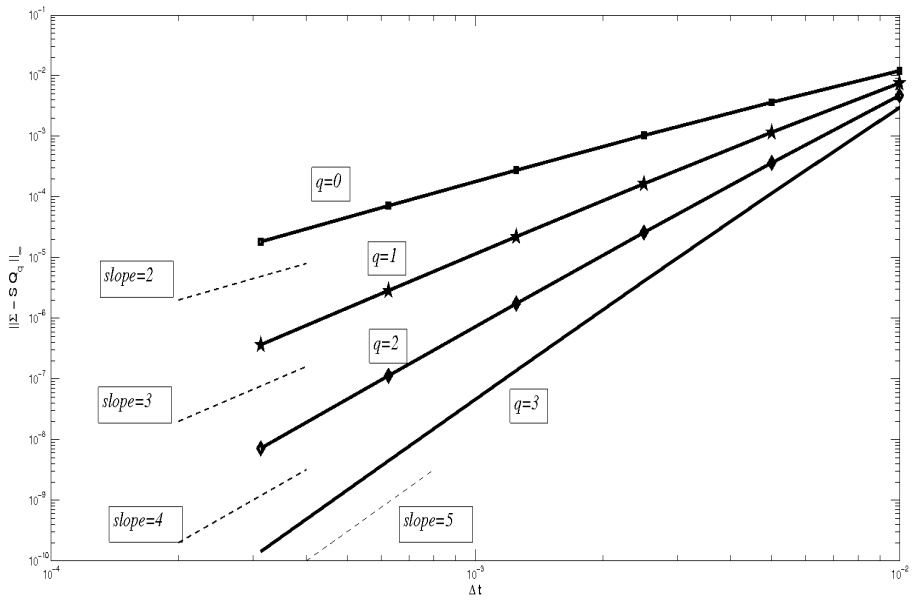


Figure 3.1: Consistency error $\|\Sigma - SQ_q\|_\infty$ induced by the splitting, for different values of q. Matrices corresponds to a $\mathbb{P}^1$-iso-$\mathbb{P}^2$, $\mathbb{P}^1$ finite element discretization on a unit square.

3.4.2 Stability analysis

Before analyzing the stability properties of the HOY schemes, let us introduce some definitions and basic results.

Consider the homogeneous time dependent Stokes problem

$$\begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_{n+1} \\ P_{n+1} \end{bmatrix} = \begin{bmatrix} \frac{M}{\Delta t} \sum_{i=1}^{p} \alpha_i \mathbf{U}_{n+1-i} \\ 0 \end{bmatrix}, \tag{3.27}$$

where the matrix $C = \frac{\alpha_0}{\Delta t} M + K$ is symmetric positive definite.

Then we say that an algebraic splitting method

$$\widehat{\mathcal{A}} \begin{bmatrix} \mathbf{U}_{n+1} \\ P_{n+1} \end{bmatrix} = \begin{bmatrix} \frac{M}{\Delta t} \sum_{i=1}^{p} \alpha_i \mathbf{U}_{n+1-i} \\ 0 \end{bmatrix}$$

for the solution of (3.27) is *conditionally stable* if

$$\exists \Delta t_{\max} \text{ such that } \|\mathbf{U}_{n+1}\|_M \leq \|\mathbf{U}_n\|_M, \ \forall \Delta t < \Delta t_{\max}, \tag{3.28}$$

where $\|\mathbf{U}_*\|_M = \mathbf{U}_*^T M \mathbf{U}_*$ denotes the discrete L^2 norm of the velocity unknown. If condition (3.28) holds for every Δt, then we say that the splitting is *unconditionally stable*.

The following result relates the stability of Yosida-like (i.e. that fulfil exactly the discrete momentum equation) algebraic splitting methods to the violation of the conservation of mass. In particular, if a splitting dissipates mass then it is unconditionally stable, and it is conditionally stable otherwise.

Proposition 3.4.2 Consider a consistent, Yosida-like inexact factorization

$$\widehat{\mathcal{A}} = \begin{bmatrix} C & B^T \\ B & -E(\Delta t) \end{bmatrix}, \lim_{\Delta t \to 0} E(\Delta t) = 0.$$

Then the algebraic splitting induced by $\widehat{\mathcal{A}}$ is

1. unconditionally stable if the error matrix E is symmetric positive semi-definite;

2. conditionally stable if the error matrix E is not symmetric positive semi-definite.

Proof For simplicity, we will consider the case of a BDF-1 time discretization. The extension to higher order BDF formulas is rather technical but standard. Application of the algebraic splitting induced by $\widehat{\mathcal{A}}$ to the time-dependent Stokes problem resorts to solve at each time step the following system:

$$\begin{bmatrix} \frac{1}{\Delta t} M + K & B^T \\ -B & E \end{bmatrix} \begin{bmatrix} \mathbf{U}_{n+1} \\ P_{n+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t} M \mathbf{U}_n \\ \mathbf{0} \end{bmatrix}.$$

Stability is proved by multiplying both sides by $[\mathbf{U}_{n+1}^T, P_{n+1}^T]$ and applying the Young inequality

$$\mathbf{U}_{n+1}^T M \mathbf{U}_n \leq \frac{1}{2}\left(\mathbf{U}_{n+1}^T M \mathbf{U}_{n+1} + \mathbf{U}_n^T M \mathbf{U}_n\right).$$

Indeed, we have

$$\frac{1}{2\Delta t}\mathbf{U}_{n+1}^T M \mathbf{U}_{n+1} + \mathbf{U}_{n+1}^T K \mathbf{U}_{n+1} + P_{n+1}^T E P_{n+1} \leq \frac{1}{2\Delta t}\mathbf{U}_n^T M \mathbf{U}_n. \tag{3.29}$$

If the error matrix E is symmetric positive semidefinite, then $\|\mathbf{U}_{n+1}\|_M^2 \leq \|\mathbf{U}_n\|_M^2$ and unconditional stability is obtained. Otherwise, since E vanishes when Δt tends to zero (consistency), we have $|P_{n+1}^T E P_{n+1}| \leq \mathbf{U}_{n+1}^T K \mathbf{U}_{n+1}$ for Δt small enough, yielding the conditional stability of the scheme. □

The definiteness of the error matrix E for the HOY splitting schemes,

$$E = \Sigma - SQ_q,\ q = 0, 1, 2, \tag{3.30}$$

has been studied in [55, 105, 112, 126], and it is summarized in the lemma below.

Lemma 3.4.3 For the Stokes problem, we have that

1. $\Sigma - SQ_0$ is positive definite;
2. $\Sigma - SQ_1$ is negative semidefinite;
3. $\Sigma - SQ_{2k}$, $\mathbb{N} \ni k \geq 1$ is positive semidefinite for Δt small enough.

$\Sigma - SQ_0 > 0$ has been originally proven in [105] by using the analogy with the Yosida regularization operator. In [55], a simpler proof, based on the symmetric positive definite matrix $H - C^{-1}$, is given. $\Sigma - SQ_1 \leq 0$ was found in [112], while the correct proof of the conditionally semipositiveness of $\Sigma - SQ_q$ for all even $q \geq 2$ was given in [126].

These results suggest that the splitting can actually affect the time stability, apart from the case $q = 0$. It is worth noticing, however, that the stability of BDFp+HOYq schemes depends on the stability of both the the BDF scheme and the algebraic splitting. The following lemma [55, 105, 112] highlights the unconditional stability of the YOS splitting ($q = 0$) when associated with absolutely stable BDF ($p = 1, 2$), and the conditional stability of the HOY splitting ($q = 1, 2$) when coupled with any BDF method.

Lemma 3.4.4 The BDFp+HOYq scheme is unconditionally stable if $p = 1, 2$ and $q = 0$, while it is conditionally stable for $p > 2$ or $q > 0$.

Proof. The lemma is an immediate consequence of the BDF method stability properties, Proposition 3.4.2, and Lemma 3.4.3. In particular, since the YOS splitting ($q = 0$) dissipates mass ($\Sigma - SQ_0 > 0$) it will lead to an unconditionally stable method when coupled with BDF1 or BDF2. The HOY1 scheme is conditionally stable (independently of the order of the BDF formula) since $\Sigma - SQ_1$ is negative definite, while HOY2 is conditionally stable since the semipositiveness of $\Sigma - SQ_2$ can be ensured under suitable restrictions on Δt. □

In Fig. 3.2 we report the error obtained when solving in a 2D rectangular domain the pressure drop problem with a sinusoidal time dependence of the pressure difference between inlet and outlet sections (*Womersley test case*: the 3D analytical solution can be found in [129], the 2D one can be found e.g. in [124]) in the case of $q = 2$. For larger time steps, the solution blows up independently of the BDF scheme adopted, as a consequence of a loss of stability induced by the splitting.

As noted in [112], the region of absolute stability of the time advancing scheme is even more drastically reduced for the HOY1 splitting, due to the fact that $\Sigma - SQ_1$ is negative definite. For this reason, the HOY1 splitting is more often used as a preconditioner (see [54]) then a solver. In Section 3.5 we will provide more details on the

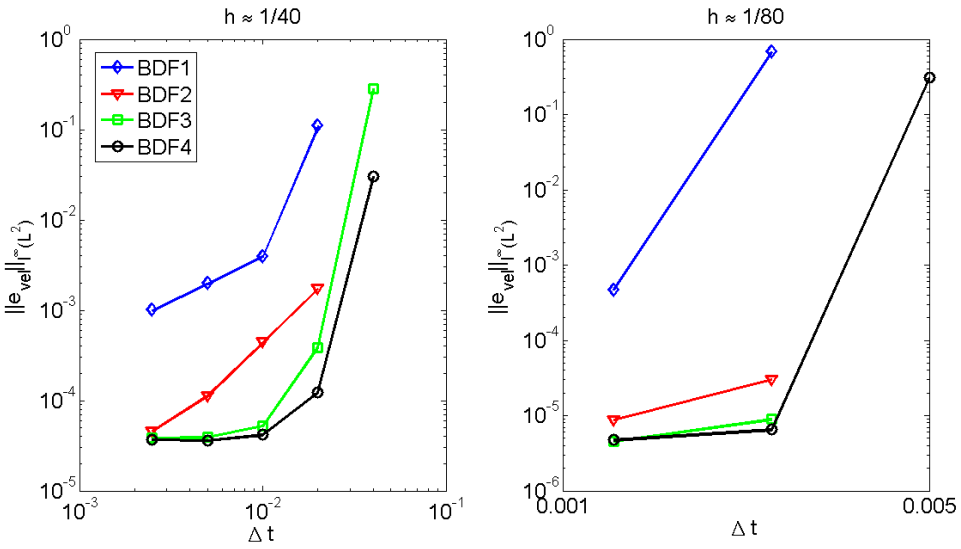


Figure 3.2: Velocity errors for a pressure corrected scheme with $q = 2$. Left: $h = 1/40$, Right: $h = 1/80$.

usage of HOY splittings as preconditioners.

3.4.3 Convergence analysis

We recall here the accuracy result proven in [55] for the time dependent Stokes problem discretized with BDF formulas of order $p = 1, 2$. In Chapter 4 we will provide an analogous result for the incremental formulation of the splitting and BDF formulas of arbitrary order.

To this aim, let us denote with $\mathbf{U}_{n+1}$, P_{n+1} the solution to the fully coupled system

$$\begin{cases} \frac{\alpha_0}{\Delta t} M\mathbf{U}_{n+1} + \nu K\mathbf{U}_{n+1} + B^T P_{n+1} = \mathbf{f}_1 - \frac{M}{\Delta t}\sum_{i=1}^{p} \mathbf{U}_{n+1-i}, \ n = 0, \cdots N_T - 1 \\ B\mathbf{U}_{n+1} = 0, \end{cases} \tag{3.31}$$

and with $\hat{\mathbf{U}}_{n+1}^{(q)}$, $\hat{P}_{n+1}^{(q)}$ the split solution computed according to the HOYq scheme

$$\begin{cases} \frac{\alpha_0}{\Delta t} M\mathbf{U}_{n+1} + \nu K\mathbf{U}_{n+1} + B^T P_{n+1} = \mathbf{f}_1 - \frac{M}{\Delta t}\sum_{i=1}^{p} \mathbf{U}_{n+1-i},\ n = 0, \cdots N_T - 1 \\ B\mathbf{U}_{n+1} - (\Sigma - SQ_q)\hat{P}_{n+1} = 0. \end{cases} \tag{3.32}$$

At each time level t_n we denote with $\mathbf{E}_n^{\mathbf{U},q} = \hat{\mathbf{U}}_n^{(q)} - \mathbf{U}_n$ and $\mathbf{E}_n^{P,q} = \hat{P}_n^{(q)} - P_n$ the vectors of the nodal values of the splitting errors. For a generic time level k, we set $||\mathbf{E}_k^{\mathbf{U},q}||_0^2 \equiv \mathbf{E}_k^{\mathbf{U},q^T} M\mathbf{E}_k^{\mathbf{U},q}$, $||\mathbf{E}_k^{\mathbf{U},q}||_1^2 \equiv \mathbf{E}_k^{\mathbf{U},q^T} K\mathbf{E}_k^{\mathbf{U},q}$ and $||\mathbf{E}_k^{P,q}||_0^2 \equiv \mathbf{E}_k^{P,q^T} M_p\mathbf{E}_k^{P,q}$ where M_p is the mass matrix for the pressure unknown.

The following theorem is proven in [55].

Theorem 3.4.5 If there exists a positive constant c such that $\sum_{n=0}^{N_T-1} \Delta t||p^{n+1}||^2 \leq c$ and Δt is sufficiently small, then there exist two positive constants c_1, c_2 dependent on the space discretization and independent of Δt such that the HOYq schemes for $q = 0, 1, 2$ applied to the Stokes problem satisfy

$$||\mathbf{E}_{N_T}^{\mathbf{U},q}||_0^2 + \nu \sum_{n=n_0}^{N_T-1} \Delta t||\mathbf{E}_n^{\mathbf{U},q}||_1^2 \leq c_1\Delta t^{2q+3}, \qquad \sum_{n=n_0}^{N_T-1} \Delta t||\mathbf{E}_n^{P,q}||^2 \leq c_2\Delta t^{2q+2}. \tag{3.33}$$

We omit the proof of this result, given that, in Chapter 4, the analysis of the local splitting error in Theorem 4.2.1 is demonstrated with similar arguments.

Here we limit to show some numerical results for the Navier-Stokes case.

In Fig. 3.3 we report results on a 2D finite element test case featuring the exact solution $u_x = sin(5\pi t)(y - y^2)$, $u_y = 0$ and $p = 2\nu(x)sin(5\pi t)$. As observed in [55] (with spectral discretization), in some cases numerical results show a better convergence than expected. In this case, velocity splitting errors converge with an order $q + 2$ versus the expected $q + 3/2$.

In Fig. 3.4 we solve the incompressible Navier-Stokes equation in a cylindrical domain with a sinusoidal pressure drop ($\Delta P = A\cos(wt)$) between the inflow and the

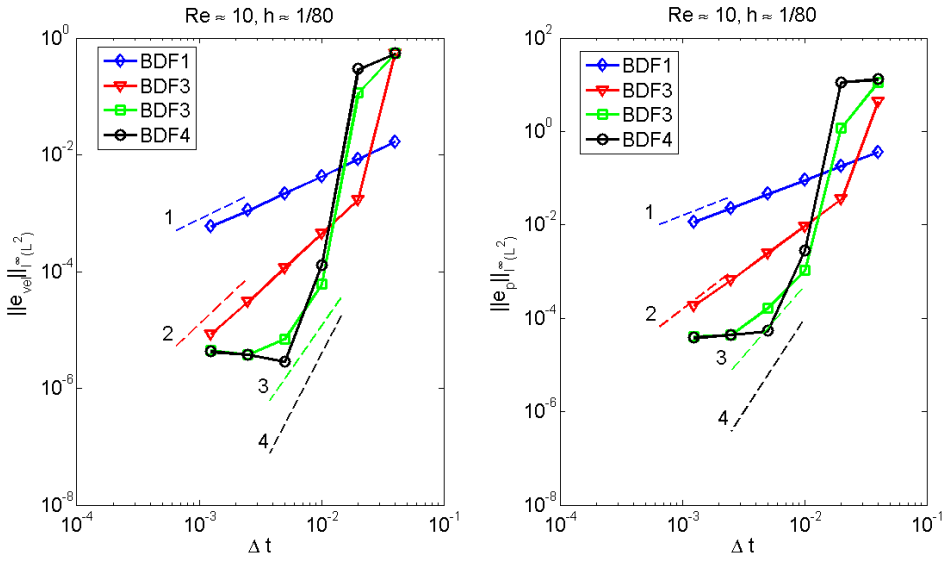


Figure 3.3: Velocity (left) and pressure (right) errors for a pressure corrected scheme with $q = 1$. As expected, for a BDF formula of order 4 the splitting is limiting the velocity accuracy with an order between 3 and 4. Pressure exhibits a slightly higher convergence order than the one predicted by the theory.

outflow and no-slip conditions on the vessel wall (*Womersley test case*). Geometrical and physical parameters (diameter d, length of the cylinder l, viscosity ν, forcing term pulsation ω) yield the Womersley number

$$W = \sqrt{\frac{d^2\omega}{4\nu}} = 15$$

which is within the human physiological range in hemo-dynamics problems.

Space discretization is obtained with Taylor-Hood finite element $Q_2 - Q_1$ for velocity and pressure respectively over a structured stretched hexahedral mesh (1080 elements). Time discretization is performed with BDF of order $q + 1$. The non-linear term has been linearized with a Picard approach, being the extrapolation of the velocity in the convective term of order $q + 1$.

Tab. 3.1 compares the numerical and the expected order of converges estimated for the Stokes problem with different q. For $q = 1, 2$ we obtain the expected order, while for

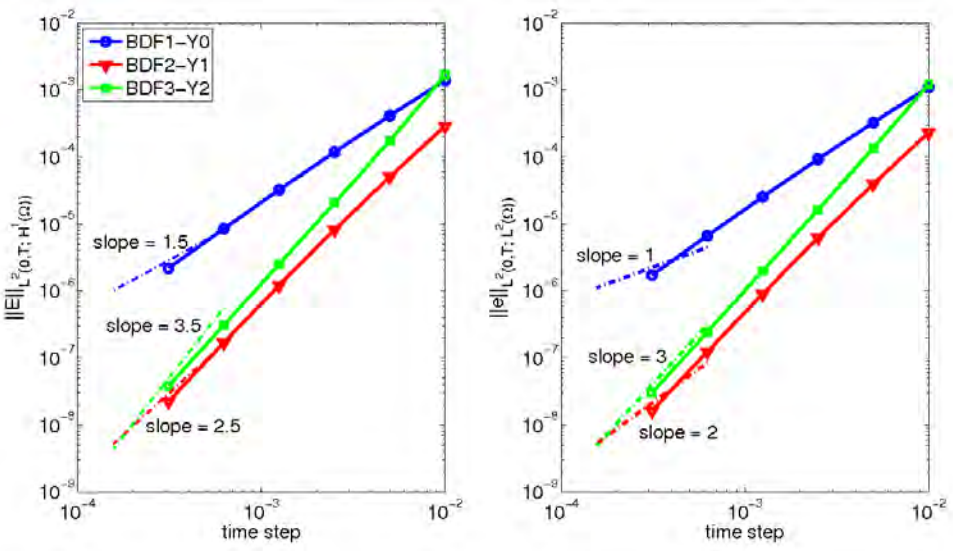


Figure 3.4: Velocity splitting error $\|E\|_{L_2(0,T;H^1(\Omega))}$ (left) and pressure splitting error $\|e\|_{L_2(0,T;L^2(\Omega))}$ (right) for the the solution of the Womersley problem.

	BDF1+HOY0	BDF2+HOY1	BDF3+HOY2
order velocity	1.86 (1.5)	2.73 (2.5)	3.08 (3.5)
order pressure	1.86 (1)	2.76 (2)	3.05 (3)

Table 3.1: Velocity splitting error $\|E\|_{L_2(0,T;H^1(\Omega))}$ and pressure splitting error $\|e\|_{L_2(0,T;L^2(\Omega))}$ for the Womersley test case (theoretical expected order in brackets).

BDF 3 with $q = 2$ we notice a degeneration of 1/2 order in the expected velocity error. This is likely due to the presence of the convective term, that prevents the symmetry of the matrices, which is on the contrary an assumption in the proofs of the convergence results.

3.5 Algebraic splitting as preconditioners

In this section, we investigate the use of block factorizations as preconditioners of system (3.1) and their connection to other block preconditioners available in the literature, namely the Cahouet-Chabard [26] and Least Squares Commutator [44, 45] preconditioners.

Two different, but related, approaches are considered. On the one hand one may consider to directly precondition the fully coupled saddle point system

$$\begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}. \tag{3.34}$$

On the other hand, one may consider the *pressure matrix method*,

$$\begin{cases} C\widehat{\mathbf{U}} = \mathbf{f}_1 \\ \Sigma P = -B\widehat{\mathbf{U}} \\ C\mathbf{U} = \mathbf{f}_1 - B^T P, \end{cases} \tag{3.35}$$

and develop preconditioners for the pressure Schur complement matrix $\Sigma = -BC^{-1}B^T$.

The first approach leads in general to more efficient algorithms (especially if designed for massive parallel computations), while the latter, in which the computations are divided into smaller size sub-problems, is still widely used when memory allocation can be a limiting resource. Indeed, the bottleneck of the pressure matrix method is that each application of Σ to a vector requires to solve a linear system in the momentum matrix C up to a tight tolerance. However, as it will be shown below, the preconditioning techniques for these two approaches are strongly related one to the other.

3.5.1 Block preconditioners and approximated Schur Complement operators

Proceeding as in [45], an initial insight can be obtained by considering the generalized eigenvalue problem

$$\mathcal{A}\begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \lambda \mathcal{P}\begin{bmatrix} \mathbf{U} \\ P \end{bmatrix}, \ \mathcal{A} = \begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix}, \ \mathcal{P} = \begin{bmatrix} C & 0 \\ 0 & -\Sigma \end{bmatrix}. \tag{3.36}$$

It is possible to prove [93] that there are precisely three eigenvalues of (3.36),

$$\lambda_1 = 1,\ \lambda_{2,3} = \frac{1 \pm \sqrt{5}}{2},$$

so that (in exact arithmetic) system (3.34) is solved in three GMRES steps. Indeed, $\mathcal{P}^{-1}\mathcal{A}$ solves $p(\lambda) = (\lambda - 1)(\lambda^2 - \lambda - 1) = 0$. Incidentally we notice that the ratios $-\frac{\lambda_1}{\lambda_3}$ and $\frac{\lambda_2}{\lambda_1}$ are equal to the golden ratio.

Next let us study the spectral properties of the generalized eigenvalue problem

$$\mathcal{A} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \lambda \widehat{\mathcal{P}} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix},\ \widehat{\mathcal{P}} = \begin{bmatrix} C & 0 \\ 0 & -\widehat{\Sigma} \end{bmatrix}, \tag{3.37}$$

where $\widehat{\Sigma}$ is an opportune (cheaper to apply) approximation of Σ. It is possible to prove that the eigenvalue $\lambda = 1$ has both algebraic and geometric multiplicity equal to $N_{\mathbf{U}} - N_P$ (see [45]). Manipulation of the first block of equations leads to

$$\mathbf{U} = \frac{1}{\lambda - 1} C^{-1} B^T \mathbf{P},$$

and then eliminating $\mathbf{U}$ from the second block gives

$$\Sigma P = \mu \widehat{\Sigma} P, \tag{3.38}$$

where $\mu = \lambda(\lambda - 1)$.

Thus, if $\widehat{\Sigma}$ is a good approximation of Σ (i.e. the eigenvalues μ are tightly clustered in a small region near $(1, 0)$ in the complex plane), then eigenvalues of (3.37) will be tightly clustered in two regions, symmetric with respect to imaginary axis, according to the mapping

$$\lambda \to \frac{1 \pm \sqrt{1 + 4\mu}}{2}. \tag{3.39}$$

Better spectral properties are achieved if one considers block lower or upper triangular preconditioners of the form

$$\widehat{\mathcal{P}}_L = \begin{bmatrix} C & 0 \\ B & \widehat{\Sigma} \end{bmatrix}, \text{ or } \widehat{\mathcal{P}}_U = \begin{bmatrix} C & B^T \\ 0 & \widehat{\Sigma} \end{bmatrix}, \tag{3.40}$$

instead of the block diagonal preconditioner in (3.37). Such modification does not entail any significant increase in the cost of applying the preconditioner, but leads to much rapid convergence.

Indeed, by using the same argument used in the block diagonal case, it is possible to prove that the solution of the generalized eigenvalue problem (3.37) with $\widehat{\mathcal{P}} = \widehat{\mathcal{P}}_L$ or $\widehat{\mathcal{P}} = \widehat{\mathcal{P}}_U$ is given by

$$\{\lambda\} = \{1\} \bigcup \{\mu\}, \tag{3.41}$$

where $\{\mu\}$ are the eigenvalues of $\Sigma P = \mu \widehat{\Sigma} P$. All the eigenvalues $\{\lambda\}$ of the preconditioned system now lie on one side of the imaginary axis, significantly reducing the number of GMRES iterations (as many as half iterations in same cases) for the solution of (3.34), see [43].

Relation (3.41) suggests that the behavior of the preconditioners $\widehat{\mathcal{P}}_L$ and $\widehat{\mathcal{P}}_U$ can be inferred by the spectral approximation properties of $\widehat{\Sigma}$, and therefore that efficient preconditioners for the pressure matrix method can be immediately extended to block preconditioner for the saddle point system (3.34).

3.5.2 Spectral properties of algebraic splitting preconditioners

We now consider the adoption of inexact (pressure corrected) LU factorization as *preconditioners* for system (3.34). More specifically, we consider the inexact factorization

matrix

$$\widehat{\mathcal{A}} = \begin{bmatrix} C & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I_{N_{\mathbf{u}}} & U_{1,2} \\ 0 & Q_q \end{bmatrix}, \tag{3.42}$$

where $U_{1,2} = C^{-1}B^T$ for the HOYq scheme, and $U_{1,2} = HB^TQ_q$ for the ACT-PC scheme.

A simple computation shows that the eigenvalues of the preconditioned matrix

$$\widehat{\mathcal{A}}^{-1}\mathcal{A} = \begin{bmatrix} I_{N_{\mathbf{u}}} & C^{-1}B^T - U_{1,2}(SQ_q)^{-1}\Sigma \\ 0 & (SQ_q)^{-1}\Sigma \end{bmatrix}$$

are independent of the choice of the block $U_{1,2}$. Indeed, we have $\lambda = 1$ with (algebraic) multiplicity $N_{\mathbf{u}}$, while the other N_p eigenvalues are those of $(SQ_q)^{-1}\Sigma$. The above observation suggests that to set $U_{1,2} = 0$ in the preconditioner can reduce the computational cost of applying the preconditioner without causing deterioration in the convergence rate of the solution of the preconditioned system [125]. In this case, in fact, we obtain a lower triangular block preconditioner $\widehat{\mathcal{L}}$,

$$\widehat{\mathcal{L}} = \begin{bmatrix} C & 0 \\ B & SQ_q \end{bmatrix}$$

of the form (3.40).

However, as observed in [125], different choices for the matrix $U_{1,2}$ give raise to a different dynamic in the preconditioned residual. In particular, the choice $U_{1,2} = C^{-1}B^T$ (HOYq) yields a null residual for the momentum equation starting from the first iteration, while the choice $U_{1,2} = HB^TQ_1$ (ACT-PC) reduces the residual of the mass conservation equation to machine precision since the first iteration.

Performances of these preconditioners have been investigated in [125] for the case $q = 0$ and in [54] for the case $q = 1$. More in general, Fig. 3.5 pinpoints how the eigenvalues of the preconditioned Schur Complement $(SQ_q)^{-1}\Sigma$ ($q = 0, 1, 2, 3$) of the non symmetric Navier-Stokes problem are clustering around 1 for a lid driven cavity

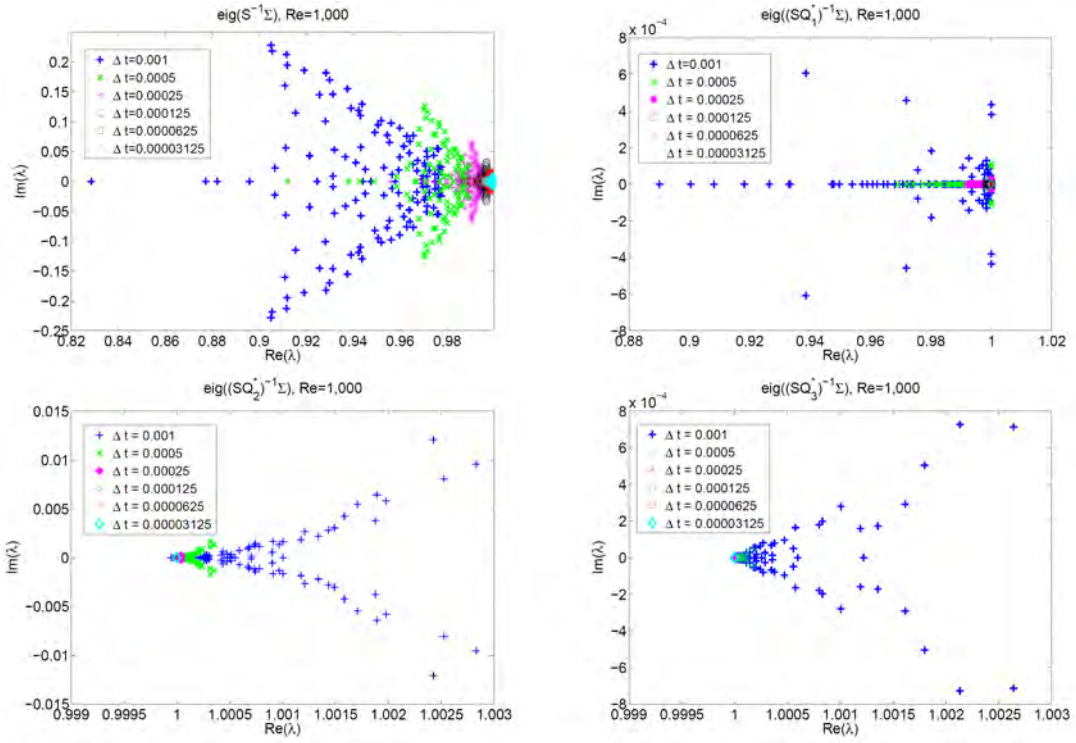


Figure 3.5: Eigenvalues of the preconditioned Schur Complement $(SQ_q)^{-1}\Sigma$ for $q = 0$ (top, left), 1 (top, right), 2 (bottom, left), 3 (bottom, right) and for different time steps. Eigenvalues are computed with Matlab on a coarse mesh (square domain with $h = 1/16$) for the non symmetric Navier-Stokes problem. As expected, when the time step gets smaller, eigenvalues are clustering around 1, the clustering being more evident when q gets larger.

with $Re = 1000$ for and different values of Δt.

In our experience and in agreement with previous studies [54], among all the pressure corrected algebraic splittings, the HOY1 approximation SQ_1 of the Schur Complement Σ is the most efficient for large time steps, providing the best trade-off between computational cost and spectral properties. In order to gain some inside of the spectral properties of SQ_1, let us write its explicit definition and compare its structure with other popular preconditioners, namely the Cahouet-Chabard [26] and the Least Squares Commutator [44] preconditioner. Recalling the hierarchical definition of the

pressure correction matrices Q_q in (3.22), an explicit computation (with $q = 1$) gives

$$SQ_1 = S\left(I + S^{-1}D_1\right)^{-1} = S\left(S - BHAHB^T\right)^{-1} S, \tag{3.43}$$

where $S = -BHB^T$ is the discrete laplacian matrix defined in (3.11). In the next two paragraphs we will use (3.43) to investigate its relationship with the Cahouet-Chabard preconditioner, and its strong similarity with the Least Squares Commutator preconditioner.

3.5.3 Comparison with the Cahouet-Chabard preconditioner

The Cahouet-Chabard (CC in the following) is one of the most popular preconditioner for both the unsteady Stokes and Navier-Stokes problems. It was originally proposed in [26] and it is an optimal preconditioner for the generalized Stokes problem, both with respect to the discretization parameter h, and the PDE coefficients σ, ν.

In particular, if we let $A = \nu K$, the corresponding generalized Stokes matrix and the CC preconditioner read

$$\mathcal{A} = \begin{bmatrix} \sigma M + \nu K & B^T \\ B & 0 \end{bmatrix}, \ \mathcal{P} = \begin{bmatrix} \sigma M + \nu K & 0 \\ 0 & (\sigma^{-1}K_p^{-1} + \nu^{-1}M_p^{-1})^{-1} \end{bmatrix}, \tag{3.44}$$

where M_p, K_p are the finite element matrices representing the discrete inner-products (p_h, q_h) and $(\nabla p_h, \nabla q_h)$, with $p_h,\ q_h \in W_h$.

Here we would like to point out the relationship between the HOY1 preconditioners and the CC one. By letting $A = \nu K$ in (3.43), we have

$$SQ_1 = S(S - \nu BHKHB^T)^{-1}S = \left(S^{-1} - \nu S^{-1}BHKHB^TS^{-1}\right)^{-1}. \tag{3.45}$$

Proceeding as in [54], let now assume that

$$KHB^T = B^T M_p^{-1} BHB^T. \tag{3.46}$$

The condition above is the algebraic counterpart of the differential identity

$$-\Delta\ \nabla = (-\nabla\ \mathrm{div} + \mathrm{curl}\ \mathrm{curl})\ \nabla = -(\nabla\ \mathrm{div})\ \nabla,$$

and has been advocated in [79] as a *compatibility condition* (holding for special set of boundary conditions) in the context of Finite Difference space discretizations. Even if the equality (3.46) does not generally hold for common choices of the finite element spaces $\mathbf{V}_h$ and W_h, we still substitute (3.46) in (3.45) and write

$$SQ_1 = \left(S^{-1} - \nu S^{-1} BHKHB^T S^{-1}\right)^{-1} \approx -\left(S^{-1} + \nu M_p^{-1}\right)^{-1}. \tag{3.47}$$

SQ_1 can be therefore interpreted as the algebraic counterpart of the CC preconditioner, where the discrete laplacian S is used instead of σK_p. If the mass matrix M_p in the CC preconditioner is lumped, then the CC preconditioner is much cheaper than SQ_1, since the latter requires the solution of two linear system in the matrix S instead of one. However, SQ_1 significantly reduces the number of iterations also in the case of a generalized Stokes problem, and it is naturally suited to account for the non-symmetric case, being based on a block factorization of the more general Oseen problem [54].

3.5.4 Comparison with the Least Squares Commutator preconditioner

Let us first provide a brief derivation of the Least Squares Commutator (LSC) preconditioner. The method is based on the notion of approximate algebraic commutator [44]. Consider the advection-diffusion-reaction operator in the momentum equation of the Oseen Problem,

$$\mathcal{L}\mathbf{u} = \sigma\mathbf{u} - \nu\Delta\mathbf{u} + (\mathbf{w}\cdot\nabla)\mathbf{u}, \tag{3.48}$$

where $\mathbf{w}$ is the wind velocity, and $\sigma \geq 0$ is the reaction term. In particular, $\sigma = 0$ for steady problems, and inversely proportional to the time-step for unsteady ones. To derive the LSC preconditioner, one assumes that there is an analogous operator, defined on the pressure space,

$$\mathcal{L}_p p = \sigma p - \nu\Delta p + \mathbf{w}\cdot\nabla p, \tag{3.49}$$

and considers the commutator of the advection-diffusion-reaction operator with the gradient operator,

$$\mathcal{E} = \mathcal{L}\nabla - \nabla\mathcal{L}_p. \tag{3.50}$$

When the wind velocity $\mathbf{w}$ is small or smooth, one expects this commutator to be small in some sense. By using the notation introduced in Section 3.1, a discrete version of the commutator takes the form

$$E = (M^{-1}C)(M^{-1}B^T) - (M^{-1}B^T)(M_p^{-1}C_p),$$

where M_p, C_p represent respectively the mass and advection-diffusion-reaction matrices in the pressure finite element space.

Next, one assumes that also the norm of the algebraic version of the commutator is small ($E \approx 0$), so that the following approximation of the pressure Schur Complement is obtained:

$$\Sigma = -BC^{-1}B^T \approx -BM^{-1}B^T C_p^{-1} M_p. \tag{3.51}$$

To complete the construction of the preconditioner one should opportunely define the matrix C_p. In the LSC preconditioner, the j-th column $[C_p]_j$ of the matrix C_p is

computed by minimizing the following algebraic norm of the commutator

$$\|[M^{-1}CM^{-1}B^T]_j - M^{-1}B^T M_p^{-1}[C_p]_j\|_M^2,\ j = 0, \ldots, N_p - 1.$$

Finally, by substituting in (3.51) the matrix C_p whose columns are the solution of the least squares problem above, one obtains the following approximation of the pressure Schur Complement matrix:

$$\Sigma \approx -\left(BM^{-1}B^T\right)\left(BM^{-1}CM^{-1}B^T\right)^{-1}\left(BM^{-1}B^T\right)^{-1}. \tag{3.52}$$

To reduce the computational cost of the preconditioner, one usually substitute the consistent matrix M with a spectrally equivalent matrix Q that it is simpler to invert, such as a lumped mass matrix or simply $\mathrm{diag}(M)$.

A simple computation show that, for unsteady problems, the LSC preconditioner (3.52) and the pressure corrected preconditioner SQ_1 are equivalent.

In fact, recalling $H = (\sigma M)^{-1}$, $S = -BHB^T$, $C = H^{-1} + A$ we rewrite (3.52) as

$$\Sigma \approx -S\left[BH(H^{-1} + A)HB^T\right]^{-1}S = S\left[S - BHAHB^T\right]^{-1}S = SQ_1, \tag{3.53}$$

where in the last equality we use the definition of SQ_1 in (3.43).

4 Time Adaptivity

In this Chapter, we introduce a time-adaptive method specifically devised for the algebraic splitting schemes discussed in the previous Chapter for the incompressible Navier-Stokes equations. Indeed such pressure corrected schemes feature a hierarchical structure, such that the final pressure is the result of a sequence of low order approximations. For this reason, the comparison between two different estimates is a natural error estimator for the problem, where "natural" means that no additional computation is required.

The time-adaptive solver implemented upon this idea is one of most important results of the present thesis. The presentation of the methods and the numerical results in this chapter closely follows manuscript [128], submitted for publication. After a brief literary review of adaptive methods in computational fluid-dynamics (Section 4.1), we proceed with the local splitting error analysis of the incremental version of the high order Yosida method (Section 4.2). Such analysis is of utmost importance for the derivation of the error estimator for our time-adaptive solver (Sections 4.3 and 4.4). We finally report several numerical results in 2D and 3D, that confirm the reliability and effectiveness of the time adaptive solver (Section 4.5).

4.1 Time adaptivity for computational fluid-dynamics

In some cases, the practical problems at hand involve a sequence of fast and slow transients, only partially (or not at all) predictable *a priori*. This is the case, for instance, of the pressure dynamics in the oil surrounding the piston of a brake (see Fig. 4.1, left),

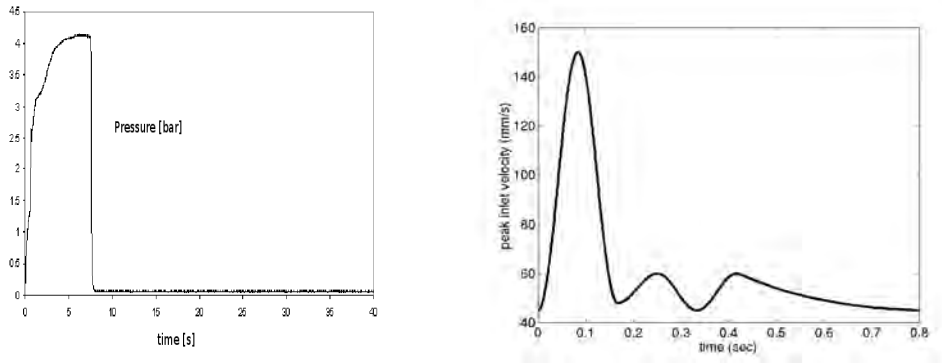


Figure 4.1: Two problems demanding for time adaptivity: pressure dynamics of the oil in a brake (left - courtesy of Brembo, Italy) and flow rate in a carotid artery (right).

that experiences a peak in corresponding of a sudden braking action; or the blood flow in arteries (see Fig. 4.1, right), featuring the sequence of the *systolic* phase, when the aortic valve is open and the velocity is high, and the *diastolic phase* where the valve is closed and the velocity falls. In these cases, a *time adaptive* implementation can reduce the computational effort by reducing the number of time steps required in the intervals of slow transients. Time adaptivity requires basically two ingredients: (a) a reliable and effective *a posteriori* error estimator for testing the adequacy of the current time step; (b) an *adapting rule* for selecting a proper time step on the basis of the estimated error. The latter finds the trade-off between the selection of the largest possible time step and the need of limiting the number of its variations. The former is a critical issue. Residual estimators check quantities that are prevented to vanish - as they would do at the continuous level - by the numerical scheme. More in general, estimators are computed either by combining different schemes with a different order of accuracy, or using the same scheme with two different time steps [104]. Both these strategies can result in an increment of the computational cost, since the numerical solution is computed in two different ways. This has probably limited the extensive adoption of time adaptive schemes for the Incompressible Navier-Stokes (INS) equations, which are intrinsically

fairly expensive at the numerical level. A few recent papers address time adaptivity for this problem. In [49] a stabilized space-time Galerkin method is presented for solving free-surface problems with a time adaptivity based on the residual of the continuity equation. In [60] adaptivity is based on the combination of two-steps linear implicit schemes with different order. In the recent papers [61, 78], a smart combination of the trapezoid rule and the Adams-Bashforth 2 schemes is used for estimating the local truncation error and eventually computing the proper time step. Adaptivity is applied to the coupled momentum-mass system rather than to segregated schemes, since the latter are expected to require smaller time steps.

4.2 Analysis of the incremental formulation of High Order Yosida schemes

We now proceed with the convergence analysis of the incremental q-pressure corrected Yosida scheme (HOYq), one of the most original contribution of this thesis. The method was presented in Chapter 3, however, for convenience, we summarize it in Algorithm 4.1.

Similarly to what already done in [55] (see also Section 3.4) for the analysis of the non-incremental method, we consider the unsteady Stokes problem, where the momentum matrix $C = \frac{\alpha_0}{\Delta t}M + \nu K$ is symmetric positive definite. However, in the incremental case, we should require additional regularity of the analytical solution in order for (3.13) to hold. In fact, while in [55] the author assumes that the pressure belongs to the space $L^\infty(0,T;L^2(\Omega))$ of measurable functions $q : [0,T] \to L^2(\Omega)$, such that $\|q\|_{L^\infty(0,T;L^2(\Omega))} = \operatorname{ess\,sup}_{0\le t\le T} \|q(t)\|_{L^2\Omega}$; here we assume $p \in W^{s,\infty}(0,T;L^2(\Omega))$, that is

$$p \in L^\infty(0,T;L^2(\Omega)), \text{ and } \frac{\partial^i p}{(\partial t)^i} \in L^\infty(0,T;L^2(\Omega)),\ i = 1,\dots,s.$$

Input: $\mathbf{f}_{n+1}$ the discretized forcing term and boundary conditions
Output: $\mathbf{U}_{n+1}$, P_{n+1} the end of step velocity and pressure

// Intermediate velocity step
$C\widehat{\mathbf{U}}_{n+1} = \mathbf{f}_{n+1} - \frac{1}{\Delta t}\sum_{i=0}\alpha_i \mathbf{U}_{n+1-i} - B^T P^*_{n+1}$;
// Intermediate pressure increment step
$S\delta\widehat{P}_{n+1} = B\widehat{\mathbf{U}} - \mathbf{b}_2$;
// Pressure increment correction step $Q_q\delta P_{n+1} = \delta\widehat{P}_{n+1}$, see Algorithm 3.3
for $k = 0, \dots, q$ **do**
  $z_k = \hat{R}_k \delta\widehat{P}_{n+1}$
end
$\delta P_{n+1} = \sum_{k=0}^{q} z_k$;
// End of step pressure
$P_{n+1} = P^*_{n+1} + \delta P_{n+1}$;
// End of step velocity
$C\mathbf{U}_{n+1} = \mathbf{f}_{n+1} - \frac{1}{\Delta t}\sum_{i=0}\alpha_i \mathbf{U}_{n+1-i} - B^T P_{n+1}$;

Algorithm 4.1: Incremental q-pressure corrected Yosida splitting method.

In the following result, we denote by $\mathbf{U}_n^{\text{unsp}}, P_n^{\text{unsp}}$ the solution of (3.14) and by $\mathbf{U}_n^{(q,s)}, P_n^{(q,s)}$ the solution of the q-pressure corrected Yosida scheme, with an increment of order s, i.e.

$$\begin{bmatrix} C & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & C^{-1}B^T \\ 0 & Q_q \end{bmatrix} \begin{bmatrix} \mathbf{U}_n^{(q,s)} \\ \delta P_n^{(q,s)} \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{f}}_n - B^T P_n^* \\ \mathbf{0} \end{bmatrix}.$$

In order to investigate the splitting error introduced at each step, we postulate the *localizing assumption*, i.e. we assume that the solution at the time steps prior the current one corresponds to the unsplit solution, $\mathbf{U}_i^{(q,s)} = \mathbf{U}_i^{\text{unsp}}$ and $P_i^{(q,s)} = P_i^{\text{unsp}}$ for $i = k-1, k-2, \dots k-p$.

We denote the global splitting error at time t_k by $\mathbf{E}_k \equiv \mathbf{U}_k^{\text{unsp}} - \mathbf{U}_k^{(q,s)}$, $\mathrm{e}_k \equiv P_k^{\text{unsp}} - P_k^{(q,s)}$, and the local splitting errors by $\mathbf{E}_{k,*}$ and $\mathrm{e}_{k,*}$. Recalling the discrete norms

$$\|\mathbf{E}_{k,*}\|_0^2 = \mathbf{E}_{k,*}^T M \mathbf{E}_{k,*},\ \|\mathbf{E}_{k,*}\|_1^2 = \mathbf{E}_{k,*}^T K \mathbf{E}_{k,*}, \|\mathrm{e}_{k,*}\|_0^2 = \mathrm{e}_{k,*}^T M_p \mathrm{e}_{k,*},$$

we have the following convergence theorem for the incremental HOYq schemes.

Theorem 4.2.1 If the pressure $p \in W^{s,\infty}(0,T,L^2(\Omega))$ for some $s \geq 0$ and Δt is sufficiently small, then the local splitting error of a q-pressure corrected s-incremental Yosida scheme applied to the generalized Stokes problem satisfies

$$\sup_k \|\mathbf{E}_{k,*}\|_0 \leq C_0 \Delta t^{q+s+2}, \quad \sup_k \|\mathbf{E}_{k,*}\|_1 \leq C_1 \Delta t^{q+s+3/2}, \quad \sup_k \|\mathrm{e}_{k,*}\|_0 \leq c_0 \Delta t^{q+s+1}. \tag{4.1}$$

Proof

A BDF approximation of order p for the Stokes system reads

$$\begin{cases} \dfrac{\alpha_0}{\Delta t} M\mathbf{U}_k^{\mathrm{unsp}} + \nu K \mathbf{U}_k^{\mathrm{unsp}} + B^T P_k^{\mathrm{unsp}} = \mathbf{f}_k + \dfrac{1}{\Delta t} \displaystyle\sum_{i=1}^{p} \alpha_i M \mathbf{U}_{k-i}^{\mathrm{unsp}} \\ B\mathbf{U}_k^{\mathrm{unsp}} = \mathbf{0} \end{cases}, \tag{4.2}$$

while the Yosida s-incremental pressure corrected splitting reads

$$\begin{cases} \dfrac{\alpha_0}{\Delta t} M\mathbf{U}_k^{(q,s)} + \nu K \mathbf{U}_k^{(q,s)} + B^T P_k^{(q,s)} = \mathbf{f}_k + \dfrac{1}{\Delta t} \displaystyle\sum_{i=1}^{p} \alpha_i M \mathbf{U}_{k-i}^{(q,s)} \\ B\mathbf{U}_k^{(q,s)} - (\Sigma - SQ_q)\delta P_k^{(q,s)} = \mathbf{0} \end{cases}. \tag{4.3}$$

Under the *localizing assumption*, we have

$$\delta P_k^{\mathrm{unsp}} = \mathrm{e}_k + \delta P_k^{(q,s)}$$

where we have set $P_k^{\mathrm{unsp}} \equiv \delta P_k^{\mathrm{unsp}} + P_k^{*,\mathrm{unsp}}$. Then, by subtracting (4.3) from (4.2) we get

$$\begin{cases} \dfrac{\alpha_0}{\Delta t} M\mathbf{E}_{k,*} + \nu K \mathbf{E}_{k,*} + B^T \mathrm{e}_{k,*} = 0 \\ B\mathbf{E}_{k,*} - (\Sigma - SQ_q)\mathrm{e}_{k,*} = -(\Sigma - SQ_q)(\delta P_k^{\mathrm{unsp}}) \end{cases}. \tag{4.4}$$

With standard manipulation, we get

$$\begin{aligned}\alpha_0(M\mathbf{E}_{k,*},\mathbf{E}_{k,*})+\nu\Delta t(K\mathbf{E}_{k,*},\mathbf{E}_{k,*})+\Delta t((\Sigma-SQ_q)\mathrm{e}_{k,*},\mathrm{e}_{k,*})\\ =\Delta t((\Sigma-SQ_q)\delta P_k^{\mathrm{unsp}},\mathrm{e}_{k,*}).\end{aligned}\tag{4.5}$$

Thanks to Cauchy-Schwarz and Young inequality, we have

$$\begin{aligned}&\alpha_0\|\mathbf{E}_{k,*}\|_0^2+\nu\Delta t\|\mathbf{E}_{k,*}\|_1^2\\ &\qquad\le\frac{1}{2\varepsilon}\|(\Sigma-SQ_q)\delta P_k^{\mathrm{unsp}}\|^2+\left(2\varepsilon+\gamma\frac{|||\Sigma-SQ_q|||}{\Delta t}\right)\Delta t^2\|\mathrm{e}_{k,*}\|^2,\end{aligned}\tag{4.6}$$

where $\gamma=0$ if $\Sigma-SQ_q$ is semidefinite positive, 1 otherwise. Notice that in [126] it has been proved that $\Sigma-SQ_q$ is semidefinite positive for all even splitting order, provided that Δt is small enough (see also Lemma 3.4.3).

Since $|||\Sigma-SQ_q|||\le c\Delta t^{q+2}$ (recall Proposition 3.4.1), we have

$$\alpha_0\|\mathbf{E}_{k,*}\|_0^2+\nu\Delta t\|\mathbf{E}_{k,*}\|_1^2\le\frac{1}{2\varepsilon}c^2\Delta t^{2q+4}\|\delta P_k^{\mathrm{unsp}}\|^2+\left(2\varepsilon+\gamma c\Delta t^{q+1}\right)\Delta t^2\|\mathrm{e}_{k,*}\|^2.\tag{4.7}$$

Thanks to the *inf-sup* condition (2.12), to $(4.4)_1$, and to the discrete Poincaré inequality $\|\mathbf{V}\|_0\le C_\Omega\|\mathbf{V}\|_1$, we have

$$\begin{aligned}&\|\mathrm{e}_{k,*}\|\le\frac{1}{\beta}\sup_{\mathbf{V}}\frac{|(B^T\mathrm{e}_{k,*},V)|}{\|\mathbf{V}\|_1}=\frac{1}{\beta}\sup_{\mathbf{V}}\frac{|(\frac{\alpha_0}{\Delta t}M\mathbf{E}_{k,*}+\nu K\mathbf{E}_{k,*},\mathbf{V})|}{\|\mathbf{V}\|_1}\le\\ &\le\frac{1}{\beta}\sup_{\mathbf{V}}\frac{\frac{\alpha_0}{\Delta t}\|\mathbf{E}_{k,*}\|_0\|\mathbf{V}\|_0}{\|\mathbf{V}\|_1}+\sup_{\mathbf{V}}\frac{|(\nu K\mathbf{E}_{k,*},\mathbf{V})|}{\|\mathbf{V}\|_1}\le\frac{1}{\beta}\left[C_\Omega\frac{\alpha_0}{\Delta t}\|\mathbf{E}_{k,*}\|_0+\nu\|\mathbf{E}_{k,*}\|_1\right],\end{aligned}\tag{4.8}$$

where $\beta=\beta(\Omega)$ is the *inf-sup* constant independent of the time step. Therefore

$$\Delta t^2\|\mathrm{e}_{k,*}\|^2\le C_*\left[\|\mathbf{E}_{k,*}\|_0^2+\Delta t^2\nu\|\mathbf{E}_{k,*}\|_1^2\right],\tag{4.9}$$

being $C_* = 2\max\left(\frac{C_\Omega^2\alpha_0^2}{\beta^2}, \frac{\nu}{\beta^2}\right)$.

Let $\varepsilon = \frac{1}{4C_*}$ and substitute (4.9) in (4.7)

$$\left(\frac{1}{2} - \gamma c\Delta t^{q+1}\right)\left(\alpha_0\|E_{k,*}\|_0^2 + \nu\Delta t\|E_{k,*}\|_1^2\right) \leq 2C_* c^2\Delta t^{2q+4}\|\delta P_k^{\mathrm{unsp}}\|^2. \tag{4.10}$$

Under the assumption of regularity for the pressure we have $\|\delta P_k^{\mathrm{unsp}}\| = \mathcal{O}(\Delta t^s)$, so that

$$\left(\frac{1}{2} - \gamma c\Delta t^{q+1}\right)\left(\alpha_0\|\mathbf{E}_{k,*}\|_0^2 + \nu\Delta t\|\mathbf{E}_{k,*}\|_1^2\right) \leq K\Delta t^{2q+4+2s}, \tag{4.11}$$

where $k = 4C_* c^2\left(\max_t\left\|\frac{\partial^s p}{\partial t^s}\right\|\right)^2$ is a constant independent of Δt. Should either $\gamma = 0$ or Δt small enough ($\Delta t \leq (4c)^{-1/(q+1)}$), the local splitting error is such that

$$\begin{aligned}
&\|\mathbf{E}_{k,*}\|_0 \leq \tfrac{(2+2\gamma)k}{\alpha_0}\Delta t^{q+s+2},\\
&\|\mathbf{E}_{k,*}\|_1 \leq \tfrac{(2+2\gamma)k}{\nu}\Delta t^{q+s+3/2},\\
&\|\mathrm{e}_{k,*}\| \leq \tfrac{(2+2\gamma)k}{\beta}(1+C_\Omega)\Delta t^{q+s+1}.
\end{aligned} \tag{4.12}$$

◇

Remark - When we remove the localizing assumption and introduce the error propagation contribution, it is possible to prove that

$$\|\mathbf{E}_k\|_0 \leq \mathcal{C}\Delta t^{q+s+1},$$

as a consequence of the previous Theorem and of the classical propagation error analysis of linear multistep method (see for example [67] pages 235-249). More complicated is the estimation of the pressure global splitting error e_k. We conjecture that also when we remove the localizing assumption the pressure splitting error still features an order $\|\mathrm{e}_k\| \leq \widetilde{\mathcal{C}}\Delta t^{q+s+1}$. A rigorous proof is still missing.

4.3 Adaptation rule

In this section we briefly describe a general time step adaptation rule based on *a posteriori estimators* for the local error, postponing to next section the specialization to the incompressible Navier-Stokes equations. Consider the abstract initial value problem

$$\begin{cases} \dfrac{d\mathbf{y}}{dt} = f(\mathbf{y}),\ t \in (0,T] \\ \mathbf{y}(0) = \mathbf{g}, \end{cases}$$

and assume two different approximations $\mathbf{y}_n$, $\hat{\mathbf{y}}_n$ of the analytical solution $\mathbf{y}(t_n)$ are computed, the first having a local error τ_i of order $\mathcal{O}(\Delta t^r)$, the second having a local error $\hat{\tau}_i$ of order $\mathcal{O}(\Delta t^{r+1})$.

Recalling the definition of local error,

$$\mathbf{y}(t_i) = \mathbf{y}_i,\ \forall i \leq n \longrightarrow \mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1} = \Delta t\ \tau_n,$$

we have

$$\begin{aligned} \tau_n = \frac{1}{\Delta t}\left[\mathbf{y}(t_{n+1}) - \mathbf{y}_{n+1}\right] = \frac{1}{\Delta t}\left[\mathbf{y}(t_{n+1}) - \hat{\mathbf{y}}_{n+1}\right] + \frac{1}{\Delta t}\left[\hat{\mathbf{y}}_{n+1} - \mathbf{y}_{i+1}\right] \\ = \hat{\tau}_n + \frac{1}{\Delta t}\left[\hat{\mathbf{y}}_{n+1} - \mathbf{y}_{i+1}\right]. \end{aligned} \quad (4.13)$$

Observe that, since $\tau_n = \mathcal{O}(\Delta t^r)$ and $\hat{\tau}_n = \mathcal{O}(\Delta t^{r+1})$, the quantity $[\hat{\mathbf{y}}_{n+1} - \mathbf{y}_{i+1}]$ represents the principal contribution to the local error τ_n. Then an *a posteriori* error estimator η is given by the difference of the solutions $\hat{\mathbf{y}}_{n+1}$ and $\mathbf{y}_{n+1}$. In particular, by letting

$$\eta = \|\hat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1}\|,$$

the local error τ_n is such that

$$\|\tau_n\| \approx \frac{\eta}{\Delta t}. \quad (4.14)$$

The *a posteriori estimator* η so designed can then be used in the adaptation rule as follows.

Since $\tau_n = \mathcal{O}(\Delta t^r)$, there exists a constant k such that $\|\tau_n\| = k\Delta t^r$. From (4.14), we then have

$$k\Delta t^r \approx \frac{\eta}{\Delta t}.$$

Now, let χ be a multiplicative factor of the current time step, so that $\Delta t_{\text{new}} = \chi \Delta t_{\text{old}}$. We look for χ such that for a prescribed tolerance ε we have

$$\|\tau_n(\chi \Delta t_{\text{old}})\| \leq \varepsilon.$$

Here ε represents the maximum allowed error per unit of time, so that the above condition guarantees that the global error at time $T = t_{n_T}$ is bounded by

$$\|\mathbf{y}(T) - \mathbf{y}_{n_T}\| \leq \varepsilon T.$$

Since

$$\|\tau_n(\chi\ \Delta t_{\text{old}})\| = k(\chi\ \Delta t_{\text{old}})^r = \chi^r (k\Delta t_{\text{old}}^r) \approx \frac{\chi^r}{\Delta t}\eta,$$

we have

$$\chi \leq \left(\frac{\varepsilon\ \Delta t_{\text{old}}}{\eta}\right)^r. \tag{4.15}$$

The value of χ is then used, according to the adaptation rule,

1. to reject, if necessary, the initial choice Δt_{old} at the time level n, and to repeat the computation using $\Delta t_{\text{new}} = \chi\ \Delta t_{\text{old}}$;

2. to accept the solution computed at the time level n, and to predict the time step $\Delta t_{\text{new}} = \chi\ \Delta t_{\text{old}}$ for the next time step.

It is worth to notice that, for multistep time advancing methods, stability constraints may restrict the selection of χ to a range of admissible values $(\chi_{\min}, \chi_{\max})$. For exam-

ples, the stability of the variable step BDF formulas is guaranteed if $\Delta t_{\text{new}} \leq \delta \Delta t_{\text{old}}$, where $1 \leq \delta \leq 1.5$ is a constant depending of the number of steps [28, 29].

In practice, more elaborated adaptation rules should be introduced in order to avoid excessive oscillations in the selection of the time step Δt and to guarantee a high quality implementation.

Below we list some technical details to make the adaptation rule more robust and efficient (see Algorithm 4.2).

1. We multiply χ by a safety coefficient $s \leq 1$ (usually between 0.8 and 0.9, cfr. [35]) in order to relax the *accept/reject* criterion.

2. We introduce three different reference values χ_0, χ_1, χ_2 ($\chi_{\min} \leq \chi_0 \leq s \leq \chi_1 \leq 1 \leq \chi_2 \leq \chi_{\max}$) in order guarantee smooth variations in the time step and to reduce the number of time-step rejected. The relaxed adaptation rule then reads

   (a) if $s\chi \leq \chi_0$, reduce the time-step and reject the solution;

   (b) if $s\chi \in [\chi_0, \chi_1]$, reduce the time-step and move to the next time step;

   (c) if $s\chi \in [\chi_1, \chi_2]$, keep current time-step and move to the next time frame

   (d) if $s\chi \geq \chi_2$, increase the time-step and move to the next time frame.

3. We restrict the time step to belong to some interval $(\Delta t_{\min}, \Delta t_{\max})$ which is typical of the application.

4.4 A posteriori error estimators for the Navier-Stokes problem

In this section, we discuss error estimators η based on the incremental q-pressure corrected schemes in Algorithm 4.1, which can be used for the automatic selection of the time-step according the the adaptation rule in Section 4.3.

Input: η a posteriori error estimator of order r.
Input: Δt the current time step used to compute the estimator

$\chi = \left(\frac{\varepsilon\,\Delta t}{\eta}\right)^r$;
$\hat{\chi} = \max\left(\min(s\chi, \chi_{\max}), \chi_{\min}\right)$;

if $\hat{\chi} \leq \chi_0$ **then**
| reject the time step; set $\Delta t \leftarrow \hat{\chi}\Delta t$, and recompute the solution;
end
else if $\hat{\chi} \in (\chi_0, \chi_1)$ **then**
| accept the time step; reduce $\Delta t \leftarrow \hat{\chi}\Delta t$; move to the next time level;
end
else if $\hat{\chi} \in (\chi_1, \chi_2)$ **then**
| accept the time step; keep the same Δt; move to the next time level;
end
else if $\hat{\chi} > \chi_2$ **then**
| accept the time step, increase $\Delta t \leftarrow \hat{\chi}\Delta t$; move to the next time level
end

if $\Delta t < \Delta t_{\min}$ **then**
| $\Delta t \leftarrow \Delta t_{\min}$;
end
else if $\Delta t > \Delta t_{\max}$ **then**
| $\Delta t \leftarrow \Delta t_{\max}$;
end

Algorithm 4.2: Adaptation rule algorithm

It is worth to notice that the error between the exact and the numerical solution consists of three different contributions: the space discretization error related to the Finite Element Method, the time discretization error associated to the time advancing technique (BDF and linearization/extrapolation of the convective term), and the splitting error induced by the pressure corrected schemes.

To be more specific, let us denote with P_n^{ex} the analytical pressure at time t_n, P_n^{semi} the solution of the semi-discrete in space Problem (2.13), and make the localizing assumption $P_k^{\text{semi}} = P_k^{\text{unspl}} = P_k^{(q,s)}$ for all time level k previous to n.

Then the total error is such that

$$\begin{aligned} \|P_n^{\text{ex}} - P_n^{(q,s)}\| \quad &\leq \|P_n^{\text{ex}} - P_n^{\text{semi}}\| \quad + \|P_n^{\text{semi}} - P_n^{\text{unspl}}\| \quad + \|P_n^{\text{unspl}} - P_n^{(q,s)}\| \\ &= \mathcal{O}(h^{\alpha}) \qquad\qquad + \mathcal{O}(\Delta t^{p+1}) \qquad\qquad + \mathcal{O}(\Delta t^{q+s+1}), \end{aligned}$$

where $\alpha \geq 1$ is an integer number depending on the choice of the finite element spaces, and p is the order of the time advancing method.

In the following, assuming that mesh size h is small enough, we neglect the spatial contribution to the error since it is independent of the time step. Thanks to the local splitting error analysis in Theorem 4.2.1, we devise error estimators to explicitly address the third contribution to the total error. If $p > q + s$, we can also neglect the time discretization error since it represents an higher order term compared to the local splitting error. On the contrary, for $p = q+s$, we heuristically assume that the two error contributions have a similar behavior, i.e. $\|P_n^{\text{semi}} - P_n^{\text{unspl}}\| \approx \|P_n^{\text{unspl}} - P_n^{(q,s)}\|$.

In the next paragraph, we will present the error estimator specifically devised for the algebraic splitting schemes discussed in the previous Chapter. Indeed such pressure corrected schemes feature a hierarchical structure, such that the final pressure is the result of a sequence of low order approximations. For this reason, the comparison between two different estimates is a natural error estimator for the problem, where "natural" means that no additional computation is required. Even though the error

estimator is reliable, the pressure corrected schemes may suffer from some stability constraints induced by the splitting that reflect in the automatic selection of the time step. For this reason, a variant of the original idea, based on using the pressure corrected scheme as preconditioners and the comparison with the unsplit solution for the error estimation, is presented in Paragraph 4.4.2. This second approach is in general computationally more expensive, but it does not suffer from the stability drawback induced by the splitting.

4.4.1 Algebraic splitting based estimators

For the HOY-q incremental schemes, the correction step $Q_q \delta P = \delta \widetilde{P}$ takes the form

$$\delta P = \sum_{k=0}^{q} z_k, \; z_k = \hat{R}_k \delta \widetilde{P}.$$

From the convergence Theorem 4.2.1 we infer that the addition of each term z_k increases the accuracy of the pressure. As a matter of fact, we deduce from $(4.1)_3$ that

$$\mathcal{O}(\Delta t^{q+s+1}) = \|P^{\mathrm{ex}} - P^{(q,s)}\| \leq \|P^{(q+1,s)} - P^{(q,s)}\| + \|P^{\mathrm{ex}} - P^{(q+1,s)}\| = \|z_{q+1}\| + \mathrm{h.o.t.},$$

where vector z_{q+1} is a by-product of the hierarchical structure of the scheme, so it does not need any additional computation.

The above inequality highlights the role of the quantity $\eta = \|z_{q+1}\|_0$ as possible error estimator (of order $r = q + s$) to be used in the adaptation rule (see Algorithm 4.2).

In Fig. 4.2 we report the quantification of the norm L^2 and L^∞ of vectors z_k for $k = 1, 2, 3$ in a 2D problem on a unit square. In Fig. 4.3 we analyze the norm L^2 of the z_k for different orders s of pressure extrapolation in a 3D lid driven cavity problem. The expected behavior is confirmed by the numerical evidence.

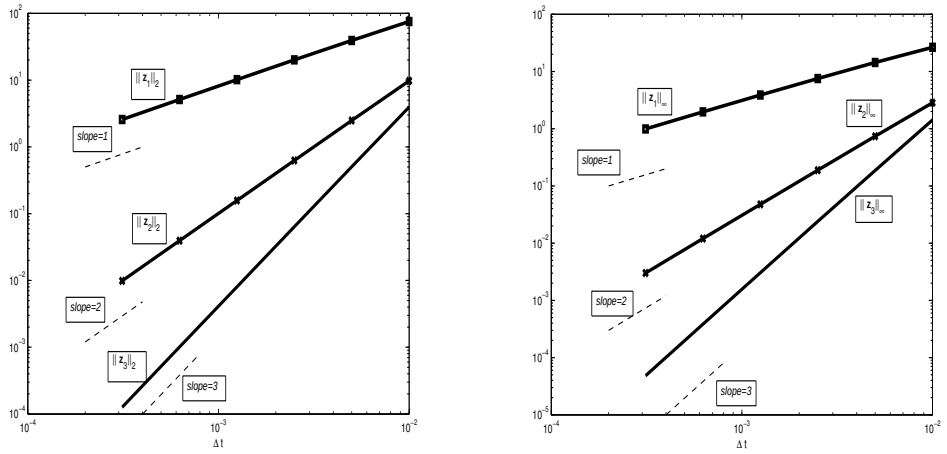
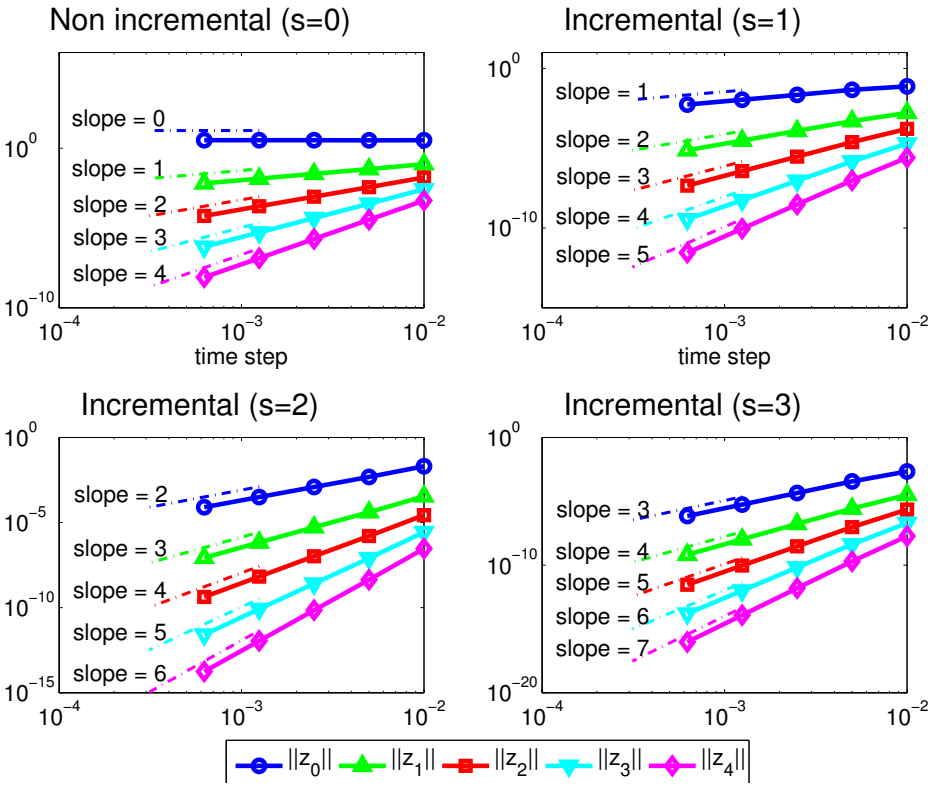


Figure 4.2: Norm of the error estimator as a function of the time step. L^2 norm $\|z\|_{L^2}$ on the left, infinity norm $\|z\|_{L^\infty}$ on the right (in 2D lid driven cavity problem).

4.4.2 Preconditioned unsplit solvers estimators

In real applications reduction of stability induced by the combination of high order schemes with the splitting can be a major issue, limiting the effectiveness of the adaptive procedure. This can prevent the automatic selection of large time steps, depending on the size of the space mesh. As a matter of fact, a few highly stretched elements or mesh singularities may drastically increase the spectral radius of the Neumann expansion matrix HA, leading to severe constraints on the choice of the time step. A possible workaround is the adoption of pressure corrected factorizations as *preconditioners* for system (3.14). Performances of Yosida-like preconditioners have been investigated in Section 3.5.1 (see also [125] for the case $q = 0$, and [54] for $q = 1$). In particular, it has been proved in [125] that each preconditioned iterations provide a progressively more accurate estimate for the pressure. A possible error estimator can be therefore obtained by comparing the solution obtained by the pressure corrected scheme (corresponding to one preconditioned iteration) $P^{(q,s)}$ (typically with $q = 0, 1$) and the converged (unsplit) solution P^{unsp}. More specifically, assuming that the time discretization for the



	$s=0$	$s=1$	$s=2$	$s=3$
$k=0$	-0.0012 (0)	0.9469 (1)	1.9980 (2)	2.9005 (3)
$k=1$	0.9987 (1)	1.9488 (2)	2.9999 (3)	3.8903 (4)
$k=2$	1.9987 (2)	2.9484 (3)	3.9972 (4)	4.8908 (5)
$k=3$	2.9987 (3)	3.9478 (4)	4.9970 (5)	5.8910 (6)
$k=4$	3.9987 (4)	4.9470(5)	5.9967 (6)	6.8906 (7)

Figure 4.3: Order of the estimator z_k for the non-incremental method and the incremental methods of order s. These results are obtained by solving a lid cavity problem driven by a $C^\infty(0,T)$ forcing term ($u_{top} = e^{\sin(2.0\pi t)} - 1.0$). Space discretization is Taylor Hood finite elements $Q_2 - Q_1$ on a 8 by 8 by 8 hexahedral mesh. Time discretization is a BDF-(s+1).

unsplit scheme is accurate enough ($p > q + s$), we have

$$\|P^{\mathrm{ex}} - P^{(q,s)}\| \leq \|P^{\mathrm{ex}} - P^{\mathrm{unsp}}\| + \|P^{\mathrm{unsp}} - P^{(q,s)}\| = \|P^{\mathrm{unsp}} - P^{(q,s)}\| + \mathrm{h.o.t.}$$

By letting $\eta = \|P^{\mathrm{unsp}} - P^{(q,s)}\|_0$, we therefore obtain, thanks to $(4.1)_3$, an a posteriori error estimator of order $r = q + s$.

Since at the end of the time step, the solution P_{unsp} is retained, the overall stability of the method is driven by the stability of the BDF scheme.

4.5 Numerical results

In this section, we present 2D and 3D numerical results to demonstrate the validity of our time adaptive procedures. 2D results have been obtained with the Fortran finite element code `LifeII`, while 3D have been obtained with the 3D library `LifeV` (www.lifev.org). We first present some preliminary 2D results in which we use the *algebraic splitting based error estimator*, and then we present 3D results in which we use the *preconditioned unsplit solvers estimators*. Results are mainly related to cardiovascular problems that have inspired the present thesis.

4.5.1 Preliminary 2D results

In Fig. 4.4 we present a 2D test case, where a forcing pressure drop between inlet and outlet boundaries in a channel is applied with a sinusoidal waveform in the first part of the time interval. Then, the pressure drop is turned off. In the first quarter of the time interval adaptivity is not activated and the time step is fine enough for the dynamics forced by the pressure drop. Successively the time step is selected according to the adaptive strategy. The results are obtained by comparing the solutions with q and $q+1$ for $q = 1$ and a tolerance $\varepsilon = 0.01$. When the time step is accepted, even though the

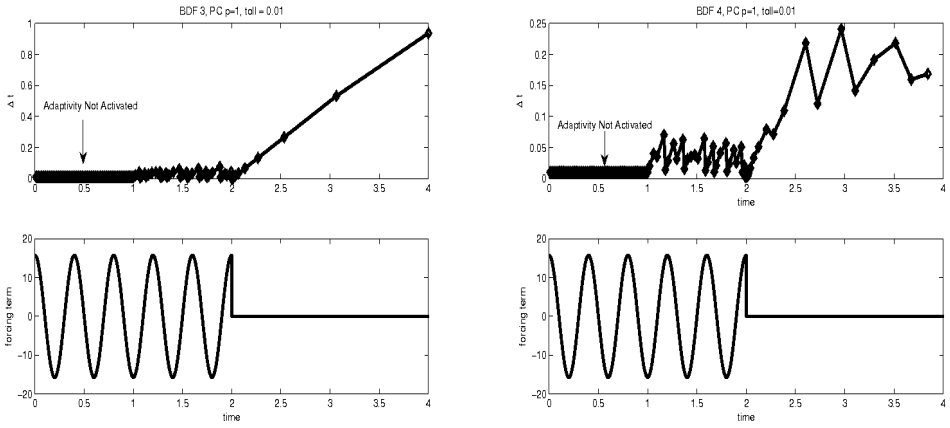


Figure 4.4: Time step automatically selected by using z_2 as error estimator for a fluid in a 2D channel with a periodic forcing term (reported in the bottom panel). For the first time unit adaptivity is off. Then it is turned on. After two time units, the forcing term is set to 0, so the adaptive scheme is supposed to select a large time step. This happens with the BDF3 (left). With the BDF4, reduced stability of the split scheme prevents the selection of large time steps (right).

estimate refers to the error for $q = 1$, the "more accurate" solution is available and it is retained.

When we adopt a BDF of order 3, we find the expected behaviour on the selection of time step, and the final Δt is selected to be large when the forcing pressure drop is off (Fig. 4.4, left). When we use the BDF of order 4, the reduced stability induced by the combination of the time advancing BDF4 and the splitting prevents the selection of large time steps (Fig. 4.4, right).

A less trivial example of adaptivity is presented in Fig. 4.5, where a 2D domain resembling a vascular bifurcation has been simulated with a physiological-like waveform in the flow rate. The waveform reported in the picture refers to three heart beats and has been used for modulating the inflow velocity profile. Neumann conditions have been prescribed at the outlets. Adaptivity has been activated after the first heart beat. The initial time step has been tuned for a correct representation of the first part of each heart beat (systole), that features a fast transient. The adaptive scheme (BDF3 with

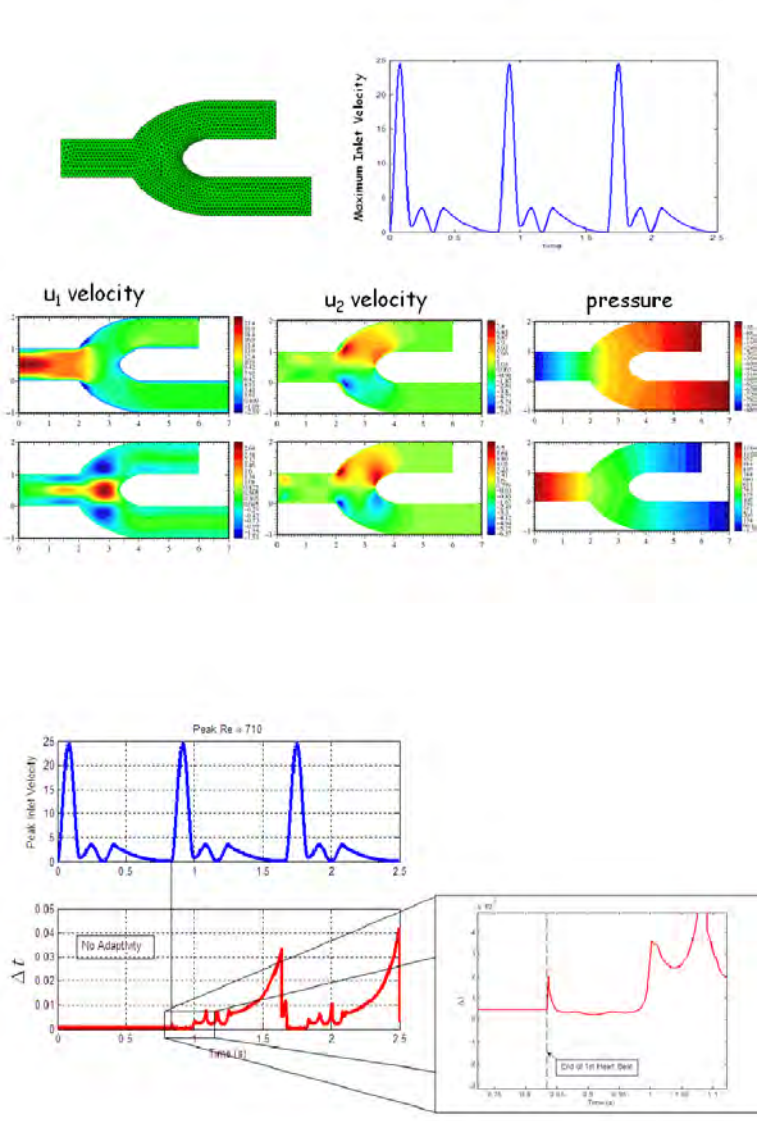


Figure 4.5: Time adaptivity in a 2D bifurcation with an input time-dependent velocity modulated by a physiological waveform. Three heart beats with a physiological peak Reynolds of about 700. In the first beat, adaptivity is off and time step is selected for capturing the fast transients of the first part of the heart beat (systole). In the second and third heart beat, the adaptivity maintains the same time step during systole (see the zoomed box below) and selects larger steps during the subsequent phase (diastole). In this way, one third of time steps used by the non-adaptive computation are required.

$q = 1$) works in the expected way. At the beginning (during the second systole) the time step is not changed significantly, then during the diastolic phase a larger Δt is automatically selected. The overall number of time steps is reduced from 3334 (with no adaptivity) to 1127 per heart beat.

4.5.2 3D Womersley test case

We solve the incompressible Navier-Stokes equations in a cylindrical domain with a sinusoidal pressure drop ($\Delta P = A\cos(w\ t)$) between the inflow and the outflow and no-slip conditions on the vessel wall (*Womersley test case*).

The maximum pressure drop is the same for each simulation ($A = 5$), while pulsation frequency w is varied so to yield to different Womersley numbers (1, 2, 4, 8, 16).

We use in particular a time adaptive method with a BDF of order 3, and a selection of the time step based on the combination of the Yosida-1 solution and the unsplit solution (see Section 4.4.2). In Fig. 4.6 we illustrate the pressure drop prescribed with the flow rate computed (top) and the time step selected (bottom) for different Womersley numbers. As expected from the analytical solution of the Womersley test case, higher is the Womersley number and larger is the phase lag between pressure and flow rate. The time step selected is more refined in correspondence of the fast transients, when the pressure gradient is steep.

In Fig. 4.7 we illustrate the period of oscillation versus the automatically selected time step. The slope of the graph (in a log-log plot) suggests a non polynomial dependence.

4.5.3 Sensitivity with respect to the mesh size

We consider a modified 3D lid driven cavity problem with the kinematic viscosity of the fluid equal to 0.01. The computational domain is the unitary cube ($[-\frac{1}{2}, \frac{1}{2}]^3$). The

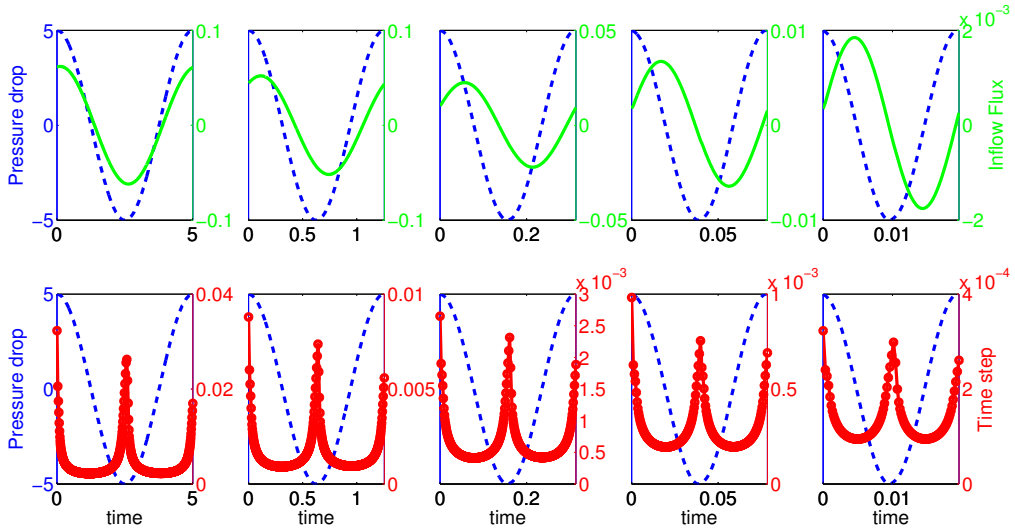


Figure 4.6: Simulation for increasing Womersley numbers: 1, 2, 4, 8, 16 (from the left to the right). The imposed pressure drop and the computed flux at the inflow section are showed on the top (in dashed blue line and solid green line respectively). The adaptive time step and the pressure are showed on the bottom (in solid red line and dashed blue line respectively). Increasing the Womersley number the phase lag between pressure and flux increases, and the amplitude of flux oscillation decreases. Note how the behaviour of the adaptive time step is the same for each simulation.

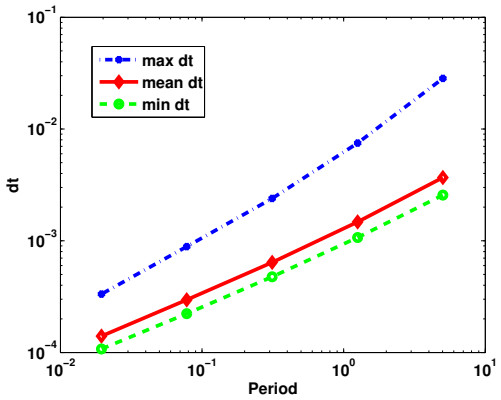


Figure 4.7: Average, minimum, maximum time step selected by the adaptive algorithm Schur complement-Yosida1 incremental approach as a function of the period of oscillation in the simulation with different Womersley numbers.

horizontal velocity prescribed on the top face of the cube ($z = \frac{1}{2}$) is equal to

$$v_x = \frac{e^{\sin(2\pi t)} - 1.0}{e} \left(1 - (2x)^4\right) \left(1 - (2y)^4\right),$$

leading to a maximum Reynolds number of 100.

Space discretization is $Q_2 - Q_1$ Taylor-Hood finite elements in a structured hexahedral mesh. Three mesh diameters are considered: $h = 0.25$, $h = 0.125$, $h = 0.0625$. Time discretization is BDF of order 3. We use the third order error estimator in which we compare the Schur complement pressure with the incremental Yosida 1 approximation ($p = 1$, $s = 1$).

In order to avoid too small time step, we set a target tolerance for the adaptive selection of the time step which varies with the mesh diameters. In particular we use a tolerance $\varepsilon = 2.25 \cdot 10^{-5}$ on the coarse mesh, $\varepsilon = 10^{-4}$ on the intermediate mesh, $\varepsilon = 4 \cdot 10^{-4}$ on the fine mesh. The ratio between the previous time step and the next is bounded from above by $\chi_{\max} = 1.05$, while we reject a time step for $\chi \geq 0.9$.

Fig. 4.8 suggests that the error estimator is robust with respect to the mesh size. The adaptively selected time step shows in fact the same dynamics for all three the mesh diameter: the time step is refined in correspondence of the accelerations of the lid, which generates fast changes in the pressure field, while it is coarsened when the lid moves slower. The quantitative differences in the selection of the time step, may be justified by the fact that a finer mesh is able to capture smaller and less stable features of the flow, which require a smaller time step to be properly resolved.

4.5.4 An adaptive 3D blood flow simulation

We finally present a real 3D case of the simulation of blood flow in a domain represented by a human aorta reconstructed from images (see Fig. 4.10) with the free software `VMTK` [8].

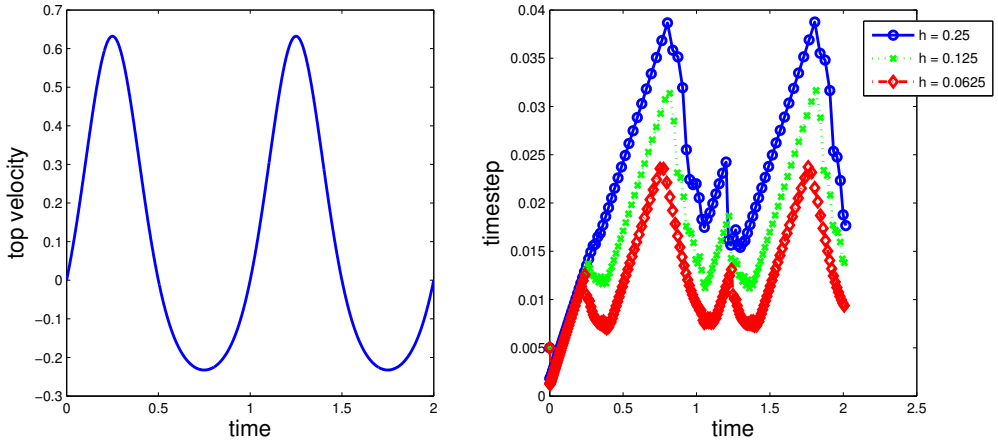


Figure 4.8: On the left the dynamic in time of the velocity of the lid. On the right the time step chosen by the time adaptive procedure. Qualitatively the adaptive choice of the time step shows the same behaviour for all the three different mesh diameters. The time step is refined when the accelerations of the lid cause fast changes in the pressure, and it is coarsened when the lid velocity varies slowly and the pressure gradients are less steep.

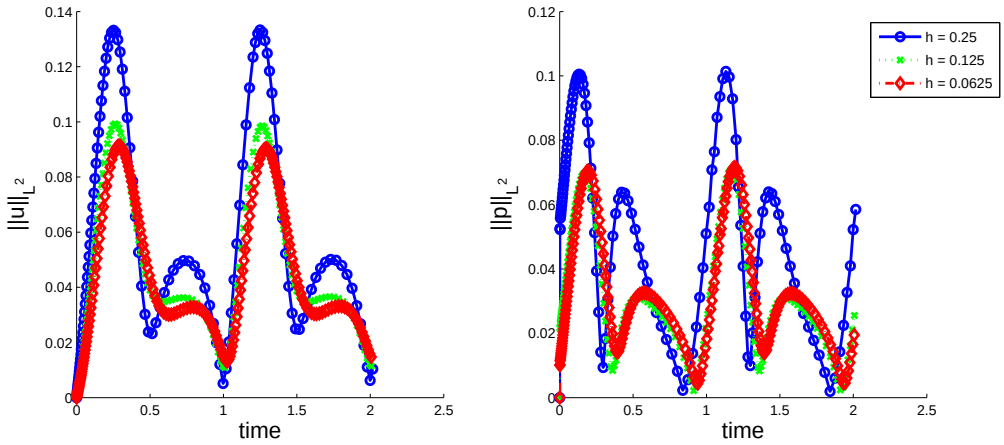


Figure 4.9: The $L^2(\Omega)$ norm of the velocity (on the left), and the $L^2(\Omega)$ norm of the pressure (on the right) for the lid driven cavity problem. The different behaviour of the velocity and pressure norms on the coarse grid suggests that the fluid-dynamics is under-resolved on that mesh.

Figure 4.10: Geometry of an aortic arch reconstructed from a CT with the code Vascular Modeling Toolkit (www.vmtk.org) and used for our time adaptive simulations.

The dataset of medical images has been acquired at *Ospedale Maggiore in Milano, Italy*, using a Siemens SOMATOM Definition Flash Dual-Source CT Scanner. From the original scanned volume, the region including the aortic arch and the thoracic aorta has been selected. Segmentation has been performed using the level set method implemented in the free software `VMTK` [8]. All the side branches have been excluded for simplicity.

We simulate three heart beat, the first one with a constant time step, the others with an adaptive time step. We prescribe an inlet parabolic velocity profile based on a realistic waveform of the flow rate. Stress-free conditions are assumed at the outflow section, and no-slip boundary condition on the vessel walls. Fig. 4.11 (on the left) shows the prescribed flux and the computed pressure on the inflow section as function of time. To discretize the geometry we use an unstructured tetrahedral mesh with 10,000 nodes. We adopt inf-sup compatible P_1^b-P_1 finite element for velocity and pressure respectively and a BDF of order 3 for the time advancing.

Adaptivity is carried out by coupling the Yosida 1 scheme with the unsplit solution. We use an incremental scheme with $s = 1$. In particular, we select a tolerance $\varepsilon = 0.1P_{peak}$, where P_{peak} is the pressure required for standing the peak flow rate in a

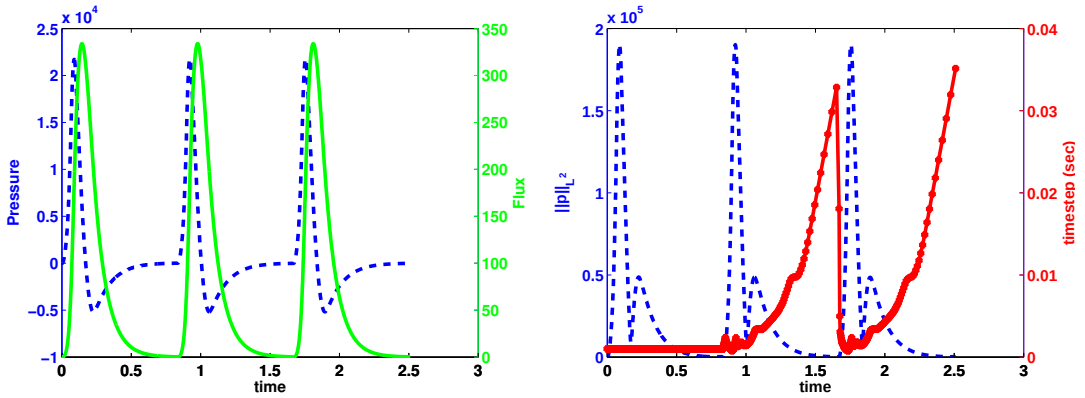


Figure 4.11: On the left, we plot the imposed flux (solid green line) and the computed pressure (dashed blue line) at the inflow section of the aorta. Note the shift between the pressure and flux pick. On the right, we display the $L^2(\Omega)$ norm of the pressure and the time step. The first stroke is simulated with a constant time step of $10^{-3}s$, in the next two heart beats time adaptivity is turned on.

steady flow. The value $P_{peak} = 383[\mathrm{g}/(\mathrm{cm} \cdot \mathrm{s}^2)]$ has been estimated from the length of the centreline and the average radius of the aorta by using the Poiseuille law. The tolerance in this way depends only on the physical properties (viscosity) of the blood and on geometry of the vessel.

The simulation of a heart beat with constant time step required 837 steps, while the adaptive one only 226. We stress again that the cost of a time step without error estimation is the same as with the estimate, since the estimator is a by-product of the segregated scheme.

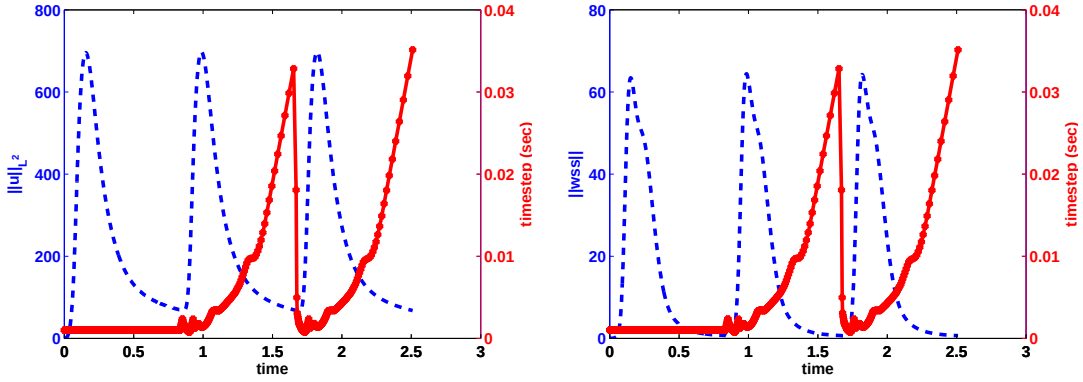


Figure 4.12: The evolution in time of the $L^2(\Omega)$ norm of the velocity field (on the left) and of the $L^2(\partial\Omega)$ norm of the wall shear stress (on the right). Note how the adaptive choice of the time step seems adequate for both velocity and shear stress dynamics.

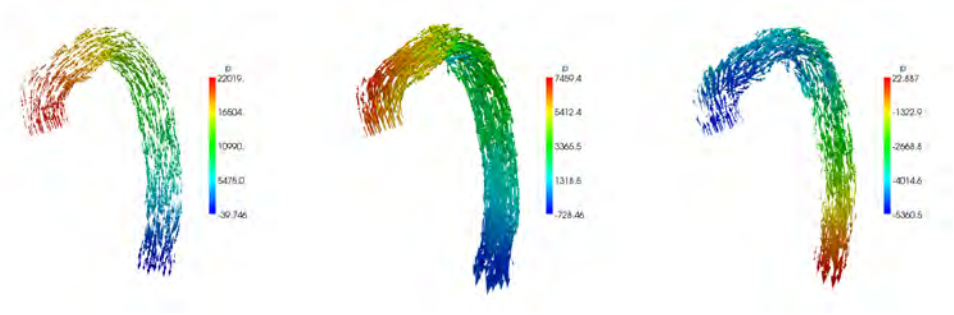


Figure 4.13: Blood flow in a human aorta in physiological conditions (72 heart beats per minute): from the left to the right we display the solution at t=0.088s, t = 0.144s, and t = 0.226s. Arrows show the flow direction, their lengths are proportional to the magnitude of the velocity, their colors reflect the magnitude of pressure field.

5 Implementation

In this Chapter we discuss implementation issues related to the algebraic splitting preconditioners presented in this thesis and parallel performances of our solvers.

In particular, as we have seen in Chapters 2 and 3, time and space discretization and linearization of the Navier-Stokes equations results in a sequence of large, sparse linear systems of the form

$$\begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}, \tag{5.1}$$

where $\mathbf{u}$, p are the finite element approximations of the velocity and pressure field respectively, and $\mathbf{f}_1$, $\mathbf{f}_2$ collect the discretization of the forcing term and boundary conditions. B^T is the pressure gradient matrix, B the velocity divergence matrix. The matrix C has the form

$$C = \sigma M + A$$

where M is the velocity mass matrix, A is the discretization of the viscous and convective terms, σ is related to time discretization.

High order algebraic splitting methods (see Section 3.2) consider an approximated LU block factorization of the matrix in (5.1) of the form

$$\begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} C & 0 \\ B & \Sigma \end{bmatrix} \begin{bmatrix} I & C^{-1}B^T \\ 0 & I \end{bmatrix} \approx \begin{bmatrix} C & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & C^{-1}B^T \\ 0 & Q \end{bmatrix}, \tag{5.2}$$

where $\Sigma = -BC^{-1}B^T$ is the Schur complement, $S = -BHB^T$ a first order in time

approximation of Σ, and Q is associated to the so called *pressure correction step*. Matrix H is the scaled inverse mass matrix $(\sigma M)^{-1}$, and it is in general a dense matrix. However, for some particular choices of the finite element spaces, the mass matrix M can be diagonalized (*mass lumping*) in order to preserve the sparsity of the matrix S (cfr. Section 2.5).

In this Chapter, we focus on efficient implementations of the pressure corrected schemes (5.2) targeting different problem sizes and hardware specifications. More specifically, we consider first the case of small-medium scale applications running on serial or multicore machines (such as a personal laptops or desktops), and later the case of large scale problems running on massively parallel architectures (high performance computing clusters or even clouds [99, 114]). In order to obtain efficient solvers, the algorithms and their implementation should be specifically designed according to the problem dimensions and hardware specifications.

In fact, in serial small-medium scale applications, effective implementation of the pressure corrected schemes (5.2) requires fast algorithms for the solution of linear systems involving the matrix S. Indeed, high order algebraic splittings method require to solve several linear systems in S to compute the action of Q^{-1}. Therefore, if memory constraints are not a limiting factor, direct factorizations of the matrix S may be a viable alternative to the iterative methods. In fact, the computational cost of the factorization can be amortized over the solution of several linear systems all involving the same matrix S and different right hand sides. Later in this Chapter, we will propose an efficient technique to compute a Cholesky factorization of S, based on the QR factorization of the matrix $H^{1/2}B^T$.

On the other hand, in massively parallel applications, the symmetric definite matrix S can be efficiently solved by conjugate gradient (CG) method and algebraic multigrid (AMG) preconditioning. Additionally Krylov subspace recycling or deflation techniques can be used to accelerate the convergence of the iterative method. The computational

bottleneck is then represented by the solution of the momentum matrix C. Indeed, the non-symmetry of C due to the convective terms poses a challenge to standard algebraic multigrid techniques (such as smoothed aggregation [120], or classical Ruge and Stuben coarsening [111]), and it demands for more sophisticated (and computationally expensive) techniques, such as multilevel additive Schwarz preconditioners or specialized coarse space corrections. We will then focus on scalable solvers for the coupled saddle point problem (5.1) with iterative methods of the GMRES family. As preconditioner, we will use a block triangular operator of the form

$$\mathcal{P} = \begin{bmatrix} \hat{C} & B^T \\ 0 & \hat{S}\hat{Q} \end{bmatrix}, \tag{5.3}$$

where $\hat{C}$, $\hat{S}$, $\hat{Q}$ are appropriate approximations (i.e. using inexact solves, or just preconditioners) of the operators C, S, Q.

The remainder of this chapter is structured as follows. In Section 5.1, we present the libraries and software for finite element assembly procedures (LifeV, [51]), parallel sparse linear algebra operators and solvers (Trilinos, [68]), and sparse factorization (SuiteSparseQR, [37, 38, 39]). In Section 5.2, we then discuss a direct solver based on SuiteSparseQR for the solution of linear systems involving the discrete laplacian matrix S, and we demonstrate the viability and effectiveness of this approach for small and medium scale applications on serial or multicore machines. We then proceed, in Section 5.3, to the presentation of a Trilinos-based general framework for the solution of block linear systems and block preconditioners. Finally, in Section 5.4, we present weak and strong scalability results of our implementation of the algebraic splitting preconditioners described in Chapter 3.

5.1 Libraries and Software

5.1.1 LifeV

LifeV is an open source library for the numerical solution of partial differential equations with the finite element method ([51], www.lifev.org).

LifeV is written in C++ and is entirely coded with an Object Oriented approach and advanced programming features. Paralization in LifeV is based on MPI. The project started in 1999 from the collaboration of the Department of Mathematics at EPFL, Lausanne, Switzerland, of the Department of Mathematics at Politecnico di Milano, Italy and the INRIA Rocquencourt, France, under the supervision of Alfio Quarteroni. Nowadays, LifeV is concurrently developed and maintained by CMCS Lausanne (group coordinator), MOX di Milano, REO and ESTIME and the E(CM)2 group at the Department of Mathematics and Computer Science at Emory University, Atlanta (GA), USA.

As for now, the library includes solvers for incompressible fluid dynamics, linear and non-linear structural problems, transport in porous media, fluid-structure interaction, electrical conduction in the heart. Even though this library is a research code oriented to the development and test of new numerical methods and algorithms, it is intended to be an effective tool for solving complex "real-life" engineering problems. One of the main field of applications (although not the only one) is cardiovascular mathematics.

Parallel management of sparse linear algebra operations, iterative solvers and preconditioner is based on the C++ library Trilinos [68], that will be described in details in the following section. Interfaces to other sparse linear algebra libraries (SuiteSparse, SuperLU, Tauchos, PETSC, Paradiso, MUMPS, SCALAPACK, DSCPACK) are available through Trilinos.

LifeV supports different mesh formats generated with some of the most popular mesh generation software, such as Netgen [113], Cubit, Gmsh [57], and also implements simple structured mesh generation algorithms. On-line or off-line mesh par-

titioning capabilities are available by using the multilevel graph partitioning library ParMetis [77]. Finally, results obtained in LifeV can be conveniently visualized and post-processed in Paraview [1] and can be stored in different (ASCII or binary) formats such as HDF5 [118], Ensight, VTK.

5.1.2 Trilinos

The Trilinos Project (www.trilinos.org, [68]) is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. The Trilinos project is divided in several packages, that can be used separately or combined together. Here we give a short description of the packages used in our applications in order to provide basics parallel sparse linear algebra operations (`Epetra` and `EpetraExt`), Krylov iterative solvers (`Aztecoo` and `Belos`), preconditioners (`Ifpack` and `ML`), general purpose utilities (`Teuchos`).

Basics parallel sparse linear algebra operations packages

`Epetra` provides the fundamental construction routines and service functions that are required for parallel linear algebra libraries and it represents the underlying foundation for all Trilinos solvers. `EpetraExt` is a set of extensions to `Epetra` to add support for additional capabilities, such as transformations (permutations, sub-block views, etc.), coloring support, and input/ouput (I/O) utilities.

In particular, `Epetra` provides classes to easily operate with maps (`Epetra_Map`), vectors (`Epetra_MultiVector`), matrices (`Epetra_CsrMatrix`) and other objects distributed on a parallel machine. Such classes support local and global permutations, overlapping Schwarz operations and many other data movement and redistribution algorithms. It also offers a specialization of the parallel sparse row matrix format optimized for finite element applications (`Epetra_FECrsMatrix`).

Krylov iterative solvers packages

`Aztecoo` provides an object-oriented interface to the well-known Aztec solver library. Furthermore, it allows flexible construction of matrix and vector arguments via `Epetra` matrix and vector classes.

`Belos` provides next-generation iterative linear solvers and a powerful linear solver framework. The collection of iterative linear solver algorithms in `Belos` achieves a high level of abstraction, since matrices, preconditioners, and vectors are treated as opaque objects with simple interfaces. These interfaces do not depend on the internal representation of these objects. `Belos`' solvers only use matrices and preconditioners as black-box "operators" that take one or more vectors as input, and return the same number of vectors as output. They only access vectors via a simple interface for summing vectors together or computing their inner products or norms. This framework includes the following abstract interfaces and implementations.

1. Abstract interfaces to linear algebra using traits mechanisms. This allows the user to leverage any existing investment in their description of matrices and vectors. In addition, `Belos` provides concrete linear algebra adapters for `Epetra`.

2. Abstract interfaces to orthogonalization; implementations of iterated classical Gram-Schmidt (ICGS), classical Gram-Schmidt with a DGKS correction step [73], and iterated modified Gram-Schmidt (IMGS) are included.

3. Abstract interfaces to iteration kernels and powerful solver managers that allows to specify user-defined stopping criterion and output information.

`Belos` can currently solve real-valued (often complex-valued), Hermitian and non-Hermitian, linear problems, via the following solvers:

1. Single-vector and block GMRES;

2. Single-vector and block CG;

3. Pseudo-block variants (perform single-vector algorithms simultaneously): pseudo-block CG, pseudo-block GMRES;

4. Recycling solvers: GCRO-DR and RCG;

5. Flexible variants: Flexible GMRES.

Preconditioner packages

The packages `Ifpack` (Inexact factorization) and `ML` (Multilevel algorithms) provide a suite of object-oriented algebraic preconditioners for the solution of preconditioned iterative solvers, implemented by the `Aztecoo` and `Belos` packages.

In particular, most `Ifpack` preconditioners can be rewritten as additive Schwarz methods of overlapping domain decomposition type. The user can adopt a minimal-overlap, or ask `Ifpack` to extend the overlap. The resulting preconditioner reads:

$$P_{IFPACK}^{-1} = \sum_{i=0}^{NumProcs-1} P_i A_i^{-1} R_i$$

where R_i is the restriction operator from the global vector to the overlapping subdomain i, and P_i is the prolongator operator. P_i is in general the transpose of R_i, but not necessarily (restricted additive Schwarz [27]). It is assumed that each subdomain solver A_i is assigned to a different processor.

A key component of the previous formula is the strategy to apply A_i^{-1}. Using `Ifpack`, this can be defined by one of the following.

1. Point or block relaxation preconditioners. Point relaxation preconditioners are probably the simplest iterative methods, and are generally used as smoothers in multilevel methods (for instance, within ML). For block relaxation, the local matrix A_i is divided into blocks, then a relaxation method is applied on the block

structure of A_i. Available choices are Jacobi, Gauss-Seidel, symmetric Gauss-Seidel, or their block counterpart.

2. Incomplete factorization (IC, ICT, ILU, ILUT) for symmetric and non-symmetric matrices, with dropping based on graph or on values.

3. An exact solve on each subdomain (such as LU or Cholesky).

`ML` is designed to solve large sparse linear systems of equations arising primarily from elliptic PDE discretizations. `ML` is used to define and build multigrid solvers and preconditioners, and it contains black-box classes to construct highly-scalable smoothed aggregation preconditioners. The primary goal of `ML` is to provide state-of-the-art iterative methods that perform well on massively parallel computers (with thousands of processes involved in the computations) and that at the same time are easy to use for application engineers.

Algebraic multigrid methods consists of two basics ingredient, a smoothing technique and a coarsering strategy. `ML` provides a large selection of parallel smoother methods: processor-based Gauss-Seidel type methods, processor-based block Jacobi, Gauss-Seidel and symmetric Gauss-Seidel, polynomial-based smoothers, Krylov subspace smoothers, additive Schwarz preconditioners (available through the `Ifpack` interface), direct subdomain solvers. Also `ML` provides several different algorithms to guide the type of coarsening and the inter-grid transfers (including the ability to drop weak coupling within the operator during inter-transfer construction), such as smoothed and unsmoothed aggregation, classical Ruge–Stüben AMG, finite element based two-level schemes, energy minimization prolongators.

General purpose utilities packages

Teuchos provides a suite of common tools for all other Trilinos packages. These tools include

1. low-level math utilities (BLAS/LAPACK wrappers, numerical scalar traits, serial dense matrices);

2. smart pointers (non-persisting or reference-counted pointers);

3. parameter lists and parsers (XML files and commandline);

4. outputting support, performance monitoring (timings and flop count), exception handling.

In particular, the class Teuchos::ParameterList is templated parameter list which holds a map of <key,value> pairs, where the "value" can be any type of object and the "key" is a string object. The type of parameter is chosen through the templated Set/Get methods, which allows it to be any standard type (float, double, int, ...), user defined class, or another parameter list.

5.1.3 SuiteSparseQR

SuiteSparseQR is a sparse QR factorization package based on the multifrontal method [37, 38]. The multifrontal method breaks the factorization of a large sparse matrix into a sequence of small dense frontal matrices. These frontal matrices are related to one another in a tree that specifies the order in which such matrices should be assembled and factorized. Within each frontal matrix, LAPACK [7] and the multithreaded BLAS [41],[59] enable the method to obtain high performance on multicore architectures. Parallelism across different frontal matrices is handled with Intel's Threading Building Blocks library [71].

More specifically, the factorization algorithm in SuiteSparseQR consists of three steps.

1. *Fill-reducing ordering* aims to reduce the computational cost of the factorization and the memory requirement for the allocation of the upper triangular matrix R,

by introducing an opportune permutation of the columns of A. Three different ordering strategies are available, AMD (Approximate Minimum Degree, [6]), COLAMD (a variant of AMD specific for rectangular matrices, [40]), Metis (a graph partitioning based reodering, [76]).

2. *Symbolic factorization* performs the symbolic supernodal Cholesky factorization of $A^TA = R^TR$, by solely analyzing A and without explicitly computing the sparsity pattern of A^TA. The first step is to compute the column elimination tree of A. This tree is required in the rest of symbolic analysis and also describes the data dependencies for the numeric factorization. Then the algorithm proceeds with the identification of the supernodes in the multifrontal factorization. A supernode in R represents a group of adjacent rows with identical or nearly identical nonzero pattern. Each supernode in R gives rise to a single frontal matrix in the numeric QR factorization. A frontal matrix is therefore a dense submatrix of A in which portions of the QR factorization of A are performed. Such frontal matrices are normally much smaller than the matrix A being factorized. Finally, the frontal matrices dependency tree is computed according to the supernodes definition and the columns elimination tree. Frontal matrices at the same level of the tree may be assembled and factorized in parallel. Symbolic factorization also provides an upper bound for the memory allocation in the rest of the computation.

3. *Numerical factorization* computes the dense Householder QR factorization of each frontal matrix starting from the leaves of the dependency tree, and then assembling and factorizing the parent frontal matrices. Such factorization is nearly the same as DGEQRF in LAPACK and uses BLAS Level-3 routines. Parallel BLAS implementation may be exploited at this point.

Parallelism

At least two opportunities for parallelism exist within a multifrontal sparse QR factorization. The first opportunity arises in the frontal matrix tree, the second in the numerical factorization of each frontal matrix.

More specifically, SuiteSparseQR determines, in the analysis phase, the frontal matrix tree, an amalgamated version of the column elimination tree, in which each node is a frontal matrix consisting of one or more rows of A. Next the size of each frontal matrix is determined, and an estimate of the floating point operations in each front is computed.

The goal of the parallel analysis phase is to assign frontal matrices to tasks, which are normally less than the number of frontal matrices. In order to ensure load balancing, the assignment of frontal matrices to tasks is performed as follows. For each frontal matrix f, the algorithm estimates the work in the subtree rooted at f. A *big* node is defined as a node for which the work in its subtree is greater than $\max(\omega/\alpha, \beta)$ where ω is the total flop count for the entire QR factorization, and α and β are two user-defined parameters that control the task tree granularity. Typically α should be at least twice the number of cores, and $\beta = 10^6$ [37]. All other nodes are *small*. The first pass assigns all small nodes to tasks. Suppose front f is a small node but its parent p is a big node. All fronts in the subtree rooted by f are placed in a new task. Additional children of p which are also small nodes are added to this task, until the task has at least $\max(\omega/\alpha, \beta)$ work. After that, a new task is created for the subtrees of the children of p.

The second pass assigns all big nodes to tasks. If all children of f are assigned to the same task, then f is also assigned to the same task. If f has no children, or if it has children assigned to different tasks, then a new task is created to which f is assigned.

To conclude the symbolic analysis phase, the algorithm precomputes the stack size for each task, so that memory is allocated by the main process before parallel tasks are

forked. Two tasks can share a stack if one is the ancestor of the other, so that there are only as many stacks as there are leaves in the task tree. In the numerical factorization phase, each task factorizes all frontal matrices in the subtree, and the results are left on the stack for the children tasks. No synchronization is needed except that a task can start when all its children are finished.

SuiteSparseQR exploits tree-based parallelism by using Intel's Threading Building Blocks (TBB) software. TBB is written in the C++ language and it is specifically designed to target shared memory multicore architectures [71, 109]. TBB-based applications need only to specify tasks and the dependencies between them, while TBB itself takes care of the scheduling and synchronization.

Additional parallelism within each task is exploited in the numerical factorization of the frontal matrices. For this second level of parallelism SuiteSparseQR relies on the multithreading already present in many high quality implementations of the Blas/Lapack libraries. For example, many vendor-specific Blas/Lapack implementations, such as the Intel Math Kernel Library (MKL, [72]) and the AMD Core Math Library (ACML, [5]), use OpenMP to parallelize their linear algebra routines.

Parallel performances achieved exploiting the two different levels of parallelism and their combination are discussed in Section 5.2.

5.2 On the numerical solution of the discrete Laplacian with direct methods

As pointed out in Chapter 3, several algebraic factorization methods/preconditioners for the INS entail the solution of $S = -BHB^T$. Since S is a symmetric negative definite matrix, a Cholesky factorization of $-S$ looks appealing. Let $D^T = H^{1/2}B^T$, Q the orthogonal matrix and R the upper triangular part matrix of the QR factorization

of D^T, then we can factorize S as

$$S = -BHB^T = -DD^T = -(QR)^T(QR) = -R^T R.$$

Observed that in this way, it is possible to compute the Cholesky factorization of $-S$ without explicitly assembling it, but using only the matrix D^T. Additionally notice that the orthogonal matrix Q does not need to be explicitly assembled, since it does not appear in the final Cholesky factorization.

The sparse QR factorization of the matrix D^T was originally proposed, for solving 2D problems, in [125], where the author used `QR27` library by P. Matstoms [88] to carry out the computations. Here, for 3D problems we use the C/C++ library *SuiteSparseQR* by Tim Davis [38],[37]. As we discussed in the previous section, the multifrontal QR method implemented in this suite splits the factorization of a large sparse matrix into a sequence of small dense frontal matrices. The latter are related one to the other in a tree, which provides one parallel level. Within each dense frontal matrix, LAPACK [7] and the Level-3 BLAS [41],[59] provide another layer for parallelism.

5.2.1 Parallel performance results

We tested the QR factorization suite by solving the Navier-Stokes equations in a unit cube, discretized with a uniform structured tetrahedral mesh. Three different mesh sizes h are considered: coarse (8 elements per edge), medium (16 elements per edge), fine (32 elements per edge). Two different couples of inf-sup compatible finite element are considered, P_1^b-P_1 and P_2-P_1 for velocity and pressure respectively. The former clearly features smaller and more sparse matrices .

In Tab. 5.1 we provide for each case the basic features (finite element space, mesh size h), size (number of rows n, columns m, and number of non zero entries nnz), results of the symbolic factorization (number of frontal matrices n_f using the *Metis*

FE	h	n (1e3)	m (1e3)	nnz (1e3)	n_f	serial time
P_1^b-P_1	1/8	11.403	0.729	52.419	57	0.19s
P_1^b-P_1	1/16	88.467	4.913	447.043	220	9.19s
P_1^b-P_1	1/32	697.635	35.937	3700.419	1386	7m35s
P_2-P_1	1/8	23.955	0.729	131.378	44	0.50s
P_2-P_1	1/16	181.539	4.913	1056.774	209	19.7s
P_2-P_1	1/32	1413.699	35.937	8490.734	1306	14m7s

Table 5.1: Finite element matrices used in the test of the parallel performance of SuiteSparseQR. The finite element space FE, the mesh size h, the number of rows n, the number of columns m, the number of non zero elements nnz, the number of frontal matrices n_f, and the serial factorization time are reported.

ordering), and serial factorization times. Notice that the QR factorization time is comparable to the timing of computing, in serial, an ILUT preconditioner of the momentum equation with Aztec/Trilinos.

In Fig. 5.1 we compare the speed-up achieved exploiting different levels of parallelism, namely

1. *tree-based only*: the large sparse matrix is split into dense smaller frontal matrices, which are grouped in subsets according to a dependency tree. Driven by load-balancing criteria, a scheduler assigns one or more of these frontal matrices subsets to tasks (basic units of work) that can be executed in parallel.

2. *Blas-based only*: the factorization of frontal matrices is performed sequentially. Parallelization is performed by a parallel implementation of the Householder reflection for the QR factorization in the Blas.

3. *combined*: both the levels of parallelism are exploited.

Numerical results are summarized in Tab. 5.2. In the comparison between the two levels of parallelism, we observe that tree-based parallelism provides better speed-up for small matrices, while Blas-based parallelism is less scalable but achieves better speed-up for large matrices. Notice that in the combined case the speed-up for the large P_2-P_1 matrix is about 15% higher than the one for P_1^b-P_1.

Matrix	Tree-based	Blas-based	Combined
P_1^b-P_1 small	1.72	1.16	1.79
P_1^b-P_1 medium	2.15	2.02	3.00
P_1^b-P_1 large	2.49	3.13	3.68
P_2-P_1 small	1.69	1.25	1.96
P_2-P_1 medium	2.03	2.08	3.08
P_2-P_1 large	2.37	3.09	4.27

Table 5.2: Maximum speed-up achieved on a Sun Microsystems SunFire V40z shared memory machine, with 4 Dual Core AMD Opteron(tm) Processors and 32 GB of memory running Linux.

Our test were performed on *crunch.mathcs.emory.edu*, a Sun Microsystems SunFire V40z shared memory machine, with 4 Dual Core AMD Opteron(tm) Processors and 32 GB of memory running linux. The total duration of the test was 7 hours 40 minutes, and 288 matrix factorizations were computed.

These results point out that the factorization performed by the SparseQR Suite is a viable approach in small-medium scale 3D computations using multicore architectures.

5.2.2 The effect of the ordering strategy

Fill-in reducing reordering of the columns of $H^{1/2}B^T$ is the first step of the symbolic analysis. It plays a fundamental role in the whole factorization since the ordering strategy strongly affects the structure of frontal matrices tree and the sparsity of the factor R.

In this test we compare the serial performances (factorization times) and parallel ones (speed-ups) for the two kinds of ordering which well fit the characteristics of the matrices in Tab. 5.1: *Metis* (a graph partitioning ordering), and *COLAMD* (a modification of the *approximate minimum degree* ordering).

For each of the two ordering strategies we consider different combinations of the maximum number of TBB and OpenMP tasks (n_{tbb} and n_{omp} respectively). The total duration of the test was 2 hours 34 minutes, and 84 matrix factorizations were com-

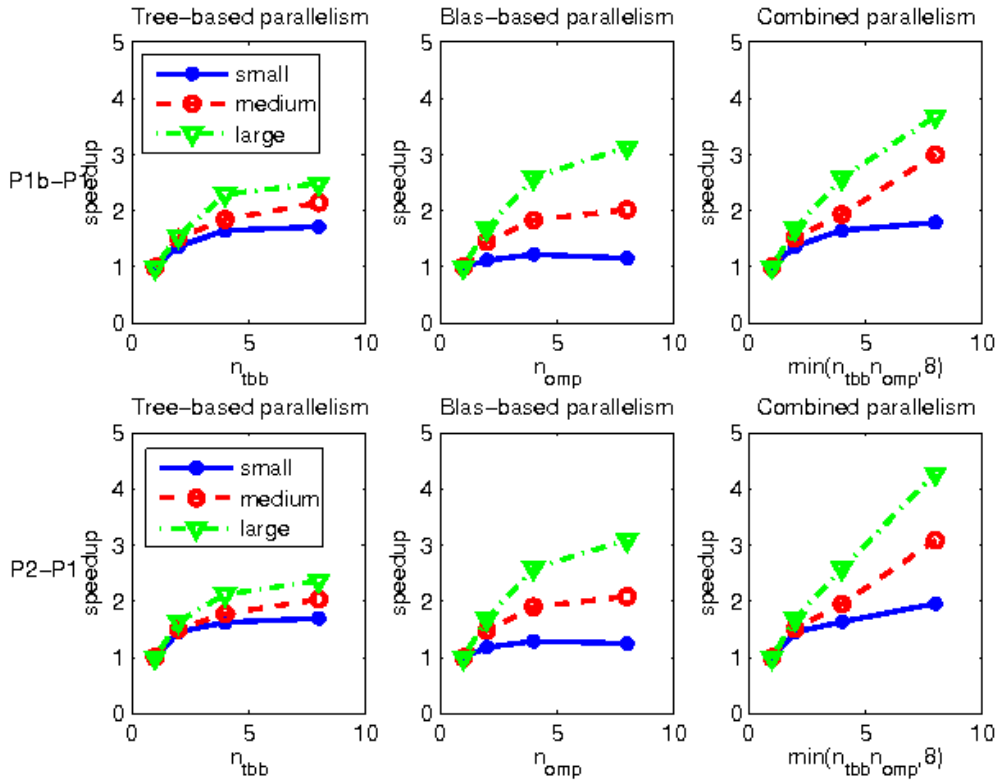


Figure 5.1: Speedup curves exploiting the different levels of parallelism (tree-based on the left, Blas-based in the middle, combined on the right). Results relative to P1b-P1 matrices are on top, relative to P2-P1 on bottom. Colors represent the matrix size: small matrices are in blue, medium in red, large in green.

	Metis		COLAMD	
	n_f	serial time	n_f	serial time
P1b-P1 small	57	0.19s	39	0.18s
P1b-P1 medium	220	9.19s	167	8.69s
P1b-P1 large	1386	7m35s	973	12m38s
P2-P1 small	44	0.50s	20	1.02s
P2-P1 medium	209	19.7s	134	22.4s
P2-P1 large	1306	14m7s	649	23m59s

Table 5.3: Number of frontal matrices and serial factorization time for the matrices in Tab.5.1 using the *Metis* and *COLAMD* ordering.

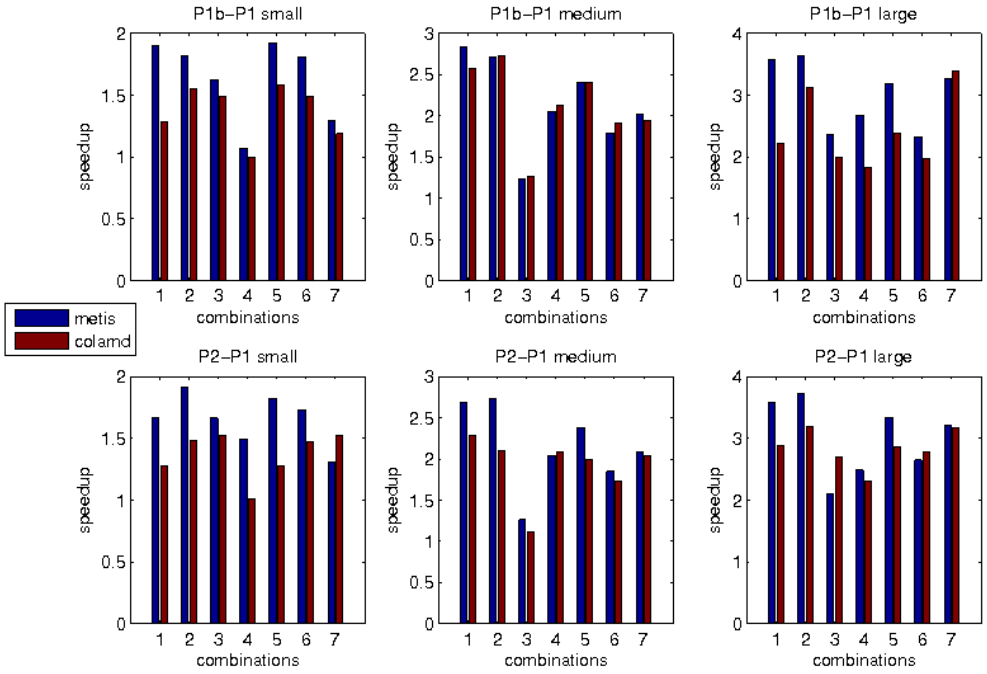
puted.

Tab. 5.3 shows the number of frontal matrices and serial factorization times for both orderings. Observe that using *Metis* we obtain a larger number of frontal matrices with respect to the one obtained using *COLAMD*. This is an advantage, since a larger number of frontal matrices provides more opportunities for tree-based parallelism. Additionally, frontal matrices obtained with the *Metis* reordering are, in average, smaller in sizes and this help in reducing the fill-in in the R factor. Serial factorization times are comparable for small and medium sized matrices, while *COLAMD* requires much more computational effort for large matrices.

Fig. 5.2 shows that *Metis* ordering, already faster in the serial case, also offers similar or better scalability properties compared to *COLAMD*, especially for $n_{tbb} = 8, 4$. This result is in agreement with the numerical test performed in [38], and it suggests that *Metis* ordering improves scalability for the tree-based parallelism.

5.3 Management of Block Operators in LifeV/Trilinos

In this section we describe one of the main contributions of this thesis to the LifeV library, that is the implementation of a flexible linear algebra module for the finite element library LifeV. The purpose of this module is to provide a general framework to the preconditioning of block linear systems (such as Navier-Stokes equations, PDE



n concurrent tasks	16			8			
combinations	1	2	3	4	5	6	7
(n_{tbb}, n_{omp})	(8,2)	(4,4)	(2,8)	(8,1)	(4,2)	(2,4)	(1,8)

Figure 5.2: Speedup for each matrix in Tab.5.1 using *Metis* and *COLAMD* ordering. The first 3 bars refers to $n = 16$, the last 4 to $n = 8$.

constrained optimization problems, fluid structure interaction) in order to simplify the implementation and the testing of different solution strategies.

To this aim, we intensively used advanced features of the C++ language (polymorphism, virtual classes, factories) in order to guarantee interoperability between the classes in the module. In particular, classes are organized in specific hierarchies according to their purpose, so that algorithmic features are hidden behind general interfaces. All classes derived from the same base class differ the one from the other only in the way they are set-up, but not in the way they are successively applied in the code. Base classes are purely virtual, derived classes are concrete (if no other class inherits from them) or implement the NVP idiom (non-virtual public interface) if they delegate the implementation of some features to children classes. The advantage of the NVP idiom is that it leads to a robust and extensible design of the class hierarchies [115, rule 39]. It is robust because the base class maintains control over the invariants, pre and post conditions of the methods in the public interface; it is extensible since the actual algorithm to produce the desired output is implemented by a derived class.

Additionally, in the implementation of the module, we introduced a strong separation between the finite element assembly routines in LifeV and the linear algebra operations involved in the construction of block preconditioners and the solution of coupled linear systems, so that the module can be also used as a stand-alone package.

The module is based on the parallel linear algebra Trilinos [68], and provides interfaces to the `Aztecoo`, `Belos`, `Ifpack`, `ML` packages.

5.3.1 Overview of the block linear algebra module

The module consists of two parts.

1. The `linear_algebra` part contains classes which provides a simple and powerful framework to manage linear operators defined in a matrix free fashion, block matrices and block vectors, preconditioners and Krylov solvers.

2. The `navier_stokes` part contains the implementation of state of the art preconditioners for the Navier-Stokes equations (Least-Squares-Commutator, High-Order-Yosida, Cahouet-Chabard, and a preliminary version of the Augmented Lagrangian preconditioner [18, 19]).

The main goal of this module is to allow the user to set at run time (through an XML files) the type of block preconditioner to use, the desired Trilinos package for Krylov iterative methods (`Aztecoo` or `Belos`), and the desired Trilinos package for preconditioning (`Ifpack` or `ML`).

To achieve such flexibility, we use dynamic polymorphism and opaque containers (`Teuchos::ParameterList`). In particular, the hierarchy of classes follows a simple and clear design. Base classes (prototypes) are purely abstract. They define the public interface and let the derived classes implement the public interface. Derived class can be concrete or abstract. Such classes may introduce new constructors or set-up methods with respect to the base class, but should not introduce new functionalities. Concrete derived classes should implement all the methods in the public interface of the prototype class, and should not have other classes derived from them. Abstract derived classes should still implement all the methods in the public interface of the base class but also define abstract virtual protected methods to provide a hook for customization to the derived classes. Factory are used to allocate concrete instances of an abstract class and for their setup.

General linear algebra classes

Here we discuss the features of the linear algebra framework. `LinearOperator` is an abstract class that provides the basic public interface for all the classes in the framework. Such class directly inherits from the abstract class `Epetra_Operator` defined by the `Epetra` module in Trilinos, and introduces some additional type definitions. `LinearOperator` defines the basic functionalities a derived class should provide in

order to be used in collaboration with other classes in the module. In particular, it provides methods to apply a linear operator (or its inverse) to a vector, and to construct vectors compatible with the domain and range of the operator.

The main reason to use `Epetra_Operator` as prototype for `LinearOperator` is that in this way all the objects of the framework can be directly used inside `Aztecoo` and `Belos`.

The following classes directly derive from `LinearOperator`.

1. `BlockOperator` is a lightweight interface to block structured linear operators and preconditioners. It provides functionality as matrix-vector multiplication, lower/upper triangular block operators solves.

2. `InvertibleOperator` accepts a `Epetra_Operator` object and it implements the method `ApplyInverse()` by using the Krylov methods implemented in `Aztecoo` (`AztecooOperator`) or `Belos` (`BelosOperators`). If desired, the user may provide as preconditioner another object of type `Epetra_Operator`.

3. `RowMatrixPreconditioner` accepts a `Epetra_CsrMatrix`, and it implements the method `ApplyInverse()` by computing an incomplete factorization (`IfpackPreconditioner`) or a AMG cycle (`MLPreconditioner`). A `RowMatrixPreconditioner` object can be provided as preconditioner to an `InvertibleOperator` object.

4. `ApproximatedInvertibleRowMatrix` accepts a `Epetra_CsrMatrix` object, and it implements the method `ApplyInverse()` by using preconditioned Krylov method. As private attributes, the class stores an `InvertibleOperator` and a `RowMatrixPreconditioner` objects.

The design of these classes is shown in Fig. 5.3 and Fig 5.4. The former is the inheritance diagram where each arrow points from the derived to the parent class. The

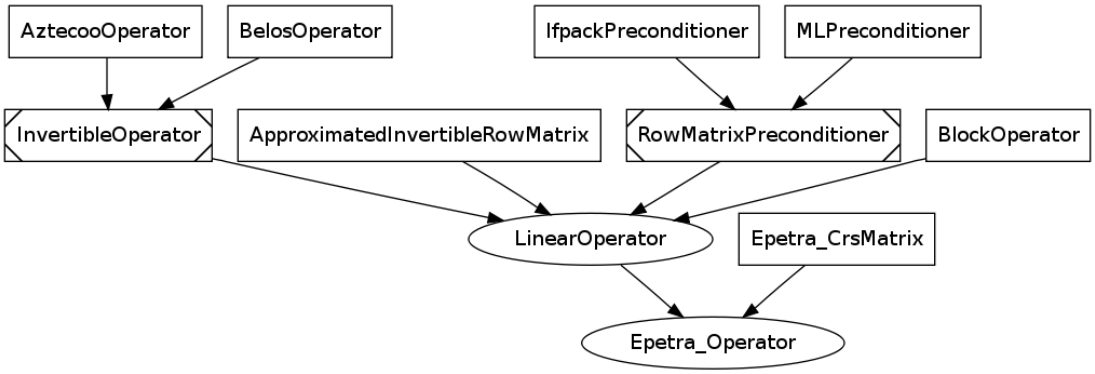


Figure 5.3: Inheritance diagram. Arrows point from the derived to the parent class.

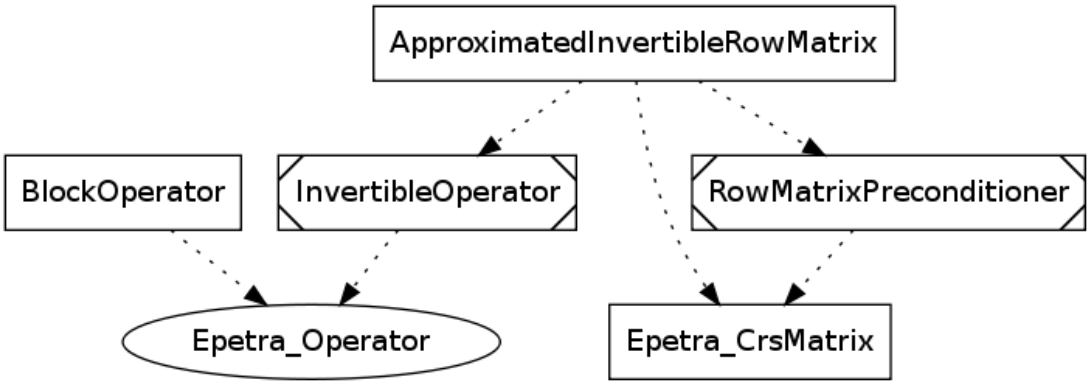


Figure 5.4: Collaboration diagram. Arrows point from the parent to the derived classes.

latter is the collaboration diagram and explains how class interacts between each other: there is an arrow connecting two classes if the first class has an instance of the second class as private (or protected) attribute. The shape of the boxes depends on the class design: we use ellipses to represent pure virtual classes, squares with diagonals for the virtual base classes in the NPI idioms, and squares for concrete classes.

Preconditioners for the Navier-Stokes equations

We now discuss the classes specifically implemented for the solution of the discrete Navier-Stokes problem. We distinguish them in two categories: linear operator classes, such as `DiscreteLaplacian`, `HighOrderYosida`, `LeastSquaresCommutator`,

and an helper class (`OseenOperatorManager` and derived class) used to set up the block preconditioner.

The helper class `OseenOperatorManager` is an abstract class (NPI idiom) from which classes implementing different methods are derived.
Such classes (`OseenOperatorManagerHOY`, `OseenOperatorManagerLSC`, `OseenOperatorManagerCC`, `OseenOperatorManagerAL`) implement the High Order Yosida, Least Squares Commutator, Cahouet-Chabard, Augmented Lagrangian block preconditioners. Such classes require as input the momentum, pressure gradient, and velocity divergence finite element matrices, and return an `InvertibleOperator` object for the solution of the Navier-Stokes saddle point problem. Given the fact that different preconditioners may require additional data (velocity mass matrix, time step, pressure mass matrix, for example) we use an opaque container (`MatrixContainer`) to provide such information.

In the next section, we will present scalability results for the algebraic pressure corrected preconditioners implemented in this framework.

5.4 Scalability Results

In this section we present scalability results for our implementation of the pressure corrected algebraic splitting preconditioners for the Navier-Stokes equations. In particular we consider both a weak and strong scaling test.

In a weak scaling test the number of unknowns proportionally increases with the number of processors involved in the simulation. In our application, this can be achieved by constructing, for the same domain Ω, a hierarchy of meshes with different size h. Perfect (weak) scaling is achieved if the measured wall time to complete the simulation is independent of the size of the problem. The (weak) scalability properties of an application depends on three factors: the computational complexity of the algorithm, the

quality of the implementation, and finally the hardware specifications. In particular, in order to obtain good weak scalability properties it is mandatory to develop solvers and preconditioners that are mesh independent, that is the number of iterations does not increase as the mesh is refined.

In a strong scaling test we keep the number of unknowns constant (i.e. we fix the number of elements in the mesh) while we increase the number of processors. Perfect (strong) scaling is achieved if the measured wall time is inversely proportional to the number of processes used in the simulation. In reality, this does not happen because of many factors, such as small load imbalances between processors, synchronization points in the application (such as MPI_Barrier or MPI_Wait), non-overlapping communications and computations, hardware and network performances.

Before presenting the numerical results let us briefly describe the overall organization of the Navier-Stokes solver application used in our tests.

Some operations are independent of the time advancing and are performed out of the temporal loop, while other operations need to be performed at each time step. These constitute the computationally-intensive kernel of the software. Schematically, we represent the stages of the application in Figure 5.5.

Here we detail each phase. Step (i) consists of the definition of the *computational domain* – given by the *mesh* – where the equations have to be solved numerically. Mesh generation is typically accomplished with in-house software (for structured meshes) or third-party software such as NetGen [113] and GMSH [57]. In the parallel application, the domain is *partitioned* so that each process takes care of only a subset of the global mesh. This splitting is achieved through the use of graph partitioning algorithms, such as those implemented in the library ParMETIS [77], guaranteeing a proper load balancing among processes. The load is measured as the number of mesh elements assigned to each process. At the same time, high quality partitionings should minimize the edge-cut, or the number of connections between disjoint partitions. This property is valuable

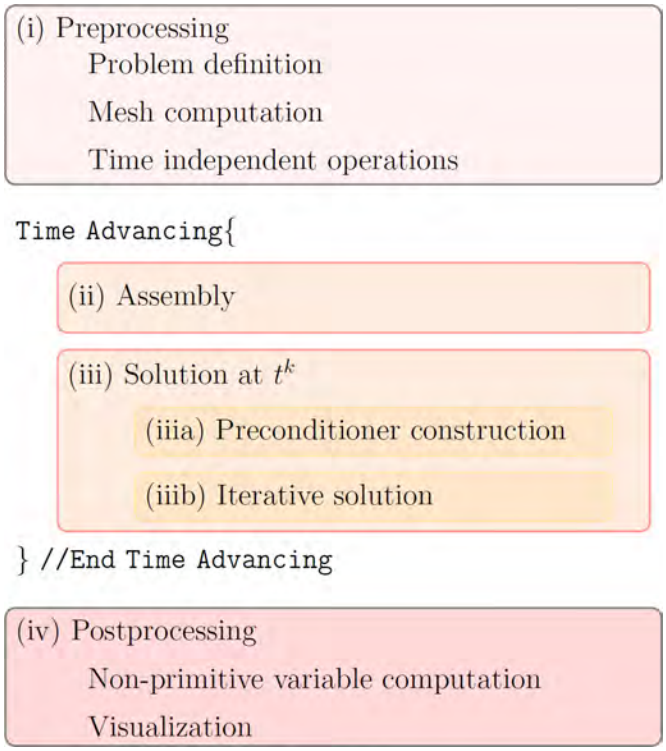


Figure 5.5: Steps for the numerical solution of a time-dependent PDE problem.

because in the considered application the communication between processes mostly involves the exchange of data related to neighbouring elements. Other operations of this step refer to all the computations that are time independent and can be performed once.

Step (ii) concerns the computation (or more specifically the *assembly*) of the matrices and vectors required for the construction of the discretized algebraic problem. Each process has local access to a subset of the matrices and vectors corresponding to its own portion of the mesh and requires limited access to adjacent submeshes. Assembly is carried out with algorithms provided by LifeV, while the data structures for the management of the distributed matrices and vectors are provided by Trilinos [68].

Trilinos also provides algorithms for the solution of the algebraic problem (Step (iii)). In particular, we use preconditioned iterative methods. We then distinguish Step (iiia) for the computation of the preconditioner, and Step (iiib) for the actual solution of the preconditioned system.

Step (iv) concerns the visualization of the solution to the differential problem, and can be delegated to third-party software such as Paraview [1]. This step may also include the computation of quantities of interest related to the solution u.

For the purpose of these tests, we are mostly concerned with Steps (ii) and (iii), which have a major impact on the entire computational cost of the application.

5.4.1 Weak scalability test

For this test, we use the popular benchmark proposed by C. R. Ethier and D. A. Steinman [47]. The exact solution of this problem is shown in Fig. 5.6. The initial velocity and boundary conditions are set to yield a Reynolds number of approximately 100.

The computational domain is a unit cube discretized with a structured tetrahedral mesh. Time discretization is second order BDF formulas. Due to accuracy considerations, the timestep Δt is chosen proportional to the mesh size h. At the coarsest level

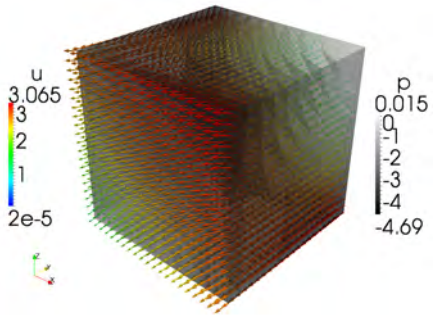


Figure 5.6: Solution of the problem proposed by C. R. Ethier and D. A. Steinman [47] for $t = 0.003$s. Arrows represent the vector field u, while in the cubic domain are shown isosurfaces of the scalar field p.

we have $h = 0.1$ and $\Delta t = 10^{-3}$.

We consider two different treatments of the convective terms: semi-implicit and fully explicit (cfr. Section 2.4.1). Semi-implicit treatment of the convective term leads to the solution of a generalized Oseen Problem. To cope with the non-symmetry in the momentum matrix $C_{\text{Oseen}} = \sigma M + K + N$, we add grad-div stabilization to the problem (cfr. Section 2.3). Fully explicit treatment of the convective term leads, on the other hand, to the solution of a generalized Stokes Problem, where the momentum matrix $C_{\text{Stokes}} = \sigma M + K$ is symmetric positive definite. Additionally, in this case we use the simplified strain tensor in order to decouple the components of the velocity field and induce a block diagonal structure in C_{Stokes}.

Concerning the choice of the finite element space, we consider both the case of Taylor Hood elements P2-P1 and MINI elements P1B-P1. The former couple of spaces yields to a second order in space approximation of the velocity field, while the latter only first order. The better accuracy of Taylor Hood elements is balanced by the fact that the linear systems to be solved are larger in dimensions and feature less sparse matrices. It is worth to notice another computational advantage of MINI elements compared to Taylor Hood elements. While for the formers the use of inexact quadrature rules gives accurate lumped velocity mass matrices, for the latter the same lumping technique does

not apply (cfr. Section 2.5), so that the consistent mass matrix should be used in the computations.

At each time step, we then solve the saddle point problem

$$\begin{bmatrix} C & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ \mathbf{f}_p \end{bmatrix}, \tag{5.4}$$

with preconditioned GMRES iterations (`Belos`). GMRES is restarted every 50 iterations, and vectors in the Krylov subspace are orthogonalized by using one or more steps of the modified Gram-Schmidt algorithm. Finally, by letting r_k the residual at iteration k, the stopping criterion is given by

$$\frac{\|r_k\|}{\|r_0\|} \leq 10^{-9}.$$

The block upper-triangular preconditioner

$$\mathcal{P} = \begin{bmatrix} C & B^T \\ 0 & \hat{\Sigma} \end{bmatrix}$$

is applied inexactly by using the AMG preconditioners available in `ML`. We use stationary AMG cycles, so that no flexible variant of GMRES is required. Here $\hat{\Sigma} = SQ_1$ stands for the HOY1 approximation of the pressure Schur Complement, and its application to a vector requires two approximate solves of the discrete laplacian matrix $S = -BHB^T$ and a few other matrix vector multiplications in the velocity space. For the symmetric positive matrices S and C_{Stokes} we used smoothed aggregation to build the coarse AMG hierarchies, while to deal with the non-symmetry induced by the convective term in C_{Oseen} we chose an unsmoothed aggregation algorithm specific for non-symmetric problems. In this case, the prolongation operator is constructed in order to maximize the reduction of the residual norm based on the spectral properties of the fine grid

operator and smoother. For both the symmetric and not symmetric case, we used three sweeps of the symmetric Gauss-Seidel algorithm as smoother. ParMETIS was used to improve load balancing at the coarser levels of the AMG hierarchy.

For this test we used `lagrange`, a hpc machine at CILEA, Italy, one of the most advanced supercomputing centres in Europe (`http://www.cilea.it/1/`). This supercomputer, when assembled, was placed at the 136-th position in the TOP500 list [89]. The machine is composed of HP ProLiant server blades with two hexacore Intel Xeon X5660 processors and 24 GB RAM each. The network infrastructure is provided by InfiniBand (IB) 4X Double Data Rate (DDR, 20 Gb/s bandwidth). Computing nodes are controlled by the CentOS version 5.6 operating system. 22 computational nodes of `lagrange` were specifically reserved to this benchmark. The code was compiled using the Intel Compiler Suites 12.1, and linked against CPU vendor-specific BLAS/LAPACK libraries (MKL [72]), ParMetis 3.1.1, Trilinos 10.6.4, LifeV 2.0.0.

We ran the simulations using up to a maximum of 8 processes per node (each node has two hexacore cpus) up to 128 processes, in order to balance overheads due to concurrent access to resources inside the node (memory and cache) on the one hand and communication between nodes on the other hand. For the last simulation (258 processes) we had to use the full capacity of each node due to limited number of nodes that were reserved to our experiment. This can partially explain the decrease of performances observed with 256 processes.

In Tab. 5.4, we report the results of our weak scalability test. For each choice of finite element spaces (Taylor Hood or MINI elements) and treatment of the convective term (semi-implicit or fully-explicit) we report the number n_p of processors involved in the simulation, the total number of degrees of freedom (N_{dof}), the average number of GMRES iterations (n_{it}) for each time step, the average wall time to solve the linear

system (t_{solve}), to compute the preconditioner (t_{prec}), and the total average time, including also the finite element assembly phase, to perform the entire time-step (t_{tot}). Timings are measured in seconds by using the `gettimeofday` function. All average values correspond to the first 100 time step of the simulation.

As expected, we observe that Taylor Hood elements incur in a much higher computational cost with respect to MINI elements, in fact not only more iterations are needed to achieve convergence but also each of them is more expensive. In the case of MINI elements, we observe a slight increase in the number of iterations as we increase the number of processors. This is expected due to the choice of parameters in the AMG cycles. However, in the case of Taylor Hood elements the number of iterations tends to decrease as we increase the number of degrees of freedom. In order obtain some insight on this phenomenon let us consider the inexact LU block factorization

$$\hat{\mathcal{A}} = \begin{bmatrix} C & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & C^{-1}B^T \\ 0 & Q \end{bmatrix},$$

where, for simplicity, we take $Q = I$ (Yosida scheme). Here we use the consistent velocity mass matrix M in the definition of the momentum matrix $C = \frac{\alpha_0}{\Delta t}M + A$ to preserve the second order of accuracy for the discrete velocity field, while we use the lumped mass matrix $\hat{M}$ given by (2.24) for an approximation of H to preserve sparsity in the discrete laplacian S. A straightforward computation gives

$$\mathcal{A} - \hat{\mathcal{A}} = \begin{bmatrix} 0 & 0 \\ & B(C^{-1} - \frac{dt}{\alpha_0}\hat{M}^{-1}) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & B(C^{-1} - \frac{\Delta t}{\alpha_0}M^{-1})B^T + \frac{\Delta t}{\alpha_0}B(M^{-1} - \hat{M}^{-1})B^T \end{bmatrix}.$$

The splitting error then consists of two contributions: the usual Yosida splitting error $\|B(C^{-1} - \frac{\Delta t}{\alpha_0}M^{-1})B^T\| = \mathcal{O}(\Delta t^2)$ and an additional error introduced by the mass lumping $\|\frac{\Delta t}{\alpha_0}B(M^{-1} - \hat{M}^{-1})B^T\| = \mathcal{O}(\Delta t)$. This second contribution leads to a slower

decay of the norm of the residual for the continuity equation resulting in a higher number of iterations for the solution of the saddle point problem (5.4). However, the mass lumping error vanishes as h goes to zeros, justifying the reduction in the number of iterations for finer meshes.

Independently of the choice of the finite element spaces, we notice that the fully explicit treatment is computationally cheaper than the semi-implicit ones, both regarding number of iterations and wall time. In the fully explicit case, in fact, the momentum matrix C has a block diagonal structure, and it allows for more efficient AMG preconditioning due to its symmetry.

In Fig. 5.7, we show the parallel speed-up achieved in the linear solver phase alone on the left, and in the whole time step computation (FE assembly, computation of the preconditioner, linear solver) on the right. For a weak scalability test, we define the parallel speed-up $\mathcal{S}$ as

$$\mathcal{S} = \frac{W_p/T_p}{W_s/T_s},$$

where W_p/T_p and W_s/T_s are the ratios between the number of degrees of freedom and the wall time required to complete the task in parallel and in serial, respectively. In case of perfect scalability $\mathcal{S}$ is proportional to the number of processes (black line). It is worth noticing that the most computationally expensive problem (Oseen Problem with Taylor Hood elements) is also the most scalable.

5.4.2 Strong scalability test

In this test we use a benchmark problem proposed in the Inaugural CFD Challenge Workshop at the ASME 2012 Summer Bioengineering Conference. This involves the simulation of blood flow in a giant aneurysm grown in the internal carotid artery. All the physical features are assigned ($\mu = 0.04$Poise and $\rho = 1$g/cm^3). We consider a sin-

		Semi-implicit convective term				Explicit convective term			
n_p	N_{dof}	n_{it}	t_{solve}	t_{prec}	t_{tot}	n_{it}	t_{solve}	t_{prec}	t_{tot}
1	29K	114	3.30	0.28	3.90	70	1.08	0.12	1.36
2	57K	110	3.42	0.34	4.20	71	1.23	0.17	1.59
4	113K	106	4.40	0.39	5.29	67	1.44	0.20	1.85
8	216K	105	6.39	0.48	7.41	66	2.12	0.24	2.58
16	428K	103	6.97	0.55	8.13	65	2.46	0.30	3.02
32	860K	102	7.33	0.59	8.57	64	2.70	0.33	3.32
64	1.66M	99	8.43	0.65	9.77	62	3.33	0.40	4.02
128	3.33M	91	9.08	0.70	10.55	61	4.35	0.45	5.16
256*	6.71M	80	13.98	1.16	16.29	57	6.78	0.79	8.25

		Semi-implicit convective term				Explicit convective term			
n_p	N_{dof}	n_{it}	t_{solve}	t_{prec}	t_{tot}	n_{it}	t_{solve}	t_{prec}	t_{tot}
1	23K	12	0.15	0.11	0.62	11	0.10	0.06	0.46
2	46K	12	0.18	0.17	0.77	10	0.10	0.10	0.55
4	93K	13	0.25	0.20	0.91	11	0.13	0.11	0.59
8	181K	15	0.42	0.25	1.16	12	0.23	0.13	0.70
16	363K	15	0.51	0.31	1.33	12	0.28	0.17	0.82
32	734K	15	0.59	0.37	1.49	12	0.35	0.20	0.91
64	1.43M	17	0.86	0.43	1.83	13	0.52	0.25	1.15
128	2.87M	18	1.20	0.49	2.30	14	0.79	0.31	1.53
256*	5.82M	21	2.10	0.78	3.70	16	1.28	0.56	2.48

* 12 processes per node instead of 8. (One mpi process per core)

Table 5.4: Weak scalability test (Taylor Hood Elements on the top, MINI Elements on the bottom): n_p represents the number of processes, N_{dof} the number of unknowns (DOFs), n_{it} the average number of iterations, t_{solve} the average linear solver time, t_{prec} the average preconditioner setup, t_{tot} the average total time per timestep. Timings are measured in seconds using the *gettimeofday* function.

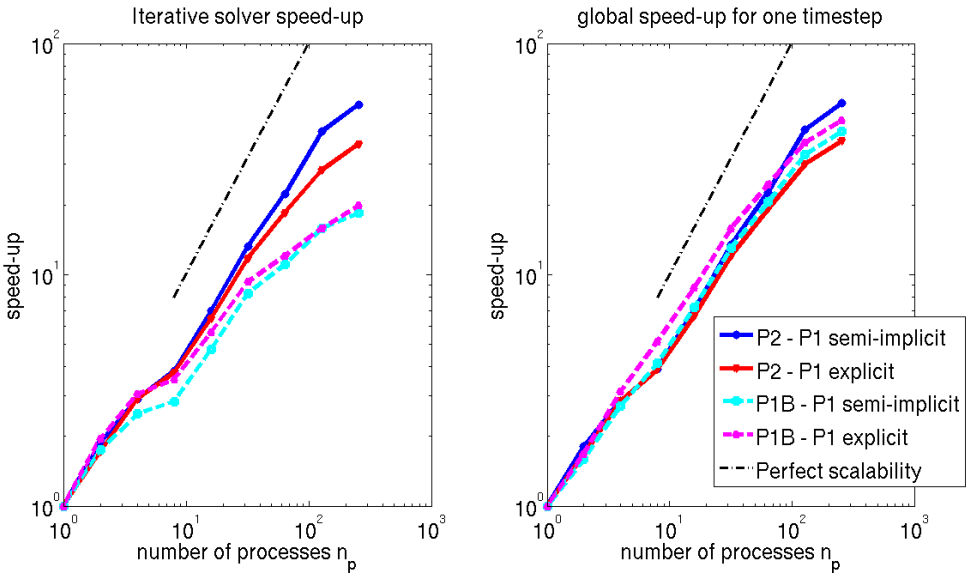


Figure 5.7: Parallel speed-ups: linear solve phase (on the left), complete time step (on the right).

gle scenario among the ones proposed in the original benchmark, that is the simulation of a pulsatile flow with a mean flow rate of 5.13mL/sec.

The physical phenomenon that we simulate is the movement of blood in the internal carotid artery and in the aneurysm, during a single heart beat. We assume that at the beginning of the simulation ($t = 0s$) the system is at rest, with the blood velocity and the pressure both being zero. In the first part of the simulation (from $t = 0s$ to $t = 0.05s$) we linearly increase the prescribed blood velocity at the inlet section until we reach the physiologic value corresponding to the beginning of the heart beat. We then prescribe a time-varying flow waveform with a period of $1s$, and we stop the simulation at $t = 1.05s$. The prescribed blood velocity at the inlet section in the last simulated frame coincides with the velocity prescribed at $t = 0.05s$.

In the numerical solver, the time derivative is discretized with a second order BDF, and we use the semi-implicit treatment of the convective term. The unknowns u and p are approximated using MINI elements. The number of unknowns in the linear system to be solved at each time step of the simulation was 3,162,146. We simulated 100

frames within the cardiac cycle (i. e. the simulation time step is $0.01s$). A snapshot of the computed solution is shown in Figure 5.8.

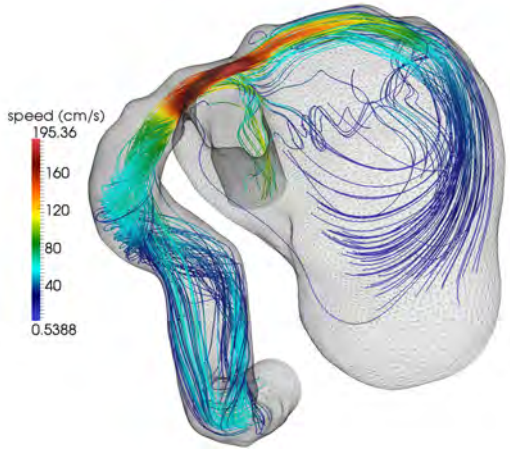


Figure 5.8: Streamlines of the velocity field in correspondence of the maximum flow rate over the cardiac cycle ($t = 0.28$s).

We performed the same test on different architectures to better evaluate the effects of hardware specifications on performances. In particular, we used the department cluster `puma`, the university one `ellipse`, the high performance computing machine `lonestar`[1], and the supercomputing cloud server `rockhopper`. Here we provide a short description of each machine, while we refer to [99] for a more detailed performance and cost analysis comparing on site resources, high performance computing clusters, and clouds for this benchmark.

The in-house computing cluster `puma` comprises thirty two four-core nodes. Each node includes two AMD 2214 processors, 8GB memory with 80GB local scratch disk space, while Gigabit Ethernet (1GbE) provides the network interconnections. This cluster is controlled by Linux CentOS 5.2, Rocks 5.1, and Portable Batch System (PBS) Torque 2.3.6. When we performed our experiment only 124 computational cores were available, due to hardware issue with one of the nodes.

[1]Access to this facility was granted by the National Science Foundation XSEDE Project, Extreme Science and Engineering Discovery Environment, `https://www.xsede.org`

The university cluster `ellipse` consists of 256 four-core nodes with AMD 2218 processors and 8GB RAM; Gigabit Ethernet provides the interconnection fabric. All nodes are controlled by CentOS 4.5.

The `lonestar` Linux Cluster consists of 1,888 compute nodes, with two 6-Core processors per node, for a total of 22,656 cores. It is configured with 44 TB of total memory and 276TB of local disk space. The theoretical peak compute performance is 302 TFLOPS. The system supports a 1PB global, parallel file storage, managed by the Lustre file system. Nodes are interconnected with InfiniBand technology in a fat-tree topology with a 40Gbit/sec point-to-point bandwidth.

`rockhopper` is supercomputing cloud appliance hosted by Indiana University. It consists of 11 Penguin Computing Altus 1804 servers, each containing four AMD Opteron 6172 12-cores processors and 128 GB of RAM. The total RAM in the system is 1.5 TB. Each server chassis has a QDR (40 Gbs) InfiniBand interconnect to the cluster's switch fabric. The `rockhopper` nodes run CentOS 5. Job management and scheduling are provided by the Sun Grid Engine (SGE) resource manager.

We performed a series of simulations on each platform, each time doubling the number of processors and starting from the minimal number of processes which could support the computation on a given target. This lower limit is governed by memory availability – the whole problem to be solved has to fit in memory in order to perform the computation.

In Tab. 5.5 we report the average wall time to compute the different phases of the time step: assembly of the finite element matrices (t_{ass}), computing the preconditioner (t_{prec}), solving the linear system (t_{solve}). Observe that the solution of the linear system is the phase were most of the total time for iteration is spent. This is, not only the most computationally intensive, but also the phase were the majority of collective communications happen. Each GMRES iterations, in fact, requires to compute a few vector inner products, that call the MPI function `MPI_AllReduce()` in their implementation. On

	puma			ellipse			lonestar			rockhopper		
n_p	t_{ass}	t_{prec}	t_{solve}	t_{ass}	t_{prec}	t_{solve}	t_{ass}	t_{prec}	t_{solve}	t_{ass}	t_{prec}	t_{solve}
8	–	–	–	19.28	28.98	863.34	9.82	13.87	559.29	–	–	–
16	9.77	4.15	507.07	9.52	12.23	468.24	5.07	7.37	306.06	9.32	11.71	428.78
32	4.99	7.61	296.67	5.07	6.92	289.51	2.61	3.94	143.34	5.24	6.91	242.24
64	2.71	4.32	170.47	3.07	5.99	204.82	1.32	2.25	79.87	3.11	4.50	177.02
128	2.26*	3.18*	134.54*	1.69	2.73	127.20	0.77	1.51	49.67	1.73	3.64	116.35
256	–	–	–	–	–	–	0.52	1.23	51.81	3.31	9.46	298.31

*These tests were performed on 124 processors instead of 128 due to hardware failure.

Table 5.5: Strong scalability test. We report the number of processors n_p, and the average time for time step to assemble the finite element matrices t_{ass}, compute the preconditioner t_{prec}, solve the linear system t_{solve}.

the converse, the most scalable part of the computation is the finite element assembly phase, where each processor need to communicate the shared degrees of freedom to its neighbours once local matrices are computed.

The graph in Fig. 5.9 shows a comparison of the performances of the different platforms (total average time per time-step), as a function of the number of computing cores. The black line represents the ideal case of perfect strong scalability, when the average time per iteration is inversely proportional to the number of processors. All platforms achieve good strong scaling up to 128 computing cores, while they show a significant decrease in performance for larger numbers of cores. In particular, the fastest execution case in our experiment corresponds to running the simulation with 128 computing cores on `lonestar`. On this platform speed up increases by a remarkable factor of 11.2 when passing from 8 computing cores to 128 computing cores. In the case of `rockhopper` we observe a severe loss of performances when we move from 128 to 258 processors. This is quite common for cloud computing platforms where out-of-node communication overheads may jeopardize scalability [99].

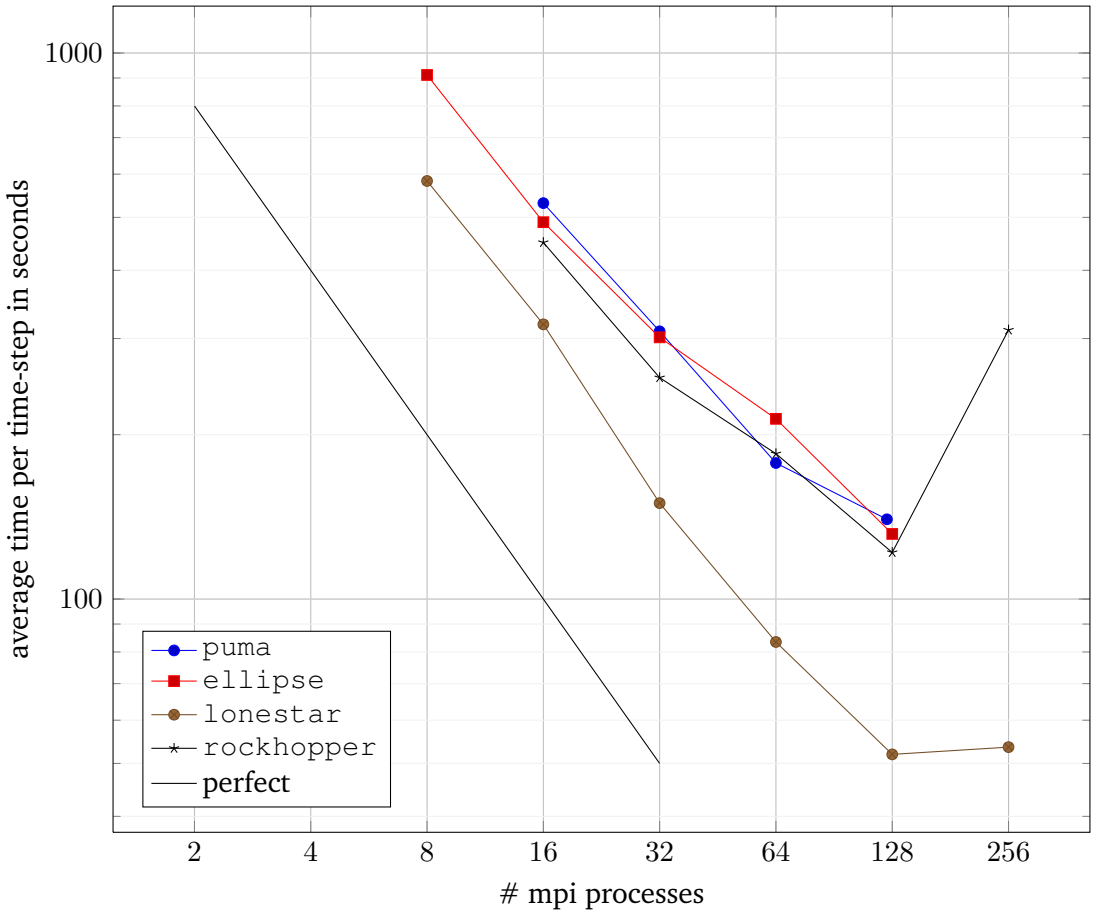


Figure 5.9: The average computation time per simulated time step, for the benchmarked architectures

6 The Brinkman Problem

The Brinkman model is a unified law governing the flow of a viscous fluid in cavity (Stokes equations) and in porous media (Darcy equations). It was initially proposed in [2], [3] as a homogenization technique for the Navier-Stokes equations. Typical applications of this model are in underground water hydrology, petroleum industry, automotive industry, biomedical engineering, and heat pipes modeling.

In this chapter we analyze a mixed formulation of the Brinkman problem, in which we introduce the (scaled) flow vorticity as additional unknown. We prove the well-posedness of this formulation in the abstract framework of Finite Element Exterior Calculus [12], extending to the Brinkman problem the analysis of the mixed formulation of the Stokes problem in [13], [9]. The numerical stability of the method is guaranteed by an appropriate choice of the finite elements spaces. The particular choice of Nédélec, Raviart-Thomas and piecewise discontinuous elements, in fact, reproduces the same embedding and mapping properties of the continuous spaces in the finite elements spaces. The linear system obtained after finite element discretization has a symmetric saddle point form. In contrast to the penalization methods for the Brinkman problem advocated in [24], [25], the proposed approach allows for a conforming discretization by standard finite elements. We also illustrate our analysis with numerical examples regarding the behavior of the discretization errors in the $H(\mathrm{curl};\Omega)$-norm of the vorticity, in the $H(\mathrm{div};\Omega)$-norm of the velocity and in the $L^2(\Omega)$-norm of the pressure. We numerically observe some suboptimal error behavior of the (scaled) vorticity in the Darcy limit, while velocity and pressure exhibit uniform error decay rates with

respect to the inverse permeability coefficient $k(\mathbf{x})$.

Additionally, the proven stability of our mixed discretization allows us to consider effective preconditioning techniques for the discrete saddle point problem. Following the approach advocated in [87], we construct in [122] a block diagonal preconditioner with optimal convergence properties based on the stability analysis of the continuous problem. Such preconditioner has on its main diagonal the finite element matrices corresponding to the $H(\text{curl};\Omega)$, $H(\text{div};\Omega)$, and $L^2(\Omega)$ norms involved in the stability estimates. To improve the efficiency of the preconditioner, we resort to the auxiliary space multigrid preconditioner for $H(\text{div})$ and $H(\text{curl})$ problems recently developed in [80, 81]. The resulting (inexact) block-diagonal preconditioner appears to be robust with respect to constant and smoothly varying coefficient $k(\mathbf{x})$. The general case of fully robust (with respect to coefficient variation) preconditioning method is still an open problem. One possible approach is to develop appropriate adaptive (element-based) algebraic multigrid techniques that respect the entire de Rham complex on coarse levels. For some progress in that direction, we refer to [98], [82].

The remainder of the present Chapter is structured as follows. In Section 6.1, we briefly derive the mixed formulation of the Brinkman problem based on the Hodge Laplacian and in Section 6.2 we provide a stability estimate for the weak formulation. In Section 6.3 we address the numerical discretization of the mixed formulation with Nédélec, Raviart-Thomas and piecewise polynomial discontinuous finite element which leads to a large sparse saddle point linear system. In Section 6.4 we present numerical results illustrating the error behavior of our discretization schemes for the case of constant coefficient, smoothly varying coefficient, and discontinuous coefficient. In Section 6.5 we derive an optimal preconditioner with respect to the mesh size. We also investigate an augmented Lagrangian approach in order to improve the robustness of the preconditioner with respect to the PDE coefficients. Finally, in Section 6.6 we investigate the performance of our preconditioner, including some parallel scalability tests

for the solution of the Brinkman Problem with constant and space-dependent inverse permeability coefficient.

The presentation and the numerical results in this Chapter closely follows manuscripts [122, 123], submitted for publication.

6.1 Mixed formulation of the Brinkman Problem

We assume that Ω is a bounded simply connected domain in $\mathbb{R}^3$ with a regular (Lipschitz continuous) simply connected boundary $\partial\Omega$ that has well-defined (almost everywhere) unit outward normal vector $\mathbf{n} \in \mathbb{R}^3$.

The generalized Brinkman problem reads

$$\begin{cases} -\nu\, \Delta \mathbf{u} + k(\mathbf{x})\, \mathbf{u} + \nabla p = \mathbf{f}(\mathbf{x}), & \forall\, \mathbf{x} \in \Omega \\ \text{div}\, \mathbf{u} = g(\mathbf{x}), & \forall\, \mathbf{x} \in \Omega \\ \mathbf{u} \times \mathbf{n} = \mathbf{g}, & \text{on } \partial\Omega \\ -p + \nu\, \text{div}\, \mathbf{u} = h, & \text{on } \partial\Omega, \end{cases} \tag{6.1}$$

where $\nu \geq 0$ is the fluid viscosity and $k(\mathbf{x})$ is the inverse permeability of the medium. The challenge of this problem is when the coefficient $k = k(\mathbf{x})$ ranges over the interval $[a, b]$ with $a = \mathcal{O}(1)$ and $b = \mathcal{O}(1/\varepsilon)$. In the part of the domain Ω with $k = \mathcal{O}(1)$, the PDE behaves like a Stokes problem, whereas in the rest of the domain it behaves like the Darcy equations.

In the present work, for simplicity, we assume natural boundary conditions on $\partial\Omega$. However, other set of boundary conditions, like the essential boundary conditions ($\mathbf{u} \cdot \mathbf{n} = u_n$, $\boldsymbol{\sigma} \times \mathbf{n} = \boldsymbol{\sigma}_\tau$), can also be treated in a similar way. For the Hodge Laplacian, natural boundary conditions are also known in the literature as *electric boundary conditions* while the essential ones as *magnetic boundary conditions* due to the close relation with Maxwell's equations. In our work, we do not consider the case of full

Dirichlet boundary condition, as the mixed formulation is harder to analyze; it leads to suboptimal discretization error behavior [9].

To obtain a mixed formulation of the Brinkman problem (6.1), we exploit the vector identity

$$\Delta \mathbf{u} = \nabla \text{ div } \mathbf{u} - \text{curl curl } \mathbf{u},$$

and we define the (scaled) vorticity variable as

$$\boldsymbol{\sigma} = \varepsilon \text{ curl } \mathbf{u}, \quad \text{with } \varepsilon = \sqrt{\nu}.$$

After some straightforward manipulations, our mixed formulation reads

$$\begin{cases} \boldsymbol{\sigma} - \varepsilon \text{ curl } \mathbf{u} = 0, & \forall\, \mathbf{x} \in \Omega \\ \varepsilon \text{ curl } \boldsymbol{\sigma} - \varepsilon^2 \nabla \text{ div } \mathbf{u} + k(\mathbf{x})\, \mathbf{u} + \nabla p = \mathbf{f}(\mathbf{x}), & \forall\, \mathbf{x} \in \Omega \\ \text{div } \mathbf{u} = g(\mathbf{x}), & \forall\, \mathbf{x} \in \Omega \\ \mathbf{u} \times \mathbf{n} = \mathbf{g}, & \text{on } \partial\Omega \\ -p + \varepsilon^2 \text{div } \mathbf{u} = h, & \text{on } \partial\Omega. \end{cases} \tag{6.2}$$

6.1.1 Functional spaces and orthogonal decompositions

To come up with the weak formulation of the system (6.2) and using the notation introduced in Section 2.2, we define the functional spaces $\mathbf{Q}$, $\mathbf{R}$ and W as

- $\mathbf{Q} \equiv H(\text{curl}; \Omega) := \left\{ \boldsymbol{\sigma} \in \mathbf{L}^2(\Omega) \mid \text{curl } \boldsymbol{\sigma} \in \mathbf{L}^2(\Omega) \right\}$, equipped with the norm

$$\|\boldsymbol{\tau}\|_{\mathbf{Q}}^2 = \|\boldsymbol{\tau}\|^2 + \|\text{curl } \boldsymbol{\tau}\|^2;$$

- $\mathbf{R} \equiv H(\text{div}; \Omega) := \left\{ \mathbf{u} \in \mathbf{L}^2(\Omega) \mid \text{div } \mathbf{u} \in L^2(\Omega) \right\}$, equipped with the norm

$$\|\mathbf{v}\|_{\mathbf{R}}^2 = \|\mathbf{v}\|^2 + \|\text{div } \mathbf{v}\|^2;$$

- $W \equiv L^2(\Omega)$, equipped with the norm

$$\|q\|_W^2 = \|q\|^2.$$

We denote with $\mathbf{Q}^*$, $\mathbf{R}^*$, and W^* the dual spaces of $\mathbf{Q}$, $\mathbf{R}$, and W, respectively. It is clear that in the case of essential (magnetic) boundary conditions, the respective spaces $\mathbf{Q}$, $\mathbf{R}$ are proper subsets of $H(\text{curl};\Omega)$, $H(\text{div};\Omega)$; $\mathbf{Q}$, $\mathbf{R}$ then consist of functions with vanishing tangential or normal boundary traces.

We need next the orthogonal decompositions of $H(\text{curl};\Omega)$ and $H(\text{div};\Omega)$ associated with the kernels of the respective differential operators. Such decompositions are of utmost importance in the stability analysis of the mixed problem, and for the derivation of the auxiliary space multigrid preconditioners for such spaces [10, 11].

The assumption that the domain Ω is simply connected with simply connected boundary, also called *contractible* (that is, any cycle in Ω is homologous to a point in Ω), guarantees that de Rham sequence

$$0 \to H^1(\Omega)\backslash\mathbb{R} \xrightarrow{\nabla} H(\text{curl};\Omega) \xrightarrow{\text{curl}} H(\text{div};\Omega) \xrightarrow{\text{div}} L^2(\Omega) \to 0 \tag{6.3}$$

is exact. In other words, the space of the k-harmonic forms has dimension 0 for all $k = 1, 2, 3$, or, equivalently, we have that $\text{range}(\nabla)^\perp \cap \ker(\text{curl}) = \{0\}$ and $\text{range}(\text{curl})^\perp \cap \ker(\text{div}) = \{0\}$. Therefore, we have that these nullspaces are defined as $\mathbf{X} = \{\boldsymbol{\tau} \in H(\text{curl};\Omega) \mid \exists \psi \in H^1(\Omega) : \ \boldsymbol{\tau} = \nabla\psi\}$ and $\mathbf{Y} = \{\mathbf{v} \in H(\text{div};\Omega) \mid \exists \boldsymbol{\psi} \in H(\text{curl};\Omega) : \mathbf{v} = \text{curl}\, \boldsymbol{\psi}\}$. The original spaces admit the following orthogonal decompositions [119, Appendix A]:

$$H(\text{curl};\Omega) = \mathbf{X} \oplus \mathbf{X}^\perp, \quad H(\text{div};\Omega) = \mathbf{Y} \oplus \mathbf{Y}^\perp,$$

which imply the Poincaré inequalities

$$\begin{aligned} & \|\boldsymbol{\tau}^{\perp}\|_{H(\text{curl};\Omega)} && \leq \gamma \|\text{curl } \boldsymbol{\tau}^{\perp}\|_{\mathbf{L}^2(\Omega)}, && \forall\, \boldsymbol{\tau}^{\perp} \in \mathbf{X}^{\perp}, \\ \text{and} \quad & \|\mathbf{v}^{\perp}\|_{H(\text{div};\Omega)} && \leq \beta \|\text{div } \mathbf{v}^{\perp}\|_{L^2(\Omega)}, && \forall\, \mathbf{v}^{\perp} \in \mathbf{Y}^{\perp}, \end{aligned} \tag{6.4}$$

with constants γ and β which depend only on the domain Ω. The orthogonal complement spaces are characterized as

$$\begin{aligned} & \mathbf{X}^{\perp} = \{\boldsymbol{\tau} \in H(\text{curl};\Omega) \cap H(\text{div};\Omega) \mid \text{div } \boldsymbol{\tau} = 0 \text{ in } \Omega,\ \boldsymbol{\tau} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega\}, \\ \text{and} \quad & \mathbf{Y}^{\perp} = \left\{\mathbf{v} \in H(\text{div};\Omega) \mid \exists \phi \in H_0^1(\Omega) :\ \mathbf{v} = \nabla\phi\right\}. \end{aligned}$$

For more complete characterization of such spaces and for the orthogonal decompositions in the case of essential (magnetic) boundary conditions we refer, for example, to [119, Appendix A] or [90, 101].

6.1.2 Weak formulation

To obtain the weak formulation of (6.2), we first multiply the system by $[\boldsymbol{\tau},\ \mathbf{v},\ q] \in \mathbf{Q} \times \mathbf{R} \times W$ and integrate over Ω:

$$\begin{cases} \displaystyle\int_{\Omega} \boldsymbol{\sigma} \cdot \boldsymbol{\tau}\, d\Omega - \int_{\Omega} \varepsilon\, (\text{curl } \mathbf{u}) \cdot \boldsymbol{\tau}\, d\Omega = 0, \qquad \forall\, \boldsymbol{\tau} \in \mathbf{Q} \\ \displaystyle\int_{\Omega} \varepsilon\, (\text{curl } \boldsymbol{\sigma}) \cdot \mathbf{v}\, d\Omega - \int_{\Omega} \varepsilon^2\, (\nabla \text{div } \mathbf{u}) \cdot \mathbf{v}\, d\Omega + \int_{\Omega} k(\mathbf{x})\, \mathbf{u} \cdot \mathbf{v}\, d\Omega + \int_{\Omega} (\nabla p) \cdot \mathbf{v}\, d\Omega = \\ \qquad\qquad = \displaystyle\int_{\Omega} \mathbf{f} \cdot \mathbf{v}\, d\Omega, \qquad \forall\, \mathbf{v} \in \mathbf{R} \\ \displaystyle\int_{\Omega} (\text{div } \mathbf{u})\, q\, d\Omega = \int_{\Omega} g\, q\, d\Omega, \qquad \forall\, q \in W. \end{cases}$$

By letting $\gamma_T(\boldsymbol{\tau}) = (\mathbf{n} \times \boldsymbol{\tau}|_{\partial\Omega}) \times \mathbf{n}$, $\gamma_n(\mathbf{v}) = \mathbf{n} \cdot \mathbf{v}|_{\partial\Omega}$ and exploiting some basic

integration-by-parts identities [91, Chapter 3], the non-conforming terms become

$$-\int_\Omega \varepsilon\ (\text{curl}\ \mathbf{u})\cdot\boldsymbol{\tau}\ d\Omega = -\int_\Omega \varepsilon\ \mathbf{u}\cdot(\text{curl}\ \boldsymbol{\tau})\ d\Omega + \int_{\partial\Omega} \varepsilon\ (\mathbf{u}\times\mathbf{n})\cdot\gamma_T(\boldsymbol{\tau})\ dS = \\ -\int_\Omega \varepsilon\ \mathbf{u}\cdot(\text{curl}\ \boldsymbol{\tau})\ d\Omega + \int_{\partial\Omega} \varepsilon\ \mathbf{g}\cdot\gamma_T(\boldsymbol{\tau})\ dS,$$

and

$$-\int_\Omega \varepsilon^2\ (\nabla\text{div}\ \mathbf{u})\cdot\mathbf{v}\ d\Omega + \int_\Omega (\nabla\ p)\cdot\mathbf{v}\ d\Omega = \\ = \int_\Omega \varepsilon^2\ (\text{div}\ \mathbf{u})\ (\text{div}\ \mathbf{v})\ d\Omega - \int_\Omega p\ (\text{div}\ \mathbf{v})\ d\Omega + \int_{\partial\Omega} (p-\varepsilon^2\ \text{div}\ \mathbf{u})\ \gamma_n(\mathbf{v})\ dS = \\ = \int_\Omega \varepsilon^2\ (\text{div}\ \mathbf{u})\ (\text{div}\ \mathbf{v})\ d\Omega - \int_\Omega p\ (\text{div}\ \mathbf{v})\ d\Omega - \int_{\partial\Omega} h\ \gamma_n(\mathbf{v})\ dS.$$

Therefore a weak solution $\boldsymbol{\sigma}\in\mathbf{Q}$, $\mathbf{u}\in\mathbf{R}$, and $p\in W$ of (6.2) satisfies the following variational problem

$$\begin{cases} \displaystyle\int_\Omega \boldsymbol{\sigma}\cdot\boldsymbol{\tau}\ d\Omega - \int_\Omega \varepsilon\ \mathbf{u}\cdot(\text{curl}\ \boldsymbol{\tau})\ d\Omega = -\int_{\partial\Omega} \varepsilon\ \mathbf{g}\cdot\gamma_T(\boldsymbol{\tau})\ dS, \qquad \forall\ \boldsymbol{\tau}\in\mathbf{Q} \\ \displaystyle\int_\Omega \varepsilon\ (\text{curl}\ \boldsymbol{\sigma})\cdot\mathbf{v}\ d\Omega + \int_\Omega \varepsilon^2\ (\text{div}\ \mathbf{u})\ (\text{div}\ \mathbf{v})\ d\Omega + \int_\Omega k(\mathbf{x})\ \mathbf{u}\cdot\mathbf{v}\ d\Omega - \int_\Omega p\ (\text{div}\ \mathbf{v})\ d\Omega = \\ \qquad\qquad = \displaystyle\int_\Omega \mathbf{f}\cdot\mathbf{v}\ d\Omega + \int_{\partial\Omega} h\ \gamma_n(\mathbf{v})\ dS, \qquad \forall\ \mathbf{v}\in\mathbf{R} \\ \displaystyle\int_\Omega (\text{div}\ \mathbf{u})\ q\ d\Omega = \int_\Omega g\ q\ d\Omega, \qquad \forall\ q\in W. \end{cases} \tag{6.5}$$

The variational problem above can be written as an abstract saddle-point problem of the form

Problem 6.1.1 Find $(\boldsymbol{\sigma}, \mathbf{u}, p) \in \mathbf{Q} \times \mathbf{R} \times W$ such that

$$\left\{ \begin{array}{llll} m(\boldsymbol{\sigma}, \boldsymbol{\tau}) & -c^*(\mathbf{u}, \boldsymbol{\tau}) & & = F(\boldsymbol{\tau}), \quad \forall\, \boldsymbol{\tau} \in \mathbf{Q} \\ -c(\boldsymbol{\sigma}, \mathbf{v}) & -a(\mathbf{u}, \mathbf{v}) - d(\mathbf{u}, \mathbf{v}) & +b^*(p, \mathbf{v}) & = G(\mathbf{v}), \quad \forall\, \mathbf{v} \in \mathbf{R} \\ & b(\mathbf{u}, q) & & = H(q), \quad \forall\, q \in W \end{array} \right. \tag{6.6}$$

where

$$\begin{array}{rcll} m(\boldsymbol{\sigma},\ \boldsymbol{\tau}) & = & (\boldsymbol{\sigma},\ \boldsymbol{\tau}), & \boldsymbol{\sigma},\ \boldsymbol{\tau} \in \mathbf{Q} \\ c(\boldsymbol{\sigma},\ \mathbf{v}) & = & \varepsilon\,(\mathcal{C}\boldsymbol{\sigma},\ \mathbf{v}), & \boldsymbol{\sigma} \in \mathbf{Q},\ \mathbf{v} \in \mathbf{R}, \\ a(\mathbf{u},\ \mathbf{v}) & = & \varepsilon^2(\mathcal{B}\mathbf{u},\ \mathcal{B}\mathbf{v}), & \mathbf{u},\ \mathbf{v} \in \mathbf{R}, \\ d(\mathbf{u},\ \mathbf{v}) & = & (k(\mathbf{x})\ \mathbf{u},\ \mathbf{v}), & \mathbf{u},\ \mathbf{v} \in \mathbf{R}, \\ b(\mathbf{u},\ q) & = & (\mathcal{B}\mathbf{u},\ q), & \mathbf{u} \in \mathbf{R},\ q \in W. \end{array} \tag{6.7}$$

Here $\mathcal{C}$, $\mathcal{B}$ denote some differentiation operators (curl and div), and $F \in \mathbf{Q}^*$, $G \in \mathbf{R}^*$, $H \in W^*$ are bounded functionals that take into account volume forces and boundary conditions.

6.2 Well-posedness of the mixed variational formulation

Before studying the well-posedness of Problem 6.1.1, we recall the following classical result due to Babuška [14].

Theorem 6.2.1 Let $\mathfrak{B} : X \times X \to \mathbb{R}$ be a symmetric bounded bilinear form on a Hilbert space X which satisfies the inf-sup condition

$$\alpha := \inf_{0 \neq y \in X} \sup_{0 \neq x \in X} \frac{\mathfrak{B}(x, y)}{\|x\|_X \|y\|_X} > 0. \tag{6.8}$$

Then, if $\mathfrak{F}(y)$ belongs to X^*, the problem of finding $\mathfrak{B}(x, y) = \mathfrak{F}(y)$ for all $y \in X$ is well-posed. Moreover, it has a unique solution x for each $\mathfrak{F} \in X^*$, and the following a

priori estimate holds

$$\|x\|_X \leq \frac{1}{\alpha}\|\mathfrak{F}\|_{X^*}.$$

In what follows, we assume that the inverse permeability coefficient $k(\mathbf{x})$ belongs to $L^\infty(\Omega) \cap L^2(\Omega)$, and we let $k_{\max} < +\infty$ be a constant such that

$$0 \leq k(\mathbf{x}) \leq k_{\max} \text{ almost everywhere in } \Omega.$$

Since by assumption the inverse permeability $k(\mathbf{x})$ and the viscosity ε^2 can not both vanish at the same time, there exist constants $\kappa_{\min} > 0$ and $\kappa_{\max} < +\infty$, such that

$$0 < \kappa_{\min} = k_{\min} + \varepsilon^2 \leq k(\mathbf{x}) + \varepsilon^2 \leq k_{\max} + \varepsilon^2 = \kappa_{\max}.$$

For the stability analysis of Problem 6.1.1, we use the following weighted norms

$$\begin{aligned} \|\boldsymbol{\tau}\|^2_{\mathbf{Q}_w} &= \|\boldsymbol{\tau}\|^2 + \frac{\varepsilon^2}{\kappa_{\max}}\|\mathcal{C}\boldsymbol{\tau}\|^2, \\ \|\mathbf{v}\|^2_{\mathbf{R}_w} &= \kappa_{\min}\|\mathbf{v}\|^2_{\mathbf{R}} = \kappa_{\min}\|\mathbf{v}\|^2 + \kappa_{\min}\|\mathcal{B}\mathbf{v}\|^2, \\ \|q\|^2_{W_w} &= \frac{1}{\kappa_{\max}}\|q\|^2. \end{aligned} \tag{6.9}$$

Remark 1 The weighted space $\mathbf{Q}_w$ can be interpreted as intersection of Hilbert spaces, that is $\mathbf{Q}_w = \mathbf{L}^2(\Omega) \bigcap \frac{\varepsilon^2}{\kappa_{\max}} H(\text{curl}; \Omega)$. This space coincides with to $H(\text{curl}; \Omega)$ as a set, but the corresponding norm approaches the $\mathbf{L}^2(\Omega)$ norm as $\frac{\varepsilon^2}{\kappa_{\max}}$ tends to zero. In other words, the smaller this ratio, the less control we have on the $\mathbf{L}^2(\Omega)$ norm of the curl of functions in $\mathbf{Q}_w$. For a detailed characterization of the intersection of Hilbert spaces and of their dual spaces we refer, for example, to [20, Chapter 2] and [87].

The stability of the mixed formulation of the Brinkman problem, which is a central result in this chapter, is stated below.

Theorem 6.2.2 If $\varepsilon \geq 0$ and $k(\mathbf{x}) \in L^\infty(\Omega) \cap L^2(\Omega)$, $0 < \kappa_{\min} \leq k(\mathbf{x}) + \varepsilon^2 \leq \kappa_{\max}$ almost everywhere in Ω, then for given continuous linear functionals $F \in \mathbf{Q}_w^*$, $G \in \mathbf{R}_w^*$, $H \in W_w^*$ the generalized Brinkman problem (6.1.1) admits a unique solution and the following a priori estimate holds

$$\|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w} + \|\mathbf{u}\|^2_{\mathbf{R}_w} + \|p\|^2_{W_w} \leq C(\Omega)(\|F\|^2_{\mathbf{Q}_w^*} + \|G\|^2_{\mathbf{R}_w^*} + \|H\|^2_{W_w^*}),$$

where $C(\Omega)$ is a constant depending only on the domain.

Proof. The variational mixed formulation of the Brinkman problem (6.1.1) can be written in the form of Theorem 6.2.1, where $X = \mathbf{Q}_w \times \mathbf{R}_w \times W_w$ and $\mathfrak{B}$ denotes the symmetric bilinear form

$$\begin{aligned}\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q) = (\boldsymbol{\sigma}, \boldsymbol{\tau}) - \varepsilon(\mathbf{u}, \mathcal{C}\boldsymbol{\tau}) \\ - \varepsilon(\mathcal{C}\boldsymbol{\sigma}, \mathbf{v}) - (k(\mathbf{x})\ \mathbf{u}, \mathbf{v}) - \varepsilon^2(\mathcal{B}\mathbf{u}, \mathcal{B}\mathbf{v}) + (p, \mathcal{B}\mathbf{v}) + (\mathcal{B}\mathbf{u}, q), \quad (6.10)\end{aligned}$$

and $\mathfrak{F}(\boldsymbol{\tau}, \mathbf{v}, q) = F(\boldsymbol{\tau}) + G(\mathbf{v}) + H(q)$.

Then, the following Lemmas 6.2.3 and 6.2.4 imply Theorem 6.2.2, by establishing respectively the boundedness of $\mathfrak{B}$ and the inf-sup condition (6.8) required by Babuška's theorem (Theorem 6.2.1).

□

Lemma 6.2.3 Under the assumptions of Theorem 6.2.2, there exists a constant M such that for any $(\boldsymbol{\sigma}, \mathbf{u}, p),\ (\boldsymbol{\tau}, \mathbf{v}, q)\ \in \mathbf{Q}_w \times \mathbf{R}_w \times W_w$

$$\begin{aligned}|\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q)| \leq \\ M\frac{\kappa_{\max}}{\kappa_{\min}} \left(\|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w} + \|\mathbf{u}\|^2_{\mathbf{R}_w} + \|p\|^2_{W_w}\right)^{\frac{1}{2}} \left(\|\boldsymbol{\tau}\|^2_{\mathbf{Q}_w} + \|\mathbf{v}\|^2_{\mathbf{R}_w} + \|q\|^2_{W_w}\right)^{\frac{1}{2}}. \quad (6.11)\end{aligned}$$

Proof. Simple applications of Cauchy–Schwarz inequalities show that the desired boundedness holds with $M = 2$. Indeed, we have

$$
\begin{aligned}
|\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q)| &\leq \|\boldsymbol{\sigma}\|\|\boldsymbol{\tau}\| + \varepsilon\|\mathbf{u}\|\|\mathcal{C}\boldsymbol{\tau}\| + \varepsilon\|\mathcal{C}\boldsymbol{\sigma}\|\|\mathbf{v}\| + \\
&\quad + k_{\max}\|\mathbf{u}\|\|\mathbf{v}\| + \varepsilon^2\|\mathcal{B}\mathbf{u}\|\|\mathcal{B}\mathbf{v}\| + \|p\|\|\mathcal{B}\mathbf{v}\| + \|\mathcal{B}\mathbf{u}\|\|q\| \leq \\
&\leq \left(\|\boldsymbol{\sigma}\|^2 + \frac{\varepsilon^2}{\kappa_{\max}}\|\mathcal{C}\boldsymbol{\sigma}\|^2 + (\kappa_{\max} + k_{\max})\|\mathbf{u}\|^2 + (\kappa_{\max} + \varepsilon^2)\|\mathcal{B}\mathbf{u}\|^2 + \frac{1}{\kappa_{\max}}\|p\|^2\right)^{\frac{1}{2}} \\
&\left(\|\boldsymbol{\tau}\|^2 + \frac{\varepsilon^2}{\kappa_{\max}}\|\mathcal{C}\boldsymbol{\tau}\|^2 + (\kappa_{\max} + k_{\max})\|\mathbf{v}\|^2 + (\kappa_{\max} + \varepsilon^2)\|\mathcal{B}\mathbf{v}\|^2 + \frac{1}{\kappa_{\max}}\|q\|^2\right)^{\frac{1}{2}}. \quad \square
\end{aligned}
$$

Lemma 6.2.4 Under the assumptions of Theorem 6.2.2, there exists a constant $\alpha(\Omega)$, depending only on the Poincaré constants γ and β in (6.4), such that for any $(\boldsymbol{\sigma}, \mathbf{u}, p) \in \mathbf{Q}_w \times \mathbf{R}_w \times W_w$, there exists a triplet $(\boldsymbol{\tau}, \mathbf{v}, q) \in \mathbf{Q}_w \times \mathbf{R}_w \times W_w$ such that

$$
\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q) \geq \alpha(\Omega)(\|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w} + \|\mathbf{u}\|^2_{\mathbf{R}_w} + \|p\|^2_{W_w})^{\frac{1}{2}}(\|\boldsymbol{\tau}\|^2_{\mathbf{Q}_w} + \|\mathbf{v}\|^2_{\mathbf{R}_w} + \|q\|^2_{W_w})^{\frac{1}{2}}.
$$

Proof. By the orthogonal decomposition of $H(\mathrm{div})$, we can write

$$
\mathbf{u} = \mathcal{C}\boldsymbol{\varphi}^{\perp} + \mathbf{u}^{\perp}, \text{ where } \boldsymbol{\varphi}^{\perp} \in \mathbf{X}^{\perp} \text{ and } \mathbf{u}^{\perp} \in \mathbf{Y}^{\perp}.
$$

Thanks to the orthogonal decompositions and the Poincaré inequalities in (6.4), we have

$$
\|\boldsymbol{\varphi}^{\perp}\|_{\mathbf{Q}} \leq \gamma\|\mathcal{C}\boldsymbol{\varphi}^{\perp}\|_{\mathbf{R}} \leq \gamma\|\mathbf{u}\|_{\mathbf{R}}, \quad \|\mathbf{u}^{\perp}\|_{\mathbf{R}} \leq \beta\|\mathcal{B}\mathbf{u}\|. \tag{6.12}
$$

In a similar way, using also the exactness of (6.3), we can write $p = \mathcal{B}\mathbf{w}^{\perp}$, where $\mathbf{w}^{\perp} \in \mathbf{Y}^{\perp}$ and

$$
\|\mathbf{w}^{\perp}\|_{\mathbf{R}} \leq \beta\|\mathcal{B}\mathbf{w}^{\perp}\| = \beta\|p\|_W. \tag{6.13}
$$

For some positive real numbers a_i, $i = 1, 2, 3, 4$ (to be specified later on), we choose

$$\boldsymbol{\tau} = \boldsymbol{\sigma} - a_1\boldsymbol{\varphi}^{\perp} \in \mathbf{Q}, \quad \mathbf{v} = -\mathbf{u} - a_2\mathcal{C}\boldsymbol{\sigma} + a_3\mathbf{w}^{\perp} \in \mathbf{R}, \quad q = p + a_4\mathcal{B}\mathbf{u} \in W. \tag{6.14}$$

By direct substitution, we obtain

$$\begin{aligned}
\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q) &= \|\boldsymbol{\sigma}\|^2 - a_1(\boldsymbol{\sigma}, \boldsymbol{\varphi}^{\perp}) - \varepsilon(\mathbf{u}, \mathcal{C}\boldsymbol{\sigma}) + \varepsilon a_1(\mathbf{u}, \mathcal{C}\boldsymbol{\varphi}^{\perp}) \\
&+ \varepsilon(\mathcal{C}\boldsymbol{\sigma}, \mathbf{u}) + \varepsilon a_2\|\mathcal{C}\boldsymbol{\sigma}\|^2 - \varepsilon a_3(\mathcal{C}\boldsymbol{\sigma}, \mathbf{w}^{\perp}) + \|\sqrt{k(\mathbf{x})}\mathbf{u}\|^2 + a_2(k(\mathbf{x})\mathbf{u}, \mathcal{C}\boldsymbol{\sigma}) - a_3(k(\mathbf{x})\mathbf{u}, \mathbf{w}^{\perp}) \\
&+ \varepsilon^2\|\mathcal{B}\mathbf{u}\|^2 + \varepsilon^2 a_2(\mathcal{B}\mathbf{u}, \mathcal{B}\mathcal{C}\sigma) - \varepsilon^2 a_3(\mathcal{B}\mathbf{u}, \mathcal{B}\mathbf{w}^{\perp}) - (p, \mathcal{B}\mathbf{u}) - a_2(p, \mathcal{B}\mathcal{C}\boldsymbol{\sigma}) + a_3(p, \mathcal{B}\mathbf{w}^{\perp}) \\
&+ (\mathcal{B}\mathbf{u}, p) + a_4\|\mathcal{B}\mathbf{u}\|^2.
\end{aligned}$$

Exploiting orthogonality and obvious simplifications, noticing that $(\mathbf{u},\ \mathcal{C}\boldsymbol{\varphi}^{\perp}) = \|\mathcal{C}\boldsymbol{\varphi}^{\perp}\|^2$, the above expression reduces to

$$\begin{aligned}
\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q) &= \|\boldsymbol{\sigma}\|^2 + \varepsilon a_1\|\mathcal{C}\boldsymbol{\varphi}^{\perp}\|^2 - a_1(\boldsymbol{\sigma}, \boldsymbol{\varphi}^{\perp}) \\
&+ \varepsilon a_2\|\mathcal{C}\boldsymbol{\sigma}\|^2 + \|\sqrt{k(\mathbf{x})}\mathbf{u}\|^2 + a_2(k(\mathbf{x})\mathbf{u}, \mathcal{C}\boldsymbol{\sigma}) - a_3(k(\mathbf{x})\mathbf{u}, \mathbf{w}^{\perp}) \\
&+ \varepsilon^2\|\mathcal{B}\mathbf{u}\|^2 - \varepsilon^2 a_3(\mathcal{B}\mathbf{u}, p) \\
&+ a_3\|p\|^2 + a_4\|\mathcal{B}\mathbf{u}\|^2.
\end{aligned}$$

Applying Cauchy-Schwarz and Young inequalities, and the Poincaré inequalities (6.12), (6.13), the remaining inner products are estimated as follows:

$$|a_1(\boldsymbol{\sigma}, \boldsymbol{\varphi}^{\perp})| \leq \frac{1}{2}\|\boldsymbol{\sigma}\|^2 + \frac{a_1^2}{2}\|\boldsymbol{\varphi}^{\perp}\|^2 \leq \frac{1}{2}\|\boldsymbol{\sigma}\|^2 + \frac{a_1^2}{2}\gamma^2\|\mathcal{C}\boldsymbol{\varphi}^{\perp}\|^2,$$

$$|a_2(k(\mathbf{x})\mathbf{u}, \mathcal{C}\boldsymbol{\sigma})| \leq \frac{1}{4}\|\sqrt{k(\mathbf{x})}\mathbf{u}\|^2 + a_2^2\|\sqrt{k(\mathbf{x})}\mathcal{C}\boldsymbol{\sigma}\|^2 \leq \frac{1}{4}\|\sqrt{k(\mathbf{x})}\mathbf{u}\|^2 + a_2^2 k_{\max}\|\mathcal{C}\boldsymbol{\sigma}\|^2,$$

$$|a_3(k(\mathbf{x})\mathbf{u}, \mathbf{w}^{\perp})| \leq \frac{1}{4}\|\sqrt{k(\mathbf{x})}\mathbf{u}\|^2 + a_3^2\|\sqrt{k(\mathbf{x})}\mathbf{w}^{\perp}\|^2 \leq \frac{1}{4}\|\sqrt{k(\mathbf{x})}\mathbf{u}\|^2 + a_3^2 k_{\max}\beta^2\|p\|^2,$$

$$|\varepsilon^2 a_3(\mathcal{B}\mathbf{u}, p)| \leq \varepsilon^2 \|\mathcal{B}\mathbf{u}\|^2 + \frac{\varepsilon^2}{4} a_3^2 \|p\|^2.$$

Substituting the above inequalities in the expression for $\mathfrak{B}$ we have

$$\begin{aligned}\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q) \geq \frac{1}{2}\|\boldsymbol{\sigma}\|^2 + \left(\varepsilon a_2 - a_2^2 k_{\max}\right) \|\mathcal{C}\boldsymbol{\sigma}\|^2 \\ + \frac{k_{\min}}{2}\|\mathbf{u}\|^2 + \left(\varepsilon a_1 - \frac{a_1^2}{2}\gamma^2\right) \|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2 + a_4 \|\mathcal{B}\mathbf{u}\|^2 \\ + \left(a_3 - a_3^2(k_{\max}\beta^2 + \frac{\varepsilon^2}{4})\right) \|p\|^2.\end{aligned}$$

Then we choose $a_1 = \frac{\varepsilon}{\gamma^2} = \mathcal{O}(\varepsilon)$, $a_2 = \frac{\varepsilon}{2k_{\max}+\varepsilon^2} = \mathcal{O}(\frac{\varepsilon}{\kappa_{\max}})$, $a_3 = \frac{2}{(4k_{\max}\beta^2+\varepsilon^2)} = \mathcal{O}(\frac{1}{\kappa_{\max}})$ and $a_4 = \kappa_{\min}$, so that

$$\begin{aligned}\mathfrak{B}(\boldsymbol{\sigma}, \mathbf{u}, p; \boldsymbol{\tau}, \mathbf{v}, q) \geq \frac{1}{2}\|\boldsymbol{\sigma}\|^2 + \frac{\varepsilon^2}{4\kappa_{\max}} \|\mathcal{C}\boldsymbol{\sigma}\|^2 \\ + \frac{k_{\min}}{2}\|\mathbf{u}\|^2 + \frac{\varepsilon^2}{2\gamma^2}\|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2 + \kappa_{\min}\|\mathcal{B}\mathbf{u}\|^2 + \frac{1}{4k_{\max}\beta^2 + \varepsilon^2}\|p\|^2.\end{aligned}$$

Recalling that $\mathbf{u} = \mathbf{u}^\perp + \mathcal{C}\boldsymbol{\varphi}^\perp$ and by using the second inequality in (6.12), we have

$$\begin{aligned}&\tfrac{k_{\min}}{2}\|\mathbf{u}\|^2 + \tfrac{\varepsilon^2}{2\gamma^2}\|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2 + \kappa_{\min}\|\mathcal{B}\mathbf{u}\|^2 \\ &\geq \tfrac{k_{\min}}{2}\left(\|\mathbf{u}^\perp\|^2 + \|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2\right) + \tfrac{\varepsilon^2}{2\gamma^2}\|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2 + \tfrac{\kappa_{\min}}{2\beta^2}\|\mathbf{u}^\perp\|^2 + \tfrac{\kappa_{\min}}{2}\|\mathcal{B}\mathbf{u}\|^2 \\ &\geq \left(\tfrac{k_{\min}}{2} + \tfrac{\kappa_{\min}}{2\beta^2}\right)\|\mathbf{u}^\perp\|^2 + \left(\tfrac{k_{\min}}{2} + \tfrac{\varepsilon^2}{2\gamma^2}\right)\|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2 + \tfrac{\kappa_{\min}}{2}\|\mathcal{B}\mathbf{u}\|^2 \\ &= \left((\tfrac{1}{2} + \tfrac{1}{2\beta^2})k_{\min} + \tfrac{\varepsilon^2}{2\beta^2}\right)\|\mathbf{u}^\perp\|^2 + \left(\tfrac{k_{\min}}{2} + \tfrac{\varepsilon^2}{2\gamma^2}\right)\|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2 + \tfrac{\kappa_{\min}}{2}\|\mathcal{B}\mathbf{u}\|^2 \\ &\geq \tfrac{1}{2}\min(1,\ \tfrac{1}{\gamma^2},\ \tfrac{1}{\beta^2})\ \kappa_{\min}\ \left(\|\mathbf{u}^\perp\|^2 + \|\mathcal{C}\boldsymbol{\varphi}^\perp\|^2\right) + \tfrac{\kappa_{\min}}{2}\|\mathcal{B}\mathbf{u}\|^2 \\ &= \tfrac{1}{2}\min(1,\ \tfrac{1}{\gamma^2},\ \tfrac{1}{\beta^2})\ \kappa_{\min}\ \|\mathbf{u}\|^2 + \tfrac{\kappa_{\min}}{2}\|\mathcal{B}\mathbf{u}\|^2 \\ &\geq \tfrac{1}{2}\min(1,\ \tfrac{1}{\gamma^2},\ \tfrac{1}{\beta^2})\ \kappa_{\min}\ \|\mathbf{u}\|_{\mathbf{R}}^2.\end{aligned}$$

Using this estimate in the above estimate for $\mathfrak{B}$, implies

$$\mathfrak{B}(\boldsymbol{\sigma},\mathbf{u},p;\boldsymbol{\tau},\mathbf{v},q) \geq \underline{c}(\Omega)\left(\|\boldsymbol{\sigma}\|^2+\frac{\varepsilon^2}{\kappa_{\max}}\|\mathcal{C}\boldsymbol{\sigma}\|^2+\kappa_{\min}\|\mathbf{u}\|^2_{\mathbf{R}}+\frac{1}{\kappa_{\max}}\|p\|^2_W\right)$$
$$= \underline{c}(\Omega)\ \left(\|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w}+\|\mathbf{u}\|^2_{\mathbf{R}_w}+\|p\|^2_{W_w}\right), \quad (6.15)$$

where $\underline{c}(\Omega)$ is a constant depending only on the Poincaré constants γ and β.

To conclude, we notice that

$$\left(\|\boldsymbol{\tau}\|^2_{\mathbf{Q}_w}+\|\mathbf{v}\|^2_{\mathbf{R}_w}+\|q\|^2_{W_w}\right) \leq [\overline{c}(\Omega)]^2\left(\|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w}+\|\mathbf{u}\|^2_{\mathbf{R}_w}+\|p\|^2_{W_w}\right), \quad (6.16)$$

where $\overline{c}(\Omega)$ is a constant depending only on the Poincaré constants γ and β. In fact, the following inequalities hold:

$$\frac{1}{2}\|\boldsymbol{\tau}\|^2_{\mathbf{Q}_w} \leq \|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w}+a_1^2\|\boldsymbol{\varphi}^\perp\|^2_{\mathbf{Q}_w} \leq \|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w}+\frac{\varepsilon^2}{\gamma^4}\|\mathbf{u}\|^2_{\mathbf{R}} \leq \|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w}+\frac{1}{\gamma^4}\|\mathbf{u}\|^2_{\mathbf{R}_w},$$

$$\frac{1}{3}\|\mathbf{v}\|^2_{\mathbf{R}_w} \leq \|\mathbf{u}\|^2_{\mathbf{R}_w}+a_2^2\kappa_{\min}\|\mathcal{C}\boldsymbol{\sigma}\|^2+a_3^2\kappa_{\min}\|\mathbf{w}^\perp\|^2_{\mathbf{R}}$$
$$\leq \|\mathbf{u}\|^2_{\mathbf{R}_w}+\frac{\kappa_{\min}\varepsilon^2}{\kappa^2_{\max}}\|\mathcal{C}\boldsymbol{\sigma}\|^2+c(\beta)\frac{\kappa_{\min}}{\kappa^2_{\max}}\|p\|^2_W$$
$$\leq \|\mathbf{u}\|^2_{\mathbf{R}_w}+\frac{\kappa_{\min}}{\kappa_{\max}}\|\boldsymbol{\sigma}\|^2_{\mathbf{Q}_w}+c(\beta)\frac{\kappa_{\min}}{\kappa_{\max}}\|p\|^2_{W_w},$$

$$\frac{1}{2}\|q\|^2_{W_w} \leq \|p\|^2_{W_w}+a_4^2\frac{1}{\kappa_{\max}}\|\mathcal{B}\mathbf{u}\|^2 \leq \|p\|^2_{W_w}+\frac{\kappa^2_{\min}}{\kappa_{\max}}\|\mathbf{u}\|^2_{\mathbf{R}} \leq \|p\|^2_{W_w}+\frac{\kappa_{\min}}{\kappa_{\max}}\|\mathbf{u}\|^2_{\mathbf{R}_w}.$$

Above, $c(\beta)=\frac{4\beta^2}{\min^2(4\beta^2,1)}$. Therefore the inf-sup condition for $\mathfrak{B}$ follows by taking $\alpha(\Omega)=\underline{c}(\Omega)\overline{c}(\Omega)$. □

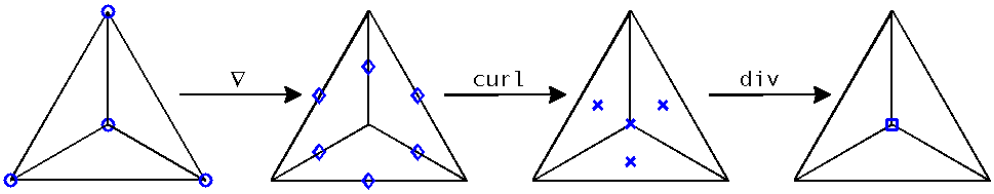


Figure 6.1: Lowest order Finite Element spaces for the discretization of the De Rham complex: P1 Lagrangian elements, first order Nédélec elements, zero order Raviart-Thomas elements, and discontinuous piecewise constant elements.

6.3 Discretization

In this section, we discuss the discretization of the complete de Rham complex, even though the Brinkman problem explicitly involves only the last three spaces of the sequence. In order to achieve stability of the discretized problem, the discrete spaces $S_h \subset H^1(\Omega)$, $\mathbf{Q}_h \subset \mathbf{Q}$, $\mathbf{R}_h \subset \mathbf{R}$, $W_h \subset W$ should preserve the de Rham complex structure of the continuous case (see [12, 13] for more details):

$$\begin{array}{ccccccc} H^1(\Omega)\backslash\mathbb{R} & \xrightarrow{\nabla} & \mathbf{Q} & \xrightarrow{\text{curl}} & \mathbf{R} & \xrightarrow{\text{div}} & W \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ S_h\backslash\mathbb{R} & \xrightarrow{\nabla} & \mathbf{Q}_h & \xrightarrow{\text{curl}} & \mathbf{R}_h & \xrightarrow{\text{div}} & W_h \end{array} .$$

A standard choice for numerical discretization of the Hodge Laplacian is the following. For a given integer $r \geq 0$, we let S_h be the continuous piecewise polynomial of degree at most $r+1$, $\mathbf{Q}_h$ the $(r+1)$-th order Nédélec finite elements [94], $\mathbf{R}_h$ the r-th order Raviart-Thomas finite elements [94, 108], and W_h the discontinuous piecewise polynomials finite element of degree at most r (see Fig. 6.1 for the case $r = 0$).

In fact, this choice of finite elements guarantees that the commutativity property

$$\text{curl}\, \Pi_h^{\mathbf{Q}} = \Pi_h^{\mathbf{R}} \,\text{curl}, \text{ and } \text{div}\, \Pi_h^{\mathbf{R}} = \Pi_h^{W} \,\text{div} \tag{6.17}$$

holds [10], [11]. Here, $\Pi_h^{\mathbf{V}} : \mathbf{V} \mapsto \mathbf{V}_h$ denotes the canonical interpolation operator from the continuous space $\mathbf{V}$ into the finite element counterpart $\mathbf{V}_h$, $\mathbf{V} := \mathbf{Q},\ \mathbf{R},\ W$.

Moreover, the above commutativity property together with the exactness of the continuous de Rham complex (for simply connected domain Ω with simply connected boundary) implies the discrete orthogonal decompositions

$$\mathbf{R}_h = \text{curl}\mathbf{Q}_h \oplus \nabla_h W_h, \text{ and } \mathbf{Q}_h = \nabla S_h \oplus \text{curl}_h \mathbf{R}_h,$$

where $\nabla_h : W_h \to \mathbf{R}_h$ is the adjoint of the map $-\text{div} : \mathbf{R}_h \to W_h$, and $\text{curl}_h : \mathbf{R}_h \to \mathbf{Q}_h$ is the adjoint map of $\text{curl} : \mathbf{Q}_h \to \mathbf{R}_h$.

An important result in [12, 13] guarantees that the discrete Poincaré inequalities

$$\begin{aligned} &\|\boldsymbol{\tau}_h^{\perp}\|_{\mathbf{Q}} \leq \gamma_h \|\text{curl } \boldsymbol{\tau}_h^{\perp}\|_{\mathbf{L}^2}, \quad \forall\ \boldsymbol{\tau}_h^{\perp} \in \text{curl}_h \mathbf{R}_h, \\ \text{and} \quad &\|\mathbf{v}_h^{\perp}\|_{\mathbf{R}} \leq \beta_h \|\text{div } \mathbf{v}_h^{\perp}\|_{L^2}, \quad \forall\ \mathbf{v}_h^{\perp} \in \nabla_h W_h \end{aligned} \tag{6.18}$$

hold with constants γ_h and β_h bounded independently of h. Since $\nabla_h W_h \not\subset \mathbf{Y}^{\perp}$ and $\text{curl}_h \mathbf{R}_h \not\subset \mathbf{X}^{\perp}$, the above results is not trivial, and it requires the construction of a bounded cochain projector operator π_h from the continuous to the discrete de Rham complex. Such bounded projectors can be derived by composing the canonical interpolation operators $\Pi_h^{\mathbf{V}}$ with commutative smoothing operators [31]. Then $\gamma_h \leq \gamma \|\pi_h\|$ and $\beta_h \leq \beta \|\pi_h\|$, where the norm of the cochain projector $\|\pi_h\|$ is uniformly bounded with respect to h. We refer to [12, 13] for the complete proof.

6.3.1 Analysis of the discrete problem

We now proceed with the analysis of the finite dimensional approximation of the variational problem, Problem 6.1.1. The Galerkin formulation of the problem reads

Problem 6.3.1 Find $(\boldsymbol{\sigma}_h, \mathbf{u}_h, p_h) \in \mathbf{Q}_h \times \mathbf{R}_h \times W_h$ such that

$$\begin{cases} m(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) & -c^*(\mathbf{u}_h, \boldsymbol{\tau}_h) & & = F(\boldsymbol{\tau}_h), & \forall\, \boldsymbol{\tau}_h \in \mathbf{Q}_h \\ -c(\boldsymbol{\sigma}_h, \mathbf{v}_h) & -a(\mathbf{u}_h, \mathbf{v}_h) - d(\mathbf{u}_h, \mathbf{v}_h) & +b^*(p_h, \mathbf{v}_h) & = G(\mathbf{v}_h), & \forall\, \mathbf{v}_h \in \mathbf{R}_h \\ & b(\mathbf{u}_h, q_h) & & = H(q_h), & \forall\, q_h \in W_h \end{cases}$$

Stability of Problem 6.3.1 is equivalent to the inf-sup condition for $\mathfrak{B}$ restricted to the finite element space [15], which is stated below.

Lemma 6.3.2 Under the hypothesis of Theorem 6.2.2, there exists a constant α_h, depending only on the constants γ_h and β_h in the discrete Poincaré inequalities (6.18), such that for any $(\boldsymbol{\sigma}_h, \mathbf{u}_h, p_h) \in \mathbf{Q}_h \times \mathbf{R}_h \times W_h$ there is a triplet $(\boldsymbol{\tau}_h, \mathbf{v}_h, q_h) \in \mathbf{Q}_h \times \mathbf{R}_h \times W_h$ for which the following estimate holds

$$\mathfrak{B}(\boldsymbol{\sigma}_h, \mathbf{u}_h, p_h; \boldsymbol{\tau}_h, \mathbf{v}_h, q_h) \geq \alpha_h(\|\boldsymbol{\sigma}_h\|^2_{\mathbf{Q}_w} + \|\mathbf{u}_h\|^2_{\mathbf{R}_w} + \|p_h\|^2_{W_w})^{\frac{1}{2}}(\|\boldsymbol{\tau}_h\|^2_{\mathbf{Q}_w} + \|\mathbf{v}_h\|^2_{\mathbf{R}_w} + \|q_h\|^2_{W_w})^{\frac{1}{2}}. \quad (6.19)$$

Proof. The proof in the discrete case closely follows the proof given before for the continuous case in Lemma 6.2.4. Given $\mathbf{u}_h \in \mathbf{R}_h$ there exists $\boldsymbol{\varphi}^\perp_h \in \text{curl}_h \mathbf{R}_h \subset \mathbf{Q}_h$ and $\mathbf{u}^\perp_h \in \nabla_h W_h \subset \mathbf{R}_h$ such that

$$\mathbf{u}_h = \text{curl}\, \boldsymbol{\varphi}^\perp_h + \mathbf{u}^\perp_h, \quad \|\boldsymbol{\varphi}^\perp_h\|_{\mathbf{Q}} \leq \gamma_h \|\text{curl}\, \boldsymbol{\varphi}^\perp\|_{\mathbf{R}}, \quad \|\mathbf{u}^\perp_h\|_{\mathbf{R}} \leq \beta_h \|\text{div}\, \mathbf{u}_h\|.$$

Similarly, given $p_h \in W_h$, there exists $\mathbf{w}^\perp_h \in \nabla_h W_h$, such that $p_h = \text{div}\ \mathbf{w}^\perp_h$ and $\|\mathbf{w}^\perp_h\|_{\mathbf{R}} \leq \beta_h \|p_h\|_W$. The result now follows by taking

$$\boldsymbol{\tau}_h = \boldsymbol{\sigma}_h - a_1 \boldsymbol{\varphi}^\perp_h \in \mathbf{Q}_h,\ \mathbf{v}_h = -\mathbf{u}_h - a_2 \text{curl}\, \boldsymbol{\sigma}_h + a_3 \mathbf{w}^\perp_h \in \mathbf{R}_h,\ q_h = p_h + a_4 \text{div}\, \mathbf{u}^\perp_h \in W_h$$

instead of (6.14), and by using the discrete Poincaré inequalities (6.18) instead of the continuous ones (6.4). □

The convergence of the discrete solution to the continuous one directly follows from the stability and consistency of the discrete problem.

Theorem 6.3.3 Let $(\boldsymbol{\sigma}, \mathbf{u}, p) \in \mathbf{Q} \times \mathbf{R} \times W$ be the solution of the continuous Problem 6.1.1 and let $(\boldsymbol{\sigma}_h, \mathbf{u}_h, p_h) \in \mathbf{Q}_h \times \mathbf{R}_h \times W_h$ be the solution of the discrete Problem 6.3.1. Then

$$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|^2_{\mathbf{Q}_w} + \|\mathbf{u} - \mathbf{u}_h\|^2_{\mathbf{R}_w} + \|p - p_h\|^2_{W_w} \leq$$
$$\left(1 + \frac{M}{\alpha_h}\frac{\kappa_{\max}}{\kappa_{\min}}\right)^2 \left(\inf_{\boldsymbol{\tau}_h \in \mathbf{Q}_h} \|\boldsymbol{\sigma} - \boldsymbol{\tau}_h\|^2_{\mathbf{Q}_w} + \inf_{\mathbf{v}_h \in \mathbf{R}_h} \|\mathbf{u} - \mathbf{v}_h\|^2_{\mathbf{R}_w} + \inf_{q_h \in W_h} \|p - q_h\|^2_{W_w} \right).$$

Proof. The proof of this theorem is a direct application of the Babuška Theorem in [14], and follows from the discrete inf-sup condition in Lemma 6.3.2 and the boundedness of the bilinear form $\mathfrak{B}$ in Lemma 6.2.3. □

Assuming certain smoothness of the continuous solutions and some regularity of the finite element mesh, the above theorem implies the standard error estimate

$$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|^2_{\mathbf{Q}_w} + \|\mathbf{u} - \mathbf{u}_h\|^2_{\mathbf{R}_w} + \|p - p_h\|^2_{W_w} = \mathcal{O}(h^{2r}),$$

for the choice of $(r+1) - th$ order Nédélec elements, r-th order Raviart-Thomas, and r-th order piecewise discontinuous elements. We refer to [4, 42, 48, 94] for a detailed analysis of the approximation properties of the Nédélec and Raviart-Thomas spaces.

In the following and in the numerical experiments, we restrict ourselves to the case $r = 0$, i.e., to the first order Nédélec space $\mathbf{Q}_h$, the lowest order Raviart-Thomas space $\mathbf{R}_h$, and the piecewise constant elements, W_h. As well known, such choice leads to a

first order discretization error.

6.4 Discretization error numerical results

The numerical results in this section aim to verify the accuracy of the mixed formulation. In particular, we analyze three different test cases: the first for a constant inverse permeability coefficient $k(\mathbf{x}) = k_0$, the second one for a variable but smooth coefficient, and the third one for discontinuous coefficient.

Concerning the choice of the finite element spaces, we will restrict ourselves to the case $r = 0$, i.e. first order Nédélec elements, lowest order Raviart-Thomas elements, and piecewise constant elements. It is worth pointing out that, in many practical applications, only discretization error of first order can be achieved due to non-smooth exact solutions and to discontinuities in the PDE coefficients. The initial meshes used in our simulation were generated with the unstructured mesh generator *netgen* [http://www.hpfem.jku.at/netgen/].

Concerning the finite element discretization of the Brinkman problem, we used the finite element library MFEM [http://code.google.com/p/mfem/], developed at Lawrence Livermore National Laboratory (LLNL). MFEM is a general, modular, parallel C++ library for finite element methods research and development. It supports a wide variety of finite element spaces in 2D and 3D, as well as many bilinear and linear forms defined on them. It includes classes for dealing with various types of triangular, quadrilateral, tetrahedral and hexahedral meshes and their global and local refinement.

6.4.1 Discretization error for constant coefficients.

We study the accuracy of the discretization as function of the ratio $\frac{k}{\varepsilon^2}$ in the case of constant inverse permeability k. Obviously if $\frac{k}{\varepsilon^2} = 0$ the Brinkman problem reduces to the Stokes problem, while if $\frac{k}{\varepsilon^2} \to \infty$ the Brinkman problem approaches the Darcy

limit.

The analytical solution is given by

$$\boldsymbol{\sigma}_{ex} = \varepsilon\pi \begin{bmatrix} \sin(\pi x)\cos(\pi y) - \cos(\pi z)\sin(\pi x) \\ \sin(\pi y)\cos(\pi z) - \cos(\pi x)\sin(\pi y) \\ \sin(\pi z)\cos(\pi x) - \cos(\pi y)\sin(\pi z) \end{bmatrix}, \qquad \mathbf{u}_{ex} = \begin{bmatrix} \sin(\pi y)\sin(\pi z) \\ \sin(\pi z)\sin(\pi x) \\ \sin(\pi x)\sin(\pi y) \end{bmatrix},$$

$$p_{ex} = 8.0\sin(\pi x)\sin(\pi y)\sin(\pi z).$$

The right hand side and the natural boundary conditions on $\partial\Omega$ are prescribed accordingly to the analytical solution. To avoid poorly scaled right hand sides, for a given ratio $\frac{k}{\varepsilon^2}$ we choose k and ε^2 such that $\max(k, \varepsilon^2) = 1$.

The domain $\Omega = [0,1]^3$ is discretized with an unstructured tetrahedral mesh with 474 elements. The original mesh is uniformly refined 5 times, and each element of the mesh is divided in 8 through a bisection algorithm for tetrahedrons. The total number of degrees of freedom ranges from around 2 thousand unknowns on the coarsest mesh up to 65 millions on the finest mesh.

In Figure 6.2 we show the relative discretization errors in the $H(\text{curl};\Omega)$, $H(\text{div};\Omega)$ and $L^2(\Omega)$ norms. The systems were solved by using preconditioned MINRES with a stopping criterion based on the relative residual norm. As expected from the theory, $\mathbf{u}_h$ linearly converges to $\mathbf{u}_{ex}$ in the $H(\text{div};\Omega)$ norm and p_h linearly converges to p_{ex} in $L^2(\Omega)$. Moreover the errors are independent of the ratio $\frac{k}{\varepsilon^2}$. The scaled vorticity $\boldsymbol{\sigma}_h$, instead, converges linearly to $\boldsymbol{\sigma}_{ex}$ for moderate values of $\frac{k}{\varepsilon^2}$, while it shows a degradation in the error behavior for higher values of such ratio.

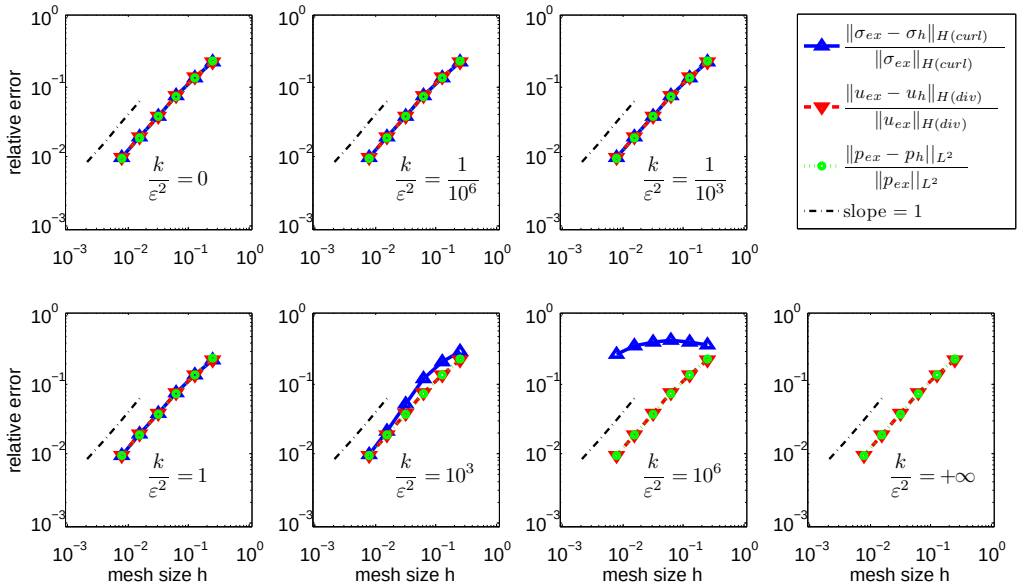


Figure 6.2: Relative discretization error in the case of constant coefficients for different choices of the ratio $\frac{k}{\varepsilon^2}$.

6.4.2 Discretization error for non-constant smooth coefficients.

Now we consider the case of non-constant coefficient $k(\mathbf{x})$. For $\Omega = [0,1]^3$ and $c \leq 1$ being a positive number, we take

$$k(\mathbf{x}) = \frac{1}{\sin(\pi y)\sin(\pi z) + c} \quad \forall\, \mathbf{x} = (x, y, z) \in \Omega. \tag{6.20}$$

The number c controls how large are the variations in the coefficient $k(\mathbf{x})$, since $k(\mathbf{x})$ ranges between $k_{\min} \sim 1$ and $k_{\max} \sim \frac{1}{c}$. We let the viscosity $\nu = \varepsilon^2 = 1$ and we choose the right hand side and the natural boundary conditions on $\partial\Omega$ be such that the analytical solution is given by

$$\boldsymbol{\sigma}_{exact} = \begin{bmatrix} 0 \\ \pi \sin(\pi y)\cos(\pi z) \\ -\pi \cos(\pi y)\sin(\pi x) \end{bmatrix}, \qquad \mathbf{u}_{exact} = \begin{bmatrix} \sin(\pi y)\sin(\pi z) \\ 0 \\ 0 \end{bmatrix}, \qquad p_{exact} = -x.$$

In this test we use the same initial mesh and refinement strategy as in the previous case of constant coefficients.

In Figure 6.3 we report the relative discretization error with respect to the analytical solution. In all cases, we observe a linear error decay in the $H(\text{curl};\Omega) - H(\text{div};\Omega) - L^2(\Omega)$ energy norms of vorticity, velocity, and pressure.

6.4.3 Discretization error for coefficients with jumps

For this test we consider the analytical solution of the so-called *circular preferential flow pathway* proposed in [74]. Such solution describes the steady flow of an incompressible fluid through a circular channel of radius R and length L in an infinite porous medium in response to a constant pressure gradient $\frac{\Delta p}{L}$ in the direction of the channel. Inside the preferential channel the inverse permeability is 0 (Stokes equations), outside is constant and equal to k. Using a cylindrical coordinate system, r stands for the distance

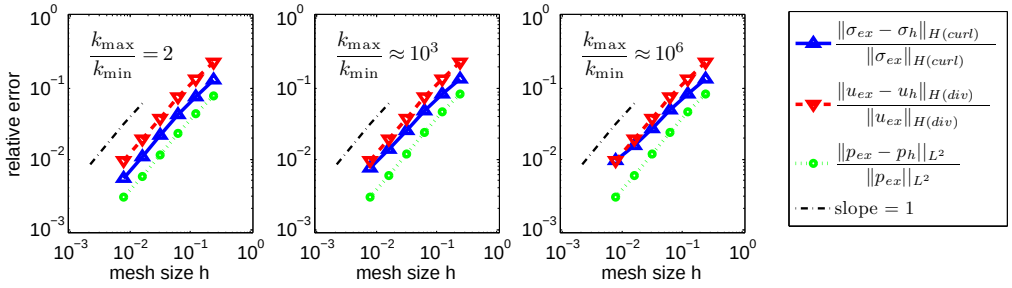


Figure 6.3: Relative discretization error in the case of non-constant coefficients for different choices of the ratio $\frac{k_{\max}}{k_{\min}}$.

from the centerline of the preferential channel, θ is the angle, and x is the coordinate along the centerline; $\hat{\mathbf{r}}$, $\hat{\boldsymbol{\theta}}$, $\hat{\mathbf{x}}$ are the unit vector in the radial, tangential, and centerline directions. The analytical solution of the flow is given by

$$\boldsymbol{\sigma} = \begin{cases} \frac{\Delta p}{4\mu L}(-2r)\hat{\boldsymbol{\theta}} & \text{if } r \leq R \\ \frac{\Delta p}{4\mu L}\left(2R\frac{K_0'(\sqrt{k}r)}{K_1(\sqrt{k}R)}\right)\hat{\boldsymbol{\theta}} & \text{if } r > R, \end{cases}$$

$$\mathbf{u} = \begin{cases} \frac{\Delta p}{4\mu L}\left(R^2 - r^2 + \frac{4}{k} + \frac{2R}{\sqrt{k}}\frac{K_0(\sqrt{k}R)}{K_1(\sqrt{k}R)}\right)\hat{\mathbf{x}} & \text{if } r \leq R \\ \frac{\Delta p}{4\mu L}\left(\frac{4}{k} + \frac{2R}{\sqrt{k}}\frac{K_0(\sqrt{k}r)}{K_1(\sqrt{k}R)}\right)\hat{\mathbf{x}} & \text{if } r > R, \end{cases}$$

$$p = \frac{\Delta p}{2L} - \frac{\Delta p}{L}x,$$

where K_0, K_1 are the modified Bessel functions of II type.

In Figure 6.4, we show the velocity and vorticity profiles in the radial direction. The velocity is continuous and differentiable with respect to r for each value of k, while the vorticity has a jump in the radial derivative at the interface between the preferential channel and the porous medium ($r = R$). Moreover for large value of k, we observe a boundary layer in the porous medium next to the interface with the preferential channel. In Figure 6.5, we show the three-dimensional solution computed on the finest

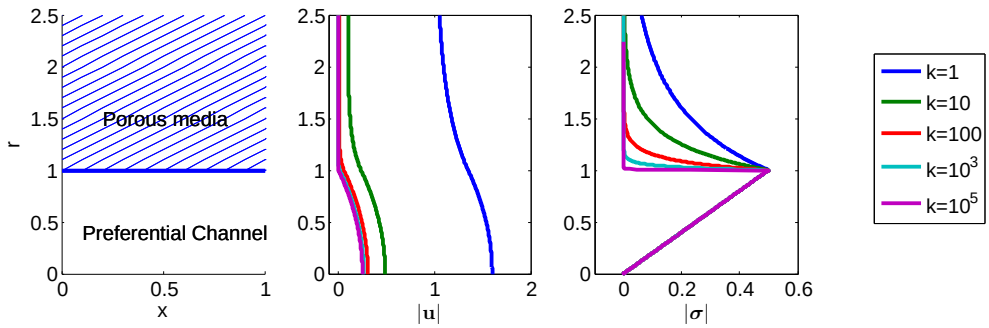


Figure 6.4: Velocity and vorticity profiles in the radial direction for different values of k.

mesh.

The external geometry for this test is a cylinder of radius 2 and length $L = 1$. An embedded cylinder of radius $R = 1$ represents the Preferential Channel. The Preferential Channel and the porous region are two non-overlapping domains in our setting. The total number of degrees of freedom ranges from three thousand on the coarser level up to 10 millions on the finest one.

In Figure 6.6, we show the behavior of the discretization error for different values of the inverse permeability k in the porous medium surrounding the preferential channel. We observe that the discretization errors in the $H(\text{div};\Omega)$-norm of the velocity and $L^2(\Omega)$-norm of the pressure are optimal with respect to the choice of finite elements ($\mathcal{O}(h)$). Similarly to what we already observed in the constant coefficient case, the $H(\text{curl};\Omega)$ norm of the discretization error for the vorticity field admits linear decay only for moderate values of the inverse permeability k; while for higher values ($k = 10^5$), we observe a sensible degradation in the converge rate.

6.5 Preconditioning

In this section, we introduce a block diagonal preconditioner for the discrete Galerkin problem, Problem 6.3.1. To this aim, let us first introduce the finite element vectors $\boldsymbol{\Sigma} \in \mathbb{R}^{\dim(\mathbf{Q}_h)}$, $\mathbf{U} \in \mathbb{R}^{\dim(\mathbf{R}_h)}$, $P \in \mathbb{R}^{\dim(W_h)}$, collecting the finite element degrees of

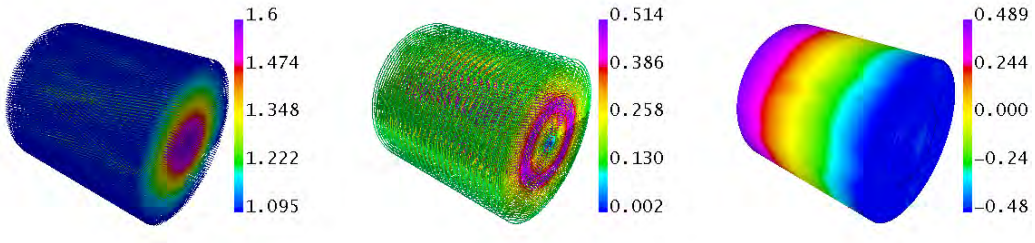


Figure 6.5: Numerical solution of the preferential channel on the finest grid ($k = 1$): velocity on the left, vorticity in the center, pressure on the right.

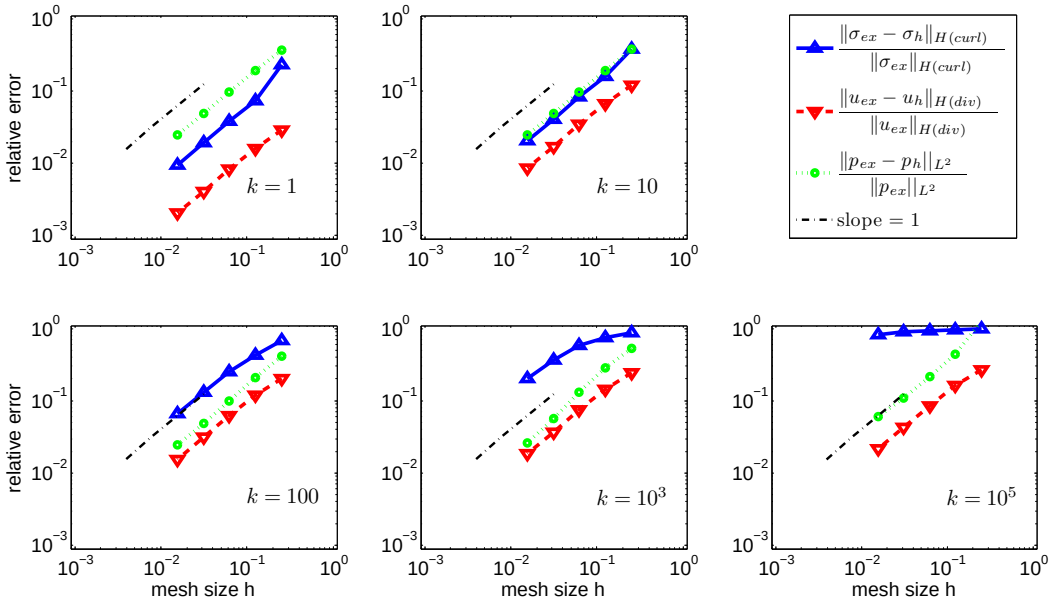


Figure 6.6: Relative discretization error for the preferential channel test case for different values of the inverse permeability k in the surrounding porous medium.

freedom $\boldsymbol{\sigma}_h^i$, $i = 1, \ldots, \dim(\mathbf{Q}_h)$, $\mathbf{u}_h^i$, $i = 1, \ldots, \dim(\mathbf{R}_h)$ and p_h^i, $i = 1, \ldots, \dim(W_h)$. Also let us denote with M, C, A, D, B the finite element matrices whose entries are given by

$$
\begin{aligned}
&M_{i,j} = m(\boldsymbol{\sigma}_h^j,\ \boldsymbol{\tau}_h^i) = (\boldsymbol{\sigma}_h^j,\ \boldsymbol{\tau}_h^i), && i,j = 1, \ldots, \dim(\mathbf{Q}_h)\\
&C_{i,j} = c(\boldsymbol{\sigma}_h^j,\ \mathbf{v}_h^i) = \varepsilon\ (\text{curl}\ \boldsymbol{\sigma}_h^j,\ \mathbf{v}_h^i), && i = 1, \ldots, \dim(\mathbf{R}_h),\ j = 1, \ldots, \dim(\mathbf{Q}_h)\\
&A_{i,j} = a(\mathbf{u}_h^j,\ \mathbf{v}_h^i) = \varepsilon^2(\text{div}\ \mathbf{u}_h^j,\ \text{div}\ \mathbf{v}_h^i), && i,j = 1, \ldots, \dim(\mathbf{R}_h)\\
&D_{i,j} = d(\mathbf{u}_h^j,\ \mathbf{v}_h^i) = (k(\mathbf{x})\ \mathbf{u}_h^j,\ \mathbf{v}_h^i), && i,j = 1, \ldots, \dim(\mathbf{R}_h)\\
&B_{i,j} = b(\mathbf{u}_h^j,\ q_h^i) = (\text{div}\ \mathbf{u}_h^j,\ q_h^i), && i = 1, \ldots, \dim(W_h),\ j = 1, \ldots, \dim(\mathbf{R}_h).
\end{aligned}
$$

Then, the discrete Galerkin problem, Problem 6.3.1, leads to the solution of a large sparse linear system

$$\mathcal{B}\mathbf{X} = \mathbf{B} \tag{6.21}$$

where the block matrix $\mathcal{B}$ and block vectors $\mathbf{X}$ and $\mathbf{B}$ read

$$\mathcal{B} = \begin{bmatrix} M & -C^T & 0 \\ -C & -A-D & B^T \\ 0 & B & 0 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \boldsymbol{\Sigma} \\ \mathbf{U} \\ P \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \\ H \end{bmatrix}. \tag{6.22}$$

The above linear system has the form of a symmetric saddle-point problem. In fact, if we reorder the unknowns as $(\boldsymbol{\Sigma},\ P,\ \mathbf{U})$, then $\mathcal{B}$ admits the form

$$\begin{bmatrix} M & 0 & -C^T \\ 0 & 0 & B \\ -C & B^T & -(A+D) \end{bmatrix}.$$

It is clear then that $\mathcal{B}$ has $\dim(\mathbf{Q}_h)+\dim(W_h)$ positive eigenvalues and $\dim(\mathbf{R}_h)$ negative eigenvalues. An effective iterative method to solve linear system with symmetric indefinite matrices is MINRES [97] employing a symmetric positive definite precondi-

tioner $\mathcal{P}$.

To derive the preconditioner, we follow the approach presented in [87] to precondition symmetric saddle point problems in a functional space setting. According to the authors, the mapping properties of the differential operators of the continuous problem suggest that block diagonal preconditioners are natural choice for saddle point problems. More specifically, given a stability estimate for the continuous problem in some functional spaces, the block diagonal matrix, in which the blocks represent the discretization of the inner products in those spaces, leads to a uniformly bounded (in terms of h) preconditioner for the discrete saddle point system of interest.

As before, we consider the positive numbers $w_Q = \frac{\varepsilon^2}{\kappa_{\max}}$, $w_R = \kappa_{\min}$, $w_W = \frac{1}{\kappa_{\max}}$, and introduce the symmetric positive definite variational forms

$$\begin{aligned} q(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) = &\quad (\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) + w_Q(\text{curl } \boldsymbol{\sigma}_h, \text{curl } \boldsymbol{\tau}_h), &\quad \boldsymbol{\sigma}_h, \boldsymbol{\tau}_h \in \mathbf{Q}_h \\ r(\mathbf{u}_h, \mathbf{v}_h) = &\quad w_R(\mathbf{u}_h, \mathbf{v}_h) + w_R(\text{div } \mathbf{u}_h, \text{div } \mathbf{v}_h), &\quad \mathbf{u}_h, \mathbf{v}_h \in \mathbf{R}_h \\ w(p_h, q_h) = &\quad w_W(p_h, q_h), &\quad p_h, q_h \in W_h. \end{aligned} \tag{6.23}$$

The above forms represent the *weighted* inner products in the spaces $\mathbf{Q}_w$, $\mathbf{R}_w$ and W_w defined by (6.9).

Therefore Lemma 6.3.2 and Lemma 6.2.3 imply that a mesh independent preconditioner for the saddle point problem (6.21) is given by

$$\mathcal{P} = \begin{bmatrix} P_Q & 0 & 0 \\ 0 & P_R & 0 \\ 0 & 0 & P_W \end{bmatrix}, \tag{6.24}$$

where P_Q, P_R, P_W are the matrix representation of the weighted inner products $q(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h)$, $r(\mathbf{u}_h, \mathbf{v}_h)$, and $w(p_h, q_h)$ in (6.23).

For completeness, we now proceed with the standard analysis of the spectral condition number $K(\mathcal{A})$ of the preconditioned saddle-point operator $\mathcal{A} = \mathcal{P}^{-\frac{1}{2}}\mathcal{B}\mathcal{P}^{-\frac{1}{2}}$. The

following result was proven in [122].

Theorem 6.5.1 The relative condition number of $\mathcal{B}$ with respect to the block–diagonal preconditioner $\mathcal{P}$ satisfies the estimate

$$K(\mathcal{P}^{-\frac{1}{2}}\mathcal{B}\mathcal{P}^{-\frac{1}{2}}) = \frac{\max|\lambda|}{\min|\lambda|} \leq \frac{M}{\alpha_h}\frac{\kappa_{\max}}{\kappa_{\min}}.$$

Proof. First of all, we remark that, if the vector $\hat{\mathbf{Y}} \in \mathbb{R}^N$ collects the degrees of freedom of the finite element functions $\boldsymbol{\tau}_h \in \mathbf{Q}_h$, $\mathbf{v}_h \in \mathbf{R}_h$, $q_h \in W_h$, then the $\mathcal{P}$ norm of $\hat{\mathbf{Y}}$ is equal to the norm of $(\boldsymbol{\tau}_h, \mathbf{v}_h, q_h)$ in $\mathbf{Q}_w \times \mathbf{R}_w \times W_w$, that is

$$\|\hat{\mathbf{Y}}\|_{\mathcal{P}} = \sqrt{\hat{\mathbf{Y}}^T\mathcal{P}\hat{\mathbf{Y}}} = \|(\boldsymbol{\tau}_h, \mathbf{v}_h, q_h)\|_{\mathbf{Q}_w\times\mathbf{R}_w\times W_w}.$$

Next, we recall the definition of the spectral condition number of an invertible (symmetric indefinite) operator $\mathcal{A}$,

$$K(\mathcal{A}) = \sup_{\mathbf{X}\in\mathbb{R}^N}\frac{\|\mathcal{A}\mathbf{X}\|_2}{\|\mathbf{X}\|_2}\sup_{\mathbf{Y}\in\mathbb{R}^N}\frac{\|\mathcal{A}^{-1}\mathbf{Y}\|_2}{\|\mathbf{Y}\|_2} = \frac{\max|\lambda|}{\min|\lambda|},\ \lambda\in\sigma(\mathcal{A}).$$

The aim now is to bound the two factors $\sup_{\mathbf{X}\in\mathbb{R}^N}\frac{\|\mathcal{A}\mathbf{X}\|_2}{\|\mathbf{X}\|_2}$ and $\sup_{\mathbf{Y}\in\mathbb{R}^N}\frac{\|\mathcal{A}^{-1}\mathbf{Y}\|_2}{\|\mathbf{Y}\|_2}$ by using the continuity and the inf-sup condition of the bilinear form $\mathfrak{B}$. For the first term, by letting $\hat{\mathbf{X}} = \mathcal{P}^{-\frac{1}{2}}\mathbf{X}$ and $\hat{\mathbf{Y}} = \mathcal{P}^{-\frac{1}{2}}\mathbf{Y}$ and by using (6.11), we have

$$\sup_{\mathbf{X}\in\mathbb{R}^N}\frac{\|\mathcal{A}\mathbf{X}\|_2}{\|\mathbf{X}\|_2} = \sup_{\mathbf{X}\in\mathbb{R}^N}\sup_{\mathbf{Y}\in\mathbb{R}^N}\frac{\mathbf{Y}^T\mathcal{A}\mathbf{X}}{\|\mathbf{Y}\|_2\,\|\mathbf{X}\|_2} = \sup_{\hat{\mathbf{X}}\in\mathbb{R}^N}\sup_{\hat{\mathbf{Y}}\in\mathbb{R}^N}\frac{\hat{\mathbf{Y}}^T\mathcal{B}\hat{\mathbf{X}}}{\|\hat{\mathbf{Y}}\|_{\mathcal{P}}\,\|\hat{\mathbf{X}}\|_{\mathcal{P}}} \leq M\frac{\kappa_{\max}}{\kappa_{\min}}.$$

Similarly, by using the discrete inf-sup condition (6.19), for the second term we have

$$\sup_{\mathbf{Y}\in\mathbb{R}^N} \frac{\|\mathcal{A}^{-1}\mathbf{Y}\|_2}{\|\mathbf{Y}\|_2} = \left(\inf_{\mathbf{X}\in\mathbb{R}^N} \frac{\|\mathcal{A}\mathbf{X}\|_2}{\|\mathbf{X}\|_2}\right)^{-1} = \left(\inf_{\mathbf{X}\in\mathbb{R}^N} \sup_{\mathbf{Y}\in\mathbb{R}^N} \frac{\mathbf{Y}^T\mathcal{A}\mathbf{X}}{\|\mathbf{Y}\|_2\,\|\mathbf{X}\|_2}\right)^{-1} = \left(\inf_{\hat{\mathbf{X}}\in\mathbb{R}^N} \sup_{\hat{\mathbf{Y}}\in\mathbb{R}^N} \frac{\hat{\mathbf{Y}}^T\mathcal{B}\hat{\mathbf{X}}}{\|\hat{\mathbf{Y}}\|_{\mathcal{P}}\,\|\hat{\mathbf{X}}\|_{\mathcal{P}}}\right)^{-1} \leq \frac{1}{\alpha_h}. \quad (6.25)$$

The thesis follows from the last two inequalities above. □

Theorem 6.5.1 implies that in the particular case of constant inverse permeability $k(\mathbf{x}) = k_0$, the condition number of the preconditioned saddle-point problem is independent of both the mesh size and k_0. The numerical experiments in the following section confirm this statement. However, in the general case of variable coefficient $k(\mathbf{x})$, the condition number increases proportionally to the ratio $\frac{\kappa_{\max}}{\kappa_{\min}}$.

Finally, we stress that the operator $\mathcal{A} = \mathcal{P}^{-\frac{1}{2}}\mathcal{B}\mathcal{P}^{-\frac{1}{2}}$ above is introduced only for the purpose of the analysis, and it does not explicitly appear in the implementation of the preconditioned MINRES algorithm; indeed only applications of $\mathcal{P}^{-1}$ to a vector are required. Moreover, to make it practical, we substitute $\mathcal{P}^{-1}$ with a spectrally equivalent operator $\hat{\mathcal{P}}^{-1}$ that is easier to apply (in fact, with optimal cost). In the numerical results section, we demonstrate that letting $\hat{\mathcal{P}}^{-1}$ be an auxiliary space algebraic multigrid preconditioner (AMG) can drastically reduce the computational effort.

6.5.1 Augmented Lagragian formulation

We now introduce an augmented lagrangian formulation of the Brinkman problem that showed better performances in our preliminary numerical tests. To this aim, we formally solve for the vorticity variable $\mathbf{\Sigma}$ the block equation $M\mathbf{\Sigma} - C^T\mathbf{U} = \mathbf{F}$ and, after some algebraic manipulation, we rewrite the block saddle point problem (6.21)

in the form

$$\begin{bmatrix} A+D+CM^{-1}C^T & -B^T \\ -B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} -\mathbf{G}-CM^{-1}\mathbf{F} \\ -H \end{bmatrix} \tag{6.26}$$

The role of the pressure variable P as lagrangian multiplier is made explicit by considering the equivalent constrained optimization formulation

$$\begin{aligned} &\min_{\mathbf{U}} \frac{1}{2}\mathbf{U}^T(A+D+CM^{-1}C^T)\mathbf{U} + \mathbf{U}^T(\mathbf{G}+CM^{-1}\mathbf{F}) \\ &\text{subject to:} \quad B\mathbf{U} = H. \end{aligned} \tag{6.27}$$

Letting $M_W \in \mathbb{R}^{N_p \times N_p}$ be the pressure mass matrix and $\gamma \in \mathbb{R}$ a positive number, we multiply the constraint $B\mathbf{U} = H$ by $\gamma B^T M_W$ and we sum the result to the first block equation of (6.28) to obtain

$$\begin{bmatrix} A+D+CM^{-1}C^T+\gamma B^T M_W^{-1}B & -B^T \\ -B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ P \end{bmatrix} = \begin{bmatrix} -\mathbf{G}-CM^{-1}\mathbf{F}+\gamma B^T M_W^{-1}H \\ -H \end{bmatrix} \tag{6.28}$$

It is worth noticing that, since M_W is a (block) diagonal matrix, the augmentation matrix $B^T M_W^{-1} B$ is sparse. In addition, thanks to the commutativity property (6.17), it is also possible to prove that this matrix has exactly the same sparsity pattern of the stiffness matrix A. As matter of fact, we can write $A = \varepsilon^2 B^T M_W^{-1} B$.

By reintroducing the vorticity $\mathbf{\Sigma}$ previously eliminated, we finally obtain the augmented lagrangian formulation

$$\mathcal{B}_\gamma \mathbf{X} = \mathbf{B}_\gamma,$$

where the augmented matrix $\mathcal{B}_\gamma$ and right hand side $\mathbf{B}_\gamma$ have the form

$$\mathcal{B}_\gamma = \begin{bmatrix} M & -C^T & 0 \\ -C & -A - D - \gamma B^T M_W^{-1} B & B^T \\ 0 & B & 0 \end{bmatrix}, \quad \mathbf{B}_\gamma = \begin{bmatrix} \mathbf{F} \\ \mathbf{G} - \gamma B^T M_W^{-1} H \\ H \end{bmatrix}. \tag{6.29}$$

The advantage of this augmented lagrangian formulation is that it, in practice, leads to the solution of better conditioned problems, provided that the augmentation parameter γ is not too large [17, 53, 84]. In the numerical result presented in the following we take γ of the order of the inverse permeability coefficient k.

Finally, to alleviate the dependence of the condition number with respect to the ratio $\frac{\kappa_{\max}}{\kappa_{\min}}$, we introduce, in Section 6.6.3, a special augmentation of the Brinkman problem and a modified version of the preconditioner, which gives optimal convergence rates for smooth coefficients $k(\mathbf{x})$.

6.6 Scalability results

The numerical results presented in the following aim to study the performance of the proposed preconditioner both in terms of number of iterations and wall time. Problems with increasing level of difficulty are considered: first the case when the inverse permeability coefficient $k(\mathbf{x})$ is constant in the domain, then when it varies smoothly, and finally we illustrate the difficulties with the block–diagonal preconditioning approach when the coefficient $k(\mathbf{x})$ admits large jumps.

In the following results, we use the stopping criterion proposed in [97] for MINRES. In particular, letting $r^k = b - Ax^k$ be the residual of the linear system $Ax = b$ at iteration k, convergence is achieved when

$$\frac{\|r^k\|_2}{\|A\|_2 \|x^k\|_2} \leq \tau \tag{6.30}$$

for some specified tolerance τ. Here $\|A\|_2$ is estimated by the MINRES algorithm as a by-product of the computations.

Two different versions of the block-diagonal preconditioner in (6.24) are compared. In the exact (*ideal*) version the blocks P_Q, P_R are solved up to a tight tolerance by using the preconditioned conjugate gradient method (auxiliary space AMG is used as preconditioner); while in the AMG (*practical*) version the inverse of blocks P_Q, P_R are approximated by one V-cycle for the auxiliary space AMG for $H(\text{curl})$ and $H(\text{div})$ problem respectively.

In practice the AMG version of the preconditioner out-performs the exact one, but we remark the theoretical importance of the latter, since it allows us to confirm the expected performance of the preconditioner, uniform with respect to the mesh.

In both versions of the preconditioner, the block P_W is inverted exactly. Indeed, due to the choice of discontinuous pressure finite elements P_W has a block diagonal structure, and it reduces to a diagonal matrix for piecewise constant elements.

6.6.1 Software and implementation details

Before discussing in detail the obtained numerical results, we describe briefly the software, compilers and hardware that we have used.

Concerning the finite element discretization, we used the parallel C++ library MFEM [http://code.google.com/p/mfem/], developed at Lawrence Livermore National Laboratory (LLNL). Parallelization in MFEM is based on MPI, and it leads to high scalability in the finite element assembly procedure. It supports several solvers from the HYPRE library (http://www.llnl.gov/CASC/hypre/). In particular, in our tests we used the auxiliary space algebraic multigrid solvers for $H(\text{curl})$ and $H(\text{div})$ [80, 81].

Our code was compiled with the Intel *mpiicc* and *mpiicpc* compilers version 11.1.046.

All the computations presented in this section were obtained on *hera*, a high per-

formance computer at LLNL. *Hera* has a total of 864 nodes connected by InfiniBand DDR (Mellanox). Each node has 16 AMD Quad-Core Opteron 2.3Ghz CPUs, and 32GB of memory. *Hera* is running CHAOS 4.4, a Linux kernel developed at LLNL, specific for high performance computing.

6.6.2 Constant coefficient weak scalability test

We study the performance of the proposed preconditioner in the case of constant coefficient $k(\mathbf{x}) = k_0$. In particular, we present results relative to the augmented formulation (6.29) with $\gamma = k_0$ and the block diagonal preconditioner $\mathcal{P}$, where the weights w_Q, w_R, w_W are given by

$$w_Q = \frac{\varepsilon^2}{k_0 + \varepsilon^2}, \quad w_R = k_0 + \varepsilon^2, \quad w_W = \frac{1}{k_0 + \varepsilon^2}. \tag{6.31}$$

For this test we use three different meshes in which the number of elements doubles from the previous to the next. By alternating between the three meshes and by using uniform refinements on each of them, we are able to build a sequence of discrete Brinkman problems whose size doubles each time. The sizes of the three meshes at the coarser level of refinement are given in Table 6.1. We use Metis [76] for the partitioning of the mesh among the processors.

Throughout the simulations we choose the number of processors in order to keep the number of unknowns per processor as constant as possible as we increase the total number of unknowns. We equally distribute the number of processes on each node of the parallel machine and we try to balance concurrency inside the node, on the one hand, with communications between nodes, on the other. In particular, we always use maximum of a quarter of the total node capacity, in order to minimize overhead due to concurrent access to memory. In particular, we start with only one node and we run 1, 2, and 4 parallel processes, then we move to 8 nodes and we run 8, 16, 64 parallel

processes (i.e. 1, 2, 4 processes per node, respectively), and so on.

In Table 6.2 we verify the uniform, with respect to the mesh, performance of the exact version of the preconditioner. The outer tolerance of MINRES is set to 10^{-10} (stopping criterion given by equation (6.30)) while the inner tolerance of the PCG (for inverting the blocks of $\mathcal{P}$) is set to 10^{-12} (two-norm of the relative residual less than the specified tolerance). The number of iterations is uniformly bounded for every value of k. Beside the case $k = 10^6$ in which, we observe a moderate increase of the iteration numbers (while decreasing h), the preconditioner shows a perfectly uniform behavior.

In Table 6.3, we show the number of iterations when using a single V-cycle of the auxiliary space AMG preconditioners for $H(\text{curl})$ and $H(\text{div})$, keeping all other parameters in the test the same as before. For fixed k, we observe a moderate increase of the number of iterations as the number of unknowns is growing. This is expected, given the particular choice of the parameters in the V-cycle which are made in order to minimize wall time rather than the number of iterations. We refer to [80] for a more detailed discussion about the choice of the multigrid parameters and their effects on number of iterations for the auxiliary space AMG preconditioner for $H(\text{curl})$ problems. Regarding the dependency of the iteration count with respect to the value of k, we notice that in the AMG version the number of iterations (even somewhat higher) is quite homogeneous with respect to k (for fixed mesh size).

Finally, in Table 6.4, we report the wall time to set-up the preconditioner (t_{setup}) and to solve iteratively the linear system with MINRES (t_{solve}). Timings are computed by using the *mpi* function `MPI_Wtime()`. The computation of the preconditioner consists in two phases. First we assemble the finite element matrices for the variational forms $q(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h)$, $r(\mathbf{u}_h, \mathbf{v}_h)$, and $w(p_h, q_h)$. Then, we compute the parameters (interpolation and coarse-level matrices) needed to build the auxiliary space solvers and to apply the V-cycle. We show only one column for t_{setup} since the preconditioner set-up is independent of the values of k. t_{setup} is usually negligible compared to t_{solve} (less than

	n_t	n_f	n_e
mesh 1	30336	62336	37940
mesh 2	57472	118304	72164
mesh 3	129920	266448	161602

Table 6.1: Number of elements n_t, faces n_f, and edges n_e on the coarser level of each unstructured mesh.

N	Number of MINRES iterations (Exact Preconditioner)						
	$k=0$	$k=10^{-6}$	$k=10^{-3}$	$k=1$	$k=10^{3}$	$k=10^{6}$	$\nu=0$
130612	16	16	16	18	30	17	10
247940	16	16	16	18	30	18	10
557970	16	16	16	18	30	20	10
1027944	16	16	16	18	30	20	10
1949480	16	16	16	18	30	22	10
4396980	16	16	16	18	30	23	10
8156368	16	16	16	18	30	23	10
15460560	16	16	16	18	30	25	10
34910120	16	16	16	18	30	27	10

Table 6.2: Number of MINRES iterations with the exact preconditioner for different values of k. N represents the total number of unknowns.

10% in all cases), and it scales well (even if not perfectly) with the number of processes. The fact that for $np = 16$ and $np = 128$ it is faster than in the cases $np = 8$ and $np = 64$, respectively, may suggest some load unbalance due to the partition of the meshes in the latter case. With respect to the solution times t_{solve}, we notice that for a fixed problem size they tend to decrease as we approach the Darcy limit since fewer iterations are required to converge. For fixed k the scaling of t_{solve} with respect to the number of processors is similar to the one reported in [80] up to 128 processes, but we observe a severe loss of scalability when we use 256 processes. Possible causes of this loss of performance could be not perfect load balancing and hardware configuration issues, which are beyond the scope of this thesis.

6.6.3 The case of non-constant smooth coefficients

Now we consider the case of non-constant coefficient $k(\mathbf{x})$ presented in Section 6.4.2.

For this test, we extend the augmentation technique discussed before to the case of

	Number of MINRES iterations (AMG Preconditioner)						
N	$k = 0$	$k = 10^{-6}$	$k = 10^{-3}$	$k = 1$	$k = 10^{3}$	$k = 10^{6}$	$\nu = 0$
130612	44	44	44	36	37	32	21
247940	48	48	48	40	39	34	23
557970	51	51	51	46	43	37	24
1027944	57	57	57	48	49	39	26
1949480	60	60	60	51	50	40	27
4396980	61	61	61	52	52	42	28
8156368	68	69	68	61	55	43	28
15460560	72	73	72	64	58	44	30
34910120	72	72	72	65	59	45	30

Table 6.3: Number of MINRES iterations with the AMG preconditioner for different values of k. N represents the total number of unknowns.

			t_{solve} (AMG Preconditioner)							t_{setup}
nn	np	N	$k = 0$	$k = 10^{-6}$	$k = 10^{-3}$	$k = 1$	$k = 10^{3}$	$k = 10^{6}$	$\nu = 0$	
1	1	130612	15.2	15.1	15.1	12.7	13.0	11.3	8.0	0.71
1	2	247940	17.7	17.7	17.7	15.1	14.7	13.0	9.5	0.96
1	4	557970	22.6	22.5	22.5	19.5	19.4	16.8	11.9	1.28
8	8	1027944	25.6	25.2	24.9	21.6	21.9	17.5	13.0	1.47
8	16	1949480	26.8	26.9	26.8	22.8	22.6	18.4	13.7	1.42
8	32	4396980	33.0	33.0	33.1	28.3	28.6	22.9	17.0	1.74
64	64	8156368	36.2	36.8	36.7	33.2	30.6	24.5	20.1	2.15
64	128	15460560	45.3	44.8	44.9	41.5	35.7	28.3	22.0	1.76
64	256	34910120	90.0	91.7	91.0	83.2	76.7	52.3	43.6	2.56

Table 6.4: Computational cost of the AMG preconditioner. nn is the number of nodes used, np is the number of processes, N the total number of degrees of freedom, t_{solve} and t_{setup} measures the time in seconds to solve the linear system and to assemble the preconditioner, respectively.

non constant coefficient. In particular, we solve the augmented saddle-point problem

$$\begin{cases} (\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) - \varepsilon\, (\mathbf{u}_h, \text{curl}\, \boldsymbol{\tau}_h) = F(\boldsymbol{\tau}_h), \quad \forall\, \boldsymbol{\tau}_h \in \mathbf{Q}_h \\ -\varepsilon\, (\text{curl}\, \boldsymbol{\sigma}_h, \mathbf{v}_h) - (k(\mathbf{x})\, \mathbf{u}_h, \mathbf{v}_h) - ((k(\mathbf{x}) + \varepsilon^2)\, \text{div}\, \mathbf{u}_h, \text{div}\, \mathbf{v}_h) + (p_h, \text{div}\, \mathbf{u}_h) = \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad G(\mathbf{v}_h) + H(k(\mathbf{x})\, \text{div}\, \mathbf{v}_h), \quad \forall\, \mathbf{v}_h \in \mathbf{R}_h \\ (\text{div}\, \mathbf{u}_h, q_h) = H(q_h), \quad \forall\, q_h \in W_h \end{cases} \tag{6.32}$$

preconditioned by a block-diagonal preconditioner with blocks corresponding to the following bilinear forms:

$$\begin{cases} (\boldsymbol{\sigma}_h,\ \boldsymbol{\tau}_h) + \varepsilon^2 \left(\dfrac{1}{k(\mathbf{x}) + \varepsilon^2}\ \text{curl}\, \boldsymbol{\sigma}_h,\ \text{curl}\, \boldsymbol{\tau}_h \right) & \boldsymbol{\sigma}_h, \boldsymbol{\tau}_h \in \mathbf{Q}_h \\ \big((k(\mathbf{x}) + \varepsilon^2)\ \mathbf{u}_h,\ \mathbf{v}_h\big) + \big((k(\mathbf{x}) + \varepsilon^2)\ \text{div}\, \mathbf{u}_h,\ \text{div}\, \mathbf{v}_h\big) & \mathbf{u}_h, \mathbf{v}_h \in \mathbf{R}_h \\ \left(\dfrac{1}{k(\mathbf{x}) + \varepsilon^2}\ p_h,\ q_h \right) & p_h, q_h \in W_h. \end{cases} \tag{6.33}$$

In Table 6.5, we report the number of MINRES iterations for the solutions of the Brinkman problem with variable coefficients (stopping criterion: norm of the relative residual less or equal to 10^{-10}). We show both the exact and inexact block-diagonal preconditioner. The exact preconditioner shows perfect uniformity with respect to the mesh behavior, and a moderate dependence on the ratio $\frac{k_{\max}}{k_{\min}}$. The inexact preconditioner consists of one V-cycle of the auxiliary space AMG applied to the weighted $H(\text{curl})$ and $H(\text{div})$ variational forms in (6.33). The number of iterations tends to grow as we refine the mesh, but it is more uniform respect to the ratio $\frac{k_{\max}}{k_{\min}}$.

6.6.4 The case of coefficients with discontinuities

For this test, we consider the analytical solution of the so-called *circular preferential flow pathway* proposed in [74] and described in Section 6.4.3.

In Table 6.6, we report the number of MINRES iteration required to achieve a reduction of 10^{-10} in the relative residual norm. Both, for the exact and AMG version

	Exact n_{it}			AMG n_{it}		
N	$\frac{k_{\max}}{k_{\min}} = 2$	$\frac{k_{\max}}{k_{\min}} \approx 10^3$	$\frac{k_{\max}}{k_{\min}} \approx 10^6$	$\frac{k_{\max}}{k_{\min}} = 2$	$\frac{k_{\max}}{k_{\min}} \approx 10^3$	$\frac{k_{\max}}{k_{\min}} \approx 10^6$
2.24K	19	30	30	26	35	36
16.9K	19	29	30	32	37	38
130K	19	29	32	46	45	48
1.03M	19	27	32	52	49	53
8.16M	19	27	32	63	61	61
65M	19	27	32	74	70	70

Table 6.5: Performances of the exact and AMG preconditioner for variable coefficient problem as a function of the ratio $\frac{k_{\max}}{k_{\min}}$. N represents the total number of unknowns and n_{it} the number of preconditioned MINRES iterations to achieve a relative reduction of the residual norm up to 10^{-10}.

of the preconditioner, the number of iterations highly depends on the size of the jump in the inverse permeability coefficient. In particular, for inverse permeability of the porous media up to 100, the qualitatively behavior of the preconditioner is the same as the constant and smoothly variable coefficient case: the number of iterations is uniform with respect to the mesh for the exact version, whereas for the AMG version it moderately increases as the mesh is refined. On the contrary, for higher values of the inverse permeability, the number of iterations tends to double at each mesh refinement for both versions of the preconditioner. Indeed, efficient preconditioning of the linear system with large jumps in the coefficient requires the introduction of a specialized coarse space correction and it is a subject of further investigation. For some progress in that direction, exploiting element-based algebraic multigrid (AMGe), we refer to [98] and [82].

	Exact preconditioner n_{it}				
N	$k=1$	$k=10$	$k=10^2$	$k=10^3$	$k=10^5$
2.9K	28	40	92	129	121
21K	27	38	94	189	221
164K	25	36	90	202	393
1.3M	23	34	83	199	640
10M	23	31	79	186	868

	AMG preconditioner n_{it}				
N	$k=1$	$k=10$	$k=10^2$	$k=10^3$	$k=10^5$
2.9K	30	43	93	133	135
21K	30	43	100	199	271
164K	31	44	105	235	526
1.3M	34	48	112	264	921
10M	40	57	128	293	> 999

Table 6.6: Number of MINRES iterations to achieve a reduction of 10^{-10} for the relative residual norm in the preferential channel test case.

7 Conclusion

In this thesis we have studied and implemented efficient methods for the solution of two important problems in computational fluid-dynamics, namely the Navier-Stokes and the Brinkman equations. It goes without saying that the set up of effective solvers for the study of incompressible flows in real-life situations requires computational power, together with a careful analysis and optimization of the numerical methods. Indeed, the effectiveness of the preconditioning and solving strategies as well as the scalability properties of the algorithms can not be inferred only from their performances on standard benchmarks and test cases. For example, in our experience in the modelling of blood flow, the complexity of the geometry (reconstructed from subject specific medical images) and therefore the flow conditions (based on subject specific flow and pressure measurements) determines challenges that can hardly be foreseen when dealing with simplified geometries or small scale problems.

The two main contributions of this thesis are (i) the development of new time adaptive algorithms for the Navier-Stokes equation, and (ii) the construction of robust solvers for the Brinkman problem.

In particular, **time adaptivity** is a viable strategy for reducing computational costs in the numerical solution of unsteady differential problems. In the case of incompressible fluid dynamics (Navier-Stokes equations) this could be of particular interest in some emerging applications like hemodynamics. In this thesis we have developed a strategy based on the (incremental) pressure corrected Yosida schemes. These schemes rely upon a sequence of guesses for the pressure field with an increasing order of time

accuracy. The difference between two guesses provides a natural error estimator for the selection of the time step.

The error estimator is thus computed with no additional cost. The results in 2D and 3D presented in this thesis show that the error is actually estimated with good reliability. Overall accuracy can be improved with the incremental formulation of the method. However, the splitting behind the pressure-correction approach, in particular with high order corrections, can affect significantly the stability of the time advancing. This eventually prevents the time adaptive method from selecting large time steps. A different - still hierarchical - approach consists of using low order pressure corrections as preconditioner. The comparison of the iterates of the preconditioned scheme with the converged solution provides in this case the error estimator. The stability of the time discretization method is retained, even if the computational costs are increased. We have successfully applied this strategy to biomedical relevant problems. As a future development, we plan to investigate a *hybrid adaptive strategy*. This means that we use the coupling between two pressure corrected solvers when accuracy forces small time steps, and the split/unsplit error estimator for larger time step.

In the last chapter of this thesis we also analyzed a **mixed formulation of the Brinkman problem** by introducing the (scaled) vorticity as an additional unknown. The well-posedness analysis of the mixed formulation was based on the Hodge decomposition, and the numerical stability of the method was guaranteed by an analogous results on the discrete level. The particular choice of Nédélec, Raviart-Thomas and piecewise discontinuous elements, in fact, reproduces the same embedding and mapping properties of the continuous spaces in the finite elements spaces. Differently from penalization methods for the Brinkman problem, our approach provides a conformal discretization by standard finite elements. Discretization errors in the $H(\mathrm{div};\Omega)$ norm of the velocity and in the $L^2(\Omega)$ norm of the pressure show uniform linear decay rate with respect to the inverse permeability coefficient $k(\mathbf{x})$. Only the (scaled) vorticity is

approximated with less accuracy as we approach the Darcy limit.

Despite the increased number of unknowns, the algebraic linear system obtained after the finite element discretization can be efficiently solved with Krylov iterative methods using block diagonal AMG preconditioning. In particular, we used the auxiliary space algebraic multigrid preconditioners for $H(\text{curl})$ and $H(\text{div})$ for the vorticity and velocity block respectively, and diagonal scaling for the pressure block. In the case of constant and smooth PDE coefficients, the proposed preconditioner exhibits good scalable properties and it is robust with respect to a wide range of values of the inverse permeability coefficient $k(\mathbf{x})$. Future developments of interest include upscaling techniques and construction of a coarse hierarchy that respects the de Rham complex with good approximation properties, to handle the non-constant coefficient case with both upscaling and solver (multigrid) purpose. The latter is feasible based on appropriate element-based algebraic multigrid (AMGe) technique. For some progress in that direction, we refer to [98], [82]. Finally, in the presence of stable hierarchy of coarse spaces, for stochastic coefficients, multilevel Monte-Carlo process can be employed in order to perform numerical simulations for underground flow of practical interest.

To summarize, the original contributions of this thesis work are the following.

1. We analysed the convergence properties of the incremental version of the high order pressure corrected schemes proposed in [56] for the solution of the incompressible Navier-Stokes equations, proving that the incremental pressure methods improve the accuracy of the splitting.

2. Based on the above splitting error analysis we proposed reliable *a posteriori estimator* for time adaptivity. Such estimator is a by-product of the pressure corrected Yosida splitting, and do not represent any additional cost.

3. We analysed the use of inexact quadrature rule to produce diagonal (*lumped*)

finite element mass matrices for *inf-sup* compatible finite elements. We used the Strang lemma to estimate the error introduced by such lumping.

4. We proposed a novel mixed formulation for the Brinkman problem. We used the tools developed in the Finite Element Exterior Calculus framework [12, 13] to prove the numerical stability of the formulation.

5. We constructed a block-diagonal algebraic multigrid preconditioner for the above formulation of the Brinkman problem that is optimal in the case of constant coefficient.

In addition, we contributed to the finite element libraries LifeV and MFEM in the following ways.

1. We provided an efficient implementation of the high order pressure corrected splittings and preconditioners for the serial and parallel version of LifeV.

2. We implemented time adaptive algorithms for the Navier-Stokes equations that are able to sensibly reduce the computational costs of hemodynamics simulation.

3. We implemented in LifeV a general parallel (Trilinos based) framework for the preconditioning of block linear systems. The framework has been widely tested in this thesis for the solution of the incompressible Navier-Stokes equations, however the same framework is now applied to the solution of defective boundary condition problems (see [69], [127] for an introduction), fluid structure interactions, heterogeneous domain decomposition.

4. We implemented in MFEM mixed discretization and preconditioners for the Brinkman problem. To this aim we also implemented a block linear algebra framework similar to the one in LifeV.

(P)ALADINS, in particular, is the (Parallel) ALgebraic ADaptive Incompressible Navier-Stokes Solver, written in C++, that implements the algebraic splitting methods and

preconditioners discussed in this thesis. ALADINS is based on the serial version of the C++ finite element LifeV, and it shows satisfactory performance in the solution of small/medium scale problems. ALADINS exploits the multifrontal QR factorization package SuiteSparseQR to drastically increase the effectiveness of the pressure correction step. As a matter of fact, in the case of fixed domain simulations, the initial computational cost to perform the factorization is amortized over the rest of the simulation. In order to solve large scale problems on high performance clusters, we also developed a parallel adaptive algebraic Navier-Stokes solver (PALADINS) based on the parallel version of LifeV. PALADINS extends the benefits of time adaptivity to large scale problem and shows good scalability properties. The algorithms and preconditioners in PALADINS represent also the numerical solver for application codes written for collaborative projects between the research group of the thesis advisor and School of Medicine at Emory (Dr. R. W. Taylor, Dr. H. Samady, Dr. F. Tong) and the Department of Biomedical Engineering at Ga Tech & Emory (Dr. D. Giddens, Dr. A. P. Yoganathan).

References

[1] A. A. HENDERSON. **ParaView Guide, A Parallel Visualization Application**. `http://www.paraview.org`, 2007. 105, 126

[2] G. ALLAIRE. **Homogenization of the Navier-Stokes equations in open sets perforated with tiny holes I. Abstract framework, a volume distribution of holes**. *Archive for Rational Mechanics and Analysis*, **113**:209–259, 1991. 6, 139

[3] G. ALLAIRE. **Homogenization of the Navier-Stokes equations in open sets perforated with tiny holes II: Non-critical sizes of the holes for a volume distribution and a surface distribution of holes**. *Archive for Rational Mechanics and Analysis*, **113**:261–298, 1991. 6, 139

[4] A. ALONSO AND A. VALLI. **An optimal domain decomposition preconditioner for low-frequency time-harmonic Maxwell equations**. *Mathematics of Computation*, **68**:607–631, 1999. 156

[5] AMD. **AMD Core Math Library (ACML)**. `http://www.amd.com/acml`, 2012. 112

[6] P. R. AMESTOY, T. A. DAVIS, AND I. S. DUFF. **Algorithm 837: AMD, an Approximate Minimum Degree Ordering Algorithm**. *ACM Trans. Math. Softw.*, **30**(3):381–388, 2004. 110

[7] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, S. OSTROUCHOV, AND D. SORENSEN. *LAPACK's User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. 109, 113

[8] L. ANTIGA, M. PICCINELLI, L. BOTTI, B. ENE-IORDACHE, A. REMUZZI, AND D. STEINMAN. **An Image-Based Modeling Framework for Patient-Specific**

Computational Hemodynamics. *Medical and Biological Engineering and Computing*, **46**:1097–1112, 2008. 95, 97

[9] D. N. ARNOLD, R. S. FALK, AND J. GOPALAKRISHNAN. **Mixed finite element approximation of the vector Laplacian with Dirichlet boundary conditions**. arXiv 1109.3668, submitted to Mathematical Models & Methods in Applied Sciences, 2011. 139, 142

[10] D. N. ARNOLD, R. S. FALK, AND R. WINTHER. **Preconditioning in H(div) and applications**. *Mathematics of Computations*, **66**:957 – 984, 1997. 10, 143, 154

[11] D. N. ARNOLD, R. S. FALK, AND R. WINTHER. **Multigrid in H(div) and H(curl)**. *Numerische Mathematik*, **85**:197 – 217, 2000. 10, 143, 154

[12] D. N. ARNOLD, R. S. FALK, AND R. WINTHER. **Finite element exterior calculus, homological techniques, and applications**. *Acta Numerica*, **15**:1–155, 2006. 9, 139, 153, 154, 182

[13] D. N. ARNOLD, R. S. FALK, AND R. WINTHER. **Finite element exterior calculus: from Hodge theory to numerical stability**. *Bull. Amer. Math. Soc. (N.S.)*, **47**:281–354, 2010. 9, 139, 153, 154, 182

[14] I. BABUŠKA. **Error-bounds for finite element method**. *Numer. Math.*, **16**:322–333, 1970/1971. 146, 156

[15] I. BABUŠKA AND A. K. AZIZ. **Survey lectures on the mathematical foundations of finite element method**. In *The Mathematical Foundations of the FEM with Application to PDE*. Academic Press, 1972. 155

[16] M. BENZI, G. H. GOLUB, AND J. LIESEN. **Numerical Solution of Saddle Point Problems**. *Acta Numerica*, **14**:1–137, 2005. 1

[17] M. BENZI AND M. A. OLSHANSKII. **An Augmented Lagrangian-Based Approach to the Oseen Problem**. *SIAM Journal on Scientific Computing*, **28**(6):2095–2113, 2006. 169

[18] M. BENZI, M. A. OLSHANSKII, AND Z. WANG. **Modified Augmented Lagrangian Preconditioners for the Incompressible Navier-Stokes Equations**. *Int. J. Num. Meth Fluids*, **66**:486–508, 2011. 120

[19] M. BENZI AND Z. WANG. **A Parallel Implementation of the Modified Augmented Lagrangian Preconditioner for the Incompressible Navier-Stokes Equations**. *Numerical Algorithms*, 2012. 120

[20] J. BERGH AND J. LÖFSTRÖM. *Interpolation spaces: an introduction*. Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1976. 147

[21] D. BOFFI, F. BREZZI, AND M. FORTIN. **Finite Elements for the Stokes Problem**. In D. BOFFI AND L. GASTALDI, editors, *Mixed Finite Elements, Compatibility Conditions, and Applications*, number 1939 in Lecture Notes in Mathematics, pages 45–100. Springer, Berlin, 2006. 24

[22] J. BRAMBLE AND J. PASCIAK. **Iterative techniques for the time dependent Stokes problems**. *Computers Math. Applic.*, **33**:13 – 30, 1997. 44

[23] F BREZZI AND M FORTIN. *Mixed and Hybrid finite element method*. Springer-Verlag, Berlin, 1991. 1, 24

[24] E. BURMAN AND P. HANSBO. **Stabilized Crouzeix-Raviart element for the Darcy-Stokes problem**. *Numerical Methods for Partial Differential Equations*, **21**(5):986–997, 2005. 139

[25] E. BURMAN AND P. HANSBO. **A unified stabilized method for Stokes' and Darcy's equations**. *Journal of Computational and Applied Mathematics*, **198**(1):35 – 51, 2007. 139

[26] J. CAHOUET AND J. P. CHABARD. **Some Fast 3D Finite Element Solvers for the Generalized Stokes Problem**. *Int. J. for Numer. Meth. Fluids*, **8**, 1988. 31, 65, 70, 71

[27] X. C. CAI AND M. SARKIS. **A Restricted Additive Schwarz Preconditioner for General Sparse Linear Systems**. *SIAM J. Sci. Comput.*, **21**(2):792–797, 1997. 107

[28] M. CALVO, T. GRANDE, AND R. D. GRIGORIEFF. **On the Zero Stability of the Variable Order Variable Stepsize BDF-Formulas**. *Numerische Mathematik*, **57**(1):39–50, 1990. 28, 84

[29] M. CALVO AND R. D. GRIGORIEFF. **Time Discretisation of Parabolic Problems with the Variable 3-Step BDF**. *BIT Numerical Mathematics*, **42**(4):689–701, 2002. 28, 84

[30] A. J. CHORIN. **Numerical Solutions of the Navier-Stokes equations**. *Math. Comp.*, **22**:745–762, 1968. 43

[31] S. H. CHRISTIANSEN AND R. WINTHER. **Smoothed projections in finite element exterior calculus**. *Mathematics of Computation*, **77**:813–829, June 2008. 154

[32] P. G. CIARLET. *The Finite Element Method for Elliptic Problems*. North-Holland, 1978. 34

[33] B. COCKBURN AND S. W. SHU. **The local discontinuous Galerkin method for time-dependent convection-diffusion systems**. *SIAM J. Num. Anal.*, **35**:2440–2463, 1998. 6

[34] G. COHEN, P. JOLY, J. E. ROBERTS, AND N. TORDJMAN. **Higher Order Triangular Finite Elements with Mass Lumping for the Wave Equation**. *SIAM J. Numer. Anal.*, **38**(6):2047–2078, 2001. 33, 34

[35] V. COMINCIOLI. *Analisi Numerica Metodi Modelli Applicazioni (Numerical Analysis: Methods, Models, and Applications)*. MCGRAW-HILL, 1990. 26, 84

[36] W. COUZY. *Spectral Element Discretization of the Unsteady Navier-Stokes equations and its Iterative Solution on Parallel Computers*. PhD thesis, EPFL, Lausanne, 1995. 44

[37] T. A. DAVIS. **Algorithm 8xx: SuiteSparseQR, a Multifrontal Multithreaded Sparse QR Factorization Package**. *ACM Trans. Math. Softw.*, 2008. 103, 109, 111, 113

[38] T. A. DAVIS. **Multifrontal Multithreaded Rank-Revealing Sparse QR Factorization**. *ACM Trans. Math. Softw.*, 2008. 103, 109, 113, 117

[39] T. A. DAVIS. **User's Guide for SuiteSparseQR, a Multifrontal Multithreaded Sparse QR Factorization Package**. *ACM Trans. Math. Software*, 2008. 103

[40] T. A. DAVIS, J. R. GILBERT, S. I. LARIMORE, AND E. G. NG. **Algorithm 836: COLAMD, a Column Approximate Minimum Degree Ordering Algorithm**. *ACM Trans. Math. Softw.*, **30**(3):377–380, 2004. 110

[41] J. J. DONGARRA, J. D. CRUZ, S. HAMMERLING, AND I. S. DUFF. **Algorithm 679: A Set of Level 3 Basic Linear Algebra Subprograms: Model Implementation and Test Programs**. *ACM Trans. Math. Softw.*, **16**(1):18–28, 1990. 109, 113

[42] J. J. DOUGLAS AND J. E. ROBERTS. **Global estimates for mixed methods for second order elliptic equations**. *Math. Comp*, **44**:39–52, 1985. 156

[43] H. ELMAN AND D. SYLVESTER. **Fast Nonsymmetric Iterations and Preconditioning for Navier-Stokes Equations**. *SIAM J. Sci. Comput.*, **20**:33–46, 1996. 68

[44] H. C. ELMAN, V. E. HOWLE, J. SHADID, R. SHUTTLEWORTH, AND R. TUMINARO. **Block Preconditioners based on Approximate Commutators**. *SIAM J. Sci. Comput.*, **27**:1651–1668, 2006. 65, 70, 72

[45] 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, USA, 2005. 1, 6, 24, 25, 44, 65, 66, 67

[46] A. ERN AND J. L. GUERMOND. *Theory and Practice of Finite Elements*. Springer Verlag, 2004. 35

[47] C.R. ETHIER AND D.A. STEINMAN. **Exact fully 3D Navier–Stokes solutions for benchmarking**. *International Journal for Numerical Methods in Fluids*, **19**(5):369–375, 1994. xvii, 126, 127

[48] R. S. FALK AND J. E. OSBORN. **Error estimates for mixed methods**. In *MR 592753 (82j:65076)*, **14** of *RAIRO Anal. Numér.*, pages 249–277, 1980. 156

[49] Y. T. FENG AND D. PERIĆ. **A Time-Adaptive Space-Time Finite Element Method for Incompressible Lagrangian Flows with Free Surfaces: Computational Issues.** *Comput. Methods Appl. Mech. Eng.* , **190**(5-7):499–518, 2000. 77

[50] G. J. FIX. *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, chapter Effect of Quadrature Errors in the Finite Element Approximation of Steady State, Eigenvalue and Parabolic Problems, pages 525–556. Academic Press, 1972. 34

[51] L. FORMAGGIA, J. F. GERBEAU, AND C. PRUD'HOMME. *LifeV Developer Manual*. EPFL, INRIA, MOX, 2008. 103, 104

[52] L. FORMAGGIA, F. SALERI, AND A. VENEZIANI. *Solving Numerical PDEs: Problems, Applications, Exercises*. Springer, Milan, 2012. 17, 32

[53] M. FORTIN AND R. GLOWINSKI. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Studies in Mathematics and its Applications. North-Holland, Amsterdam, 1983. Transl.of : Méthodes de Lagrangian augmentées. Paris : Dunod, 1982. 169

[54] A. GAUTHIER, F. SALERI, AND A. VENEZIANI. **A Fast Preconditioner for the Incompressible Navier Stokes Equations**. *Computing and Visualization in Science*, **6**(2):105–112, 2004. 47, 61, 69, 70, 72, 88

[55] P. GERVASIO. **Convergence Analysis of High Order Algebraic Fractional Step Schemes for Time-Dependent Stokes Equations**. *SIAM Journal on Numerical Analysis*, **46**(4):1682–1703, 2008. 9, 47, 51, 52, 56, 57, 60, 61, 62, 63, 77

[56] P. GERVASIO, F. SALERI, AND A. VENEZIANI. **Algebraic Fractional-Step Schemes with Spectral Methods for the Incompressible Navier-Stokes Equations**. *J. Comput. Phys.*, **214**(1):347–365, 2006. 47, 51, 181

[57] C. GEUZAINE AND J.F. REMACLE. **Gmsh: a three-dimensional finite element mesh generator with built-in pre-and post-processing facilities**. *Int J Numer Meth Engng*, **79**:1309–1331, 2009. 104, 124

[58] R. GLOWINSKI. **Splitting Methods for the Numerical Solution of the Incompressible Navier-Stokes Equations**, 1986. 43

[59] K. GOTO AND R. VAN DE GEIJN. **High-Performance Implementation of the Level-3 BLAS**. *ACM Trans. Math. Softw.*, **35**(1):1–14, 2008. 109, 113

[60] B. GOTTERMEIER AND J. LANG. **Adaptive Two-Step Peer Methods for Incompressible Navier-Stokes Equations**. In G. KREISS, P. LÖTSTEDT, A. MÅLQVIST, AND M. NEYTCHEVA, editors, *Numerical Mathematics and Advanced Applications 2009*, pages 387–395. Springer Berlin Heidelberg, 2010. 1, 77

[61] P. M. GRESHO, D. F. GRIFFITHS, AND D. J. SILVESTER. **Adaptive Time-Stepping for Incompressible Flow Part I: Scalar Advection-Diffusion**. *SIAM J. Sci. Comput.*, **30**:2018–2054, May 2008. 1, 77

[62] P. M. GRESHO AND R. L. SANI. *Incompressible Flow and the Finite Element Method. Vol. 1: Advection-Diffusion. Vol. 2: Isothermal Laminar Flow.* Chichester: Wiley, 2000. 50

[63] L. G. GUERMOND AND L. QUARTAPELLE. **On Stability and Convergence of Projection Methods Based on Pressure Poisson Equation**. *Int. J. Numer. Meth. Fluids*, **26**:1039–1053, 1998. 50

[64] J. GUZMÁN AND M. NEILAN. **A family of nonconforming elements for the Brinkman problem**. *IMA Journal of Numerical Analysis*, 2012. 7

[65] 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**. *J. Biomech. Engin.*, **133**(4):041002, 2011. 3

[66] MARTIN OFSTAD HENRIKSEN AND JENS HOLMEN. **Algebraic Splitting for Incompressible Navier-Stokes Equations**. *J. Comput. Phys.*, **175**(2):438–453, 2002. 46, 50, 51

[67] P. HERNICI. *Discrete Variable Methods in Ordinary Differential Equations*. John Wiley & Sons, New York, 1964. 81

[68] M. A. HEROUX, R. A. BARTLETT, V. E. HOWLE, R. J. HOEKSTRA, J. J. HU, G. K. TAMARA, R. B. LEHOUCQ, K. R. LONG, R. P. PAWLOWSKI, E. T. PHIPPS, A. G. SALINGER, H. K. THORNQUIST, R. S. TUMINARO, J. M. WILLENBRING,

A. WILLIAMS, AND K. S. STANLEY. **An overview of the Trilinos project**. *ACM Trans. Math. Softw.*, **31**(3):397–423, 2005. 103, 104, 105, 119, 126

[69] J. G. HEYWOOD, R. RANNACHER, AND S. TUREK. **Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations**. *Int. J. Numer. Meth. Fluids*, **22**:325–352, 1996. 182

[70] R. HIPTMAIR AND J. XU. **Nodal auxiliary space preconditioning in H(curl) and H(div) spaces**. *SIAM Journal on Numerical Analysis*, **45**(6):2483–2509, 2007. 10

[71] Intel. *Intel(R) Threading Building Blocks*, 2008. 109, 112

[72] INTEL. **Intel Math Kernel Library (Intel MKL)**. `http://software.intel.com/en-us/articles/intel-mkl`, 2012. 112, 129

[73] J. W. J. DANIEL, W. B. GRAGG, L. KAUFMAN, AND G. W. STEWART. **Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization**. *Math. Comput.*, **30**:772–995, 1976. 106

[74] A. A. JENNINGS AND R. PISIPATI. **The impact of Brinkman's extension of Darcy's law in the neighborhood of a circular preferential flow pathway**. *Environmental Modelling and Software with Environment Data News*, **14**(5):427–435, 1999. 160, 175

[75] G. E. KARNIADAKIS, M. ISRAELI, AND S. A. ORSZAG. **Higher Order Splitting methods for the Incompressible Navier-Stokes Equations**. *J. Comput. Phys.*, **97**:414 – 443, 1991. 30, 44

[76] G. KARYPIS AND V. KUMAR. **A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs**. *SIAM J. Sci. Comput.*, **20**:359–392, 1998. 110, 171

[77] G. KARYPIS AND V. KUMAR. **MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0**. `http://www.cs.umn.edu/~metis`, 2009. 105, 124

[78] D. A. KAY, P. M. GRESHO, D. F. GRIFFITHS, AND D. J. SILVESTER. **Adaptive Time-Stepping for Incompressible Flow Part II: Navier-Stokes Equations**. *SIAM J. Sci. Comput.*, **32**:111–128, 2010. 77

[79] G. M. KOBELKOW AND M. OLSHANSKII. **Effective Preconditioning of Uzawa Type Schemes for a Generalized Stokes Problem**. *Num. Mat.*, pages 443–470, 2000. 72

[80] T. KOLEV AND P. S. VASSILEVSKI. **Parallel Auxiliary Space AMG for H(curl) Problems**. *J. of Computational Mathematics*, **27**, 2009. 10, 140, 170, 172, 173

[81] T. KOLEV AND P. S. VASSILEVSKI. **Parallel Auxiliary Space AMG Solver for H(div) Problems**. Technical Report LLNL-JRNL-520391, December 15, 2011. 10, 140, 170

[82] I. V. LASHUK AND P. S. VASSILEVSKI. **Element Agglomeration Coarse Raviart-Thomas Spaces With Improved Approximation Properties**. *Numerical Linear Algebra with Applications*, **19**(2):414–426, 2012. 140, 176, 181

[83] W. J. LAYTON AND L. G. REBHOLZ. *Approximate Deconvolution Models of Turbulence: Analysis, Phenomenology and Numerical Analysis*, **2042** of *Lecture Notes in Mathematics*. Springer, 2012. 3

[84] P. LE TALLEC AND T. SASSI. **Domain decomposition with nonmatching grids: augmented Lagrangian approach**. *Math. Comp.*, **64**(212):1367–1396, 1995. 169

[85] J. G. LIU AND C. W. SHU. **A high order discontinuous Galerkin method for 2D incompressible flows**. *J. Comput. Phys.*, **160**:577–596, 2000. 6

[86] G. I. MARCHUK. **Splitting and Alternating Direction Methods**. In P. G. CIARLET AND J. L. LIONS, editors, *Handbook of Numerical Analysis, vol. 1*, pages 197 – 462. Elsevier Science Publisher B. V., North Holland, 1990. 44

[87] K. A. MARDAL AND R. WINTHER. **Preconditioning discretizations of systems of partial differential equations**. *Numerical Linear Algebra with Applications*, **18**(1):1–40, 2011. 9, 140, 147, 165

[88] P. MATSTOMS. **Parallel Sparse QR Factorization on Shared Memory Architectures**. *Parallel Computing*, **21**(3):473 – 486, 1995. 113

[89] H. MEUER, E. STROHMAIER, J. DONGARRA, AND H. SIMON. **TOP500 Project**. `http://top500.org/`, 2012. 129

[90] A. MILANI AND R. PICARD. **Decomposition theorems and their application to non-linear electro- and magneto-static boundary value problems**. In S. HILDEBRANDT AND R. LEIS, editors, *Partial Differential Equations and Calculus of Variations*, **1357** of *Lecture Notes in Mathematics*, pages 317–340. Springer Berlin / Heidelberg, 1988. 144

[91] P. MONK. *Finite Element Methods for Maxwell's Equations*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, UK, 2003. 145

[92] W. A. MULDER. **A Comparison between Higher-Order Finite Elements and Finite Differences for Solving the Wave Equation**. In *J. A. Desideri, P. Le Tallec, E. Onate, J. Periaux and E. Stein (eds), Proceedings of the Second ECCOMAS Conference on Numerical Methods in Engineering (Paris, Sept. 9-13, 1996)*, pages 344–350. John Wiley and Sons, 1996. xiii, 34

[93] M. F. MURPHY, G. H. GOLUB, AND A. J. WATHEN. **A Note on Preconditioning for Indefinite Linear Systems**. *SIAM J. Sci. Comput.*, **21**:1969–1972, 2000. 67

[94] J. C. NÉDÉLEC. **Mixed finite elements in** R^3. *Numerische Mathematik*, **35**:315–341, 1980. 10.1007/BF01396415. 153, 156

[95] M. OLSHANSKII, G. LUBE, T. HEISTER, AND J. LÖVE. **Grad-Div Stabilization and Subgrid Pressure Models for the Incompressible Navier-Stokes Equations**. *Comput. Meth. Appl. Mech. Engrg.*, **198**:3975–3988, 2009. 3, 25

[96] M. OLSHANSKII AND L. G. REBHOLZ. **Velocity-Vorticity-Helicity formulation and a solver for the Navier-Stokes equations**. *J. of Comput. Phys.*, **229**:4291–4303, 2010. 9

[97] C. C. PAIGE AND M. A. SAUNDERS. **Solution of sparse indefinite systems of linear equations**. *SIAM J. Numerical Analysis*, pages 617–629, 1975. 164, 169

[98] J. E. PASCIAK AND P. S. VASSILEVSKI. **Exact de Rham Sequences of Spaces Defined on Macro-elements in Two and Three Spatial Dimensions**. *SIAM J. on Scientific Computing*, **30**(5):2427–2446, 2008. 140, 176, 181

[99] T. PASSERINI, J. SLAWINSKI, U. VILLA, A. VENEZIANI, AND V. SUNDERAM. **Experiences with a computational fluid dynamics code on clouds, grids, and on-premise resources**. *submitted to J. Par. Distrib. Comput.*, 2012. 102, 134, 136

[100] J. B. PEROT. **An Analysis of the Fractional Step Method**. *J. Comput. Phys.*, **108**(1):51 – 58, 1993. 44, 45

[101] R. PICARD. **Some decomposition theorems and their application to nonlinear potential theory and Hodge theory**. *Mathematical Methods in the Applied Sciences*, **12**(1):35–52, 1990. 144

[102] A. PROHL. *Projection and Quasi-Compressibility Methods for Solving the Incompressible Navier-Stokes Equations*. Wiley Teubner Advances in Numerical Mathematics. Stuttgart: B. G. Teubner, 1997. 50

[103] L. QUARTAPELLE. *Numerical Solution of the Incompressible Navier-Stokes equations*. International Series of Numerical Mathematics. Birkhäuser-Verlag, Basel, 1993. 9, 43

[104] A. QUARTERONI, R. SACCO, AND F. SALERI. *Numerical Mathematics*. Springer Verlag, 2007. 6, 76

[105] A. QUARTERONI, F. SALERI, AND A. VENEZIANI. **Analysis of the Yosida Method for the Incompressible Navier-Stokes Equations**. *Journal des Mathématiques Pures et Appliqués*, **78**(5):473–503, 1999. 44, 45, 46, 60, 61

[106] A. QUARTERONI, F. SALERI, AND A. VENEZIANI. **Factorization Methods for the Numerical Approximation of Navier-Stokes Equations**. *Computer Methods in Applied Mechanics and Engineering*, **188**(1-3):505–526, 2000. 1, 44, 45

[107] A. QUARTERONI AND A. VALLI. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, Berlin, 1994. 1, 24, 25, 30, 32, 36, 43

[108] P. A. RAVIART AND J. M. THOMAS. **A mixed finite element method for 2nd order elliptic problems**. *Mathematical Aspects of the Finite Element Method, Lecture Notes in Mathematics*, **606**:292–315, 1977. 153

[109] J. REINDERS. *Intel Threading Building Blocks: Outfitting C++ for Multi-Core Processor Parallelism*. O'Reilly, 2007. 112

[110] N. ROTT. **Note on the History of the Reynolds Number**. *Annual Review of Fluid Mechanics*, **22**(1):1–12, 1990. 2

[111] J. W. RUGE AND K. STÜBEN. **Algebraic multigrid**. In *In Multigrid methods*, pages 73–130, 1987. 103

[112] F. SALERI AND A. VENEZIANI. **Pressure Correction Algebraic Splitting Methods for the Incompressible Navier-Stokes Equations**. *SIAM Journal on Numerical Analysis*, **43**(1):174–194, 2006. 1, 9, 47, 48, 50, 55, 60, 61

[113] J. SCHBERL, H. GERSTMAYR, AND R. GAISBAUER. **NETGEN - automatic mesh generator**. `http://www.hpfem.jku.at/netgen`, 2012. 104, 124

[114] J. SLAWINSKI, T. PASSERINI, U. VILLA, A. VENEZIANI, AND V. SUNDERAM. **Experiences with Target-Platform Heterogeneity in Clouds, Grids, and On-Premises Resources**. In *2012 26th International Parallel and Distributed Processing Symposium (IPDPS-HCW)*, pages 41–52. IEEE, 2012. 102

[115] H. SUTTER AND A. ALEXANDRESCU. *C++ Coding Standards: 101 Rules, Guidelines, and Best Practices*. C++ In-Depth Series. Addison-Wesley, Boston, 2004. 119

[116] R. TEMAM. **Sur l'Approximation de la Solution des Équations de Navier-Stokes par la méthode de Pas Fractionaires (II)**. *Arch. Rat. Mech. Anal.*, **33**:377–385, 1969. 43

[117] R. TEMAM. *Navier-Stokes Equations: Theory and Numerical Analysis*. CHEL Series. Ams Chelsea Pub., 2001. 21

[118] THE HDF GROUP. **Hierarchical data format version 5**. `http://www.hdfgroup.org/HDF5`, 2012. 105

[119] A. TOSELLI AND O. B. WIDLUND. *Domain Decomposition Methods–Algorithms and Theory*. Springer Series in Computational Mathematics. Springer, 2005. 143, 144

[120] R. S. TUMINARO AND T. TONG. **Parallel Smoothed Aggregation Multigrid: Aggregation Strategies on Massively Parallel Machines**. *SC Conference*, **0**:5, 2000. 103

[121] S. TUREK AND M. SCHÄFER. **Recent Benchmark Computations of Laminar Flow Around a Cylinder**, 1996. 3

[122] P. S. VASSILEVSKI AND U. VILLA. **A Block-Diagonal Algebraic Multigrid Preconditioner for the Brinkman Problem**. *SIAM J. Sci. Comput.*, 2012. accepted. 140, 141, 166

[123] P. S. VASSILEVSKI AND U. VILLA. **A mixed formulation for the Brinkman problem**. submitted to SINUM, available as LLNL-JNRL-563632, 2012. 141

[124] A. VENEZIANI. *Mathematical and Numerical Modeling of Blood Flow Problems*. PhD thesis, University of Milan, Italy, 1998. 61

[125] A. VENEZIANI. **Block Factorized Preconditioners for High-Order Accurate in Time Approximation of the Navier-Stokes Equations**. *Numerical Methods for Partial Differential Equations*, **19**(4):487–510, 2003. 1, 44, 45, 46, 69, 88, 113

[126] A. VENEZIANI. **A Note on the Consistency and Stability Properties of Yosida Fractional Step Schemes for the Unsteady Stokes Equations**. *SIAM Journal on Numerical Analysis*, **47**(4):2838–2843, 2009. 9, 51, 53, 56, 57, 60, 80

[127] A. VENEZIANI AND C. VERGARA. **An approximate method for solving incompressible Navier-Stokes problems with flow rate conditions**. *Comput. Methods Appl. Mech. Engrg.*, **196 (9-12)**:1685–1700, 2007. 182

[128] A VENEZIANI AND U VILLA. **ALADINS: an ALgebraic splitting time ADaptive solver for the Incompressible Navier-Stokes equations**. *J. Comput. Phys.*, 2012. accepted. 1, 9, 55, 75

[129] J. WOMERSLEY. **Method for the Calculation of Velocity, Rate of Flow and Viscous Drag in Arteries when the Pressure Gradient is Known**. *J. Physiol.*, 1955. 61

[130] X. P. XIE, J. C. XU, AND G. R XUE. **Uniformly-stable finite element methods for Darcy-Stokes-Brinkman models**. *J. Comput. Math.*, 2008. 7

[131] N. N. YANENKO. *The Method of Fractional Steps*. Springer, New York, 1971. 44

[132] O. C. ZIENKIEWICZ, R. L. TAYLOR, AND J. Z. ZHU. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 2005. 32