THE UNIVERSITY
of EDINBURGH

Preconditioned Iterative Methods for Optimal Control Problems with Time-Dependent PDEs as Constraints

Santolo Leveque

Doctor of Philosophy
University of Edinburgh
2022

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Santolo Leveque*)

Acknowledgements

The first person I would like to thank is my supervisor, John Pearson. I am grateful to him for introducing me to the interesting field of PDE-constrained optimization, but also for his support during my studies. Thanks to his comments and suggestions, I have been able to write this thesis and the journal papers it is based on. Working with him has allowed me to grow as a student, but, more than that, as a researcher in the field of numerical linear algebra. I have really appreciated the freedom he gave me in carrying out my research, synonymous of his trust in my skills.

I would like to thank Spyros Pougkakiotis, with whom I had the pleasure to work. He has been a constant source of inspiration for his commitment to research and for his insights into mathematical optimization. I would also like to thank Jacek Gondzio for his encouragement in the last years, especially at the early stage of my project.

I would like to thank the School of Mathematics at the University of Edinburgh for funding my PhD studies. I am also very grateful to the organizers of the workshop *Beyond the Discrete: Iterative Methods from the Continuum Perspective* and of the *Autumn School on Optimal Control and Optimization with PDEs* for their financial support that helped me to develop my research. I would like also to acknowledge the Society for Industrial and Applied Mathematics (SIAM) for the SIAM Student Travel Award that allowed me to participate at the *SIAM Conference on Applied Linear Algebra 2021 (LA21)*.

I am grateful to many friends and colleagues at the School of Mathematics, in particular within the Applied and Computational Mathematics and the Optimization and Operational Research groups, for creating such a stimulating environment. A special thank goes also to Filippo, Ivona, Jonna, Michael, Nagisa, and Yvonne, with whom I was able to ease into my life as a PhD student.

As we went through some restrictions due to the COVID pandemic, I have really appreciated to have people to talk to and to share my worries with during the lockdowns that took place. For this reason, I would like to thank my former flatmates Chris, Enrique, Hannah, Jenny, Kelly, and Rhys for their support and for all the laughs we had together. As I have enjoyed the time with my flatmates indoors, I appreciated even more the easening of the restrictions, for I have been able to spend more of my time with my friends Alex, Luca, and Pushpi. All of our dinners together, the picnics, the walks, the small birthday parties, and more generally all the time spent together made me believe as if we were not living during a pandemic. But more than that, I have really appreciated finding such good people as them, that really care for their friends.

I believe that a separate place should be given to all my closest friends and to my family. For all the laughs, the jokes, the good times, and the encouraging words, I would like to thank all my friends Angelo, Ciccio, Dario, Marco, Tore, and Umberto. Despite the (long) time away from one another and despite the (many) kilometers between us, it has always seemed as if we never departed from one another. Here, I would like to thank also my parents, my brother, and my sister, whose support has helped me in completing my PhD studies. Their words and their love have always been pushing me towards my goals. I would not be here without them.

Last, I would greatly acknowledge my girlfriend Natalia ("Finally", she would say), who has been supportive to (and bearing, if one wants to make jokes in Italian) me in the last two and a bit years. The time we spent together has been a *crescendo*, and I have never thought to find a partner that fits as she does in my life. I would like to thank her for all her support and love, knowing that she will always be my source of inspiration. After all, the turning point of my research began right after we started dating!

Abstract

In this work, we study fast and robust solvers for optimal control problems with Partial Differential Equations (PDEs) as constraints. Specifically, we devise preconditioned iterative methods for time-dependent PDE-constrained optimization problems, usually when a higher-order discretization method in time is employed as opposed to most previous solvers. We also consider the control of stationary problems arising in fluid dynamics, as well as that of unsteady Fractional Differential Equations (FDEs). The preconditioners we derive are employed within an appropriate Krylov subspace method.

The first key contribution of this thesis involves the study of fast and robust preconditioned iterative solution strategies for the all-at-once solution of optimal control problems with time-dependent PDEs as constraints, when a higher-order discretization method in time is employed. In fact, as opposed to most work in preconditioning this class of problems, where a (first-order accurate) backward Euler method is used for the discretization of the time derivative, we employ a (second-order accurate) Crank–Nicolson method in time. By applying a carefully tailored invertible transformation, we symmetrize the system obtained, and then derive a preconditioner for the resulting matrix. We prove optimality of the preconditioner through bounds on the eigenvalues, and test our solver against a widely-used preconditioner for the linear system arising from a backward Euler discretization. These theoretical and numerical results demonstrate the effectiveness and robustness of our solver with respect to mesh-sizes and regularization parameter. Then, the optimal preconditioner so derived is generalized from the heat control problem to time-dependent convection–diffusion control with Crank–Nicolson discretization in time. Again, we prove optimality of the approximations of the main blocks of the preconditioner through bounds on the eigenvalues, and, through a range of numerical experiments, show the effectiveness and robustness of our approach with respect to all the parameters involved in the problem.

For the next substantial contribution of this work, we focus our attention on the control of problems arising in fluid dynamics, specifically, the Stokes and the Navier–Stokes equations. We firstly derive fast and effective preconditioned iterative methods for the stationary and time-dependent Stokes control problems, then generalize those methods to the case of the corresponding Navier–Stokes control problems when employing an Oseen approximation to the non-linear term. The key ingredients of the solvers are a saddle-point type approximation for the linear systems, an inner iteration for the $(1,1)$-block accelerated by a preconditioner for convection–diffusion control problems, and an approximation to the Schur complement based on a potent commutator argument applied to an appropriate block

matrix. Through a range of numerical experiments, we show the effectiveness of our approximations, and observe their considerable parameter-robustness.

The final chapter of this work is devoted to the derivation of efficient and robust solvers for convex quadratic FDE-constrained optimization problems, with box constraints on the state and/or control variables. By employing an Alternating Direction Method of Multipliers for solving the non-linear problem, one can separate the equality from the inequality constraints, solving the equality constraints and then updating the current approximation of the solutions. In order to solve the equality constraints, a preconditioner based on multilevel circulant matrices is derived, and then employed within an appropriate preconditioned Krylov subspace method. Numerical results show the efficiency and scalability of the strategy, with the cost of the overall process being proportional to $\bar{N} \log \bar{N}$, where $\bar{N}$ is the dimension of the problem under examination. Moreover, the strategy presented allows the storage of a highly dense system, due to the memory required being proportional to $\bar{N}$.

Lay Summary

Optimal control problems often arise in industrial and real-life applications. In particular, this class of problems often concerns finding the "work" that should act on a given physical system in order to obtain a desired outcome, with a cost that is minimal.

A simple example of an optimal control problem is given by the control of the temperature in a room. For instance, suppose during winter one wishes to keep the temperature of a room around 20°C. In order to obtain this, one may turn on the heating for the whole day. However, this choice may impact one's finances, therefore one wishes to find a solution to this problem in a more efficient way. This can be done by solving an optimal control problem, in which the model drives the temperature of the room towards a desired one while reducing the energy provided to the room for the heating. Other examples are given by the control of the flow of a fluid in a pipe, the separation of some chemicals, or the modification of the atmospheric conditions of some system.

As one can infer from our surroundings, many real-world processes are evolutionary, meaning that they depend on the time. Therefore, one can expect that a large class of optimal control problems are modeled using time-dependent equations. Of late, a great effort has been devoted to the solution of this class of problems. However, optimal control problems (either steady or time-dependent) do not in general have a closed form solution, therefore numerical methods are employed in order to find an approximation of it. This in turn requires the solution of a (sequence of) system(s) of linear equations. Although direct methods can be employed as a black-box for the solution of the resulting linear systems, they can suffer from memory limitations, in particular when finding an accurate solution of time-dependent problems. As an alternative, one can employ iterative methods in order to find (approximations of) the solutions of the corresponding linear systems to be solved. These methods are quite cheap to apply, and usually do not suffer from memory limitations. However, the convergence of this class of methods is often determined by the conditioning of the system to be solved. For this reason, one introduces a preconditioner in order to produce a better conditioning of the linear system considered and therefore to speed up the convergence of the method.

In this thesis, we delevop fast and robust solvers for the numerical solution of optimal control problems. Specifically, this work focuses on the development of suitable preconditioners for the linear systems arising from a range of optimal control problems. For a large part of this thesis, we focus our study on time-dependent problems, and we also consider time-independent systems.

Contents

Chapter 0

Introduction

"Ho un foglio bianco
Per volare alto lo macchio
Come l'ala di un albatro
Per la città della China
Mi metto in viaggio (da bravo)
Pellegrinaggio
Ma non a Santiago
Vado a China Town"

["I have a white sheet
To fly high I stain it
Like the wing of an albatross
To China Town
I get on the road (as a good boy)
Pilgrimage
But not to Santiago
I am going to China Town"]

– Caparezza, *China Town*[1]

For centuries, scientists have been using differential operators in order to model reality. In fact, many physical processes, from mechanics to dynamics, from biology to economics, from engineering to chemistry, passing through continuum mechanics and thermodynamics, can be described by the relation of some physical quantities with a differential operator. For instance, one can model the evolution of an epidemic through the *SIR model* (on the topic of the ongoing coronavirus pandemic). In addition, thanks to the advances in scientific computing and in numerical analysis, scientists have also been focusing on the numerical solution of such models. In particular, many researchers have been working on the development of robust and efficient linear solvers for the discretizations of the differential operator analysed. A particular class of differential operators is given by *Partial Differential Equations (PDEs)*, which relate a physical quantity to its

[1]This song was inspired by Malevič's Black Square, and is a declaration of the author's love for writing (indeed, China is the Italian word for Indian ink). Broadly speaking, the song may also refer to mathematicians (or scientists in general), as they know quite well how to "stain a sheet".

partial derivatives.

As above, PDEs are very suitable for describing the physics of real-life problems. However, rather than only determining the realisation of a physical process, one may be interested in modifying a given physical quantity. For instance, this could represent the act of modifying the flow of a liquid in a pipe. Alternatively, one may wish to keep the temperature of a room close enough to a given desired temperature. This type of problems falls into the category of inverse problems. Indeed, one wishes to find what type of "work" should act on the physical mechanism in order to obtain their stated objective. Specifically, one wishes to *control* the physics in some sense. The class of problems so described is referred to as PDE-constrained optimization problems. Coming back to the example of keeping the temperature of a room close to a desired one, this problem can be described by the *optimal control of the heat equation*, or simply *heat control*.

As we will see in the following chapter, PDE-constrained optimization problems are very well suited for describing the act of modifying a given physical system, and arise quite often in industrial applications. In recent years, many researchers have devoted their effort to devise methods for the numerical solution of the optimal control of PDEs. This thesis is the result of one such effort. Specifically, our study focuses on the development of parameter-robust preconditioned iterative methods for the solution of optimal control problems with PDEs as constraints. For a large part of this thesis, we consider time-dependent PDEs as constraints, but we also study stationary problems on occasions. In addition, we consider also the optimal control of *Fractional Differential Equations (FDEs)* as constraints, with additional algebraic (box) constraints imposed on the variables. The contents of this thesis can be summarized as follows.

In Chapter 1, we introduce the type of problems considered in this thesis, that is optimal control problems with differential operators as constraints. Then, we describe the strategies employed for obtaining the optimality conditions that a solution of the problem has to satisfy, namely the *optimize-then-discretize* and the *discretize-then-optimize* approaches. Finally, we describe some of the methods employed to obtain a numerical solution of an optimal control problem with differential operators as constraints, with additional algebraic constraints on the variables. All of these approaches lead to discretized systems of linear equations.

In Chapter 2, we give an overview of a number of iterative methods used for the solution of (very large) linear systems of equations. Among those, we mainly focus on Krylov subspace methods, as they require only matrix–vector or vector–vector operations at each iteration. Since the convergence of those methods depends (but not only) on the distribution of the eigenvalues, a preconditioned version of the Krylov subspace methods is usually employed in practice. The latter leads us towards the theory of saddle-point systems, and the introduction of optimal preconditioners for linear systems with a saddle-point type of structure. As we will be mentioning quite often in this work, the "ideal" preconditioners are never applied in practice, as the cost of applying their inverse operator is almost as costly as inverting the original system. For this reason, one would rather find easy-to-invert approximation of the main blocks of the preconditioners considered. This is the "philosophy" that we will adopt for the rest of this work. In particular, we look for suitable approximations of the Schur complements of each system,

that are optimal with respect to all the parameters involved in the problem, meaning that the eigenvalues of the preconditioned Schur complement are (in the best case) clustered in a region of the complex plane independently of the parameters involved, or that the linear solver converges in a (roughly) constant number of iterations.

The first novel contribution that we present in this work is the optimal preconditioner for the heat control problem when applying the Crank–Nicolson discretization in time, which we discuss in Chapter 3. We would like to note that the Crank–Nicolson method may refer to either the implicit midpoint rule or to the implicit trapezoidal rule. In general, these two methods are not equivalent when applied to non-linear problems for instance. For the rest of this work, we will refer to the implicit trapezoidal method as the Crank–Nicolson method. By employing an optimize-then-discretize strategy, we derive the first-order optimality conditions of the heat control problem. Those conditions are given by a coupled system of linear PDEs, one moving forward in time, the other backward. Most of the previous work in preconditioning for the control of time-dependent PDEs is based on a (first-order accurate) backward Euler discretization for the time variable. However, in this work we employ a (second-order accurate) Crank–Nicolson method in time, which leads to a (non-symmetric) system with much more complex structure. We apply a tailored invertible transformation, and symmetrize the system so obtained. From here, we employ saddle-point theory to devise an optimal preconditioner for the system under examination. The key components of this preconditioner are an accurate mass matrix approximation, a good approximation of the Schur complement based on a matching strategy, and an appropriate multigrid process to apply this latter approximation – these are constructed using our work in transforming the linear system. We prove the optimality of our approximation of the Schur complement through bounds on the eigenvalues that are independent of any parameters within the problem setup. This optimality of the preconditioner is also observed in the numerical results, as the iterative solver is able to reach a prescribed accuracy in a constant number of iterations. In addition, the numerical results show the substantial speed-up obtained by employing our strategy compared to the state-of-the-art preconditioned backward Euler method.

In Chapter 4, the optimal preconditioner that we derive for heat control problems with Crank–Nicolson in time is extended to the control of the time-dependent (stabilized) convection–diffusion equation. By employing a similar strategy as for heat control, one is able to symmetrize the discrete optimality conditions. Again, by exploiting the transformation used for the symmetrization of the linear system, we are able to derive an optimal preconditioner for the problem considered. In this case, the optimality of the Schur complement approximation is guaranteed only under a suitable assumption. Numerical results show the efficiency of the preconditioner, as the linear solver requires a constant number of iterations for reaching convergence.

The second major contribution of this work is that of devising robust preconditioners for the control of viscous fluid flow, which is the topic of Chapter 5 and Chapter 6. For those problems, the constraints are the (linear) incompressible Stokes equations, or the (non-linear) incompressible Navier–Stokes equations,

that we consider in both the stationary and time-dependent settings. For the time-dependent problems, we employ both backward Euler or Crank–Nicolson as alternatives for discretizing the time derivative. Again, by adopting an optimize-then-discretize approach, we derive the first-order optimality conditions that a critical point has to satisfy. Since the conditions for the Navier–Stokes control problems are given by a system of non-linear PDEs, we adopt a linearization for deriving an approximation to a critical point. Specifically, we employ an Oseen linearization of the convection term. Then, by discretizing the optimality conditions, we are faced with systems with saddle-point type of structure. The leading block of these systems can be considered, in the stationary case, as the discrete optimality conditions of an incompressible Poisson or stationary convection–diffusion control problem, and, in the time-dependent case, as the discrete optimality conditions of an incompressible heat or time-dependent convection–diffusion control problem. Thanks to this structure, we employ optimal preconditioners for Poisson control, heat control, or (stationary or time-dependent) convection–diffusion control problems within a suitable Krylov solver in order to approximately invert the $(1,1)$-block. For the Stokes control problem, we observe that the inverse operator of the $(1,1)$-block can also be applied by employing a fixed number of Uzawa iterations. For Stokes and Navier–Stokes control problems, the most complex task in order to find robust preconditioners is to approximate the Schur complement arising from each discretized problem. We do so by applying a block commutator argument to a suitable block matrix. The preconditioners so derived are robust and efficient, showing only a mild dependence on the regularization parameter β and on the viscosity ν (the latter only for Navier–Stokes control problems). In addition, the CPU times scale linearly with the problem size, aside from the multigrid routine employed. Finally, the flexibility of the block commutator argument allows us to solve the problems in both the stationary and time-dependent settings, and to apply both backward Euler or Crank–Nicolson in time for instationary problems. To our knowledge, no existing preconditioner offers similar flexibility and parameter-robustness when solving Stokes and Navier–Stokes control problems.

The last novel contribution of this work is that of devising robust and efficient preconditioned iterative methods for the optimal control of FDEs with additional algebraic constraints on the state and the control variables, for FDEs defined on spatial domains of dimension at least one, and also in time. This is discussed in Chapter 7. For this class of problems, we employ a discretize-then-optimize strategy, which leads to a convex quadratic programming problem. In order to find an approximation of the solution, we employ the *Alternating Direction Method of Multipliers (ADMM)*, which allows us to separate inequality from equality constraints. The latter are then solved by employing a multilevel circulant-based preconditioner within a suitable Krylov subspace solver. Then, we update the remaining solutions, and iterate the process until convergence. The proposed preconditioner is proved to be optimal with respect to the mesh-size, and can be readily generalized to the case of FDE-constrained optimization problems in higher dimensions. Numerical results show the efficiency of our approach, with the CPU time scaling as $\bar{N}\log\bar{N}$ and storage cost of order $\bar{N}$, where $\bar{N}$ is the dimension of the grid used.

The work of this thesis has resulted in the following publications:

- Leveque S., Pearson J. W.: *Fast iterative solver for the optimal control of time-dependent PDEs with Crank–Nicolson discretization in time*, Numer. Linear Algebra Appl. **29**, e2419, 2022 (Ref. [100]);

- Leveque S., Pearson J. W.: *Parameter-Robust Preconditioning for Oseen Iteration Applied to Stationary and Instationary Navier–Stokes Control*, to appear in SIAM J. Sci. Comput., arXiv:2108.00282 (Ref. [101]);

- Leveque S., Pearson J. W.: *Parameter-Robust Preconditioning for Unsteady Stokes Control Problems*, PAMM **21**, e202100131, 2021 (Ref. [99]);

- Pougkakiotis S., Pearson J. W., Leveque S., Gondzio J.: *Fast Solution Methods for Convex Quadratic Optimization of Fractional Differential Equations*, SIAM J. Matrix Anal. Appl. **41**, 1443–1476, 2020 (Ref. [146]).

Chapter 3 and Chapter 4 are based on [100]. Chapter 5 is based on [99] and on some of the work in [101]. Chapter 6 is based on some of the work in [101]. Finally, Chapter 7 is based on the work in [146].

Chapter 1

Optimal Control Problems with PDEs as Constraints

"Hast thou, spirit,
Perform'd to point the tempest that I bade thee?"
– William Shakespeare, *The Tempest*

As one can imply from the title, the major topic of this work is the design of preconditioned iterative methods for time-dependent PDE-constrained optimization problems. Although the expression "iterative methods" is mentioned first, we leave their description to the next chapter, and focus here on a mathematical formulation of the problems under examination, that is optimal control problems with differential equations as constraints.

As we described in Chapter 0, PDE-constrained optimization problems are very suitable models for describing the act of modifying a given physical system (like Prospero modifying the elements at will). Due to this property, problems of this type often arise in industrial and real-life applications. For instance, the control of the heat equation can describe the heating of a tissue during a medical treatment or of a substance during a chemical process; in addition, in a chemical process one could control the separation of immiscible fluids, whose physics is governed by *phase field equations*. Alternatively, one could optimize the shape of an object to obtain a desired physical outcome (such as the shape of a wing in an airplane to avoid turbulence). Finally, one could control the motion of fluid flows, or modify the atmospheric conditions of some system.

In the following, we will introduce the mathematical formulation for a general PDE-constrained optimization problem, and show how to obtain a solution. The latter has to satisfy some conditions, that may vary depending on whether we are solving the problem in the Hilbert space, or are solving it only on a finite-dimensional subset. In the former case, we have to introduce some "differentiation rule" that will lead to the optimality conditions, that then have to be discretized; in the second case, we have to decide the subset of points on which we want to approximately solve the problem by discretizing the PDE, and then deriving optimality conditions by mean of classical optimization theorems. These two ways of action lead to two different strategies, namely the *optimize-then-discretize strategy* (we solve the problem in the Hilbert space and then discretize the condi-

tions derived), and the *discretize-then-optimize strategy* (we discretize the PDE we want to optimize, and then derive optimality conditions).

Since in both the strategies we will need a discretization, we will first introduce the *finite element method*. The numerical solution obtained by this class of methods is typically defined by a projection: the PDE has to be satisfied only on a subset of the Hilbert space. This subset is chosen *a priori* by the type of elements one wants to adopt, and will influence the resulting approximation properties of the method. Finite element discretization usually leads to systems of linear equations that are very large and very sparse if one desires a highly accurate solution, but, as mentioned above, we will leave the solution strategies for those systems as the main topic of the next chapter.

1.1 Introduction to Optimal Control of PDEs

In the following, we introduce the mathematical formulation of the problem under examination, namely optimal control problems with PDEs as constraints. We refer to [86, 104, 175] for a detailed discussion of these problems.

Given a domain $\Omega \subset \mathbb{R}^d$, with $d \in \mathbb{N}$, let us suppose we can relate some observation (or *state*) v to the physics acting inside the domain through a differential operator $\mathcal{D}$, that is, we can write a relation of the type

$$\mathcal{D}v = f,$$

f being the *force function* representing the physics acting on Ω. Clearly, for the problem to be well posed we need some *boundary conditions* on v. These conditions can be given as *Dirichlet conditions*

$$v = g_D \qquad \text{on } \partial\Omega_D,$$

on part of the boundary, that is, we know the value of the function v on $\partial\Omega_D$, or as *Neumann conditions*

$$\frac{\partial v}{\partial \vec{n}} = g_N \qquad \text{on } \partial\Omega_N,$$

on part of the boundary, with $\dfrac{\partial v}{\partial \vec{n}}$ the (outward) normal derivative of v on $\partial\Omega_N$, that is, we know the *flux* of v through $\partial\Omega_N$. The portion of the boundary $\partial\Omega_D$ and $\partial\Omega_N$ are such that $\overline{\partial\Omega} = \overline{\partial\Omega}_D \cup \overline{\partial\Omega}_N$ and $\partial\Omega_D \cap \partial\Omega_N = \varnothing$. The conditions on v presented here are not the only possibility, as one could also impose *Robin boundary conditions*.

In addition, if the differential operator $\mathcal{D}$ involves time derivatives, we require some additional information on v (and on some of its partial derivatives with respect to the time variable) at the initial time t_0; for instance, if $\mathcal{D}$ is a *parabolic differential operator* defined on $\Omega \times (t_0, t_f)$, with t_0 and t_f the initial and final time respectively, an *initial condition*

$$v(x, t_0) = v_0(x) \qquad \text{in } \Omega$$

has to be given to define a suitable PDE-constrained optimization problem.

For most of the following work, we will be dealing with a differential operator $\mathcal{D}$ represented by (a system of) PDEs. We refer the interested reader to [46] for an introduction and a thorough analysis of (forward) PDEs.

Once the physics is modelled by the differential operator $\mathcal{D}$ subject to appropriate boundary or initial conditions, the state v will be (uniquely, if the operator $\mathcal{D}$ is linear) defined. However, one may be interested in obtaining a state v with a particular shape or value; for instance, given a body to heat one wishes to have an average given temperature. In practice, one wishes to obtain a state v "close enough" in some sense to a *desired state* v_d. Since we are interested in modifying the physics, we introduce a *control* u in the formulation of the differential operator. The problem so described can be formulated as an optimization problem whose constraints are the differential operator $\mathcal{D}$ and its associated (boundary or initial) conditions. This formulation is classified as the *optimal control of PDEs* or a *PDE-constrained optimization problem*.

As for many optimization problems, the optimal control of PDEs concerns the minimization of a cost functional $J(v, u)$ that takes into account how close is the state v to our desired state v_d. In the following, the cost functional $J(v, u)$ will contain (a multiple of) the squared $L^2(\Omega)$-norm of the difference between v and v_d, that is $\|v - v_d\|^2_{L^2(\Omega)}$.

As mentioned above, in an optimal control problem involving PDEs the cost functional $J(v, u)$ is subject to the differential operator $\mathcal{D}$ and its boundary or initial conditions, which take also into account the action of the control u. Depending on how we introduce the control u, we may have different types of optimal control problems of PDEs. Although the main results obtained in this work relate to *distributed control* problems with PDEs as contraints (the control is acting on the whole domain), below we also introduce another type of control, namely the *boundary control* problem (as one can easily understand from the name, the control lives only on some part of the boundary). Other types of control are of course possible, but we will limit our exposition to those two. For instance, one could apply the control u only on part of the domain, obtaining in this way a *subdomain control problem*.

Let us introduce first the distributed optimal control of PDEs. In this case, the control u is defined on the whole domain Ω, and introduced in the differential equation as

$$\mathcal{D}v = u + f \qquad \text{in } \Omega;$$

again, the equation above is a constraint with respect to which we minimize our cost functional $J(v, u)$.

On the other hand, rather than introducing the control in the whole domain one could introduce the control only on the (Dirichlet) boundary, obtaining in this way the optimal boundary control problems of PDEs. The latter is defined as the minimization of $J(v, u)$ subject to

$$\begin{cases} \mathcal{D}v = f & \text{in } \Omega, \\ v = u + g_D & \text{on } \partial\Omega_D, \\ \dfrac{\partial v}{\partial \vec{n}} = g_N & \text{on } \partial\Omega_N. \end{cases}$$

Similarly, one could introduce the control only on the Neumann boundary, obtaining a Neumann boundary control problem. The latter is defined as the minimization of $J(v,u)$ subject to

$$\begin{cases} \mathcal{D}v = f & \text{in } \Omega, \\ v = g_D & \text{on } \partial\Omega_D, \\ \dfrac{\partial v}{\partial \vec{n}} = u + g_N & \text{on } \partial\Omega_N. \end{cases}$$

In addition, given a subset $\Omega_{\text{sub}} \subset \Omega$, the subdomain control problem is defined as the minimization of $J(v,u)$ subject to

$$\begin{cases} \mathcal{D}v = \begin{cases} u + f & \text{in } \Omega_{\text{sub}}, \\ f & \text{in } \Omega \setminus \Omega_{\text{sub}}, \end{cases} \\ v = g_D & \text{on } \partial\Omega_D, \\ \dfrac{\partial v}{\partial \vec{n}} = g_N & \text{on } \partial\Omega_N. \end{cases}$$

As one can understand from the description so far, optimal control problems of PDEs fall within the class of inverse problems. As these problems may be ill-posed, one adds a cost term in the cost functional $J(v,u)$ in order to ensure the existence of a solution. In the following, we will introduce a *Tikhonov regularization* for the problem under examination; this is defined as a multiple of the squared $L^2(\Omega)$-norm of the control u. Specifically, in the following we will consider

$$J(v,u) = \frac{1}{2}\|v - v_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} \tag{1.1}$$

as cost functional to minimize; here, the value $\beta > 0$ is referred to as *regularization parameter*. From the point of view of the physics, if we look at the term $\|u\|^2_{L^2(\Omega)}$ as the energy that we introduce into the system for modifing the state v, the parameter β can be viewed as a "trade-off" between how close we want to drive v to our desired state v_d, and the cost can afford in order to achieve it. We would like to note that other cost functionals can be considered; for instance, one could replace the $L^2(\Omega)$-norm with the $L^1(\Omega)$-norm in order to promote sparsity of the solutions. In addition, in the case of subdomain control problems the cost functional $J(v,u)$ has to take into account that the control u is defined only on a subset Ω_{sub} of the whole domain Ω; this is done by replacing the term $\|u\|^2_{L^2(\Omega)}$ with $\|u\|^2_{L^2(\Omega_{\text{sub}})}$.

We can finally formulate the structure of the problems we will be studying for the rest of this work. The distributed optimal control problem of PDE is defined as

$$\min_{v,u} \quad J(v,u) \tag{1.2}$$

subject to

$$\begin{cases} \mathcal{D}v = u + f & \text{in } \Omega, \\ v = g_D & \text{on } \partial\Omega_D, \\ \dfrac{\partial v}{\partial \vec{n}} = g_N & \text{on } \partial\Omega_N, \end{cases} \tag{1.3}$$

with $J(v, u)$ defined as in (1.1). It is worth mentioning that, depending on the differential operator $\mathcal{D}$, other conditions on v may be given; for instance, if $\mathcal{D}$ is a parabolic differential operator, the constraints (1.3) of the problem will take into account the initial condition on the state v, while the boundary conditions have to be satisfied over $\partial\Omega \times (t_0, t_f)$.

In the formulation (1.2)–(1.3), the constraints take into account only the physics of the problem under examination, namely the differential operator $\mathcal{D}$. However, as we are adding the control to the system, it may happen, for instance, that we have some technological limitation on the control we may apply, or we may be interested in a state v that is bounded in a certain region of the domain Ω. For this reason, problems of the type (1.2)–(1.3) can also take into account algebraic constraints on the state and/or the control variables. The formulation of these problems is the same as in (1.2)–(1.3), but the constraints (1.3) also include bounds of the type

$$v_{\min} \leqslant v \leqslant v_{\max}, \qquad u_{\min} \leqslant u \leqslant u_{\max}, \tag{1.4}$$

where $v_{\min}$, $v_{\max}$, $u_{\min}$, and $u_{\max}$ could be constants or functions. Optimal control problems of PDEs with additional algebraic constraints on the state or the control variables are more complex to solve than the corresponding "unconstrained" problems, as to find an approximation of the solution we need to employ a nonlinear iteration within the solver, even when the rest of the problem is linear. For most of this thesis, we will be dealing only with problems of the type (1.2)–(1.3), and will consider the constraints on the variables (1.4) only in the last main chapter.

As we are interested in the numerical solution of optimal control problems with PDEs as constraints, below we are going to describe two strategies for deriving such an approximation, that is, the optimize-then-discretize strategy, and the discretize-then-optimize strategy. As for both the approaches we need to discretize a system of PDEs, in the next section we are going to introduce a widely used discretization method for PDEs, namely the *finite element method*.

Before moving on with our exposition, we would like to introduce two PDE-constrained optimization problems that will be considered below, namely, the *distributed Poisson control* and the *distributed (stationary) Navier–Stokes control* problems. While devising robust solvers for the numerical solution of the distributed (stationary and instationary) Navier–Stokes control problem will be one of the final goals of this thesis, the distributed Poisson control problem will be introduced only as a "proof of concept" to illustrate the main ideas presented in the background sections.

Given a spatial domain $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, a regularization parameter $\beta > 0$, and a desired state v_d, the distributed Poisson control problem is defined as

$$\min_{v,u} \quad \frac{1}{2}\|v - v_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} \tag{1.5}$$

subject to the Poisson equation

$$\begin{cases} -\nabla^2 v = u + f & \text{in } \Omega, \\ v = g & \text{on } \partial\Omega, \end{cases} \tag{1.6}$$

where we have imposed only Dirichlet boundary conditions. Problem (1.5)–(1.6) can represent, for instance, a body heated by electromagnetic induction or by microwaves (*optimal stationary heat source*), see, for example, [175, Section 2.8.2].

As we will describe more in detail in Chapter 6, the Navier–Stokes equations describe the motion of an incompressible viscous fluid. Given a spatial domain $\Omega \subset \mathbb{R}^d$, with $d = 2, 3$, a regularization parameter $\beta > 0$, and a desired state $\vec{v}_d$ defined over d dimensions, the distributed control of the stationary Navier–Stokes equations is defined as

$$\min_{\vec{v},\vec{u}} \quad \frac{1}{2}\|\vec{v} - \vec{v}_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|\vec{u}\|^2_{L^2(\Omega)}$$

subject to

$$\begin{cases} -\nu\nabla^2\vec{v} + \vec{v}\cdot\nabla\vec{v} + \nabla p = \vec{u} + \vec{f} & \text{in } \Omega, \\ -\nabla\cdot\vec{v} = 0 & \text{in } \Omega, \\ \vec{v} = \vec{g} & \text{on } \partial\Omega, \end{cases}$$

where the parameter $\nu > 0$ is the *viscosity* of the fluid, with $\vec{v}$ representing the *velocity* of the fluid and p the *kinematic pressure*; the velocity $\vec{v}$ and the control $\vec{u}$ are vector functions, while the pressure p is a scalar function. Note that the fluid is Newtonian and has constant density. The main difference between this problem and the previous one lies in the non-linear term $\vec{v}\cdot\nabla\vec{v}$, which requires (as we will see) a careful treatment for the numerical solution of the problem.

1.1.1 Applications

We would like to conclude this section with some further motivation for the study of the numerical solutions of optimal control problems with PDEs as constraints. For further overviews of the applications of PDE-constrained optimization problems in science and engineering, we refer the reader to [74, 133].

Problems of the type (1.2)–(1.3) arise very naturally in industrial and real-life applications. The classical example is given by the control of the heat equation, describing the heating of a tissue during a medical treatment or a chemical in an industrial process. Another important application of distributed PDE-constrained optimization problems in industrial processes is given by the control of the Navier–Stokes equations introduced above, see, for instance, [72]. We survey some additional applications below.

In [37], the authors employ PDE-constrained optimization to image denoising problems. More specifically, the authors solve a bilevel optimization problem defined as a minimization of a functional whose constraints are other minimization problems. The functional takes into account the discrepancies between a set of noisy images and the corresponding true image, as well as some weights of the different noise models considered. The minimization problems that represent the

constraints involve non-smooth PDEs that take into account the noise, and the total variation of the given images. The choice of introducing the total variation in the constraints is made to accurately preserving the edges within the images.

In [85], the authors devise a Newton method for solving the optimal design of semiconductor devices; the latter is formulated as the minimization of a cost functional subject to the drift-diffusion model for semiconductor devices.

In [160], the authors study model predictive control of the cooling process during wine fermentation. The problem is formulated as the minimization of a cost functional that takes into account for the energy consumption during the cooling process, while driving the sugar consumption towards a desired sugar reduction process, with the state variable being the solution of an *ordinary differential equation (ODE)*; an initial condition on the state variable and additional constraints on the differential states and parameter are also imposed.

Other practical examples of PDE-constrained optimization problems include shape optimization problems [1, 126, 148], medical imaging and tomography [5, 33, 93], mathematical finance [20, 42], reaction–diffusion control for chemical processes [10, 71], the Monge–Kantorovich mass transfer problem [3, 14], and flow control in porous media [47, 76, 171].

1.2 Finite Element Method

In this section we introduce a widely used numerical method for the solution of differential equations, namely the finite element method. For a more comprehensive discussion on the finite element method, we refer the reader to [26, 44, 149].

Starting from the weak formulation of the differential operator under examination, the finite element method constructs a bilinear form (on real spaces), which is then employed as an inner product in an appropriate space. Then, the method selects a subspace and imposes the inner product of the numerical error and any element of the subspace to be 0, that is to say, the method imposes the numerical error and the chosen subspace to be mutually orthogonal. The subspace is usually determined by *basis functions*, defined on the *elements* of the constructed mesh in the domain. Finally, the orthogonality condition of the numerical error and the subspace results in a system of linear equations that has to be solved.

In order to better explain how the finite element method works, given a domain $\Omega \subset \mathbb{R}^d$, we consider the (forward) Poisson equation with Dirichlet boundary conditions

$$\begin{cases} -\nabla^2 v = f & \text{in } \Omega, \\ v = g & \text{on } \partial\Omega, \end{cases} \tag{1.7}$$

with f and g given functions. We first introduce the space $L^2(\Omega)$ of square-integrable functions in $\mathbb{R}$ with domain in Ω, that is

$$L^2(\Omega) := \left\{ v : \int_\Omega |v|^2 \mathrm{d}\Omega < \infty \right\}.$$

The solutions of the problem (1.7) are usually sought in the space

$$V := \left\{ v \in \mathcal{H}^1(\Omega) \mid v = g \text{ on } \partial\Omega \right\},$$

where $\mathcal{H}^1(\Omega)$ is the *Sobolev space* of square-integrable functions in $\mathbb{R}$ with square-integrable weak derivatives, namely

$$\mathcal{H}^1(\Omega) := \left\{ v \in L^2(\Omega) \ : \ \frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2}, \ldots, \frac{\partial v}{\partial x_d} \in L^2(\Omega) \right\},$$

with $x = (x_1, x_2, \ldots, x_d) \in \Omega$. Finally, we introduce the space of Sobolev functions with zero *trace* on the boundary $\partial\Omega$

$$V_0 := \mathcal{H}_0^1(\Omega) = \left\{ v \in \mathcal{H}^1(\Omega) \mid v = 0 \text{ on } \partial\Omega \right\}.$$

Then, the weak formulation of (1.7) reads as:
Find $v \in V$ such that

$$\int_\Omega \nabla v \cdot \nabla w \, \mathrm{d}\Omega = \int_\Omega f w \, \mathrm{d}\Omega \qquad \text{for all } w \in V_0. \tag{1.8}$$

The weak formulation introduces the (bi)linear form $a(\cdot,\cdot) : \mathcal{H}^1(\Omega) \times \mathcal{H}^1(\Omega) \to \mathbb{R}$ defined as

$$a(v, w) = (\nabla v, \nabla w),$$

where $(\cdot,\cdot)$ is the L^2-inner product on Ω. Defining also the linear functional $\ell(\cdot) : \mathcal{H}^1(\Omega) \to \mathbb{R}$ as $\ell(w) = (f, w)$, we can rewrite (1.8) as

$$a(\nabla v, \nabla w) = \ell(w) \qquad \text{for all } w \in V_0.$$

As we are looking for a numerical appoximation of v, we select a finite-dimensional subspace V_h of V. In order to define a generic element of V, we consider an n_x-dimensional subspace of *test functions* $V_{0,h}$ of V_0; we suppose that $\{\phi_1, \phi_2, \ldots, \phi_{n_x}\}$ is a basis for $V_{0,h}$. Since the solution we are looking for is required to be equal to g on the boundary, we introduce also the functions $\{\phi_{n_x+1}, \phi_{n_x+2}, \ldots, \phi_{n_x+n_\partial}\}$, with $n_\partial \in \mathbb{N}$. Then, the subspace V_h is defined to be

$$V_h := \operatorname{span}\{\phi_1, \phi_2, \ldots, \phi_{n_x}\} + \sum_{i=n_x+1}^{n_x+n_\partial} v_i \phi_i,$$

where the coefficients v_i are chosen such that the function $\sum_{i=n_x+1}^{n_x+n_\partial} v_i \phi_i$ interpolates the boundary data; the functions $\{\phi_1, \phi_2, \ldots, \phi_{n_x}\}$ in the previous expression are called *trial functions*. In practice, the function v is approximated by the function $v_h := \sum_{i=1}^{n_x} v_i \phi_i + \sum_{i=n_x+1}^{n_x+n_\partial} v_i \phi_i$. From here, since the function g is fixed, we can say that the approximation v_h is uniquely determinined by the vector $\boldsymbol{v} := [v_1, v_2, \ldots, v_{n_x}]^\top$.

As we mentioned above, the finite element method typically imposes some orthogonality condition between the numerical error $e_h = v - v_h$ and an appropriate subspace through a bilinear form. Specifically, we impose that the error e_h

and the subspace $V_{h,0}$ are orthogonal with respect to the inner product induced by the bilinear form $a(\cdot,\cdot)$, that is, we impose that

$$a(e_h, w_h) = 0 \qquad \text{for all } w_h \in V_{0,h}.$$

The previous expression can be rewritten as

$$a(v_h, w_h) = \ell(w_h) \qquad \text{for all } w_h \in V_{0,h}, \tag{1.9}$$

by substituting the expression $e_h = v - v_h$ and noting that $a(v, w_h) = \ell(w_h)$ for all $w_h \in V_{0,h}$. As each element of $V_{0,h}$ is a linear combination of the elements of the basis $\{\phi_1, \phi_2, \ldots, \phi_{n_x}\}$, we need to impose the expression (1.9) only on each ϕ_i, for $i = 1, 2, \ldots, n_x$. Further, since $v_h := \sum_{i=1}^{n_x} v_i\phi_i + \sum_{i=n_x+1}^{n_x+n_\partial} v_i\phi_i$, equation (1.9) results into a linear system to solve for finding the vector $\boldsymbol{v}$; specifically, (1.9) can be rewritten as

$$a\Big(\textstyle\sum_{i=1}^{n_x} v_i\phi_i, \phi_l\Big) = \ell(\phi_l) - a\Big(\textstyle\sum_{i=n_x+1}^{n_x+n_\partial} v_i\phi_i, \phi_l\Big) \qquad \text{for } l = 1, 2, \ldots, n_x,$$

or equivalently

$$K\boldsymbol{v} = \hat{\boldsymbol{f}},$$

with

$$\begin{aligned} K = \{k_{i,l}\}_{i,l=1}^{n_x}, \qquad & k_{i,l} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_l \,\mathrm{d}\Omega, \\ \hat{\boldsymbol{f}} = \{\hat{f}_i\}_{i=1}^{n_x}, \qquad & \hat{f}_i = \int_\Omega f\phi_i \,\mathrm{d}\Omega - \sum_{l=n_x+1}^{n_x+n_\partial} v_l \int_\Omega \nabla\phi_l \cdot \nabla\phi_i \,\mathrm{d}\Omega. \end{aligned}$$

Note that the formulation derived here holds if only Dirichlet boundary conditions are imposed; if within the problem formulation Neumann boundary conditions have to be satisfied, extra terms will arise in the right-hand side from those conditions.

The matrix K is generally referred to as a *stiffness matrix*, and is symmetric positive definite (unless only Neumann boundary conditions are imposed, in which case it is symmetric positive semi-definite). The symmetry easily follows from the definition. In order to prove that it is positive definite, we suppose that the basis functions are continuous and bounded, and follow the working in [44, p. 17]: letting $\mathbf{0} \neq \boldsymbol{v} \in \mathbb{R}^{n_x}$, and setting $v_h := \sum_{i=1}^{n_x} v_i\phi_i$, we have

$$\begin{aligned} \boldsymbol{v}^\top K\boldsymbol{v} &= \sum_{i=1}^{n_x}\sum_{l=1}^{n_x} v_i \left(\int_\Omega \nabla\phi_i \cdot \nabla\phi_l \,\mathrm{d}\Omega\right) v_l \\ &= \int_\Omega \left(\sum_{i=1}^{n_x} v_i\nabla\phi_i\right) \cdot \left(\sum_{l=1}^{n_x} v_l\nabla\phi_l\right) \,\mathrm{d}\Omega \\ &= \int_\Omega (\nabla v_h \cdot \nabla v_h) \,\mathrm{d}\Omega \\ &= \|\nabla v_h\|^2_{L^2(\Omega)} \geqslant 0, \end{aligned}$$

where we have used that $\sum_{i=1}^{n_x} v_i \nabla \phi_i = \nabla v_h$. So, K is at least semi-definite; in addition, from the previous expression we have $\boldsymbol{v}^\top K \boldsymbol{v} = 0$ if and only if $\nabla v_h = 0$, that is, if and only if v_h is constant. From $v_h \in V_{0,h}$, we have that v_h is continuous in Ω and it is zero on the boundary $\partial\Omega$, which implies that $v_h = 0$ in Ω. Finally, the latter implies $\boldsymbol{v} = \boldsymbol{0}$.

As we have described the method so far, in order to find a numerical solution for (1.7), from a subspace $V_{0,h}$ of the space V_0 (that contains the solutions of the PDE with zero trace on the boundary) we consider a subspace V_h of V such that

$$V_h = V_{0,h} + \hat{g},$$

with $\hat{g}$ a given function that interpolates the boundary data. A finite element method constructed in this way is referred to as *Galerkin* (or more precisely *Bubnov–Galerkin*) method. In general, one can choose different basis for the subspaces $V_{0,h}$ and V_h, that is to say, test functions and trial functions can be different; the resulting finite element method is referred to as a *Petrov–Galerkin* method. In the following, we will employ both type of methods. It is worth mentioning that also other approaches can be used; for instance in [102] the authors employ the *finite difference method* for discretizing the (spatial) differential operator for solving the optimal control of the heat equation.

1.3 First-Order Optimality Conditions

In this section, we present the strategies one can adopt in order to find a numerical solution of a general PDE-constrained optimization problem of the form (1.2)–(1.3). This solution has to satisfy some *first-order optimality conditions*, known also as *Karush–Kuhn–Tucker (KKT) conditions*.

As a generic solution of (1.2)–(1.3) lies in an infinite-dimensional Hilbert space, and since the differential operator $\mathcal{D}$ is defined on such a space, it is clear that in order to find a numerical solution we will have to discretize the problem at some point. From here, we are given the choice of either take the optimization step first and then apply the discretization, or the other way around. Specifically, we can choose to derive first-order necessary optimality conditions for the problem in the Hilbert space first and then discretize the conditions obtained (optimize-then-discretize strategy), or to discretize the cost functional and the constraint first and then derive the first-order necessary optimality conditions for the discrete optimization probled obtained (discretize-then-optimize strategy). In general, the optimize-then-discretize and the discretize-then-optimize strategies may produce different outcomes, see for example [35]. For most of this thesis, we will adopt the optimize-then-discretize strategy, empoying the discretize-then-optimize approach only in Chapter 7.

Before proceeding with the presentation of the two strategies, we wish to emphasize that the optimility conditions we will derive below are only necessary, that is, a solution of the problem (1.2)–(1.3) has to satisfy them. Although for most of this work those conditions will be also sufficient (as the PDEs we will consider are linear), this is not always the case; for instance, in Chapter 6 we

will consider the optimal control of the non-linear Navier–Stokes equations. In case the differential operator (1.3) is non-linear, one may also consider *second-order optimality conditions* in order to understand if the critical point found is a minimum for the cost functional (1.2). As in this thesis we are only interested in deriving optimal preconditioners for the problem under examination, we will not be deeply concerned as to whether the critical point is a minimum or not, and refer to it as the numerical solution of the problem.

We will now describe the optimize-then-discretize and the discretize-then-optimize strategies. For the sake of exposition, we will only consider the cost functional $J(v, u)$ as defined in (1.1), and we will suppose that the Neumann boundary is empty, that is to say $\partial\Omega_D = \partial\Omega$. In addition, we will employ the Poisson control problem (1.5)–(1.6) as a simple example to explain the ideas below, deriving the discretized optimality conditions for both the optimize-then-discretize and the discretize-then-optimize strategies.

1.3.1 Optimize-Then-Discretize Approach

We will introduce the optimize-then-discretize strategy in this section. As we are seeking optimality conditions in Hilbert spaces, we need to define a differentiation rule that generalizes the classical one on vector spaces to functionals. For more details on this strategy, see for instance [175, Chapter 2].

Let be X_1 and X_2 Banach spaces endowed with norms $\|\cdot\|_{X_1}$ and $\|\cdot\|_{X_2}$ respectively; further, let be U a nonempty open subset of X_1, and $\mathcal{F} : U \to X_2$. Given $u \in U$, $w \in X_1$, and $\varrho > 0$ such that $u + \varrho w \in U$, if the limit

$$\delta\mathcal{F}(u, w) := \lim_{\varrho \to 0^+} \frac{1}{\varrho} \left(\mathcal{F}(u + \varrho w) - \mathcal{F}(u)\right)$$

exists in X_2, then it is called the *directional derivative* of $\mathcal{F}$ at u in the direction w. If this limit exists for all $w \in X_1$, then the map

$$\delta\mathcal{F}(u, \cdot) : w \in X_1 \to \delta\mathcal{F}(u, w) \in X_2$$

is called the *first variation* of $\mathcal{F}$ at u. Supposing that the first variation $\delta\mathcal{F}(u, \cdot)$ at $u \in U$ exists, if it is a bounded linear operator, then $\mathcal{F}$ is said to be *Gâteaux differentiable* at u, and this linear operator is called the *Gâteaux derivative* of $\mathcal{F}$ at u, see for instance [103]. We say that $\mathcal{F}$ is Gâteaux differentiable if $\mathcal{F}$ is Gâteaux differentiable at every $u \in U$.

Another important notion of derivatives in Banach spaces is the *Fréchet derivative*. The mapping $\mathcal{F}$ is said to be *Fréchet differentiable* at u if there exists a bounded linear operator $\mathrm{d}F : X_1 \to X_2$ such that, for every $w \in X_1$ for which $u + w \in U$, we have

$$\frac{\|\mathcal{F}(u + w) - \mathcal{F}(u) - \mathrm{d}Fw\|_{X_2}}{\|w\|_{X_1}} \to 0 \qquad \text{as } \|w\|_{X_1} \to 0.$$

The operator $\mathrm{d}F$ is called the Fréchet derivative of $\mathcal{F}$ at u. If in addition $\mathcal{F}$ is Fréchet differentiable at every $u \in U$, then $\mathcal{F}$ is said to be Fréchet differentiable.

The Fréchet derivative will be the main tool for deriving the first-order optimality conditions for the problems we will consider in this thesis; specifically, we will derive the KKT conditions for the following distributed control problem with PDEs as constraints:

$$\min_{v,u} \quad J(v,u) = \frac{1}{2}\|v - v_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)}$$

subject to

$$\begin{cases} \mathcal{D}v = u + f & \text{in } \Omega, \\ v = g & \text{on } \partial\Omega. \end{cases}$$

We will employ the formal Lagrangian technique for deriving the optimality conditions. In this case, we will introduce the *adjoint variable* or *Lagrange multiplier* ζ (split into interior and Dirichlet components ζ_Ω and $\zeta_{\partial\Omega}$ respectively) and consider the continuous Lagrangian

$$\begin{aligned}\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega}) := \frac{1}{2}\|v - v_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} + (\mathcal{D}v - u - f, \zeta_\Omega) \\ + \int_{\partial\Omega} (v-g)\zeta_{\partial\Omega}\,\mathrm{d}s,\end{aligned}$$

where again $(\cdot,\cdot)$ is the L^2-inner product on Ω. In our derivation, we will assume that the Banach spaces X_1 and X_2 are defined over $\mathbb{R}$. Then, the first-order optimality conditions read as

$$\begin{cases} \mathrm{d}_v\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w = 0, \\ \mathrm{d}_u\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w = 0, \\ \mathrm{d}_\zeta\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w = 0, \end{cases} \tag{1.10}$$

for all $w \in X_1$, where $\mathrm{d}_v\mathcal{L}$, $\mathrm{d}_u\mathcal{L}$, and $\mathrm{d}_\zeta\mathcal{L}$ are the Fréchet derivatives of the Lagrangian $\mathcal{L}$ with respect to v, u, and ζ respectively. In the previous system, the first, the second, and the third equation are called the *adjoint equation*, *gradient equation*, and *state equation* respectively.

As we will see in the following, the first-order optimality conditions (1.10) are a system of coupled PDEs. For the problems we are going to study, it can be rewritten as

$$\begin{cases} \begin{cases} v - v_d + \mathcal{D}^*\zeta_\Omega = 0 & \text{in } \Omega, \\ \zeta_{\partial\Omega} = 0 & \text{on } \partial\Omega, \end{cases} \\ \beta u - \zeta = 0 \qquad \text{in } \Omega, \\ \begin{cases} \mathcal{D}v - u - f = 0 & \text{in } \Omega, \\ v - g = 0 & \text{on } \partial\Omega, \end{cases} \end{cases}$$

where $\mathcal{D}^*$ is the adjoint differential operator of $\mathcal{D}$. Below, we use the Poisson control problem as an example and show how the system above is derived.

It is worth noting that we can eliminate the gradient equation $\beta u - \zeta = 0$ for all the problems we are going to consider. This is because the control u is applied

within the differential operator directly, with no means of another operator $\mathcal{U}$; in the latter case, in fact, we would have read the gradient equation as

$$\beta u - \mathrm{d}_u \mathcal{U} \zeta = 0 \qquad \text{in } \Omega, \tag{1.11}$$

with $\mathrm{d}_u \mathcal{U}$ the Fréchet derivative of $\mathcal{U}$ with respect to u.

As above, the first-order optimality conditions (1.10) are a coupled system of PDEs; as one could imagine, we then need to take the discretization step, so that we may obtain an approximation of the solution by solving the resulting linear system. Thus, letting $M_1^{-1}\mathbf{D}$ and $M_2^{-1}\mathbf{D}^*$ (for appropriate M_1 and M_2) be suitable discretizations of $\mathcal{D}$ and $\mathcal{D}^*$ respectively, a solution of the problem (1.2)–(1.3), with $J(v,u)$ defined as in (1.1), has to satisfy the following

$$\begin{cases} M_v \boldsymbol{v} - \boldsymbol{v}_d + \mathbf{D}^* \boldsymbol{\zeta} = \mathbf{0}, \\ \beta M_u \boldsymbol{u} - M_\zeta \boldsymbol{\zeta} = \mathbf{0}, \\ \mathbf{D}\boldsymbol{v} - M_u \boldsymbol{u} - \boldsymbol{f} = \mathbf{0}, \end{cases}$$

for appropriate M_v, M_u, and M_ζ; the vectors $\boldsymbol{v}$, $\boldsymbol{v}_d$, $\boldsymbol{u}$, $\boldsymbol{\zeta}$, and $\boldsymbol{f}$ contain the numerical approximation of v, v_d, u, ζ, and f respectively, with the vectors $\boldsymbol{v}$ and $\boldsymbol{\zeta}$ also taking into account the boundary conditions on v and ζ.

In order to explain in detail the optimize-then-discretize strategy, we now consider the distributed Poisson control problem (1.5)–(1.6), with $\Omega \subset \mathbb{R}^d$ bounded. As a first step, we have to consider the continuous Lagrangian associated to this problem, and then write the Fréchet derivatives with respect to each variable. The Lagrangian is given by

$$\begin{aligned} \mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega}) := \frac{1}{2}\|v - v_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} + (-\nabla^2 v - u - f, \zeta_\Omega) \\ + \int_{\partial\Omega} (v - g)\zeta_{\partial\Omega} \, \mathrm{d}s. \end{aligned}$$

One of the Banach spaces we will be working in clearly is the Hilbert space $L^2(\Omega)$; in fact, for all the problems considered in this work we have that $\mathcal{L} : L^2(\Omega) \to \mathbb{R}$. In particular, the state, the control, and the adjoint variables lie in a subset of $L^2(\Omega)$. Specifically, for the Poisson control problem the state v belongs to $\mathcal{H}^1(\Omega)$, as it is required to solve the Poisson equation (1.6) in a weak sense. The space to which ζ belongs will be derived below. Finally, the space to which u belongs to is $L^2(\Omega)$.

We now have to consider the Fréchet derivative of $\mathcal{L}$ with respect to all the variables. We start with the Fréchet derivative with respect to u, ζ_Ω, and $\zeta_{\partial\Omega}$, as these are the easiest ones to work with.

In order to write the Fréchet derivative of $\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})$ with respect to u, we will not split the adjoint variable ζ into interior and boundary components. From the definition of the Fréchet derivative, we have to consider a generic direction

$w \in L^2(\Omega)$, and then consider the difference

$$\mathcal{L}(v, u+w, \zeta_\Omega, \zeta_{\partial\Omega}) - \mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega}) = \frac{\beta}{2}\left(\|u+w\|^2_{L^2(\Omega)} - \|u\|^2_{L^2(\Omega)}\right) - (w, \zeta).$$

The first term in the previous expression can be rewritten as

$$\begin{aligned}\|u+w\|^2_{L^2(\Omega)} - \|u\|^2_{L^2(\Omega)} &= \int_\Omega |u+w|^2 \,\mathrm{d}\,\Omega - \int_\Omega |u|^2 \,\mathrm{d}\,\Omega \\ &= \int_\Omega |u|^2 \,\mathrm{d}\,\Omega + \int_\Omega |w|^2 \,\mathrm{d}\,\Omega + 2(u, w) - \int_\Omega |u|^2 \,\mathrm{d}\,\Omega \\ &= \int_\Omega |w|^2 \,\mathrm{d}\,\Omega + 2(u, w).\end{aligned}$$

The latter expression leads to

$$\begin{aligned}\mathcal{L}(v, u+w, \zeta_\Omega, \zeta_{\partial\Omega}) - \mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega}) &= \frac{\beta}{2}\left(\int_\Omega |w|^2 \,\mathrm{d}\,\Omega + 2(u, w)\right) - (w, \zeta) \\ &. = \frac{\beta}{2}\int_\Omega |w|^2 \,\mathrm{d}\,\Omega + (\beta u - \zeta, w) \\ &= \frac{\beta}{2}\|w\|^2_{L^2(\Omega)} + (\beta u - \zeta, w),\end{aligned}$$

due to the symmetry of the $L^2(\Omega)$-inner product. If we write

$$\mathrm{d}_u\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w = (\beta u - \zeta, w),$$

we clearly have

$$\frac{|\mathcal{L}(v, u+w, \zeta_\Omega, \zeta_{\partial\Omega}) - \mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega}) - \mathrm{d}_u\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w|}{\|w\|_{L^2(\Omega)}} = \frac{\beta}{2}\|w\|_{L^2(\Omega)},$$

which tends to 0 as $\|w\|_{L^2(\Omega)} \to 0$. Since the functional $\mathrm{d}_u\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w$ so defined is bounded and linear, we have that it is the Fréchet derivative of $\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})$ with respect to u.

As we wish that

$$\mathrm{d}_u\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w = 0,$$

for all $w \in L^2(\Omega)$, we can infer that, in a strong sense,

$$\beta u - \zeta = 0 \qquad \text{in}\,\Omega.$$

We have derived in this way the gradient equation in (1.10). From here it is clear how we could obtain the expression (1.11) in case the control u is applied by mean of another operator $\mathcal{U}$.

Working in a similar way, we can write the Fréchet derivative of the Lagrangian $\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})$ with respect to ζ; in this case, we will be splitting the adjoint in the interior component ζ_Ω and boundary component $\zeta_{\partial\Omega}$.

Taking $w_\Omega \in L^2(\Omega)$, if we proceed as above we have that

$$\mathrm{d}_{\zeta_\Omega}\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w_\Omega = (-\nabla^2 v - u - f, w_\Omega)$$

is the Fréchet derivative of $\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})$ with respect to ζ_Ω. In addition, by taking $w_{\partial\Omega} \in L^2(\partial\Omega)$ we have that

$$\mathrm{d}_{\zeta_{\partial\Omega}}\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w_{\partial\Omega} = \int_{\partial\Omega}(v-g)w_{\partial\Omega}\,\mathrm{d}s$$

is the Fréchet derivative of $\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})$ with respect to $\zeta_{\partial\Omega}$. Since we want

$$\begin{aligned}\mathrm{d}_{\zeta_\Omega}\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w_\Omega &= 0\\ \mathrm{d}_{\zeta_{\partial\Omega}}\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w_{\partial\Omega} &= 0\end{aligned}$$

for all $w_\Omega \in L^2(\Omega)$ and $w_{\partial\Omega} \in L^2(\partial\Omega)$, in a strong sense we require that

$$\begin{cases} -\nabla^2 v = f + u & \text{in } \Omega,\\ v = g & \text{on } \partial\Omega,\end{cases}$$

which is the state equation in (1.10). Note that in this way we have recovered the constraints (1.6).

We only have to derive the adjoint equation now. This is given by setting the Fréchet derivative of $\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})$ with respect to v equal to 0. In order to do so, we take $w \in \mathcal{H}^1(\Omega)$ and consider the difference

$$\begin{aligned}\mathcal{L}(v+w,u,\zeta_\Omega,\zeta_{\partial\Omega}) - \mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega}) = \frac{1}{2}\left(\|v+w-v_d\|^2_{L^2(\Omega)} - \|v-v_d\|^2_{L^2(\Omega)}\right)\\ + (-\nabla^2 w,\zeta_\Omega) + \int_{\partial\Omega} w\zeta_{\partial\Omega}\,\mathrm{d}s;\end{aligned}$$

here, we made the choice of $w \in \mathcal{H}^1(\Omega)$ to allow the expression $(-\nabla^2 w,\zeta_\Omega)$ to have a suitable meaning. Proceeding as for the derivative with respect to u, we obtain

$$\|v+w-v_d\|^2_{L^2(\Omega)} - \|v-v_d\|^2_{L^2(\Omega)} = \int_\Omega |w|^2\,\mathrm{d}\Omega + 2(v-v_d,w).$$

If we then write

$$\mathrm{d}_v\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w = (v-v_d,w) + (-\nabla^2 w,\zeta_\Omega) + \int_{\partial\Omega} w\zeta_{\partial\Omega}\,\mathrm{d}s,$$

we have[2]

$$\frac{|\mathcal{L}(v+w,u,\zeta_\Omega,\zeta_{\partial\Omega}) - \mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega}) - \mathrm{d}_v\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w|}{\|w\|_{\mathcal{H}^1(\Omega)}} \leqslant \frac{1}{2}\|w\|_{\mathcal{H}^1(\Omega)},$$

which tends to 0 as $\|w\|_{\mathcal{H}^1(\Omega)} \to 0$. In addition, $\mathrm{d}_v\mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega})w$ is bounded,

[2]Note that $\|w\|^2_{L^2(\Omega)} \leqslant \|w\|^2_{\mathcal{H}^1(\Omega)}$.

and from here we establish that it is the Fréchet derivative of $\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})$ with respect to v.

Once we have found the expression for $\mathrm{d}_v\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w$, we have to impose it equal to 0 for all $w \in \mathcal{H}^1(\Omega)$; we will impose it by analysing different cases. Before doing so, we revrite

$$\begin{aligned}\mathrm{d}_v\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w &= (v - v_d, w) + (-\nabla^2 w, \zeta_\Omega) + \int_{\partial\Omega} w\zeta_{\partial\Omega}\,\mathrm{d}s \\ &= (v - v_d, w) + (-\nabla^2\zeta_\Omega, w) - \int_{\partial\Omega} \frac{\partial w}{\partial\vec{n}}\zeta_\Omega\,\mathrm{d}s \\ &\quad + \int_{\partial\Omega} \frac{\partial\zeta_\Omega}{\partial\vec{n}} w\,\mathrm{d}s + \int_{\partial\Omega} w\zeta_{\partial\Omega}\,\mathrm{d}s\end{aligned}$$

by employing the Divergence Theorem[3]. The previous expression has to be 0 for all $w \in \mathcal{H}^1(\Omega)$, in particular for all $w \in C_0^\infty(\Omega)$ with $\dfrac{\partial w}{\partial\vec{n}} = 0$ on $\partial\Omega$, where $C_0^\infty(\Omega)$ is the class of infinitely differentiable functions equal to 0 on the boundary $\partial\Omega$. This choice implies that

$$\mathrm{d}_v\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w = (v - v_d - \nabla^2\zeta_\Omega, w) = 0,$$

which, since $w \in C_0^\infty(\Omega)$ and $C_0^\infty(\Omega)$ is dense in $L^2(\Omega)$, in a strong sense may be rewritten as

$$v - v_d - \nabla^2\zeta_\Omega = 0.$$

If we now choose $w \in C_0^\infty(\Omega)$ with $w = 0$ on $\partial\Omega$ and let $\dfrac{\partial w}{\partial\vec{n}}$ vary, we can derive that

$$\mathrm{d}_v\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w = \int_{\partial\Omega} \frac{\partial w}{\partial\vec{n}}\zeta_\Omega\,\mathrm{d}s = 0$$

has to be satisfied for all such w, and therefore we have

$$\zeta_\Omega = 0 \qquad \text{on } \partial\Omega.$$

Finally, if we choose $w \in C_0^\infty(\Omega)$ with $\dfrac{\partial w}{\partial\vec{n}} = 0$ on $\partial\Omega$ and let w vary, we have

$$\mathrm{d}_v\mathcal{L}(v, u, \zeta_\Omega, \zeta_{\partial\Omega})w = \int_{\partial\Omega} \left(\zeta_{\partial\Omega} + \frac{\partial\zeta_\Omega}{\partial\vec{n}}\right) w\,\mathrm{d}s = 0$$

has to hold for all such w, implying that

$$\zeta_{\partial\Omega} + \frac{\partial\zeta_\Omega}{\partial\vec{n}} = 0 \qquad \text{on } \partial\Omega.$$

We can now write the adjoint equation in (1.10) as

$$\begin{cases} -\nabla^2\zeta + v = v_d & \text{in } \Omega, \\ \zeta = 0 & \text{on } \partial\Omega, \end{cases}$$

[3]Note that $-\zeta_\Omega\nabla^2 w + \nabla\cdot(\zeta_\Omega\nabla w) = \nabla\zeta_\Omega\cdot\nabla w = -w\nabla^2\zeta_\Omega + \nabla\cdot(w\nabla\zeta_\Omega)$.

from which is clear that the adjoint variable ζ has to belong to $\mathcal{H}_0^1(\Omega)$.

The first-order necessary conditions for the Poisson control problem read as

$$\begin{cases} \begin{cases} v - \nabla^2 \zeta = v_d & \text{in } \Omega, \\ \zeta = 0 & \text{on } \partial\Omega, \end{cases} \\ \beta u - \zeta = 0 & \text{in } \Omega, \\ \begin{cases} -\nabla^2 v - u = f & \text{in } \Omega, \\ v = g & \text{on } \partial\Omega. \end{cases} \end{cases} \tag{1.12}$$

We now have to discretize these equations. Before doing so, we wish to mention that, although the derivation of the optimality conditions above is the most rigorous one, in general it is not easy to write the Fréchet derivative of the Lagrangian of a PDE-constrained optimization problem, as this requires a strong knowledge of functional analysis and experience in matching the operators; indeed, one has to properly choose the Banach spaces, has to derive the adjoint equation, and has to find the correct space where the adjoint variables exist.

In order to discretize the first-order optimality conditions so derived, we will employ finite elements, and choose the same basis $\{\phi_1, \phi_2, \ldots, \phi_{n_x}\}$ for the state v, the control u, and the adjoint ζ. Although in principle one could choose different bases for each variable, it is often preferable to use the same basis for all of them for this example, as this will result in a system of convenient structure for which we can eliminate the control variable *a priori*. With this choice of the basis, the weak formulation of (1.12) reads as:

Find $v \in V$, $u \in V_0$, and $\zeta \in V_0$ such that

$$\begin{cases} \displaystyle\int_\Omega vw \,\mathrm{d}\Omega + \int_\Omega \nabla\zeta \cdot \nabla w \,\mathrm{d}\Omega = \int_\Omega v_d w \,\mathrm{d}\Omega & \text{for all } w \in V_0, \\ \displaystyle\int_\Omega \beta uw \,\mathrm{d}\Omega - \int_\Omega \zeta w \,\mathrm{d}\Omega = 0 & \text{for all } w \in V_0, \\ \displaystyle\int_\Omega \nabla v \cdot \nabla w \,\mathrm{d}\Omega - \int_\Omega uw \,\mathrm{d}\Omega = \int_\Omega fw \,\mathrm{d}\Omega & \text{for all } w \in V_0. \end{cases}$$

If we introduce also the functions $\{\phi_{n_x+1}, \phi_{n_x+2}, \ldots, \phi_{n_x+n_\partial}\}$ for interpolating the function g on $\partial\Omega$, we are looking for approximations

$$v \approx \sum_{i=1}^{n_x} v_i \phi_i + \sum_{i=n_x+1}^{n_x+n_\partial} v_i \phi_i, \qquad u \approx \sum_{i=1}^{n_x} u_i \phi_i, \qquad \zeta \approx \sum_{i=1}^{n_x} \zeta_i \phi_i.$$

Then, working as in Section 1.2, the discrete version of the weak formulation above is given by

$$\begin{cases} M\boldsymbol{v} + K\boldsymbol{\zeta} = \hat{\boldsymbol{v}}_d, \\ \beta M\boldsymbol{u} - M\boldsymbol{\zeta} = \boldsymbol{0}, \\ K\boldsymbol{v} - M\boldsymbol{u} = \hat{\boldsymbol{f}}, \end{cases}$$

where the vectors $\boldsymbol{v}$, $\boldsymbol{u}$, and $\boldsymbol{\zeta}$ contain the numerical approximation of v, u, and ζ respectively, with $\hat{\boldsymbol{v}}_d$ and $\hat{\boldsymbol{f}}$ suitable discretizations of the desired state v_d and

of the force function f; note that $\hat{\boldsymbol{v}}_d$ and $\hat{\boldsymbol{f}}$ also contain information about the boundary conditions on v. The matrix K in the system above is the stiffness matrix derived in Section 1.2, while the matrix M is the so called *mass matrix*, defined as

$$M = \{m_{i,l}\}_{i,l=1}^{n_x}, \qquad m_{i,l} = \int_\Omega \phi_i \phi_l \, \mathrm{d}\Omega,$$

and it is symmetric positive definite.

From the discrete gradient equation $\beta M\boldsymbol{u} - M\boldsymbol{\zeta} = \boldsymbol{0}$, we can eliminate the control variable, and rewrite the discrete optimality conditions as

$$\begin{cases} M\boldsymbol{v} + K\boldsymbol{\zeta} = \hat{\boldsymbol{v}}_d, \\ K\boldsymbol{v} - \dfrac{1}{\beta} M\boldsymbol{\zeta} = \hat{\boldsymbol{f}}. \end{cases} \tag{1.13}$$

For the rest of this thesis, we will be devising numerical methods for the solution of systems with this general structure, that lead to systems that are said to be of *saddle-point type*. In the following chapter, we will introduce strategies for solving this type of systems, and show numerically how those strategies are efficient and robust.

1.3.2 Discretize-Then-Optimize Approach

We are going now to describe the discretize-then-optimize strategy. As opposed to the previous strategy, we are first going to discretize the problem, by projecting it onto a finite-dimensional subspace of the space containing the solutions; the resulting minimization problem will be then analysed as a classical optimization problem in $\mathbb{R}^n$, with $n \in \mathbb{N}$ being the dimension of the subspace considered. The first-order optimality conditions will be then derived by making use of classical constrained optimization theory, see for instance [119].

Let consider again the PDE-constrained optimization problem with Dirichlet boundary conditions

$$\min_{v,u} \quad J(v,u) = \frac{1}{2}\|v - v_d\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)}$$

subject to

$$\begin{cases} \mathcal{D}v = f + u & \text{in } \Omega, \\ v = g & \text{on } \partial\Omega. \end{cases}$$

The first thing to do with this strategy is discretize the problem. We will do so by employing finite elements.

Let $\{\phi_i\}_{i=1}^{n_v}$ and $\{\varphi_i\}_{i=1}^{n_u}$ be finite element bases for the spaces containing the state v and the control u respectively. We are looking for approximations of the form

$$v \approx \sum_{i=1}^{n_v} v_i \phi_i + \sum_{i=n_v+1}^{n_v+n_\partial} v_i \phi_i, \qquad u \approx \sum_{i=1}^{n_u} u_i \varphi_i,$$

with $\{\phi_{n_v+1}, \phi_{n_v+2}, \ldots, \phi_{n_v+n_\partial}\}$ interpolating the function g on $\partial\Omega$. In the first part of this section, we will consider different bases for the state v and the control

u, but, as we mentioned for the optimize-then-discretize strategy, it is possible to employ the same finite element bases, and in the second part of this section we will show how this will lead to convenient structure of the discretized system, as we may eliminate the gradient equation *a priori*.

Using the finite element bases above, we may discretize the PDE as follows:

$$\mathbf{D}_v \boldsymbol{v} = \hat{\boldsymbol{f}} + M_{v,u}\boldsymbol{u},$$

where $\mathbf{D}_v$ is the discretized operator $\mathcal{D}$ in the finite element basis for v, and the matrix $M_{v,u} \in \mathbb{R}^{n_v \times n_u}$ is given by

$$M_{v,u} = \{m_{i,l}^{v,u}\}, \qquad m_{i,l}^{v,u} = \int_\Omega \phi_i \varphi_l \, \mathrm{d}\Omega.$$

Note also that the vector $\hat{\boldsymbol{f}}$ contains information about the boundary conditions. In addition, if $\boldsymbol{v}_d$ is the vector containing a suitable discretization of the desired state v_d, we can rewrite the discrete cost functional $J_h(\boldsymbol{v}, \boldsymbol{u})$ as

$$J_h(\boldsymbol{v}, \boldsymbol{u}) = \frac{1}{2}(\boldsymbol{v} - \boldsymbol{v}_d)^\top M_v (\boldsymbol{v} - \boldsymbol{v}_d) + \frac{\beta}{2}\boldsymbol{u}^\top M_u \boldsymbol{u},$$

where M_v is the mass matrix in the finite element basis for v, and M_u is the mass matrix in the finite element basis for u, that is

$$\begin{aligned} M_v &= \{m_{i,l}^v\}_{i,l=1}^{n_v}, \qquad m_{i,l}^v = \int_\Omega \phi_i \phi_l \, \mathrm{d}\Omega, \\ M_u &= \{m_{i,l}^u\}_{i,l=1}^{n_u}, \qquad m_{i,l}^u = \int_\Omega \varphi_i \varphi_l \, \mathrm{d}\Omega. \end{aligned}$$

Finally, the discretized problem we want to solve reads as

$$\min_{\boldsymbol{v},\boldsymbol{u}} \quad J_h(\boldsymbol{v}, \boldsymbol{u}) = \min_{\boldsymbol{v},\boldsymbol{u}} \quad \frac{1}{2}(\boldsymbol{v} - \boldsymbol{v}_d)^\top M_v (\boldsymbol{v} - \boldsymbol{v}_d) + \frac{\beta}{2}\boldsymbol{u}^\top M_u \boldsymbol{u} \tag{1.14}$$

subject to

$$\mathbf{D}_v \boldsymbol{v} = \hat{\boldsymbol{f}} + M_{v,u}\boldsymbol{u}. \tag{1.15}$$

In order to write the first-order optimality conditions, we introduce the adjoint variable $\boldsymbol{\zeta}$, and consider the (discrete) Lagrangian

$$\mathcal{L}_h(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{\zeta}) = \frac{1}{2}(\boldsymbol{v} - \boldsymbol{v}_d)^\top M_v (\boldsymbol{v} - \boldsymbol{v}_d) + \frac{\beta}{2}\boldsymbol{u}^\top M_u \boldsymbol{u} + \boldsymbol{\zeta}^\top (\mathbf{D}_v \boldsymbol{v} - M_{v,u}\boldsymbol{u} - \hat{\boldsymbol{f}}).$$

Then, a critical point $(\boldsymbol{v}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*)$ of (1.14)–(1.15) has to satisfy the following conditions

$$\begin{cases} \dfrac{\partial \mathcal{L}_h}{\partial \boldsymbol{v}}(\boldsymbol{v}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*) = \mathbf{0}, \\ \dfrac{\partial \mathcal{L}_h}{\partial \boldsymbol{u}}(\boldsymbol{v}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*) = \mathbf{0}, \\ \dfrac{\partial \mathcal{L}_h}{\partial \boldsymbol{\zeta}}(\boldsymbol{v}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*) = \mathbf{0}. \end{cases}$$

The previous conditions can be rewritten in term of a linear system as

$$\begin{cases} M_v \boldsymbol{v} + \mathbf{D}_v^\top \boldsymbol{\zeta} = M_v \boldsymbol{v}_d, \\ \beta M_u \boldsymbol{u} - M_{v,u}^\top \boldsymbol{\zeta} = \mathbf{0}, \\ \mathbf{D}_v \boldsymbol{v} - M_{v,u} \boldsymbol{u} = \hat{\boldsymbol{f}}. \end{cases}$$

From the previous system, we immediately realize that choosing the same finite element basis for the state v and the control u will allow us to eliminate the gradient equation $\beta M_u \boldsymbol{u} - M_{v,u}^\top \boldsymbol{\zeta} = \mathbf{0}$, since in this case we have $M_{v,u} = M_{v,u}^\top = M_u$ (which is square). As for the optimize-then-discretize strategy, we may eliminate the gradient equation only because the control u is applied within the differential operator directly, with no means of another operator $\mathcal{U}$. On the other hand, if one employs different finite element basis for the state v and the control u, we have that the matrix $M_{v,u}$ is rectangular. In this case it is preferable to avoid eliminating the gradient equation $\beta M_u \boldsymbol{u} - M_{v,u}^\top \boldsymbol{\zeta} = \mathbf{0}$ (unless applying the inverse operator of the matrix M_u is cheap and feasible), as this would lead to the following system:

$$\begin{cases} M_v \boldsymbol{v} + \mathbf{D}_v^\top \boldsymbol{\zeta} = M_v \boldsymbol{v}_d, \\ \mathbf{D}_v \boldsymbol{v} - \dfrac{1}{\beta} M_{v,u}\, M_u^{-1}\, M_{v,u}^\top \boldsymbol{\zeta} = \hat{\boldsymbol{f}}. \end{cases}$$

In practice, if we eliminate the gradient equation we would have to work with the matrix $M_{v,u}\, M_u^{-1}\, M_{v,u}^\top$, which is not desirable from the linear algebra point of view.

We will give now an example of the first-order optimality conditions derived from the discretize-then-optimize strategy. We will consider again the Poisson control problem (1.5)–(1.6), and employ the same finite element basis $\{\phi_1, \phi_2, \ldots, \phi_{n_x}\}$ for both the state v and the control u.

The discrete version of (1.5)–(1.6) reads as

$$\min_{\boldsymbol{v},\boldsymbol{u}} \quad \frac{1}{2}(\boldsymbol{v} - \boldsymbol{v}_d)^\top M (\boldsymbol{v} - \boldsymbol{v}_d) + \frac{\beta}{2} \boldsymbol{u}^\top M \boldsymbol{u}$$

subject to

$$K\boldsymbol{v} = \hat{\boldsymbol{f}} + M\boldsymbol{u},$$

where K and M are the stiffness and mass matrices in the chosen finite element bases, respectively, and the vector $\hat{\boldsymbol{f}}$ takes into account the boundary condition on v.

By introducing the adjoint variable $\boldsymbol{\zeta}$ and proceeding as above, we derive that a critical point has to satisfy the following conditions:

$$\begin{cases} M\boldsymbol{v} + K\boldsymbol{\zeta} = M\boldsymbol{v}_d, \\ \beta M\boldsymbol{u} - M\boldsymbol{\zeta} = \mathbf{0}, \\ K\boldsymbol{v} - M\boldsymbol{u} = \hat{\boldsymbol{f}}, \end{cases}$$

where we have used the symmetry of K and M. Equivalently, since we can eliminate the gradient equation *a priori*, a solution of (1.5)–(1.6) has to satisfy the system

$$\begin{cases} M\boldsymbol{v} + K\boldsymbol{\zeta} = M\boldsymbol{v}_d, \\ K\boldsymbol{v} - \dfrac{1}{\beta}M\boldsymbol{\zeta} = \hat{\boldsymbol{f}}. \end{cases}$$

It is interesting to note the similarities between the previous system derived by a discretize-then-optimize strategy with the system (1.13) derived employing an optimize-then-discretize strategy. Despite the fact that the two strategies give the same results for Poisson control, one should not jump to the conclusion that this is always true. In fact, in the following chapters we will have to make a choice between the two strategies described so far.

Either by applying an optimize-then-discretize strategy or a discretize-then-optimize approach, in order to find a numerical solution of the PDE-constrained optimization problem under examination we have to solve a (very large) linear system. The following chapter will be devoted to present the numerical methods employed in the rest of this thesis to obtain a numerical solution of the problems we will be tackling. As an example, we will present an optimal solver for the Poisson control problem, and will show how methods of this type are important for the problems under examination.

1.4 State and/or Control Constrained Problems

We would like to spend some time discussing the resolution of distributed PDE-constrained optimization problems with additional algebraic bounds on the state and/or the control variables. In this case, the problem formulation is given by (1.2)–(1.4); we suppose that $\boldsymbol{v}_{\min} < \boldsymbol{v}_{\max}$ and $\boldsymbol{u}_{\min} < \boldsymbol{u}_{\max}$. The additional bound constraints make the problem more difficult to solve than the "unconstrained" problem, as we need to run a non-linear process in order to obtain an approximate solution, even if the PDE constraints are linear. In addition, the solutions may have some regularity issue, as we specify in the following. In order to simplify the exposition, we limit ourself to the case of the discretize-then-optimize strategy. Here and below, we employ the same notation as in Section 1.3.2.

Let us suppose we want to solve the problem (1.2)–(1.4). Adopting the same finite element bases as in Section 1.3.2, the discrete version of the problem under examination is given by

$$\min_{\boldsymbol{v},\boldsymbol{u}} \quad J_h(\boldsymbol{v},\boldsymbol{u}) = \frac{1}{2}(\boldsymbol{v}-\boldsymbol{v}_d)^\top M_v(\boldsymbol{v}-\boldsymbol{v}_d) + \frac{\beta}{2}\boldsymbol{u}^\top M_u \boldsymbol{u} \tag{1.16}$$

subject to

$$\begin{cases} \mathbf{D}_v\boldsymbol{v} = \hat{\boldsymbol{f}} + M_{v,u}\boldsymbol{u}, \\ \boldsymbol{v}_{\min} \leqslant \boldsymbol{v} \leqslant \boldsymbol{v}_{\max}, \\ \boldsymbol{u}_{\min} \leqslant \boldsymbol{u} \leqslant \boldsymbol{u}_{\max}, \end{cases} \tag{1.17}$$

where the relation $\leqslant$ has to be satisfied componentwise, with $\boldsymbol{v}_{\min}$, $\boldsymbol{v}_{\max}$, $\boldsymbol{u}_{\min}$,

and $\boldsymbol{u}_{\max}$ being the vectors containing the numerical approximations of $v_{\min}$, $v_{\max}$, $u_{\min}$, and $u_{\max}$ respectively. Note that (1.16)–(1.17) is a *quadratic programming* (QP) problem.

We can now derive the first-order optimality conditions for this case by introducing the adjoint variable $\boldsymbol{\zeta}$ and the *Lagrange multipliers*

$$\boldsymbol{\lambda} = (\boldsymbol{\lambda}_{v,\min}, \boldsymbol{\lambda}_{v,\max}, \boldsymbol{\lambda}_{u,\min}, \boldsymbol{\lambda}_{u,\max}).$$

By employing a Lagrangian technique as above, the KKT conditions read as [119, Section 12.3]

$$\begin{cases} M_v \boldsymbol{v} + \mathbf{D}_v^\top \boldsymbol{\zeta} - \boldsymbol{\lambda}_{v,\min} + \boldsymbol{\lambda}_{v,\max} = M_v \boldsymbol{v}_d, \\ \beta M_u \boldsymbol{u} - M_{v,u}^\top \boldsymbol{\zeta} - \boldsymbol{\lambda}_{u,\min} + \boldsymbol{\lambda}_{u,\max} = \mathbf{0}, \\ \mathbf{D}_v \boldsymbol{v} - M_{v,u} \boldsymbol{u} = \hat{\boldsymbol{f}}, \\ \boldsymbol{v}_{\min} \leqslant \boldsymbol{v} \leqslant \boldsymbol{v}_{\max}, \quad \boldsymbol{u}_{\min} \leqslant \boldsymbol{u} \leqslant \boldsymbol{u}_{\max}, \\ \boldsymbol{\lambda}_{v,\min} \geqslant \mathbf{0}, \quad \boldsymbol{\lambda}_{v,\max} \geqslant \mathbf{0}, \\ \boldsymbol{\lambda}_{u,\min} \geqslant \mathbf{0}, \quad \boldsymbol{\lambda}_{u,\max} \geqslant \mathbf{0}, \\ \boldsymbol{\lambda}_{v,\min}^\top (\boldsymbol{v} - \boldsymbol{v}_{\min}) = 0, \quad \boldsymbol{\lambda}_{v,\max}^\top (\boldsymbol{v}_{\max} - \boldsymbol{v}) = 0, \\ \boldsymbol{\lambda}_{u,\min}^\top (\boldsymbol{u} - \boldsymbol{u}_{\min}) = 0, \quad \boldsymbol{\lambda}_{u,\max}^\top (\boldsymbol{u}_{\max} - \boldsymbol{u}) = 0. \end{cases} \tag{1.18}$$

From the previous conditions, it is clear that the inequality constraints on the state and the control variables (with the consequential introduction of the Lagrange multipliers) make the problem more difficult to solve, as the state, the gradient, and the adjoint equations have to take into account the influence of the Lagrange multipliers $\boldsymbol{\lambda}$. In addition, a critical point $(\boldsymbol{v}^*, \boldsymbol{u}^*, \boldsymbol{\zeta}^*, \boldsymbol{\lambda}^*)$ for the discrete minimization problem (1.16)–(1.17) has to satisfy the *dual feasibility conditions*

$$\boldsymbol{\lambda}_{v,\min} \geqslant \mathbf{0}, \qquad \boldsymbol{\lambda}_{v,\max} \geqslant \mathbf{0}, \qquad \boldsymbol{\lambda}_{u,\min} \geqslant \mathbf{0}, \qquad \boldsymbol{\lambda}_{u,\max} \geqslant \mathbf{0}, \tag{1.19}$$

and the *complementarity conditions*

$$\begin{aligned} &\boldsymbol{\lambda}_{v,\min}^\top (\boldsymbol{v} - \boldsymbol{v}_{\min}) = 0, \qquad &&\boldsymbol{\lambda}_{v,\max}^\top (\boldsymbol{v}_{\max} - \boldsymbol{v}) = 0, \\ &\boldsymbol{\lambda}_{u,\min}^\top (\boldsymbol{u} - \boldsymbol{u}_{\min}) = 0, \qquad &&\boldsymbol{\lambda}_{u,\max}^\top (\boldsymbol{u}_{\max} - \boldsymbol{u}) = 0. \end{aligned} \tag{1.20}$$

Note that, since each component of the vectors above is non-negative for the bounds on each variable, the previous conditions are equivalent to

$$\begin{aligned} &(\boldsymbol{\lambda}_{v,\min})_i (\boldsymbol{v} - \boldsymbol{v}_{\min})_i = 0, \qquad &&(\boldsymbol{\lambda}_{v,\max})_i (\boldsymbol{v}_{\max} - \boldsymbol{v})_i = 0, \\ &(\boldsymbol{\lambda}_{u,\min})_i (\boldsymbol{u} - \boldsymbol{u}_{\min})_i = 0, \qquad &&(\boldsymbol{\lambda}_{u,\max})_i (\boldsymbol{u}_{\max} - \boldsymbol{u})_i = 0. \end{aligned}$$

Due to their non-linearity, the latter complementarity conditions represent the hardest part to be satisfied in the KKT conditions (1.18). In the literature, various numerical techniques have been devised in order to solve problems of the form (1.18). Below, we will briefly introduce the most widely used methods for PDE-constrained optimization problems, while referring to [119] for a detailed description of the techniques below as well as other methods for solving convex

optimization problems.

1.4.1 Active Set Method

We start by introducing the *active set method* for convex quadratic programming. For an overview on this method, see [119, Section 16.5].

Within an active set method for convex QP, the constraints are split into two disjoint sets: the constraints that belong to the *active set* $\mathcal{I}_{\text{act}}$, and the constraints that belong to the *inactive set* $\mathcal{I}_{\text{inact}}$. The active set is defined as the set of the indices of the constraints that are satisfied as equality at the current approximation of the solution; note that the indices of the equality constraints are always contained in the active set. On the other hand, the inactive set is defined as the complement of the active set in the set of all indices.

The knowledge of the active set $\mathcal{I}^*_{\text{act}}$ at the solution of an optimization problem is of great importance; for instance, in linear programming, one can recover the optimal point by knowing the active set at the solution, as one needs to solve the equality constraints only for the indices in this set. From here, the idea behind an active set method is easy to explain: a method of this type tries to construct the active set $\mathcal{I}^*_{\text{act}}$ at the solution of a minimization problem of the type (1.16)–(1.17) starting from an approximation $\widetilde{\mathcal{I}}_{\text{act}}$. Specifically, given the active set $\widetilde{\mathcal{I}}_{\text{act}}$, the method decides which constraints have to be active and which ones have to be inactive by looking at the dual feasibility conditions (1.19): any constraints related to a Lagrange multiplier that is negative become inactive at the next iteration of the method, and is removed from the active set.

Once the update of the active set is defined, the algorithm is quite straightforward to write: given an approximation $\widetilde{\mathcal{I}}_{\text{act}}$ of the active set at the optimal point, the method solves a quadratic subproblem of (1.16)–(1.17), in which only the constraints with indices in the active set $\widetilde{\mathcal{I}}_{\text{act}}$ are considered; then, the method construct the active set for the next iteration. This process is repeated until some stopping criterion is satisfied.

In order to present the algorithm for (1.16)–(1.17), we decompose the active set $\widetilde{\mathcal{I}}_{\text{act}}$ as

$$\widetilde{\mathcal{I}}_{\text{act}} = \widetilde{\mathcal{I}}^{v,-}_{\text{act}} \cup \widetilde{\mathcal{I}}^{v,+}_{\text{act}} \cup \widetilde{\mathcal{I}}^{u,-}_{\text{act}} \cup \widetilde{\mathcal{I}}^{u,+}_{\text{act}},$$

where $\widetilde{\mathcal{I}}^{v,-}_{\text{act}}$, $\widetilde{\mathcal{I}}^{v,+}_{\text{act}}$, $\widetilde{\mathcal{I}}^{u,-}_{\text{act}}$, and $\widetilde{\mathcal{I}}^{u,+}_{\text{act}}$ are the indices of the constraints for which $\boldsymbol{v} = \boldsymbol{v}_{\min}$, $\boldsymbol{v} = \boldsymbol{v}_{\max}$, $\boldsymbol{u} = \boldsymbol{u}_{\min}$, and $\boldsymbol{u} = \boldsymbol{u}_{\max}$ respectively. In addition, we denote with $\widetilde{\mathcal{I}}_{\text{inact}}$ the current inactive set, which is decomposed as

$$\widetilde{\mathcal{I}}_{\text{inact}} = \widetilde{\mathcal{I}}^{v,-}_{\text{inact}} \cup \widetilde{\mathcal{I}}^{v,+}_{\text{inact}} \cup \widetilde{\mathcal{I}}^{u,-}_{\text{inact}} \cup \widetilde{\mathcal{I}}^{u,+}_{\text{inact}},$$

where $\widetilde{\mathcal{I}}^{v,-}_{\text{inact}}$, $\widetilde{\mathcal{I}}^{v,+}_{\text{inact}}$, $\widetilde{\mathcal{I}}^{u,-}_{\text{inact}}$, and $\widetilde{\mathcal{I}}^{u,+}_{\text{inact}}$ are the indices of the constraints for which $\boldsymbol{v} \neq \boldsymbol{v}_{\min}$, $\boldsymbol{v} \neq \boldsymbol{v}_{\max}$, $\boldsymbol{u} \neq \boldsymbol{u}_{\min}$, and $\boldsymbol{u} \neq \boldsymbol{u}_{\max}$ respectively. Then, it is straightforward to see that the KKT conditions of the quadratic subproblem that are solved at

each iteration are

$$\begin{cases} M_v \boldsymbol{v}^{(k)} + \mathbf{D}_v^\top \boldsymbol{\zeta}^{(k)} - \boldsymbol{\lambda}_{v,\min}^{(k)} + \boldsymbol{\lambda}_{v,\max}^{(k)} = M_v \boldsymbol{v}_d, \\ \beta M_u \boldsymbol{u}^{(k)} - M_{v,u}^\top \boldsymbol{\zeta}^{(k)} - \boldsymbol{\lambda}_{u,\min}^{(k)} + \boldsymbol{\lambda}_{u,\max}^{(k)} = \mathbf{0}, \\ \mathbf{D}_v \boldsymbol{v}^{(k)} - M_{v,u} \boldsymbol{u}^{(k)} = \hat{\boldsymbol{f}}, \end{cases} \tag{1.21}$$

with the i-th components of $\boldsymbol{v}^{(k)}$, $\boldsymbol{u}^{(k)}$, $\boldsymbol{\lambda}_{v,\min}^{(k)}$, $\boldsymbol{\lambda}_{v,\max}^{(k)}$, $\boldsymbol{\lambda}_{u,\min}^{(k)}$, and $\boldsymbol{\lambda}_{u,\max}^{(k)}$ given by

$$\begin{aligned} &(\boldsymbol{v}^{(k)})_i = (\boldsymbol{v}_{\min})_i \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{act}}^{v,-}, && (\boldsymbol{v}^{(k)})_i = (\boldsymbol{v}_{\max})_i \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{act}}^{v,+}, \\ &(\boldsymbol{u}^{(k)})_i = (\boldsymbol{u}_{\min})_i \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{act}}^{u,-}, && (\boldsymbol{u}^{(k)})_i = (\boldsymbol{u}_{\max})_i \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{act}}^{u,+}, \\ &(\boldsymbol{\lambda}_{v,\min}^{(k)})_i = 0 \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{inact}}^{v,-}, && (\boldsymbol{\lambda}_{v,\max}^{(k)})_i = 0 \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{inact}}^{v,+}, \\ &(\boldsymbol{\lambda}_{u,\min}^{(k)})_i = 0 \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{inact}}^{u,-}, && (\boldsymbol{\lambda}_{u,\max}^{(k)})_i = 0 \quad \text{for } i \in \widetilde{\mathcal{I}}_{\text{inact}}^{u,+}. \end{aligned} \tag{1.22}$$

Once the solutions $\boldsymbol{v}^{(k)}$, $\boldsymbol{\zeta}^{(k)}$, $\boldsymbol{u}^{(k)}$, $\boldsymbol{\lambda}_{v,\min}^{(k)}$, $\boldsymbol{\lambda}_{v,\max}^{(k)}$, $\boldsymbol{\lambda}_{u,\min}^{(k)}$, and $\boldsymbol{\lambda}_{u,\max}^{(k)}$ of (1.21)–(1.22) are obtained, the algorithm checks that the Lagrange multipliers are all non-negative. If one of the Lagrange multipliers is negative, the method evaluate the new active set at the current solution. It is worth noting that, if one of the Lagrange multiplier related to a lower bound is non-zero, then the Lagrange multiplier related to the corresponding upper bound is zero, and the other way around. This can be understood from the complementarity conditions (1.20). Suppose in fact that $(\boldsymbol{\lambda}_{v,\min})_i$ is non-zero, for some i, for instance; then, in order for the complementarity conditions (1.20) to be satisfied it has to hold that $(\boldsymbol{v} - \boldsymbol{v}_{\min})_i = 0$, and consequently $(\boldsymbol{\lambda}_{v,\max})_i = 0$ for $(\boldsymbol{v}_{\max} - \boldsymbol{v}_{\min})_i \neq 0$. From here, we can introduce the Lagrange multipliers $\boldsymbol{\lambda}_v$ and $\boldsymbol{\lambda}_u$, one for each of the corresponding variable, as follows:

$$\boldsymbol{\lambda}_v = \boldsymbol{\lambda}_{v,\max} - \boldsymbol{\lambda}_{v,\min}, \qquad \boldsymbol{\lambda}_u = \boldsymbol{\lambda}_{u,\max} - \boldsymbol{\lambda}_{u,\min}. \tag{1.23}$$

Then, we can rewrite the first two equations in (1.21) as follows:

$$\begin{cases} M_v \boldsymbol{v}^{(k)} + \mathbf{D}_v^\top \boldsymbol{\zeta}^{(k)} + \boldsymbol{\lambda}_v^{(k)} = M_v \boldsymbol{v}_d, \\ \beta M_u \boldsymbol{u}^{(k)} - M_{v,u}^\top \boldsymbol{\zeta}^{(k)} + \boldsymbol{\lambda}_u^{(k)} = \mathbf{0}. \end{cases}$$

The active sets are then updated by looking at which constraints are active and then looking at the the sign of the corresponding Lagrange multipliers $(\boldsymbol{\lambda}_v^{(k)})_i$ and $(\boldsymbol{\lambda}_u^{(k)})_i$. This process is repeated until the current active sets are invariant (that is they remain the same between two iterations). The pseudocode of the algorithm for the active set method is given in Algorithm 1.

Although active set methods have proven to be efficient when solving QP problems, when dealing with optimal control of PDEs with additional constraints on the state and/or the control variables those methods have some drawbacks. Specifically, when solving a state-constrained optimal control problem, one can derive from the optimize-then-discretize strategy that the Lagrange multiplier corresponding to the bounds on the state is only a Borel measure (see, for instance, [29]). Below, we are going to introduce a strategy in order to overcome this issue.

Algorithm 1 Active Set Method for Convex QP

Choose $\boldsymbol{v}^{(0)}$, $\boldsymbol{u}^{(0)}$
Find the active set $\widetilde{\mathcal{I}}_{\text{act}}^{(0)}$ at $\boldsymbol{v}^{(0)}$, $\boldsymbol{u}^{(0)}$
for $k = 1$ **until** convergence, **do**
  Solve (1.21)–(1.22) for $\boldsymbol{v}^{(k)}$, $\boldsymbol{\zeta}^{(k)}$, $\boldsymbol{u}^{(k)}$, $\boldsymbol{\lambda}_v^{(k)}$, and $\boldsymbol{\lambda}_u^{(k)}$
  Update the active set $\widetilde{\mathcal{I}}_{\text{act}}^{(k)}$
  Test for convergence: if $\widetilde{\mathcal{I}}_{\text{act}}^{(k)} = \widetilde{\mathcal{I}}_{\text{act}}^{(k-1)}$, then **stop**
end for

1.4.2 Primal-Dual Active Set Method with a Moreau–Yosida Regularization

As we mentioned above, when solving state-constrained optimal control problems, the Lagrange multiplier arising from the optimize-then-discretize strategy is only a Borel measure. Consequently, optimization algorithms have to be adapted in order to deal with those problems. In particular, regularization techniques have been employed in order to overcome the regularity issue of the Lagrange multiplier. In this section, we introduce a *primal-dual* active set method for solving PDE-constrained optimization problems with additional constraints on the state and/or the control variables. Here, we follow the discussion in [81].

Primal-dual active set method are usually employed in conjunction with a *Moreau–Yosida regularization*. The latter allows one to impose the inequality constraints as a nondifferentiable convex function. This function is then employed as a measure to derive the active set at the current iteration.

Suppose we want to solve a problem of the type (1.16)–(1.17). Then, given a constant $c > 0$, the inequality constraints

$$\boldsymbol{v}_{\min} \leqslant \boldsymbol{v} \leqslant \boldsymbol{v}_{\max}, \qquad \boldsymbol{u}_{\min} \leqslant \boldsymbol{u} \leqslant \boldsymbol{u}_{\max},$$

can be equivalently expressed as

$$\begin{aligned}
\boldsymbol{\lambda}_v &= \max(\mathbf{0}, \boldsymbol{\lambda}_v + c(\boldsymbol{v} - \boldsymbol{v}_{\max})) + \min(\mathbf{0}, \boldsymbol{\lambda}_v + c(\boldsymbol{v} - \boldsymbol{v}_{\min})),\\
\boldsymbol{\lambda}_u &= \max(\mathbf{0}, \boldsymbol{\lambda}_u + c(\boldsymbol{u} - \boldsymbol{u}_{\max})) + \min(\mathbf{0}, \boldsymbol{\lambda}_u + c(\boldsymbol{u} - \boldsymbol{u}_{\min})),
\end{aligned}$$

where the Lagrange multipliers $\boldsymbol{\lambda}_v$ and $\boldsymbol{\lambda}_u$ are defined as in (1.23), with the max and the min operator considered componentwise. The conditions above can be derived by employing Moreau–Yosida approximations to nonsmooth convex functions, see [89]. In addition, they can be employed for devising the primal-dual active set method. In fact, we can define the following active sets:

$$\begin{aligned}
\widetilde{\mathcal{I}}_{\text{act}}^{v,-} &= \left\{ i \mid (\boldsymbol{\lambda}_v^{(k)} + c(\boldsymbol{v}^{(k)} - \boldsymbol{v}_{\min}))_i < 0 \right\},\\
\widetilde{\mathcal{I}}_{\text{act}}^{v,+} &= \left\{ i \mid (\boldsymbol{\lambda}_v^{(k)} + c(\boldsymbol{v}^{(k)} - \boldsymbol{v}_{\max}))_i > 0 \right\},\\
\widetilde{\mathcal{I}}_{\text{act}}^{u,-} &= \left\{ i \mid (\boldsymbol{\lambda}_u^{(k)} + c(\boldsymbol{u}^{(k)} - \boldsymbol{u}_{\min}))_i < 0 \right\},\\
\widetilde{\mathcal{I}}_{\text{act}}^{u,+} &= \left\{ i \mid (\boldsymbol{\lambda}_u^{(k)} + c(\boldsymbol{u}^{(k)} - \boldsymbol{u}_{\max}))_i > 0 \right\},
\end{aligned} \tag{1.24}$$

and at each iteration solve (1.21)–(1.22) with this definition of active sets. Then, the new active sets are constructed through (1.24). Finally, this process is repeated until the current active sets are invariant. A pseudocode of the primal-dual active set method so described is given in Algorithm 2.

Algorithm 2 Primal-Dual Active Set Method for PDE-Constrained Optimization Problems

Choose $\boldsymbol{v}^{(0)}$, $\boldsymbol{u}^{(0)}$
Find the active set $\widetilde{\mathcal{I}}_{\text{act}}^{(0)}$ at $\boldsymbol{v}^{(0)}$, $\boldsymbol{u}^{(0)}$ through (1.24)
for $k = 1$ **until** convergence, **do**
  Solve (1.21)–(1.22) for $\boldsymbol{v}^{(k)}$, $\boldsymbol{\zeta}^{(k)}$, $\boldsymbol{u}^{(k)}$, $\boldsymbol{\lambda}_v^{(k)}$, and $\boldsymbol{\lambda}_u^{(k)}$
  Update the active set $\widetilde{\mathcal{I}}_{\text{act}}^{(k)}$ through (1.24)
  Test for convergence: if $\widetilde{\mathcal{I}}_{\text{act}}^{(k)} = \widetilde{\mathcal{I}}_{\text{act}}^{(k-1)}$, then **stop**
end for

The primal-dual active set method described so far has been derived in [81] for the discretize-then-optimize strategy only when (upper) bounds on the control variable are imposed. In this case, the method has been proved to be convergent, and in addition it has been proved to be equivalent to a semismooth Newton method.

Although we followed a discretize-then-optimize strategy, the primal-dual active set method arises naturally from an optimize-then-discretize approach with a Moreau–Yosida regularization technique, see [19, 45, 81]. Also in this case, the primal-dual active set method can be considered as a semismooth Newton method, see [81, 90].

1.4.3 Primal-Dual Interior Point Methods

In this section, we introduce *interior point methods (IPMs)* for solving QP problems of the type (1.16)–(1.17). For an overview of interior point methods for convex QP, see [119, Section 16.6]. For a survey on interior point methods, we recommend [65].

As we mentioned above, the most difficult part of the KKT conditions (1.18) to be solved are the complementarity conditions (1.20). Active set methods try to solve those conditions by dividing the set of the indices into active and inactive sets, and then forcing either the Lagrange multiplier to be zero or the corresponding inequality constraint to be satisfied with equality.

Here, we are going to present a different approach for obtaining a solution of (1.18): the complementarity conditions (1.20) are not imposed to be zero, but rather we solve some perturbed conditions. Interior point methods impose those perturbed conditions by replacing the inequality constraints with a *logarithmic barrier penalty function*, then deriving the first-order optimality conditions of the resulting problem. Due to their non-linearity, the KKT conditions so obtained are then solved by a Newton method, and the new approximation of the critical point is updated in such a way that it is not "too far away" from the *central path*.

Let consider the QP given in (1.16)–(1.17). Given the *barrier term* $\varepsilon > 0$, we can introduce a logarithmic barrier penalty function for each inequality constraint

in the problem, and consider the Lagrangian

$$\begin{aligned}\mathcal{L}_h^{\mathrm{IPM}}(\boldsymbol{v},\boldsymbol{u},\boldsymbol{\zeta}) = \frac{1}{2}(\boldsymbol{v}-\boldsymbol{v}_d)^\top M_v(\boldsymbol{v}-\boldsymbol{v}_d) + \frac{\beta}{2}\boldsymbol{u}^\top M_u\boldsymbol{u} + \boldsymbol{\zeta}^\top(\mathbf{D}_v\boldsymbol{v} - M_{v,u}\boldsymbol{u} - \hat{\boldsymbol{f}}) \\ - \varepsilon\sum_i \log(\boldsymbol{v}-\boldsymbol{v}_{\min})_i - \varepsilon\sum_i \log(\boldsymbol{v}_{\max}-\boldsymbol{v})_i \\ - \varepsilon\sum_i \log(\boldsymbol{u}-\boldsymbol{u}_{\min})_i - \varepsilon\sum_i \log(\boldsymbol{u}_{\max}-\boldsymbol{u})_i.\end{aligned}$$

If we set

$$\begin{aligned}(\boldsymbol{z}_{v,\min})_i &= \varepsilon/(\boldsymbol{v}-\boldsymbol{v}_{\min})_i, & (\boldsymbol{z}_{v,\max})_i &= \varepsilon/(\boldsymbol{v}_{\max}-\boldsymbol{v})_i,\\ (\boldsymbol{z}_{u,\min})_i &= \varepsilon/(\boldsymbol{u}-\boldsymbol{u}_{\min})_i, & (\boldsymbol{z}_{u,\max})_i &= \varepsilon/(\boldsymbol{u}_{\max}-\boldsymbol{u})_i,\end{aligned}$$

then, the first-order optimality conditions read as

$$\begin{cases} M_v\boldsymbol{v} - M_v\boldsymbol{v}_d + \mathbf{D}_v^\top\boldsymbol{\zeta} - \boldsymbol{z}_{v,\min} + \boldsymbol{z}_{v,\max} = \mathbf{0}, \\ \beta M_u\boldsymbol{u} - M_{v,u}^\top\boldsymbol{\zeta} - \boldsymbol{z}_{u,\min} + \boldsymbol{z}_{u,\max} = \mathbf{0}, \\ \mathbf{D}_v\boldsymbol{v} - M_{v,u}\boldsymbol{u} - \hat{\boldsymbol{f}} = \mathbf{0}, \\ \boldsymbol{V}_{\min}\boldsymbol{Z}_{v,\min}\mathbf{1}_v = \varepsilon\mathbf{1}_v, \qquad \boldsymbol{V}_{\max}\boldsymbol{Z}_{v,\max}\mathbf{1}_v = \varepsilon\mathbf{1}_v, \\ \boldsymbol{U}_{\min}\boldsymbol{Z}_{u,\min}\mathbf{1}_u = \varepsilon\mathbf{1}_u, \qquad \boldsymbol{U}_{\max}\boldsymbol{Z}_{u,\max}\mathbf{1}_u = \varepsilon\mathbf{1}_u, \\ \boldsymbol{v}_{\min} \leqslant \boldsymbol{v} \leqslant \boldsymbol{v}_{\max}, \qquad \boldsymbol{u}_{\min} \leqslant \boldsymbol{u} \leqslant \boldsymbol{u}_{\max}, \\ \boldsymbol{z}_{v,\min} \geqslant \mathbf{0}, \qquad \boldsymbol{z}_{v,\max} \geqslant \mathbf{0}, \\ \boldsymbol{z}_{u,\min} \geqslant \mathbf{0}, \qquad \boldsymbol{z}_{u,\max} \geqslant \mathbf{0}, \end{cases} \tag{1.25}$$

where $\mathbf{1}_v$ and $\mathbf{1}_u$ are the vectors of the same size as $\boldsymbol{v}$ and $\boldsymbol{u}$, respectively, with all entries equal to one. Here, $\boldsymbol{V}_{\min}$, $\boldsymbol{V}_{\max}$, $\boldsymbol{U}_{\min}$, and $\boldsymbol{U}_{\max}$ are the diagonal matrices containing the entries of $\boldsymbol{v}-\boldsymbol{v}_{\min}$, $\boldsymbol{v}_{\max}-\boldsymbol{v}$, $\boldsymbol{u}-\boldsymbol{u}_{\min}$, and $\boldsymbol{u}_{\max}-\boldsymbol{u}$, respectively, while $\boldsymbol{Z}_{v,\min}$, $\boldsymbol{Z}_{v,\max}$, $\boldsymbol{Z}_{u,\min}$, and $\boldsymbol{Z}_{u,\max}$ are the diagonal matrices containing the entries of $\boldsymbol{z}_{v,\min}$, $\boldsymbol{z}_{v,\max}$, $\boldsymbol{z}_{u,\min}$, and $\boldsymbol{z}_{u,\max}$, respectively.

From (1.25), we can see that an interior point method does not solve the complementarity conditions (1.20) exactly, but rather the following perturbations:

$$\begin{aligned}(\boldsymbol{z}_{v,\min})_i(\boldsymbol{v}-\boldsymbol{v}_{\min})_i &= \varepsilon, & (\boldsymbol{z}_{v,\max})_i(\boldsymbol{v}_{\max}-\boldsymbol{v})_i &= \varepsilon,\\ (\boldsymbol{z}_{u,\min})_i(\boldsymbol{u}-\boldsymbol{u}_{\min})_i &= \varepsilon, & (\boldsymbol{z}_{u,\max})_i(\boldsymbol{u}_{\max}-\boldsymbol{u})_i &= \varepsilon.\end{aligned} \tag{1.26}$$

The complementarity conditions above are (clearly, although mildly) non-linear. For this reason, in order to find a solution of (1.25), IPMs employ a Newton method: starting from approximations $\boldsymbol{v}^{(k)}$, $\boldsymbol{u}^{(k)}$, $\boldsymbol{\zeta}^{(k)}$, $\boldsymbol{z}_{v,\min}^{(k)}$, $\boldsymbol{z}_{v,\max}^{(k)}$, $\boldsymbol{z}_{u,\min}^{(k)}$, and $\boldsymbol{z}_{u,\max}^{(k)}$ of the solutions $\boldsymbol{v}$, $\boldsymbol{u}$, $\boldsymbol{\zeta}$, $\boldsymbol{z}_{v,\min}$, $\boldsymbol{z}_{v,\max}$, $\boldsymbol{z}_{u,\min}$, and $\boldsymbol{z}_{u,\max}$, respectively, IPMs move in the Newton direction $(\boldsymbol{\delta v}^{(k)}, \boldsymbol{\delta u}^{(k)}, \boldsymbol{\delta\zeta}^{(k)}, \boldsymbol{\delta z}_{v,\min}^{(k)}, \boldsymbol{\delta z}_{v,\max}^{(k)}, \boldsymbol{\delta z}_{u,\min}^{(k)}, \boldsymbol{\delta z}_{u,\max}^{(k)})$ up to a point that satisfies the inequality constraints in (1.25). The Newton

direction is determined by the solution of the following Newton system

$$\begin{bmatrix} M_v & 0 & \mathbf{D}_v^\top & -I_v & I_v & 0 & 0 \\ 0 & \beta M_u & -M_{v,u}^\top & 0 & 0 & -I_u & I_u \\ \mathbf{D}_v & -M_{v,u} & 0 & 0 & 0 & 0 & 0 \\ \boldsymbol{Z}_{v,\min}^{(k)} & 0 & 0 & \boldsymbol{V}_{\min}^{(k)} & 0 & 0 & 0 \\ -\boldsymbol{Z}_{v,\max}^{(k)} & 0 & 0 & 0 & \boldsymbol{V}_{\max}^{(k)} & 0 & 0 \\ 0 & \boldsymbol{Z}_{u,\min}^{(k)} & 0 & 0 & 0 & \boldsymbol{U}_{\min}^{(k)} & 0 \\ 0 & -\boldsymbol{Z}_{u,\max}^{(k)} & 0 & 0 & 0 & 0 & \boldsymbol{U}_{\max}^{(k)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta v}^{(k)} \\ \boldsymbol{\delta u}^{(k)} \\ \boldsymbol{\delta \zeta}^{(k)} \\ \boldsymbol{\delta z}_{v,\min}^{(k)} \\ \boldsymbol{\delta z}_{v,\max}^{(k)} \\ \boldsymbol{\delta z}_{u,\min}^{(k)} \\ \boldsymbol{\delta z}_{u,\max}^{(k)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1^{(k)} \\ \boldsymbol{r}_2^{(k)} \\ \boldsymbol{r}_3^{(k)} \\ \boldsymbol{r}_4^{(k)} \\ \boldsymbol{r}_5^{(k)} \\ \boldsymbol{r}_6^{(k)} \\ \boldsymbol{r}_7^{(k)} \end{bmatrix},$$

where $\boldsymbol{V}_{\min}^{(k)}$, $\boldsymbol{V}_{\max}^{(k)}$, $\boldsymbol{U}_{\min}^{(k)}$, $\boldsymbol{U}_{\max}^{(k)}$, $\boldsymbol{Z}_{v,\min}^{(k)}$, $\boldsymbol{Z}_{v,\max}^{(k)}$, $\boldsymbol{Z}_{u,\min}^{(k)}$, and $\boldsymbol{Z}_{u,\max}^{(k)}$ are the diagonal matrices containing the current approximations of the solutions, with the vectors $\boldsymbol{r}_i^{(k)}$, $i = 1, 2, \ldots, 7$ taking into account for the non-linear residuals. Specifically, we have

$$\begin{aligned} \boldsymbol{r}_1^{(k)} &= M_v \boldsymbol{v}_d - M_v \boldsymbol{v}^{(k)} - \mathbf{D}_v^\top \boldsymbol{\zeta}^{(k)} + \boldsymbol{z}_{v,\min}^{(k)} - \boldsymbol{z}_{v,\max}^{(k)}, \\ \boldsymbol{r}_2^{(k)} &= -\beta M_u \boldsymbol{u}^{(k)} + M_{v,u}^\top \boldsymbol{\zeta}^{(k)} + \boldsymbol{z}_{u,\min}^{(k)} - \boldsymbol{z}_{u,\max}^{(k)}, \\ \boldsymbol{r}_3^{(k)} &= \hat{\boldsymbol{f}} - \mathbf{D}_v \boldsymbol{v}^{(k)} + M_{v,u} \boldsymbol{u}^{(k)}, \\ \boldsymbol{r}_4^{(k)} &= \varepsilon \mathbf{1}_v - \boldsymbol{V}_{\min}^{(k)} \boldsymbol{Z}_{v,\min}^{(k)} \mathbf{1}_v, \\ \boldsymbol{r}_5^{(k)} &= \varepsilon \mathbf{1}_v - \boldsymbol{V}_{\max}^{(k)} \boldsymbol{Z}_{v,\max}^{(k)} \mathbf{1}_v, \\ \boldsymbol{r}_6^{(k)} &= \varepsilon \mathbf{1}_u - \boldsymbol{U}_{\min}^{(k)} \boldsymbol{Z}_{u,\min}^{(k)} \mathbf{1}_u, \\ \boldsymbol{r}_7^{(k)} &= \varepsilon \mathbf{1}_u - \boldsymbol{U}_{\max}^{(k)} \boldsymbol{Z}_{u,\max}^{(k)} \mathbf{1}_u. \end{aligned}$$

Once the method solves the system above, IPMs move in the Newton direction until the inequality constraints in (1.25) are no longer satisfied. Specifically, introducing the stepsize $\bar{\alpha}_P$ and $\bar{\alpha}_D$ as

$$\begin{aligned} \bar{\alpha}_P = \ & \max \left\{ \alpha \,|\, \boldsymbol{v}_{\min} \leqslant \boldsymbol{v}^{(k)} + \alpha \boldsymbol{\delta v}^{(k)} \leqslant \boldsymbol{v}_{\max} \ \wedge \ \boldsymbol{u}_{\min} \leqslant \boldsymbol{u}^{(k)} + \alpha \boldsymbol{\delta u}^{(k)} \leqslant \boldsymbol{u}_{\max} \right\}, \\ \bar{\alpha}_D = \ & \max \Big\{ \alpha \,|\, \boldsymbol{z}_{v,\min}^{(k)} + \alpha \boldsymbol{\delta z}_{v,\min}^{(k)} \geqslant \mathbf{0} \ \wedge \ \boldsymbol{z}_{v,\max}^{(k)} + \alpha \boldsymbol{\delta z}_{v,\max}^{(k)} \geqslant \mathbf{0} \ \wedge \\ & \qquad \boldsymbol{z}_{u,\min}^{(k)} + \alpha \boldsymbol{\delta z}_{u,\min}^{(k)} \geqslant \mathbf{0} \ \wedge \ \boldsymbol{z}_{u,\max}^{(k)} + \alpha \boldsymbol{\delta z}_{u,\max}^{(k)} \geqslant \mathbf{0} \Big\}, \end{aligned}$$

the new approximation of the solution is given by the following:

$$\begin{bmatrix} \boldsymbol{v}^{(k+1)} \\ \boldsymbol{u}^{(k+1)} \\ \boldsymbol{\zeta}^{(k+1)} \\ \boldsymbol{z}_{v,\min}^{(k+1)} \\ \boldsymbol{z}_{v,\max}^{(k+1)} \\ \boldsymbol{z}_{u,\min}^{(k+1)} \\ \boldsymbol{z}_{u,\max}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}^{(k)} + \alpha_P \boldsymbol{\delta v}^{(k)} \\ \boldsymbol{u}^{(k)} + \alpha_P \boldsymbol{\delta u}^{(k)} \\ \boldsymbol{\zeta}^{(k)} + \alpha_D \boldsymbol{\delta \zeta}^{(k)} \\ \boldsymbol{z}_{v,\min}^{(k)} + \alpha_D \boldsymbol{\delta z}_{v,\min}^{(k)} \\ \boldsymbol{z}_{v,\max}^{(k)} + \alpha_D \boldsymbol{\delta z}_{v,\max}^{(k)} \\ \boldsymbol{z}_{u,\min}^{(k)} + \alpha_D \boldsymbol{\delta z}_{u,\min}^{(k)} \\ \boldsymbol{z}_{u,\max}^{(k)} + \alpha_D \boldsymbol{\delta z}_{u,\max}^{(k)} \end{bmatrix},$$

where $\alpha_P = \alpha_0 \bar{\alpha}_P$ and $\alpha_D = \alpha_0 \bar{\alpha}_D$. Here, the parameter $\alpha_0 > 0$ is a positive parameter that is chosen in such a way that the new iterate is not "too close" to the boundary (for example, $\alpha_D = 0.995$), that is to say, the new iterate is still in the *interior* of the feasible region defined by the constraints in (1.25). From here, we can clearly understand the reason behind the name of the method.

At the next step, the barrier parameter ε is reduced by a factor of $\bar{\varepsilon} \in (0, 1)$, in such a way that, in the limit, the sequence will converge to a solution of (1.18). Of course, IPMs do not run until the critical point is reached, but rather stop when some reduction on the primal feasibility, dual feasibility, and optimality tolerance is reached.

As we mention, the name IPMs is assigned as the method evaluates at each iteration a point that is in the interior of the feasible region. The constraints that determine how much the current iterate is in the interior of the feasible region are of course the perturbed complementarity conditions (1.26). In particular, the smaller the parameter ε will be, the closer the iterate will be to the bound constraints. For this reason, the choice of the barrier parameter is critical for the method to progress smoothly. At the beginning of the non-linear process, the values ε is chosen large enough to allow *centrality* (that is, the iterate stays away from the bound constraints); then, the barrier parameter is driven towards zero, allowing the method to converge to the optimal solution. We would like to note that, for every barrier parameter $\varepsilon > 0$, the KKT conditions (1.25) of a convex QP problem have a unique solutions $\boldsymbol{v}^*$, $\boldsymbol{u}^*$, $\boldsymbol{\zeta}^*$, $\boldsymbol{z}^*_{v,\min}$, $\boldsymbol{z}^*_{v,\max}$, $\boldsymbol{z}^*_{u,\min}$, and $\boldsymbol{z}^*_{u,\max}$ that satisfy the inequality constraints in (1.25) strictly. The family of these solutions (that clearly depend on the barrier parameter ε) is called the central path. Convergence of IPMs can be derived providing each iterate stays close to the central path. The latter is ensured by letting the error on the perturbed complementarity conditions (1.26) (measured in some norm of the residual) to be small.

Interior point methods have proven to be a useful tool for solving PDE-constrained optmization problems with additional constraints on the state and/or control variables, especially when advanced numerical linear algebra techniques are employed for the solution of the resulting Newton systems. The majority of the works in the literature employ IPMs in conjunction with the discretize-then-optimize strategy, see, for instance, [18, 134, 135, 137]. We also refer to [18] for a comparison between IPMs and a primal-dual active-set method with Moreau–Yosida regularization. Finally, for a description of interior point methods when an optimize-then-discretize strategy is employed, we point out [161, 177, 183, 184].

Chapter 2

Iterative Solvers for Linear Systems of Equations

"Every day is a new day. It is better to be lucky. But I would rather be exact. Then when luck comes you are ready."
– Ernest Hemingway, *The Old Man and the Sea*

The goal of this thesis is the developement of parameter-robust algorithms to solve linear systems arising from PDE-constrained optimization problems. The two "schools of thought" related to the numerical solution of linear systems are so-called *direct methods*, and *iterative methods*.

On one hand, methods that belong to the first class employ some factorization of the matrix involved as a product of "easier to invert" matrices, and then subsequently solve the linear systems so derived in order to obtain the exact (numerical) solution. Direct methods are quite robust, and the precision of the solution depends mainly on the condition number of the matrix, apart from the machine precision ϵ_{mac}; however, when solving very large problems the computational cost can be prohibitive, and computers can easily run out of memory.

On the other hand, methods belonging to the second class of iterative methods produce a sequence of numerical approximations of the exact solution in such a way that the numerical residual is ideally reduced as the iteration progresses, thus converging to the exact (numerical) solution in the limit (provided that the system is non-singular). At each iteration, these types of methods perform only matrix–vector and vector–vector operations, therefore they are quite cheap to apply; in addition, as long as matrices and vectors can be stored, running out of memory is much less frequent. Finally, iterative methods can produce solutions accurate up to machine precision, although for ill-conditioned problems convergence can be slow.

As the systems considered in this thesis are very large and direct methods suffer from computer memory limitations, in the following description we will focus on iterative methods (apologies, Hemingway: although we want to be exact in our exposition, our methods would rather be iterative).

2.1 Notation

In the following, the matrix $I_{\bar{n}}$ will represent the identity matrix in $\mathbb{R}^{\bar{n}\times\bar{n}}$, for some $\bar{n} \in \mathbb{N}$. Given a matrix $A \in \mathbb{R}^{n_A \times n_A}$ for some $n_A \in \mathbb{N}$, we will write $A \succ 0$ (respectively, $A \succeq 0$) to say that the matrix A is *symmetric positive definite* (respectively, *symmetric positive semi-definite*), meaning that its eigenvalues are real and positive (respectively, real and non-negative). We will indicate with $\lambda(A)$ the set of eigenvalues of A. Further, we will denote with $\rho(A)$ the spectral radius of the matrix A, defined as

$$\rho(A) = \max\{|\lambda| : \lambda \in \lambda(A)\}.$$

Given two matrices $A = [a_{i,l}] \in \mathbb{R}^{n_1 \times n_2}$ and $\bar{A} \in \mathbb{R}^{n_3 \times n_4}$, with $n_1, n_2, n_3, n_4 \in \mathbb{N}$, we will denote with $A \otimes \bar{A}$ the *Kronecker product* of A and $\bar{A}$, defined as the following block matrix:

$$A \otimes \bar{A} = \begin{bmatrix} a_{1,1}\bar{A} & \dots & a_{1,n_2}\bar{A} \\ \vdots & \ddots & \vdots \\ a_{n_1,1}\bar{A} & \dots & a_{n_1,n_2}\bar{A} \end{bmatrix}.$$

Note that $A \otimes \bar{A} \in \mathbb{R}^{n_1 n_3 \times n_2 n_4}$.

For the rest of this thesis we are concerned with the fast solution of linear systems of the form

$$\mathcal{A}\mathbf{x} = \mathbf{b}, \tag{2.1}$$

where $\mathcal{A} \in \mathbb{R}^{n_\mathcal{A} \times n_\mathcal{A}}$ is a square, non-singular matrix, and $\mathbf{b} \in \mathbb{R}^{n_\mathcal{A}}$ is a generic column vector, for some $n_\mathcal{A} \in \mathbb{N}$; we say that $\mathbf{x}$ is the (numerical) solution of the linear system. For general linear systems, the matrix $\mathcal{A}$ has no particular structure; however, for the problems considered in this thesis, the structure of the matrix $\mathcal{A}$ is very specific, and it will be exploited when deriving our solvers. For this reason, we may split the matrix $\mathcal{A}$ into $\bar{m} \times \bar{m}$ blocks, with $\bar{m} \in \mathbb{N}$, as follows:

$$\mathcal{A} = \begin{bmatrix} A_{1,1} & \dots & A_{1,\bar{m}} \\ \vdots & \ddots & \vdots \\ A_{\bar{m},1} & \dots & A_{\bar{m},\bar{m}} \end{bmatrix}, \tag{2.2}$$

where $A_{i,l} \in \mathbb{R}^{n_i \times n_l}$, $i, l = 1, \dots, \bar{m}$; the sizes of each block n_i and n_l are such that $\sum_{i=1}^{\bar{m}} n_i = n_\mathcal{A}$. We emphasize here the case of $\bar{m} = 2$: in this case, it is often possible to derive optimal iterative methods, that require in fact a (roughly) fixed number of iterations for obtaining convergence up to a given tolerance. For those methods, supposing that the matrix $\mathcal{A}$ is sparse, the computational time ideally scales linearly with the size of the system. As a particular case with $\bar{m} = 2$, we will frequently consider the following block matrix:

$$\mathcal{A} = \begin{bmatrix} \Phi & \Psi^\top \\ \Psi & -\Theta \end{bmatrix}, \tag{2.3}$$

with $\Theta \succeq 0$; a matrix $\mathcal{A}$ with this structure is said to be of saddle-point type. In

this case, we split the right-hand side as well as the solution vectors as

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}.$$

2.2 Simple Iteration

Many basic iterative methods fall within the class of the *simple iteration*. The latter can be described as follow. Given the linear system (2.1) with $\mathcal{A}$ as in (2.2), we first define the splitting $\mathcal{A} = \mathcal{P} - \mathcal{N}$ such that $\mathcal{P}$ is invertible; then, given an initial guess $\mathbf{x}^{(0)}$, we define the iterate

$$\mathcal{P}\mathbf{x}^{(j)} = \mathcal{N}\mathbf{x}^{(j-1)} + \mathbf{b}, \qquad j = 1, 2, \dots \tag{2.4}$$

Note that if $\mathbf{x}^{(j)} = \mathbf{x}$ for some j, then $\mathbf{x}^{(j+1)} = \mathbf{x}$.

As we will see, the choice of the matrix $\mathcal{P}$ is very important not only for the convergence of the method, but above all to ensure its fast convergence. From (2.4), it is clear that, in order to find the new iterate $\mathbf{x}^{(j)}$, we need to solve a system with $\mathcal{P}$; thus, the matrix $\mathcal{P}$ has to be chosen in such a way that it is "easy" to apply its inverse to a generic vector, or at least it is not as expensive as inverting $\mathcal{A}$. For instance, one can take $\mathcal{P}$ to be the diagonal of $\mathcal{A}$, obtaining in this way the *Jacobi iteration*; alternatively, one can decide to take the lower triangular part of $\mathcal{A}$, obtaining the *Gauss–Seidel iteration*.

Aside from the computational cost per iteration, the matrix $\mathcal{P}$ determines how fast the error drops per iteration, and therefore the convergence of the method. To see this, we first rewrite (2.4) as

$$\mathbf{x}^{(j)} = (I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})\mathbf{x}^{(j-1)} + \mathcal{P}^{-1}\mathbf{b}. \tag{2.5}$$

The matrix $I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A}$ is referred to as *iteration matrix*. We also rewrite (2.1) as

$$\mathbf{x} = (I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})\mathbf{x} + \mathcal{P}^{-1}\mathbf{b}. \tag{2.6}$$

Denoting with $\mathbf{e}^{(j)} = \mathbf{x} - \mathbf{x}^{(j)}$ the error at the j-th iteration, and subtracting (2.5) from (2.6), we obtain

$$\mathbf{e}^{(j)} = (I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})\mathbf{e}^{(j-1)} = \dots = (I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})^j\mathbf{e}^{(0)}. \tag{2.7}$$

Then, we can derive

$$\|\mathbf{e}^{(j)}\| \leqslant \|(I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})^j\| \, \|\mathbf{e}^{(0)}\|, \tag{2.8}$$

where $\|\cdot\|$ here denotes a vector norm, with a matrix norm induced from a corresponding vector norm defined as

$$\|A\| = \sup_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|.$$

It can be proved that the simple iteration (2.4) converges to the true solution

$\mathbf{x}$ for every initial guess $\mathbf{x}^{(0)}$ if and only if [68, Lemma 2.1.1]

$$\lim_{j\to\infty} \|(I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})^j\| = 0.$$

In an equivalent way, the simple iteration (2.4) is convergent if and only if the spectral radius $\rho(I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A}) < 1$, as we have [68, Corollary 1.3.1]

$$\rho(I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A}) = \lim_{j\to\infty} \|(I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})^j\|.$$

From here, we observe that the choice of the splitting is essential for the convergence of the method; besides, from (2.8) we understand that with an appropriate choice of the matrix $\mathcal{P}$ the error can drop quickly. The few notions presented here can be said to be the most basic idea of preconditioning, which represents the main topic of this thesis. In fact, in the following we will be concerned about finding a matrix $\mathcal{P}$ that is spectrally equivalent to the matrix $\mathcal{A}$ (that is to say, the eigenvalues of the matrix $\mathcal{P}^{-1}\mathcal{A}$ are clustered in a region of the complex plane), with the further property that it is cheap to apply its inverse to a generic vector; in such a way, the iterative process shall converge, and the error or the residual will drop fast enough, with the overall cost depending mainly on matrix–vector and vector–vector operations. A matrix $\mathcal{P}$ that presents these properties is usually refered to as a *preconditioner*.

Different choices of the splitting will give different methods, with different convergence properties. For example, for the matrix $\mathcal{A}$ defined in (2.2), with the choices

$$\mathcal{P}_\mathrm{J} = \begin{bmatrix} A_{1,1} & & 0 \\ & \ddots & \\ 0 & & A_{\bar{m},\bar{m}} \end{bmatrix}, \quad \mathcal{P}_\mathrm{GS} = \begin{bmatrix} A_{1,1} & & 0 \\ \vdots & \ddots & \\ A_{\bar{m},1} & \dots & A_{\bar{m},\bar{m}} \end{bmatrix}$$

one would obtain the *(block-)Jacobi iteration* or the *(block-)Gauss–Seidel iteration*, respectively. As above, if each block is a scalar, we obtain the classical Jacobi or Gauss–Seidel methods, and in this case we will denote the splitting as $\overline{\mathcal{P}}_\mathrm{J}$ and $\overline{\mathcal{P}}_\mathrm{GS}$, respectively. More complex methods arise with different choices of $\mathcal{P}$. In the following section we will introduce one of them: the *Uzawa iteration*.

2.2.1 Uzawa Iteration

Suppose we want to employ a simple iteration to solve (2.1), with $\mathcal{A}$ given in (2.3). We will suppose that $\Phi \succ 0$; we also recall that $\Theta \succeq 0$. Let us consider the following splitting

$$\mathcal{P}_\mathrm{U} = \begin{bmatrix} \Phi & 0 \\ \Psi & -\alpha I \end{bmatrix},$$

where I is the identity matrix of appropriate dimension, and α is a parameter to be determined in order to achieve fast convergence. This choice of the splitting defines the Uzawa method [6, 52], whose iterate is defined as in Algorithm 3, given some initial guess $\mathbf{x}_1^{(0)}$, $\mathbf{x}_2^{(0)}$.

Algorithm 3 Uzawa algorithm

Choose $\mathbf{x}_1^{(0)}$, $\mathbf{x}_2^{(0)}$
for $j = 1$ **until** convergence, **do**
  Solve $\Phi \mathbf{x}_1^{(j)} = \mathbf{b}_1 - \Psi^\top \mathbf{x}_2^{(j-1)}$
  Compute $\mathbf{x}_2^{(j)} = \mathbf{x}_2^{(j-1)} + \frac{1}{\alpha}\left(\Psi \mathbf{x}_1^{(j)} - \Theta \mathbf{x}_2^{(j-1)} - \mathbf{b}_2\right)$
end for

If from Algorithm 3 we substitute the expression of $\mathbf{x}_1^{(j)}$ into $\mathbf{x}_2^{(j)}$, we obtain

$$\mathbf{x}_2^{(j)} = \mathbf{x}_2^{(j-1)} + \frac{1}{\alpha}\left(\Psi\Phi^{-1}\mathbf{b}_1 - S\mathbf{x}_2^{(j-1)} - \mathbf{b}_2\right),$$

where $S = \Psi\Phi^{-1}\Psi^\top + \Theta$ is the *(negative) Schur complement* of the matrix $\mathcal{A}$. Having a good knowledge of the spectrum of S is of fundamental importance for tuning the parameter α, as the correct choice of the latter will result in fast convergence of the method. In fact, supposing that $\lambda(S) \in [\lambda_{\min}, \lambda_{\max}]$, it can be proved that the optimal value of α is $(\lambda_{\min} + \lambda_{\max})/2$ [43].

From Algorithm 3, it is clear that, for any given α, the bulk of the work involves solving a system with the matrix Φ. For many problems, a solve for the $(1,1)$-block Φ can be quite expensive (if feasible and no run out of memory occurs); for this reason, we prefer to consider a solve with some (easily invertible) approximation $\widehat{\Phi}$ of Φ. The splitting in this case is given by

$$\mathcal{P}_{\mathrm{IU}} = \begin{bmatrix} \widehat{\Phi} & 0 \\ \Psi & -\alpha I \end{bmatrix},$$

while the corresponding iterate is defined as in Algorithm 4. This method is referred to as *inexact Uzawa*, and it has been proved to be convergent provided that the approximation $\widehat{\Phi}$ of Φ is good enough (we refer to [43] for a detailed justification).

Algorithm 4 Inexact Uzawa algorithm

Choose $\mathbf{x}_1^{(0)}$, $\mathbf{x}_2^{(0)}$
for $j = 1$ **until** convergence, **do**
  $\mathbf{x}_1^{(j)} = \mathbf{x}_1^{(j-1)} + \widehat{\Phi}^{-1}\left(\mathbf{b}_1 - \Phi \mathbf{x}_1^{(j-1)} - \Psi^\top \mathbf{x}_2^{(j-1)}\right)$
  $\mathbf{x}_2^{(j)} = \mathbf{x}_2^{(j-1)} + \frac{1}{\alpha}\left(\Psi \mathbf{x}_1^{(j)} - \Theta \mathbf{x}_2^{(j-1)} - \mathbf{b}_2\right)$
end for

As noted above, a suitable choice of α will result in fast convergence of the method; this choice can be made by knowing the spectrum of the Schur complement S, but in many applications this is not often practical. However, we can speed up the convergence of the method by finding an approximation $\widehat{S}$ of the Schur complement S. As for the $(1,1)$-block, we would like the matrix $\widehat{S}$ to be

easily invertible. With this in mind, we consider the splitting

$$\mathcal{P}_{\text{PU}} = \begin{bmatrix} \Phi & 0 \\ \Psi & -\widehat{S} \end{bmatrix},$$

and define the simple iteration as in Algorithm 5; the resulting method is referred to as *preconditioned Uzawa*. Note that we have absorbed the parameter α within our approximation $\widehat{S}$. As for the inexact Uzawa method, assuming that $\widehat{S}$ approximates well the Schur complement S, one can derive convergence of the method [43].

Algorithm 5 Preconditioned Uzawa algorithm

Choose $\mathbf{x}_1^{(0)}$, $\mathbf{x}_2^{(0)}$
for $j = 1$ **until** convergence, **do**
  Solve $\Phi \mathbf{x}_1^{(j)} = \mathbf{b}_1 - \Psi^\top \mathbf{x}_2^{(j-1)}$
  Compute $\mathbf{x}_2^{(j)} = \mathbf{x}_2^{(j-1)} + \widehat{S}^{-1}\left(\Psi \mathbf{x}_1^{(j)} - \Theta \mathbf{x}_2^{(j-1)} - \mathbf{b}_2\right)$
end for

Even in this case, inverting the matrix Φ results in the most expensive computational task; for this reason, an inexact version of the preconditioned Uzawa algorithm has been widely studied, see [24, 43, 189]. The inexact version of Algorithm 5 is defined by the splitting

$$\mathcal{P}_{\text{IPU}} = \begin{bmatrix} \widehat{\Phi} & 0 \\ \Psi & -\widehat{S} \end{bmatrix}.$$

In this case, we write the simple iteration as in Algorithm 6. Inexact preconditioned Uzawa has been proved to be convergent, with the error per-iteration depending on the spectral properties of the matrices $\widehat{\Phi}^{-1}\Phi$ and $\widehat{S}^{-1}S$, see [24, 43, 189] and [151, Section 4.1].

Algorithm 6 Inexact preconditioned Uzawa algorithm

Choose $\mathbf{x}_1^{(0)}$, $\mathbf{x}_2^{(0)}$
for $j = 1$ **until** convergence, **do**
  $\mathbf{x}_1^{(j)} = \mathbf{x}_1^{(j-1)} + \widehat{\Phi}^{-1}\left(\mathbf{b}_1 - \Phi \mathbf{x}_1^{(j-1)} - \Psi^\top \mathbf{x}_2^{(j-1)}\right)$
  $\mathbf{x}_2^{(j)} = \mathbf{x}_2^{(j-1)} + \widehat{S}^{-1}\left(\Psi \mathbf{x}_1^{(j)} - \Theta \mathbf{x}_2^{(j-1)} - \mathbf{b}_2\right)$
end for

2.3 Relaxation Methods

Let us consider again the simple iteration (2.4). Defining the *residual* $\mathbf{r}^{(j)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(j)}$, we can rewrite (2.5) as

$$\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \mathcal{P}^{-1}\mathbf{r}^{(j-1)}.$$

We note that, from the definition of the residual, we have $\mathcal{A}\mathbf{e}^{(j)} = \mathbf{r}^{(j)}$. As done for the Uzawa iteration, we can introduce a splitting $\mathcal{P}_\alpha$ depending on some parameter α in order to accelerate convergence, and consider the following iteration

$$\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \mathcal{P}_\alpha^{-1}\mathbf{r}^{(j-1)}.$$

A method of this type is called a *relaxation method*. As in Section 2.2, we choose $\mathcal{P}_\alpha$ in such a way that it is easy to apply its inverse to a vector, but it also approximates $\mathcal{A}$ in some sense. For instance, if we make the cheapest choice $\mathcal{P}_\alpha = \frac{1}{\alpha} I_{n_\mathcal{A}}$, we obtain

$$\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \alpha\mathbf{r}^{(j-1)},$$

which is called the *Richardson iteration* [154]. As in (2.7), the error at each step is given by

$$\mathbf{e}^{(j)} = (I_{n_\mathcal{A}} - \alpha\mathcal{A})\,\mathbf{e}^{(j-1)},$$

which implies

$$\|\mathbf{e}^{(j)}\| \leqslant \|I_{n_\mathcal{A}} - \alpha\mathcal{A}\|\,\|\mathbf{e}^{(j-1)}\|,$$

with $\|\cdot\|$ again defining a generic vector norm with corresponding induced matrix norm defined as above. If we suppose that $\|I_{n_\mathcal{A}} - \alpha\mathcal{A}\| < 1$, then the method converges. It is not easy to find an optimal value for α, as the convergence depends on the vector norm we are considering, and in general this would require some *a priori* information about the spectrum of $\mathcal{A}$. Let us suppose that in fact the matrix $\mathcal{A}$ is symmetric positive definite, with eigenvalues λ_i, $i = 1, 2, \ldots, n_\mathcal{A}$, and we want to minimize the error in the 2-norm. Then, we have $\|I_{n_\mathcal{A}} - \alpha\mathcal{A}\|_2 = \max_i |1 - \alpha\lambda_i|$. Denoting with $\lambda_{\max}$ and $\lambda_{\min}$ the maximum and the minimum eigenvalue of $\mathcal{A}$ respectively, we have $\|I_{n_\mathcal{A}} - \alpha\mathcal{A}\|_2 < 1$ provided that $0 < \alpha < \frac{2}{\lambda_{\max}}$. Further, the optimal choice of α is given by

$$\alpha = \frac{2}{\lambda_{\max} + \lambda_{\min}}. \tag{2.9}$$

Adopting this choice of α, we have that

$$\|I_{n_\mathcal{A}} - \alpha\mathcal{A}\|_2 = 1 - \frac{2}{\kappa + 1} = \frac{\kappa - 1}{\kappa + 1},$$

with $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$ the *condition number* of $\mathcal{A}$, due to $\mathcal{A} \succ 0$.

Other choices of $\mathcal{P}_\alpha$ are possible. For instance, we can take $\mathcal{P}_\alpha = \frac{1}{\alpha}\overline{\mathcal{P}}_\mathrm{J}$ to be a multiple of the diagonal of the matrix $\mathcal{A}$. In this case, if we choose $\alpha = 1$, we obtain the classical Jacobi iteration, while with different values we obtain the so called *relaxed Jacobi iteration*. If $\mathcal{A} \succ 0$, proceding as above we can derive that the optimal value for α is given by (2.9), with $\lambda_{\max}$ and $\lambda_{\min}$ the maximum and the minimum eigenvalue of the matrix $\overline{\mathcal{P}}_\mathrm{J}^{-1}\mathcal{A}$.

In a similar way, we can employ a relaxed version of the Gauss–Seidel iteration. In this case, we take $\mathcal{P}_\alpha = \alpha\overline{\mathcal{P}}_\mathrm{GS} + (1-\alpha)\overline{\mathcal{P}}_\mathrm{J}$; this splitting defines the method of *Successive Over Relaxation (SOR)*. As for the relaxed Jacobi method, if $\alpha = 1$ we obtain the classical Gauss–Seidel iteration. Note that in this case the iteration matrix is not symmetric. If we suppose that the matrix $\mathcal{A}$ is symmetric, we

can modify the SOR method in order to make the iteration matrix symmetric, by making the choice $\mathcal{P}_\alpha = \frac{1}{2-\alpha}\left(\alpha\overline{\mathcal{P}}_{\mathrm{GS}} + (1-\alpha)\overline{\mathcal{P}}_{\mathrm{J}}\right)(\alpha\overline{\mathcal{P}}_{\mathrm{J}})^{-1}\left(\alpha\overline{\mathcal{P}}_{\mathrm{GS}} + (1-\alpha)\overline{\mathcal{P}}_{\mathrm{J}}\right)^\top$. This method is called the *Symmetric Successive Over Relaxation (SSOR)* method. For more information on those methods, see, for example, [62] or [178].

2.4 Chebyshev Semi-Iteration

Another way to accelerate the convergence of the simple iteration is employing Chebyshev polynomials. Let consider again the simple interation (2.4), and suppose we have generated the iterates $\mathbf{x}^{(0)}$, $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(j)}$. We seek an improvement of the current solution $\mathbf{x}^{(j)}$ of the form

$$\mathbf{x}^{(j)}_{\mathrm{cheb}} = \sum_{i=0}^{j} c_i^{(j)}\,\mathbf{x}^{(i)}, \tag{2.10}$$

with the coefficients $c_i^{(j)}$, $i = 0, 1, \ldots, j$ to be determined. Associated to those coefficients, we will considering the polynomial

$$p_j(z) = \sum_{i=0}^{j} c_i^{(j)} z^i \in \Pi_j,$$

with Π_j the space of polynomials of degree at most j. As noted in Section 2.2, if the initial guess $\mathbf{x}^{(0)}$ is equal to the exact solution $\mathbf{x}$, then $\mathbf{x}^{(i)} = \mathbf{x}$ for all i. Then, it would be reasonable to have also $\mathbf{x}^{(j)}_{\mathrm{cheb}} = \mathbf{x}$. For this to happen, we require that $p_j(1) = 1$. With this choice of $p_j(z)$, from (2.7) we can now write the error as

$$\mathbf{x} - \mathbf{x}^{(j)}_{\mathrm{cheb}} = \sum_{i=0}^{j} c_i^{(j)}(\mathbf{x} - \mathbf{x}^{(i)}) = \sum_{i=0}^{j} c_i^{(j)}(I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})^i\,\mathbf{e}^{(0)} = p_j(\mathcal{G})\,\mathbf{e}^{(0)},$$

with $\mathcal{G} = I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A}$. By taking the norm in the expression above, we obtain

$$\|\mathbf{x} - \mathbf{x}^{(j)}_{\mathrm{cheb}}\| \leqslant \|p_j(\mathcal{G})\|\,\|\mathbf{e}^{(0)}\|;$$

thus, we can have a reduction in the error if we find a polynomial $p_j(z)$ of degree at most j such that $p_j(1) = 1$ and it minimizes the norm $\|p_j(\mathcal{G})\|$.

For simplifying the analysis, we will suppose that we want to minimize the 2-norm $\|\cdot\|_2$ and that the matrix $\mathcal{G}$ is symmetric. Since $\mathcal{G}$ is symmetric, we can write $\mathcal{G} = Q\Lambda Q^\top$, with Q an orthogonal matrix (its columns form an orthonormal basis in $\mathbb{R}^{n_\mathcal{A}}$), and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{n_\mathcal{A}})$ a diagonal matrix whose entries are the eigenvalues of $\mathcal{G}$. We first suppose that

$$-1 < a \leqslant \lambda_{n_\mathcal{A}} \leqslant \ldots \leqslant \lambda_2 \leqslant \lambda_1 \leqslant b < 1,$$

where a and b are known estimates. From the orthogonality of Q, we have that

$$\mathcal{G}^i = (Q\,\Lambda\,Q^\top)^i = Q\,\Lambda^i\,Q^\top,$$

that in turns implies

$$p_j(\mathcal{G}) = \sum_{i=0}^{j} c_i^{(j)} Q\,\Lambda^i\,Q^\top = Q\,p_j(\Lambda)\,Q^\top,$$

where $p_j(\Lambda) = \mathrm{diag}(p_j(\lambda_1), p_j(\lambda_2), \ldots, p_j(\lambda_{n_\mathcal{A}}))$. Using again the orthogonality of Q and recalling that we are minimizing the 2-norm, we can write

$$\|p_j(\mathcal{G})\|_2 = \|Q\,p_j(\Lambda)\,Q^\top\|_2 = \|p_j(\Lambda)\|_2 = \max_{\lambda_i \in \lambda(\mathcal{G})} |p_j(\lambda_i)| \leqslant \max_{a \leqslant \lambda \leqslant b} |p_j(\lambda)|.$$

From here, the degree j polynomial we are looking for has also the convenient property that the maximum absolute value it attains in the interval $[a, b]$ is minimal, and it is such that $p_j(1) = 1$. This polynomial is proved to be the Chebyshev polynomial (of the first kind) of degree j, scaled by an appropriate factor [108, Corollary 3.4B].

The Chebyshev polynomial $T_j(z)$ (of the first kind) is a polynomial in z of degree j, defined for $z \in [-1, 1]$ as

$$T_j(z) = \cos j\theta, \quad \text{where } z = \cos\theta.$$

The above formula defines a polynomial of degree j in $\cos\theta$, and, by employing the trigonometric identity

$$\cos j\theta + \cos(j-2)\theta = 2\cos\theta\,\cos(j-1)\theta,$$

can be shown to satisfy the following recurrence relation:

$$T_j(z) = 2zT_{j-1}(z) - T_{j-2}(z), \quad j = 2,\, 3, \ldots \tag{2.11}$$

with the initial conditions $T_0(z) = 1$ and $T_1(z) = z$. We can also define Chebyshev polynomials on a general interval $[\bar{a}, \bar{b}]$, by making use of the affine transformation

$$s = \frac{2z - (\bar{a} + \bar{b})}{\bar{b} - \bar{a}},$$

and defining the scaled Chebyshev polynomial of degree j (of the first kind) as

$$\overline{T}_j(z) = T_j(s).$$

Note the special case of $[\bar{a}, \bar{b}] \equiv [0, 1]$, for which we have $s = 2z - 1$ and we obtain the so called shifted Chebyshev polynomial of the first kind.

For our analysis, as it is not always true that the eigenvalues of $\mathcal{G}$ lie in the interval $[-1, 1]$, we consider the shifted and scaled Chebyshev polynomial of degree j:

$$\widehat{T}_j(z) := \frac{T_j(\frac{2z-(a+b)}{b-a})}{T_j(\frac{2-(a+b)}{b-a})}.$$

With this choice of $p_j(\lambda)$, we have $p_j(1) = 1$ and

$$\max_{a \leqslant \lambda \leqslant b} |p_j(\lambda)| = \frac{1}{\left|T_j(\frac{2-(a+b)}{b-a})\right|},$$

and therefore the larger the value of $\frac{2-(a+b)}{b-a}$ is, the faster the convergence of the iterative method will be. This choice of $p_j(\lambda)$ defines the *Chebyshev semi-iterative method* [63, 64]. Supposing that $\mathcal{G}$ is symmetric (that is equivalent to say that the matrix $\mathcal{P}^{-1}\mathcal{A}$ is symmetric), it can be proved that the reduction of the error at the j-th iteration is given by

$$\|\mathbf{x} - \mathbf{x}_{\text{cheb}}^{(j)}\|_2 \leqslant 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^j \|\mathbf{x} - \mathbf{x}^{(0)}\|_2,$$

where κ is the condition number of $\mathcal{P}^{-1}\mathcal{A}$, see [151, Section 3.1.4].

As we have defined the process so far, at each iteration j we need to evaluate the coefficients of the Chebyshev polynomial $\widehat{T}_j(z)$, and then evaluate the new approximation $\mathbf{x}_{\text{cheb}}^{(j)}$ defined in (2.10), and it is clear that this way of proceeding is not efficient as we would have to store all the approximations $\mathbf{x}^{(i)}$, $i = 0, 1, \ldots, j$. Fortunately, it is possible to derive a three-term recurrence formula by exploiting (2.11); the more efficient algorithm derived is given in Algorithm 7, supposing that $\lambda(\mathcal{G}) \in [a, b]$, as presented in [115, Section 5.8].

Algorithm 7 Chebyshev semi-iteration to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$

Given lower and upper bounds a and b for the spectrum $\lambda(\mathcal{G})$
Choose $\mathbf{x}^{(0)}$, set $\mathbf{x}^{(-1)} = \mathbf{0}$
$\omega_0 = 0$, $\omega = \frac{2-(a+b)}{b-a}$
for $j = 0$ **until** convergence, **do**
  if $j = 1$ **then**
    $\omega_{j+1} = (1 - \frac{1}{2\omega^2})^{-1}$
  else
    $\omega_{j+1} = (1 - \frac{\omega_j^2}{4\omega^2})^{-1}$
  end if
  Solve $\mathcal{P}\mathbf{z}^{(j)} = \mathbf{b} - \mathcal{A}\mathbf{x}_{\text{cheb}}^{(j)}$
  $\mathbf{x}_{\text{cheb}}^{(j+1)} = \omega_{j+1}\left(\frac{2}{2-(a+b)}\mathbf{z}^{(j)} + \mathbf{x}_{\text{cheb}}^{(j)} - \mathbf{x}_{\text{cheb}}^{(j-1)}\right) + \mathbf{x}_{\text{cheb}}^{(j-1)}$
end for

The analysis carried out so far was assuming that the iteration matrix $\mathcal{G}$ is symmetric. In [106, 107], the Chebyshev semi-iteration has been extended also to non-symmetric matrices, provided that convex hull containing all the eigenvalues is known. For more information on Chebyshev semi-iteration, see [62, Section 11.2.8] and [178, Chapter 5].

As it should be clear now, for the method to be effective we need good knowledge of the spectrum of the iteration matrix. Although this is sometimes difficult to accomplish, in some special cases the Chebyshev semi-iteration is well suited. In fact, as described in [181], the Chebyshev semi-iteration is a linear method,

meaning that the polynomial $p_j(z)$ adopted at each iteration j is independent of the right hand side vector $\mathbf{b}$ and the initial guess $\mathbf{x}^{(0)}$. This, together with the fast convergence property derived from a good knowledge of the spectrum of the iteration matrix $\mathcal{G}$, makes the Chebyshev semi-iteration a potent part of the preconditioners for the problems considered in this thesis. In particular, the Chebyshev semi-iteration will be used below for accelerating the convergence of the relaxed Jacobi applied for inverting mass matrices, with $\mathcal{P}$ corresponding to the diagonal of the mass matrix. In fact, in [179] the author provided lower and upper bounds of the spectrum of the Jacobi iteration matrix applied to mass matrices arising from any finite element method; those values have been proved to be independent of the mesh size, and depend only on the type of element employed. For instance, for $\mathbf{Q}_1$ finite elements mass matrices in 2D it results that $\lambda(\mathcal{G}) \in [\frac{1}{4}, \frac{9}{4}]$; this information may be utilized to pre-specify a and b within the Chebyshev semi-iterative scheme.

2.5 Multigrid Methods

In the previous sections we have introduced the simple iteration and its acceleration for solving linear systems of the type (2.1). In order to work effectively, those methods require, apart from the matrix $\mathcal{A}$ being invertible and in some cases symmetric, some knowledge of the spectrum of (the whole, or some blocks of) the iteration matrix $(I_{n_\mathcal{A}} - \mathcal{P}^{-1}\mathcal{A})$. In this section, we introduce another class of methods that fall within the category of iterative methods for linear systems, and that will be frequently used in the following chapters for approximating some blocks of our systems: *Multigrid (MG) methods*. For a more detailed discussion on multigrid techniques, we refer the reader to [27, 75, 185].

Multigrid methods works quite well as iterative methods in their own right, but in general require that the linear system being solved is sparse, an assumption that we will make for the rest of this section. Despite this very desirable property, the potential of these methods for solving sparse linear systems also lies in their use as preconditioners. When designed correctly, multigrid methods can be used as "black-box" preconditioners, and in general require no prior knowledge on the spectrum of the iteration matrix. In fact, multigrid is based more on the "geometry" defined by the matrix $\mathcal{A}$ rather than its spectrum: the method constructs, as the name suggests, a sequence of (hierarchical) grids in order to evaluate an approximation of the solution. Here, we put the word geometry between quotation marks, as we refer to the grid (or, more appropriately, the graph) represented by the matrix $\mathcal{A}$. The idea of constructing this hierarchy of grids is based on the observation that a few steps of a simple iteration are capable of reducing only the *high-frequency* components of the solution error; thus, in order to reduce the *low-frequency* errors, one projects the current error onto a coarser grid, and then applies again a simple iteration. This process is repeated up to a grid coarse enough to allow a direct solver to determine the error on this level. Finally, one projects the solution so obtained back from the coarsest to the finest grid, and updates the solution on each grid.

The main components of a multigrid routine are a (*pre-* and *post-*)*smoothing*

operator, a *prolongation operator*, and a *restriction operator*. The smoothing operator is usually given by a (fixed number of steps of a) simple iteration, like Jacobi, Gauss–Seidel, or their relaxation; as mentioned above, it is employed to reduce the high-frequency components of the error. Although the smoothing operator is fairly ubiquitous in multigrid methods, what really defines such a routine is the way one projects the error onto the coarser grids (restriction), and then how the solution is recovered on the finer grid (prolongation). From here, one can see restriction and prolongation as one being the adjoint operator of the other; in fact, usually the matrix R that defines the restriction is chosen to be the transpose of the matrix P that defines the projection. Of course, different choices are possible, but the main multigrid routines (included the ones used in this work) set $R = P^\top$, so we will focus our description on this choice.

In order to define the multigrid scheme we need to define the prolongation P. This is done by selecting the nodes that will form the coarse grid from a given fine grid. A possible approach is to exploit the geometry of the grid under examination, decide which nodes will become part of the coarse grid, and then appropriately weight each node. This approach defines a *Geometric Multigrid (GMG)* method, and it was devised for solving linear systems arising from the discretization of a differential operator [25]. Although the implementation of GMG can be quite straightforward for simple problems on structured grids, the main drawback of this method is that it requires a detailed knowledge of the domain, and this is difficult to obtain for general problems. For instance, for industrial problems the domain can be very complex, with internal boundaries, and unstructured meshes are usually employed. Therefore, of late much research has moved to devising multigrid techniques that overcome these issues, while still keeping the convenient properties of multigrid schemes; this led to the design of *Algebraic Multigrid (AMG)* routines.

Algebraic multigrid methods construct a sequence of grids based only on some properties of the entries of the matrix $\mathcal{A}$; specifically, given the unknowns x_i they find the subset of unknowns $\{x_l\}$ that are strongly connected to it, as in, for instance, [22, 121]:

$$C_i = \left\{ \bar{k} \neq i \,|\, -a_{i,\bar{k}} \geqslant \vartheta \max_{a_{i,l}<0} -a_{i,l} \right\},$$

with $0 < \vartheta \leqslant 1$. From here, the routine chooses which nodes have to be taken out from the coarse level and which have to be saved based on the influence (that is, the weight) one node has on the others. This approach has the advantage of not requiring any information on the geometry of the domain, and may also be applied to solving systems that are not a discretization of a differential operator. However, as for most multigrid methods, the routine requires the matrix $\mathcal{A}$ to be sparse; in addition, in order to define which nodes are strongly connected to the others, the matrix $\mathcal{A}$ often has to be ("close" to) an M-matrix, see, for instance, [22, 118, 122]. A matrix $\mathcal{A}$ is said to be an M-matrix if all the diagonal entries are positive, all the off-diagonal entries are non-positive, it is invertible, and all the entries of its inverse $\mathcal{A}^{-1}$ are positive [156, Definition 1.30].

Once we have defined the smoothing, the prolongation, and the restriction

operators, we can rewrite the overall process in terms of a simple iteration, defined by the following iteration matrix in the case of two grid levels:

$$T = (I_{n_\mathcal{A}} - \mathcal{P}_{\text{smooth}}^{-1}\mathcal{A})^{s_\text{l}}(I_{n_\mathcal{A}} - P\mathcal{A}_c^{-1}P^\top\mathcal{A})(I_{n_\mathcal{A}} - (\mathcal{P}_{\text{smooth}}^\top)^{-1}\mathcal{A})^{s_\text{r}},$$

where $\mathcal{P}_{\text{smooth}}$ is the smoothing operator employed, P is the prolongation operator, and $\mathcal{A}_c = P^\top\mathcal{A}P$ is the coarse grid matrix; note that we allow here a different number of steps s_r and s_l for the pre- and post-smoothing operators. The simple iteration above is known as *2-grid scheme* [44, Section 2.5], and can be generalized to more grid levels.

In the following chapters, we will employ algebraic multigrid methods as black-box preconditioners for specific matrix blocks; in particular, we will apply a fixed number of V-cycles of the `HSL_MI20` solver [22] or of the AGMG routine [118, 121, 122, 123]. Roughly speaking, a V-cycle is defined as the application of the iteration matrix T until a coarsest grid level $\texttt{l}_c$ is reached, on which the matrix $\mathcal{A}_c$ is solved exactly. More precisely, starting from the whole matrix $\mathcal{A}$, the algorithm applies the smoothing operator on the current error, then constructs the prolongation and the restriction operators, projects the error on the coarser grid, and applies the 2-grid scheme to the matrix $\mathcal{A}_c$ defined above. This process is repeated until the coarsest grid level $\texttt{l}_c$ is reached, on which an exact solve is performed. After this, the algorithm recovers the solution on each of the finer grid by applying the prolongation and the smoothing operator. A pseudocode is given in Algorithm 8.

Algorithm 8 Multigrid V-cycle to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$, with smoother $\mathcal{P}_{\text{smooth}}$

function $\mathbf{x} = \text{VCycle}(\mathcal{A}, \mathbf{b}, \mathbf{x}, level)$
for $i = 1, \ldots, s_\text{r}$ **do**
  $\mathbf{x} = (I_{n_\mathcal{A}} - (\mathcal{P}_{\text{smooth}}^\top)^{-1}\mathcal{A})\mathbf{x} - (\mathcal{P}_{\text{smooth}}^\top)^{-1}\mathbf{b}$
end for
if $level = \texttt{l}_c$ **then**
  Solve $\mathcal{A}\mathbf{x} = \mathbf{b}$
else
  $\mathbf{r} = P^\top(\mathbf{b} - \mathcal{A}\mathbf{x})$
  $\mathcal{A}_c = P^\top\mathcal{A}P$
  $\mathbf{e} = \text{VCycle}(\mathcal{A}_c, \mathbf{r}, \mathbf{e}, level + 1)$
  $\mathbf{x} = \mathbf{x} + P\mathbf{e}$
end if
for $i = 1, \ldots, s_\text{l}$ **do**
  $\mathbf{x} = (I_{n_\mathcal{A}} - \mathcal{P}_{\text{smooth}}^{-1}\mathcal{A})\mathbf{x} - \mathcal{P}_{\text{smooth}}^{-1}\mathbf{b}$
end for

2.6 Preconditioned Krylov Subspace Methods

The iterative methods presented so far in this chapter construct a sequence of approximations $\mathbf{x}^{(0)}$, $\mathbf{x}^{(1)}$, ... to the solution $\mathbf{x}$ of a linear system, by applying a

specific (iteration) matrix to the current residual $\mathbf{r}^{(j)}$, then updating the iterate $\mathbf{x}^{(j)}$ until convergence is achieved. This can happen quite quickly if we choose the iteration matrix appropriately. However, having good spectral information on the iteration matrix is not always possible, and in fact the method could also diverge sometimes. In this section, we introduce another class of methods: the *(preconditioned) Krylov subspace methods*, that will allow us to have reliable guarantees of convergence for a number of problems.

One early method discovered belonging to this class was the Conjugate Gradient (CG) method, devised by Hestenes and Stiefel [80] in 1952 for solving a linear system (2.1) with $\mathcal{A} \succ 0$. This method, first used as a direct method, was proved to converge (in exact arithmetic) in no more than $n_{\mathcal{A}}$ steps. Despite this appealing property, only in the 1970s was the Conjugate Gradient method conceived as an iterative method, and from then the research moved to extend the method to more general cases, leading to the discovery of the Minimal Residual (MINRES) method [128], the Generalized Minimal Residual (GMRES) method [157], and the BiConjugate Gradient (BiCG) method [51], just to name a few. As the speed of convergence often depends on the eigenvalue/eigenvector properties of the matrix $\mathcal{A}$, a *preconditioned* version of all those methods has been devised, in order to accelerate the iterative process. In the following, we will introduce the general formulation of a Krylov subspace method, that will then be specified based on the properties of the matrix $\mathcal{A}$ considered. For a more comprehensive discussion on Krylov subspace methods, on which we base the discussion below, see, for example, [68, 156], or the survey [54].

Let suppose we want to solve the system (2.1), with $\mathcal{A}$ invertible. Given an initial guess $\mathbf{x}^{(0)}$ of the solution $\mathbf{x}$, a Krylov subspace method construct a sequence $\left\{\mathbf{x}^{(j)}\right\}$ of approximations of $\mathbf{x}$, such that

$$\mathbf{x}^{(j)} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)}),$$

where $\mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)}) = \text{span}\left\{\mathbf{r}^{(0)}, \mathcal{A}\mathbf{r}^{(0)}, \dots, \mathcal{A}^{j-1}\mathbf{r}^{(0)}\right\}$ is the Krylov subspace generated by $\mathcal{A}$ and $\mathbf{r}^{(0)}$; with the previous expression we mean that

$$\mathbf{x}^{(j)} - \mathbf{x}^{(0)} \in \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)}).$$

From the definition of the Krylov subspace $\mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$, the residual $\mathbf{r}^{(j)}$ at the j-th iterate can be rewritten as

$$\mathbf{r}^{(j)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(j)} = p_{j-1}(\mathcal{A})\mathbf{r}^{(0)},$$

where $p_{j-1}(z)$ is a polynomial of degree $j-1$ such that $p_{j-1}(0) = 1$.

As we seek a good approximation $\mathbf{x}^{(j)}$ of the solution $\mathbf{x}$, we wish that the residual is "small" in some sense; in fact, one condition the iterate $\mathbf{x}^{(j)}$ may be required to satisfy is minimizing the residual $\mathbf{r}^{(j)}$ in some norm $\|\cdot\|$:

$$\begin{aligned} \|\mathbf{r}^{(j)}\| &= \min_{\hat{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})} \|\mathbf{b} - \mathcal{A}\hat{\mathbf{x}}\| \\ &= \min_{p_{j-1}(z) \in \Pi_{j-1}, p_{j-1}(0)=1} \|p_{j-1}(\mathcal{A})\mathbf{r}^{(0)}\|, \end{aligned} \tag{2.12}$$

where, as above, Π_{j-1} is the space of polynomials of degree at most $j-1$. We would like to mention that the choice of the norm $\|\cdot\|$ characterizes the Krylov subspace method. For instance, the Conjugate Gradient method minimizes the residual in the $\mathcal{A}^{-1}$-norm. On the other hand, the class of Minimal Residual methods (like MINRES or GMRES) minimizes the residual in the Euclidean norm. Another required property for the residual $\mathbf{r}^{(j)}$ to hold is to be orthogonal to some subspace W_j of dimension j:

$$\mathbf{w}^\top \mathbf{r}^{(j)} = 0, \quad \forall \mathbf{w} \in W_j. \tag{2.13}$$

Different choices of W_j produce different Krylov subspace methods. From (2.13), Krylov subspace methods fall within the class of *projection methods* for solving linear systems of equations.

From (2.12), we can derive

$$\|\mathbf{r}^{(j)}\| \leqslant \min_{p_{j-1}(z)\in\Pi_{j-1}, p_{j-1}(0)=1} \|p_{j-1}(\mathcal{A})\| \, \|\mathbf{r}^{(0)}\|, \tag{2.14}$$

finding again a min–max problem as in Section 2.4. However, in this case the problem has no explicit solution, as the polynomial $p_{j-1}(z)$ is chosen to satisfy (2.12), which is clearly dependent not only on the matrix $\mathcal{A}$, but also on the initial residual $\mathbf{r}^{(0)}$: starting from a different initial guess $\bar{\mathbf{x}}^{(0)}$, in fact, causes the polynomial $p_{j-1}(z)$ to be different. In a similar way, changing the right hand side $\bar{\mathbf{b}}$ but starting from the same $\mathbf{x}^{(0)}$ will give a different polynomial $p_{j-1}(z)$. From here, we can deduce that any Krylov subspace method is a non-linear process, as opposed to the other methods presented so far.

For general $\mathcal{A}$, if we want to find $\mathbf{x}^{(j)} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$ we need to construct a basis for the space $\mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$; the latter is done by employing an Arnoldi process, and will lead to the GMRES algorithm. As at each iteration we need to construct an orthonormal basis of increasing dimension, we expect the method to be more expensive as the process moves forward. However, practical implementations make use of a *restart* parameter $\bar{\mathtt{j}}$, giving an upper bound for the dimension of the basis sought: if the computed solution found after $\bar{\mathtt{j}}$ iterations does not satisfy the given convergence criterion, this approximation of the solution is employed as initial guess for the algorithm. Besides, there are special cases in which the new approximation $\mathbf{x}^{(j)}$ can be constructed by a short-term recurrence formula, as for the Chebyshev semi-iteration method. In particular, if the matrix $\mathcal{A}$ is symmetric, we can derive a three-term recurrence formula by employing a Lanczos process; this will lead to the MINRES algorithm. If we suppose also that the matrix $\mathcal{A}$ is positive definite, the process is simplified even more to a two-term recurrence; in this case, we obtain the CG algorithm.

Since the reduction of the residual at the j-th iterate (2.14) for a Krylov subspace method depends on the norm of a polynomial in $\mathcal{A}$, we may expect that, as for in Section 2.4, the convergence properties of the method depends on the spectrum of $\mathcal{A}$. As this matrix may be ill-conditioned, usually Krylov methods are considered in their *preconditioned* version. In this case, given an appropriate preconditioner $\mathcal{P}$, the Krylov subspace within which we seek the approximation $\mathbf{x}^{(j)}$ is of the form $\mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{P}^{-1}\mathcal{A}, \mathbf{r}^{(0)})$. For some of these methods,

the preconditioner $\mathcal{P}$ has not only to be easily invertible and spectrally equivalent to $\mathcal{A}$, but also needs to be symmetric positive definite, in order for the norm in (2.12) to be well defined.

In the following sections, we are going to introduce the mostly known Krylov methods, namely, CG, MINRES, and GMRES, in their preconditioned version. For the GMRES method, we also introduce its flexible implementation.

2.7 Preconditioned Conjugate Gradient Method

In this section we introduce the Conjugate Gradient method [80] for solving symmetric positive definite linear systems.

Let suppose that the matrix $\mathcal{A}$ in (2.1) is symmetric positive definite; in particular, the matrix $\mathcal{A}$ is invertible. We can then consider the $\mathcal{A}$-norm, defined for any $\mathbf{z} \in \mathbb{R}^{n_\mathcal{A}}$ as

$$\|\mathbf{z}\|_\mathcal{A} = \sqrt{\mathbf{z}^\top \mathcal{A}\, \mathbf{z}}.$$

In addition, since $\mathcal{A} \succ 0$, the same holds for $\mathcal{A}^{-1}$, and we can analogously consider the norm $\|\cdot\|_{\mathcal{A}^{-1}}$. Then, the Conjugate Gradient method produces a sequence of approximations $\mathbf{x}^{(j)}$ that minimize the residual $\mathbf{r}^{(j)}$ in the $\mathcal{A}^{-1}$-norm, namely

$$\|\mathbf{r}^{(j)}\|_{\mathcal{A}^{-1}} = \min_{\hat{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})} \|\mathbf{b} - \mathcal{A}\hat{\mathbf{x}}\|_{\mathcal{A}^{-1}}.$$

Recalling that

$$\mathbf{r}^{(j)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(j)} = \mathcal{A}(\mathbf{x} - \mathbf{x}^{(j)}) = \mathcal{A}\mathbf{e}^{(j)},$$

the Conjugate Gradient iterate can be defined equivalently as a sequence of approximations $\mathbf{x}^{(j)}$ that minimize the error $\mathbf{e}^{(j)}$ in the $\mathcal{A}$-norm. In fact, we have

$$\begin{aligned}
\|\mathbf{r}^{(j)}\|_{\mathcal{A}^{-1}} &= \min_{\hat{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})} \|\mathbf{b} - \mathcal{A}\hat{\mathbf{x}}\|_{\mathcal{A}^{-1}} \\
&= \min_{\hat{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})} \sqrt{(\mathbf{b} - \mathcal{A}\hat{\mathbf{x}})^\top \mathcal{A}^{-1} (\mathbf{b} - \mathcal{A}\hat{\mathbf{x}})} \\
&= \min_{\hat{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})} \sqrt{\mathbf{e}^\top \mathcal{A}\, \mathbf{e}} \\
&= \|\mathbf{e}^{(j)}\|_\mathcal{A}.
\end{aligned}$$

If we evaluate explicitly $\|\mathbf{e}\|_\mathcal{A}^2$, we obtain

$$\begin{aligned}
\|\mathbf{e}\|_\mathcal{A}^2 &= (\mathbf{x} - \hat{\mathbf{x}})^\top \mathcal{A} (\mathbf{x} - \hat{\mathbf{x}}) \\
&= \mathbf{x}^\top \mathcal{A}\, \mathbf{x} + \hat{\mathbf{x}}^\top \mathcal{A}\, \hat{\mathbf{x}} - \mathbf{x}^\top \mathcal{A}\, \hat{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathcal{A}\, \mathbf{x} \\
&= \hat{\mathbf{x}}^\top \mathcal{A}\, \hat{\mathbf{x}} + \mathbf{x}^\top \mathbf{b} - \mathbf{b}^\top \hat{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{b} \\
&= \hat{\mathbf{x}}^\top \mathcal{A}\, \hat{\mathbf{x}} - 2\hat{\mathbf{x}}^\top \mathbf{b} + \mathbf{x}^\top \mathbf{b}.
\end{aligned}$$

Then, defining the quadratic function

$$\phi(\hat{\mathbf{x}}) = \frac{1}{2} \hat{\mathbf{x}}^\top \mathcal{A}\, \hat{\mathbf{x}} - \hat{\mathbf{x}}^\top \mathbf{b},$$

we have $\|\mathbf{e}\|_{\mathcal{A}}^2 = 2\phi(\hat{\mathbf{x}}) + c$, with $c = \mathbf{x}^\top \mathbf{b}$ a given constant. Therefore, it is clear that minimizing $\|\mathbf{e}\|_{\mathcal{A}}$ is equivalent to minimizing the function $\phi(\hat{\mathbf{x}})$, and, since $\nabla\phi(\hat{\mathbf{x}}) = \mathcal{A}\hat{\mathbf{x}} - \mathbf{b}$, if $\bar{\mathbf{x}}$ is the minimizer of $\phi(\hat{\mathbf{x}})$, then $\bar{\mathbf{x}} = \mathbf{x}$ due to the invertibility of $\mathcal{A}$.

Given $\mathbf{x}^{(j-1)}$ an approximation of $\mathbf{x}$, we are then looking for a vector $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \alpha_{j-1}\mathbf{s}^{(j-1)}$ that improves our cost functional $\phi(\mathbf{x}^{(j-1)})$. The vector $\mathbf{s}^{(j-1)}$ is called the *search direction*, while the parameter α_{j-1} is chosen to be the optimal step-length in this direction. The latter can be easily computed, for one has to minimize the cost functional $\phi(\hat{\mathbf{x}})$ in the one-dimensional subspace $\mathbf{x}^{(j-1)} + \operatorname{span}\left\{\mathbf{s}^{(j-1)}\right\}$; in fact, the optimal value for α is obtained by imposing the condition

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\phi(\mathbf{x}^{(j-1)} + \alpha\mathbf{s}^{(j-1)}) = 0,$$

that leads to

$$\alpha_{j-1} = \frac{(\mathbf{s}^{(j-1)})^\top \mathbf{r}^{(j-1)}}{(\mathbf{s}^{(j-1)})^\top \mathcal{A}\,\mathbf{s}^{(j-1)}}. \tag{2.15}$$

Now that we have the optimal step-length α_{j-1}, we need to find the direction $\mathbf{s}^{(j-1)}$ along which we should search for the new iterate $\mathbf{x}^{(j)}$. As the negative gradient $-\nabla\phi(\mathbf{x}^{(j-1)}) = \mathbf{r}^{(j-1)}$ is a direction pointing towards the minimizer of the functional $\phi(\hat{\mathbf{x}})$, one is tempted to choose $\mathbf{s}^{(j-1)} = \mathbf{r}^{(j-1)}$ for all j. This choice of search directions gives the *Steepest Descent* method [119, Chapter 2], and it is easy to see that at step j the error $\mathbf{e}^{(j)}$ and the residual $\mathbf{r}^{(j-1)}$ are $\mathcal{A}$-conjugate, that is, $(\mathbf{e}^{(j)})^\top \mathcal{A}\,\mathbf{r}^{(j-1)} = 0$ for all j, which implies that the residual $\mathbf{r}^{(j)}$ is orthogonal to the residual $\mathbf{r}^{(j-1)}$. However, the latter is not true for all the previous residuals, and indeed the vectors $\left\{\mathbf{r}^{(i)}\right\}_{i=0}^{j}$ may be linearly dependent, with the consequence of slow convergence of the method. Therefore, we are looking for a smarter choice of the search directions in order to obtain faster convergence properties.

In order to find the search directions $\mathbf{s}^{(j-1)}$, we will make use of one fundamental property of symmetric positive definite matrices: given a set $\left\{\mathbf{s}^{(i)}\right\}_{i=0}^{j-1}$ of non-zero vectors which are mutually $\mathcal{A}$-conjugate (that is, $(\mathbf{s}^{(i)})^\top \mathcal{A}\,\mathbf{s}^{(l)} = 0$, for all $i \neq l$), then they are linearly independent vectors. With this in mind, we may construct vectors $\left\{\mathbf{s}^{(j)}\right\}$ which are mutually $\mathcal{A}$-conjugate and span the whole space $\mathbb{R}^{n_{\mathcal{A}}}$ in $n_{\mathcal{A}}$ iterations, thus obtaining a good approximation of the solution in fewer steps. In addition, we will require a condition of the type (2.13) on the residual $\mathbf{r}^{(j)}$ to hold; specifically, we will impose that the j-th residual is orthogonal to $W_j = \operatorname{span}\left\{\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(j-1)}\right\}$. In order to ensure $\mathcal{A}$-conjugacy and orthogonality, we will employ a Lanczos process, that will require us to impose those properties only on two subsequent search directions and residuals, due to the matrix $\mathcal{A}$ being symmetric positive definite.

Since we are looking for an approximation of the type $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \alpha_{j-1}\mathbf{s}^{(j-1)}$, it is clear that $\mathbf{x}^{(j)}$ belongs to $\mathbf{x}^{(0)} + \operatorname{span}\left\{\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(j-1)}\right\}$. From the observation above, we thus look for vectors $\left\{\mathbf{s}^{(i)}\right\}_{i=0}^{j-1}$ that are mutually $\mathcal{A}$-conjugate; in addition, we require that the choice of the search direction $\mathbf{s}^{(j-1)}$ results in a residual $\mathbf{r}^{(j)}$ that is orthogonal to all previous residuals $\left\{\mathbf{r}^{(i)}\right\}_{i=0}^{j-1}$. For these two

conditions to hold, we look for a search direction

$$\mathbf{s}^{(j-1)} = \mathbf{r}^{(j-1)} + \beta_{j-1}\mathbf{s}^{(j-2)}, \tag{2.16}$$

where β_{j-1} is chosen such that $\mathbf{s}^{(j-1)}$ and $\mathbf{s}^{(j-2)}$ are $\mathcal{A}$-conjugate; we note that with this choice of search direction $\mathbf{s}^{(j-1)} \in \text{span}\left\{\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(j-1)}\right\}$. In order to ensure the $\mathcal{A}$-conjugacy of the vectors $\mathbf{s}^{(j-1)}$ and $\mathbf{s}^{(j-2)}$, we take

$$\beta_{j-1} = -\frac{(\mathbf{r}^{(j-1)})^\top \mathcal{A}\,\mathbf{s}^{(j-2)}}{(\mathbf{s}^{(j-2)})^\top \mathcal{A}\,\mathbf{s}^{(j-2)}}. \tag{2.17}$$

From $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \alpha_{j-1}\mathbf{s}^{(j-1)}$, we see that $\mathbf{r}^{(j)} = \mathbf{r}^{(j-1)} - \alpha_{j-1}\mathcal{A}\,\mathbf{s}^{(j-1)}$, and, since we want $(\mathbf{r}^{(j)})^\top \mathbf{r}^{(j-1)} = 0$, we require

$$\alpha_{j-1} = \frac{(\mathbf{r}^{(j-1)})^\top \mathbf{r}^{(j-1)}}{(\mathbf{r}^{(j-1)})^\top \mathcal{A}\,\mathbf{s}^{(j-1)}}.$$

The previous expression can be simplified as

$$\alpha_{j-1} = \frac{(\mathbf{r}^{(j-1)})^\top \mathbf{r}^{(j-1)}}{(\mathbf{r}^{(j-1)} + \beta_{j-1}\mathbf{s}^{(j-2)})^\top \mathcal{A}\,\mathbf{s}^{(j-1)}} = \frac{(\mathbf{r}^{(j-1)})^\top \mathbf{r}^{(j-1)}}{(\mathbf{s}^{(j-1)})^\top \mathcal{A}\,\mathbf{s}^{(j-1)}}, \tag{2.18}$$

due to $\mathbf{s}^{(j-1)}$ and $\mathbf{s}^{(j-2)}$ being $\mathcal{A}$-conjugate. Note that this values of α_{j-1} is the same as in (2.15) with $\mathbf{s}^{(j-1)}$ replaced by $\mathbf{r}^{(j-1)}$; however, as $\mathbf{r}^{(j-1)}$ is orthogonal to $W_{j-1} = \text{span}\left\{\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(j-2)}\right\}$, and since we have $\mathbf{s}^{(j-2)} \in W_{j-1}$, from (2.16) the two expressions are the same. Thus, the steplength α_{j-1} defined in (2.18) is optimal.

In order to also simplify the expression (2.17) for β_{j-1}, from

$$\mathbf{r}^{(j-1)} = \mathbf{r}^{(j-2)} - \alpha_{j-2}\mathcal{A}\,\mathbf{s}^{(j-2)}$$

we derive

$$-\mathcal{A}\,\mathbf{s}^{(j-2)} = \frac{1}{\alpha_{j-2}}(\mathbf{r}^{(j-1)} - \mathbf{r}^{(j-2)}).$$

By employing the orthogonality property $(\mathbf{r}^{(j-1)})^\top \mathbf{r}^{(j-2)} = 0$ together with (2.18), we finally obtain the following expression for β_{j-1}:

$$\beta_{j-1} = \frac{(\mathbf{r}^{(j-1)})^\top \mathbf{r}^{(j-1)}}{(\mathbf{r}^{(j-2)})^\top \mathbf{r}^{(j-2)}}.$$

The algorithm so derived is the Conjugate Gradient Method originally presented by Hestenes and Stiefel in [80]; the pseudocode is given in Algorithm 9. With this definition of $\mathbf{x}^{(j)}$ and of $\mathbf{s}^{(j)}$, it can be proved [174, Theorem 38.1] that

$$\begin{aligned}\text{span}\left\{\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(j-1)}\right\} &= \text{span}\left\{\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(j-1)}\right\} \\ &= \text{span}\left\{\mathbf{r}^{(0)}, \mathcal{A}\mathbf{r}^{(0)}, \ldots, \mathcal{A}^{j-1}\mathbf{r}^{(0)}\right\},\end{aligned}$$

implying that the iterate $\mathbf{x}^{(j)}$ belongs to the Krylov subspace $\mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$.

In addition, the search directions are mutually $\mathcal{A}$-conjugate, with the residuals being mutually orthogonal, see [156, Section 6.7] and [174, Theorem 38.1].

Algorithm 9 Conjugate Gradient Method to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$, with $\mathcal{A} \succ 0$

Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
Set $\mathbf{s}^{(0)} = \mathbf{r}^{(0)}$
for $j = 1$ **until** convergence, **do**
  $\alpha_{j-1} = \frac{(\mathbf{r}^{(j-1)})^\top \mathbf{r}^{(j-1)}}{(\mathbf{s}^{(j-1)})^\top \mathcal{A}\,\mathbf{s}^{(j-1)}}$
  $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \alpha_{j-1}\mathbf{s}^{(j-1)}$
  $\mathbf{r}^{(j)} = \mathbf{r}^{(j-1)} - \alpha_{j-1}\mathcal{A}\,\mathbf{s}^{(j-1)}$
  Test for convergence
  $\beta_j = \frac{(\mathbf{r}^{(j)})^\top \mathbf{r}^{(j)}}{(\mathbf{r}^{(j-1)})^\top \mathbf{r}^{(j-1)}}$
  $\mathbf{s}^{(j)} = \mathbf{r}^{(j)} + \beta_j \mathbf{s}^{(j-1)}$
end for

As $\mathbf{x}^{(j)} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$, we can rewrite the error $\mathbf{e}^{(j)}$ as

$$\mathbf{e}^{(j)} = \mathbf{e}^{(0)} + p_{j-1}(\mathcal{A})\mathbf{r}^{(0)},$$

for some polynomial $p_{j-1}(z)$ of degree at most $j-1$ such that $p_{j-1}(0) = 1$. Writing $\mathbf{r}^{(0)} = \mathcal{A}\mathbf{e}^{(0)}$, we then obtain

$$\mathbf{e}^{(j)} = q_j(\mathcal{A})\mathbf{e}^{(0)},$$

with $q_j(z)$ a polynomial of degree at most j such that $q_j(0) = 1$. Recalling that at each step CG minimizes the residual $\mathbf{r}^{(j)}$ in the $\mathcal{A}^{-1}$-norm, and that this is equivalent to minimizing the error $\mathbf{e}^{(j)}$ in the $\mathcal{A}$-norm, we can rewrite the reduction in the residual (2.12) as

$$\|\mathbf{r}^{(j)}\|_{\mathcal{A}^{-1}} = \|\mathbf{e}^{(j)}\|_{\mathcal{A}} = \min_{q_j(z)\in\Pi_j, q_j(0)=1} \|q_j(\mathcal{A})\mathbf{e}^{(0)}\|_{\mathcal{A}},$$

where Π_j is the space of polynomials of degree at most j. If we expand the error $\mathbf{e}^{(0)}$ in terms of the orthonormal eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n_\mathcal{A}}\}$ of $\mathcal{A}$, we have $\mathbf{e}^{(0)} = \sum_{i=1}^{n_\mathcal{A}} c_i \mathbf{v}_i$, and thus we can derive the following estimate for the error:

$$\begin{aligned}
\|\mathbf{e}^{(j)}\|_{\mathcal{A}} &= \min_{q_j(z)\in\Pi_j, q_j(0)=1} \left\| q_j(\mathcal{A}) \sum_{i=1}^{n_\mathcal{A}} c_i \mathbf{v}_i \right\|_{\mathcal{A}} \\
&= \min_{q_j(z)\in\Pi_j, q_j(0)=1} \left\| \sum_{i=1}^{n_\mathcal{A}} c_i q_j(\lambda_i) \mathbf{v}_i \right\|_{\mathcal{A}} \\
&\leqslant \min_{q_j(z)\in\Pi_j, q_j(0)=1} \left\| \max_i |q_j(\lambda_i)| \sum_{i=1}^{n_\mathcal{A}} c_i \mathbf{v}_i \right\|_{\mathcal{A}}. \qquad (2.19)
\end{aligned}$$

Before moving on the error estimate, it is worth noting that from the previous expression we can see that Conjugate Gradient will converge at most in $n_\mathcal{A}$ it-

erations, see [174, Theorem 38.4]. In fact, supposing that the matrix $\mathcal{A}$ has $n_{\mathcal{A}}$ different eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n_{\mathcal{A}}}$, we can consider the polynomial

$$q_{n_{\mathcal{A}}}(z) = \prod_{i=1}^{n_{\mathcal{A}}} \left(1 - \frac{z}{\lambda_i}\right),$$

that will be zero on the set of the eigenvalues, and as a consequence the error $\mathbf{e}^{(n_{\mathcal{A}})}$ will be zero.

From (2.19), we can write

$$\|\mathbf{e}^{(j)}\|_{\mathcal{A}} \leqslant \min_{q_j(z) \in \Pi_j, q_j(0)=1} \max_{a \leqslant \lambda \leqslant b} |q_j(\lambda)| \, \|\mathbf{e}^{(0)}\|_{\mathcal{A}},$$

with $[a, b]$ the interval containing the eigenvalues of the matrix $\mathcal{A}$. A similar term was found in Section 2.4 when deriving the error for the Chebyshev semi-iterative method, and we were able to find an upper bound by employing the Chebyshev polynomials. In fact, it can be proved [174, Theorem 38.5] that the reduction of the error at the j-th iteration is given by

$$\|\mathbf{e}^{(j)}\|_{\mathcal{A}} \leqslant 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^j \|\mathbf{e}^{(0)}\|_{\mathcal{A}}, \tag{2.20}$$

with κ the condition number of the matrix $\mathcal{A}$.

Algorithm 10 Preconditioned Conjugate Gradient Method to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$, with $\mathcal{A} > 0$ and preconditioner $\mathcal{P} > 0$

Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
Solve $\mathcal{P}\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$
Set $\mathbf{s}^{(0)} = \mathbf{z}^{(0)}$
for $j = 1$ **until** convergence, **do**
  $\alpha_{j-1} = \frac{(\mathbf{z}^{(j-1)})^\top \mathbf{r}^{(j-1)}}{(\mathbf{s}^{(j-1)})^\top \mathcal{A}\, \mathbf{s}^{(j-1)}}$
  $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \alpha_{j-1}\mathbf{s}^{(j-1)}$
  $\mathbf{r}^{(j)} = \mathbf{r}^{(j-1)} - \alpha_{j-1}\mathcal{A}\, \mathbf{s}^{(j-1)}$
  Test for convergence
  Solve $\mathcal{P}\mathbf{z}^{(j)} = \mathbf{r}^{(j)}$
  $\beta_j = \frac{(\mathbf{z}^{(j)})^\top \mathbf{r}^{(j)}}{(\mathbf{z}^{(j-1)})^\top \mathbf{r}^{(j-1)}}$
  $\mathbf{s}^{(j)} = \mathbf{z}^{(j)} + \beta_j \mathbf{s}^{(j-1)}$
end for

From (2.20), one may observe that the convergence rate of CG can be poor when applied to an ill-conditioned matrix $\mathcal{A}$. However, the rate (2.20) is only a worst-case scenario: in fact, supposing that the matrix $\mathcal{A}$ has only two different eigenvalues λ_1 and λ_2, from (2.19) with $q_2(z) = (1 - z/\lambda_1)(1 - z/\lambda_2)$ we can imply that the error is zero after only two iterations. In general, one may experience also fast convergence in practice when the eigenvalues of $\mathcal{A}$ are clustered around some values. However, this is not always the case; for this reason, a preconditioner

$\mathcal{P}$ is generally employed for the purpose of obtaining fast convergence. As we mentioned in Section 2.6, this preconditioner $\mathcal{P}$ has to be easily invertible and ideally spectrally equivalent to $\mathcal{A}$. In addition, as we want to preserve symmetry and positive definiteness, the preconditioner $\mathcal{P}$ needs to be symmetric positive definite. In fact, in this case we can write $\mathcal{P} = HH^\top$, and the preconditioned system is equivalent to

$$H^{-1}\mathcal{A}H^{-\top}\bar{\mathbf{x}} = H^{-1}\mathbf{b},$$

where $\bar{\mathbf{x}} = H^\top\mathbf{x}$. As $H^{-1}\mathcal{A}H^{-\top}$ is symmetric positive definite, we can employ CG to solve the previous system. However, practical implementations of Preconditioned Conjugate Gradients do not require one to solve with the matrix H, but rather to solve with $\mathcal{P}$. The pseudocode for Preconditioned Conjugate Gradient is given in Algorithm 10.

Comparing Algorithm 9 with its preconditioned version in Algorithm 10, we observe that the only computational difference is the solve with $\mathcal{P}$ at each iteration. From here, it is clear the reason why the preconditioner $\mathcal{P}$ has to be easy to invert. As we will discuss in Section 2.10, in general, rather than solving with $\mathcal{P}$, we will require an approximation of its inverse operation on a generic vector.

2.8 Preconditioned MINRES

In the previous section we discussed the Conjugate Gradient method for solving linear systems of equations. As we saw, the power of CG lies in the two-term recurrence formula for the search directions, that ensures orthogonality of the residuals and $\mathcal{A}$-conjugacy of the search directions. The main drawback of the method is that $\mathcal{A}$ is required to be symmetric positive definite. In this section we introduce a generalization of CG to symmetric (possibly indefinite) matrices discovered by Paige and Saunders [128], that is MINRES.

Let suppose that the matrix $\mathcal{A}$ in (2.1) is symmetric and invertible. Given an initial guess $\mathbf{x}^{(0)}$, we iteratively find an approximation $\mathbf{x}^{(j)}$ of $\mathbf{x}$ within the subspace $\mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$. As opposed to CG, in this case we cannot consider the $\mathcal{A}$-norm, as the matrix $\mathcal{A}$ may be indefinite. For this reason, at each iteration the residual $\mathbf{r}^{(j)}$ will be minimized in the Euclidean norm

$$\|\mathbf{r}^{(j)}\|_2 = \min_{p_{j-1}(z)\in\Pi_{j-1}, p_{j-1}(0)=1} \|p_{j-1}(\mathcal{A})\mathbf{r}^{(0)}\|_2.$$

As for CG, we want to span $\mathbb{R}^{n_\mathcal{A}}$ in (at most) $n_\mathcal{A}$ iterations by constructing a basis for the space. Since the matrix $\mathcal{A}$ is symmetric, this can be done again by employing the short-term recurrence formula of a Lanczos process; however, as opposed to CG, in this case the formula is given by the following three-term recurrence:

$$\gamma_{i+1}\mathbf{q}^{(i+1)} = \mathcal{A}\mathbf{q}^{(i)} - \delta_i\mathbf{q}^{(i)} - \gamma_i\mathbf{q}^{(i-1)}, \tag{2.21}$$

setting $\mathbf{q}^{(1)} = \frac{\mathbf{r}^{(0)}}{\|\mathbf{r}^{(0)}\|_2}$, for $i = 1, 2, \ldots, j-1$, where $\delta_i = (\mathbf{q}^{(i)})^\top\mathcal{A}\,\mathbf{q}^{(i)}$, and γ_{i+1} is such that $\|\mathbf{q}^{(i+1)}\|_2 = 1$. The vectors $\left\{\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(j)}\right\}$ so derived are an orthonormal basis for $\mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$; in addition, equation (2.21) can be rewritten

as

$$\mathcal{A}Q_j = Q_{j+1}\widehat{H}_j, \tag{2.22}$$

where $Q_j = [\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(j)}]$ is orthogonal, and $\widehat{H}_j = \begin{bmatrix} H_j \\ \gamma_{j+1}\mathbf{e}_j \end{bmatrix}$. Here, $H_j = \text{tridiag}(\gamma_i, \delta_i, \gamma_{i+1})$ is a tridiagonal matrix, and $\mathbf{e}_j$ is the j-th vector of the standard basis in $\mathbb{R}^j$. It is worth noting that the matrix H_j is tridiagonal due to the symmetry of $\mathcal{A}$ being preserved by the Lanczos process. Since $\mathbf{x}^{(j)} \in \mathbf{x}^{(0)} + \mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$, with this notation we can rewrite

$$\mathbf{x}^{(j)} = \mathbf{x}^{(0)} + Q_j\mathbf{z}^{(j)},$$

for some vector $\mathbf{z}^{(j)} \in \mathbb{R}^j$. From the previous expression, we can write

$$\begin{aligned}
\|\mathbf{r}^{(j)}\|_2 &= \|\mathbf{b} - \mathcal{A}\mathbf{x}^{(j)}\|_2 \\
&= \|\mathbf{r}^{(0)} - \mathcal{A}Q_j\mathbf{z}^{(j)}\|_2 \\
&= \|\mathbf{r}^{(0)} - Q_{j+1}\widehat{H}_j\mathbf{z}^{(j)}\|_2 \\
&= \|\|\mathbf{r}^{(0)}\|_2\,\mathbf{e}_1 - \widehat{H}_j\mathbf{z}^{(j)}\|_2,
\end{aligned}$$

the last equality holding due to Q_{j+1} being orthogonal. Finally, recalling the minimization property of the j-th residual, the iterate $\mathbf{x}^{(j)}$ is the solution of the following least squares problem:

$$\|\mathbf{r}^{(j)}\|_2 = \min_{\mathbf{z}^{(j)}} \|\|\mathbf{r}^{(0)}\|_2\,\mathbf{e}_1 - \widehat{H}_j\mathbf{z}^{(j)}\|_2,$$

that is solved by employing Givens rotations within a QR factorization. A pseudocode for the MINRES method is found in Algorithm 11. It is worth noting that the MINRES algorithm performs a few more operations per iteration than the Conjugate Gradient method.

As for CG, we can specify the reduction of the residual (2.14) as follows:

$$\|\mathbf{r}^{(j)}\|_2 \leqslant \min_{p_j(z)\in\Pi_j, p_j(0)=1} \max_{\lambda\in[-a_1,-b_1]\cup[a_2,b_2]} |p_j(\lambda)|\ \|\mathbf{r}^{(0)}\|_2.$$

Here, as the matrix may be indefinite, we split the interval $[a, b]$ containing the eigenvalues of $\mathcal{A}$ in a positive and a negative part, and suppose that the spectrum of $\mathcal{A}$ is contained in the union $[-a_1, -b_1] \cup [a_2, b_2]$ of two intervals of the same length, that is, $a_1 - b_1 = b_2 - a_2$, with $a_1, a_2, b_1, b_2 > 0$. Then, it is possible to prove (see [44, Theorem 4.14] and [164]) the following reduction on the residual after $2j$ steps of MINRES:

$$\|\mathbf{r}^{(2j)}\|_2 \leqslant 2\left(\frac{\sqrt{a_1b_2} - \sqrt{a_2b_1}}{\sqrt{a_1b_2} + \sqrt{a_2b_1}}\right)^j \|\mathbf{r}^{(0)}\|_2. \tag{2.23}$$

From the bound above, it is clear that a substantial reduction in the residual may be obtained only every other iteration; this in turn produces the typical plots of the behavior of the residual obtained from numerical experiments: in fact, one can clearly see a staircasing effect.

Algorithm 11 MINRES Method to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$, with $\mathcal{A}$ symmetric

$\mathbf{q}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(1)} = \mathbf{0}$
Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{q}^{(1)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
Set $\gamma_1 = \|\mathbf{q}^{(1)}\|_2$
Set $\eta = \gamma_1$, $s_0 = s_1 = 0$, $c_0 = c_1 = 1$
for $j = 1$ **until** convergence, **do**
  $\mathbf{q}^{(j)} = \frac{\mathbf{q}^{(j)}}{\gamma_j}$
  $\delta_j = (\mathbf{q}^{(j)})^\top \mathcal{A}\,\mathbf{q}^{(j)}$
  $\mathbf{q}^{(j+1)} = \mathcal{A}\mathbf{q}^{(j)} - \delta_j \mathbf{q}^{(j)} - \gamma_j \mathbf{q}^{(j-1)}$
  $\gamma_{j+1} = \|\mathbf{q}^{(j+1)}\|_2$
  $\alpha_0 = c_j \delta_j - c_{j-1} s_j \gamma_j$
  $\alpha_1 = \sqrt{\alpha_0^2 + \gamma_{j+1}^2}$
  $\alpha_2 = s_j \delta_j + c_{j-1} c_j \gamma_j$
  $\alpha_3 = s_{j-1} \gamma_j$
  $c_{j+1} = \frac{\alpha_0}{\alpha_1}$, $s_{j+1} = \frac{\gamma_{j+1}}{\alpha_1}$
  $\mathbf{w}^{(j+1)} = \frac{1}{\alpha_1}(\mathbf{q}^{(j)} - \alpha_3 \mathbf{w}^{(j-1)} - \alpha_2 \mathbf{w}^{(j)})$
  $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + c_{j+1}\eta \mathbf{w}^{(j+1)}$
  $\eta = -s_{j+1}\eta$
  Test for convergence
end for

As for CG, it is clear that convergence depends mainly on the spectrum of the matrix $\mathcal{A}$; for this reason, in order to obtain faster convergence, a preconditioned version of MINRES has been devised. Since we want to preserve the symmetry of the system, we have to employ again a symmetric positive definite preconditioner $\mathcal{P}$; the algorithm so derived is given in Algorithm 12.

For the preconditioned MINRES algorithm with preconditioner $\mathcal{P}$, it can be shown that the residual $\mathbf{r}^{(j)}$ is reduced in the $\mathcal{P}^{-1}$-norm, see [44, Section 4.1]. Further, a similar result to (2.23) can be derived, supposing that the eigenvalues of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{A}$ lie in the union $[-a_1, -b_1] \cup [a_2, b_2]$ of intervals of same length, being $a_1, a_2, b_1, b_2 > 0$ as above, see [44, Theorem 4.14].

2.9 Preconditioned GMRES and Flexible GMRES

As we noted in the previous section, the difference in computations between CG and MINRES is minimal, with the latter performing only a few more vector–vector operations. The reason behind this is that both are based on a Lanczos process, which allows to construct an orthonormal basis for the eigenspace of a symmetric matrix $\mathcal{A}$. The generalization of this process to non-symmetric matrices is called the Arnoldi method: starting from an arbitrary vector $\mathbf{q}^{(0)}$, the method produces a sequence of orthonormal vector $\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(j)}$ by making use of the modified Gram–Schmidt process. From here, we can already see

Algorithm 12 Preconditioned MINRES Method to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$, with $\mathcal{A}$ symmetric and preconditioner $\mathcal{P} \succ 0$

$\mathbf{q}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(1)} = \mathbf{0}$
Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{q}^{(1)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
Solve $\mathcal{P}\mathbf{z}^{(1)} = \mathbf{q}^{(1)}$
Set $\gamma_1 = \sqrt{(\mathbf{z}^{(1)})^\top \mathbf{q}^{(1)}}$
Set $\eta = \gamma_1$, $s_0 = s_1 = 0$, $c_0 = c_1 = 1$
for $j = 1$ **until** convergence, **do**
  $\mathbf{z}^{(j)} = \frac{\mathbf{z}^{(j)}}{\gamma_j}$
  $\delta_j = (\mathbf{z}^{(j)})^\top \mathcal{A}\, \mathbf{z}^{(j)}$
  $\mathbf{q}^{(j+1)} = \mathcal{A}\mathbf{q}^{(j)} - \frac{\delta_j}{\gamma_j}\mathbf{q}^{(j)} - \frac{\gamma_j}{\gamma_{j-1}}\mathbf{q}^{(j-1)}$
  Solve $\mathcal{P}\mathbf{z}^{(j+1)} = \mathbf{q}^{(j+1)}$
  $\gamma_{j+1} = \sqrt{(\mathbf{z}^{(j+1)})^\top \mathbf{q}^{(j+1)}}$
  $\alpha_0 = c_j\delta_j - c_{j-1}s_j\gamma_j$
  $\alpha_1 = \sqrt{\alpha_0^2 + \gamma_{j+1}^2}$
  $\alpha_2 = s_j\delta_j + c_{j-1}c_j\gamma_j$
  $\alpha_3 = s_{j-1}\gamma_j$
  $c_{j+1} = \frac{\alpha_0}{\alpha_1}$, $s_{j+1} = \frac{\gamma_{j+1}}{\alpha_1}$
  $\mathbf{w}^{(j+1)} = \frac{1}{\alpha_1}(\mathbf{z}^{(j)} - \alpha_3\mathbf{w}^{(j-1)} - \alpha_2\mathbf{w}^{(j)})$
  $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + c_{j+1}\eta\mathbf{w}^{(j+1)}$
  $\eta = -s_{j+1}\eta$
  Test for convergence
end for

what is lost when solving a generic system, as we cannot employ any short-term recurrence formula for constructing the basis.

Let suppose that the matrix $\mathcal{A}$ in (2.1) is invertible but not symmetric. By starting from an initial guess $\mathbf{x}^{(0)}$, we construct a sequence $\mathbf{x}^{(j)}$ of approximations of the solution $\mathbf{x}$ that belong to the j-th Krylov subspace $\mathcal{K}_j(\mathcal{A}, \mathbf{r}^{(0)})$. Since the matrix $\mathcal{A}$ is not symmetric, as for MINRES we have to impose the minimization property of the residual $\mathbf{r}^{(j)}$ in the Euclidean norm:

$$\|\mathbf{r}^{(j)}\|_2 = \min_{p_{j-1}(z)\in\Pi_{j-1}, p_{j-1}(0)=1} \|p_{j-1}(\mathcal{A})\mathbf{r}^{(0)}\|_2.$$

In addition, we can generalize (see [156, Proposition 6.5]) the formula (2.22) for $\mathcal{A}$ in terms of the orthonormal basis $\{\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(j)}\}$ as

$$\mathcal{A}Q_j = Q_{j+1}\widehat{H}_j,$$

where as above $Q_j = [\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(j)}]$, and $\widehat{H}_j = \begin{bmatrix} H_j \\ h_{j+1,j}\mathbf{q}^{(j+1)} \end{bmatrix}$, with $H_j = [h_{i,l}]$, and $h_{i,l} = (\mathbf{q}^{(l)})^\top \mathcal{A}\mathbf{q}^{(i)}$. Note that now the matrix H_j is properly an upper Hessenberg matrix, as the matrix $\mathcal{A}$ is no longer symmetric. Since at each

iteration we are minimizing the residual, we can write

$$\begin{aligned}\|\mathbf{r}^{(j)}\|_2 &= \|\mathbf{b} - \mathcal{A}\mathbf{x}^{(j)}\|_2 \\ &= \|\mathbf{r}^{(0)} - \mathcal{A}Q_j\mathbf{z}^{(j)}\|_2 \\ &= \|\mathbf{r}^{(0)} - Q_{j+1}\widehat{H}_j\mathbf{z}^{(j)}\|_2 \\ &= \|\|\mathbf{r}^{(0)}\|_2\,\mathbf{e}_1 - \widehat{H}_j\mathbf{z}^{(j)}\|_2,\end{aligned}$$

using the orthogonality of Q_{j+1}. Again, we have to solve a least squares problem, which is done by employing a Givens rotation within a QR factorization. The resulting method is the GMRES method derived by Saad and Schultz [157].

As for CG and MINRES, it is possible to prove an upper bound for the reduction of the residual; however, the derivation of this estimate for GMRES is not as straightforward as for CG and MINRES, and indeed no neat expression of the type (2.20) or (2.23) can be obtained unless $\mathcal{A}$ is normal. In fact, supposing that $\mathcal{A}$ is diagonalizable with eigendecomposition $\mathcal{A} = Q\Lambda Q^{-1}$, we can rewrite

$$\begin{aligned}\|\mathbf{r}^{(j)}\|_2 &= \min_{p_{j-1}(z)\in\Pi_{j-1},p_{j-1}(0)=1} \|p_{j-1}(\mathcal{A})\mathbf{r}^{(0)}\|_2 \\ &\leqslant \min_{p_{j-1}(z)\in\Pi_{j-1},p_{j-1}(0)=1} \|p_{j-1}(\mathcal{A})\|_2\,\|\mathbf{r}^{(0)}\|_2 \\ &= \min_{p_{j-1}(z)\in\Pi_{j-1},p_{j-1}(0)=1} \|Qp_{j-1}(\Lambda)Q^{-1}\|_2\,\|\mathbf{r}^{(0)}\|_2 \\ &\leqslant \kappa(Q) \min_{p_{j-1}(z)\in\Pi_{j-1},p_{j-1}(0)=1} \max_{\lambda_i\in\lambda(\mathcal{A})} |p_{j-1}(\lambda_i)|\,\|\mathbf{r}^{(0)}\|_2, \qquad (2.24)\end{aligned}$$

with $\kappa(Q) = \|Q\|_2\,\|Q^{-1}\|_2$ the condition number of Q, see [174, Theorem 35.2] or [156, Proposition 6.32].

From the previous expression, we can infer that also for GMRES the convergence of the method can be aided by a clustered eigenvalue distribution of the matrix considered; for this reason, it is useful to consider a preconditioned GMRES when solving general linear systems. We would like to note that the convergence of GMRES is not determined only by a clustered eigenvalue distribution, as the upper bound in (2.24) depends on the condition number of Q (thus, from how far the matrix $\mathcal{A}$ is from normality). In fact, one can show that any non-increasing sequence describes the reduction of the residual at each iteration of the GMRES method applied to solve a linear system; in addition, the matrix can be constructed with any eigenvalue distribution, see [69].

In this case, as we are not concerned anymore with symmetry, the preconditioner $\mathcal{P}$ can itself be non-symmetric. A pseudocode for the GMRES algorithm with right preconditioning is given in Algorithm 13. Clearly, the computational cost of the algorithm grows as the iterations increase, as at each iteration we have to evaluate an orthonormal basis for the current Krylov subspace by the Arnoldi process. In order to save on computational time, GMRES is often restarted after a fixed number of iterations $\bar{\mathtt{j}}$: if the test for convergence is not satisfied after $\bar{\mathtt{j}}$ iterations, the approximation $\mathbf{x}^{(\bar{\mathtt{j}})}$ becomes the initial guess $\mathbf{x}^{(0)}$ of the following cycle. The resulting method is called GMRES($\bar{\mathtt{j}}$) or GMRES with restart.

As formulated in Algorithm 13, the preconditioner $\mathcal{P}$ is fixed at every iteration.

Algorithm 13 Preconditioned GMRES Method to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$, with right preconditioner $\mathcal{P}$

Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
Set $\beta_0 = \|\mathbf{r}^{(0)}\|_2$
Set $\mathbf{q}^{(1)} = \frac{\mathbf{r}^{(0)}}{\beta_0}$
for $j = 1$ **until** convergence, **do**
  Solve $\mathcal{P}\mathbf{z}^{(j)} = \mathbf{q}^{(j)}$
  $\mathbf{w}_1^{(j+1)} = \mathcal{A}\mathbf{z}^{(j)}$
  for $i = 1, 2, \ldots, j$ **do**
    $h_{i,j} = (\mathbf{w}_i^{(j+1)})^\top \mathbf{q}^{(i)}$
    $\mathbf{w}_{i+1}^{(j+1)} = \mathbf{w}_i^{(j+1)} - h_{i,j}\mathbf{q}^{(i)}$
  end for
  $h_{j+1,j} = \|\mathbf{w}_{j+1}^{(j+1)}\|_2$
  $\mathbf{q}^{(j+1)} = \frac{\mathbf{w}^{(j+1)j+1}}{h_{j+1,j}}$
  Find $\mathbf{s}^{(j)}$ that minimizes $\beta_j = \|\beta_0\mathbf{e}_1 - \widehat{H}_j\mathbf{s}^{(j)}\|_2$
  Test for convergence
end for
$Q_j = [\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(j)}]$
Solve $\mathcal{P}\mathbf{z}^{(j)} = Q_j\mathbf{s}^{(j)}$
$\mathbf{x}^{(j)} = \mathbf{x}^{(0)} + \mathbf{z}^{(j)}$

In terms of numerics, this means that, in order to approximate $\mathcal{P}^{-1}$, we have to employ a fixed number of a linear iterations per GMRES step; in particular, it is not possible to employ GMRES (or any other Krylov subspace method) within GMRES as it is. This question led researches to developing a *flexible* version of Krylov subspace methods, that allows the user to modify the preconditioner $\mathcal{P}$ at each iteration. As a consequence, one can employ any Krylov subspace method within a flexible Krylov subspace method. In the following, we will employ the flexible version of GMRES (FGMRES) derived by Saad in [155]. The algorithm is given in Algorithm 14. By comparing Algorithm 14 with Algorithm 13, one can see that the computational cost per iteration is essentially the same, with the additional storage of the vectors $\mathbf{z}^{(i)}, i = 1, 2, \ldots, j$. However, when breakdown occurs in FGMRES (i.e., $h_{j+1,j} = 0$) we are not assured anymore that the approximate solution $\mathbf{x}^{(j)}$ is exact; in fact, for the latter to be true, one needs the additional assumption of the matrix H_j being non-singular, see [155, Proposition 2.2]. A restarted version of this algorithm is also possible, and indeed will be used in the following chapters.

For more discussion about convergence of GMRES, see, for instance, [68, Section 3.2]; for an approximation of the upper bound in (2.24), see, for instance, [155, Section 6.11.4]. For a more comprehensive analysis on flexible, inner–outer Krylov subspace methods, we highlight the text [165].

Algorithm 14 Flexible GMRES Method to solve $\mathcal{A}\mathbf{x} = \mathbf{b}$, with variable right preconditioner $\mathcal{P}^{(j)}$

Choose $\mathbf{x}^{(0)}$
Compute $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$
Set $\beta_0 = \|\mathbf{r}^{(0)}\|_2$
Set $\mathbf{q}^{(1)} = \frac{\mathbf{r}^{(0)}}{\beta_0}$
for $j = 1$ **until** convergence, **do**
  Solve $\mathcal{P}^{(j)}\mathbf{z}^{(j)} = \mathbf{q}^{(j)}$
  $\mathbf{w}_1^{(j+1)} = \mathcal{A}\mathbf{z}^{(j)}$
  for $i = 1, 2, \ldots, j$ **do**
    $h_{i,j} = (\mathbf{w}_i^{(j+1)})^\top \mathbf{q}^{(i)}$
    $\mathbf{w}_{i+1}^{(j+1)} = \mathbf{w}_i^{(j+1)} - h_{i,j}\mathbf{q}^{(i)}$
  end for
  $h_{j+1,j} = \|\mathbf{w}_{j+1}^{(j+1)}\|_2$
  $\mathbf{q}^{(j+1)} = \frac{\mathbf{w}^{(j+1)j+1}}{h_{j+1,j}}$
  Find $\mathbf{s}^{(j)}$ that minimizes $\beta_j = \|\beta_0\mathbf{e}_1 - \widehat{H}_j\mathbf{s}^{(j)}\|_2$
  Test for convergence
end for
$Z_j = [\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(j)}]$
$\mathbf{x}^{(j)} = \mathbf{x}^{(0)} + Z_j\mathbf{s}^{(j)}$

2.10 Preconditioning 2-by-2 Block Matrices

So far we have emphasized the importance of finding a suitable approximation $\mathcal{P}$ of a matrix $\mathcal{A}$ in order to obtain fast convergence of our iterative method. The approximation $\mathcal{P}$ has to be found in such a way that it is spectrally equivalent to $\mathcal{A}$ (meaning that the eigenvalues of the matrix $\mathcal{P}^{-1}\mathcal{A}$ are clustered in a region of the complex plane), with the further property that $\mathcal{P}$ has to be cheaper to invert than $\mathcal{A}$. If $\mathcal{P}$ has those properties, we say that $\mathcal{P}$ is a good preconditioner for $\mathcal{A}$. In this thesis, we study preconditioners for linear systems arising from PDE-constrained optimization problems; in this setting, the matrix $\mathcal{A}$ will present a structure that will be exploited in order to find an optimal preconditioner, namely, it will be of saddle-point type. This section will be thus devoted to preconditioners for general non-symmetric 2-by-2 block matrices, then specifying the preconditioners for the class of saddle-point matrices.

Given an invertible system of the form

$$\mathcal{A} = \begin{bmatrix} \Phi & \Psi_1 \\ \Psi_2 & -\Theta \end{bmatrix} \tag{2.25}$$

with Φ invertible, a good candidate for a preconditioner is the block triangular matrix:

$$\mathcal{P}_1 = \begin{bmatrix} \Phi & 0 \\ \Psi_2 & -S \end{bmatrix}, \tag{2.26}$$

with $S = \Theta + \Psi_2\Phi^{-1}\Psi_1$ the negative Schur complement of $\mathcal{A}$. Indeed, if S is also

invertible, it can be proved that $\lambda(\mathcal{P}_1^{-1}\mathcal{A}) = \{1\}$, see [88, 117]. The latter can be easily derived, as we have

$$\mathcal{P}_1^{-1} = \begin{bmatrix} \Phi^{-1} & 0 \\ S^{-1}\Psi_2\Phi^{-1} & -S^{-1} \end{bmatrix},$$

which implies that

$$\mathcal{P}_1^{-1}\mathcal{A} = \begin{bmatrix} I & \Phi^{-1}\Psi_1 \\ 0 & I \end{bmatrix},$$

where I is the identity matrix of appropriate dimension. As clearly $\mathcal{P}_1^{-1}\mathcal{A}$ is a block-triangular matrix, we can imply that the eigenvalues of $\mathcal{P}_1^{-1}\mathcal{A}$ are given by the eigenvalues of the blocks on its diagonal, that is, we have $\lambda(\mathcal{P}_1^{-1}\mathcal{A}) = \{1\}$. Note that the preconditioned matrix $\mathcal{P}_1^{-1}\mathcal{A}$ is not diagonalizable.

Although in this thesis we will not employ it, it is worth mentioning that the authors in [88, 117] also analysed a preconditioner similar to $\mathcal{P}_1$, that is

$$\mathcal{P}_2 = \begin{bmatrix} \Phi & 0 \\ \Psi_2 & S \end{bmatrix}; \tag{2.27}$$

note that the only difference is the sign in front of S. With this choice of the preconditioner, it can be proved that $\lambda(\mathcal{P}_2^{-1}\mathcal{A}) = \{\pm 1\}$, see [88, 117]. The latter can be derived by employing a similar argument as above. Note that, since both the preconditioners $\mathcal{P}_1$ and $\mathcal{P}_2$ are non-symmetric, the matrices $\mathcal{P}_1^{-1}\mathcal{A}$ and $\mathcal{P}_2^{-1}\mathcal{A}$ will be non-symmetric as well.

From the spectral analysis presented above, it is clear that $\mathcal{P}_1$ and $\mathcal{P}_2$ given in (2.26) and (2.27) respectively could work very well as a preconditioner for general non-symmetric matrices; the question we want to address now is if we can adapt this analysis to the case the matrix $\mathcal{A}$ is of saddle-point type (note that in this case we have $\Psi := \Psi_2 = \Psi_1^\top$). Suppose thus that the system we have to solve is given as in (2.3), with $\Theta \succeq 0$; suppose also that $\Phi \succ 0$. Consider the following block diagonal matrix:

$$\mathcal{P}_3 = \begin{bmatrix} \Phi & 0 \\ 0 & S \end{bmatrix} \tag{2.28}$$

as a preconditioner, with $S = \Theta + \Psi\Phi^{-1}\Psi^\top$; then, supposing also that $S \succ 0$, it is possible to prove that (see [9, 164] and [130, Theorem 4])

$$\lambda(\mathcal{P}_3^{-1}\mathcal{A}) \in \left[-1, \frac{1-\sqrt{5}}{2}\right] \cup \left[1, \frac{1+\sqrt{5}}{2}\right]. \tag{2.29}$$

We note that if $\Theta = 0$ the result above simplifies to $\lambda(\mathcal{P}_3^{-1}\mathcal{A}) = \left\{1, \frac{1\pm\sqrt{5}}{2}\right\}$, see [97, 117]. Finally, due to Φ and S being symmetric positive definite we derive $\mathcal{P}_3 \succ 0$, which in turn implies that the matrix $\mathcal{P}_3^{-1}\mathcal{A}$ is similar to a symmetric matrix, namely, it is similar to $\mathcal{P}_3^{-1/2}\mathcal{A}\mathcal{P}_3^{-1/2}$.

The previous analysis tells us that, if we want to solve a general non-symmetric system $\mathcal{A}$, ideal preconditioners are given by $\mathcal{P}_1$ and $\mathcal{P}_2$ defined as in (2.26) and (2.27) respectively; if we also suppose that $\mathcal{A}$ is of saddle-point type, with $\Phi \succ 0$,

another ideal preconditioner is given by $\mathcal{P}_3$ defined in (2.28). In fact, as described in [88], supposing that $\mathcal{A}$ is invertible, when employing the preconditioners $\mathcal{P}_1$ or $\mathcal{P}_2$ an appropriate iterative method should converge in at most two iterations (in exact arithmetic), as the preconditioned matrix $\mathcal{P}_1^{-1}\mathcal{A}$ has only one as an eigenvalue, but it is not diagonalizable (as we showed above), while the preconditioned matrix $\mathcal{P}_2^{-1}\mathcal{A}$ has two distinct eigenvalues. On the other hand, supposing that $\mathcal{A}$ is invertible and $\Theta = 0$, when employing the preconditioner $\mathcal{P}_3$ an appropriate iterative method should converge in at most three iterations (in exact arithmetic); besides, if $\Theta \geq 0$, from (2.29) we have that the eigenvalues are clustered in two (small) intervals, independently of any parameter involved in the system. Clearly, as the optimal preconditioner we are looking for has also the property of being cheap to invert, we do not want to apply the inverse of $\mathcal{P}_i$, $i = 1, 2, 3$, as defined in (2.26), (2.27), and (2.28) respectively, as the computational cost would be comparable to that of applying the inverse of $\mathcal{A}$. In particular, applying S^{-1} would be problematic, as even when Φ and Θ are sparse S is generally dense. For this reason, optimal preconditioners are given by suitable approximations $\widehat{\mathcal{P}}_i$ of $\mathcal{P}_i$, for $i = 1, 2, 3$, or, more precisely, a cheap application of the effect of $\widehat{\mathcal{P}}_i^{-1}$ on a generic vector. Specifically, the optimal preconditioners that will be studied in this thesis will have the form

$$\widehat{\mathcal{P}}_1 = \begin{bmatrix} \widehat{\Phi} & 0 \\ \Psi_2 & -\widehat{S} \end{bmatrix}, \qquad \widehat{\mathcal{P}}_3 = \begin{bmatrix} \widehat{\Phi} & 0 \\ 0 & \widehat{S} \end{bmatrix}, \tag{2.30}$$

where $\widehat{\Phi}$ and $\widehat{S}$ are "easy-to-invert" approximations of the matrices Φ and S respectively. As the approximation $\widehat{S}$ is usually more difficult to find, we will introduce in the following section a technique that will result in optimal approximations of the Schur complement for a number of problems considered in this thesis.

For more information on the numerical solution of saddle-point systems, see the survey [17]; for more insight on preconditioning, see the surveys [16], [136], and [180].

2.11 Matching Strategy

Suppose we want to solve a system given as in (2.3). In the previous section we presented ideal preconditioners to adopt in order to obtain fast convergence of our iterative method. The common feature of each preconditioner is the application of the inverse operators of the the $(1, 1)$-block Φ and the Schur complement S; however, in order to obtain optimal preconditioners, we will rather consider easy-to-invert approximations $\widehat{\Phi}$ and $\widehat{S}$ of Φ and S respectively. As noted above, the more difficult to find is an approximation $\widehat{S}$ of S that will result in robust convergence with respect to every parameter of the problem; in fact, in the problem we will consider in this thesis the Schur complement S will depend not only on the discretization parameter(s) of space (and time) variables h (and τ), but also on a regularization parameter β within the problem set-up (and, for more complex problems, also on other parameters). Therefore, in this section we introduce

the *matching strategy* for deriving our approximation $\widehat{S}$. This technique was first derived in [140], and has been proved to be powerful in deriving approximations of the Schur complement of linear systems arising from optimal control of PDEs.

Let be $\mathcal{A}$ given as in (2.3). In the following analysis, we will suppose that $\Phi \succ 0$ and $\Theta \succeq 0$. From this assumption, we can write that $\Phi^{\frac{1}{2}}$ and $\Theta^{\frac{1}{2}}$ exist; note that $\Phi^{\frac{1}{2}} \succ 0$ is unique and $\Theta^{\frac{1}{2}} \succeq 0$. Consider now the Schur complement $S = \Theta + \Psi\Phi^{-1}\Psi^\top$, and suppose that $S \succ 0$. From the previous expression and from our assumptions on Φ and Θ, the Schur complement S is the sum of two symmetric positive semi-definite matrices, namely, $\Psi\Phi^{-1}\Psi^\top$ and Θ. As we do not want to construct the matrix S and the approximation we are looking for has to be easy to invert, we will be considering the following matrix:

$$\widehat{S} = (\Psi + \widehat{\Lambda})\Phi^{-1}(\Psi + \widehat{\Lambda})^\top \approx S \tag{2.31}$$

as an approximation, with the matrix $\widehat{\Lambda}$ such that $\widehat{S}$ 'captures' both terms of S. As the term $\Psi\Phi^{-1}\Psi^\top$ already appears in the definition (2.31) of $\widehat{S}$, we wish that the matrix $\widehat{\Lambda}$ is such that

$$\widehat{\Lambda}\Phi^{-1}\widehat{\Lambda}^\top = \Theta. \tag{2.32}$$

Noting that

$$\Phi^{-1} = \Phi^{-\frac{1}{2}}\Phi^{-\frac{1}{2}} = \Phi^{-\frac{1}{2}}(\Phi^{-\frac{1}{2}})^\top,$$
$$\Theta = \Theta^{\frac{1}{2}}\Theta^{\frac{1}{2}} = \Theta^{\frac{1}{2}}(\Theta^{\frac{1}{2}})^\top,$$

we can rewrite (2.32) as

$$\widehat{\Lambda}\Phi^{-\frac{1}{2}}(\widehat{\Lambda}\Phi^{-\frac{1}{2}})^\top = \Theta^{\frac{1}{2}}(\Theta^{\frac{1}{2}})^\top;$$

finally, we want that the matrix $\widehat{\Lambda}$ is such that

$$\widehat{\Lambda}\Phi^{-\frac{1}{2}} = \Theta^{\frac{1}{2}},$$

that is

$$\widehat{\Lambda} = \Theta^{\frac{1}{2}}\Phi^{\frac{1}{2}}. \tag{2.33}$$

We emphasize here that the choice of introducing $\widehat{\Lambda}^\top$ in the definition (2.31) of $\widehat{S}$ has been made only to keep the approximation symmetric; note also that $\widehat{S} \succ 0$ (if $\Psi + \widehat{\Lambda}$ has full rank, which will be the case for the settings in which we apply this approximation).

Now that we have an explicit expression for our approximation, we would like to understand how potent it is, that is, we want to study the spectrum of the matrix $\widehat{S}^{-1}S$. Since the matrices S and $\widehat{S}$ are both symmetric positive definite, we may use the (generalized) Rayleigh quotient in order to find upper and lower bounds for the eigenvalues of the matrix $\widehat{S}^{-1}S$, as done in [141], for instance. Let be λ an eigenvalue of $\widehat{S}^{-1}S$ with $\mathbf{x}$ the corresponding eigenvector; then

$$\widehat{S}^{-1}S\mathbf{x} = \lambda\mathbf{x} \;\Rightarrow\; S\mathbf{x} = \lambda\widehat{S}\mathbf{x} \;\Rightarrow\; \lambda = \frac{\mathbf{x}^\top S\mathbf{x}}{\mathbf{x}^\top\widehat{S}\mathbf{x}} =: R.$$

Thus upper and lower bounds for R will be also upper and lower bounds for the

eigenvalues of $\widehat{S}^{-1}S$ respectively. Before starting our analysis, we rewrite the generalized Rayleigh quotient R as

$$R := \frac{\mathbf{x}^\top S\mathbf{x}}{\mathbf{x}^\top \widehat{S}\mathbf{x}} = \frac{\mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b}}{\mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b} + \mathbf{a}^\top\mathbf{b} + \mathbf{b}^\top\mathbf{a}},$$

where $\mathbf{a} = (\Psi\Phi^{-\frac{1}{2}})^\top\mathbf{x}$ and $\mathbf{b} = (\Theta^{\frac{1}{2}})^\top\mathbf{x}$. Since $S \succ 0$ and $\widehat{S} \succ 0$, we have

$$\begin{aligned} R \geqslant \frac{1}{2} \quad &\Leftrightarrow \quad \mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b} \geqslant \frac{1}{2}\left(\mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b} + \mathbf{a}^\top\mathbf{b} + \mathbf{b}^\top\mathbf{a}\right) \\ &\Leftrightarrow \quad \frac{1}{2}\left(\mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b} - \mathbf{a}^\top\mathbf{b} - \mathbf{b}^\top\mathbf{a}\right) \geqslant 0 \\ &\Leftrightarrow \quad \frac{1}{2}\left(\mathbf{a} - \mathbf{b}\right)^\top\left(\mathbf{a} - \mathbf{b}\right) \geqslant 0, \end{aligned}$$

which is clearly satisfied since $\left(\mathbf{a} - \mathbf{b}\right)^\top\left(\mathbf{a} - \mathbf{b}\right) = \|\mathbf{a} - \mathbf{b}\|^2$.

We have then proved the following theorem:

Theorem 1. *Let $\mathcal{A}$ be defined as in* (2.3)*. Let be S the Schur complement of $\mathcal{A}$, and let be $\widehat{S}$ be defined as in* (2.31)*, with $\widehat{\Lambda}$ defined in* (2.32)*. If S and $\widehat{S}$ are symmetric positive definite, then $\frac{1}{2}$ is a lower bound for the eigenvalues of the matrix $\widehat{S}^{-1}S$.*

From Theorem 1, we have that the minimum eigenvalue of the matrix $\widehat{S}^{-1}S$ is bounded away from zero. However, this is usually not enough to ensure effective convergence of an iterative solver, as the convergence process depends on the whole spectrum of the matrix, and if we have that the upper bound on the eigenvalues is, for instance, of order $\mathcal{O}(10^4)$ the process may struggle to converge in few iterations. We are therefore interested in finding also an upper bound for the spectrum of the matrix $\widehat{S}^{-1}S$. In order to do so, we need a further assumption on the matrix $\widehat{\Lambda}$; specifically, we have to assume that the mixed term $\widehat{\Lambda}\Phi^{-1}\Psi^\top + \Psi\Phi^{-1}\widehat{\Lambda}^\top$ is positive semi-definite. Under this assumption, we can derive

$$\begin{aligned} \mathbf{a}^\top\mathbf{b} + \mathbf{b}^\top\mathbf{a} \geqslant 0 \quad &\Leftrightarrow \quad \mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b} + \mathbf{a}^\top\mathbf{b} + \mathbf{b}^\top\mathbf{a} \geqslant \mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b} \\ &\Leftrightarrow \quad R \leqslant 1. \end{aligned}$$

Note that, if $\widehat{\Lambda}\Phi^{-1}\Psi^\top + \Psi\Phi^{-1}\widehat{\Lambda}^\top$ is not positive semi-definite, the upper bound will be greater than 1; in particular, the more the matrix $\widehat{\Lambda}\Phi^{-1}\Psi^\top + \Psi\Phi^{-1}\widehat{\Lambda}^\top$ is indefinite, the greater the upper bound will be. From here, the spectral properties of $\widehat{\Lambda}\Phi^{-1}\Psi^\top + \Psi\Phi^{-1}\widehat{\Lambda}^\top$ represent a way to measure the approximation adopted.

We have then proved the following theorem:

Theorem 2. *Let $\mathcal{A}$ be defined as in* (2.3)*. Let be S the Schur complement of $\mathcal{A}$, and let be $\widehat{S}$ be defined as in* (2.31)*, with $\widehat{\Lambda}$ defined in* (2.32)*. If S and $\widehat{S}$ are symmetric positive definite, and the matrix $\widehat{\Lambda}\Phi^{-1}\Psi^\top + \Psi\Phi^{-1}\widehat{\Lambda}^\top$ is positive semi-definite, then 1 is an upper bound for the eigenvalues of the matrix $\widehat{S}^{-1}S$.*

Under the assumption that the mixed term $\widehat{\Lambda}\Phi^{-1}\Psi^\top + \Psi\Phi^{-1}\widehat{\Lambda}^\top$ is positive semi-definite, we have that the matrix $\widehat{S}$ defined in (2.31), with $\widehat{\Lambda}$ defined in

(2.32), is an optimal approximation of the Schur complement S. In fact, the spectrum of the matrix $\widehat{S}^{-1}S$ is clustered in the interval $[\frac{1}{2}, 1]$, independently of any parameter involved in the problem under examination. For this reason, we will adopt the matching strategy for devising optimal preconditioners for the problem considered in this thesis. As the $(1,1)$- and the $(2,2)$-blocks of the system under examination will satisfy the assumptions in Theorem 1 and Theorem 2, in the following working of this thesis one key concern will be proving that the mixed term $\widehat{\Lambda}\Phi^{-1}\Psi^\top + \Psi\Phi^{-1}\widehat{\Lambda}^\top$ is positive semi-definite.

We would like to mention that the approximation $\widehat{S}$ defined in (2.31) can be generalized also to non-symmetric matrices. Suppose in fact that we want to solve a system with $\mathcal{A}$ defined as in (2.25). Then, supposing that $\Phi \succ 0$ and $\Theta \succeq 0$, we would like to use as an approximation of the Schur complement $S = \Theta + \Psi_2\Phi^{-1}\Psi_1$ the matrix

$$\widehat{S} = (\Psi_1 + \widehat{\Lambda}_1)\Phi^{-1}(\Psi_2 + \widehat{\Lambda}_2),$$

with $\widehat{\Lambda}_1$ and $\widehat{\Lambda}_2$ such that

$$\widehat{\Lambda}_1\Phi^{-1}\widehat{\Lambda}_2 = \Theta.$$

Although this type of approximation gives more freedom as to the choice of the matrices $\widehat{\Lambda}_1$ and $\widehat{\Lambda}_2$, finding bounds on the eigenvalue becomes a more complex task as, due to S and $\widehat{S}$ being non-symmetric, the eigenvalues of the matrix $\widehat{S}^{-1}S$ will in general be complex.

In order to show how the proposed approximation works within a preconditioner of the form (2.26) or of the form (2.28), in the following section we will show some numerical results for the Poisson control problem.

2.11.1 Preconditioning for the Poisson Control Problem

Let us consider the distributed Poisson control problem (1.5)–(1.6). By employing finite elements with the same finite element basis used for the state v and the control u, after either applying an optimize-then-discretize strategy or a discretize-then-optimize approach, one has to solve the saddle-point system given in (1.13).

As discussed above, optimal preconditioners are the block triangular and block diagonal matrices $\mathcal{P}_1$ and $\mathcal{P}_3$ defined in (2.26) and (2.28) respectively. In the case of solving the discrete optimality conditions (1.13) of the Poisson control problem, they read as follows:

$$\mathcal{P}_1 = \begin{bmatrix} M & 0 \\ K & -S_{\text{PC}} \end{bmatrix}, \qquad \mathcal{P}_3 = \begin{bmatrix} M & 0 \\ 0 & S_{\text{PC}} \end{bmatrix},$$

with $S_{\text{PC}} = KM^{-1}K + \frac{1}{\beta}M$ the Schur complement of the system under examination, with M and K the mass and the stiffness matrix in the chosen finite element basis. Note that S_{PC} is symmetric positive definite. Again, rather than using the matrices above as a preconditioner, we are looking for approximations of the relevant blocks M and S_{PC}. We first find an approximation of the $(1,1)$-block M, then employ the matching strategy to devise an approximation of S_{PC}, as done in [140].

As discussed in Section 2.4, a cheap and effective way of approximately applying the inverse of a mass matrix to a generic vector is employing the Chebyshev semi-iterative method. More specifically, our approximation of M is given by M_c, with M_c a fixed number of steps of Chebyshev semi-iteration applied to M.

Let us now focus on finding an approximation $\widehat{S}_{\mathrm{PC}}$ of S_{PC}. From the convenient structure of S_{PC}, we employ the matching strategy described above in order to do so. We can easily derive that the approximation we seek is given by

$$\widehat{S}_{\mathrm{PC}} = (K + \widehat{\Lambda}_{\mathrm{PC}})M^{-1}(K + \widehat{\Lambda}_{\mathrm{PC}}),$$

where, by employing (2.33) and recalling that $M \succ 0$, the matrix $\widehat{\Lambda}_{\mathrm{PC}}$ is given by

$$\widehat{\Lambda}_{\mathrm{PC}} = \frac{1}{\sqrt{\beta}} M^{\frac{1}{2}} M^{\frac{1}{2}} = \frac{1}{\sqrt{\beta}} M.$$

Note that in the expression above we exploited the symmetry of K and M. Note also that $\widehat{S}_{\mathrm{PC}}$ is symmetric positive definite.

In order to understand how good the approximation $\widehat{S}_{\mathrm{PC}}$ is, we need to study the spectrum of the preconditioned Schur complement $\widehat{S}_{\mathrm{PC}}^{-1} S_{\mathrm{PC}}$. We will employ Theorem 1 and Theorem 2 to find lower and upper bounds for the spectrum of $\widehat{S}_{\mathrm{PC}}^{-1} S_{\mathrm{PC}}$.

Since both S_{PC} and $\widehat{S}_{\mathrm{PC}}$ are symmetric positive definite, from Theorem 1 we have that $\frac{1}{2}$ is a lower bound for the eigenvalues of $\widehat{S}_{\mathrm{PC}}^{-1} S_{\mathrm{PC}}$. In order to find an upper bound, we need to consider the mixed term $\frac{1}{\sqrt{\beta}} K + \frac{1}{\sqrt{\beta}} K = \frac{2}{\sqrt{\beta}} K$, which is clearly symmetric positive semi-definite in general (e.g., Neumann boundary conditions). Therefore, from Theorem 2 we infer that 1 is an upper bound for the eigenvalues of the matrix $\widehat{S}_{\mathrm{PC}}^{-1} S_{\mathrm{PC}}$. Finally, we have that $\lambda(\widehat{S}_{\mathrm{PC}}^{-1} S_{\mathrm{PC}}) \in [\frac{1}{2}, 1]$.

We can now write the approximation of the preconditioners $\mathcal{P}_1$ and $\mathcal{P}_3$ as

$$\widehat{\mathcal{P}}_1 = \begin{bmatrix} M_c & 0 \\ K & -\widehat{S}_{\mathrm{PC}} \end{bmatrix}, \qquad \widehat{\mathcal{P}}_3 = \begin{bmatrix} M_c & 0 \\ 0 & \widehat{S}_{\mathrm{PC}} \end{bmatrix},$$

with M_c and $\widehat{S}_{\mathrm{PC}}$ defined as above. The preconditioners $\widehat{\mathcal{P}}_1$ and $\widehat{\mathcal{P}}_3$ are optimal, as the approximation of the relevant blocks is independent of any parameter involved in the problem, including the mesh-size. Therefore, we expect a suitable preconditioned iterative method to converge in a (roughly) constant number of iterations. However, we would like to recall that for GMRES the convergence is not determined only by a clustered eigenvalues distribution, as we discussed above. In order to show the robustness of the solver, we will consider a simple numerical example.

Let consider the Poisson control problem defined in (1.5)–(1.6) with $d = 2$, $\Omega = (-1, 1)^2$, $f = 0$. Suppose the boundary condition is given by

$$v(x_1, x_2) = 1, \quad (x_1, x_2) \in \partial\Omega,$$

and the desired state is given by

$$v_d(x_1, x_2) = \cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right) + 1.$$

We implement a finite element method, using $\mathbf{Q}_1$ basis functions for state, control, and adjoint variables. In all our tests, we consider a (spatial) uniform grid of mesh-size $h = 2^{1-\mathtt{l}}$, with `l` the level of grid refinement. When approximating the $(1,1)$-block, we apply 20 steps of Chebyshev semi-iteration to M. For the approximation of the Schur complement, we approximate $K + \widehat{\Lambda}_{\mathrm{PC}}$ with 2 V-cycles of the `HSL_MI20` algebraic multigrid solver [22]. We employ MATLAB's preconditioned GMRES in conjuction with the preconditioner $\widehat{\mathcal{P}}_1$ defined above, and also employ preconditioned MINRES with the preconditioner $\widehat{\mathcal{P}}_3$ defined above. We run the solvers until a tolerance of 10^{-6} on the relative residual is reached. The iteration count for all tests presented starts from 0. In Table 2.1 we report the results for preconditioned GMRES, while in Table 2.2 we report the results for preconditioned MINRES. All tests are run on MATLAB R2018b, using a 1.70GHz Intel quad-core i5 processor and 8 GB RAM on an Ubuntu 18.04.1 LTS operating system.

Table 2.1: Poisson control problem: GMRES iterations required, with preconditioner $\widehat{\mathcal{P}}_1$, for a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
3	5	0.036	6	0.033	7	0.032	7	0.045	6	0.041
4	5	0.033	6	0.042	7	0.043	7	0.045	7	0.041
5	5	0.062	6	0.065	6	0.080	7	0.070	7	0.069
6	4	0.17	5	0.17	6	0.19	6	0.19	7	0.21
7	4	1.00	5	0.78	5	0.68	6	0.86	6	0.91
8	4	2.06	4	2.41	5	2.73	5	2.69	6	1.95

Table 2.2: Poisson control problem: MINRES iterations required, with preconditioner $\widehat{\mathcal{P}}_3$, for a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
3	8	0.015	10	0.014	13	0.020	15	0.021	16	0.035
4	10	0.023	12	0.027	14	0.027	18	0.033	19	0.034
5	10	0.041	12	0.048	15	0.060	18	0.074	21	0.083
6	10	0.15	12	0.17	15	0.21	18	0.26	21	0.29
7	10	0.63	12	0.77	15	0.90	18	1.08	21	1.24
8	10	2.78	12	3.27	15	4.09	18	4.78	21	5.57

As shown in Tables 2.1–2.2, the preconditioners derived in this section behave robustly with respect to all the parameters involved in the problems. In fact, the number of iterations required for both solvers is parameter-robust, with preconditioned GMRES converging in at most 7 iterations, while preconditioned MINRES converges in at most 21 iterations. Further, the CPU times scale linearly with the problem size.

As we will see in the following chapters, the preconditioning techniques described above will be very useful for deriving robust solvers for the more complex PDE-constrained optimization problems considered in this work.

Chapter 3

Preconditioning the Heat Control Problem with Crank–Nicolson Discretization in Time

"When the student is ready, the teacher will appear."
– Proverb

Starting from this chapter, we will be deriving and describing the main contributions of this thesis, namely parameter-robust preconditioning techniques for distributed optimal control problems with time-dependent PDEs as constraints (although in Chapter 5 and Chapter 6 we will also be dealing with stationary problems). We begin with the easiest in this class of problems, that is, the optimal control of the heat equation, or simply *heat control*. The content of this chapter is based on some of the work in [100].

Due to the complex structure and high dimensionality of time-dependent PDE-constrained optimization problems when an accurate discrete solution is sought, preconditioned iterative solvers have been employed for the 'all-at-once' resolution of such formulations, see for example [152, 162] for early work in this direction. It is worth noting that the all-at-once approach suffers of memory limitations, as one has to store global-in-time solutions. In particular, for very fine discretizations it may be prohibitive to even store the right-hand side of the linear system to be solved. For this reason, an alternative approach may be employing a gradient-type method, that solves simpler problems until an accurate enough solution is found, see for instance [67, 78]. An alternative approach may be employing a low-rank approximation of the problem to solve, see for example [40, 166, 167, 168]. For an overview of other possible approaches, we refer the reader to [36, 86].

When preconditioners are sought for certain time-dependent problems, it is often preferable to apply a (first-order accurate) backward Euler method in time, as this not only avoids restrictions on the time step τ, but also leads to particularly convenient structures within the matrix and facilitates effective preconditioning; see [139] for a mesh- and β-robust preconditioner for the heat control problem, and [40, 138, 167, 169, 186] for applications to different problems. While the natural choice $\tau = \mathcal{O}(h^2)$ would be required to keep the approximation order

proportional to h^2, with h the mesh-size in space, this would lead to problems of huge magnitude. The question we wish to investigate here is whether applying a higher-order Crank–Nicolson method in time is beneficial, due to the reduced number of time steps (which in turn leads to a reduced dimension of the linear system) required to obtain a similar discretization error. The challenge here is that the much more complex structure of the resulting linear system makes preconditioning a highly non-trivial task, so a more sophisticated numerical method and preconditioning strategy need to be devised to achieve fast and robust convergence of the iterative solver.

This chapter is structured as follows. In Section 3.1 we introduce the problems we consider, arising from the heat control formulation. In Section 3.2 we describe the linear systems obtained upon discretization of the first-order optimality conditions, and in particular demonstrate a suitable transformation which allows symmetrization of the linear system obtained from the Crank–Nicolson method. This enables us to apply a symmetric iterative solver such as MINRES [128], which is highly desirable from the perspective of proving convergence of the iterative method. In Section 3.3 we derive our proposed new preconditioner, using saddle-point theory along with suitable approximations of the $(1,1)$-block and Schur complement, and provide eigenvalue results for the preconditioned linear system. In Section 3.4 we benchmark our method against the widely-used backward Euler method, coupled with the preconditioner derived in [139], and demonstrate our new preconditioner's efficiency and robustness with respect to all parameters involved in the heat control problem.

3.1 Problem Formulation

As above, in this chapter we consider the fast and robust numerical solution of time-dependent PDE-constrained optimization problems. In particular, we examine distributed heat control problems of the form:

$$\min_{v,u} \quad J(v,u) = \frac{1}{2}\int_0^{t_f}\int_\Omega |v(x,t)-v_d(x,t)|^2 \,\mathrm{d}\Omega\mathrm{d}t + \frac{\beta}{2}\int_0^{t_f}\int_\Omega |u(x,t)|^2 \,\mathrm{d}\Omega\mathrm{d}t \quad (3.1)$$

subject to

$$\begin{cases} \dfrac{\partial v}{\partial t} - \nabla^2 v = u + f(x,t) & \text{in } \Omega\times(0,t_f), \\ v(x,t) = g_D(x,t) & \text{on } \partial\Omega_D\times(0,t_f), \\ \dfrac{\partial v}{\partial \vec{n}}(x,t) = g_N(x,t) & \text{on } \partial\Omega_N\times(0,t_f), \\ v(x,0) = v_0(x) & \text{in } \Omega, \end{cases} \quad (3.2)$$

where the variables v, v_d, and u are the state, desired state, and control variables, respectively, and $\beta > 0$ is a regularization parameter. The problem is solved on a spatial domain $\Omega \subset \mathbb{R}^d$, $d \in \{1,2,3\}$, up to a final time $t_f > 0$, that is $(x,t) \in \Omega \times (0,t_f)$. In addition, the boundary is such that $\overline{\partial\Omega} = \overline{\partial\Omega}_D \cup \overline{\partial\Omega}_N$, $\partial\Omega_D \cap \partial\Omega_N = \varnothing$. Here $\dfrac{\partial v}{\partial \vec{n}}(x,t)$ represents the (outward) normal derivative of v

on $\partial\Omega_N$. The functions f, g_D, g_N, and v_0 are known.

This problem provides a valuable benchmark of our method against widely-used preconditioned iterative solution with a backward Euler method in time [139]. For simplicity the working of this chapter considers Dirichlet boundary conditions, that is $\partial\Omega_D = \partial\Omega$, but we note that our method can be readily generalized to heat control problems with Neumann and mixed boundary conditions.

3.2 First-Order Optimality Conditions and Discretizations in Time

We now describe the strategy used for obtaining an approximate solution of (3.1)–(3.2). We apply an all-at-once approach coupled with the optimize-then-discretize scheme, in which the continuous Lagrangian is used to arrive at first-order optimality conditions, which are then discretized. Introducing the adjoint variable (or Lagrange multiplier) ζ, we consider the Lagrangian associated to (3.1)–(3.2) as in [139]. Then, by deriving the Karush–Kuhn–Tucker (KKT) conditions, the solution of (3.1)–(3.2) satisfies:

$$\left\{\begin{array}{rll} \dfrac{\partial v}{\partial t} - \nabla^2 v = \dfrac{1}{\beta}\,\zeta + f & \text{in } \Omega \times (0,t_f) & \\ v(x,t) = g(x,t) & \text{on } \partial\Omega \times (0,t_f) & \text{state} \\ v(x,0) = v_0(x) & \text{in } \Omega & \text{equation} \\ -\dfrac{\partial \zeta}{\partial t} - \nabla^2 \zeta \;=\; v_d - v & \text{in } \Omega \times (0,t_f) & \\ \zeta(x,t) = 0 & \text{on } \partial\Omega \times (0,t_f) & \text{adjoint} \\ \zeta(x,t_f) = 0 & \text{in } \Omega & \text{equation} \end{array}\right. \tag{3.3}$$

where we have substituted the gradient equation $\beta u - \zeta = 0$ into the state equation.

Before moving on with our exposition, we would like to devote some attention to the derivation of the KKT conditions above. Specifically, we would like to show how the initial time condition on the state variable v becomes a final time condition on the adjoint variable ζ. The state and the gradient equations can be derived similarly as in Section 1.3.1.

As in Section 1.3.1, we introduce an adjoint variable for each constraint in (3.2) and consider the Lagrangian associated to (3.1)–(3.2)

$$\begin{aligned} \mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega},\zeta_0) :=& \frac{1}{2}\int_0^{t_f} \|v(x,t) - v_d(x,t)\|^2_{L^2(\Omega)}\mathrm{d}t + \frac{\beta}{2}\int_0^{t_f} \|u(x,t)\|^2_{L^2(\Omega)}\mathrm{d}t \\ &+ \int_0^{t_f}\left(\frac{\partial v}{\partial t} - \nabla^2 v - u - f, \zeta_\Omega\right)\mathrm{d}t + \int_0^{t_f}\int_{\partial\Omega}(v-g)\zeta_{\partial\Omega}\mathrm{d}s\mathrm{d}t \\ &+ \int_\Omega (v(x,0) - v_0)\zeta_0 \mathrm{d}\Omega, \end{aligned}$$

where $(\cdot,\cdot)$ is the L^2-inner product in Ω. As we are interested in deriving the adjoint equation, we will consider only the Fréchet derivative of $\mathcal{L}$ with respect to v. In particular, we consider a generic direction w in an appropriate Hilbert space and then write the Fréchet derivative of the term

$$\int_0^{t_f} \left(\frac{\partial (v+w)}{\partial t}, \zeta_\Omega \right) \mathrm{d}t.$$

Integrating $\dfrac{\partial w}{\partial t}$ by parts with respect to t, one can write

$$\begin{aligned}\int_0^{t_f} \left(\frac{\partial (v+w)}{\partial t}, \zeta_\Omega \right) \mathrm{d}t = &\int_0^{t_f} \left(\frac{\partial v}{\partial t}, \zeta_\Omega \right) \mathrm{d}t - \int_0^{t_f} \left(w, \frac{\partial \zeta_\Omega}{\partial t} \right) \mathrm{d}t \\ &+ (w(x,t_f), \zeta_\Omega(x,t_f)) - (w(x,0), \zeta_\Omega(x,0)).\end{aligned}$$

Then, following the strategy employed in Section 1.3.1, one can easily derive that the Fréchet derviative of $\mathcal{L}$ with respect to v is given by

$$\begin{aligned}\mathrm{d}_v \mathcal{L}(v,u,\zeta_\Omega,\zeta_{\partial\Omega},\zeta_0) = &\int_0^{t_f} \left(v - v_d - \frac{\partial \zeta_\Omega}{\partial t} - \nabla^2 \zeta_\Omega, w \right) \mathrm{d}t + \int_0^{t_f} \int_{\partial\Omega} \zeta_{\partial\Omega} w \, \mathrm{d}s\mathrm{d}t \\ &- \int_0^{t_f} \int_{\partial\Omega} \frac{\partial w}{\partial \vec{n}} \zeta_\Omega \, \mathrm{d}s\mathrm{d}t + \int_0^{t_f} \int_{\partial\Omega} \frac{\partial \zeta_\Omega}{\partial \vec{n}} w \, \mathrm{d}s\mathrm{d}t \\ &+ (\zeta_\Omega(x,t_f), w(x,t_f)) - (\zeta_\Omega(x,0), w(x,0)) + (\zeta_0, w).\end{aligned}$$

Setting the above expression equal to zero and choosing w appropriately as in Section 1.3.1, one can easily obtain the adjoint equation as defined in (3.3). In particular, the L^2-inner product $(\zeta_\Omega(x,t_f), w(x,t_f))$ has to be zero for all directions w, implying that $\zeta_\Omega(x,t_f) = 0$, that is, we have derived the final condition on the adjoint ζ in (3.3). Finally, as done for the Poisson control problem in Section 1.3.1, one can derive that, on the boundary $\partial\Omega$, the normal derivative of the adjoint variable ζ_Ω is equal to $-\zeta_{\partial\Omega}$, and that its initial condition $\zeta_\Omega(x,0)$ is equal to ζ_0. For this reason, we may only consider one adjoint variable ζ for the whole problem.

Problem (3.3) is a coupled system of (time-dependent) PDEs, consisting of a forward PDE combined with a backward problem for the adjoint. Due to this structure, when trying to find numerical approximation of the solution, an A–stable method is usually applied for discretizing the time derivative, since this will allow the user to choose any time step. In particular, the time step may be independent of the spatial mesh-size. Further, in order to obtain a consistent system of linear equations, both functions v and ζ are approximated at the same time points.

We now introduce methods for approximating the time derivative when solving (3.3), and describe the resulting linear systems, starting with the widely-used backward Euler method in Section 3.2.1, followed by the Crank–Nicolson method in Section 3.2.2. For the remainder of the chapter, we discretize the interval $(0, t_f)$ into n_t subintervals of length $\tau = \frac{t_f}{n_t}$, and we use the notation $\boldsymbol{v}_n \approx v(x,t_n)$, $\boldsymbol{\zeta}_n \approx \zeta(x,t_n)$ for our approximations for all $x \in \Omega$, with $t_n = n\tau$. Further, we will

employ the same finite element basis $\{\phi_i\}_{i=1}^{n_x}$ for the state v, the control u, and the adjoint ζ, with $n_x \in \mathbb{N}$.

3.2.1 Backward Euler Discretization

Many widely-used preconditioned iterative methods for the solution of time-dependent PDE-constrained optimization problems of type (3.1)–(3.2) involve a backward Euler discretization for the time variable [40, 138, 139, 167, 169, 186]. Applying this scheme gives the following system of equations:

$$\begin{cases} \tau M \boldsymbol{v}_n + L^{\mathrm{BE}} \boldsymbol{\zeta}_n - M \boldsymbol{\zeta}_{n+1} = \tau M \boldsymbol{v}_d^n, & n = 0, 1, ..., n_t - 1, \\ \boldsymbol{\zeta}_{n_t} = \mathbf{0}, \\ \boldsymbol{v}_0 = \boldsymbol{v}^0, \\ -M \boldsymbol{v}_n + L^{\mathrm{BE}} \boldsymbol{v}_{n+1} - \dfrac{\tau}{\beta} M \boldsymbol{\zeta}_{n+1} = \tau \boldsymbol{f}^{n+1}, & n = 0, 1, ..., n_t - 1, \end{cases}$$

where $L^{\mathrm{BE}} = \tau K + M$, $\boldsymbol{v}^0$ is the discretization of the initial condition for v, and

$$\boldsymbol{f}^n = \{f_i^n\}_{i=1}^{n_x}, \qquad f_i^n = \int_\Omega f(x, t_n) \phi_i \, \mathrm{d}\Omega \, . \tag{3.4}$$

Here, K and M are the stiffness and mass matrix in the chosen finite element basis, respectively. With this notation, we obtain the following (symmetric) linear system:

$$\begin{bmatrix} \Phi^{\mathrm{BE}} & (\Psi^{\mathrm{BE}})^\top \\ \Psi^{\mathrm{BE}} & -\Theta^{\mathrm{BE}} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{v}} \\ \bar{\boldsymbol{\zeta}} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^{\mathrm{BE}} \\ \mathbf{b}_2^{\mathrm{BE}} \end{bmatrix}, \tag{3.5}$$

where

$$\begin{aligned} \Phi^{\mathrm{BE}} &= \begin{bmatrix} \tau M & & & \\ & \ddots & & \\ & & \tau M & \\ & & & 0 \end{bmatrix}, \quad \Psi^{\mathrm{BE}} = \begin{bmatrix} L^{\mathrm{BE}} & & & \\ -M & L^{\mathrm{BE}} & & \\ & \ddots & \ddots & \\ & & -M & L^{\mathrm{BE}} \end{bmatrix}, \\ \Theta^{\mathrm{BE}} &= \begin{bmatrix} 0 & & & \\ & \frac{\tau}{\beta} M & & \\ & & \ddots & \\ & & & \frac{\tau}{\beta} M \end{bmatrix}, \quad \begin{aligned} \bar{\boldsymbol{v}} &= \begin{bmatrix} (\boldsymbol{v}_0)^\top & (\boldsymbol{v}_1)^\top & \dots & (\boldsymbol{v}_{n_t})^\top \end{bmatrix}^\top, \\ \bar{\boldsymbol{\zeta}} &= \begin{bmatrix} (\boldsymbol{\zeta}_0)^\top & (\boldsymbol{\zeta}_1)^\top & \dots & (\boldsymbol{\zeta}_{n_t})^\top \end{bmatrix}^\top, \end{aligned} \end{aligned} \tag{3.6}$$

and the right-hand side vectors $\mathbf{b}_1^{\mathrm{BE}}$ and $\mathbf{b}_2^{\mathrm{BE}}$ take into account the initial and final-time conditions on v and ζ, as well as information from the desired state v_d and the force function f. Note that we have symmetrized the system by replacing the discretized initial and final-time conditions with

$$(L^{\mathrm{BE}})^\top \boldsymbol{\zeta}_{n_t} = \mathbf{0}, \qquad L^{\mathrm{BE}} \boldsymbol{v}_0 = L^{\mathrm{BE}} \boldsymbol{v}^0.$$

The structure of the previous system is very convenient from the point of view of numerical linear algebra because it facilitates effective preconditioning.

3.2.2 Crank–Nicolson Discretization and Symmetrization of the System

Despite the convenient structure of the linear system arising from the backward Euler method, the essential drawback is that the method is only first-order accurate in time. For instance, if a second-order accurate method is used for the spatial discretization, the numerical error is of order $\mathcal{O}(\tau) + \mathcal{O}(h^2)$, given sufficient smoothness properties of the solution. Therefore, it is reasonable to choose $\tau = \mathcal{O}(h^2)$. However, this leads to a large number of time-steps and hence matrix blocks within the discretized problem, which will result in a linear system of huge dimension, and hence an extremely high CPU time required for its solution. In recent years, a significant effort has been invested in improving the accuracy of the discretized solution of time dependent PDE-constrained optimization problems involving the backward Euler method: see [73] for an application of deferred correction to time-dependent PDE-constrained optimization problems, [40, 166, 167, 168] for low-rank tensor approaches to speed up the convergence of backward Euler, [84] for a multigrid technique applied to optimal flow control problems, and [187] for a preconditioned iterative solver for problem of the type (3.1)–(3.2) that uses a reduction in dimensionality of the arising system.

We also highlight recent works in which higher-order time discretizations are considered. For example, in [67] the authors derive a parallel-in-time algorithm coupled with a gradient-based method, using a Lobatto IIIA scheme in the time variable, and in [40] the authors note that their low-rank method can also be adapted to the Crank–Nicolson approach. We also point out [102], in which the authors employ a leapfrog finite difference scheme in time for solving parabolic optimal control problems, prove that second-order convergence is achieved and that the method is unconditionally stable, then derive a parameter-robust multigrid solver for the discretized system. Other valuable approaches for addressing time-dependent PDE-constrained optimization problems include multiple shooting methods [78], parareal schemes [105, 109], and ideas from instantaneous control [34, 83]. However, to the best of our knowledge, the crucial question of preconditioning PDE-constrained optimization methods by exploiting the precise matrix structures arising from higher-order discretization schemes in time has not been resolved, perhaps due to the increased complexity in the structure of the linear systems, an issue pointed out in [84] for instance. In this work, we aim to reduce the dimensionality of the system which needs to be solved in order to obtain a fixed accuracy for certain time-dependent PDE-constrained optimization problems, by devising a fast and effective preconditioner for the linear system arising from a Crank–Nicolson discretization, and hence reducing the number of time-steps which need to be taken.

Remark 1. *We note here that both backward Euler and Crank–Nicolson are unconditionally A–stable, that is, no restriction on τ and h is required. There are often other stability properties to consider when selecting an appropriate time-stepping scheme: for instance Crank–Nicolson, as opposed to backward Euler, is not L–stable, which is an important consideration when applying long-range integrations. For instance, in fluid dynamics one may be interested in integrating*

the problem until the physics reaches a steady state. In this work, we do not address specific questions about stability properties of the two methods, but rather whether it is possible, in principle, to take advantage of a higher-order discretization scheme in the time variable for increasing the rate of convergence of solvers for the optimal control of time-dependent PDEs. It is likely that our proposed method could be extended to time-stepping methods that achieve even faster convergence than Crank–Nicolson, including L–stable methods. We highlight the reference [7], which considers the question of preconditioning linear systems arising from L–stable methods for forward evolutionary PDEs.

Remark 2. *We also note that, in order for Crank–Nicolson to achieve a provably second-order convergence rate, we require the state v and the adjoint variable ζ in* (3.3) *to have sufficient smoothness. Therefore, in the following discussion, we allow the assumptions of regularity and the results in [4, Sec. 3] to hold for the problem considered.*

Considering again (3.3), we now employ the Crank–Nicolson method for discretizing the time derivative. Denoting

$$L^+ = \frac{\tau}{2}K + M, \quad L^- = \frac{\tau}{2}K - M, \quad \bar{M} = \frac{\tau}{2}M, \quad \bar{M}_\beta = \frac{\tau}{2\beta}M,$$

we have that the numerical solution of (3.3) satisfies

$$\begin{cases} \bar{M}\left(\boldsymbol{v}_n + \boldsymbol{v}_{n+1}\right) + L^+\boldsymbol{\zeta}_n + L^-\boldsymbol{\zeta}_{n+1} &= \bar{M}\left(\boldsymbol{v}_d^n + \boldsymbol{v}_d^{n+1}\right), \\ L^-\boldsymbol{v}_n + L^+\boldsymbol{v}_{n+1} - \bar{M}_\beta\left(\boldsymbol{\zeta}_n + \boldsymbol{\zeta}_{n+1}\right) &= \frac{\tau}{2}\left(\boldsymbol{f}^n + \boldsymbol{f}^{n+1}\right), \end{cases}$$

for $n = 0, 1, ..., n_t - 1$, with $M\boldsymbol{\zeta}_{n_t} = \mathbf{0}$ and $M\boldsymbol{v}_0 = M\boldsymbol{v}^0$ an appropriate discretization of the final and initial conditions on ζ and v, and $\boldsymbol{f}^n$ defined as in (3.4). In matrix form, we write

$$\begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{v}} \\ \bar{\boldsymbol{\zeta}} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{b}}_1 \\ \bar{\mathbf{b}}_2 \end{bmatrix}, \tag{3.7}$$

where the vectors $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{\zeta}}$ are the numerical solution for the state and adjoint variables, and as before the right-hand side accounts for the initial and final-time conditions on v and ζ, as well as the desired state v_d and force function f. The matrices $\bar{A}_{il}$, $i, l = 1, 2$, are given by

$$\bar{A}_{11} = \begin{bmatrix} \bar{M} & \bar{M} & & \\ & \ddots & \ddots & \\ & & \bar{M} & \bar{M} \\ & & & 0 \end{bmatrix}, \qquad \bar{A}_{12} = \begin{bmatrix} L^+ & L^- & & \\ & \ddots & \ddots & \\ & & L^+ & L^- \\ & & & M \end{bmatrix},$$

$$\bar{A}_{21} = \begin{bmatrix} M & & & \\ L^- & L^+ & & \\ & \ddots & \ddots & \\ & & L^- & L^+ \end{bmatrix}, \qquad \bar{A}_{22} = -\begin{bmatrix} 0 & & & \\ \bar{M}_\beta & \bar{M}_\beta & & \\ & \ddots & \ddots & \\ & & \bar{M}_\beta & \bar{M}_\beta \end{bmatrix}.$$

We immediately realize that the matrices $\bar{A}_{11}$ and $\bar{A}_{22}$ are not symmetric, and therefore no iterative method for symmetric matrices, such as MINRES, may be applied to (3.7). We therefore wish to apply a transformation to (3.7) to convert it to a symmetric system. We partition the matrix in (3.7) as

$$\left[\begin{array}{ccccc|ccccc|c}
\bar{M} & \bar{M} & & & & L^+ & L^- & & & & 0 \\
0 & \bar{M} & \bar{M} & & & & L^+ & L^- & & & \\
 & & \ddots & \ddots & & & & \ddots & \ddots & & \\
 & & & \bar{M} & \bar{M} & & & & & L^+ & L^- \\
\hline
 & & & & 0 & & & & & & M \\
\hline
M & & & & & 0 & & & & & \\
\hline
L^- & L^+ & & & & -\bar{M}_\beta & -\bar{M}_\beta & & & & 0 \\
 & L^- & L^+ & & & & -\bar{M}_\beta & -\bar{M}_\beta & & & \\
 & & \ddots & \ddots & & & & \ddots & \ddots & & \\
 & & & L^- & L^+ & & & & -\bar{M}_\beta & -\bar{M}_\beta
\end{array}\right].$$

Then if we eliminate the initial and final-time conditions on v and ζ, we can rewrite

$$\underbrace{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & -A_{22} \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \tag{3.8}$$

with the vectors $\boldsymbol{v}$, $\boldsymbol{\zeta}$, $\mathbf{b}_1$, $\mathbf{b}_2$ modified accordingly, and the matrices A_{il}, $i, l = 1, 2$, given by

$$A_{11} = \begin{bmatrix} \bar{M} & & & \\ \bar{M} & \bar{M} & & \\ & \ddots & \ddots & \\ & & \bar{M} & \bar{M} \end{bmatrix}, \qquad A_{12} = \begin{bmatrix} L^+ & L^- & & \\ & \ddots & \ddots & \\ & & L^+ & L^- \\ & & & L^+ \end{bmatrix}, \tag{3.9}$$

$$A_{21} = \underbrace{\begin{bmatrix} L^+ & & & \\ L^- & L^+ & & \\ & \ddots & \ddots & \\ & & L^- & L^+ \end{bmatrix}}_{=A_{12}^\top}, \qquad A_{22} = \begin{bmatrix} \bar{M}_\beta & \bar{M}_\beta & & \\ & \ddots & \ddots & \\ & & \bar{M}_\beta & \bar{M}_\beta \\ & & & \bar{M}_\beta \end{bmatrix}. \tag{3.10}$$

In order to symmetrize the system, we now apply the following linear transformation:

$$T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix}, \tag{3.11}$$

where

$$T_1 = \begin{bmatrix} I_{n_x} & I_{n_x} & & \\ & \ddots & \ddots & \\ & & I_{n_x} & I_{n_x} \\ & & & I_{n_x} \end{bmatrix}, \qquad T_2 = \underbrace{\begin{bmatrix} I_{n_x} & & & \\ I_{n_x} & I_{n_x} & & \\ & \ddots & \ddots & \\ & & I_{n_x} & I_{n_x} \end{bmatrix}}_{=T_1^\top}, \tag{3.12}$$

with $T_1, T_2 \in \mathbb{R}^{(n_t n_x)\times(n_t n_x)}$, and $I_{n_x} \in \mathbb{R}^{n_x \times n_x}$ denoting the identity matrix. Then,

$$T \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi & \Psi^\top \\ \Psi & -\Theta \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \end{bmatrix} = T \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \tag{3.13}$$

where

$$\Phi = T_1 A_{11} = \begin{bmatrix} 2\bar{M} & \bar{M} & & \\ \bar{M} & \ddots & \ddots & \\ & \ddots & 2\bar{M} & \bar{M} \\ & & \bar{M} & \bar{M} \end{bmatrix}, \; \Theta = T_2 A_{22} = \begin{bmatrix} \bar{M}_\beta & \bar{M}_\beta & & \\ \bar{M}_\beta & 2\bar{M}_\beta & \ddots & \\ & \ddots & \ddots & \bar{M}_\beta \\ & & \bar{M}_\beta & 2\bar{M}_\beta \end{bmatrix}.$$

Note that the matrix Φ resembles a $\mathbf{P}_1$ Galerkin mass matrix; this is probably a coincidence. Furthermore, it holds that

$$\Psi = T_2 A_{21} = \begin{bmatrix} L^+ & & & & \\ \tau K & L^+ & & & \\ L^- & \tau K & L^+ & & \\ & \ddots & \ddots & \ddots & \\ & & L^- & \tau K & L^+ \end{bmatrix} = (T_1 A_{12})^\top.$$

We have thus transformed (3.7) to a symmetric system using matrices T_1, T_2 which are easy and cheap to apply. Further, we may easily apply their inverses to a vector using only a sequence of block updates. We also highlight the further properties that

$$\Phi = T_1 \Phi_D T_1^\top, \qquad \Theta = T_2 \Theta_D T_2^\top, \tag{3.14}$$

where

$$\Phi_D = \begin{bmatrix} \bar{M} & & \\ & \ddots & \\ & & \bar{M} \end{bmatrix}, \qquad \Theta_D = \begin{bmatrix} \bar{M}_\beta & & \\ & \ddots & \\ & & \bar{M}_\beta \end{bmatrix}, \tag{3.15}$$

so we may work with the matrices Φ and Θ cheaply, using T_1, T_2, Φ_D, Θ_D. Further, since both Φ_D and Θ_D are symmetric positive definite, the same holds for Φ and Θ.

Therefore, in order to find an approximate solution to (3.3), we may now consider the saddle-point system (3.13), to which we can apply a preconditioned Krylov subspace method for symmetric indefinite matrices, such as MINRES.

Remark 3. *It is worth noting that, for a fixed number of intervals n_t in time, the cost of a matrix–vector multiplication with the Crank–Nicolson system is slightly more costly than that of backward Euler. Note first that the system* (3.7) *(without applying the linear transformation T) is more dense than* (3.5)*, as the* $(1,1)$*- and the* $(2,2)$*-blocks are block bidiagonal. This will imply a marginally higher cost per iteration of the preconditioned Krylov subspace method for Crank–Nicolson, which increases if we apply also the linear transformation T. However, as we will see in the numerical results, the overall cost per iteration remains linear in the number of unknowns, as the* $(1,1)$*- and* $(2,2)$*-blocks of the Crank–Nicolson system are sparse and the matrices T_1, T_2, T_1^{-1}, T_2^{-1} are cheap to apply. Clearly, in our numerical results we do not consider the same number of time-steps for both backward Euler and Crank–Nicolson methods, as the latter should require far fewer steps for the same accuracy.*

The work outlined here is based on an optimize-then-discretize approach, with the Crank–Nicolson scheme applied in time for the discretization. This led to a non-symmetric system to be solved. We would like to mention that the symmetrization of such a system outlined above may be avoided if one employs a discretize-then-optimize approach coupled with a Crank–Nicolson discretization in time. In fact, in this case the linear system to be solved would be symmetric from the start, as we show here. Discretizing the cost functional (3.1) with the trapezoidal rule and the heat equation (3.2) with a Crank–Nicolson method in time, one has to solve the following QP problem:

$$\min_{\boldsymbol{v},\boldsymbol{u}} \quad J_h(\boldsymbol{v},\boldsymbol{u}) = \frac{1}{2}(\boldsymbol{v}-\boldsymbol{v}_d)^\top \breve{\Phi}_D(\boldsymbol{v}-\boldsymbol{v}_d) + \frac{\beta}{2}\boldsymbol{u}^\top \breve{\Phi}_D \boldsymbol{u} \tag{3.16}$$

subject to

$$A_{21}\boldsymbol{v} = \boldsymbol{f} + A_{11}\boldsymbol{u}, \tag{3.17}$$

with $\boldsymbol{v}_0 = \boldsymbol{v}^0$ an appropriate discretization of the initial condition on v. Here, $\boldsymbol{v}_d$ and $\boldsymbol{f}$ are an appropriate discretization of the desired state v_d and the force function f, respectively, A_{11} and A_{21} are defined as in (3.9)–(3.10), and

$$\breve{\Phi}_D = \begin{bmatrix} \bar{M} & & & & \\ & 2\bar{M} & & & \\ & & \ddots & & \\ & & & 2\bar{M} & \\ & & & & \bar{M} \end{bmatrix}.$$

Then, the first-order optimality conditions of (3.16)–(3.17) read as follows:

$$\begin{cases} \breve{\Phi}_D \boldsymbol{v} + A_{21}^\top \boldsymbol{\zeta} = \breve{\mathbf{b}}_1, \\ \beta\, \breve{\Phi}_D \boldsymbol{u} - A_{11}^\top \boldsymbol{\zeta} = \mathbf{0}, \\ A_{21}\boldsymbol{v} - A_{11}\boldsymbol{u} = \breve{\mathbf{b}}_2, \end{cases} \tag{3.18}$$

where the vectors $\breve{\mathbf{b}}_1$ and $\breve{\mathbf{b}}_2$ account for the boundary and initial conditions on v, as well as information from the force function $\boldsymbol{f}$ and the desired state $\boldsymbol{v}_d$. Clearly,

the system above is symmetric. In addition, it is worth noting that some of the blocks of the system (3.18) share a similar structure to the ones in (3.8). With this in mind, one may adapt the preconditioning strategy described below for deriving a preconditioner for the system (3.18). We will come back to this in the following.

Before deriving the proposed preconditioner for the system (3.13), we would like to spend some words on the transformations T_1 and T_2 defined in (3.12). These transformations are (up to a factor of 2) averaging values at different time levels, and arise from the time-stepping scheme that we are employing. Specifically, we have that

$$A_{11} = \breve{T}_2 \otimes M, \qquad A_{22} = \frac{1}{\beta}\breve{T}_1 \otimes M,$$

where

$$\begin{aligned}
\breve{T}_1 &= \begin{bmatrix} \frac{\tau}{2} & \frac{\tau}{2} & & \\ & \ddots & \ddots & \\ & & \frac{\tau}{2} & \frac{\tau}{2} \\ & & & \frac{\tau}{2} \end{bmatrix} = \underbrace{\begin{bmatrix} \tau & & & \\ & \ddots & & \\ & & \tau & \\ & & & \tau \end{bmatrix}}_{=:\Phi_\tau} \underbrace{\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & & \\ & \ddots & \ddots & \\ & & \frac{1}{2} & \frac{1}{2} \\ & & & \frac{1}{2} \end{bmatrix}}_{=:T_{\text{step}}}, \\
\breve{T}_2 = \breve{T}_1^\top &= \begin{bmatrix} \frac{\tau}{2} & & & \\ \frac{\tau}{2} & \frac{\tau}{2} & & \\ & \ddots & \ddots & \\ & & \frac{\tau}{2} & \frac{\tau}{2} \end{bmatrix} = \begin{bmatrix} \tau & & & \\ & \ddots & & \\ & & \tau & \\ & & & \tau \end{bmatrix} \begin{bmatrix} \frac{1}{2} & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ & \ddots & \ddots & \\ & & \frac{1}{2} & \frac{1}{2} \end{bmatrix}.
\end{aligned} \tag{3.19}$$

Note that the matrix Φ_τ contains the time-steps adopted, while the matrix T_{step} contains the weights of the time-stepping scheme employed. In order to symmetrize the blocks A_{11} and A_{22} we first find transformations that symmetrize the matrices $\breve{T}_2$ and $\breve{T}_1$ respectively. Such transformations are clearly given by T_{step} and $T_{\text{step}}^\top$, respectively. Then, since the matrix M is symmetric and due to the properties of the Kronecker product, it is easy to see that the transformations

$$T_{\text{step}} \otimes I_{n_x} = \frac{1}{2}T_1, \qquad T_{\text{step}}^\top \otimes I_{n_x} = \frac{1}{2}T_2$$

symmetrize the blocks A_{11} and A_{22}, respectively. It is worth noting that due to the time-symmetry in the Crank–Nicolson scheme we can simplify the sought linear transformations by multiplying the above expressions by 2. In addition, from the expressions of $\breve{T}_1$ and $\breve{T}_2$ in (3.19), we can clearly see that the strategy described above can be applied also if we employ a non-constant time-steps in the discretization, as the matrices T_1 and T_2 defined in (3.12) will still symmetrize the system (3.8). Finally, since the matrix T_{step} contains the weights of the time-stepping scheme employed, we can see how the strategy described above may be extended to linear multistep methods other than Crank–Nicolson.

3.3 Preconditioning Approach

In this section we describe an optimal preconditioner for the system (3.13), by making use of saddle-point theory as well as suitable approximations for the blocks of the matrix $\mathcal{A}$.

As we discussed in Section 2.10, a good candidate for a preconditioner for an invertible system of the form (3.13), with invertible Φ, is the block diagonal matrix $\mathcal{P}_3$ defined in (2.28). However, the preconditioner $\mathcal{P}_3$ defined in (2.28) is not practical, as we noted in Section 2.10. For this reason, we wish to find a suitable approximation $\widehat{\mathcal{P}}_3$ of $\mathcal{P}_3$, with

$$\widehat{\mathcal{P}}_3 = \begin{bmatrix} \widehat{\Phi} & 0 \\ 0 & \widehat{S} \end{bmatrix},$$

or, more precisely, a cheap application of the effect of $\widehat{\mathcal{P}}_3^{-1}$ on a generic vector. In the following section we commence by finding a good approximation of the inverse of the matrix Φ, then in Section 3.3.2 we describe the approach used for approximating the Schur complement S.

Since we will benchmark our new preconditioning strategy against the preconditioner derived in [139] for heat control after applying the optimize-then-discretize approach with backward Euler time-stepping, we briefly describe the preconditioner for the system (3.5), itself also a saddle-point system. In this case the $(1,1)$-block Φ^{BE}, defined in (3.6), is not invertible; a good preconditioner is found to be

$$\widehat{\mathcal{P}}_3^{\mathrm{BE}} = \begin{bmatrix} \widehat{\Phi}^{\mathrm{BE}} & 0 \\ 0 & \widehat{S}^{\mathrm{BE}} \end{bmatrix},$$

where

$$\widehat{\Phi}^{\mathrm{BE}} = \begin{bmatrix} \tau M & & & \\ & \ddots & & \\ & & \tau M & \\ & & & \xi\tau M \end{bmatrix}, \quad \widehat{S}^{\mathrm{BE}} = (\Psi^{\mathrm{BE}} + \widehat{M}^{\mathrm{BE}})(\widehat{\Phi}^{\mathrm{BE}})^{-1}(\Psi^{\mathrm{BE}} + \widehat{M}^{\mathrm{BE}})^{\top}, \tag{3.20}$$

with

$$\widehat{M}^{\mathrm{BE}} = \frac{\tau}{\sqrt{\beta}} \begin{bmatrix} 0 & & & \\ & M & & \\ & & \ddots & \\ & & & M \\ & & & & \sqrt{\xi}M \end{bmatrix}. \tag{3.21}$$

Here, $0 < \xi \ll 1$ is chosen such that the (invertible) matrix $\widehat{\Phi}^{\mathrm{BE}}$ is 'close enough' to the matrix Φ^{BE} in some sense. The $(1,1)$-block is approximated by a 'perturbed' matrix $\widehat{\Phi}^{\mathrm{BE}}$, whose inverse is then applied within the matrix $\widehat{S}^{\mathrm{BE}}$. A term of the form (3.21) is justified in Section 3.3.2.

Before devising a preconditioner for the linear system (3.13), we mention it is possible to precondition the non-symmetric system (3.8). Naturally, we

would need to apply a non-symmetric Krylov solver, such as GMRES [157], in which case we could employ as an 'ideal' preconditioner the matrix $\mathcal{P}_1$ defined in (2.26). Although the forthcoming strategies to approximate the relevant blocks of the preconditioner may be adapted to this case, it is not possible to prove similar bounds on the eigenvalues as the one described in Theorem 5, nor could convergence of the iterative method be described solely using the distribution of the eigenvalues. For this reason, we focus our analysis on the symmetric preconditioner $\mathcal{P}_3$ defined in (2.28).

3.3.1 Approximation of the $(1,1)$-Block

We now focus on devising a preconditioner for the linear system (3.13), arising from a Crank–Nicolson discretization, starting with a cheap and effective approximation of the matrix Φ. We recall from (3.14) that Φ can be written as $\Phi = T_1 \Phi_D T_1^\top$, with Φ_D defined as in (3.15). We observe that Φ_D is a block diagonal matrix with each diagonal block a multiple of M. Therefore, in order to obtain a cheap approximation for Φ, we require a suitable way to approximate the inverse of a mass matrix. As discussed in [63, 64, 181] and in Section 2.4, the Chebyshev semi-iterative method represents a cheap and effective way to do this. From the discussion above, a good approximation of Φ is therefore given by $\widehat{\Phi} = T_1 \widehat{\Phi}_D T_1^\top$, with

$$\widehat{\Phi}_D = \frac{\tau}{2} \begin{bmatrix} M_c & & \\ & \ddots & \\ & & M_c \end{bmatrix},$$

where M_c denotes a fixed number of steps (20, in our tests) of Chebyshev semi-iteration applied to M. Further, since T_1 and $T_2 = T_1^\top$ are (block) upper- and lower-triangular matrices respectively, we use simple (backward/forward) block updates in order to apply their inverse operations.

It is trivial to prove the following, on the effectiveness of our approximation of Φ:

Lemma 1. *The minimum (maximum) eigenvalue of $\widehat{\Phi}^{-1}\Phi$ is bounded below (above) by the minimum (maximum) eigenvalue of $M_c^{-1}M$.*

3.3.2 Approximation of Schur Complement

We now find a suitable approximation for the Schur complement S of (3.13). Recalling how the matrices Φ, Ψ, and Θ can be written as linear transformations involving the matrices T_1 and T_2, we first rewrite S in the following way:

$$S = T_2 \Theta_D T_2^\top + T_2 A_{21} (T_1 \Phi_D T_1^\top)^{-1} A_{21}^\top T_2^\top = T_2 \left[\Theta_D + \Psi_D \Phi_D^{-1} \Psi_D^\top \right] T_2^\top, \quad (3.22)$$

where we set

$$\Psi_D = A_{21} T_2^{-1}. \quad (3.23)$$

It is hence clear that if we find a symmetric positive definite approximation $\widetilde{S}_{\text{int}}$ of

$$S_{\text{int}} = \Theta_D + \Psi_D \Phi_D^{-1} \Psi_D^\top, \tag{3.24}$$

then $\widehat{S} := T_2 \widetilde{S}_{\text{int}} T_2^\top$ is a symmetric positive definite approximation of S. We may therefore use the (generalized) Rayleigh quotient in order to find upper and lower bounds for the eigenvalues of the matrix $\widehat{S}^{-1}S$ as follows. Let λ be an eigenvalue of $\widehat{S}^{-1}S$ with $\mathbf{x}$ the corresponding eigenvector; then

$$\widehat{S}^{-1}S\mathbf{x} = \lambda \mathbf{x} \;\Rightarrow\; S\mathbf{x} = \lambda \widehat{S}\mathbf{x} \;\Rightarrow\; \lambda = \frac{\mathbf{x}^\top S \mathbf{x}}{\mathbf{x}^\top \widehat{S}\mathbf{x}} = \frac{\widetilde{\mathbf{x}}^\top S_{\text{int}} \widetilde{\mathbf{x}}}{\widetilde{\mathbf{x}}^\top \widetilde{S}_{\text{int}} \widetilde{\mathbf{x}}}, \tag{3.25}$$

where we set $\widetilde{\mathbf{x}} = T_2^\top \mathbf{x} \neq \mathbf{0}$ and use (3.22) together with the definition of $\widehat{S}$. Thus upper and lower bounds for the eigenvalues of $\widehat{S}^{-1}S$ are given by the maximum and the minimum eigenvalues of $\widetilde{S}_{\text{int}}^{-1} S_{\text{int}}$, respectively.

From (3.22), and due to the convenient structure of the matrices Φ_D and Θ_D in (3.15), we can devise an approximation $\widetilde{S}_{\text{int}}$ of S_{int} using the matching strategy discussed in [139, 142] and in Section 2.11 as follows. We seek an approximation:

$$\widetilde{S}_{\text{int}} = (A_{21} + \widehat{M})\Phi^{-1}(A_{21} + \widehat{M})^\top \approx S_{\text{int}} \tag{3.26}$$

such that $\widetilde{S}_{\text{int}}$ 'captures' both terms of S_{int}, namely Θ_D and $\Psi_D \Phi_D^{-1} \Psi_D^\top$ (noting that $A_{21}\Phi^{-1}A_{21}^\top = \Psi_D \Phi_D^{-1} \Psi_D^\top$). We therefore wish that

$$\widehat{M}\Phi^{-1}\widehat{M}^\top = \left[\widehat{M}(T_1^\top)^{-1}\right] \Phi_D^{-1} \left[T_1^{-1}\widehat{M}^\top\right] = \Theta_D. \tag{3.27}$$

Using the definitions of Θ_D and Φ_D from (3.15), we obtain that for this to hold the matrix $\widehat{M}(T_1^\top)^{-1}$ is given by

$$M_D := \widehat{M}(T_1^\top)^{-1} = \frac{\tau}{2\sqrt{\beta}} \begin{bmatrix} M & & \\ & \ddots & \\ & & M \end{bmatrix}, \tag{3.28}$$

and therefore

$$\widehat{M} = \frac{\tau}{2\sqrt{\beta}} \begin{bmatrix} M & & & \\ M & M & & \\ & \ddots & \ddots & \\ & & M & M \end{bmatrix}. \tag{3.29}$$

Finally, our approximation of S is given by

$$\widehat{S} = T_2(A_{21} + \widehat{M})\Phi^{-1}(A_{21} + \widehat{M})^\top T_2^\top = (A_{21} + \widehat{M})\Phi_D^{-1}(A_{21} + \widehat{M})^\top, \tag{3.30}$$

with $\widehat{M}$ as defined in (3.29), and the two expressions are equivalent since T_2 commutes with both A_{21} and $\widehat{M}$. To understand the effectiveness of this approximation, we recall (3.25), telling us that we only need to study the spectrum of

the matrix $\widetilde{S}_{\text{int}}^{-1} S_{\text{int}}$. We next rewrite $\widetilde{S}_{\text{int}}$ as follows:

$$\begin{aligned}\widetilde{S}_{\text{int}} &= (A_{21} + \widehat{M})\Phi^{-1}(A_{21} + \widehat{M})^\top \\ &= \widehat{M}\Phi^{-1}\widehat{M}^\top + A_{21}\Phi^{-1}A_{21}^\top + \widehat{M}\Phi^{-1}A_{21}^\top + A_{21}\Phi^{-1}\widehat{M}^\top \\ &= \Theta_D + \Psi_D\Phi_D^{-1}\Psi_D^\top + M_D\Phi_D^{-1}\Psi_D^\top + \Psi_D\Phi_D^{-1}M_D^\top \\ &= S_{\text{int}} + M_D\Phi_D^{-1}\Psi_D^\top + \Psi_D\Phi_D^{-1}M_D^\top, \end{aligned} \tag{3.31}$$

where we have used (3.27), (3.14), (3.23), and (3.28) in turn.

Since S_{int} and $\widetilde{S}_{\text{int}}$ are symmetric positive definite, we again consider the generalized Rayleigh quotient:

$$R := \frac{\mathbf{x}^\top S_{\text{int}}\mathbf{x}}{\mathbf{x}^\top \widetilde{S}_{\text{int}}\mathbf{x}} = \frac{\mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b}}{\mathbf{a}^\top\mathbf{a} + \mathbf{b}^\top\mathbf{b} + \mathbf{a}^\top\mathbf{b} + \mathbf{b}^\top\mathbf{a}}, \tag{3.32}$$

where $\mathbf{a} = (\Psi_D\Phi_D^{-1/2})^\top\mathbf{x}$ and $\mathbf{b} = (\Theta_D^{1/2})^\top\mathbf{x}$, noting from (3.15) and (3.28) that $\Theta_D^{1/2} = M_D\Phi_D^{-1/2}$. Working as in Section 2.11, since $\Phi \succ 0$ and $\Theta \succ 0$, for Theorem 1 we have $R \geqslant \dfrac{1}{2}$.

In order to find an upper bound for the Rayleigh quotient (3.32), we return to (3.31). Noting that

$$M_D\Phi_D^{-1} = \frac{1}{\sqrt{\beta}}\begin{bmatrix} I & & \\ & \ddots & \\ & & I \end{bmatrix},$$

we can rewrite

$$\widetilde{S}_{\text{int}} = S_{\text{int}} + \frac{1}{\sqrt{\beta}}\left(\Psi_D^\top + \Psi_D\right).$$

Following the reasoning in [139] and the reasoning in Theorem 2, we can prove that $R \leqslant 1$; from (3.32), this holds if

$$\begin{aligned}\mathbf{a}^\top\mathbf{b} + \mathbf{b}^\top\mathbf{a} &= \mathbf{x}^\top\left(M_D\Phi_D^{-1}\Psi_D^\top + \Psi_D\Phi_D^{-1}M_D^\top\right)\mathbf{x} = \frac{1}{\sqrt{\beta}}\,\mathbf{x}^\top\left(\Psi_D^\top + \Psi_D\right)\mathbf{x} \\ &= \frac{1}{\sqrt{\beta}}\,\mathbf{z}^\top\left(A_{21}^\top T_2 + T_2^\top A_{21}\right)\mathbf{z} \geqslant 0\,,\end{aligned}$$

where the last line uses (3.23) and sets $\mathbf{z} = T_2^{-1}\mathbf{x}$. Therefore, we wish to show that the matrix $\mathcal{X} = A_{21}^\top T_2 + T_2^\top A_{21}$ is positive semi-definite. We easily obtain that

$$\mathcal{X} = \frac{\tau}{2}\underbrace{\begin{bmatrix} 2\widetilde{L} & \widetilde{L} & & \\ \widetilde{L} & \ddots & \ddots & \\ & \ddots & 2\widetilde{L} & \widetilde{L} \\ & & \widetilde{L} & \widetilde{L} \end{bmatrix}}_{=:\widetilde{\mathcal{L}}} + \underbrace{\begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & 2M \end{bmatrix}}_{=:\widetilde{\mathcal{M}}},$$

with $\widetilde{L} = K + K^\top = 2K$ since K is symmetric. Furthermore, since K is positive definite in this case, $\widetilde{L}$ is also positive definite. Moreover, it is clear that

$$\mathbf{z}^\top \widetilde{\mathcal{L}}\, \mathbf{z} \geqslant 0 \ \wedge \ \mathbf{z}^\top \widetilde{\mathcal{M}}\, \mathbf{z} \geqslant 0 \quad \Rightarrow \quad \mathbf{z}^\top \mathcal{X}\, \mathbf{z} \geqslant 0. \tag{3.33}$$

We now use the following classical result on Kronecker products (see [98, Theorem 13.12], for instance) to show that $\widetilde{\mathcal{L}}$ is positive definite:

Theorem 3. *Let $X_1 \in \mathbb{R}^{n_1 \times n_1}$ have eigenvalues λ_i, $i = 1, 2, \ldots, n_1$, and let $X_2 \in \mathbb{R}^{n_2 \times n_2}$ have eigenvalues μ_l, $l = 1, 2, \ldots, n_2$. Then, the $n_1 n_2$ eigenvalues of $X_1 \otimes X_2$ are*

$$\lambda_1\mu_1, \ldots, \lambda_1\mu_{n_2}, \lambda_2\mu_1, \ldots, \lambda_2\mu_{n_2}, \ldots, \lambda_{n_1}\mu_{n_2}.$$

Moreover, if $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m_1}$ are linearly independent right eigenvectors of X_1 corresponding to $\lambda_1, \lambda_2, \ldots, \lambda_{m_1}$, $m_1 \leqslant n_1$, and $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{m_2}$ are linearly independent right eigenvectors of X_2 corresponding to $\mu_1, \mu_2, \ldots, \mu_{m_2}$, $m_2 \leqslant n_2$, then $\mathbf{x}_i \otimes \mathbf{z}_l \in \mathbb{R}^{n_1 n_2}$ are linearly independent right eigenvectors of $X_1 \otimes X_2$ corresponding to $\lambda_i\mu_l$, $i = 1, 2, \ldots, m_1$, $l = 1, 2, \ldots, m_2$.

Since both $\widetilde{L}$ and

$$\mathcal{T} = \begin{bmatrix} 2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & 2 & 1 \\ & & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix} = \mathcal{T}_1^\top \mathcal{T}_1$$

are symmetric positive definite (since $\mathcal{T}_1$ has full rank), Theorem 3 gives that $\widetilde{\mathcal{L}} = \mathcal{T} \otimes \widetilde{L}$ is symmetric positive definite. Since $\widetilde{\mathcal{M}}$ is clearly symmetric positive semi-definite, we infer from (3.33) that $\mathcal{X}$ is symmetric positive definite, and hence that

$$\mathbf{a}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{a} \geqslant 0,$$

with $\mathbf{a}$ and $\mathbf{b}$ as defined above. Finally, for Theorem 2 the last inequality guarantees that the Rayleigh quotient R in (3.32) satisfies $R \leqslant 1$.

We have hence proved the following result:

Theorem 4. *Let S_{int} and $\widetilde{S}_{\text{int}}$ be defined as in* (3.24) *and* (3.26) *respectively, with the matrices Φ_D, Ψ_D, Θ_D, A_{21}, Φ, $\widehat{M}$ defined as in* (3.15), (3.23), (3.10), (3.14), *and* (3.29). *Then:*

$$\lambda(\widetilde{S}_{\text{int}}^{-1} S_{\text{int}}) \in \left[\frac{1}{2}, 1\right].$$

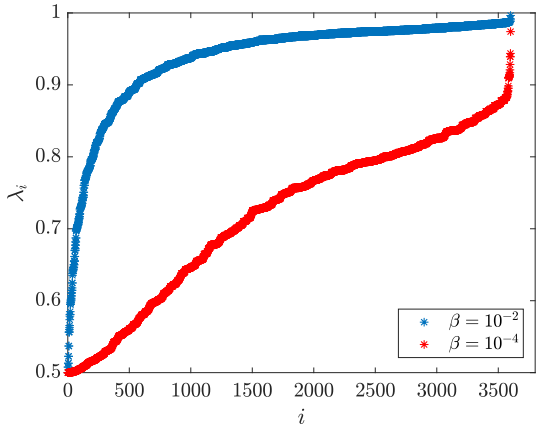
In Figure 3.1 we report the eigenvalue distribution of $\widetilde{S}_{\text{int}}^{-1} S_{\text{int}}$ for a range of values of β, for a particular Dirichlet test problem.

Further, using Theorem 4 and (3.25), we can prove the following:

Theorem 5. *Let S and $\widehat{S}$ be defined as in* (3.22) *and* (3.30), *with the matrices defined as in Theorem 4, and T_1, T_2 as in* (3.12). *Then:*

$$\lambda(\widehat{S}^{-1} S) \in \left[\frac{1}{2}, 1\right].$$

Figure 3.1: Eigenvalues of $\widetilde{S}_{\text{int}}^{-1} S_{\text{int}}$ for $\beta = 10^{-j}$, $j = 2, 4$, with $d = 2$ ($x = [x_1,\ x_2]^\top$) and $t_f = 2$, employing $\mathbf{Q}_1$ finite elements on an evenly spaced space–time grid $(-1, 1)^2 \times (0, 2)$ with $\tau = h = \frac{1}{8}$.



Remark 4. *We note that no assumption has been made on the grid structure, meaning that the bounds in Theorem 4 and in Theorem 5 still hold in case of non-uniform meshes. In addition, the preconditioner above can be easily generalized when Neumann or mixed boundary conditions are imposed.*

By Theorem 5, the matrix $\widehat{S}$ in (3.30) is an effective approximation of the Schur complement S defined in (3.22). We highlight that solving (exactly) a system involving the matrix $\widehat{S}$ is costly, so we look for a cheap approximation of the effect of $\widehat{S}^{-1}$ on a generic vector. From (3.30), it is clear that the bulk of the work involves approximately applying the inverse of $A_{21} + \widehat{M}$ and its transpose. From (3.10) and (3.29), we have that $A_{21} + \widehat{M}$ is block-lower triangular, and a cheap application of its inverse on a vector is given by block-forward substitution, with each block diagonal approximated using a fixed number of V-cycles of a multigrid routine, for example. The matrix $(A_{21} + \widehat{M})^\top$ can be handled analogously using block-backward substitution.

Before showing the robustness of the proposed preconditioner, we would like to spend some words on the linear system (3.18) arising from a discretize-then-optimize approach. In matrix form, we rewrite system (3.18) as

$$\begin{bmatrix} \breve{\Phi}_D & 0 & A_{21}^\top \\ 0 & \beta\breve{\Phi}_D & -A_{11}^\top \\ A_{21} & -A_{11} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{u} \\ \boldsymbol{\zeta} \end{bmatrix} = \begin{bmatrix} \breve{\mathbf{b}}_1 \\ \mathbf{0} \\ \breve{\mathbf{b}}_2 \end{bmatrix},$$

which presents a saddle-point structure. In this case, an optimal preconditioner is given by

$$\breve{\mathcal{P}}_3 = \begin{bmatrix} \breve{\Phi}_D & 0 & 0 \\ 0 & \beta\breve{\Phi}_D & 0 \\ 0 & 0 & \breve{S} \end{bmatrix},$$

where

$$\begin{aligned}
\breve{S} &= A_{21}\breve{\Phi}_D^{-1}A_{21}^\top + A_{11}(\beta\breve{\Phi}_D)^{-1}A_{11}^\top \\
&= A_{21}\breve{\Phi}_D^{-1}A_{21}^\top + \frac{1}{2\beta}T_2\begin{bmatrix} 2\bar{M} & & & & \\ & \bar{M} & & & \\ & & \ddots & & \\ & & & \bar{M} & \\ & & & & 2\bar{M} \end{bmatrix}T_2^\top \\
&= A_{21}\breve{\Phi}_D^{-1}A_{21}^\top + T_2\underbrace{\begin{bmatrix} 2\bar{M}_\beta & & & & \\ & \bar{M}_\beta & & & \\ & & \ddots & & \\ & & & \bar{M}_\beta & \\ & & & & 2\bar{M}_\beta \end{bmatrix}}_{\breve{\Theta}_D}T_2^\top.
\end{aligned}$$

Then, we can rewrite

$$\breve{S} = T_2(\breve{\Theta}_D + \breve{\Psi}_D\breve{\Phi}_D^{-1}\breve{\Psi}_D^\top)T_2^\top,$$

with $\breve{\Psi}_D = T_2^{-1}A_{21}$. From here, we can see that the strategy described above may be extended also to the case of the discretize-then-optimize approach. Of course, one should prove the optimality of such a preconditioner, which we do not address here. However, supposing that the derived preconditioner is optimal, one could also apply the discretize-then-optimize strategy to the problems we will consider in the following chapters.

3.4 Numerical Results

We now provide numerical evidence of the effectiveness of our preconditioning strategy. Below, we show how this preconditioner results in more rapid convergence than the state-of-the-art backward Euler solver with the existing preconditioner of [139], for the heat control problem.

In all our tests we consider only Dirichlet boundary conditions (i.e., $\partial\Omega_N = \varnothing$), but we emphasize again that the method is easily generalized to Neumann and mixed boundary conditions. We implement a finite element method, using $\mathbf{Q}_1$ basis functions for state, control, and adjoint variables. As discussed in Section 3.3.1, when approximating the $(1,1)$-block it is trivial to invert both matrices T_1 and $T_2 = T_1^\top$, and we apply 20 steps of Chebyshev semi-iteration to each mass matrix on the diagonal of Φ_D. For the approximation of the Schur complement, we employ block-forward and block-backward substitution to solve for the matrix $A_{21} + \widehat{M}$ and its transpose, and approximate each block on the diagonal with 3 V-cycles of an appropriate multigrid routine, unless otherwise stated. Specifically, we employ the `HSL_MI20` solver [22] for approximating each block diagonal. The iteration count for all tests presented starts from 0. All tests are run on MATLAB R2018b, using a 1.70GHz Intel quad-core i5 processor and 8 GB RAM on an Ubuntu 18.04.1 LTS operating system.

We benchmark our method against the backward Euler method coupled with the bespoke, mesh- and β-independent preconditioner derived in [139], for the heat control problem (3.1)–(3.2). Here, $d = 2$ (so $x = [x_1,\ x_2]^\top$), $\Omega = (-1, 1)^2$, $t_f = 2$, $f = 0$, and

$$v_d(x_1, x_2, t) = 1 + \left[\left(\frac{2}{\pi^2 \beta} + \frac{\pi^2}{2} \right) e^{t_f} + \left(1 - \frac{2}{(2 + \pi^2)\beta} - \frac{\pi^2}{2} \right) e^t \right] s(x_1, x_2),$$

where $s(x_1, x_2) = \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right)$. The analytic solutions to this problem are:

$$v(x_1, x_2, t) = 1 + \left(\frac{2}{\pi^2 \beta} e^{t_f} - \frac{2}{(2 + \pi^2)\beta} e^t \right) s(x_1, x_2),$$

$$\zeta(x_1, x_2, t) = (e^{t_f} - e^t)\, s(x_1, x_2),$$

with initial condition v_0 and Dirichlet boundary condition g obtained from this v.

In our first tests, we consider a (spatial) uniform grid of mesh-size $h = 2^{1-\mathtt{l}}$, with $\mathtt{l}$ the level of grid refinement, and set $\tau = h^2$ for the backward Euler scheme and $\tau = h$ for Crank–Nicolson, motivated by the predicted convergence rates of these methods. We test both schemes with a range of values of h and β, running preconditioned MINRES to a tolerance of 10^{-6} on the relative residual norm. We take $\xi = 10^{-3}$ in (3.20)–(3.21) for the backward Euler implementation. Tables 3.1–3.3 present the number of MINRES iterations `it` required by each method for a range of β, the CPU time taken in seconds, and the relative errors v_{error} and ζ_{error} (in the scaled vector ℓ^∞-norm) obtained for the state and adjoint variables. Specifically, for the state variable we define

$$v_{\text{error}} = \frac{|v_{\mathtt{i}} - v_{\mathtt{i}}^{\text{sol}}|}{|v_{\mathtt{i}}^{\text{sol}}|}, \quad \text{with} \quad \mathtt{i} = \arg\max_i |v_i - v_i^{\text{sol}}|,$$

with $v_{\mathtt{i}}$ and $v_{\mathtt{i}}^{\text{sol}}$ the entries of the computed solution $\boldsymbol{v}$ and the (discretized) exact solution for v; in the same way we define the error for the adjoint variable. In Table 3.4 we report the degrees of freedom for each problem solved, together with the average time per iteration required to apply the inverses of the $(1, 1)$-block and the Schur complement within the preconditioners employed for the two methods, for $\beta = 10^{-3}$. We emphasize that we obtain similar results for the other two values of β, so for the sake of brevity we do not report these. Further, in order to show that our new preconditioner does not require a constant mesh-size, we also test our solver on a piecewise-uniform *Shishkin mesh* [95] with transition points 0.5 and -0.6 on the two spatial axes. We report in Table 3.5 the resulting MINRES iterations and CPU times; here, for level of grid refinement `l`, $2^{\mathtt{l}-1}$ mesh points on each axis are either side of the transition point. We note that for this particular example we apply 5 V-cycles whenever a multigrid routine is required, in order to allow for the additional difficulty for the multigrid solver with a non-uniform grid.

[4] † means that the solver ran out of memory.

Table 3.1: Heat control problem: MINRES iterations, CPU times, and resulting relative errors in v and ζ, for $\beta = 10^{-2}$.

	Backward Euler				**Crank–Nicolson**			
`l`	`it`	CPU	v_{error}	ζ_{error}	`it`	CPU	v_{error}	ζ_{error}
3	22	0.71	2.8907e-2	9.3751e-3	17	0.16	6.1670e-3	9.8782e-3
4	23	4.57	1.4472e-3	2.3445e-3	20	0.47	1.3137e-3	2.2102e-3
5	23	39.2	3.6034e-4	5.8555e-4	20	2.18	3.2492e-4	5.4963e-4
6	24	536	9.0050e-5	1.4634e-4	20	14.2	8.1064e-5	1.3721e-4
7	†[4]	–	–	–	20	116	2.0278e-5	3.4425e-5
8	†	–	–	–	20	1038	5.0895e-6	8.5887e-6

Table 3.2: Heat control problem: MINRES iterations, CPU times, and resulting relative errors in v and ζ, for $\beta = 10^{-3}$.

	Backward Euler				**Crank–Nicolson**			
`l`	`it`	CPU	v_{error}	ζ_{error}	`it`	CPU	v_{error}	ζ_{error}
3	25	0.79	1.3769e-2	1.3732e-2	18	0.16	9.6046e-4	1.5371e-2
4	24	4.57	2.3059e-3	3.4227e-3	20	0.53	1.9959e-4	3.5192e-3
5	25	42.5	5.0815e-4	8.5306e-4	21	2.29	3.7307e-5	8.0235e-4
6	27	597	1.2430e-4	2.1314e-4	21	14.8	9.2697e-6	2.0067e-4
7	†	–	–	–	21	121	2.4974e-6	5.0395e-5
8	†	–	–	–	21	1089	8.3046e-7	1.2632e-5

Table 3.3: Heat control problem: MINRES iterations, CPU times, and resulting relative errors in v and ζ, for $\beta = 10^{-4}$.

	Backward Euler				**Crank–Nicolson**			
`l`	`it`	CPU	v_{error}	ζ_{error}	`it`	CPU	v_{error}	ζ_{error}
3	18	0.67	1.0649e-2	1.4138e-2	16	0.16	6.2442e-4	1.6239e-2
4	24	6.12	1.0727e-3	3.5073e-3	19	0.55	2.2555e-4	3.5765e-3
5	26	47.0	1.7096e-4	8.7596e-4	22	2.44	5.6915e-5	8.7154e-4
6	23	513	4.0696e-5	2.1973e-4	22	15.6	1.4861e-5	2.1171e-4
7	†	–	–	–	22	127	4.0217e-6	5.2142e-5
8	†	–	–	–	22	1139	2.6336e-7	1.3867e-5

We see from Tables 3.1–3.3 that the Crank–Nicolson approach with our new preconditioner achieves more accurate solutions than the backward Euler method, in lower CPU time. Comparing the results for grid refinement $\texttt{l} = 4, 5, 6$, that is, for $h = 2^{-3}, 2^{-4}, 2^{-5}$, this can occur in orders of magnitude lower CPU time for the tests presented here. This is clear because the size of the system required to

Table 3.4: Heat control problem: degrees of freedom (DoF) and (average) CPU times required for inverting the two blocks of the preconditioners, for $\beta = 10^{-3}$.

	Backward Euler			**Crank–Nicolson**		
`l`	DoF	CPU $\widehat{\Phi}^{\mathrm{BE}}$	CPU $\widehat{S}^{\mathrm{BE}}$	DoF	CPU $\widehat{\Phi}$	CPU $\widehat{S}$
3	3234	0.0011	0.028	784	0.00038	0.0075
4	58,050	0.0091	0.16	7200	0.0013	0.023
5	985,986	0.12	1.40	61,504	0.0078	0.088
6	16,264,962	1.78	17.8	508,032	0.059	0.56
7	264,289,794	–	–	4,129,024	0.49	4.31
8	4,261,608,450	–	–	33,292,800	5.33	37.7

Table 3.5: Heat control problem: MINRES iterations and CPU times with non-uniform grid, for a range of `l` and β.

	β					
	10^{-2}		10^{-3}		10^{-4}	
`l`	`it`	CPU	`it`	CPU	`it`	CPU
4	20	0.76	21	0.85	19	0.88
5	20	4.15	21	4.25	22	4.83
6	20	28.1	21	29.2	22	30.6
7	20	223	21	234	22	245
8	20	1946	24	2324	23	2232

obtain a fixed accuracy should grow like $\mathcal{O}(h^{-4})$ for backward Euler as opposed to $\mathcal{O}(h^{-3})$ for Crank–Nicolson, and the effectiveness of our new preconditioner allows this to materialize in terms of CPU time. For instance, with this Ω and t_f, the choice $h = 2^{-5}$ leads to a (Schur complement) system of dimension $8,132,481$ for backward Euler and $254,016$ for Crank–Nicolson, with the total number of degrees of freedom being $16,264,962$ for backward Euler and $508,032$ for Crank–Nicolson; as our preconditioner is optimal for all values of β, h, and τ, this leads to a substantial speed-up. Further, our preconditioned Crank–Nicolson approach is also able to obtain a solution for levels of refinement (e.g., `l` $= 7$) for which preconditioned backward Euler runs out of memory. From Table 3.4 it is also possible to understand the reason behind the latter: with our choice of τ for level of refinement `l` $= 7$, the total number of degrees of freedom is $264,289,794$; thus, simply storing the right-hand side vector requires approximately 2 GB of memory. We would like to note that, in order to overcome the running out of memory, one can exploit the Kronecker structure of the linear system arising from either a backward Euler or a Crank–Nicolson discretization, and employ a low-rank approximation of the problem, see for example [40, 166, 167, 168].

As expected, due to the bounds in Lemma 1 and Theorem 5 being independent

of h, τ, and β, the number of MINRES iterations in Tables 3.1–3.3 for the Crank–Nicolson method are roughly constant. Further, we can clearly see that the cost per MINRES iteration (as well as for the overall process) scales linearly with the problem size. This is also evident from Table 3.4, which shows the average time required to apply the preconditioner for Crank–Nicolson is linear in problem size. The second-order convergence of Crank–Nicolson is evident from Tables 3.1–3.3, until the MINRES tolerance causes slight pollution of the discretized solution (note v_{error} for `l` $= 8$ in Table 3.2 in particular). From Table 3.5 we also note the robustness of the preconditioner for a non-uniform grid.

We emphasize the importance of the choice of the multigrid routine to approximate the diagonal blocks of $A_{21} + \widehat{M}$ and its transpose. To demonstrate that our choice is appropriate, we compared the iteration numbers in Tables 3.1–3.3 with those obtained by applying the approximate inverses of each diagonal block of $A_{21} + \widehat{M}$ and its transpose using MATLAB's `backslash` routine, running the tests for the coarsest levels of refinement `l` $= 3, 4, 5$; we report the results in Table 3.6. We verified that the number of MINRES iterations for reaching convergence was the same as shown in Table 3.1–3.3, up to a difference of at most one iteration (only for `l` $= 3$ and backward Euler). This shows that the choice of 3 V-cycles of the `HSL_MI20` solver is optimal, as it resembles an "exact" solution closely enough. We also compared the results in Table 3.2 (for $\beta = 10^{-3}$) with those obtained using 5 V-cycles of the AGMG algebraic multigrid routine [118, 121, 122, 123], for all levels of refinement `l`. We report the number of MINRES iterations required when employing 5 V-cycles of the AGMG algebraic multigrid routine in Table 3.7. The numbers of iterations is the same for all levels of refinement, apart from the backward Euler system with `l` $= 3$ which requires one more MINRES iteration using AGMG, and the Crank–Nicolson system with `l` $= 8$ which requires three more iterations. This shows that AGMG is also a viable choice of multigrid routine provided more V-cycles are applied, however we elect to use 3 V-cycles of the `HSL_MI20` routine to achieve a more computationally efficient solver.

Table 3.6: Heat control problem: MINRES iterations, employing MATLAB's `backslash` for applying the approximate inverses of each diagonal block of $A_{21} + \widehat{M}$ and its transpose, for a range of β.

	Backward Euler			**Crank–Nicolson**		
	β			β		
	10^{-2}	10^{-3}	10^{-4}	10^{-2}	10^{-3}	10^{-4}
`l`	`it`	`it`	`it`	`it`	`it`	`it`
3	22	26	17	17	18	16
4	23	24	24	20	20	19
5	23	25	26	20	21	22

Finally, we comment on the issue of potential parallelism. Examining (3.20), the $(1,1)$-block $\widehat{\Phi}^{\text{BE}}$ of the backward Euler preconditioner (as opposed to the Schur complement approximation $\widehat{S}^{\text{BE}}$) affords an embarrassingly parallel imple-

Table 3.7: Heat control problem: MINRES iterations, employing 5 V-cycles of the AGMG multigrid routine for applying the approximate inverses of each diagonal block of $A_{21} + \widehat{M}$ and its transpose, for $\beta = 10^{-3}$.

	Backward Euler	**Crank–Nicolson**
`l`	`it`	`it`
3	26	18
4	24	20
5	25	21
6	27	21
7	†	21
8	†	24

mentation, that could in principle alter the balance of complexity between the two approaches shown in Tables 3.1–3.3. However, as we can see from Table 3.4, the average cost per iteration for (approximately) inverting $\widehat{\Phi}^{\mathrm{BE}}$ is negligible, in fact an order of magnitude lower in CPU time than the cost of (approximately) inverting $\widehat{S}^{\mathrm{BE}}$. Therefore, even with a "parallel" implementation of the backward Euler solver, our preconditioner for the Crank–Nicolson method would outperform it. We also point out that, if a parallel implementation of $\widehat{S}^{\mathrm{BE}}$ is possible, this could be adapted to the Crank–Nicolson system, as the structure of $\widehat{S}$ is similar to $\widehat{S}^{\mathrm{BE}}$. A parallel implementation of $\widehat{\Phi}$ may also be carried out, by suitably permuting T_1 and T_2 and applying their inverse operations in parallel.

We can therefore conclude that the Crank–Nicolson method, coupled with our new preconditioner, is significantly more potent that the widely-used preconditioned backward Euler method for all values of β.

3.5 Summary and Comments

In this chapter, we have applied an optimize-then-discretize strategy to tackle the optimal control of time-dependent PDEs, coupled with a Crank–Nicolson scheme in time. We have devised an invertible linear transformation that symmetrizes the resulting linear system, and derived a new, fast, and robust preconditioner for the saddle-point matrix, which possesses a complex structure. We have also proved that the Schur complement approximation used is optimal with respect to all parameters involved through bounds on the eigenvalues, and therefore that the preconditioner is optimal and scales linearly in CPU time with respect to matrix dimension. Finally, we have presented numerical results to demonstrate the effectiveness and speed of our preconditioned Crank–Nicolson method. In the following chapters, we will build on the strategy described here to tackle more complex problems.

Before moving on in our exposition, we would like to spend some words on the strategy presented in this chapter. Specifically, we would like to comment on the choice of eliminating the initial and the final time conditions on the state

and the adjoint variables. With this choice, rather than considering the system in (3.7), we have been working on the linear system defined in (3.8), and, by applying a suitable invertible transformation, we were able to transform it into the symmetric matrix (3.13).

We want to highlight here that it is possible to apply a transformation similar to (3.11) and transform the system (3.7) into a symmetric one. In fact, by applying the linear transformation

$$\breve{T} = \begin{bmatrix} T_2 & 0 \\ 0 & T_1 \end{bmatrix}$$

to the system (3.7), we can still obtain a symmetric matrix. However, in this case the preconditioner we derive employing the matching strategy will not be optimal anymore, as we are not able to derive bounds as in Theorem 5. In particular, the upper bounds will not hold any more, as the corresponding mixed term will be indefinite. In addition, by employing the transformation $\breve{T}$, the Schur complement approximation that one derives by employing the matching strategy will not be so easy to invert, as the approximation requires one to solve for a block tridiagonal matrix and its transpose, as well as multiplying with block diagonal matrix. In practice, not only we would lose optimality of the preconditioner, but we would also have to solve for a more complex Schur complement! This shows how a small change in perspective could lead to an impressive improvement of the solver. In addition, we would like to mention that the change in perspective we described above was not immediate, but a result of considerate thought. In practice, we needed to "get ready" before "seeing" this strategy. And now the reader is also able to understand the meaning of the inspirational quote at the beginning of the chapter.

Chapter 4

Preconditioning Time-Dependent Convection–Diffusion Control Problems with Crank–Nicolson Discretization in Time

"-Chello che è stato è stato, basta! Ricomincio da tre!
-Da zero!
-Eh?
-Da zero! Ricominci da zero!
-Nossignore, ricomincio da... cioè, tre cose me so' riuscite dint' 'a vita, pecché aggia perdere pure chelle? Che aggia ricomincia' da zero?! Da tre!"

["-What's past is past, enough! I'm starting over from three!
-From zero!
-Eh?
-From zero! You're starting over from zero!
-No, I'm starting over from... I mean, three things I succeeded in in my life, why should I lose these too? Should I start over from zero?! From three!"]

– Massimo Troisi, *Ricomincio da tre*

In the previous chapter, we derived an optimal preconditioner for the heat control problem when a Crank–Nicolson discretization is employed for approximating the time derivative. We want now to build on the advances achieved for the heat control problem, and generalize the preconditioner derived in Section 3.3 to more complex problems (quoting the inspirational quote above, we are "starting over from Chapter 3"). A natural extension of the heat control problem includes also a *convection term* in the PDE under examination: in this case, the problem we want to tackle is the *time-dependent convection–diffusion control problem*. The content of this chapter is based on some of the work in [100].

This chapter is structured as follows. In Section 4.1 we introduce the problem we consider, that is time-dependent convection–diffusion control, and outline the matrices arising upon discretization of such a problem as well as a stabilization

technique used for convection–diffusion control. In Section 4.2 we describe the linear systems obtained upon discretization of the first-order optimality conditions, and employ the linear transformation (3.11) which allows symmetrization of the linear system obtained from the Crank–Nicolson method. As for the heat control problem, this enables us to apply the symmetric iterative solver MINRES [128], which is highly desirable from the perspective of proving convergence of the iterative method. In Section 4.3 we extend the results of the preconditioner derived in Chapter 3 to time-dependent convection–diffusion control problems, using saddle-point theory along with suitable approximations of the $(1,1)$-block and Schur complement, and provide eigenvalue results for the preconditioned linear system. In Section 4.4 we demonstrate our new preconditioner's efficiency and robustness with respect to all parameters involved in the convection–diffusion control problem.

4.1 Problem Formulation

As above, in this chapter we consider the fast and robust numerical solution of time-dependent PDE-constrained optimization problems. In particular, we examine distributed convection–diffusion control problems of the form:

$$\min_{v,u} \quad J(v,u) = \frac{1}{2}\int_0^{t_f}\int_\Omega |v(x,t) - v_d(x,t)|^2 \,\mathrm{d}\Omega\mathrm{d}t + \frac{\beta}{2}\int_0^{t_f}\int_\Omega |u(x,t)|^2 \,\mathrm{d}\Omega\mathrm{d}t \quad (4.1)$$

subject to

$$\begin{cases} \dfrac{\partial v}{\partial t} - \epsilon\nabla^2 v + \mathbf{w}\cdot\nabla v = u + f(x,t) & \text{in } \Omega\times(0,t_f), \\ v(x,t) = g_D(x,t) & \text{on } \partial\Omega_D\times(0,t_f), \\ \dfrac{\partial v}{\partial \vec{n}}(x,t) = g_N(x,t) & \text{on } \partial\Omega_N\times(0,t_f), \\ v(x,0) = v_0(x) & \text{in } \Omega, \end{cases} \quad (4.2)$$

where the variables v, v_d, and u are the state, desired state, and control variables, respectively, $\beta > 0$ is a regularization parameter, $\epsilon > 0$ is the diffusion coefficient, and $\mathbf{w}$ is a divergence-free wind (or flow) vector (i.e., $\nabla\cdot\mathbf{w} = 0$). The problem is solved on a spatial domain $\Omega \subset \mathbb{R}^d$, $d \in \{1,2,3\}$, with boundary such that $\overline{\partial\Omega} = \overline{\partial\Omega}_D \cup \overline{\partial\Omega}_N$, $\partial\Omega_D \cap \partial\Omega_N = \varnothing$, up to a final time $t_f > 0$, that is $(x,t) \in \Omega\times(0,t_f)$. Here $\dfrac{\partial v}{\partial\vec{n}}(x,t)$ represents the (outward) normal derivative of v on $\partial\Omega_N$. The functions f, g_D, g_N, and v_0 are known.

In (4.2), the term $-\epsilon\nabla^2 v$ denotes the diffusive element, and the term $\mathbf{w}\cdot\nabla v$ represents convection. In physical (real-world) problems, as pointed out for example in [44, Ch. 6], convection typically plays a more significant physical role than diffusion. In particular, defining the *Péclet number* as $Pe = Le\|\mathbf{w}\|/\epsilon$, where Le denotes the characteristic length for the domain Ω, we have that $Pe \gg 1$ for many practical problems. However this in turn makes the problem more difficult to solve [44, 150] as the solution procedure will need to be robust with respect to

the direction of the wind $\mathbf{w}$ and any boundary or internal layers that form. The presence of boundary or internal layers is also an issue that we have to deal with when discretizing the convection–diffusion differential operator. Indeed, when solving a convection–diffusion problem, a stabilization procedure is often utilized in order to 'capture' all the layers.

4.1.1 Discretization Matrices and Stabilization

To illustrate the matrices involved in the finite element discretizations of the optimal control problem under examination, consider a standard Galerkin finite element discretization for the (steady-state) convection–diffusion problem:

$$-\epsilon\nabla^2 v + \mathbf{w}\cdot\nabla v = u + f(x) \qquad \text{in } \Omega\,. \tag{4.3}$$

Letting $\{\phi_i\}_{i=1}^{n_x}$ be the same finite element basis functions for v and u, then we would like to find approximations $v(x) \approx \sum_{i=1}^{n_x} v_i\phi_i$, $u(x) \approx \sum_{i=1}^{n_x} u_i\phi_i$. We recall that for control problems of the form (4.1)–(4.2) it is possible to choose different bases for v and u, but it is often preferable to use the same finite element basis functions for both the state and the control, and we do so here to obtain a system of convenient structure, as we will eliminate the control variable *a priori*. Letting the vectors $\boldsymbol{v} = \{v_i\}_{i=1}^{n_x}$, $\boldsymbol{u} = \{u_i\}_{i=1}^{n_x}$, a discretized version of (4.3) is

$$L\boldsymbol{v} := (\epsilon K + N + W_{\mathbf{w}})\boldsymbol{v} = M\boldsymbol{u} + \boldsymbol{f},$$

where K and M are the stiffness and mass matrix in the chosen finite element basis, respectively, and

$$N = \{n_{il}\}_{i,l=1}^{n_x}, \qquad n_{il} = \int_\Omega (\mathbf{w}\cdot\nabla\phi_l)\phi_i \,\mathrm{d}\Omega\,,$$
$$\boldsymbol{f} = \{f_i\}_{i=1}^{n_x}, \qquad f_i = \int_\Omega f\phi_i \,\mathrm{d}\Omega\,,$$

and the matrix $W_{\mathbf{w}}$ denotes a possible stabilization matrix for the convection operator; here, the subscript denotes the dependence on the wind $\mathbf{w}$. Note that these definitions exclude the effects of the boundary conditions: for instance, if non-zero Dirichlet conditions are present extra terms will appear in $\boldsymbol{f}$ relating to these, and the appropriate dimensions of the matrices will depend on the structure of the boundary conditions. We recall that K is generally referred to as a stiffness matrix, and is symmetric positive definite (unless $\partial\Omega_D = \varnothing$ in which case it is symmetric positive semi-definite), and M is referred to as a mass matrix, which is symmetric positive definite. The matrix N is skew-symmetric (meaning $N + N^\top = 0$) in the case of Dirichlet problems; otherwise we obtain that [44, Sec. 6.5]

$$n_{il} + n_{li} = \int_{\partial\Omega_N} \phi_i\phi_l\mathbf{w}\cdot\boldsymbol{n} \,\mathrm{d}s\,,$$

using the Divergence Theorem, where $\boldsymbol{n}$ denotes the (outer) unit normal vector. In this latter case the spectral properties of the matrix $H := N + N^\top$, which can

be indefinite, will be useful for our subsequent analysis. Defining the *boundary mass matrix* $M_{\partial\Omega_N} = \{m_{il}^{\partial\Omega_N}\}_{i,l=1}^{n_x}$, where $m_{il}^{\partial\Omega_N} = \int_{\partial\Omega_N} \phi_i \phi_l \,\mathrm{d}s$, and letting $c = \max_x \|\mathbf{w}\|$, we may write

$$\begin{aligned}\mathbf{y}^\top (H + cM_{\partial\Omega_N})\mathbf{y} &= \sum_{i=1}^{n_x}\sum_{l=1}^{n_x} y_i \left[\int_{\partial\Omega_N} (\mathbf{w}\cdot\boldsymbol{n} + c)\,\phi_i\phi_l \,\mathrm{d}s\right] y_l \\ &= \int_{\partial\Omega_N} (\mathbf{w}\cdot\boldsymbol{n} + c)\, y^2 \,\mathrm{d}s \geqslant 0\end{aligned}$$

for any $\mathbf{0} \neq \mathbf{y} \in \mathbb{R}^{n_x}$, where we set $y := \sum_{i=1}^{n_x} y_i \phi_i$. Therefore,

$$H \geq -cM_{\partial\Omega_N}, \tag{4.4}$$

where the notation $\Upsilon_1 \geq \Upsilon_2$ means $\Upsilon_1 - \Upsilon_2$ is positive semi-definite. This observation will be useful when discussing our approach for problems with Neumann or mixed boundary conditions.

Concerning the stabilization scheme used for solving the forward convection–diffusion problem, a popular stabilized finite element method is the *Streamline Upwind Petrov–Galerkin (SUPG)* method [87]. This stabilization method modifies the right-hand side $\boldsymbol{f}$, aside from adding a further matrix arising in the discretization of the differential operator. In fact, for the forward problem the stabilization is defined as

$$W_{\mathbf{w}} = \{w_{il}\}_{i,l=1}^{n_x}, \quad w_{il} = \delta \int_\Omega (\mathbf{w}\cdot\nabla\phi_i)(\mathbf{w}\cdot\nabla\phi_l)\,\mathrm{d}\Omega - \epsilon\delta \sum_m \int_{\Delta_m} (\nabla^2\phi_i)(\mathbf{w}\cdot\nabla\phi_l)\mathrm{d}\Omega\,,$$

where Δ_m is the m-th element in our finite element discretization, while the right-hand side $\boldsymbol{f}$ is defined as

$$\boldsymbol{f} = \{f_i\}_{i=1}^{n_x}, \qquad f_i = \int_\Omega f\phi_i \,\mathrm{d}\Omega + \delta \int_\Omega f\mathbf{w}\cdot\nabla\phi_i \,\mathrm{d}\Omega\,.$$

Here, the parameter $\delta > 0$ is called the *stabilization parameter*. The SUPG method has been proven to have order of convergence of $\mathcal{O}(h^{3/2})$ in the *streamline diffusion norm*[5] in the case of the forward convection–diffusion equation when using bilinear finite elements for instance [44, Ch. 6]. However, as pointed out in [35, 79], applying this scheme to the (steady-state) control problem gives rise to extra difficulties. Specifically, in [35] it has been shown that applying the scheme to the control problem with the discretize-then-optimize strategy leads to symmetric discrete equations in which the discrete adjoint problem is not a consistent discretization of the continuous adjoint problem, whereas the optimize-then-discretize approach gives rise to a different, non-symmetric discretized system which therefore does not possess the structure of a discrete optimization problem. Further, in [79] the authors prove that the order of convergence of the SUPG method applied to the control problem is only linear in the presence of boundary layers. In this work, we thus employ the adjoint-consistent *Local Pro-*

[5]The streamline diffusion norm is defined as $\|v\|_{\mathrm{sd}} := (\epsilon\|\nabla v\|^2 + \delta\|\mathbf{w}\cdot\nabla v\|^2)^{1/2}$.

jection Stabilization (LPS) approach described in [11, 12, 23, 141], for which the discretization and optimization steps commute in the stationary case. Further, these stabilized finite elements leads to an order of convergence of $\mathcal{O}(h^{3/2})$ for the L^2-error [12], which is optimal on general quasi-uniform meshes for the forward problem, see, e.g., [188]. Before describing the LPS formulation in detail, we mention that the strategy described in this work can be employed with any stabilized finite element method for which the stabilization of the adjoint of the convection operator is equal to the adjoint of the stabilization applied to the forward operator, that is to say $W_{-\mathbf{w}} = (W_{\mathbf{w}})^\top$. In the LPS formulation, the stabilization matrix $W_{\mathbf{w}}$ is defined as

$$W_{\mathbf{w}} = \{w_{il}\}_{i,l=1}^{n_x}, \quad w_{il} = \delta \int_\Omega [\mathbf{w}\cdot\nabla\phi_i - \pi_h(\mathbf{w}\cdot\nabla\phi_i)][\mathbf{w}\cdot\nabla\phi_l - \pi_h(\mathbf{w}\cdot\nabla\phi_l)]\mathrm{d}\Omega. \quad (4.5)$$

Here, $\delta > 0$ denotes a stabilization parameter, and π_h is an orthogonal projection operator. From (4.5), the matrix $W_{\mathbf{w}}$ can be viewed as a shifted discrete diffusion operator associated with the streamline direction defined by $\mathbf{w}$. It is symmetric and positive semi-definite, as shown by following the working in [44, p. 17]: letting $\mathbf{0} \neq \mathbf{y} \in \mathbb{R}^{n_x}$, and setting $\pi_h^i = \pi_h(\mathbf{w}\cdot\nabla\phi_i)$, $y := \sum_{i=1}^{n_x} y_i\phi_i$, $\widetilde{\pi} := \sum_{i=1}^{n_x} v_i\pi_h^i$, we have

$$\begin{aligned}\mathbf{y}^\top W_{\mathbf{w}}\mathbf{y} &= \delta \sum_{i=1}^{n_x}\sum_{l=1}^{n_x} y_i \left[\int_\Omega [\mathbf{w}\cdot\nabla\phi_i - \pi_h^i][\mathbf{w}\cdot\nabla\phi_l - \pi_h^l]\,\mathrm{d}\Omega\right] y_l \\ &= \delta \int_\Omega (\mathbf{w}\cdot\nabla y - \widetilde{\pi})\,(\mathbf{w}\cdot\nabla y - \widetilde{\pi})\,\mathrm{d}\Omega = \delta\,\|\mathbf{w}\cdot\nabla y - \widetilde{\pi}\|^2_{L^2(\Omega)} \geqslant 0,\end{aligned}$$

where we have used that $\sum_{i=1}^{n_x} y_i\nabla\phi_i = \nabla y$.

For the convergence of the method, we require π_h to be an L^2-orthogonal (discontinuous) projection operator defined on patches of the domain Ω that satisfies the approximation and stability properties specified in [12], where by a patch we mean the union of elements of our finite element discretization; the projection operator π_h is left free to be discontinuous on the edges of the patches. In our implementation we will make use of $\mathbf{Q}_1$ elements, so the domain is divided into patches consisting of 2 elements in each dimension. To ensure the aforementioned properties are satisfied, as in [11] we define π_h as

$$\pi_h(q)|_{\mathtt{P}} = \frac{1}{|\mathtt{P}|}\int_{\mathtt{P}} q\,\mathrm{d}\mathtt{P}, \quad \forall q \in L^2(\Omega), \quad (4.6)$$

where $\pi_h(q)|_{\mathtt{P}}$ is the restriction of $\pi_h(q)$ to the patch $\mathtt{P}$, and $|\mathtt{P}|$ is the (Lebesgue) measure of the patch. We refer again to [12] for the theoretical proof of the convergence of this method with this definition of the local projection operator. In order to simplify the notation we note that, from the definition (4.5) of $W_{\mathbf{w}}$ with the choice of (4.6) as local projection operator, it results that $W_{\mathbf{w}} = (W_{\mathbf{w}})^\top =$

$W_{-\mathbf{w}} =: W$. As in [44, p. 253], we choose δ locally on each patch $\mathtt{P}_m$ as δ_m, with

$$\delta_m = \begin{cases} \dfrac{h_m}{2\|\mathbf{w}_m\|}\left(1 - \dfrac{1}{Pe_m}\right) & \text{if } Pe_m > 1, \\ 0 & \text{if } Pe_m \leqslant 1, \end{cases}$$

where $\|\mathbf{w}_m\|$ is the (vector) ℓ^2-norm of the wind at the patch centroid, h_m is a measure of the patch length in the direction of the wind, and $Pe_m = \|\mathbf{w}_m\| h_m/(2\epsilon)$ is the *patch Péclet number*.

We observe that, with this (non-constant) choice of δ, the matrix W is still positive semi-definite. To prove this, it is sufficient to define $\pi_h^{i,m} = \pi_h(\mathbf{w}\cdot\nabla\phi_i)|_{\mathtt{P}_m}$ and $\widetilde{\pi}_m := \sum_{i=1}^{n_x} y_i \pi_h^{i,m}$ locally on each patch and then proceed as above, obtaining

$$\mathbf{y}^\top W \mathbf{y} = \sum_m \delta_m \|\mathbf{w}\cdot\nabla y - \widetilde{\pi}_m\|^2_{L^2(\mathtt{P}_m)} \geqslant 0.$$

The spectral properties of the matrices K, M, and W (whether Dirichlet, Neumann or mixed boundary conditions are imposed) will be useful later, when discussing the optimality of our preconditioning approach.

Informed by the definitions of this section, we make the following assumption when carrying out the theoretical analysis in the remainder of this chapter, and later discuss how our methodology could be applied if the assumption is relaxed:

Assumption 1. *We assume* $\mathbf{w}$ *is such that* $\mathbf{w}\cdot\boldsymbol{n} = 0$ *on* $\partial\Omega_N$. *In case a pure Dirichlet problem (i.e.,* $\partial\Omega_N = \varnothing$*) is solved, we remove this restriction on the wind* $\mathbf{w}$.

When Assumption 1 holds, the matrix $H = 0$. There are a number of wind vectors $\mathbf{w}$ that satisfy the property above on the whole of $\partial\Omega$, see for example the well-known 'recirculating wind' example described in [44, p. 240].

4.2 First-Order Optimality Conditions and Discretization in Time

We now describe the strategy used for obtaining an approximate solution of (4.1)–(4.2). We apply an all-at-once approach coupled with the optimize-then-discretize scheme, in which the continuous Lagrangian is used to arrive at first-order optimality conditions, which are then discretized. For simplicity the working of this section considers Dirichlet boundary conditions, that is $\partial\Omega_D = \partial\Omega$, but it may be readily extended to problems where $\partial\Omega_D \subset \partial\Omega$. Introducing the adjoint variable ζ, we consider the Lagrangian associated to (4.1)–(4.2) as in [12]. Then, by deriving the Karush–Kuhn–Tucker conditions, the solution of (4.1)–(4.2) satisfies

(see, for instance, [12] for stationary convection–diffusion control):

$$\left\{\begin{array}{rll} \dfrac{\partial v}{\partial t} - \epsilon\nabla^2 v + \mathbf{w}\cdot\nabla v = \dfrac{1}{\beta}\,\zeta + f & \text{in } \Omega\times(0,t_f) & \\ v(x,t) = g(x,t) & \text{on } \partial\Omega\times(0,t_f) & \text{state} \\ v(x,0) = v_0(x) & \text{in } \Omega & \text{equation} \\ -\dfrac{\partial \zeta}{\partial t} - \epsilon\nabla^2\zeta - \mathbf{w}\cdot\nabla\zeta \;=\; v_d - v & \text{in } \Omega\times(0,t_f) & \\ \zeta(x,t) = 0 & \text{on } \partial\Omega\times(0,t_f) & \text{adjoint} \\ \zeta(x,t_f) = 0 & \text{in } \Omega & \text{equation} \end{array}\right. \tag{4.7}$$

where we have substituted the gradient equation $\beta u - \zeta = 0$ into the state equation.

Problem (4.7) is a coupled system of (time-dependent) PDEs, consisting of a forward PDE combined with a backward problem for the adjoint. As done in Chapter 3, we will employ a Crank–Nicolson discretization for approximating the time derivative, since this will allow the user to choose a time step that is independent of the spatial mesh-size. Again, in order to obtain a consistent system of linear equations, both functions v and ζ are approximated at the same time points. For the remainder of the chapter, we discretize the interval $(0,t_f)$ into n_t subintervals of length $\tau = \frac{t_f}{n_t}$, and we use the notation $\boldsymbol{v}_n \approx v(x,t_n)$, $\boldsymbol{\zeta}_n \approx \zeta(x,t_n)$ for our approximations for all $x\in\Omega$, with $t_n = n\tau$.

Considering again (4.7), we now employ the Crank–Nicolson method for discretizing the time derivative. Denoting

$$L^+ = \frac{\tau}{2}L + M, \quad L^- = \frac{\tau}{2}L - M, \quad \bar{M} = \frac{\tau}{2}M, \quad \bar{M}_\beta = \frac{\tau}{2\beta}M,$$

we have that the numerical solution of (4.7) satisfies

$$\left\{\begin{array}{l} \bar{M}\left(\boldsymbol{v}_n + \boldsymbol{v}_{n+1}\right) + (L^+)^\top\boldsymbol{\zeta}_n + (L^-)^\top\boldsymbol{\zeta}_{n+1} = \bar{M}\left(\boldsymbol{v}_d^n + \boldsymbol{v}_d^{n+1}\right), \\ \\ \qquad L^-\boldsymbol{v}_n + L^+\boldsymbol{v}_{n+1} - \bar{M}_\beta\left(\boldsymbol{\zeta}_n + \boldsymbol{\zeta}_{n+1}\right) = \dfrac{\tau}{2}\left(\boldsymbol{f}^n + \boldsymbol{f}^{n+1}\right), \end{array}\right.$$

for $n = 0, 1, ..., n_t-1$, with $M\boldsymbol{\zeta}_{n_t} = \mathbf{0}$ and $M\boldsymbol{v}_0 = M\boldsymbol{v}^0$ appropriate discretizations of the final and initial conditions on ζ and v, and $\boldsymbol{f}^n$ defined as in (3.4). In matrix form, we write

$$\left[\begin{array}{cc} \bar{A}_{11} & \bar{A}_{12} \\ \bar{A}_{21} & \bar{A}_{22} \end{array}\right]\left[\begin{array}{c} \bar{\boldsymbol{v}} \\ \bar{\boldsymbol{\zeta}} \end{array}\right] = \left[\begin{array}{c} \bar{\mathbf{b}}_1 \\ \bar{\mathbf{b}}_2 \end{array}\right], \tag{4.8}$$

where the vectors $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{\zeta}}$ are the numerical solution for the state and adjoint variables, and as before the right-hand side accounts for the initial and final-time conditions on v and ζ, as well as the desired state v_d and force function f. The

matrices $\bar{A}_{ij}$, $i,j=1,2$, are given by

$$\bar{A}_{11}=\begin{bmatrix} \bar{M} & \bar{M} & & \\ & \ddots & \ddots & \\ & & \bar{M} & \bar{M} \\ & & & 0 \end{bmatrix}, \qquad \bar{A}_{12}=\begin{bmatrix} (L^+)^\top & (L^-)^\top & & \\ & \ddots & \ddots & \\ & & (L^+)^\top & (L^-)^\top \\ & & & M \end{bmatrix},$$

$$\bar{A}_{21}=\begin{bmatrix} M & & & \\ L^- & L^+ & & \\ & \ddots & \ddots & \\ & & L^- & L^+ \end{bmatrix}, \qquad \bar{A}_{22}=-\begin{bmatrix} 0 & & & \\ \bar{M}_\beta & \bar{M}_\beta & & \\ & \ddots & \ddots & \\ & & \bar{M}_\beta & \bar{M}_\beta \end{bmatrix}.$$

As done in Section 3.2.2 for the heat control problem, if we eliminate the initial and final-time conditions on v and ζ, we can rewrite

$$\underbrace{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & -A_{22} \end{bmatrix}}_{\bar{\mathcal{A}}} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix},$$

with the vectors $\boldsymbol{v}$, $\boldsymbol{\zeta}$, $\mathbf{b}_1$, $\mathbf{b}_2$ modified accordingly, and the matrices A_{ij}, $i,j=1,2$, given by

$$A_{11}=\begin{bmatrix} \bar{M} & & & \\ \bar{M} & \bar{M} & & \\ & \ddots & \ddots & \\ & & \bar{M} & \bar{M} \end{bmatrix}, \qquad A_{12}=\begin{bmatrix} (L^+)^\top & (L^-)^\top & & \\ & \ddots & \ddots & \\ & & (L^+)^\top & (L^-)^\top \\ & & & (L^+)^\top \end{bmatrix},$$

$$A_{21}=\underbrace{\begin{bmatrix} L^+ & & & \\ L^- & L^+ & & \\ & \ddots & \ddots & \\ & & L^- & L^+ \end{bmatrix}}_{=A_{12}^\top}, \qquad A_{22}=\begin{bmatrix} \bar{M}_\beta & \bar{M}_\beta & & \\ & \ddots & \ddots & \\ & & \bar{M}_\beta & \bar{M}_\beta \\ & & & \bar{M}_\beta \end{bmatrix}. \tag{4.9}$$

In order to symmetrize the system, we now apply the linear transformation T defined as in (3.11), with $T_1, T_2 \in \mathbb{R}^{(n_t n_x)\times(n_t n_x)}$ defined as in (3.12). Then,

$$T\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi & \Psi^\top \\ \Psi & -\Theta \end{bmatrix}}_{\mathcal{A}}\begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \end{bmatrix} = T\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \tag{4.10}$$

where

$$\Phi = T_1 A_{11} = \begin{bmatrix} 2\bar{M} & \bar{M} & & \\ \bar{M} & \ddots & \ddots & \\ & \ddots & 2\bar{M} & \bar{M} \\ & & \bar{M} & \bar{M} \end{bmatrix}, \Theta = T_2 A_{22} = \begin{bmatrix} \bar{M}_\beta & \bar{M}_\beta & & \\ \bar{M}_\beta & 2\bar{M}_\beta & \ddots & \\ & \ddots & \ddots & \bar{M}_\beta \\ & & \bar{M}_\beta & 2\bar{M}_\beta \end{bmatrix}.$$

Furthermore, it holds that

$$\Psi = T_2 A_{21} = \begin{bmatrix} L^+ & & & & \\ \tau L & L^+ & & & \\ L^- & \tau L & L^+ & & \\ & \ddots & \ddots & \ddots & \\ & & L^- & \tau L & L^+ \end{bmatrix} = (T_1 A_{12})^\top.$$

As for the heat control problem, we have transformed (4.8) to a symmetric system using matrices T_1, T_2 which are easy and computationally cheap to apply. Further, we may easily apply their inverses to a vector using only a sequence of block updates. Finally, we can rewrite the $(1,1)$- and the $(2,2)$-blocks of the matrix $\mathcal{A}$ as in (3.14), with Φ_D and Θ_D defined as in (3.15). We recall that, in this way, we can work with the matrices Φ and Θ cheaply, using T_1, T_2, Φ_D, Θ_D, and that, since both Φ_D and Θ_D are symmetric positive definite, the same holds for Φ and Θ.

Therefore, in order to find an approximate solution to (4.7), we may now consider the saddle-point system (4.10), to which we can apply a preconditioned Krylov subspace method for symmetric indefinite matrices, such as MINRES.

4.3 Preconditioning Approach

In this section we describe an optimal preconditoner for the system (4.10), by making use of saddle-point theory as well as suitable approximations for the blocks of the matrix $\mathcal{A}$. The preconditioner we derive is a natural extension of the one for heat control problems derived in Section 3.3.

As we discussed in Section 2.10, given an invertible system of the form (4.10), with invertible Φ, one may use the block diagonal matrix $\mathcal{P}_3$ defined in (2.28) as a preconditioner. However, as we noted in Section 2.10, the computational cost for applying the inverse of $\mathcal{P}_3$ would be comparable to that of applying the inverse of $\mathcal{A}$. For this reason, we wish to find a suitable approximation $\widehat{\mathcal{P}}_3$ of $\mathcal{P}_3$, with

$$\widehat{\mathcal{P}}_3 = \begin{bmatrix} \widehat{\Phi} & 0 \\ 0 & \widehat{S} \end{bmatrix},$$

or, more precisely, a cheap application of the effect of $\widehat{\mathcal{P}}_3^{-1}$ on a generic vector.

As we described in Section 3.3.1, a good approximation of Φ is given by $\widehat{\Phi} = T_1 \widehat{\Phi}_D T_1^\top$, with

$$\widehat{\Phi}_D = \frac{\tau}{2} \begin{bmatrix} M_c & & \\ & \ddots & \\ & & M_c \end{bmatrix},$$

where M_c denotes a fixed number of steps (20, in our tests) of Chebyshev semi-iteration applied to M. We thus need only to find an approximation to the Schur complement S. We will do so by extending the strategy described in Section 3.3.2 for the heat control problem.

4.3.1 Approximation of Schur Complement

We now find a suitable approximation for the Schur complement S of (4.10). In the forthcoming theory, we assume that Assumption 1 holds, and later discuss the case where this is relaxed. Note that the approximation we derive below is a generalization of the Schur complement approximation derived in Section 3.3.2 for heat control.

As done in Section 3.3.2, we rewrite the matrices Φ, Ψ, and Θ in terms of T_1 and T_2, obtaining the following expression for S:

$$S = T_2\Theta_D T_2^\top + T_2 A_{21}(T_1\Phi_D T_1^\top)^{-1}A_{21}^\top T_2^\top = T_2\left[\Theta_D + \Psi_D\Phi_D^{-1}\Psi_D^\top\right]T_2^\top, \quad (4.11)$$

where we set

$$\Psi_D = A_{21}T_2^{-1}. \quad (4.12)$$

As for the heat control problem, if we find a symmetric positive definite approximation $\widetilde{S}_{\text{int}}$ of

$$S_{\text{int}} = \Theta_D + \Psi_D\Phi_D^{-1}\Psi_D^\top, \quad (4.13)$$

then $\widehat{S} := T_2\widetilde{S}_{\text{int}}T_2^\top$ is a symmetric positive definite approximation of S. Then, we can derive upper and lower bounds for the eigenvalues of the matrix $\widehat{S}^{-1}S$ by employing the (generalized) Rayleigh quotient (3.25), as done in Section 3.3.2.

As for heat control, we can exploit the structure of the matrices Φ_D and Θ_D in (3.15), and apply the matching strategy to (4.13) in order to find an approximation $\widetilde{S}_{\text{int}}$ of S_{int}. We seek an approximation:

$$\widetilde{S}_{\text{int}} = (A_{21} + \widehat{M})\Phi^{-1}(A_{21} + \widehat{M})^\top \approx S_{\text{int}} \quad (4.14)$$

such that

$$\widehat{M}\Phi^{-1}\widehat{M}^\top = \left[\widehat{M}(T_1^\top)^{-1}\right]\Phi_D^{-1}\left[T_1^{-1}\widehat{M}^\top\right] = \Theta_D.$$

Following the work in Section 3.3.2, for this to hold the matrix $\widehat{M}$ is given by

$$\widehat{M} = \frac{\tau}{2\sqrt{\beta}}\begin{bmatrix} M & & & \\ M & M & & \\ & \ddots & \ddots & \\ & & M & M \end{bmatrix}. \quad (4.15)$$

Finally, our approximation of S is given by

$$\widehat{S} = T_2(A_{21} + \widehat{M})\Phi^{-1}(A_{21} + \widehat{M})^\top T_2^\top = (A_{21} + \widehat{M})\Phi_D^{-1}(A_{21} + \widehat{M})^\top, \quad (4.16)$$

with $\widehat{M}$ as defined in (4.15), and the two expressions are equivalent since T_2 commutes with both A_{21} and $\widehat{M}$. To understand the effectiveness of this approximation, we again follow the work in Section 3.3.2, and study the spectrum of the matrix $\widetilde{S}_{\text{int}}^{-1}S_{\text{int}}$. We first rewrite $\widetilde{S}_{\text{int}}$ as follows, as in (3.31):

$$\widetilde{S}_{\text{int}} = S_{\text{int}} + M_D\Phi_D^{-1}\Psi_D^\top + \Psi_D\Phi_D^{-1}M_D^\top.$$

Then, we consider the generalized Rayleigh quotient:

$$R := \frac{\mathbf{x}^\top S_{\text{int}} \mathbf{x}}{\mathbf{x}^\top \widetilde{S}_{\text{int}} \mathbf{x}} = \frac{\mathbf{a}^\top \mathbf{a} + \mathbf{b}^\top \mathbf{b}}{\mathbf{a}^\top \mathbf{a} + \mathbf{b}^\top \mathbf{b} + \mathbf{a}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{a}}, \tag{4.17}$$

where $\mathbf{a} = (\Psi_D \Phi_D^{-1/2})^\top \mathbf{x}$ and $\mathbf{b} = (\Theta_D^{1/2})^\top \mathbf{x}$. Working as in Section 2.11, since $\Phi \succ 0$ and $\Theta \succ 0$, for Theorem 1 we have $R \geqslant \frac{1}{2}$.

In order to find an upper bound for the Rayleigh quotient (4.17), we again note that

$$\widetilde{S}_{\text{int}} = S_{\text{int}} + \frac{1}{\sqrt{\beta}} \left(\Psi_D^\top + \Psi_D \right).$$

Then, following the reasoning in [139] and the reasoning from the proof of Theorem 2, we can prove that $R \leqslant 1$. From (4.17), this holds if

$$\mathbf{a}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{a} = \frac{1}{\sqrt{\beta}} \mathbf{z}^\top \left(A_{21}^\top T_2 + T_2^\top A_{21} \right) \mathbf{z} \geqslant 0 \,,$$

where we set $\mathbf{z} = T_2^{-1} \mathbf{x}$. Therefore, we wish to show that the matrix $\mathcal{X} = A_{21}^\top T_2 + T_2^\top A_{21}$ is positive semi-definite. We easily obtain that

$$\mathcal{X} = \frac{\tau}{2} \underbrace{\begin{bmatrix} 2\widetilde{L} & \widetilde{L} & & \\ \widetilde{L} & \ddots & \ddots & \\ & \ddots & 2\widetilde{L} & \widetilde{L} \\ & & \widetilde{L} & \widetilde{L} \end{bmatrix}}_{=:\widetilde{\mathcal{L}}} + \underbrace{\begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & 2M \end{bmatrix}}_{=:\widetilde{\mathcal{M}}},$$

with $\widetilde{L} = L + L^\top = 2(\epsilon K + W)$ since both K and W are symmetric, and N is skew-symmetric due to Assumption 1. Furthermore, since K is positive definite in this case, with W positive semi-definite, $\widetilde{L}$ is also positive definite. Therefore, by observing that also the matrix $\mathcal{T}$ as defined as in Section 3.3.2 is symmetric positive definite, and by employing Theorem 3, we can infer that $\widetilde{\mathcal{L}} = \mathcal{T} \otimes \widetilde{L}$ is symmetric positive definite. Again, the matrix $\widetilde{\mathcal{M}}$ is clearly symmetric positive semi-definite. Finally, from $\widetilde{\mathcal{L}} \succ 0$ and $\widetilde{\mathcal{M}} \succeq 0$ we can imply that $\mathcal{X} \succ 0$, and therefore

$$\mathbf{a}^\top \mathbf{b} + \mathbf{b}^\top \mathbf{a} \geqslant 0,$$

with $\mathbf{a}$ and $\mathbf{b}$ as defined above. Finally, the last inequality guarantees that the Rayleigh quotient R in (4.17) satisfies $R \leqslant 1$.

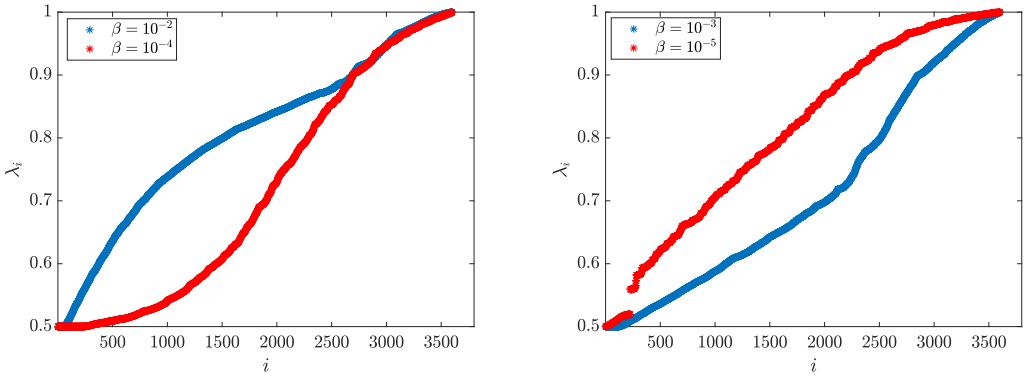
We have hence proved the following result:

Theorem 6. *Let S_{int} and $\widetilde{S}_{\text{int}}$ be defined as in* (4.13) *and* (4.14) *respectively, with the matrices Φ_D, Ψ_D, Θ_D, A_{21}, Φ, $\widehat{M}$ defined as in* (3.15), (4.12), (4.9), (3.14), *and* (4.15). *Then, given Assumption 1:*

$$\lambda(\widetilde{S}_{\text{int}}^{-1} S_{\text{int}}) \in \left[\frac{1}{2}, 1 \right].$$

In Figure 4.1 we report the eigenvalue distribution of $\widetilde{S}_{\text{int}}^{-1}S_{\text{int}}$ for a range of values of β with diffusion coefficient $\epsilon = \frac{1}{100}$, for a particular Dirichlet test problem.

Figure 4.1: Eigenvalues of $\widetilde{S}_{\text{int}}^{-1}S_{\text{int}}$ for $\beta = 10^{-j}$, $j = 2, 3, 4, 5$, with $\epsilon = \frac{1}{100}$, $\mathbf{w} = [2x_2(1-x_1^2),\ -2x_1(1-x_2^2)]^\top$ (where $x = [x_1,\ x_2]^\top$), employing $\mathbf{Q}_1$ finite elements on an evenly spaced space–time grid $(-1,1)^2 \times (0,2)$ with $\tau = h = \frac{1}{8}$.



Further, using Theorem 6 and the Rayleigh quotient (3.25), we can prove the following:

Theorem 7. *Let S and $\widehat{S}$ be defined as in* (4.11) *and* (4.16)*, with the matrices defined as in Theorem 6, and T_1, T_2 as in* (3.12)*. Then, given Assumption 1:*

$$\lambda(\widehat{S}^{-1}S) \in \left[\frac{1}{2}, 1\right].$$

Remark 5. *As for Theorems 4 and 5 in Chapter 3, no assumption has been made on the grid, meaning that the bounds in Theorem 6 and in Theorem 7 still hold in case of non-uniform meshes. We also highlight that Theorem 6 and Theorem 7 hold if no stabilization is applied to the convection–diffusion control problem.*

As we noted at the beginning of this section, the Schur complement approximation derived here is a generalization of the Schur complement approximation derived in Section 3.3.2 for heat control. As for the Schur complement approximation derived in the previous chapter, from Theorem 7 we can imply that the matrix $\widehat{S}$ in (4.16) is an effective approximation of the Schur complement S defined in (4.11). Again, we do not solve exactly a system involving the matrix $\widehat{S}$, but rather employ a cheap approximation of the effect of $\widehat{S}^{-1}$ on a generic vector. The latter is done as for heat control: since the bulk of the work involves approximately applying the inverse of $A_{21} + \widehat{M}$ and $(A_{21} + \widehat{M})^\top$, we employ block-forward and block-backward substitution, respectively, with each block diagonal approximated using a fixed number of V-cycles of a multigrid routine, for example.

Remark 6. *Let us now briefly discuss the applicability of our method if Assumption 1 does not hold, that is $\mathbf{w} \cdot \boldsymbol{n} \neq 0$ on a portion of $\partial\Omega_N$. As above, this*

results in the matrix N not being skew-symmetric. Therefore, assuming (non-trivially) a discretization of the adjoint operator has been performed such that one is examining a symmetric linear system, from (4.4) *we can derive that*

$$\widetilde{L} = 2(\epsilon K + W) + H \geq 2(\epsilon K + W) - cM_{\partial\Omega_N},$$

so a sufficient condition for the matrix $\widetilde{L}$ to be positive semi-definite is that $2(\epsilon K + W) \geq cM_{\partial\Omega_N}$. Even if the matrix $\widetilde{L}$ were not positive semi-definite, one could also argue that the upper bound on the eigenvalues in Theorem 6 (and hence that of Theorem 7) would be only slightly larger than 1 for moderate $\|\mathbf{w}\|$ and large diffusion coefficient ϵ, as the contribution (in terms of eigenvalues) of the matrix H to $\widetilde{L}$ is no greater than that of the (very sparse) mass matrix on $\partial\Omega_N$ multiplied by $\|\mathbf{w}\|$. If the discretization of the adjoint operator is such that the resulting linear system is not symmetric, we would need to apply a non-symmetric Krylov solver such as GMRES [157], potentially in conjunction with a preconditioner of the form (2.26) *for $\overline{\mathcal{A}}$.*

4.4 Numerical Results

We now provide numerical evidence of the effectiveness of our preconditioning strategy, showing the robustness of our solver with respect to all the parameters involved, for the time-dependent convection–diffusion control problem.

In all our tests we consider only Dirichlet boundary conditions (i.e., $\partial\Omega_N = \varnothing$), but we emphasize again that the method is easily generalized to Neumann and mixed boundary conditions (with the caveats previously outlined for convection–diffusion control). We employ $\mathbf{Q}_1$ finite elements for state, control, and adjoint variables. As discussed in Section 4.3, when approximating the $(1,1)$-block we employ (backward and forward) block updates for inverting T_1 and $T_2 = T_1^\top$, and apply 20 steps of Chebyshev semi-iteration to each mass matrix on the diagonal of Φ_D. Regarding the approximation of the Schur complement, as we did for heat control, in order to approximately apply the inverse operator of $A_{21} + \widehat{M}$ and its transpose we employ block-forward and block-backward substitution, respectively, with each block on the diagonal approximated with 3 V-cycles of an appropriate multigrid routine. For this problem, we employ the AGMG algebraic multigrid routine [118, 121, 122, 123], as AGMG is particularly well suited to convection-driven problems [121]. The iteration count for all tests presented starts from 0. All tests are run on MATLAB R2018b, using a 1.70GHz Intel quad-core i5 processor and 8 GB RAM on an Ubuntu 18.04.1 LTS operating system.

We consider a convection–diffusion control problem of the type (4.1)–(4.2), with $d = 2$, $\Omega = (-1,1)^2$, $t_f = 2$, $f = 0$, and wind $\mathbf{w} = [2x_2(1-x_1^2),\ -2x_1(1-x_2^2)]^\top$. The initial condition on the state v is given by

$$v(x_1, x_2, 0) = \begin{cases} 1 & \text{if } x_1 = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Setting $\partial\Omega_1 := \{1\} \times [-1, 1]$ and $\partial\Omega_2 = \partial\Omega \backslash \partial\Omega_1$, the boundary condition is given by

$$v(x_1, x_2, t) = \begin{cases} 1 & \text{on } \partial\Omega_1 \times (0, t_f), \\ 0 & \text{on } \partial\Omega_2 \times (0, t_f). \end{cases}$$

Finally, the desired state is given by

$$v_d(x_1, x_2, t) = e^{-10(1-x_1)}.$$

We run preconditioned MINRES to a tolerance of 10^{-6}, constructing a (spatial) uniform grid of mesh-size $h = 2^{1-\mathtt{l}}$ at level $\mathtt{l}$, and setting $\tau = h$ in all the tests presented. In Figure 4.2–4.3 we show the numerical solutions for the state v and adjoint variable ζ at time $t = 1$, for $\beta = 10^{-2}$ and $\mathtt{l} = 5$, with both $\epsilon = \frac{1}{20}$ and $\epsilon = \frac{1}{100}$. In Tables 4.1–4.3 we report the number of iterations `it` required for achieving convergence together with the elapsed CPU time taken in seconds, with values of $\epsilon = \frac{1}{20}$, $\frac{1}{100}$, and $\frac{1}{500}$, for a range of $\mathtt{l}$ and β. Finally, for the total size of the systems solved, we refer the reader to the column concerning the Crank–Nicolson discretization in Table 3.4.

Table 4.1: Convection–diffusion control problem: MINRES iterations and CPU times with $\epsilon = \frac{1}{20}$, for a range of $\mathtt{l}$ and β.

	$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
l	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU
3	22	0.16	24	0.14	25	0.17	21	0.10	15	0.08	10	0.05
4	22	0.82	25	1.25	26	1.07	24	1.01	19	0.76	13	0.55
5	20	1.89	25	2.20	26	1.37	26	1.38	23	3.12	17	5.81
6	22	6.75	23	6.99	25	7.59	26	7.91	26	8.06	24	7.41
7	24	64.7	24	64.4	25	66.9	25	67.1	26	68.9	26	68.9
8	24	787	24	775	26	848	25	801	25	550	26	571

Table 4.2: Convection–diffusion control problem: MINRES iterations and CPU times with $\epsilon = \frac{1}{100}$, for a range of $\mathtt{l}$ and β.

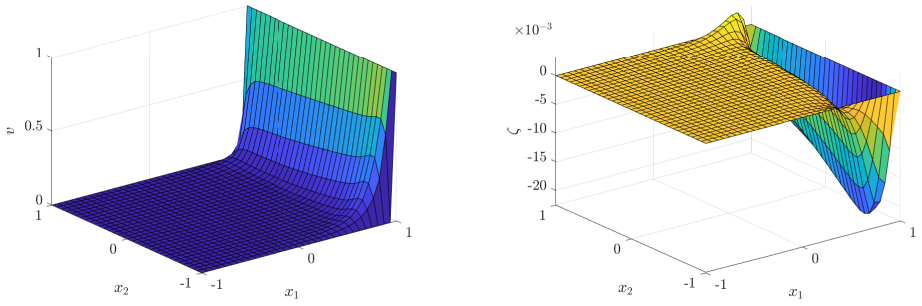
	$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
l	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU
3	23	0.14	26	0.12	23	0.10	21	0.10	15	0.10	10	0.06
4	25	0.84	26	1.14	26	1.05	25	1.04	21	0.86	15	0.63
5	24	3.68	27	3.87	26	3.78	26	3.34	23	7.01	18	5.53
6	24	14.6	25	15.2	26	14.3	26	13.4	26	19.1	22	58.8
7	23	105	25	112	26	129	26	109	26	107	25	175
8	22	647	25	783	25	791	27	705	26	705	26	652

Table 4.3: Convection–diffusion control problem: MINRES iterations and CPU times with $\epsilon = \frac{1}{500}$, for a range of $\mathtt{l}$ and β.

	$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
$\mathtt{l}$	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU
3	23	0.10	26	0.17	23	0.15	21	0.09	15	0.08	10	0.09
4	25	1.06	26	1.08	26	1.05	25	1.02	21	0.90	15	0.61
5	25	4.04	27	4.26	27	3.88	26	4.01	25	7.54	19	5.63
6	26	17.5	27	18.7	27	18.6	27	17.1	27	28.1	22	58.5
7	26	146	27	150	27	148	27	146	25	141	25	299
8	26	1174	27	1179	27	1164	27	1087	27	1010	25	1251

As can be seen from Tables 4.1–4.3, our new preconditioner is highly effective and robust, leading to convergence for all tests in at most 27 iterations. For $\beta = 10^{-5}$ or 10^{-6}, and larger values h, convergence is achieved in a lower number of iterations: this is not surprising as for these values the Schur complement is spectrally 'close' to a mass matrix, making the problem easier to solve. Apart from this, we notice that the number of iterations is independent of the parameters involved. In addition, the CPU time scales approximately linearly with problem size: as we refine the grid, the number of degrees of freedom increases by a factor of 8, and so do (roughly) the CPU times. We therefore deduce that our method is a potent one for the resolution of time-dependent convection–diffusion control problems, a class of problems which consists of substantial numerical difficulties. The number of iterations required to solve these problems is independent of mesh-size h, regularization parameter β, and diffusion coefficient ϵ.

Figure 4.2: Convection–diffusion control problem: Numerical solutions for state and adjoint variables at time $t = 1$, with $\epsilon = \frac{1}{20}$, $\beta = 10^{-2}$, and $\mathtt{l} = 5$.

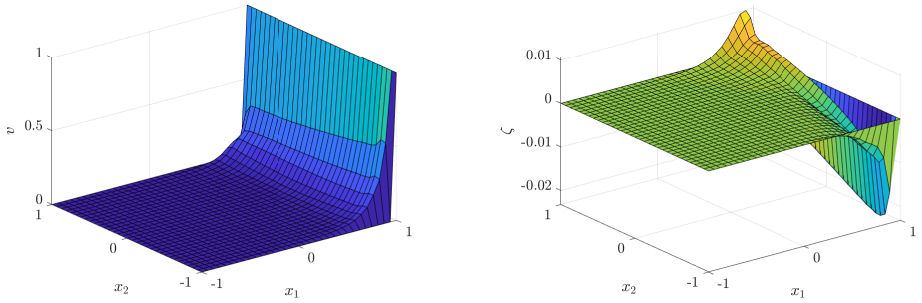


(a) State v (b) Adjoint variable ζ

4.5 Summary

In this chapter, we have applied an optimize-then-discretize strategy to tackle the optimal control of the time-dependent convection–diffusion equation, coupled

Figure 4.3: Convection–diffusion control problem: Numerical solutions for state and adjoint variables at time $t = 1$, with $\epsilon = \frac{1}{100}$, $\beta = 10^{-2}$, and $\mathtt{l} = 5$.



(a) State v (b) Adjoint variable ζ

with a Crank–Nicolson scheme in time. After eliminating the initial and final time conditions for the state and adjoint variables, we employed an invertible linear transformation that symmetrized the resulting linear system. The latter was used for deriving a new, fast, and robust preconditioner for the resulting saddle-point matrix, which possesses a complex structure. We also proved that the Schur complement approximation used is optimal with respect to all parameters involved through bounds on the eigenvalues. Finally, numerical results showed the optimality of our preconditioned Crank–Nicolson method, as the number of iterations required to reach a prescribed accuracy is roughly constant.

Chapter 5

Preconditioning Stationary and Instationary Stokes Control Problems

> *"Humankind cannot gain anything without first giving something in return. To obtain, something of equal value must be lost. That is Alchemy's First Law of Equivalent Exchange. In those days, we really believed that to be the world's one, and only, truth."*
>
> – Hiromu Arakawa, *Fullmetal Alchemist*

As we mentioned in the first chapters, one of the goals of this work is devising robust and efficient preconditioners for the distributed control of incompressible viscous fluid flow problems, which will be the topic of the following two chapters. We begin here with the distributed control of the Stokes equations, in both the stationary and instationary case, leaving the control of the Navier–Stokes equations to the following chapter. The content of this chapter is based on some of the work in [99, 101].

An example of a highly challenging problem attracting significant attention of late is the (distributed) control of incompressible viscous fluid flow problems. For such control problems, the constraints may be the (non-linear) incompressible Navier–Stokes equations or, in the limiting case of viscous flow, the (linear) incompressible Stokes equations. The study of the incompressible Stokes control problems is of particular interest *per se* in saddle-point theory (e.g., [8, 96, 153, 170, 190]), but also from a practical point of view, as the numerical solutions to those problems can be used as a starting point for the linearization of the corresponding incompressible Navier–Stokes control problems. In addition, robust and efficient preconditioners for the incompressible Stokes control problems may be "adjusted" and "tailored" in order to solve the incompressible Navier–Stokes control problems. For this reason, in this chapter we consider the control of the incompressible Stokes equations, in both the stationary and instationary settings. Our aim is to devise robust and efficient preconditioners for those type of problems, which may be then generalized to the corresponding incompressible Navier–Stokes control problems. In particular, in the following two chapters, we utilize a commutator argument for a block matrix in conjunction

with saddle-point theory in order to derive robust preconditioners for the optimal control of the incompressible Stokes and Navier–Stokes equations, in both the stationary and instationary settings. For instationary problems our approach leads to potent preconditioners when either the backward Euler or Crank–Nicolson scheme is used in the time variable.

This chapter is structured as follows. In Section 5.1, we define the problems that we examine, that is the stationary and instationary Stokes control problems; we then outline the linear systems arising upon discretization of the forward problem. In Section 5.2, we introduce the preconditioner for the forward stationary Stokes equation in combination with the commutator argument presented in [164]; the latter will then be generalized in Section 5.3 when multiple differential operators are involved in the system of equations. In Section 5.4, we derive the first-order optimality conditions of the control problems and their discretization. In Section 5.5, we present our suggested preconditioners, and in particular the commutator argument applied to the Schur complements arising from the control problems. Then, in Section 5.6 we provide numerical results that show the robustness and efficiency of our approach.

5.1 Problem Formulation

In this chapter we derive fast and robust preconditioned iterative methods for the distributed control of incompressible fluid flow, in the limiting case of viscous flow; in this case, the physics is described by the (stationary or instationary) incompressible Stokes equations. The corresponding distributed control problem is defined as a minimization of a least-squares cost functional subject to the PDEs.

Specifically, given a spatial domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, the stationary Stokes control problem we consider is

$$\min_{\vec{v},\vec{u}} \quad J_{\mathrm{S}}(\vec{v},\vec{u}) = \frac{1}{2}\int_\Omega |\vec{v}(x) - \vec{v}_d(x)|^2 \,\mathrm{d}\Omega + \frac{\beta}{2}\int_\Omega |\vec{u}(x)|^2 \,\mathrm{d}\Omega \tag{5.1}$$

subject to

$$\begin{cases} -\nabla^2\vec{v} + \nabla p = \vec{u} + \vec{f}(x) & \text{in } \Omega, \\ -\nabla\cdot\vec{v}(x) = 0 & \text{in } \Omega, \\ \vec{v}(x) = \vec{g}(x) & \text{on } \partial\Omega, \end{cases} \tag{5.2}$$

where the state variables $\vec{v}$ and p denote velocity and pressure respectively, $\vec{v}_d$ is the desired state (velocity), and $\vec{u}$ is the control variable; it is worth mentioning that $\vec{v}$, $\vec{u}$, $\vec{f}$, and $\vec{g}$ are vector functions (in $\mathbb{R}^d$), whereas p is a scalar function. Further, $\beta > 0$ is a regularization parameter. The functions $\vec{f}$ and $\vec{g}$ are known.

Similarly, the control of the instationary Stokes equations is defined as

$$\min_{\vec{v},\vec{u}} \quad J_{\mathrm{I}}(\vec{v},\vec{u}) = \frac{1}{2}\int_0^{t_f}\int_\Omega |\vec{v}(x,t) - \vec{v}_d(x,t)|^2 \,\mathrm{d}\Omega\,\mathrm{d}t + \frac{\beta}{2}\int_0^{t_f}\int_\Omega |\vec{u}(x,t)|^2 \,\mathrm{d}\Omega\,\mathrm{d}t, \tag{5.3}$$

given also a final time $t_f > 0$, subject to

$$\begin{cases} \frac{\partial \vec{v}}{\partial t} - \nabla^2 \vec{v} + \nabla p = \vec{u} + \vec{f}(x,t) & \text{in } \Omega \times (0,t_f), \\ -\nabla \cdot \vec{v}(x,t) = 0 & \text{in } \Omega \times (0,t_f), \\ \vec{v}(x,t) = \vec{g}(x,t) & \text{on } \partial\Omega \times (0,t_f), \\ \vec{v}(x,0) = \vec{v}_0(x) & \text{in } \Omega, \end{cases} \tag{5.4}$$

using the same notation as above. As for the stationary case, the functions $\vec{f}$ and $\vec{g}$ are known; the initial condition $\vec{v}_0$ is also given. In the following, we typically assume that $\vec{v}_0$ is solenoidal, i.e. $\nabla \cdot \vec{v}_0 = 0$, while adapting our strategy to the general case when possible.

Many parameter-robust preconditioners for the optimal control of the stationary incompressible Stokes equations have been derived in the literature (see, e.g., [153, 170, 190]); however, less progress has been made towards the parameter-robust solution of instationary Stokes control problems, except in the time-periodic setting [8, 96]. Below, we derive a preconditioner that will also result in a robust solver for the general formulation of the instationary distributed Stokes control problem.

5.1.1 Discretization Matrices

To introduce the discretization matrices, we consider the stationary Stokes equations:

$$\begin{cases} -\nabla^2 \vec{v} + \nabla p = \vec{u} + \vec{f}(x) & \text{in } \Omega, \\ -\nabla \cdot \vec{v}(x) = 0 & \text{in } \Omega, \end{cases} \tag{5.5}$$

with $\vec{v} = \vec{g}$ on $\partial\Omega$. First, we introduce the weak formulation of (5.5) as follows. Let $V := \{\vec{v} \in \mathcal{H}^1(\Omega)^d \mid \vec{v} = \vec{g} \text{ on } \partial\Omega\}$, $V_0 := \{\vec{v} \in \mathcal{H}^1(\Omega)^d \mid \vec{v} = \vec{0} \text{ on } \partial\Omega\}$, and $Q := L^2(\Omega)$, with $\mathcal{H}^1(\Omega)^d$ the Sobolev space of square-integrable functions in $\mathbb{R}^d$ with square-integrable weak derivatives; then, the weak formulation reads as:

Find $\vec{v} \in V$ and $p \in Q$ such that

$$\begin{cases} (\nabla \vec{v}, \nabla \vec{w}) - (p, \nabla \cdot \vec{w}) = (\vec{u}, \vec{w}) + (\vec{f}, \vec{w}) & \text{for all } \vec{w} \in V_0, \\ -(q, \nabla \cdot \vec{v}\,) = 0 & \text{for all } q \in Q, \end{cases} \tag{5.6}$$

where $(\cdot,\cdot)$ is the L^2-inner product on Ω.

Then, letting $\{\vec{\phi}_i\}_{i=1}^{n_v}$ and $\{\psi_i\}_{i=1}^{n_p}$ be inf–sup stable finite element basis functions, we seek approximations $\vec{v}(x) \approx \sum_{i=1}^{n_v} \boldsymbol{v}_i \vec{\phi}_i$, $\vec{u}(x) \approx \sum_{i=1}^{n_v} \boldsymbol{u}_i \vec{\phi}_i$, $p(x) \approx \sum_{i=1}^{n_p} p_i \psi_i$. Denoting the vectors $\boldsymbol{v} = \{\boldsymbol{v}_i\}_{i=1}^{n_v}$, $\boldsymbol{u} = \{\boldsymbol{u}_i\}_{i=1}^{n_v}$, $\boldsymbol{p} = \{p_i\}_{i=1}^{n_p}$, a discretized version of (5.6) is:

$$\begin{cases} \mathbf{K}\,\boldsymbol{v} + B^\top \boldsymbol{p} = \mathbf{M}\boldsymbol{u} + \boldsymbol{f}, \\ B\,\boldsymbol{v} = \mathbf{0}, \end{cases} \tag{5.7}$$

where

$$\mathbf{K} = \{k_{il}\}_{i,l=1}^{n_v},\ k_{il} = \int_\Omega \nabla\vec{\phi}_i : \nabla\vec{\phi}_l, \qquad \mathbf{M} = \{m_{il}\}_{i,l=1}^{n_v},\ m_{il} = \int_\Omega \vec{\phi}_i \cdot \vec{\phi}_l,$$

$$B = \{b_{il}\}_{i=1,\dots,n_p}^{l=1,\dots,n_v},\ b_{il} = -\int_\Omega \psi_i \nabla\cdot\vec{\phi}_l, \qquad \boldsymbol{f} = \{f_i\}_{i=1}^{n_v},\ f_i = \int_\Omega \vec{f}\cdot\vec{\phi}_i.$$

The matrix $\mathbf{K}$ is generally referred to as a *(vector-)stiffness matrix*, and the matrix $\mathbf{M}$ is referred to as a *(vector-)mass matrix*; both the matrices are symmetric positive definite. The matrix B is referred to as the *(negative) divergence matrix*.

In the following, we will employ inf–sup stable Taylor–Hood $\mathbf{Q}_2$–$\mathbf{Q}_1$ finite elements in the spatial dimensions. This approach leads to a block diagonal vector-mass matrix $\mathbf{M}$. Of late, some research has been devoted to non-standard Lagrange finite elements, which lead to mass matrices on the velocity space that have not necessarily a block diagonal structure, see for instance [48, 49, 50].

5.2 Preconditioning Forward Stationary Stokes Equations

In this section we introduce an optimal preconditioner for the forward Stokes equations. This preconditioner (first derived in [164]) consists of a symmetric positive definite 2-by-2 block matrix, within which the approximation for the $(2,2)$-block can be considered as a special case of the commutator argument derived in [92] for the forward stationary Navier–Stokes equations.

As discussed in Section 2.10, an optimal preconditioner for the matrix arising from (5.7) is given by the symmetric positive definite matrix $\mathcal{P}_3$ defined in (2.28), with $\Phi = \mathbf{K}$ and $S = B\mathbf{K}^{-1}B^\top$. As we have repeatedly mentioned so far, we do not apply the inverse operator of each of the two blocks of $\mathcal{P}_3$, but rather find approximations $\widehat{\Phi}$ and $\widehat{S}$ for Φ and S, respectively. As discussed in [44, Section 4.2], an efficient preconditioner is given by the matrix $\widehat{\mathcal{P}}_3$ defined in (2.30), with $\widehat{\Phi}$ being the approximation of $\mathbf{K}$ using a multigrid routine, for example, and $\widehat{S}$ being the (scalar) mass matrix $M_p = [(\psi_i, \psi_l)]$ in the pressure finite element space. Again, rather than solving for M_p, we approximate it by taking its diagonal [164], or by applying Chebyshev semi-iteration [181]. The preconditioner $\widehat{\mathcal{P}}_3$ so defined is robust with respect to the mesh-size h. Indeed, by employing stable finite elements and assuming boundedness of the matrix B, it is possible to prove that the non-zero eigenvalues of the preconditioned Schur complement $M_p^{-1}S$ lie in the interval $[\gamma_l^2, \gamma_u^2]$, see, for instance, [44, Section 4.2] and [182], where γ_l is the inf–sup constant, and γ_u is the boundedness constant. Due to this property, the approximation of the Schur complement S by the pressure mass matrix M_p has been widely used for the fast numerical solution of discrete Stokes problems, see, for instance, [182] and references therein. Although the bounds on the eigenvalues of $M_p^{-1}S$ can be derived analytically, the approximation employed here can be considered a special case of the so called *pressure convection–diffusion preconditioner* [44, pp. 365–370] (first derived in [92]) for the Schur complement arising from the discretization of the *Picard iteration* applied to the forward

Navier–Stokes equations. Here, we adapt the strategy presented in [92] to the case of the Stokes problem, leaving the derivation of the proper pressure convection–diffusion preconditioner to the next chapter.

Consider the diffusion operator $\mathcal{D} = -\nabla^2$ defined on the velocity space as in (5.5), and suppose the analogous operator $\mathcal{D}_p = (-\nabla^2)_p$ on the pressure space is well defined. Suppose also that the commutator

$$\mathcal{E} = \mathcal{D}\nabla - \nabla\mathcal{D}_p \tag{5.8}$$

is small in some sense. Then, discretizing (5.8) with stable finite elements leads to

$$(\mathbf{M}^{-1}\mathbf{K})\mathbf{M}^{-1}B^\top - \mathbf{M}^{-1}B^\top(M_p^{-1}K_p) \approx 0,$$

where $K_p = [(\nabla\psi_i, \nabla\psi_l)]$ is the (scalar) stiffness matrix in the pressure finite element space. Given invertibility of $\mathbf{K}$ and K_p, pre- and post-multiplying by $B\mathbf{K}^{-1}\mathbf{M}$ and $K_p^{-1}M_p$, the previous expression then gives

$$B\mathbf{M}^{-1}B^\top K_p^{-1}M_p \approx B\mathbf{K}^{-1}B^\top.$$

We still have no practical preconditioner due to the matrix $B\mathbf{M}^{-1}B^\top$; however, it can be proved that $K_p \approx B\mathbf{M}^{-1}B^\top$ for problems with enclosed flow [44, pp. 176–177]. Finally, a good approximation of the Schur complement $S = B\mathbf{K}^{-1}B^\top$ is

$$\widehat{S} = K_p K_p^{-1} M_p = M_p \approx S.$$

As we mentioned, the preconditioners we derive in this chapter make use of a generalization of the commutator argument (5.8).

5.3 Block Commutator Argument

In this section, we generalize the pressure convection–diffusion preconditioner, applying the commutator argument in (5.8) to the case where the differential operators involved are to be considered vectorial differential operators, i.e.

$$\mathcal{E}_{\bar{m}} = \mathcal{D}\nabla_{\bar{m}} - \nabla_{\bar{m}}\mathcal{D}_p, \tag{5.9}$$

where

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}^{1,1} & \dots & \mathcal{D}^{1,\bar{m}} \\ \vdots & \ddots & \vdots \\ \mathcal{D}^{\bar{m},1} & \dots & \mathcal{D}^{\bar{m},\bar{m}} \end{bmatrix}, \qquad \mathcal{D}_p = \begin{bmatrix} \mathcal{D}_p^{1,1} & \dots & \mathcal{D}_p^{1,\bar{m}} \\ \vdots & \ddots & \vdots \\ \mathcal{D}_p^{\bar{m},1} & \dots & \mathcal{D}_p^{\bar{m},\bar{m}} \end{bmatrix},$$

for some $\bar{m} \in \mathbb{N}$. Here $\mathcal{D}^{i,l}$ is a differential operator on the velocity space with $\mathcal{D}_p^{i,l}$ the corresponding differential operator on the pressure space, for $i, l = 1, 2, \dots, \bar{m}$, and $\nabla_{\bar{m}} = I_{\bar{m}} \otimes \nabla$, with $I_{\bar{m}} \in \mathbb{R}^{\bar{m}\times\bar{m}}$ the identity matrix. As above, we suppose that each $\mathcal{D}_p^{i,l}$, $i, l = 1, 2, \dots, \bar{m}$, is well defined, and that the commutator $\mathcal{E}_{\bar{m}}$ is small in some sense. Again, after discretizing with stable finite elements we can

rewrite

$$\left(\mathcal{M}^{-1}\mathbf{D}\right)\mathcal{M}^{-1}\vec{B}^{\top} - \mathcal{M}^{-1}\vec{B}^{\top}\left(\mathcal{M}_p^{-1}D_p\right) \approx 0, \tag{5.10}$$

where $\mathcal{M} = I_{\bar{m}} \otimes \mathbf{M}$, $\mathcal{M}_p = I_{\bar{m}} \otimes M_p$, $\vec{B} = I_{\bar{m}} \otimes B$, and

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}^{1,1} & \dots & \mathbf{D}^{1,\bar{m}} \\ \vdots & \ddots & \vdots \\ \mathbf{D}^{\bar{m},1} & \dots & \mathbf{D}^{\bar{m},\bar{m}} \end{bmatrix}, \qquad D_p = \begin{bmatrix} D_p^{1,1} & \dots & D_p^{1,\bar{m}} \\ \vdots & \ddots & \vdots \\ D_p^{\bar{m},1} & \dots & D_p^{\bar{m},\bar{m}} \end{bmatrix},$$

with $\mathbf{M}^{-1}\mathbf{D}^{i,l}$ and $M_p^{-1}D_p^{i,l}$ the corresponding discretizations of $\mathcal{D}^{i,l}$ and $\mathcal{D}_p^{i,l}$, respectively. Assuming invertibility of $\mathbf{D}$, and the corresponding block matrix D_p on the pressure space, pre-multiplying (5.10) by $\vec{B}\mathbf{D}^{-1}\mathcal{M}$, and post-multiplying by $D_p^{-1}\mathcal{M}_p$, gives that

$$\vec{B}\mathcal{M}^{-1}\vec{B}^{\top}D_p^{-1}\mathcal{M}_p \approx \vec{B}\mathbf{D}^{-1}\vec{B}^{\top}.$$

Noting that $\vec{B}\mathcal{M}^{-1}\vec{B}^{\top} = I_{\bar{m}} \otimes (B\mathbf{M}^{-1}B^{\top})$ and recalling that $K_p \approx B\mathbf{M}^{-1}B^{\top}$, we derive the following approximation:

$$\mathcal{K}_p D_p^{-1}\mathcal{M}_p \approx \vec{B}\mathbf{D}^{-1}\vec{B}^{\top}, \tag{5.11}$$

where $\mathcal{K}_p = I_{\bar{m}} \otimes K_p$. In Section 5.5.2 we employ the approach outlined here to devise preconditioners for the discrete optimality conditions of Stokes control problems. Those preconditioners will be then generalized to the Navier–Stokes control problems in the next chapter.

We would like to devote some discussion to the generalized commutator argument discussed above. Although we were not able to prove any spectral property for the approach outlined here, the numerical results reported in Section 5.6 show the robustness of our approach. We believe (although this is only a conjecture) that the generalized commutator argument is likely to be most effective when each block $\mathbf{D}^{i,l}$ of the matrix $\mathbf{D}$ is diagonally dominant. In addition, although we consider only standard Lagrangian finite elements in this work, we believe that the approach outlined here may also be applied to more general vector-valued function spaces. In fact, we believe (this is another conjecture) that the generalized commutator argument may be related to operator preconditioning.

5.4 First-Order Optimality Conditions and Discretization in Time

We now describe the strategy used for obtaining a numerical solution of (5.1)–(5.2) and of (5.3)–(5.4). We introduce adjoint variables $\vec{\zeta}$ and μ and make use of an optimize-then-discretize scheme, stating the first-order optimality conditions. We then discretize the conditions so obtained, for both the stationary and instationary Stokes control problems. For the instationary problem (5.3)–(5.4), we consider employing both backward Euler and Crank–Nicolson schemes in time. While only first-order accurate, backward Euler can be easily generalized to the

setting when the initial condition $\vec{v}_0$ is not solenoidal. On the other hand, Crank–Nicolson is second-order accurate, however if $\vec{v}_0$ is not solenoidal pre-processing is required in order to write the discrete optimality conditions.

We note that the optimality conditions stated below are a special case of the ones derived in Chapter 6 for the corresponding Navier–Stokes control problems. Indeed, the first-order optimality conditions for the Stokes control problems can be derived from those for the Navier–Stokes control problems by neglecting the non-linear term. For this reason, we refer the reader to Section 6.3.2 for the derivation of the optimality conditions for the Stokes control problems.

5.4.1 Stationary Stokes Control

Introducing the adjoint velocity $\vec{\zeta}$ and the adjoint pressure μ, we may consider the Lagrangian associated with (5.1)–(5.2), and write the Karush–Kuhn–Tucker conditions as:

$$\left\{\begin{array}{rll} -\nabla^2\vec{v} + \nabla p = \frac{1}{\beta}\,\vec{\zeta} + \vec{f} & \text{in } \Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \begin{array}{c}\text{state}\\\text{equations}\end{array} \\ -\nabla\cdot\vec{v}(x) = 0 & \text{in } \Omega & \\ \vec{v}(x) = \vec{g}(x) & \text{on } \partial\Omega & \\ -\nabla^2\vec{\zeta} + \nabla\mu \;=\; \vec{v}_d - \vec{v} & \text{in } \Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \begin{array}{c}\text{adjoint}\\\text{equations}\end{array} \\ -\nabla\cdot\vec{\zeta}(x) = 0 & \text{in } \Omega & \\ \vec{\zeta}(x) = \vec{0} & \text{on } \partial\Omega & \end{array}\right. \tag{5.12}$$

where we have substituted the gradient equation $\beta\vec{u} - \vec{\zeta} = 0$ into the state equation.

Problem (5.12) is a coupled system of linear, stationary PDEs. In order to find a numerical solution of (5.12), we discretize those optimality conditions by employing finite elements. Thus, we first need to write the weak formulation of (5.12). Letting V, V_0, and Q be defined as in Section 5.1.1, this reads as:

Find $\vec{v} \in V$, $p \in Q$, $\vec{\zeta} \in V_0$, and $\mu \in Q$ such that

$$\left\{\begin{array}{rl} (\nabla\vec{v}, \nabla\vec{w}) - (p, \nabla\cdot\vec{w}) - \frac{1}{\beta}(\vec{\zeta}, \vec{w}) = (\vec{f}, \vec{w}), \\ -(q, \nabla\cdot\vec{v}) = 0, \\ (\vec{v}, \vec{w}) + (\nabla\vec{\zeta}, \nabla\vec{w}) - (\mu, \nabla\cdot\vec{w}) = (\vec{v}_d, \vec{w}), \\ -(q, \nabla\cdot\vec{\zeta}) = 0, \end{array}\right.$$

for any $\vec{w} \in V_0$ and $q \in Q$.

The problem above is posed on the continuous level, so we need to discretize it in order to obtain a numerical solution of (5.1)–(5.2). Let $\boldsymbol{v} = \{\boldsymbol{v}_i\}_{i=1}^{n_v}$, $\boldsymbol{p} = \{p_i\}_{i=1}^{n_p}$, $\boldsymbol{\zeta} = \{\boldsymbol{\zeta}_i\}_{i=1}^{n_v}$, $\boldsymbol{\mu} = \{\mu_i\}_{i=1}^{n_p}$ be the vectors containing the numerical solutions for $\vec{v}$, p, $\vec{\zeta}$, and μ, respectively, that is, $\vec{v} \approx \sum_{i=1}^{n_v} \boldsymbol{v}_i\vec{\phi}_i$, $p \approx \sum_{i=1}^{n_p} p_i\psi_i$, $\vec{\zeta} \approx$

$\sum_{i=1}^{n_v} \boldsymbol{\zeta}_i \vec{\phi}_i$, $\mu \approx \sum_{i=1}^{n_p} \mu_i \psi_i$. Then, the discrete optimality conditions read as

$$\begin{cases} \mathbf{K}\boldsymbol{v} + B^\top \boldsymbol{p} - \mathbf{M}_\beta \boldsymbol{\zeta} = \boldsymbol{f}, \\ B\boldsymbol{v} = \mathbf{0}, \\ \mathbf{M}\boldsymbol{v} + \mathbf{K}\boldsymbol{\zeta} + B^\top \boldsymbol{\mu} = \mathbf{M}\boldsymbol{v}_d, \\ B\boldsymbol{\zeta} = \mathbf{0}, \end{cases} \tag{5.13}$$

where $\mathbf{M}_\beta = \frac{1}{\beta}\mathbf{M}$, and $\boldsymbol{v}_d$ is the vector corresponding to the discretized desired state $\vec{v}_d$. Note that the right-hand side may also take into account boundary conditions (as done in our implementation).

In matrix form, we rewrite system (5.13) as

$$\underbrace{\begin{bmatrix} \Phi_{\mathrm{S}} & \Psi_{\mathrm{S}}^\top \\ \Psi_{\mathrm{S}} & -\Theta_{\mathrm{S}} \end{bmatrix}}_{\mathcal{A}_{\mathrm{S}}} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \\ \boldsymbol{\mu} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\boldsymbol{v}_d \\ \boldsymbol{f} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{5.14}$$

where

$$\Phi_{\mathrm{S}} = \begin{bmatrix} \mathbf{M} & \mathbf{K} \\ \mathbf{K} & -\mathbf{M}_\beta \end{bmatrix}, \quad \Psi_{\mathrm{S}} = \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix}, \quad \Theta_{\mathrm{S}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.15}$$

It is worth noting that the matrix Φ_{S} represents the discrete optimality conditions for a (vector) Poisson control problem derived in Section 1.3.1. This observation will be used below when deriving our proposed preconditioner.

5.4.2 Instationary Stokes Control

We now state the KKT conditions for the instationary problem (5.3)–(5.4). As before, introducing the adjoint variables $\vec{\zeta}$ and μ, we consider the Lagrangian associated to (5.3)–(5.4). Then, by deriving the KKT conditions and substituting the gradient equation $\beta\vec{u} - \vec{\zeta} = 0$ into the state equation, the solution of (5.3)–(5.4) satisfies:

$$\begin{cases} \frac{\partial \vec{v}}{\partial t} - \nabla^2 \vec{v} + \nabla p = \frac{1}{\beta}\vec{\zeta} + \vec{f} & \text{in } \Omega \times (0, t_f), \\ -\nabla \cdot \vec{v}(x,t) = 0 & \text{in } \Omega \times (0, t_f), \\ \vec{v}(x,t) = \vec{g}(x,t) & \text{on } \partial\Omega \times (0, t_f), \\ \vec{v}(x,0) = \vec{v}_0(x) & \text{in } \Omega, \\ -\frac{\partial \vec{\zeta}}{\partial t} - \nabla^2 \vec{\zeta} + \nabla \mu = \vec{v}_d - \vec{v} & \text{in } \Omega \times (0, t_f), \\ -\nabla \cdot \vec{\zeta}(x,t) = 0 & \text{in } \Omega \times (0, t_f), \\ \vec{\zeta}(x,t) = \vec{0} & \text{on } \partial\Omega \times (0, t_f), \\ \vec{\zeta}(x,t_f) = \vec{0} & \text{in } \Omega. \end{cases} \tag{5.16}$$

Problem (5.16) is a coupled system of linear, instationary PDEs. Working as in the previous section, we first write the weak formulation of (5.16), then discretize it.

We first introduce the following spaces. Let

$$\begin{aligned}\bar{V} &:= \{\vec{v} \in L^2(0,t_f;\mathcal{H}^1(\Omega)^d) \mid \tfrac{\partial \vec{v}}{\partial t}(\cdot,t) \in L^2(0,t_f;\mathcal{H}^{-1}(\Omega)^d) \text{ for a.e. } t \in (0,t_f),\\ &\qquad \vec{v} = \vec{g} \text{ on } \partial\Omega,\ \vec{v}(x,0) = \vec{v}_0(x)\},\\ \bar{Q} &:= L^2(0,t_f;L^2(\Omega)),\end{aligned}$$

and $\bar{V}_0$ be the corresponding space for the adjoint velocity. See [175, pp. 315–321] and [86, pp. 88–95], for instance, for Navier–Stokes control in the case $d = 2$. Then, the weak formulation of (5.16) reads as:

Find $\vec{v} \in \bar{V}$, $p \in \bar{Q}$, $\vec{\zeta} \in \bar{V}_0$, and $\mu \in \bar{Q}$ such that

$$\begin{cases} (\frac{\partial}{\partial t}\vec{v},\vec{w}) + (\nabla\vec{v},\nabla\vec{w}) - (p,\nabla\cdot\vec{w}) - \frac{1}{\beta}(\vec{\zeta},\vec{w}) = (\vec{f},\vec{w}), \\ \qquad\qquad -(q,\nabla\cdot\vec{v}\,) = 0, \\ -(\frac{\partial}{\partial t}\vec{\zeta},\vec{w}) + (\nabla\vec{\zeta},\nabla\vec{w}) - (\mu,\nabla\cdot\vec{w}) + (\vec{v},\vec{w}) = (\vec{v}_d,\vec{w}), \\ \qquad\qquad -(q,\nabla\cdot\vec{\zeta}\,) = 0, \end{cases} \tag{5.17}$$

for any $\vec{w} \in V_0$ and $q \in Q$.

Problem (5.17) is a coupled system of instationary diffusion equations and divergence-free conditions. Due to this structure, we present two discretized versions of (5.17), one making use of backward Euler time-stepping, the other employing the Crank–Nicolson scheme. For the sake of exposition, we introduce the matrices $I_{\bar{n},1}$, $I_{\bar{n},2}$, $I_{\bar{n},3}$, $I_{\bar{n},4} \in \mathbb{R}^{\bar{n}\times\bar{n}}$, with

$$\begin{aligned} I_{\bar{n},1} &= \mathtt{diag}(1,\ldots,1,0), \\ I_{\bar{n},2} &= \mathtt{diag}(0,1,\ldots,1), \\ I_{\bar{n},4} &= I_{\bar{n}} + I_{\bar{n},3}, \end{aligned} \qquad I_{\bar{n},3} = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ & & & 0 \end{bmatrix}. \tag{5.18}$$

Here, `diag` denotes a diagonal matrix with the diagonal entries stated.

Backward Euler for Instationary Stokes Control

In this section we introduce the backward Euler scheme for approximating (5.17), and then derive the resulting linear system. We discretize the interval $(0,t_f)$ into n_t subintervals of length $\tau = \frac{t_f}{n_t}$, denoting the grid points as $t_n = n\tau$, for $n = 0,1,\ldots,n_t$. We approximate all the functions on this time grid, excluding the initial and final time points for the state and adjoint pressure, respectively. Specifically, our approximations of the solutions are given by $\boldsymbol{v}_n \approx \vec{v}(x,t_n)$, $\boldsymbol{\zeta}_n \approx \vec{\zeta}(x,t_n)$, for $n = 0,1,\ldots,n_t$, and $\boldsymbol{p}_{n+1} \approx p(x,t_{n+1})$, $\boldsymbol{\mu}_n \approx \mu(x,t_n)$, for $n = 0,1,\ldots,n_t-1$, for all $x \in \Omega$. We also introduce the following finite element matrices:

$$\mathbf{L}^{\mathrm{BE}} = \tau\mathbf{K} + \mathbf{M}, \qquad \bar{\mathbf{M}}^{\mathrm{BE}} = \tau\mathbf{M}, \qquad \bar{\mathbf{M}}^{\mathrm{BE}}_{\beta} = \frac{\tau}{\beta}\mathbf{M}, \qquad \bar{B} = \tau B,$$

We then write the following discretization of (5.17):

$$\begin{cases} \bar{\mathbf{M}}^{\mathrm{BE}}\boldsymbol{v}_n + \mathbf{L}^{\mathrm{BE}}\boldsymbol{\zeta}_n - \mathbf{M}\boldsymbol{\zeta}_{n+1} + \bar{B}^\top\boldsymbol{\mu}_n = \bar{\mathbf{M}}^{\mathrm{BE}}\boldsymbol{v}_d^n, \\ -\mathbf{M}\boldsymbol{v}_n + \mathbf{L}^{\mathrm{BE}}\boldsymbol{v}_{n+1} + \bar{B}^\top\boldsymbol{p}_{n+1} - \bar{\mathbf{M}}_\beta^{\mathrm{BE}}\boldsymbol{\zeta}_{n+1} = \tau\boldsymbol{f}^{n+1}, \\ B\boldsymbol{v}_{n+1} = \mathbf{0}, \\ B\boldsymbol{\zeta}_n = \mathbf{0}, \end{cases} \tag{5.19}$$

for $n = 0, 1, ..., n_t - 1$, with $\boldsymbol{v}_0 = \boldsymbol{v}^0$, $\boldsymbol{\zeta}_{n_t} = \mathbf{0}$, where $\boldsymbol{v}^0$ is the discretization of the initial condition for $\vec{v}$, and

$$\boldsymbol{f}^{n+1} = \{f_i^{n+1}\}_{i=1}^{n_v} \qquad f_i^{n+1} = (\vec{f}(x, t_{n+1}), \vec{\phi}_i).$$

We immediately realize that the system described in (5.19) is not symmetric, due the initial and final time conditions $\boldsymbol{v}_0 = \boldsymbol{v}^0$ and $\boldsymbol{\zeta}_{n_t} = \mathbf{0}$. However, it can be made symmetric by employing the following projections onto the space of divergence-free functions (*solenoidal projection*), as done in [84] for the instationary Navier–Stokes control problem, for instance. Given a vector $\bar{\boldsymbol{b}}$, its solenoidal projection is defined as $\boldsymbol{b}$, with

$$\begin{cases} \mathbf{L}^{\mathrm{BE}}\boldsymbol{b} + \bar{B}^\top\bar{\boldsymbol{p}} = \mathbf{L}^{\mathrm{BE}}\bar{\boldsymbol{b}}, \\ B\,\boldsymbol{b} = \mathbf{0}. \end{cases} \tag{5.20}$$

As the vector $\boldsymbol{v}^0$ is clearly divergence-free, the condition $\boldsymbol{v}_0 = \boldsymbol{v}^0$ is equivalent to

$$\begin{cases} \mathbf{L}^{\mathrm{BE}}\boldsymbol{v}_0 + \bar{B}^\top\boldsymbol{p}_0 = \mathbf{L}^{\mathrm{BE}}\boldsymbol{v}^0, \\ B\boldsymbol{v}_0 = \mathbf{0}. \end{cases} \tag{5.21}$$

Note that, if $\vec{v}_0$ is not incompressible, the previous solenoidal projection gives the first backward Euler step of our discretization. Analogously, the condition $\boldsymbol{\zeta}_{n_t} = \mathbf{0}$ is equivalent to

$$\begin{cases} \mathbf{L}^{\mathrm{BE}}\boldsymbol{\zeta}_{n_t} + \bar{B}^\top\boldsymbol{\mu}_{n_t} = \mathbf{0}, \\ B\boldsymbol{\zeta}_{n_t} = \mathbf{0}. \end{cases}$$

By imposing the previous projections and multiplying the incompressibility conditions by τ, the linear system of (5.19) can be rewritten as

$$\underbrace{\begin{bmatrix} \Phi_{\mathrm{BE}} & (\Psi_{\mathrm{BE}})^\top \\ \Psi_{\mathrm{BE}} & -\Theta_{\mathrm{BE}} \end{bmatrix}}_{\mathcal{A}_{\mathrm{BE}}} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \\ \boldsymbol{\mu} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{bmatrix}, \tag{5.22}$$

where the right-hand side accounts for the the initial and final-time conditions on $\vec{v}$ and $\vec{\zeta}$, as well as information from the desired state $\vec{v}_d$ and the force function $\vec{f}$. Further,

$$\Phi_{\mathrm{BE}} = \begin{bmatrix} \mathcal{M}^{\mathrm{BE}} & (\mathcal{L}^{\mathrm{BE}})^\top \\ \mathcal{L}^{\mathrm{BE}} & -\mathcal{M}_\beta^{\mathrm{BE}} \end{bmatrix}, \quad \Psi_{\mathrm{BE}} = \begin{bmatrix} \mathcal{B}^{\mathrm{BE}} & 0 \\ 0 & \mathcal{B}^{\mathrm{BE}} \end{bmatrix}, \quad \Theta_{\mathrm{BE}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{5.23}$$

with $\mathcal{M}^{\mathrm{BE}} = I_{n_t+1,1} \otimes \bar{\mathbf{M}}^{\mathrm{BE}}$, $\mathcal{M}_\beta^{\mathrm{BE}} = I_{n_t+1,2} \otimes \bar{\mathbf{M}}_\beta^{\mathrm{BE}}$, $\mathcal{B}^{\mathrm{BE}} = I_{n_t+1} \otimes \bar{B}$, and

$$\mathcal{L}^{\mathrm{BE}} = \begin{bmatrix} \mathbf{L}^{\mathrm{BE}} & & & \\ -\mathbf{M} & \mathbf{L}^{\mathrm{BE}} & & \\ & \ddots & \ddots & \\ & & -\mathbf{M} & \mathbf{L}^{\mathrm{BE}} \end{bmatrix}.$$

Crank–Nicolson for Instationary Stokes Control

In this section we present the linear system arising upon employing Crank–Nicolson in time when solving (5.17). Again discretizing the interval $(0, t_f)$ into n_t subintervals of length $\tau = \frac{t_f}{n_t}$, we approximate $\vec{v}$ and $\vec{\zeta}$ at the time points $t_n = n\tau$, $n = 0, 1, \ldots, n_t$, and use a staggered grid for p and μ, as in [13]. Specifically, our approximations of the solutions are given by $\boldsymbol{v}_n \approx \vec{v}(x, t_n)$, $\boldsymbol{\zeta}_n \approx \vec{\zeta}(x, t_n)$, for $n = 0, 1, \ldots, n_t$, and $\boldsymbol{p}_{n+\frac{1}{2}} \approx p(x, t_n + \frac{1}{2}\tau)$, $\boldsymbol{\mu}_{n+\frac{1}{2}} \approx \mu(x, t_n + \frac{1}{2}\tau)$, for $n = 0, 1, \ldots, n_t - 1$, for all $x \in \Omega$. Let us introduce the following finite element matrices:

$$\mathbf{L}^{\pm} = \tfrac{\tau}{2}\mathbf{K} \pm \mathbf{M}, \qquad \bar{\mathbf{M}}^{\mathrm{CN}} = \tfrac{\tau}{2}\mathbf{M}, \qquad \bar{\mathbf{M}}_\beta^{\mathrm{CN}} = \tfrac{\tau}{2\beta}\mathbf{M}.$$

Then, the discrete optimality conditions read as follows:

$$\begin{cases} \bar{\mathbf{M}}^{\mathrm{CN}}(\boldsymbol{v}_n + \boldsymbol{v}_{n+1}) + \mathbf{L}^{+}\boldsymbol{\zeta}_n + \mathbf{L}^{-}\boldsymbol{\zeta}_{n+1} + \bar{B}^{\top}\boldsymbol{\mu}_{n+\frac{1}{2}} = \bar{\mathbf{M}}^{\mathrm{CN}}(\boldsymbol{v}_d^n + \boldsymbol{v}_d^{n+1}), \\ \mathbf{L}^{-}\boldsymbol{v}_n + \mathbf{L}^{+}\boldsymbol{v}_{n+1} + \bar{B}^{\top}\boldsymbol{p}_{n+\frac{1}{2}} - \bar{\mathbf{M}}_\beta^{\mathrm{CN}}(\boldsymbol{\zeta}_n + \boldsymbol{\zeta}_{n+1}) = \tfrac{\tau}{2}(\boldsymbol{f}^n + \boldsymbol{f}^{n+1}), \\ B\boldsymbol{v}_{n+1} = \mathbf{0}, \\ B\boldsymbol{\zeta}_n = \mathbf{0}, \end{cases}$$

for $n = 0, 1, ..., n_t - 1$, with $\boldsymbol{v}_0 = \boldsymbol{v}^0$, $\boldsymbol{\zeta}_{n_t} = \mathbf{0}$, and $\boldsymbol{f}^n$ defined as for backward Euler.

In matrix form, after multipling the incompressibility constraints by τ, we write

$$\begin{bmatrix} \bar{\mathcal{M}}^{\mathrm{CN}} & \bar{\mathcal{L}}_1^{\mathrm{CN}} & (\bar{\mathcal{B}}_2^{\mathrm{CN}})^{\top} & 0 \\ \bar{\mathcal{L}}_2^{\mathrm{CN}} & -\bar{\mathcal{M}}_\beta^{\mathrm{CN}} & 0 & (\bar{\mathcal{B}}_1^{\mathrm{CN}})^{\top} \\ \bar{\mathcal{B}}_1^{\mathrm{CN}} & 0 & 0 & 0 \\ 0 & \bar{\mathcal{B}}_2^{\mathrm{CN}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{v}} \\ \bar{\boldsymbol{\zeta}} \\ \bar{\boldsymbol{\mu}} \\ \bar{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{b}}_1 \\ \bar{\mathbf{b}}_2 \\ \bar{\mathbf{b}}_3 \\ \bar{\mathbf{b}}_4 \end{bmatrix}, \tag{5.24}$$

where the right-hand side accounts for the initial and final time conditions on $\vec{v}$ and $\vec{\zeta}$, and information from $\vec{v}_d$ and $\vec{f}$. The blocks in the previous matrix are

given by

$$\bar{\mathcal{L}}_1^{\mathrm{CN}} = \begin{bmatrix} \mathbf{L}^+ & \mathbf{L}^- & & \\ & \ddots & \ddots & \\ & & \mathbf{L}^+ & \mathbf{L}^- \\ & & & \mathbf{M} \end{bmatrix}, \qquad \bar{\mathcal{L}}_2^{\mathrm{CN}} = \begin{bmatrix} \mathbf{M} & & & \\ \mathbf{L}^- & \mathbf{L}^+ & & \\ & \ddots & \ddots & \\ & & \mathbf{L}^- & \mathbf{L}^+ \end{bmatrix},$$

$$\bar{\mathcal{B}}_1^{\mathrm{CN}} = \begin{bmatrix} 0 & \bar{B} & & \\ & & \ddots & \\ & & & \bar{B} \end{bmatrix}, \qquad \bar{\mathcal{B}}_2^{\mathrm{CN}} = \begin{bmatrix} \bar{B} & & & \\ & \ddots & & \\ & & \bar{B} & 0 \end{bmatrix},$$

and $\bar{\mathcal{M}}^{\mathrm{CN}} = (I_{n_t+1,1} + I_{n_t+1,3}) \otimes \bar{\mathbf{M}}^{\mathrm{CN}}$, $\bar{\mathcal{M}}_\beta^{\mathrm{CN}} = (I_{n_t+1,2} + I_{n_t+1,3}^\top) \otimes \bar{\mathbf{M}}_\beta^{\mathrm{CN}}$.

The system (5.24) is clearly not symmetric; however, we work as in [100] and in Chapter 3 in order to transform the linear system above and make it symmetric. In fact, eliminating the initial and final-time conditions on $\vec{v}$ and $\vec{\zeta}$, we can rewrite

$$\begin{bmatrix} \widetilde{\mathcal{M}}^{\mathrm{CN}} & (\widetilde{\mathcal{L}}^{\mathrm{CN}})^\top & (\widetilde{\mathcal{B}}^{\mathrm{CN}})^\top & 0 \\ \widetilde{\mathcal{L}}^{\mathrm{CN}} & -\widetilde{\mathcal{M}}_\beta^{\mathrm{CN}} & 0 & (\widetilde{\mathcal{B}}^{\mathrm{CN}})^\top \\ \widetilde{\mathcal{B}}^{\mathrm{CN}} & 0 & 0 & 0 \\ 0 & \widetilde{\mathcal{B}}^{\mathrm{CN}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \\ \boldsymbol{\mu} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{bmatrix},$$

with $\boldsymbol{v}$, $\boldsymbol{\zeta}$, $\boldsymbol{\mu}$, $\boldsymbol{p}$ as well as the right-hand side modified accordingly. The matrices $\widetilde{\mathcal{M}}^{\mathrm{CN}} = I_{n_t,4}^\top \otimes \bar{\mathbf{M}}^{\mathrm{CN}}$, $\widetilde{\mathcal{M}}_\beta^{\mathrm{CN}} = I_{n_t,4} \otimes \bar{\mathbf{M}}_\beta^{\mathrm{CN}}$, $\widetilde{\mathcal{B}}^{\mathrm{CN}} = I_{n_t} \otimes \bar{B}$, and

$$\widetilde{\mathcal{L}}^{\mathrm{CN}} = \begin{bmatrix} \mathbf{L}^+ & & & \\ \mathbf{L}^- & \mathbf{L}^+ & & \\ & \ddots & \ddots & \\ & & \mathbf{L}^- & \mathbf{L}^+ \end{bmatrix}.$$

Using `blkdiag` to define a block diagonal matrix, we consider the linear transformation

$$T = \mathtt{blkdiag}(T_1, T_2, T_3, T_4), \tag{5.25}$$

where

$$\begin{aligned} T_1 &= I_{n_t,4} \otimes I_{n_v}, \quad & T_2 &= T_1^\top = I_{n_t,4}^\top \otimes I_{n_v}, \\ T_3 &= I_{n_t,4}^\top \otimes I_{n_p}, \quad & T_4 &= T_3^\top = I_{n_t,4} \otimes I_{n_p}. \end{aligned} \tag{5.26}$$

Then, we may equivalently consider the following linear system:

$$\underbrace{\begin{bmatrix} \Phi_{\mathrm{CN}} & (\Psi_{\mathrm{CN}})^\top \\ \Psi_{\mathrm{CN}} & -\Theta_{\mathrm{CN}} \end{bmatrix}}_{\mathcal{A}_{\mathrm{CN}}} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\zeta} \\ \boldsymbol{\mu} \\ \boldsymbol{p} \end{bmatrix} = T \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \\ \mathbf{b}_4 \end{bmatrix}. \tag{5.27}$$

Here, the matrix blocks are given by

$$\Phi_{\mathrm{CN}} = \begin{bmatrix} \mathcal{M}^{\mathrm{CN}} & (\mathcal{L}^{\mathrm{CN}})^\top \\ \mathcal{L}^{\mathrm{CN}} & -\mathcal{M}_\beta^{\mathrm{CN}} \end{bmatrix}, \quad \Psi_{\mathrm{CN}} = \begin{bmatrix} \mathcal{B}_1^{\mathrm{CN}} & 0 \\ 0 & \mathcal{B}_2^{\mathrm{CN}} \end{bmatrix}, \quad \Theta_{\mathrm{CN}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{5.28}$$

with

$$\begin{aligned} \mathcal{M}^{\mathrm{CN}} &= T_1 \widetilde{\mathcal{M}}^{\mathrm{CN}} = \left(I_{n_t,4}\ I_{n_t,4}^\top\right) \otimes \bar{\mathbf{M}}^{\mathrm{CN}}, & \mathcal{B}_1^{\mathrm{CN}} &= T_3 \widetilde{\mathcal{B}}^{\mathrm{CN}} = I_{n_t,4}^\top \otimes \bar{B}, \\ \mathcal{M}_\beta^{\mathrm{CN}} &= T_2 \widetilde{\mathcal{M}}_\beta^{\mathrm{CN}} = \left(I_{n_t,4}^\top\ I_{n_t,4}\right) \otimes \bar{\mathbf{M}}_\beta^{\mathrm{CN}}, & \mathcal{B}_2^{\mathrm{CN}} &= T_4 \widetilde{\mathcal{B}}^{\mathrm{CN}} = I_{n_t,4} \otimes \bar{B}, \end{aligned}$$

and the matrix

$$\mathcal{L}^{\mathrm{CN}} = T_1 \widetilde{\mathcal{L}}^{\mathrm{CN}}. \tag{5.29}$$

We have thus transformed the system (5.27) into a symmetric one. We observe that the transformations $T_i,\ i = 1, 2, 3, 4$, as well as their inverse operations are easy and computationally cheap to apply, as they require only a sequence of block updates.

It is worth noting that the matrix Φ_{CN} represents the symmetrized system for a (vector) heat control problem derived in Section 3.2.2. In particular, as done in Section 3.2.2 we may rewrite

$$\mathcal{M}^{\mathrm{CN}} = T_1 \mathcal{M}_D^{\mathrm{CN}} T_1^\top, \qquad \mathcal{M}_\beta^{\mathrm{CN}} = T_2 \mathcal{M}_{D,\beta}^{\mathrm{CN}} T_2^\top, \tag{5.30}$$

where

$$\mathcal{M}_D^{\mathrm{CN}} = I_{n_t} \otimes \bar{\mathbf{M}}^{\mathrm{CN}}, \qquad \mathcal{M}_{D,\beta}^{\mathrm{CN}} = I_{n_t} \otimes \bar{\mathbf{M}}_\beta^{\mathrm{CN}}. \tag{5.31}$$

We may therefore work efficiently with $\mathcal{M}^{\mathrm{CN}}$ and $\mathcal{M}_\beta^{\mathrm{CN}}$, using T_1, T_2, $\mathcal{M}_D^{\mathrm{CN}}$, $\mathcal{M}_{D,\beta}^{\mathrm{CN}}$. Further, since both $\mathcal{M}_D^{\mathrm{CN}}$ and $\mathcal{M}_{D,\beta}^{\mathrm{CN}}$ are symmetric positive definite, the same holds for $\mathcal{M}^{\mathrm{CN}}$ and $\mathcal{M}_\beta^{\mathrm{CN}}$.

We point out that it is not straightforward to generalize the Crank–Nicolson discretization to the case where $\vec{v}_0$ is not incompressible. In fact, in this case we must also solve an appropriate solenoidal projection; however, the projection cannot be solved along with the other equations, as our approach requires the elimination of the initial and final conditions on $\vec{v}$ and $\vec{\zeta}$. Therefore, before applying our solver we must solve the projection, which is of the form (5.7).

5.5 Preconditioning Approach

As the discretizations (5.14), (5.22), and (5.27) of the optimality conditions for the problems under examination lead to matrices of the structure (2.3), we now devise preconditioners for each system by making use of saddle-point theory. Although the matrices considered are symmetric, it is worth noting that the $(1,1)$-block of each of them is not positive definite, but (clearly) indefinite. For this reason, we cannot employ the block diagonal preconditioner $\mathcal{P}_3$ defined in (2.28), as this is required to be positive definite, but rather employ the approximation $\widehat{\mathcal{P}}_1$ defined in (2.30) of the block triangular preconditioner $\mathcal{P}_1$. As we have to (approximately) apply the inverse of the corresponding $(1,1)$-block of each matrix analysed, we again employ an approximation $\widehat{\mathcal{P}}_1$ defined in (2.30). In the following, subscripts

refer to the corresponding matrix we are considering.

5.5.1 Approximation of the $(1,1)$-Block

We now describe suitable approximations of the inverses of the $(1,1)$-blocks for the systems (5.14), (5.22), and (5.27). As each of these matrices is symmetric, we employ two alternative strategies to approximate the inverse of the $(1,1)$-block, accelerated with the preconditioners described below. The first strategy employs a fixed number of steps of the inexact preconditioned Uzawa iteration presented in Section 2.2.1 [43]; on the other hand, the second strategy makes use of a fixed number of GMRES iterations [157].

Stationary Stokes Control

Consider the $(1,1)$-block Φ_{S} defined in (5.15). As we mentioned above, this matrix can be considered as the discretization of the optimality conditions for a Poisson control problem in a vectorial sense. Using saddle-point theory, a suitable preconditioner is given by

$$\mathcal{P}_{\Phi,\mathrm{S}} = \begin{bmatrix} \mathbf{M} & 0 \\ \mathbf{K} & -\mathbf{S}_{\Phi,\mathrm{S}} \end{bmatrix},$$

with $\mathbf{S}_{\Phi,\mathrm{S}} = \mathbf{M}_\beta + \mathbf{K}\mathbf{M}^{-1}\mathbf{K}$ the corresponding Schur complement. As described in [140] and as we discussed in Section 2.11.1, an optimal preconditioner for $\mathcal{P}_{\Phi,\mathrm{S}}$ is given by

$$\widehat{\mathcal{P}}_{\Phi,\mathrm{S}} = \begin{bmatrix} \mathbf{M}_c & 0 \\ \mathbf{K} & -\widehat{\mathbf{S}}_{\Phi,\mathrm{S}} \end{bmatrix}.$$

Here, $\mathbf{M}_c$ represents a fixed number of steps of the Chebyshev semi-iterative method [63, 64, 181], and

$$\widehat{\mathbf{S}}_{\Phi,\mathrm{S}} = \big(\mathbf{K} + \mathbf{M}_{\sqrt{\beta}}\big)\mathbf{M}^{-1}\big(\mathbf{K} + \mathbf{M}_{\sqrt{\beta}}\big),$$

with $\mathbf{M}_{\sqrt{\beta}} = \frac{1}{\sqrt{\beta}}\mathbf{M}$ and the block $\mathbf{K} + \mathbf{M}_{\sqrt{\beta}}$ approximated by the action of a multigrid routine, for example. Following the work in [140] and Section 2.11.1, it can be proved that $\lambda(\widehat{\mathbf{S}}_{\Phi,\mathrm{S}}^{-1}\,\mathbf{S}_{\Phi,\mathrm{S}}) \in \big[\frac{1}{2}, 1\big]$.

Instationary Stokes Control with Backward Euler

We now derive a preconditioner for the matrix Φ_{BE} defined in (5.23). It is worth noting the similarities between the matrix Φ_{BE} and the saddle-point system obtained in Section 3.2.1 after discretizing the first-order optimality conditions for heat control with backward Euler in time. In fact, as for the stationary Stokes control problem, the matrix Φ_{BE} can be considered as the discrete optimality conditions of a vector heat control problem. For this reason, we will employ the block triangular preconditioner $\widehat{\mathcal{P}}_1$ defined in (2.30), whose diagonal blocks are the approximations of the main blocks for the heat control problem derived in [139] and discussed in Section 3.3. Specifically, as the matrix $\mathcal{M}^{\mathrm{BE}}$ is not

invertible, we seek a preconditioner of the form:

$$\widetilde{\mathcal{P}}_{\Phi,\mathrm{BE}} = \begin{bmatrix} \widetilde{\mathcal{M}}^{\mathrm{BE}} & 0 \\ \mathcal{L}^{\mathrm{BE}} & -\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}} \end{bmatrix},$$

with $\widetilde{\mathcal{M}}^{\mathrm{BE}}$ an invertible approximation of $\mathcal{M}^{\mathrm{BE}}$, and the perturbed Schur complement $\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}} = \mathcal{M}^{\mathrm{BE}}_{\beta} + \mathcal{L}^{\mathrm{BE}}\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1}(\mathcal{L}^{\mathrm{BE}})^{\top}$. Following the work in [139] and the discussion in Section 3.3, a suitable approximation of $\mathcal{M}^{\mathrm{BE}}$ is given by

$$\widetilde{\mathcal{M}}^{\mathrm{BE}} = \mathtt{blkdiag}(\bar{\mathbf{M}}^{\mathrm{BE}}, \ldots, \bar{\mathbf{M}}^{\mathrm{BE}}, \xi\bar{\mathbf{M}}^{\mathrm{BE}}),$$

with $0 < \xi \ll 1$. In addition, a good approximation for $\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}$ is the matrix

$$\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}} = \big(\mathcal{L}^{\mathrm{BE}} + \mathcal{M}^{\mathrm{BE}}_{\sqrt{\beta}}\big)\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1}\big(\mathcal{L}^{\mathrm{BE}} + \mathcal{M}^{\mathrm{BE}}_{\sqrt{\beta}}\big),$$

with

$$\mathcal{M}^{\mathrm{BE}}_{\sqrt{\beta}} = \frac{\tau}{\sqrt{\beta}}\mathtt{blkdiag}(0, \mathbf{M}, \ldots, \mathbf{M}, \sqrt{\xi}\,\mathbf{M}).$$

It is worth noting that the approximation $\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}}$ of $\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}$ is optimal, as it is possible to prove that $\lambda(\widehat{\mathcal{S}}^{-1}_{\Phi,\mathrm{BE}}\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}) \in [\frac{1}{2}, 1]$, see [139]. As above, we do not apply the inverses of $\mathcal{L}^{\mathrm{BE}} + \mathcal{M}^{\mathrm{BE}}_{\sqrt{\beta}}$ and its transpose exactly, but rather we apply block substitution, with each block on the diagonal approximated by the action of a multigrid process, for instance. Thus, a suitable approximation of the matrix $\widetilde{\mathcal{P}}_{\Phi,\mathrm{BE}}$ is given by

$$\widehat{\mathcal{P}}_{\Phi,\mathrm{BE}} = \begin{bmatrix} \widehat{\mathcal{M}}^{\mathrm{BE}}_{c} & 0 \\ \mathcal{L}^{\mathrm{BE}} & -\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}} \end{bmatrix}, \qquad \widehat{\mathcal{M}}^{\mathrm{BE}}_{c} = \tau\,\mathtt{blkdiag}(\mathbf{M}_c, \ldots, \mathbf{M}_c, \xi\mathbf{M}_c).$$

Instationary Stokes Control with Crank–Nicolson

We focus now on devising a preconditioner for the matrix Φ_{CN} defined in (5.28), arising from a Crank–Nicolson discretization. Similarly to the backward Euler case, this matrix can be considered as the (symmetrized) discretization of the optimality conditions for a vector heat control problem discretized using Crank–Nicolson in time. Again, we seek to use the block triangular matrix

$$\mathcal{P}_{\Phi,\mathrm{CN}} = \begin{bmatrix} \mathcal{M}^{\mathrm{CN}} & 0 \\ \mathcal{L}^{\mathrm{CN}} & -S_{\Phi,\mathrm{CN}} \end{bmatrix}$$

as a preconditioner, where $S_{\Phi,\mathrm{CN}} = \mathcal{M}^{\mathrm{CN}}_{\beta} + \mathcal{L}^{\mathrm{CN}}(\mathcal{M}^{\mathrm{CN}})^{-1}(\mathcal{L}^{\mathrm{CN}})^{\top}$. In order to find an approximation of $\mathcal{P}_{\Phi,\mathrm{CN}}$, we follow the work in [100] and the discussion in Section 3.3.

From (5.30)–(5.31), $\mathcal{M}^{\mathrm{CN}}$ can be written as $\mathcal{M}^{\mathrm{CN}} = T_1\mathcal{M}^{\mathrm{CN}}_{D}T_1^{\top}$, with $\mathcal{M}^{\mathrm{CN}}_{D}$ a block diagonal matrix with each diagonal block a multiple of $\mathbf{M}$. Therefore, a good approximation of $\mathcal{M}^{\mathrm{CN}}$ is given by

$$\widehat{\mathcal{M}}^{\mathrm{CN}} = T_1\widehat{\mathcal{M}}^{\mathrm{CN}}_{D}T_1^{\top},$$

with $\widehat{\mathcal{M}}_D^{\mathrm{CN}} = \frac{\tau}{2} I_{n_t} \otimes \mathbf{M}_c$.

Following the work in Section 3.3.2, we can use (5.30) together with (5.29) to rewrite

$$S_{\Phi,\mathrm{CN}} = T_2 \underbrace{\left[\mathcal{M}_{D,\beta}^{\mathrm{CN}} + \widetilde{\mathcal{L}}^{\mathrm{CN}} \left(\mathcal{M}^{\mathrm{CN}}\right)^{-1} \left(\widetilde{\mathcal{L}}^{\mathrm{CN}}\right)^{\top}\right]}_{S_{\Phi,\mathrm{CN}}^{\mathrm{int}}} T_1, \tag{5.32}$$

recalling that $T_1 = T_2^{\top}$. Then, we apply the matching strategy to $S_{\Phi,\mathrm{CN}}^{\mathrm{int}}$ as done in Section 3.3.2, and find that a suitable approximation of $S_{\Phi,\mathrm{CN}}^{\mathrm{int}}$ is given by

$$\widehat{S}_{\Phi,\mathrm{CN}}^{\mathrm{int}} = \left(\widetilde{\mathcal{L}}^{\mathrm{CN}} + \widehat{\mathcal{M}}\right) T_2^{-1} \left(\mathcal{M}_D^{\mathrm{CN}}\right)^{-1} T_1^{-1} \left(\widetilde{\mathcal{L}}^{\mathrm{CN}} + \widehat{\mathcal{M}}\right)^{\top},$$

with $\widehat{\mathcal{M}} = \frac{\tau}{2\sqrt{\beta}} I_{n_t,4}^{\top} \otimes \mathbf{M}$. Finally, substituting $\widehat{S}_{\Phi,\mathrm{CN}}^{\mathrm{int}}$ into (5.32) and observing that T_1 commutes with both $\widetilde{\mathcal{L}}^{\mathrm{CN}}$ and $\widehat{\mathcal{M}}$, we obtain that our approximation of $S_{\Phi,\mathrm{CN}}$ is given by

$$\widehat{S}_{\Phi,\mathrm{CN}} = \left(\widetilde{\mathcal{L}}^{\mathrm{CN}} + \widehat{\mathcal{M}}\right) \left(\mathcal{M}_D^{\mathrm{CN}}\right)^{-1} \left(\widetilde{\mathcal{L}}^{\mathrm{CN}} + \widehat{\mathcal{M}}\right)^{\top}.$$

As for backward Euler, we approximate the blocks $\widetilde{\mathcal{L}}^{\mathrm{CN}} + \widehat{\mathcal{M}}$ and its transpose using block substitution, with the action of a multigrid process used to apply the inverse of each block diagonal entry inexactly.

It is worth noting that the preconditioner derived here reduces to that derived in [100] for the heat control problem, which was proved to be optimal, and such that the spectrum of the preconditioned Schur complement is contained in $[\frac{1}{2}, 1]$, see Theorem 5 in Section 3.3.2.

5.5.2 Approximation of Schur Complement

We now derive efficient approximations for each Schur complement of the systems (5.14), (5.22), and (5.27). Since the $(2,1)$- and the $(1,2)$-blocks of these systems can be considered as a (negative) *vector-divergence matrix* and its transpose, we make use of the block commutator argument presented in Section 5.3.

Stationary Stokes Control

Let us consider the Schur complement $S_{\mathcal{A},\mathrm{S}} = \Psi_{\mathrm{S}}(\Phi_{\mathrm{S}})^{-1}\Psi_{\mathrm{S}}^{\top}$ of the system (5.14), with Φ_{S} and Ψ_{S} defined as in (5.15). We apply the commutator argument to $\mathcal{E}_{\bar{m}}$ as defined in (5.9) with $\bar{m} = 2$, with the differential operator on the velocity space defined as

$$\mathcal{D} = \begin{bmatrix} \mathrm{Id} & -\nabla^2 \\ -\nabla^2 & -\frac{1}{\beta}\mathrm{Id} \end{bmatrix},$$

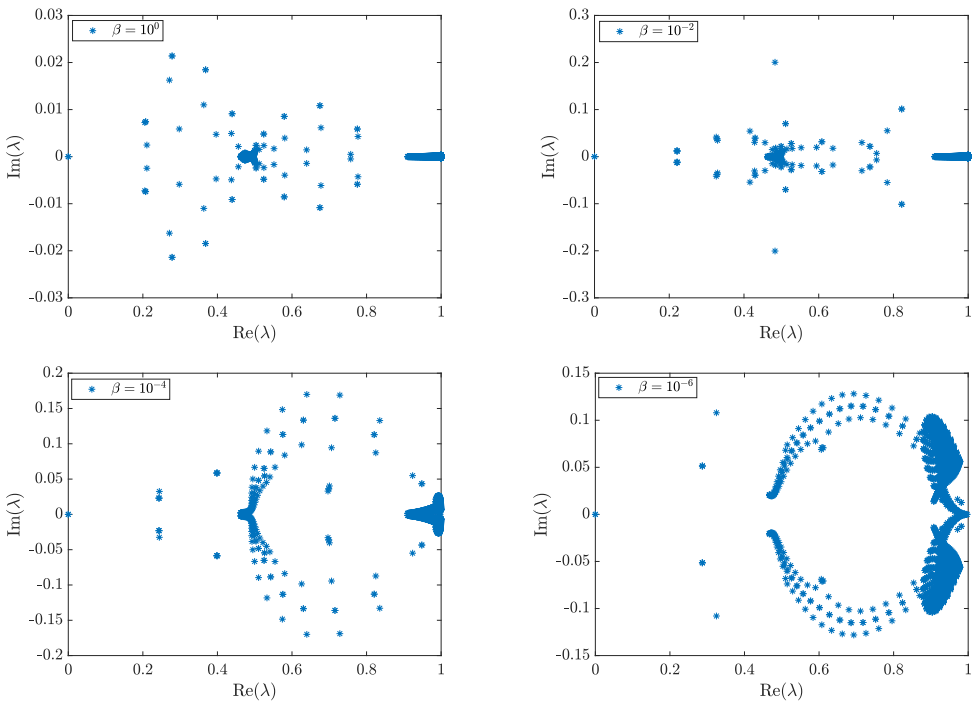
and $\mathcal{D}_p$ the corresponding differential operator on the pressure space; here, Id represents the identity operator. Employing stable finite elements and working as in Section 5.3, we obtain the following expression for (5.11):

$$\widehat{S}_{\mathcal{A},\mathrm{S}} = \begin{bmatrix} K_p & 0 \\ 0 & K_p \end{bmatrix} \begin{bmatrix} M_p & K_p \\ K_p & -M_{\beta,p} \end{bmatrix}^{-1} \begin{bmatrix} M_p & 0 \\ 0 & M_p \end{bmatrix} \approx S_{\mathcal{A},\mathrm{S}}, \tag{5.33}$$

where we set $M_{\beta,p} = \frac{1}{\beta} M_p$. We approximate the actions of M_p^{-1} by a fixed number of Chebyshev semi-iteration, and of K_p^{-1} by the action of a multigrid routine, for example.

Before moving to find an approximation to the Schur complements for the instationary case, we would like to show the effectiveness of our approach for the stationary problem. In Figure 5.1, we report the eigenvalues of the matrix $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$, for level of refinement $\mathtt{l} = 5$, and for some range of β, where $\mathtt{l}$ represents a (spatial) uniform grid of mesh-size $h = 2^{1-\mathtt{l}}$ for $\mathbf{Q}_1$ basis functions, and $h = 2^{-\mathtt{l}}$ for $\mathbf{Q}_2$ elements, in each dimension. Here, the matrix $\widehat{S}_{\mathcal{A},\mathrm{S}}$ denotes the Schur complement approximation derived above when "pinning" the value of one of the nodes of the matrix K_p, for each K_p in $\mathtt{blkdiag}(K_p, K_p)$.

Figure 5.1: Commutator approximation for stationary Stokes control. Eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$, with $\Omega = (-1,1)^2$, for $\beta = 10^{-j}$, $j = 0, 2, 4, 6$, and $\mathtt{l} = 5$.



We note that there is a strong cluster of eigenvalues around 0.5 and around 1. In addition, we note that the real part of the (non-zero) eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ are all clustered between 0.2 and 1, independently of the regularization parameter β. We would like to mention that in Figure 5.1 we have two zero eigenvalues in each plot, for the matrix $B^\top$ not being of full rank. Specifically, for enclosed flow we have $\mathtt{null}(B^\top) = \{\mathbf{1}\}$ for standard Lagrangian finite elements, where $\mathbf{1}$ is the vector of all ones, implying that the Schur complement $S_{\mathcal{A},\mathrm{S}}$ has exactly two zero eigenvalues. In addition, from here we expect the Schur complements arising from the instationary case to have a number of zero eigenvalues proportional to the number of time steps. Specifically, the number of zero eigenvalues of $S_{\mathcal{A},\mathrm{BE}}$ is $2(n_t + 1)$, whereas the ones of $S_{\mathcal{A},\mathrm{CN}}$ are $2n_t$, where n_t is the number of time steps. Finally, we would like to note that, since both the Schur complement $S_{\mathcal{A},\mathrm{S}}$

and its approximation $\widehat{S}_{\mathcal{A},\mathrm{S}}$ are indefinite, we cannot expect the eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ to be real.

Instationary Stokes Control with Backward Euler

We now derive an efficient approximation to the Schur complement $S_{\mathcal{A},\mathrm{BE}} = \Psi_{\mathrm{BE}}(\Phi_{\mathrm{BE}})^{-1}\Psi_{\mathrm{BE}}^{\top}$ of (5.22). As above, we apply the commutator argument (5.9); however, we do not consider the heat equation as part of the differential operator $\mathcal{D}$, but rather employ an operator that "mimics" the blocks of Φ_{BE} defined in (5.23). With this aim, we consider (5.9) with $\bar{m} = 2(n_t+1)$ and the differential operator:

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}_{\mathrm{BE}}^{1,1} & \mathcal{D}_{\mathrm{BE}}^{1,2} \\ \mathcal{D}_{\mathrm{BE}}^{2,1} & \mathcal{D}_{\mathrm{BE}}^{2,2} \end{bmatrix},$$

where $\mathcal{D}_{\mathrm{BE}}^{1,1} = \tau I_{n_t+1,1} \otimes \mathrm{Id}$, $\mathcal{D}_{\mathrm{BE}}^{2,2} = -\frac{\tau}{\beta} I_{n_t+1,2} \otimes \mathrm{Id}$, and

$$\mathcal{D}_{\mathrm{BE}}^{1,2} = \begin{bmatrix} \mathcal{D}_{\mathrm{BE}} & -\mathrm{Id} & & \\ & \ddots & \ddots & \\ & & \mathcal{D}_{\mathrm{BE}} & -\mathrm{Id} \\ & & & \mathcal{D}_{\mathrm{BE}} \end{bmatrix}, \quad \mathcal{D}_{\mathrm{BE}}^{2,1} = \begin{bmatrix} \mathcal{D}_{\mathrm{BE}} & & & \\ -\mathrm{Id} & \mathcal{D}_{\mathrm{BE}} & & \\ & \ddots & \ddots & \\ & & -\mathrm{Id} & \mathcal{D}_{\mathrm{BE}} \end{bmatrix},$$

with $\mathcal{D}_{\mathrm{BE}} = -\tau\nabla^2 + \mathrm{Id}$. As above, we define $\mathcal{D}_p$ as the corresponding differential operator on the pressure space. Discretizing (5.9) and observing that $S_{\mathcal{A},\mathrm{BE}} = \tau^2 \vec{B}\, \mathbf{D}^{-1} \vec{B}^{\top}$, with $\mathbf{D}$ the discretization of the differential operator $\mathcal{D}$ and $\vec{B} = I_{2(n_t+1)} \otimes B$, we obtain the following approximation:

$$\widehat{S}_{\mathcal{A},\mathrm{BE}} = \tau^2 \mathcal{K}_p^{\mathrm{BE}} \begin{bmatrix} D_{p,\,\mathrm{BE}}^{1,1} & D_{p,\,\mathrm{BE}}^{1,2} \\ D_{p,\,\mathrm{BE}}^{2,1} & D_{p,\,\mathrm{BE}}^{2,2} \end{bmatrix}^{-1} \mathcal{M}_p^{\mathrm{BE}} \approx S_{\mathcal{A},\mathrm{BE}}.$$

Here, we set

$$\begin{aligned} \mathcal{K}_p^{\mathrm{BE}} &= I_{2(n_t+1)} \otimes K_p, & \mathcal{M}_p^{\mathrm{BE}} &= I_{2(n_t+1)} \otimes M_p, \\ \mathcal{D}_{p,\,\mathrm{BE}}^{1,1} &= \tau I_{n_t+1,1} \otimes M_p, & \mathcal{D}_{p,\,\mathrm{BE}}^{2,2} &= -\frac{\tau}{\beta} I_{n_t+1,2} \otimes M_p, \end{aligned}$$

and

$$D_{p,\,\mathrm{BE}}^{1,2} = \begin{bmatrix} L_p^{\mathrm{BE}} & -M_p & & \\ & \ddots & \ddots & \\ & & L_p^{\mathrm{BE}} & -M_p \\ & & & L_p^{\mathrm{BE}} \end{bmatrix}, \quad D_{p,\,\mathrm{BE}}^{2,1} = \begin{bmatrix} L_p^{\mathrm{BE}} & & & \\ -M_p & L_p^{\mathrm{BE}} & & \\ & \ddots & \ddots & \\ & & -M_p & L_p^{\mathrm{BE}} \end{bmatrix},$$

with $L_p^{\mathrm{BE}} = \tau K_p + M_p$. As above, M_p^{-1} and K_p^{-1} are approximated, for example, by a fixed number of Chebyshev semi-iteration and by the action of a multigrid routine, respectively.

Before moving to find an approximation to the Schur complement arising from the application of Crank–Nicolson in time, we would like to make some

remarks on the differential operator $\mathcal{D}$ employed in the commutator argument above. As we mentioned, rather than considering the heat equation as part of the differential operator, we have chosen $\mathcal{D}$ in such a way that mimics the blocks of Φ_{BE}. We made this choice to address two not obvious questions, both related to the time discretization. In fact, one may decide to employ a different scheme to discretize the time derivative (for instance, Crank–Nicolson), and this will lead to a linear system with different spectral properties to the one desired. Alternatively, one can choose to discretize the time interval into $\bar{n}_t$ subintervals, with $\bar{n}_t \neq n_t$, with a similar result on the approximation obtained. In practice, we have chosen $\mathcal{D}$ as a spatial differential operator that does not account for the time discretization used. In addition, our choice of $\mathcal{D}$ avoids us having to impose artificial initial and final time conditions on the pressure space. We would like to mention that those conditions would have been imposed only in the definition of the preconditioning operator, meaning that we would not have modified the right-hand side of the whole system. A similar argument can be made for the Crank–Nicolson discretization.

Instationary Stokes Control with Crank–Nicolson

As we have done for the Schur complement arising from the backward Euler discretization, we apply the commutator argument (5.9), employing a differential operator $\mathcal{D}$ that mimics the blocks of a suitable matrix. Before presenting $\mathcal{D}$, we note that the Schur complement $S_{\mathcal{A},\mathrm{CN}} = \Psi_{\mathrm{CN}}(\Phi_{\mathrm{CN}})^{-1}\Psi_{\mathrm{CN}}^{\top}$ can be rewritten as

$$S_{\mathcal{A},\mathrm{CN}} = \begin{bmatrix} T_3 & 0 \\ 0 & T_4 \end{bmatrix} \begin{bmatrix} \widetilde{\mathcal{B}}^{\mathrm{CN}} & 0 \\ 0 & \widetilde{\mathcal{B}}^{\mathrm{CN}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathcal{M}}^{\mathrm{CN}} & (\widetilde{\mathcal{L}}^{\mathrm{CN}})^{\top} \\ \widetilde{\mathcal{L}}^{\mathrm{CN}} & -\widetilde{\mathcal{M}}_{\beta}^{\mathrm{CN}} \end{bmatrix}^{-1} \begin{bmatrix} \widetilde{\mathcal{B}}^{\mathrm{CN}} & 0 \\ 0 & \widetilde{\mathcal{B}}^{\mathrm{CN}} \end{bmatrix}^{\top}.$$

We now consider (5.9) with $\bar{m} = 2n_t$ and the differential operator

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}_{\mathrm{CN}}^{1,1} & \mathcal{D}_{\mathrm{CN}}^{1,2} \\ \mathcal{D}_{\mathrm{CN}}^{2,1} & \mathcal{D}_{\mathrm{CN}}^{2,2} \end{bmatrix},$$

where $\mathcal{D}_{\mathrm{CN}}^{1,1} = \frac{\tau}{2} I_{n_t,4}^{\top} \otimes \mathrm{Id}$, $\mathcal{D}_{\mathrm{CN}}^{2,2} = -\frac{\tau}{2\beta} I_{n_t,4} \otimes \mathrm{Id}$, and

$$\mathcal{D}_{\mathrm{CN}}^{1,2} = \begin{bmatrix} \mathcal{D}^{+} & \mathcal{D}^{-} & & \\ & \ddots & \ddots & \\ & & \mathcal{D}^{+} & \mathcal{D}^{-} \\ & & & \mathcal{D}^{+} \end{bmatrix}, \quad \mathcal{D}_{\mathrm{CN}}^{2,1} = \begin{bmatrix} \mathcal{D}^{+} & & & \\ \mathcal{D}^{-} & \mathcal{D}^{+} & & \\ & \ddots & \ddots & \\ & & \mathcal{D}^{-} & \mathcal{D}^{+} \end{bmatrix},$$

with $\mathcal{D}^{\pm} = -\frac{\tau}{2}\nabla^2 \pm \mathrm{Id}$. Again, we define $\mathcal{D}_p$ as the corresponding differential operator on the pressure space. Proceeding as above, we then derive the following approximation:

$$\widehat{S}_{\mathcal{A},\mathrm{CN}} = \tau^2 \begin{bmatrix} T_3 & 0 \\ 0 & T_4 \end{bmatrix} \mathcal{K}_p^{\mathrm{CN}} \begin{bmatrix} D_{p,\,\mathrm{CN}}^{1,1} & D_{p,\,\mathrm{CN}}^{1,2} \\ D_{p,\,\mathrm{CN}}^{2,1} & D_{p,\,\mathrm{CN}}^{2,2} \end{bmatrix}^{-1} \mathcal{M}_p^{\mathrm{CN}} \approx S_{\mathcal{A},\mathrm{CN}}.$$

Here, we set

$$\mathcal{K}_p^{\mathrm{CN}} = I_{2n_t} \otimes K_p, \qquad \mathcal{M}_p^{\mathrm{CN}} = I_{2n_t} \otimes M_p,$$
$$\mathcal{D}_{p,\,\mathrm{CN}}^{1,1} = \frac{\tau}{2} I_{n_t,4}^{\top} \otimes M_p, \qquad \mathcal{D}_{p,\,\mathrm{CN}}^{2,2} = -\frac{\tau}{2\beta} I_{n_t,4} \otimes M_p,$$

and

$$D_{p,\,\mathrm{CN}}^{1,2} = \begin{bmatrix} L_p^+ & L_p^- & & \\ & \ddots & \ddots & \\ & & L_p^+ & L_p^- \\ & & & L_p^+ \end{bmatrix}, \quad D_{p,\,\mathrm{CN}}^{2,1} = \begin{bmatrix} L_p^+ & & & \\ L_p^- & L_p^+ & & \\ & \ddots & \ddots & \\ & & L_p^- & L_p^+ \end{bmatrix},$$

with $L_p^{\pm} = \frac{\tau}{2} K_p \pm M_p$.

Remark 7. *To summarize, aside from matrix–vector products, the main computational work for our Crank–Nicolson preconditioner involves n_t applications of Chebyshev semi-iteration to $\mathbf{M}$ and $2n_t$ multigrid processes per Uzawa or inner GMRES iteration, in addition to $2n_t$ applications of Chebyshev semi-iteration to M_p and $2n_t$ multigrid processes for K_p to approximate the Schur complement. This is a similar computational workload as for the backward Euler preconditioner, as the latter requires $n_t + 1$ applications of Chebyshev semi-iteration and $2(n_t + 1)$ applications of a multigrid process per Uzawa or inner GMRES iteration, and $2(n_t + 1)$ approximations of M_p and K_p for the Schur complement approximation.*

5.6 Numerical Results

We now demonstrate the effectiveness of our preconditioners by presenting numerical results. In all our tests, $d = 2$ (that is, $x = [x_1, x_2]^{\top}$), and $\Omega = (-1, 1)^2$. We employ inf–sup stable Taylor–Hood $\mathbf{Q}_2$–$\mathbf{Q}_1$ finite elements in the spatial dimensions, with level of refinement `l` representing a (spatial) uniform grid of mesh-size $h = 2^{1-\mathtt{l}}$ for $\mathbf{Q}_1$ basis functions, and $h = 2^{-\mathtt{l}}$ for $\mathbf{Q}_2$ elements, in each dimension.

As our preconditioners are non-symmetric and require an inner solve for the $(1, 1)$-block, for the outer solver we apply flexible GMRES [155] restarted every 10 iterations, up to a tolerance 10^{-6} on the relative residual (unless otherwise stated). Our implementation is based on the flexible GMRES routine in the TT-Toolbox [127]. As we mentioned, we can approximately invert the $(1, 1)$-block by employing a fixed number of Uzawa or GMRES iterations. Although the preconditioner derived above for the $(1, 1)$-block of stationary Stokes control can be employed within an Uzawa iteration, we report only results when using GMRES, and apply both methods only in the instationary case. Specifically, to apply the approximate inverse of the $(1, 1)$-block, we take 5 iterations of GMRES (or Uzawa when specified). We employ the GMRES routine implemented in MATLAB. We apply 20 steps of Chebyshev semi-iteration to mass matrices (on the velocity or pressure space); we use 4 V-cycles of the AGMG routine [118, 121, 122, 123] to approximate other matrices constructed on the velocity space, while

employing 2 V-cycles (with 2 symmetric Gauss–Seidel iterations for pre-/post-smoothing) of the `HSL_MI20` solver [22] for stiffness matrices on the pressure space within our Schur complement approximation.

All tests are run on MATLAB R2018b, using a 1.70GHz Intel quad-core i5 processor and 8 GB RAM on an Ubuntu 18.04.1 LTS operating system. All CPU times below are reported in seconds.

5.6.1 Stationary Stokes Control

We first test our solver on the stationary Stokes control problem (5.1)–(5.2). We set $\vec{f} = \vec{0}$, $\vec{v}_d = \vec{0}$, and

$$\vec{g} = \begin{cases} [1,0]^\top & \text{on } \partial\Omega_1 := (-1,1) \times \{1\}\,, \\ [0,0]^\top & \text{on } \partial\Omega \backslash \partial\Omega_1. \end{cases}$$

In Table 5.1 we report the number of GMRES iterations and the elapsed CPU time, together with the degrees of freedom for the stationary Stokes control problem considered, for different levels of refinement `l`, and values of β.

Table 5.1: Degrees of freedom (DoF), GMRES iterations, and CPU times for stationary Stokes control problem, for a range of `l` and β.

		$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	DoF	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
3	1062	15	0.34	18	0.41	17	0.32	16	0.31	15	0.27	13	0.09	10	0.12
4	4422	15	0.89	19	1.10	18	0.98	16	0.78	16	0.97	15	0.80	14	0.66
5	18,054	20	4.07	20	4.06	23	4.69	16	3.18	16	2.90	16	2.88	15	2.15
6	72,966	26	24.0	33	30.2	23	20.9	19	17.2	16	14.2	16	13.5	15	12.2
7	293,382	27	97.2	27	96.2	29	103	22	77.1	17	59.1	14	47.9	17	57.1
8	1,176,582	36	594	37	612	36	594	26	428	20	330	18	296	15	246

Table 5.1 demonstrates the robustness of our proposed preconditioner. The numbers of iterations are roughly constant, showing a slight increase only for large values of β. The CPU time scales approximately linearly with respect to the dimension of the systems, with a marginal increase for very fine grids; in this case we observe that the AGMG multigrid routine does not scale exactly linearly.

5.6.2 Instationary Stokes Control

We now test the robustness of our solver on the instationary Stokes control problem (5.3)–(5.4), where we set $t_f = 2$, $\vec{f}(x,t) = \vec{0}$, the initial condition $\vec{v}_0(x) = \vec{0}$, and boundary conditions

$$\vec{g}(x,t) = \begin{cases} [t,0]^\top & \text{on } \partial\Omega_1 \times (0,1), \\ [1,0]^\top & \text{on } \partial\Omega_1 \times [1,t_f), \\ [0,0]^\top & \text{on } (\partial\Omega \backslash \partial\Omega_1) \times (0,t_f). \end{cases}$$

We present results obtained when employing backward Euler and Crank–Nicolson discretizations in time, when applying both Uzawa and GMRES for approximating the inverse of the $(1,1)$-block of the corresponding matrices. Setting

$$c_1 = 1 - \sqrt{(\tfrac{100}{49}(x_1 - \tfrac{1}{2}))^2 + (\tfrac{100}{99}x_2)^2},$$
$$c_2 = 1 - \sqrt{(\tfrac{100}{49}(x_1 + \tfrac{1}{2}))^2 + (\tfrac{100}{99}x_2)^2},$$

we seek the (divergence-free) desired state:

$$\vec{v}_d(x,t) = \begin{cases} c_1 \cos(\tfrac{\pi t}{2})\,[(\tfrac{100}{99})^2 x_2, -(\tfrac{100}{49})^2(x_1 - \tfrac{1}{2})]^\top & \text{if } c_1 \geqslant 0, \\ c_2 \cos(\tfrac{\pi t}{2})\,[-(\tfrac{100}{99})^2 x_2, (\tfrac{100}{49})^2(x_1 + \tfrac{1}{2})]^\top & \text{if } c_2 \geqslant 0, \\ [0,0]^\top & \text{otherwise.} \end{cases}$$

Backward Euler for Instationary Stokes Control

We first report the results obtained when employing the backward Euler scheme in time. In Table 5.2 we provide the number of GMRES iterations and the elapsed CPU time, together with the degrees of freedom for the instationary Stokes control problem with backward Euler in time, when GMRES is employed as the inner solver to approximately invert the $(1,1)$-block Φ_{BE}. For this test, we choose the time-step $\tau = 0.05$ (that is, $n_t = 40$), while the level of refinement `l` refers to a spatial uniform grid constructed as above. We report the results for different levels of refinement `l` and regularization parameters β. In Table 5.3 we provide the number of GMRES iterations and the elapsed CPU time, together with the degrees of freedom for the instationary Stokes control problem with backward Euler applied in time, when Uzawa iteration is employed as the inner solver for approximately inverting the $(1,1)$-block Φ_{BE}. Here, we fix the level of refinement `l` $= 5$, and report the results for different values of n_t and β.

Table 5.2: Degrees of freedom (DoF), GMRES iterations, and CPU times for instationary Stokes control problem with backward Euler in time, with GMRES as the inner solver for Φ_{BE}, and $n_t = 40$, for a range of `l` and β.

		$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	DoF	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	10,086	15	5.79	16	6.10	18	6.87	17	6.44	15	5.67	14	5.34	20	7.61
3	43,542	16	14.0	18	16.0	19	15.3	16	13.8	16	12.6	16	6.07	22	13.6
4	181,302	16	36.7	19	43.5	19	42.0	17	44.7	17	43.7	17	41.1	22	49.1
5	740,214	16	155	22	211	20	191	17	160	17	150	17	150	21	156
6	2,991,606	25	1123	24	1080	23	1027	17	754	16	709	17	725	22	932

From Tables 5.2–5.3 we observe that our solvers demonstrate mesh- and parameter-robustness, for wide variations of n_t, `l`, and β. The CPU times scale approximately linearly with respect to degrees of freedom, except for very fine grids. In particular, from Table 5.2 we see that, as we refine the spatial grid,

Table 5.3: Degrees of freedom (DoF), GMRES iterations, and CPU times for instationary Stokes control problem with backward Euler in time, with Uzawa as the inner solver for Φ_{BE}, and $\mathtt{l} = 5$, for a range of n_t and β.

		$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
n_t	DoF	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU	it	CPU
10	198,594	14	26.0	15	28.3	17	31.6	17	31.5	18	31.6	21	36.6	40	60.1
20	379,134	14	51.0	15	54.0	17	60.8	17	60.5	17	56.9	18	59.3	29	80.8
40	740,214	15	103	15	103	16	110	17	115	17	107	18	113	22	116
80	1,462,374	15	201	15	202	16	213	17	223	17	213	17	211	18	187
160	2,906,694	15	375	15	377	16	400	17	421	17	426	17	411	18	370

the number of degrees of freedom increases by a factor of 4, and so do the CPU times. In addition, from Table 5.3 we see that, as we double the number of time steps, the number of degrees of freedom is doubled, with a similar effect on the CPU times.

Crank–Nicolson for Instationary Stokes Control

We now test our solver when applying Crank–Nicolson in time. Here, for level of refinement $\mathtt{l}$ we divide the time interval into subintervals of length $2^{1-\mathtt{l}}$ and consider a spatial uniform grid of refinement level $\mathtt{l}$. Before showing the robustness of our solver by solving the problem above, we test our solver on the instationary Stokes control problem (5.3)–(5.4), for an artificial problem with exact solution. This will allow us to verify the predicted order of convergence of the Crank–Nicolson method. We take $t_f = 2$, the desired state

$$\begin{aligned}\vec{v}_d(x_1,x_2,t) = 4\beta\big[&x_2\big(2(3x_1^2-1)(x_2^2-1)+3(x_1^2-1)^2\big),\\ &-x_1\big(3(x_2^2-1)^2+2(x_1^2-1)(3x_2^2-1)\big)\big]^\top\\ &+e^{t_f-t}\big[20x_1x_2^3+2\beta x_2\big((x_1^2-1)^2(x_2^2-7)-4(3x_1^2-1)(x_2^2-1)+2\big),\\ &5(x_1^4-x_2^4)-2\beta x_1\big((x_2^2-1)^2(x_1^2-7)-4(x_1^2-1)(3x_2^2-1)-2\big)\big]^\top,\end{aligned}$$

and the force function

$$\begin{aligned}\vec{f}(x_1,x_2,t) = e^{t_f-t}\big[&-20x_1x_2^3-2x_2(x_1^2-1)^2(x_2^2-1),\\ &5(x_2^4-x_1^4)+2x_1(x_1^2-1)(x_2^2-1)^2\big]^\top\\ &+\big[2x_2(x_1^2-1)^2(x_2^2-1),-2x_1(x_1^2-1)(x_2^2-1)^2\big]^\top.\end{aligned}$$

The analytic solutions for this problem are:

$$\begin{aligned}&\vec{v}(x_1,x_2,t) = e^{t_f-t}[20x_1x_2^3,5x_1^4-5x_2^4]^\top,\\ &p(x_1,x_2,t) = e^{t_f-t}\,(60x_1^2x_2-20x_2^3)+\text{constant},\\ &\vec{\zeta}(x_1,x_2,t) = \beta\,(e^{t_f-t}-1)\,[2x_2(x_1^2-1)^2(x_2^2-1),-2x_1(x_1^2-1)(x_2^2-1)^2]^\top,\\ &\mu(x_1,x_2,t) = \beta e^{t_f-t}(4x_1x_2)+\text{constant},\end{aligned}$$

with initial and boundary conditions obtained from this $\vec{v}$. Here, up to a time-dependent function the state velocity and pressure are the solutions of the (forward) Stokes equations presented in [120, Section 3.1]. In Table 5.4 we report the level of refinement `l`, the number of GMRES iterations[6], the CPU time, and the resulting errors for different values of β. The error is evaluated in the $L^\infty(L^2)$ norm, approximated for $\vec{v}$ as

$$\vec{v}_{\mathrm{err}} = \max_n \left[(\boldsymbol{v}_n - \boldsymbol{v}_{\mathrm{sol},n})^\top \mathbf{M} (\boldsymbol{v}_n - \boldsymbol{v}_{\mathrm{sol},n})\right]^{1/2},$$

where $\boldsymbol{v}_{\mathrm{sol},n}$ is the discretized exact solution for $\vec{v}$ at time t_n. In the same way we define the error for the adjoint velocity $\vec{\zeta}_{\mathrm{err}}$. Finally, for the total size of the systems solved, we refer the reader to Table 5.5.

Table 5.4: GMRES iterations, CPU times, and errors for instationary Stokes control problem, solved using Crank–Nicolson, with GMRES as the inner solver for Φ_{CN}, for a range of `l` and β.

	$\beta = 10^0$				$\beta = 10^{-2}$				$\beta = 10^{-4}$			
`l`	`it`	CPU	$\vec{v}_{\mathrm{err}}$	$\vec{\zeta}_{\mathrm{err}}$	`it`	CPU	$\vec{v}_{\mathrm{err}}$	$\vec{\zeta}_{\mathrm{err}}$	`it`	CPU	$\vec{v}_{\mathrm{err}}$	$\vec{\zeta}_{\mathrm{err}}$
2	22	0.85	4.76e-1	2.49e-1	22	0.79	5.66e-1	1.16e-1	16	0.73	8.63e0	5.45e-2
3	22	4.28	3.34e-2	5.68e-2	22	4.24	7.07e-2	3.42e-2	19	3.39	2.47e0	2.67e-2
4	23	23.9	2.25e-3	1.15e-2	24	24.1	7.35e-3	7.79e-3	20	23.2	3.73e-1	7.30e-3
5	23	200	1.74e-4	2.15e-3	27	232	6.70e-4	1.59e-3	20	162	3.84e-2	1.55e-3
6	26	2082	2.16e-5	4.00e-4	37	2960	5.97e-5	3.02e-4	23	1830	3.38e-3	3.00e-4

From the discretization errors reported in Table 5.4, we first note that the method is converging at second-order. We experienced similar convergence behaviour for the pressure variables, after shifting the numerical approximation of the pressures at each time step by the values of the corresponding numerical solution at the origin of the axis at the corresponding time step; we shifted the numerical approximation of the pressures as for enclosed flow the pressure solution is only unique up to an additive constant. Secondly, we note that the preconditioner behaves robustly with respect to the level of refinement `l` and the regularization parameter β, with the number of iterations slightly increasing for very fine grids. The elapsed CPU time scales almost exactly linearly, aside from the AGMG multigrid routine for very fine grids.

We now report the results obtained when applying Crank–Nicolson in time, showing the robustness of our approach by solving the problem defined at the beginning of Section 5.6.2. In Table 5.5 we provide the number of GMRES iterations and the elapsed CPU time, together with the degrees of freedom for the instationary Stokes control problem with Crank–Nicolson in time, when GMRES is employed as the inner solver for approximately inverting the $(1,1)$-block Φ_{CN}.

[6]For this problem we run GMRES until a relative reduction on the residual of 10^{-9} is achieved, in order to clearly show the predicted second-order convergence of the time-stepping scheme.

In Table 5.6 we provide the number of GMRES iterations and the elapsed CPU time, for the instationary Stokes control problem solved using Crank–Nicolson in time, when Uzawa iteration is employed as the inner solver for approximately inverting the $(1,1)$-block Φ_{CN}. We again report the results for different levels of refinement `l` and regularization parameters β. In Figure 5.2 we show the numerical solutions of the state and adjoint velocities $\vec{v}$ and $\vec{\zeta}$, at time $t = 1$, and of the pressure p, at time $t = 1.0625$, for $\beta = 10^{-1}$ and `l` $= 4$.

Table 5.5: Degrees of freedom (DoF), GMRES iterations, and CPU times for instationary Stokes control problem, with Crank–Nicolson in time ($\tau = h$), with GMRES as the inner solver for Φ_{CN}, for a range of `l` and β.

		$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	DoF	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	984	14	0.54	15	0.68	16	0.67	15	0.62	12	0.50	10	0.42	9	0.37
3	8496	15	2.89	16	3.11	17	3.28	16	2.88	15	2.69	13	1.23	10	1.36
4	70,752	16	16.5	18	18.3	19	18.7	16	18.3	16	18.4	15	16.0	13	12.6
5	577,728	16	139	19	163	20	171	19	158	16	128	15	119	15	103
6	4,669,824	22	1758	24	1915	26	2087	18	1437	17	1344	15	1149	15	1155

Table 5.6: GMRES iterations, and CPU times for instationary Stokes control problem, with Crank–Nicolson in time ($\tau = h$), with Uzawa as the inner solver for Φ_{CN}, for a range of `l` and β.

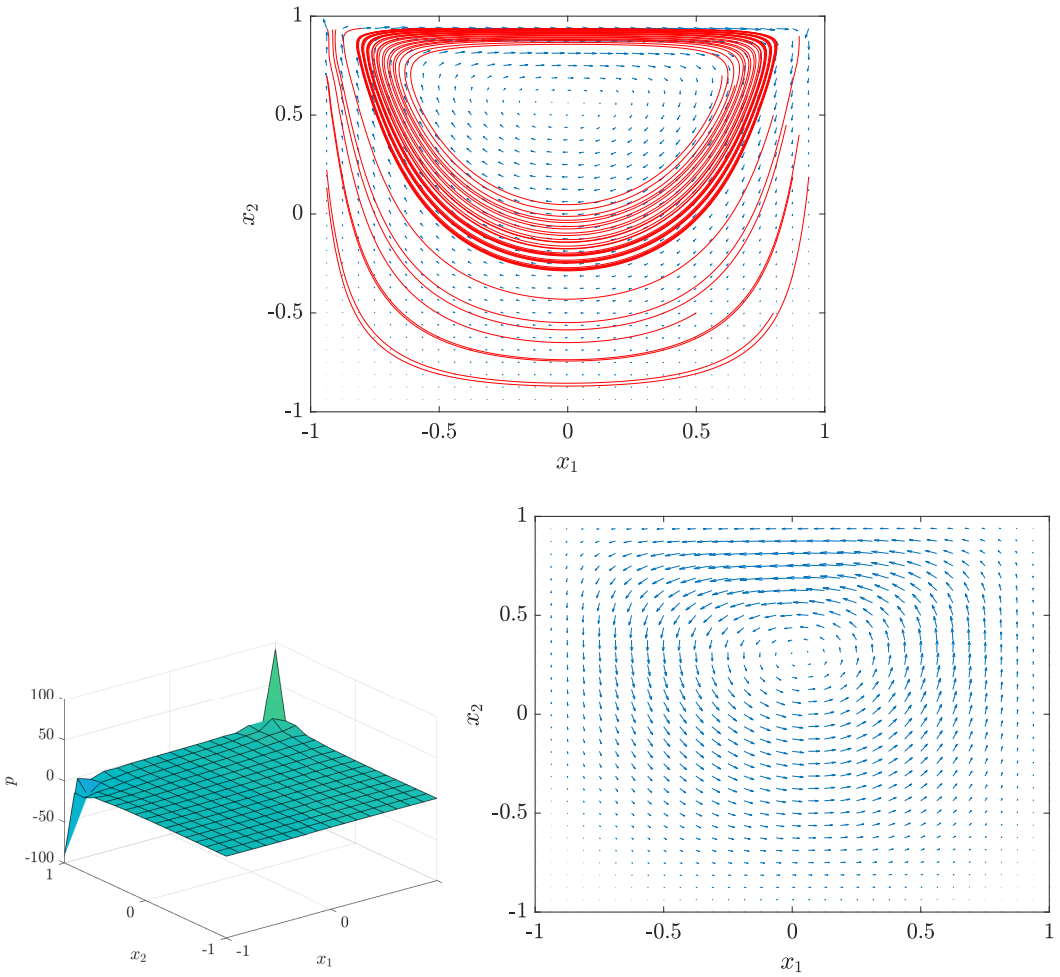
	$\beta = 10^0$		$\beta = 10^{-2}$		$\beta = 10^{-4}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	16	0.39	18	0.46	14	0.35	10	0.31
3	16	2.20	18	2.66	17	2.11	12	1.13
4	16	11.8	18	13.0	17	14.0	15	10.6
5	15	92.0	16	98.2	17	96.4	17	82.7
6	14	806	16	918	16	911	17	940

From Tables 5.5–5.6 we observe the mesh- and parameter-robustness of our solvers, with the CPU time scaling approximately linearly with the size of the system, except for very fine grids. Also in this case, we observe that the AGMG multigrid routine does not scale exactly linearly.

5.7 Summary and Comments

In this chapter, we presented mesh- and parameter-robust preconditioners for distributed Stokes control problems, of both stationary and instationary type. The preconditioners were based on a generalization of the pressure convection–diffusion preconditioner, and were applied within the flexible GMRES algorithm,

Figure 5.2: Solution plots for the instationary Stokes control problem, for $\beta = 10^{-1}$ and $\mathtt{l} = 4$. Top: velocity $\vec{v}$ at $t = 1$. Bottom left: pressure p at $t = 1.0625$. Bottom right: adjoint velocity $\vec{\zeta}$ at $t = 1$.



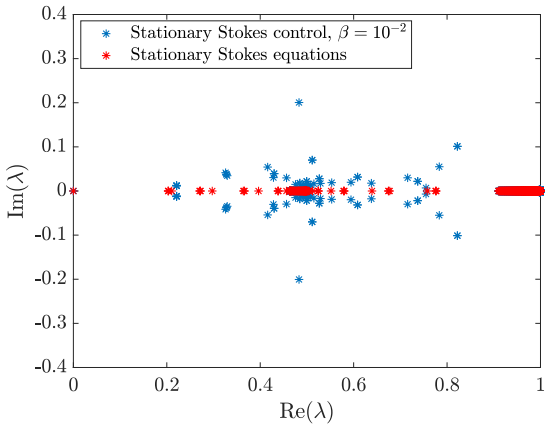
and in the instationary setting using backward Euler and Crank–Nicolson discretizations in time. Numerical results demonstrated the versatility and effectiveness of this approach when solving a range of huge-scale linear systems. In the following chapter, we will adapt the strategy presented here to the distributed Navier–Stokes control problems, in both the stationary and instationary settings.

Before moving on to next chapter, we would like to devote some discussion to the approach we have outlined above regarding the commutator (5.9), and try to link our work to the inspirational quote of this chapter. For ease of exposition, we will consider only the stationary Stokes control problem, although some of the following comments can also be applied to the instationary case.

Although we were not able to prove any spectral property of the matrix $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ due to the indefiniteness of both matrices and hence the presence of complex eigenvalues, we showed (numerically) the robustness of our approach for a wide class of problems. In fact, the number of iterations required to reach a prescribed accuracy was roughly constant for the tests presented. In addition, in Figure 5.1 we reported the eigenvalues of the matrix $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$, for $\mathtt{l} = 5$, and for

some range of β. As we observed, the real part of the (non-zero) eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ are all clustered between 0.2 and 1, independently of the regularization parameter β. We want to mention that we obtained similar results also for different level of grid refinement $\mathtt{l}$. For a diligent eye, the eigenvalue distributions shown in Figure 5.1 are even more interesting: it seems that the real part of the non-zero eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ lie in an interval whose endpoints depend on an inf–sup constant and on a boundedness constant (as for the forward Stokes equations). For this reason, in Figure 5.3 we report the eigenvalues of the matrix $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$, for $\beta = 10^{-2}$, together with the eigenvalues of $M_p^{-1}(B\mathbf{K}^{-1}B^\top)$, for level of refinement $\mathtt{l} = 5$. From Figure 5.3, we do believe that further investigation would be of interest in order to understand the effectiveness of the approach devised in this chapter, specifically if it is possible to prove some bounds on the real parts of the eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$.

Figure 5.3: Commutator approximations for stationary Stokes control and stationary Stokes equations, with $\Omega = (-1,1)^2$. In blue, eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$, for $\beta = 10^{-2}$, and $\mathtt{l} = 5$. In red, eigenvalues of $M_p^{-1}(B\mathbf{K}^{-1}B^\top)$, for $\mathtt{l} = 5$.



Finally, we have a last comment on the preconditioners derived in this chapter. As we mentioned above, we have employed GMRES as the Krylov solver due to the preconditioners not being symmetric. However, it is worth noting that we did so only to obtain more flexibility within the preconditioners. In fact, what we really lost in our commutator-based approach is not the symmetry of the preconditioners, but their positive definiteness. In order to show this, we consider again the stationary Stokes control problem.

Let us consider the discrete optimality conditions (5.14) of the stationary Stokes control problem. The starting point of our preconditioner was the (ideal) block triangular matrix

$$\mathcal{P}_1 = \begin{bmatrix} \Phi_{\mathrm{S}} & 0 \\ \Psi_{\mathrm{S}} & -S_{\mathcal{A},\mathrm{S}} \end{bmatrix}.$$

We then derived approximation of the $(1,1)$-block Φ_{S} and the Schur complement $S_{\mathcal{A},\mathrm{S}}$. The latter lead to the approximation $\widehat{S}_{\mathcal{A},\mathrm{S}}$ defined in (5.33). We know that the matrix K_p may be not invertible in its classical form, as it represents a discrete (negative) Laplacian when Neumann boundary conditions are imposed.

Again, we suppose that K_p is invertible, for example by "pinning" the value of one of the nodes. Then, we may write the inverse of $\widehat{S}_{\mathcal{A},\mathrm{S}}$ as follows:

$$\begin{aligned}\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} &= \begin{bmatrix} M_p & 0 \\ 0 & M_p \end{bmatrix}^{-1} \begin{bmatrix} M_p & K_p \\ K_p & -M_{\beta,p} \end{bmatrix} \begin{bmatrix} K_p & 0 \\ 0 & K_p \end{bmatrix}^{-1} \\ &= \begin{bmatrix} K_p^{-1} & M_p^{-1} \\ M_p^{-1} & -K_{\beta,p}^{-1} \end{bmatrix},\end{aligned}$$

where we set $K_{\beta,p}^{-1} = \frac{1}{\beta} K_p^{-1}$. From here, it is clear that the Schur complement approximation $\widehat{S}_{\mathcal{A},\mathrm{S}}$ we found is symmetric indefinite, as it is the "inverse" of a symmetric indefinite matrix. Recalling that also the $(1,1)$-block Φ_{S} is symmetric indefinite, we may conclude that the block diagonal preconditioner $\widehat{\mathcal{P}}_3$ defined as

$$\widehat{\mathcal{P}}_3 = \begin{bmatrix} \Phi_{\mathrm{S}} & 0 \\ 0 & \widehat{S}_{\mathcal{A},\mathrm{S}} \end{bmatrix}$$

is symmetric indefinite in this setting. A similar discussion also holds for the instationary case. In practice, in order to obtain a more flexible solver, we had to "give in return" some other property: the positive definiteness of our preconditioners.

Chapter 6

Preconditioning Stationary and Instationary Navier–Stokes Control Problems

> *"Tu saresti capace di piantare tutto e ricominciare la vita da capo? E scegliere una cosa, una cosa sola e di essere fedele a quella? Riuscire a farla diventare la ragione della tua vita, una cosa che raccolga tutto, che diventi tutto proprio perché è la tua fedeltà che la fa diventare infinita. Saresti capace?"*
>
> *["Would you be able to give up everything, to start life all over again... to choose one thing, just one thing, and be faithful to it... to make it the thing that gives meaning to your life... something that contains everything else... that becomes everything else just because of your boundless faith in it? Could you do that?"]*
>
> – Federico Fellini, *8½*

The solution strategies that we have devised so far required only a linear solver for the discrete optimality conditions of the problem under examination, until a specified tolerance on the relative residual is reached. We were able to use this condition because all the problems treated in the previous chapters were characterized by linear PDEs as constraints with no additional algebraic constraints on the state and/or the control variables. As opposed to those problems, the ones we tackle in the following two chapters will require us to run a non-linear process, either due to the PDEs considered being non-linear, or due to additional constraints being imposed on the variables.

In this chapter we will deal with the distributed control of the incompressible Navier–Stokes equations, in both the stationary and time-dependent settings. The incompressible Navier–Stokes equations are non-linear PDEs that describe the motion of an incompressible, viscous Newtonian fluid flow, in case the convection of the fluid plays a non-negligible role in the physics. Due to the non-linearity involved, to find a solution linearizations of the constrained problem need to be repeatedly solved until a sufficient reduction on the non-linear residual is achieved [82, 84, 145]. This has motivated researchers to devise solvers for

this type of problem which exhibit robustness with respect to all the parameters involved; see [84] for a robust multigrid method applied to Newton iteration for instationary Navier–Stokes control, for instance. Despite the recent development of parameter-robust preconditioners for the control of the (stationary and instationary time-periodic) Stokes equations [8, 96, 153, 190], to our knowledge no such preconditioner has proved to be completely robust when applied to the Navier–Stokes control problem considered below. We also point out [77] for a preconditioned iterative solver for Stokes and Navier–Stokes boundary control problems, and [147] for an efficient and robust preconditioning technique for in-domain Navier–Stokes control.

A popular preconditioner for the *Oseen linearization* of the forward stationary Navier–Stokes equations combines saddle-point theory with a commutator argument for approximating the Schur complement [92]. This type of preconditioner shows only a mild dependence on the viscosity parameter, and is robust with respect to the discretization parameter. In [132] the combination of saddle-point theory with a commutator argument has been adapted to the control of the stationary Navier–Stokes equations; we note that a commutator argument of this type was previously introduced in [131, 170] for the control of the stationary and time-dependent Stokes equations.

In the following, we will employ an Oseen linearization of the Navier–Stokes equations. The preconditioners employed for solving the discretized optimality conditions will make use of most of the techniques presented in the previous chapters. Specifically, we will make use of saddle-point theory in conjunction with the block commutator argument presented in the previous chapter for approximating the Schur complement of the corresponding systems, with the commutator including also a convection term in the differential operator. In addition, the inverses of the $(1,1)$-blocks will be applied inexactly by employing an inner preconditioned GMRES solver accelerated by a block triangular preconditioner. Again, we will employ the matching strategy described in Section 2.11, and generalize the approximations of the Schur complements of each $(1,1)$-block arising from the Stokes control problems. In practice, the preconditioners derived here can be constructed from the preconditioners for the Stokes control problems described in the previous chapter and (suitable) block matrices that include convection terms.

This chapter is structured as follows. In Section 6.1, we define the problems that we examine, that is the stationary and instationary Navier–Stokes control problems; we then present the linearization adopted in this work, and outline the linear systems arising upon discretization of the forward problem. In Section 6.2, we introduce a preconditioner for the forward stationary Navier–Stokes equation in combination with the commutator argument presented in [92]. In Section 6.3, we derive the first-order optimality conditions of the control problems and their discretization. In Section 6.4, we generalize the preconditioners derived in the previous chapter for Stokes control problems to the corresponding Navier–Stokes control problems, again applying a commutator argument to the Schur complements of the linear systems being solved. Then, in Section 6.5 we provide numerical results that show the robustness and efficiency of our approach. This chapter is based on some of the work in [101].

6.1 Problem Formulation

In this chapter we derive fast and robust preconditioned iterative methods for the distributed control of incompressible viscous fluid flow, in case of non-negligible convection; here, the physics is described by the (stationary or instationary) incompressible Navier–Stokes equations. The corresponding distributed control problem is defined as a minimization of a least-squares cost functional subject to the PDEs.

Specifically, given a spatial domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, the stationary Navier–Stokes control problem we consider is

$$\min_{\vec{v},\vec{u}} \quad J_{\mathrm{S}}(\vec{v},\vec{u}) = \frac{1}{2}\int_\Omega |\vec{v}(x) - \vec{v}_d(x)|^2 \,\mathrm{d}\Omega + \frac{\beta}{2}\int_\Omega |\vec{u}(x)|^2 \,\mathrm{d}\Omega \tag{6.1}$$

subject to

$$\begin{cases} -\nu\nabla^2\vec{v} + \vec{v}\cdot\nabla\vec{v} + \nabla p = \vec{u} + \vec{f}(x) & \text{in } \Omega, \\ -\nabla\cdot\vec{v}(x) = 0 & \text{in } \Omega, \\ \vec{v}(x) = \vec{g}(x) & \text{on } \partial\Omega. \end{cases} \tag{6.2}$$

As for the stationary Stokes control problem, $\vec{v}$ and p denote the (state) velocity and the (state) pressure respectively, $\vec{u}$ is the control variable, $\vec{v}_d$ is the desired state (velocity), and $\beta > 0$ is a regularization parameter. Further, the parameter $\nu > 0$ denotes the viscosity of the fluid. Finally, the functions $\vec{f}$ and $\vec{g}$ are given.

Similarly, the control of the instationary Navier–Stokes equations is defined as

$$\min_{\vec{v},\vec{u}} \quad J_{\mathrm{I}}(\vec{v},\vec{u}) = \frac{1}{2}\int_0^{t_f}\int_\Omega |\vec{v}(x,t) - \vec{v}_d(x,t)|^2 \,\mathrm{d}\Omega\,\mathrm{d}t + \frac{\beta}{2}\int_0^{t_f}\int_\Omega |\vec{u}(x,t)|^2 \,\mathrm{d}\Omega\,\mathrm{d}t, \tag{6.3}$$

given also a final time $t_f > 0$, subject to

$$\begin{cases} \frac{\partial\vec{v}}{\partial t} - \nu\nabla^2\vec{v} + \vec{v}\cdot\nabla\vec{v} + \nabla p = \vec{u} + \vec{f}(x,t) & \text{in } \Omega\times(0,t_f), \\ -\nabla\cdot\vec{v}(x,t) = 0 & \text{in } \Omega\times(0,t_f), \\ \vec{v}(x,t) = \vec{g}(x,t) & \text{on } \partial\Omega\times(0,t_f), \\ \vec{v}(x,0) = \vec{v}_0(x) & \text{in } \Omega, \end{cases} \tag{6.4}$$

using the same notation as above. The functions $\vec{f}$ and $\vec{g}$ as well as the initial condition $\vec{v}_0$ are known. In the following, we employ the same strategy devised for the time-dependent Stokes control problems described in the previous chapter. For this reason, we assume that $\nabla\cdot\vec{v}_0 = 0$, while generalizing our strategy to the case of $\vec{v}_0$ compressible when possible.

The constraints (6.2) and (6.4) are a system of non-linear (stationary or instationary) PDEs. In order to obtain a solution of the corresponding control problem, we make use of the Oseen linearization of the non-linear term $\vec{v}\cdot\nabla\vec{v}$, as in [145]. Note that, if the non-linear term $\vec{v}\cdot\nabla\vec{v}$ is dropped in (6.2) or (6.4), with $\nu = 1$, we obtain the corresponding distributed Stokes control problem defined in Section 5.1.

6.1.1 Non-Linear Iteration and Discretization Matrices

To introduce the linearization adopted for the control case as well as the discretization matrices, we consider the stationary Navier–Stokes equations:

$$\begin{cases} -\nu\nabla^2\vec{v} + \vec{v}\cdot\nabla\vec{v} + \nabla p = \vec{u} + \vec{f}(x) & \text{in } \Omega, \\ -\nabla\cdot\vec{v}(x) = 0 & \text{in } \Omega, \end{cases} \tag{6.5}$$

with $\vec{v} = \vec{g}$ on $\partial\Omega$. First, we introduce the weak formulation of (6.5) as follows. Let $V := \{\vec{v} \in \mathcal{H}^1(\Omega)^d \mid \vec{v} = \vec{g} \text{ on } \partial\Omega\}$, $V_0 := \{\vec{v} \in \mathcal{H}^1(\Omega)^d \mid \vec{v} = \vec{0} \text{ on } \partial\Omega\}$, and $Q := L^2(\Omega)$, with $\mathcal{H}^1(\Omega)^d$ the Sobolev space of square-integrable functions in $\mathbb{R}^d$ with square-integrable weak derivatives; then, the weak formulation reads as:

Find $\vec{v} \in V$ and $p \in Q$ such that

$$\begin{cases} \nu(\nabla\vec{v}, \nabla\vec{w}) + (\vec{v}\cdot\nabla\vec{v}, \vec{w}) - (p, \nabla\cdot\vec{w}) = (\vec{u}, \vec{w}) + (\vec{f}, \vec{w}) & \text{for all } \vec{w} \in V_0, \\ -(q, \nabla\cdot\vec{v}\,) = 0 & \text{for all } q \in Q, \end{cases} \tag{6.6}$$

where $(\cdot,\cdot)$ is the L^2-inner product on Ω. The main issue in (6.6) is how to deal with the non-linear term $(\vec{v}\cdot\nabla\vec{v}, \vec{w})$. A common strategy employs the *Picard iteration*, which is described as follows. Given the approximations $\vec{v}^{\,(k)} \in V$ and $p^{(k)} \in Q$ to $\vec{v}$ and p respectively, we consider the non-linear residuals:

$$\begin{cases} \vec{R}^{\,(k)} = (\vec{u}, \vec{w}) + (\vec{f}, \vec{w}) - \nu(\nabla\vec{v}^{\,(k)}, \nabla\vec{w}) - (\vec{v}^{\,(k)}\cdot\nabla\vec{v}^{\,(k)}, \vec{w}) + (p^{(k)}, \nabla\cdot\vec{w}), \\ r^{(k)} = (q, \nabla\cdot\vec{v}^{\,(k)}), \end{cases} \tag{6.7}$$

for any $\vec{w} \in V_0$ and $q \in Q$. Then, the Picard iteration is defined as [44, pp. 345–346]:

$$\vec{v}^{\,(k+1)} = \vec{v}^{\,(k)} + \vec{\delta v}^{\,(k)}, \qquad p^{(k+1)} = p^{(k)} + \delta p^{(k)}, \tag{6.8}$$

where $\vec{\delta v}^{\,(k)}$ and $\delta p^{(k)}$ are the solutions of

$$\begin{cases} \nu(\nabla\vec{\delta v}^{\,(k)}, \nabla\vec{w}) + (\vec{v}^{\,(k)}\cdot\nabla\vec{\delta v}^{\,(k)}, \vec{w}) - (\delta p^{(k)}, \nabla\cdot\vec{w}) = \vec{R}^{\,(k)}, \\ -(q, \nabla\cdot\vec{\delta v}^{\,(k)}) = r^{(k)}, \end{cases} \tag{6.9}$$

for any $\vec{w} \in V_0$ and $q \in Q$. Equations (6.9) are the Oseen equations for the forward stationary Navier–Stokes equations. These are posed on the continuous level, so in order to find a solution to (6.5) we now discretize them. Before doing so, we note that (6.9) represents an incompressible convection–diffusion equation, with wind vector defined by $\vec{v}^{\,(k)}$. Then, defining the *Reynolds number* as $Re = \frac{Le\,Ve}{\nu}$, where Le and Ve denote the characteristic length and velocity scale of the flow respectively, it is clear that, for $Re \gg 1$, the problem is convection-dominated. This requires us to make use of a stabilization procedure.

Letting $\{\vec{\phi}_i\}_{i=1}^{n_v}$ and $\{\psi_i\}_{i=1}^{n_p}$ be inf–sup stable finite element basis functions, we seek approximations $\vec{v}(x) \approx \sum_{i=1}^{n_v} \boldsymbol{v}_i^{(k)}\vec{\phi}_i$, $\vec{u}(x) \approx \sum_{i=1}^{n_v} \boldsymbol{u}_i\vec{\phi}_i$, $p(x) \approx \sum_{i=1}^{n_p} p_i^{(k)}\psi_i$. Denoting the vectors $\boldsymbol{v}^{(k)} = \{\boldsymbol{v}_i^{(k)}\}_{i=1}^{n_v}$, $\boldsymbol{u} = \{\boldsymbol{u}_i\}_{i=1}^{n_v}$, $\boldsymbol{p}^{(k)} = \{p_i^{(k)}\}_{i=1}^{n_p}$, a discretized

version of (6.7) is:

$$\begin{cases} \boldsymbol{R}^{(k)} = \mathbf{M}\boldsymbol{u} + \boldsymbol{f} - \mathbf{L}^{(k)}\,\boldsymbol{v}^{\,(k)} - B^{\top}\boldsymbol{p}^{(k)}, \\ \boldsymbol{r}^{(k)} = -B\,\boldsymbol{v}^{\,(k)}, \end{cases}$$

where we set $\mathbf{L}^{(k)} = \nu\mathbf{K} + \mathbf{N}^{(k)} + \mathbf{W}^{(k)}$, with

$$\mathbf{N}^{(k)} = \{n_{il}^{(k)}\}_{i,l=1}^{n_v}, \quad n_{il}^{(k)} = \int_\Omega (\vec{v}^{\,(k)} \cdot \nabla\vec{\phi}_l) \cdot \vec{\phi}_i,$$
$$\boldsymbol{f} = \{f_i\}_{i=1}^{n_v}, \quad f_i = \int_\Omega \vec{f} \cdot \vec{\phi}_i,$$

and the matrix $\mathbf{W}^{(k)}$ denotes a possible stabilization matrix for the convection operator. Here, the matrices $\mathbf{K}$ and $\mathbf{M}$ are the vector-stiffness and vector-mass matrices, respectively, and the matrix B is the negative divergence matrix. The matrix $\mathbf{N}^{(k)}$ is referred to as a *(vector-)convection matrix*, and is skew-symmetric (i.e. $\mathbf{N}^{(k)} + (\mathbf{N}^{(k)})^{\top} = 0$) in the artificial case that the $\nabla \cdot \vec{v}^{\,(k)} = 0$. We would like to note that this is never achieved in practice: even if one imposes the incompressibility constraints $-(\psi_i, \nabla \cdot \vec{v}\,)$ equal to zero for all $i = 1, 2, \ldots, n_p$, the velocity $\vec{v}^{\,(k)}$ is not exactly incompressible, but will be nearly so, see for instance [70, Section 2.2.3].

Then, the Picard iterate (6.8) may be written in discrete form as

$$\boldsymbol{v}^{\,(k+1)} = \boldsymbol{v}^{\,(k)} + \boldsymbol{\delta v}^{(k)}, \qquad \boldsymbol{p}^{(k+1)} = \boldsymbol{p}^{(k)} + \boldsymbol{\delta p}^{(k)},$$

with $\boldsymbol{\delta v}^{(k)}$ and $\boldsymbol{\delta p}^{(k)}$ solutions of

$$\begin{cases} \mathbf{L}^{(k)}\,\boldsymbol{\delta v}^{(k)} + B^{\top}\,\boldsymbol{\delta p}^{(k)} = \boldsymbol{R}^{(k)}, \\ B\,\boldsymbol{\delta v}^{(k)} = \boldsymbol{r}^{(k)}. \end{cases} \tag{6.10}$$

Regarding the stabilization procedure applied, we note that the matrix $\mathbf{W}^{(k)}$ represents a differential operator that is not physical, and is introduced only to enhance coercivity (that is, increase the positivity of the real part of the eigenvalues) of the discretization, thereby allowing it to be stable. For the reasons discussed in [100, 141], in the following we employ the Local Projection Stabilization (LPS) approach described in [11, 12, 23]. We point out [110], where the authors derive the order of convergence of one- and two-level LPS applied to the Oseen problem. For other possible stabilizations applied to the Oseen problem, see [28, 53, 58, 91, 173].

In the LPS formulation, the stabilization matrix $\mathbf{W}^{(k)}$ is defined as

$$\mathbf{W}^{(k)} = \{w_{il}^{(k)}\}_{i,l=1}^{n_v}, \qquad w_{il}^{(k)} = \delta^{(k)} \int_\Omega \kappa_h(\vec{v}^{\,(k)} \cdot \nabla\vec{\phi}_i) \cdot \kappa_h(\vec{v}^{\,(k)} \cdot \nabla\vec{\phi}_l)\,. \tag{6.11}$$

Here, $\delta^{(k)} > 0$ denotes a stabilization parameter, and $\kappa_h = \mathrm{Id} - \pi_h$ is the fluctuation operator, with Id the identity operator and π_h an L^2-orthogonal (discontinuous) projection operator defined on patches of Ω, where by a patch we mean a union of elements of our finite element discretization. In our implementation, the

domain is divided into patches consisting of two elements in each dimension. Approximation and stability properties for convergence of the method are discussed in [12, Section 3]. We define the projection π_h and the stabilization parameter $\delta^{(k)}$ locally on each patch $\mathtt{P}_m$. Specifically, as in [11] the projection is defined as

$$\pi_h(q)|_{\mathtt{P}_m} = \frac{1}{|\mathtt{P}_m|}\int_{\mathtt{P}_m} q \, \mathrm{d}\mathtt{P}_m, \quad \forall q \in L^2(\Omega),$$

where $|\mathtt{P}_m|$ is the (Lebesgue) measure of the patch, and as in [44, p. 253] the stabilization parameter is taken to be

$$\delta_m^{(k)} = \begin{cases} \dfrac{h_m}{2\|\vec{v}_m^{(k)}\|}\left(1 - \dfrac{1}{Pe_m}\right) & \text{if } Pe_m > 1, \\ 0 & \text{if } Pe_m \leqslant 1, \end{cases}$$

where $\|\vec{v}_m^{(k)}\|$ is the Euclidean norm of $\vec{v}^{\,(k)}$ at the patch centroid, h_m is a measure of the patch length in the direction of the wind, and $Pe_m = \|\vec{v}_m^{(k)}\|h_m/(2\nu)$ is the patch Péclet number.

6.2 Preconditioning Forward Stationary Navier–Stokes Equations

In this section we introduce a widely used preconditioner for solving the forward stationary Navier–Stokes equations; this preconditioner makes use of saddle-point theory in conjunction with the commutator argument derived in [92]. These will be the main ingredients for devising our preconditioners.

As discussed in Section 2.10, since the matrix arising from (6.10) is non-symmetric, an optimal preconditioner is given by the matrix $\mathcal{P}_1$ defined in (2.26), with $\Phi = \mathbf{L}^{(k)}$, $\Psi_2 = B$, and $S = B(\mathbf{L}^{(k)})^{-1}B^\top$. Again, we look for approximations of $\mathbf{L}^{(k)}$ and S. As for the forward Stokes equation, the $(1,1)$-block $\mathbf{L}^{(k)}$ can be efficiently approximated by employing a multigrid routine, for example. On the other hand, an approximation $\widehat{S}$ for the Schur complement S is the so called pressure convection–diffusion preconditioner [44, pp. 365–370] (first derived in [92]) for S. The latter is derived by mean of a commutator argument as follows. Consider the convection–diffusion operator $\mathcal{D} = -\nu\nabla^2 + \vec{v}^{\,(k)} \cdot \nabla$ defined on the velocity space as in (6.9), and suppose the analogous operator $\mathcal{D}_p = (-\nu\nabla^2 + \vec{v}^{\,(k)} \cdot \nabla)_p$ on the pressure space is well defined. Suppose also that the commutator defined in (5.8) is small in some sense. Then, discretizing (5.8) with stable finite elements leads to

$$(\mathbf{M}^{-1}\mathbf{L}^{(k)})\mathbf{M}^{-1}B^\top - \mathbf{M}^{-1}B^\top(M_p^{-1}L_p^{(k)}) \approx 0,$$

where $L_p^{(k)} = \nu K_p + N_p^{(k)} + W_p^{(k)}$ is the discretization of $\mathcal{D}_p$ in the finite element

basis for the pressure, with

$$N_p^{(k)} = \left[(\vec{v}^{\,(k)} \cdot \nabla\psi_l, \psi_i)\right],$$
$$W_p^{(k)} = \left[\delta^{(k)}(\kappa_h(\vec{v}^{\,(k)} \cdot \nabla\psi_i), \kappa_h(\vec{v}^{\,(k)} \cdot \nabla\psi_l))\right]$$

the (scalar) convection and stabilization matrices, respectively, in the pressure finite element space. As above, $\kappa_h = \mathrm{Id} - \pi_h$, and $\delta^{(k)}$ as well as π_h are defined as in (6.11). Here, K_p and M_p are the (scalar) mass and stiffness matrices, respectively, in the pressure finite element space. Then, given invertibility of $\mathbf{L}^{(k)}$ and $L_p^{(k)}$, pre- and post-multiplying by $B(\mathbf{L}^{(k)})^{-1}\mathbf{M}$ and $(L_p^{(k)})^{-1}M_p$, the previous expression then gives

$$B\mathbf{M}^{-1}B^\top(L_p^{(k)})^{-1}M_p \approx B(\mathbf{L}^{(k)})^{-1}B^\top.$$

The approximation above is still not practical due to presence of the matrix $B\mathbf{M}^{-1}B^\top$; however, as mentioned in Section 5.2, it can be proved that $K_p \approx B\mathbf{M}^{-1}B^\top$ for problems with enclosed flow [44, pp. 176–177]. Finally, a good approximation of the Schur complement $S = B(\mathbf{L}^{(k)})^{-1}B^\top$ is

$$\widehat{S} = K_p(L_p^{(k)})^{-1}M_p \approx S.$$

Note that in our derivation we have also included the stabilization matrices on the velocity and the pressure space, which was not done in [92].

In the following we are going to use the generalization of the commutator argument presented in Section 5.3 for deriving our efficient preconditioners.

6.3 First-Order Optimality Conditions and Discretization in Time

In this section, we derive the first-order optimality conditions that have to be satisfied at a critical point of (6.1)–(6.2) and of (6.3)–(6.4). We make use of an optimize-then-discretize scheme, and introduce the adjoint variables $\vec{\zeta}$ and μ. We then derive the corresponding Oseen linearized problems, for both stationary and instationary Navier–Stokes control problems, and discretize the conditions so obtained. As for instationary Stokes control problems, we consider employing both backward Euler and Crank–Nicolson schemes in time. In addition, if the initial condition $\vec{v}_0$ is not solenoidal, we can generalize the discretization presented here for backward Euler, while we require a pre-processing in order to write the Oseen iteration for the Crank–Nicolson scheme.

Remark 8. *It is worth noting that, since the problems considered here are nonlinear, first-order optimality conditions are not sufficient on their own for a critical point to be a global minimizer, and indeed second-order optimality conditions should also be tested (see [145, Proposition 2.3] and [175, Sections 4.10 & 5.7]). However, as we are interested in deriving optimal preconditioners for Navier–Stokes control problems, in the following we will only address the numerical solu-*

tion of the first-order optimality conditions, and we will refer to such a solution as a numerical solution of the problem.

6.3.1 Stationary Navier–Stokes Control

Introducing the adjoint velocity $\vec{\zeta}$ and the adjoint pressure μ, we may consider the Lagrangian associated with (6.1)–(6.2) as in [145], and write the Karush–Kuhn–Tucker conditions as:

$$\left\{\begin{array}{rll} -\nu\nabla^2\vec{v}+\vec{v}\cdot\nabla\vec{v}+\nabla p=\frac{1}{\beta}\vec{\zeta}+\vec{f} & \text{in } \Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \text{ state} \\ -\nabla\cdot\vec{v}(x)=0 & \text{in } \Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \text{ equations} \\ \vec{v}(x)=\vec{g}(x) & \text{on } \partial\Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \\ -\nu\nabla^2\vec{\zeta}-\vec{v}\cdot\nabla\vec{\zeta}+(\,\nabla\vec{v}\,)^\top\vec{\zeta}+\nabla\mu\;=\;\vec{v}_d-\vec{v} & \text{in } \Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \text{ adjoint} \\ -\nabla\cdot\vec{\zeta}(x)=0 & \text{in } \Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \text{ equations} \\ \vec{\zeta}(x)=\vec{0} & \text{on } \partial\Omega & \left.\begin{array}{c}\\\\\end{array}\right\} \end{array}\right. \tag{6.12}$$

where we have substituted the gradient equation $\beta\vec{u}-\vec{\zeta}=0$ into the state equation.

Problem (6.12) is a coupled system of non-linear, stationary PDEs. In order to find a numerical solution of (6.12), we need to solve a sequence of linearizations of the system. As in [145], we solve at each step the Oseen approximation as follows. Letting $\vec{v}^{\,(k)}\in V$, $p^{(k)}\in Q$, $\vec{\zeta}^{\,(k)}\in V_0$, $\mu^{(k)}\in Q$ be the current approximations to $\vec{v}$, p, $\vec{\zeta}$, and μ, respectively, with V, V_0, Q defined as in Section 6.1.1, the Oseen iterate is defined as

$$\begin{array}{ll} \vec{v}^{\,(k+1)}=\vec{v}^{\,(k)}+\delta\vec{v}^{\,(k)}, & p^{(k+1)}=p^{(k)}+\delta p^{(k)}, \\ \vec{\zeta}^{\,(k+1)}=\vec{\zeta}^{\,(k)}+\delta\vec{\zeta}^{\,(k)}, & \mu^{(k+1)}=\mu^{(k)}+\delta\mu^{(k)}, \end{array} \tag{6.13}$$

with $\delta\vec{v}^{\,(k)}$, $\delta p^{(k)}$, $\delta\vec{\zeta}^{\,(k)}$, $\delta\mu^{(k)}$ the solutions of the following Oseen problem:

$$\left\{\begin{array}{r} \nu(\nabla\delta\vec{v}^{\,(k)},\nabla\vec{w})+(\vec{v}^{\,(k)}\cdot\nabla\delta\vec{v}^{\,(k)},\vec{w})-(\delta p^{(k)},\nabla\cdot\vec{w})-\frac{1}{\beta}(\delta\vec{\zeta}^{\,(k)},\vec{w})=\vec{R}_1^{\,(k)}, \\ -(q,\nabla\cdot\delta\vec{v}^{\,(k)})=r_1^{(k)}, \\ (\delta\vec{v}^{\,(k)},\vec{w})+\nu(\nabla\delta\vec{\zeta}^{\,(k)},\nabla\vec{w})-(\vec{v}^{\,(k)}\cdot\nabla\delta\vec{\zeta}^{\,(k)},\vec{w})-(\delta\mu^{(k)},\nabla\cdot\vec{w})=\vec{R}_2^{\,(k)}, \\ -(q,\nabla\cdot\delta\vec{\zeta}^{\,(k)})=r_2^{(k)}, \end{array}\right. \tag{6.14}$$

for any $\vec{w}\in V_0$ and $q\in Q$. The residuals $\vec{R}_1^{\,(k)}$, $r_1^{(k)}$, $\vec{R}_2^{\,(k)}$, $r_2^{(k)}$ are given by

$$\left\{\begin{array}{l} \vec{R}_1^{\,(k)}=(\vec{f},\vec{w})-\nu(\nabla\vec{v}^{\,(k)},\nabla\vec{w})-(\vec{v}^{\,(k)}\cdot\nabla\vec{v}^{\,(k)},\vec{w})+(p^{(k)},\nabla\cdot\vec{w})+\frac{1}{\beta}(\vec{\zeta}^{\,(k)},\vec{w}), \\ r_1^{(k)}=(q,\nabla\cdot\vec{v}^{\,(k)}), \\ \vec{R}_2^{\,(k)}=(\vec{v}_d,\vec{w})-(\vec{v}^{\,(k)},\vec{w})-\nu(\nabla\vec{\zeta}^{\,(k)},\nabla\vec{w})+(\vec{v}^{\,(k)}\cdot\nabla\vec{\zeta}^{\,(k)},\vec{w}) \\ \qquad -((\,\nabla\vec{v}^{\,(k)}\,)^\top\vec{\zeta}^{\,(k)},\vec{w})+(\mu^{(k)},\nabla\cdot\vec{w}), \\ r_2^{(k)}=(q,\nabla\cdot\vec{\zeta}^{\,(k)}). \end{array}\right.$$

The Oseen problem (6.14) is posed on the continuous level, so we need to

discretize it in order to obtain a numerical solution of (6.1)–(6.2). Let $\boldsymbol{v}^{(k)} = \{\boldsymbol{v}_i^{(k)}\}_{i=1}^{n_v}$, $\boldsymbol{p}^{(k)} = \{p_i^{(k)}\}_{i=1}^{n_p}$, $\boldsymbol{\zeta}^{(k)} = \{\boldsymbol{\zeta}_i^{(k)}\}_{i=1}^{n_v}$, $\boldsymbol{\mu}^{(k)} = \{\mu_i^{(k)}\}_{i=1}^{n_p}$ be the vectors containing the numerical solutions at the k-th iteration for $\vec{v}^{\,(k)}$, $p^{(k)}$, $\vec{\zeta}^{\,(k)}$, and $\mu^{(k)}$, respectively, that is, $\vec{v}^{\,(k)} \approx \sum_{i=1}^{n_v} \boldsymbol{v}_i^{(k)} \vec{\phi}_i$, $p^{\,(k)} \approx \sum_{i=1}^{n_p} p_i^{(k)} \psi_i$, $\vec{\zeta}^{\,(k)} \approx \sum_{i=1}^{n_v} \boldsymbol{\zeta}_i^{(k)} \vec{\phi}_i$, $\mu^{\,(k)} \approx \sum_{i=1}^{n_p} \mu_i^{(k)} \psi_i$. Then, the (discrete) Oseen iterate is defined as

$$\begin{aligned} \boldsymbol{v}^{\,(k+1)} &= \boldsymbol{v}^{\,(k)} + \boldsymbol{\delta v}^{\,(k)}, \qquad & \boldsymbol{p}^{(k+1)} &= \boldsymbol{p}^{(k)} + \boldsymbol{\delta p}^{(k)}, \\ \boldsymbol{\zeta}^{\,(k+1)} &= \boldsymbol{\zeta}^{\,(k)} + \boldsymbol{\delta \zeta}^{\,(k)}, \qquad & \boldsymbol{\mu}^{(k+1)} &= \boldsymbol{\mu}^{(k)} + \boldsymbol{\delta \mu}^{(k)}, \end{aligned}$$

where

$$\left\{ \begin{aligned} \mathbf{L}^{(k)}\, \boldsymbol{\delta v}^{\,(k)} + B^{\top}\, \boldsymbol{\delta p}^{(k)} - \mathbf{M}_{\beta}\, \boldsymbol{\delta \zeta}^{\,(k)} &= \boldsymbol{R}_1^{(k)}, \\ B \boldsymbol{\delta v}^{\,(k)} &= \boldsymbol{r}_1^{(k)}, \\ \mathbf{M}\, \boldsymbol{\delta v}^{\,(k)} + \mathbf{L}_{\text{adj}}^{(k)}\, \boldsymbol{\delta \zeta}^{\,(k)} + B^{\top} \boldsymbol{\delta \mu}^{(k)} &= \boldsymbol{R}_2^{(k)}, \\ B \boldsymbol{\delta \zeta}^{\,(k)} &= \boldsymbol{r}_2^{(k)}, \end{aligned} \right. \tag{6.15}$$

with $\mathbf{M}_{\beta} = \frac{1}{\beta}\mathbf{M}$, $\mathbf{L}_{\text{adj}}^{(k)} = \nu\mathbf{K} - \mathbf{N}^{(k)} + \mathbf{W}^{(k)}$, and the discrete residuals given by

$$\left\{ \begin{aligned} \boldsymbol{R}_1^{(k)} &= \boldsymbol{f} - \mathbf{L}^{(k)}\, \boldsymbol{v}^{\,(k)} - B^{\top}\, \boldsymbol{p}^{(k)} + \mathbf{M}_{\beta}\, \boldsymbol{\zeta}^{\,(k)}, \\ \boldsymbol{r}_1^{(k)} &= -B \boldsymbol{v}^{\,(k)}, \\ \boldsymbol{R}_2^{(k)} &= \mathbf{M}\, \boldsymbol{v}_d - \mathbf{M}\, \boldsymbol{v}^{\,(k)} - \mathbf{L}_{\text{adj}}^{(k)}\, \boldsymbol{\zeta}^{\,(k)} - B^{\top} \boldsymbol{\mu}^{(k)} - \boldsymbol{\omega}^{(k)}, \\ \boldsymbol{r}_2^{(k)} &= -B \boldsymbol{\zeta}^{\,(k)}. \end{aligned} \right.$$

Here $\boldsymbol{v}_d$ is the vector corresponding to the discretized desired state $\vec{v}_d$, and $\boldsymbol{\omega}^{(k)} = \{\big((\nabla \vec{v}^{\,(k)})^{\top} \vec{\zeta}^{\,(k)}, \vec{\phi}_i\big)\}_{i=1}^{n_v}$. In our tests, the initial guesses $\boldsymbol{v}^{(1)}$ and $\boldsymbol{\zeta}^{(1)}$ for the nonlinear process are the state and adjoint velocity solutions of the KKT conditions for the corresponding stationary Stokes control problem, with discretization given by (6.15) with $\mathbf{L}^{(k)} = \mathbf{L}_{\text{adj}}^{(k)} = \mathbf{K}$, and residuals $\boldsymbol{R}_1^{(k)} = \boldsymbol{f}$, $\boldsymbol{R}_2^{(k)} = \mathbf{M}\, \boldsymbol{v}_d$, $\boldsymbol{r}_1^{(k)} = \boldsymbol{r}_2^{(k)} = \mathbf{0}$. Note that the right-hand side may also take into account boundary conditions (as done in our implementation).

In matrix form, we rewrite system (6.15) as

$$\underbrace{\begin{bmatrix} \Phi_{\text{S}}^{(k)} & \Psi_{\text{S}}^{\top} \\ \Psi_{\text{S}} & -\Theta_{\text{S}} \end{bmatrix}}_{\mathcal{A}_{\text{S}}^{(k)}} \begin{bmatrix} \boldsymbol{\delta v}^{(k)} \\ \boldsymbol{\delta \zeta}^{(k)} \\ \boldsymbol{\delta \mu}^{(k)} \\ \boldsymbol{\delta p}^{(k)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_2^{(k)} \\ \boldsymbol{R}_1^{(k)} \\ \boldsymbol{r}_1^{(k)} \\ \boldsymbol{r}_2^{(k)} \end{bmatrix}, \tag{6.16}$$

where

$$\Phi_{\text{S}}^{(k)} = \begin{bmatrix} \mathbf{M} & \mathbf{L}_{\text{adj}}^{(k)} \\ \mathbf{L}^{(k)} & -\mathbf{M}_{\beta} \end{bmatrix}, \quad \Psi_{\text{S}} = \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix}, \quad \Theta_{\text{S}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{6.17}$$

The matrix $\mathcal{A}_{\text{S}}^{(k)}$ is of saddle-point type; however, since the incompressibility constraints $\nabla \cdot \vec{v} = 0$ are not solved exactly, $\Phi_{\text{S}}^{(k)}$ is not symmetric in general. It is worth noting that the matrix $\Phi_{\text{S}}^{(k)}$ represents the discrete optimality conditions for a stationary (vector) convection–diffusion control problem. This observation

will be used when deriving our preconditioners.

6.3.2 Instationary Navier–Stokes Control

We now state the KKT conditions for the instationary problem (6.3)–(6.4). As before, introducing the adjoint variables $\vec{\zeta}$ and μ, we consider the Lagrangian associated to (6.3)–(6.4) as in [175, p. 318]. Then, by deriving the KKT conditions and substituting the gradient equation $\beta\vec{u} - \vec{\zeta} = 0$ into the state equation, the solution of (6.3)–(6.4) satisfies:

$$\begin{cases} \frac{\partial \vec{v}}{\partial t} - \nu\nabla^2\vec{v} + \vec{v}\cdot\nabla\vec{v} + \nabla p = \frac{1}{\beta}\vec{\zeta} + \vec{f} & \text{in } \Omega\times(0,t_f), \\ -\nabla\cdot\vec{v}(x,t) = 0 & \text{in } \Omega\times(0,t_f), \\ \vec{v}(x,t) = \vec{g}(x,t) & \text{on } \partial\Omega\times(0,t_f), \\ \vec{v}(x,0) = \vec{v}_0(x) & \text{in } \Omega, \\ -\frac{\partial\vec{\zeta}}{\partial t} - \nu\nabla^2\vec{\zeta} - \vec{v}\cdot\nabla\vec{\zeta} + (\,\nabla\vec{v}\,)^\top\vec{\zeta} + \nabla\mu \;=\; \vec{v}_d - \vec{v} & \text{in } \Omega\times(0,t_f), \\ -\nabla\cdot\vec{\zeta}(x,t) = 0 & \text{in } \Omega\times(0,t_f), \\ \vec{\zeta}(x,t) = \vec{0} & \text{on } \partial\Omega\times(0,t_f), \\ \vec{\zeta}(x,t_f) = \vec{0} & \text{in } \Omega. \end{cases} \tag{6.18}$$

Problem (6.18) is a coupled system of non-linear, instationary PDEs. In order to find a numerical solution of (6.18), as for the stationary case we take an Oseen linearization. Before showing the linearization adopted, we would like to show how to derive the optimality conditions above, observing that the optimality conditions for stationary Navier–Stokes control problem can be derived from those by neglecting the time derivative. Similarly, we can recover the first-order optimality conditions for Stokes control by neglecting the non-linear term $\vec{v}\cdot\nabla\vec{v}$. We will follow the work in [175, p. 318], and consider only the case of $d = 2$ (that is, $\Omega\subset\mathbb{R}^2$).

We introduce an adjoint variable for each constraint in (6.4), and consider the Lagrangian associated to (6.3)–(6.4)

$$\begin{aligned} \mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu) = {} & J_{\mathrm{I}}(\vec{v},\vec{u}) \\ & + \int_0^{t_f}\!\!\int_\Omega \left(\frac{\partial\vec{v}}{\partial t} - \nu\nabla^2\vec{v} + \vec{v}\cdot\nabla\vec{v} + \nabla p - \vec{u} - \vec{f}\right)\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t \\ & - \int_0^{t_f}\!\!\int_\Omega (\mu, \nabla\cdot\vec{v}\,)\,\mathrm{d}\Omega\,\mathrm{d}t + \int_0^{t_f}\!\!\int_\Omega (\vec{v}-\vec{g}\,)\cdot\vec{\zeta}_{\partial\Omega}\,\mathrm{d}\Omega\,\mathrm{d}t \\ & + \int_\Omega (\vec{v}-\vec{v}_0\,)\cdot\vec{\zeta}_0\,\mathrm{d}\Omega. \end{aligned}$$

Following the work in Section 1.3.1, we may easily derive that the Fréchet derivative of $\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)$ with respect to u leads to the gradient equation $\beta\vec{u} - \vec{\zeta}_\Omega = 0$, while the Fréchet derivatives of $\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)$ with

respect to $\vec{\zeta}_\Omega$, μ, $\vec{\zeta}_{\partial\Omega}$, and $\vec{\zeta}_0$ lead to the state equations

$$\begin{cases} \frac{\partial \vec{v}}{\partial t} - \nu\nabla^2\vec{v} + \vec{v}\cdot\nabla\vec{v} + \nabla p = \vec{u} + \vec{f} & \text{in } \Omega\times(0,t_f), \\ -\nabla\cdot\vec{v}(x,t) = 0 & \text{in } \Omega\times(0,t_f), \\ \vec{v}(x,t) = \vec{g}(x,t) & \text{on } \partial\Omega\times(0,t_f), \\ \vec{v}(x,0) = \vec{v}_0(x) & \text{in } \Omega. \end{cases}$$

Let us now consider the Fréchet derivative of $\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)$ with respect to p. We take a generic direction q in an appropriate Hilbert space, and then write the Fréchet derivative of the term

$$\int_0^{t_f}\!\!\int_\Omega \nabla(p+q)\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t = \int_0^{t_f}\!\!\int_\Omega \nabla p\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t + \int_0^{t_f}\!\!\int_\Omega \nabla q\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t.$$

Applying the Divergence Theorem to the term $\displaystyle\int_0^{t_f}\!\!\int_\Omega \nabla q\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t$, we can write

$$\int_0^{t_f}\!\!\int_\Omega \nabla q\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t = -\int_0^{t_f}\!\!\int_\Omega q\,\nabla\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t + \int_0^{t_f}\!\!\int_{\partial\Omega} q\,\vec{\zeta}_\Omega\cdot\boldsymbol{n}\,\mathrm{d}s\,\mathrm{d}t,$$

where $\boldsymbol{n}$ denotes the (outer) unit normal vector. Then, one can easily derive that the Fréchet derivative of $\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)$ with respect to p is given by

$$\mathrm{d}_p\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)q = -\int_0^{t_f}\!\!\int_\Omega q\,\nabla\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t + \int_0^{t_f}\!\!\int_{\partial\Omega} q\,\vec{\zeta}_\Omega\cdot\boldsymbol{n}\,\mathrm{d}s\,\mathrm{d}t.$$

As we wish that $\mathrm{d}_p\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)q = 0$ for any appropriate choice of q, in particular by choosing q in $C_0^\infty(0,t_f;\Omega)$ we can infer that

$$-\int_0^{t_f}\!\!\int_\Omega q\,\nabla\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t = 0,$$

where $C_0^\infty(0,t_f;\Omega)$ is the class of infinitely differentiable functions on $(0,t_f)\times\Omega$ that are equal to 0 on the boundary $\partial\Omega$. From the latter expression we recover the incompressibility constraint on the adjoint variable $\vec{\zeta}$ in (6.18). In addition, since the previous expression has to be equal to 0 for any q in the appropriate Hilbert space, from $\mathrm{d}_p\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)q = 0$ we may also infer that $\vec{\zeta}_\Omega\cdot\boldsymbol{n} = 0$. However, we do not include the latter condition in (6.18), since as we will see (and as we may expect from the control problems described in the previous chapters) the adjoint variable $\vec{\zeta}_\Omega$ has to be 0 on $\partial\Omega$, for all times $t\in(0,t_f)$.

We now have to derive the Fréchet derivative of $\mathcal{L}(\vec{v},p,\vec{u},\vec{\zeta}_\Omega,\vec{\zeta}_{\partial\Omega},\vec{\zeta}_0,\mu)$ with respect to $\vec{v}$, and we will analyse only two terms, as we have already derived the Fréchet derivatives of the remaining ones in Section 3.2. We will start with finding the Fréchet derivative of the term

$$\int_0^{t_f}\!\!\int_\Omega (\mu,\nabla\cdot\vec{v}\,)\,\mathrm{d}\Omega\,\mathrm{d}t.$$

We take a generic direction $\vec{w}$ in an appropriate Hilbert space, and then consider the quantity

$$\int_0^{t_f}\int_\Omega (\mu, \nabla \cdot (\vec{v} + \vec{w})\,)\,\mathrm{d}\Omega\,\mathrm{d}t.$$

Then, by working as in Section 1.3.1 it is easy to prove that the Fréchet derivative of $\int_0^{t_f}\int_\Omega (\mu, \nabla \cdot \vec{v}\,)\,\mathrm{d}\Omega\,\mathrm{d}t$ with respect to $\vec{v}$ is given by

$$\int_0^{t_f}\int_\Omega (\mu, \nabla \cdot \vec{w}\,)\,\mathrm{d}\Omega\,\mathrm{d}t.$$

We now focus on the non-linear term

$$\int_0^{t_f}\int_\Omega (\vec{v} \cdot \nabla\vec{v}\,) \cdot \vec{\zeta}_\Omega\,\mathrm{d}\Omega\,\mathrm{d}t.$$

As above, we consider a generic direction $\vec{w}$ in an appropriate Hilbert space, and then consider the following quantity:

$$\int_0^{t_f}\int_\Omega \Big((\vec{v} + \vec{w}\,) \cdot \nabla(\vec{v} + \vec{w}\,)\Big) \cdot \vec{\zeta}_\Omega\,\mathrm{d}\Omega\,\mathrm{d}t.$$

Then, by expanding the previous quantity and working as in Section 1.3.1, we can derive that the Fréchet derivatives of $\int_0^{t_f}\int_\Omega (\vec{v} \cdot \nabla\vec{v}\,) \cdot \vec{\zeta}_\Omega\,\mathrm{d}\Omega\,\mathrm{d}t$ with respect to $\vec{v}$ is given by

$$\int_0^{t_f}\int_\Omega \Big(\vec{v} \cdot \nabla\vec{w}\Big) \cdot \vec{\zeta}_\Omega\,\mathrm{d}\Omega\,\mathrm{d}t + \int_0^{t_f}\int_\Omega \Big(\vec{w} \cdot \nabla\vec{v}\Big) \cdot \vec{\zeta}_\Omega\,\mathrm{d}\Omega\,\mathrm{d}t.$$

In order to derive the optimality conditions (6.18), we observe that

$$\vec{w} \cdot \nabla\vec{v} = \left[\; w_1\,\frac{\partial v_1}{\partial x_1} + w_2\,\frac{\partial v_1}{\partial x_2}, \quad w_1\,\frac{\partial v_2}{\partial x_1} + w_2\,\frac{\partial v_2}{\partial x_2} \;\right],$$

and therefore we have that

$$\begin{aligned}\int_0^{t_f}\int_\Omega \Big(\vec{w} \cdot \nabla\vec{v}\Big) \cdot \vec{\zeta}_\Omega\,\mathrm{d}\Omega\,\mathrm{d}t &= \int_0^{t_f}\int_\Omega \sum_{i,l=1}^{2} w_i\,\frac{\partial v_l}{\partial x_i}\,\zeta_{\Omega,l}\,\mathrm{d}\Omega\,\mathrm{d}t \\ &= \int_0^{t_f}\int_\Omega \Big(\big[\;\nabla v_1 \quad \nabla v_2\;\big]\,\zeta_\Omega\Big) \cdot \vec{w}\,\mathrm{d}\Omega\,\mathrm{d}t.\end{aligned}$$

In addition, we use the fact that the trilinear form

$$c(\vec{v},\,\vec{w},\,\vec{\zeta}_\Omega\,) := \int_0^{t_f}\int_\Omega \Big(\vec{v} \cdot \nabla\vec{w}\Big) \cdot \vec{\zeta}_\Omega\,\mathrm{d}\Omega\,\mathrm{d}t$$

is skew-symmetric if $\vec{\zeta}_\Omega = \vec{0}$ on $\partial\Omega$[7]. Specifically, for this $\vec{\zeta}_\Omega$ we have

$$c(\vec{v},\, \vec{w},\, \vec{\zeta}_\Omega\,) = -c(\vec{v},\, \vec{\zeta}_\Omega,\, \vec{w}\,).$$

From here, we can rewrite the Fréchet derivatives of $\int_0^{t_f}\int_\Omega (\vec{v}\cdot\nabla\vec{v}\,)\cdot\vec{\zeta}_\Omega \,\mathrm{d}\Omega\,\mathrm{d}t$ with respect to $\vec{v}$ as follows:

$$-\int_0^{t_f}\int_\Omega \left(\vec{v}\cdot\nabla\vec{\zeta}_\Omega\right)\cdot\vec{w}\,\mathrm{d}\Omega\,\mathrm{d}t + \int_0^{t_f}\int_\Omega \left((\nabla\vec{v})^\top\vec{\zeta}_\Omega\right)\cdot\vec{w}\,\mathrm{d}\Omega\,\mathrm{d}t,$$

where $\nabla\vec{v} = \begin{bmatrix} \nabla v_1 & \nabla v_2 \end{bmatrix}^\top$. We have thus recovered the convection and the "mixed" terms of the adjoint equation in (6.18).

Finally, we can employ the same strategy as in Section 3.2 for all the other terms, and find out that the Fréchet derivatives of $\mathcal{L}(\vec{v}, p, \vec{u}, \vec{\zeta}_\Omega, \vec{\zeta}_{\partial\Omega}, \vec{\zeta}_0, \mu)$ with respect to $\vec{v}$ is given by

$$\begin{aligned}
\mathrm{d}_{\vec{v}}\mathcal{L}(\vec{v}, p, \vec{u}, \vec{\zeta}_\Omega, \vec{\zeta}_{\partial\Omega}, \vec{\zeta}_0, \mu) = &\int_0^{t_f}\int_\Omega \left(\vec{v} - \vec{v}_d - \frac{\partial\vec{\zeta}_\Omega}{\partial t} - \nu\nabla^2\vec{\zeta}_\Omega - \vec{v}\cdot\nabla\vec{\zeta}_\Omega\right)\cdot\vec{w}\,\mathrm{d}\Omega\,\mathrm{d}t \\
&+\int_0^{t_f}\int_\Omega \left((\nabla\vec{v})^\top\vec{\zeta}_\Omega + \nabla\mu\right)\cdot\vec{w}\,\mathrm{d}\Omega\,\mathrm{d}t + \int_0^{t_f}\int_{\partial\Omega}\vec{\zeta}_{\partial\Omega}\cdot\vec{w}\,\mathrm{d}s\mathrm{d}t \\
&- \int_0^{t_f}\int_{\partial\Omega}\vec{\zeta}_\Omega\frac{\partial\vec{w}}{\partial\vec{n}}\,\mathrm{d}s\mathrm{d}t + \int_0^{t_f}\int_{\partial\Omega}\frac{\partial\vec{\zeta}_\Omega}{\partial\vec{n}}\vec{w}\,\mathrm{d}s\mathrm{d}t \\
&+ (\vec{\zeta}_\Omega(x,t_f),\vec{w}(x,t_f)) - (\vec{\zeta}_\Omega(x,0),\vec{w}(x,0)) + (\vec{\zeta}_0,\vec{w}).
\end{aligned}$$

Then, we can derive the optimality conditions (6.18) by setting the previous expression equal to zero and choosing $\vec{w}$ appropriately, as done in Section 1.3.1 and in Section 3.2.

We can now derive the Oseen linearization for the system (6.18). Let $\vec{v}^{\,(k)} \in \bar{V}$, $p^{(k)} \in \bar{Q}$, $\vec{\zeta}^{\,(k)} \in \bar{V}_0$, $\mu^{(k)} \in \bar{Q}$ be the current approximation to $\vec{v}$, p, $\vec{\zeta}$, μ, with

$$\begin{aligned}
\bar{V} := \{&\vec{v} \in L^2(0,t_f;\mathcal{H}^1(\Omega)^d) \mid \tfrac{\partial\vec{v}}{\partial t}(\cdot,t) \in L^2(0,t_f;\mathcal{H}^{-1}(\Omega)^d) \text{ for a.e. } t\in(0,t_f), \\
&\vec{v} = \vec{g} \text{ on } \partial\Omega,\ \vec{v}(x,0) = \vec{v}_0(x)\}, \\
\bar{Q} := {}& L^2(0,t_f;L^2(\Omega)),
\end{aligned}$$

and $\bar{V}_0$ the corresponding space for the adjoint velocity (see [175, pp. 315–321] and [86, pp. 88–95], for instance, for the case $d = 2$). Then, the Oseen iterate is of the form (6.13), with $\delta\vec{v}^{\,(k)}$, $\delta p^{(k)}$, $\delta\vec{\zeta}^{\,(k)}$, $\delta\mu^{(k)}$ the solution of:

$$\begin{cases}
(\frac{\partial}{\partial t}\delta\vec{v}^{\,(k)},\vec{w}) + \nu(\nabla\delta\vec{v}^{\,(k)},\nabla\vec{w}) + (\vec{v}^{(k)}\cdot\nabla\delta\vec{v}^{\,(k)},\vec{w}) - (\delta p^{(k)},\nabla\cdot\vec{w}) - \frac{1}{\beta}(\delta\vec{\zeta}^{\,(k)},\vec{w}) = \vec{R}_1^{(k)}, \\
\qquad\qquad -(q,\nabla\cdot\delta\vec{v}^{\,(k)}) = r_1^{(k)}, \\
-(\frac{\partial}{\partial t}\delta\vec{\zeta}^{\,(k)},\vec{w}) + \nu(\nabla\delta\vec{\zeta}^{\,(k)},\nabla\vec{w}) - (\vec{v}^{(k)}\cdot\nabla\delta\vec{\zeta}^{\,(k)},\vec{w}) - (\delta\mu^{(k)},\nabla\cdot\vec{w}) + (\delta\vec{v}^{\,(k)},\vec{w}) = \vec{R}_2^{(k)}, \\
\qquad\qquad -(q,\nabla\cdot\delta\vec{\zeta}^{\,(k)}) = r_2^{(k)},
\end{cases} \tag{6.19}$$

[7] We recall that $\nabla\cdot\vec{v} = 0$ to derive this.

for any $\vec{w} \in V_0$ and $q \in Q$. The residuals $\vec{R}_1^{(k)}$, $r_1^{(k)}$, $\vec{R}_2^{(k)}$, $r_2^{(k)}$ are given by

$$\begin{cases} \vec{R}_1^{(k)} = (\vec{f}, \vec{w}) - (\frac{\partial}{\partial t}\vec{v}^{\,(k)}, \vec{w}) - \nu(\nabla\vec{v}^{\,(k)}, \nabla\vec{w}) - (\vec{v}^{\,(k)} \cdot \nabla\vec{v}^{\,(k)}, \vec{w}) \\ \qquad\qquad +(p^{(k)}, \nabla \cdot \vec{w}) + \frac{1}{\beta}\,(\vec{\zeta}^{\,(k)}, \vec{w}), \\ r_1^{(k)} = (q, \nabla \cdot \vec{v}^{\,(k)}), \\ \vec{R}_2^{(k)} = (\vec{v}_d, \vec{w}) - (\vec{v}^{\,(k)}, \vec{w}) + (\frac{\partial}{\partial t}\vec{\zeta}^{\,(k)}, \vec{w}) - \nu(\nabla\vec{\zeta}^{\,(k)}, \nabla\vec{w}) \\ \qquad\qquad +(\vec{v}^{\,(k)} \cdot \nabla\vec{\zeta}^{\,(k)}, \vec{w}) - ((\nabla\vec{v}^{\,(k)})^\top\vec{\zeta}^{\,(k)}, \vec{w}) + (\mu^{(k)}, \nabla \cdot \vec{w}), \\ r_2^{(k)} = (q, \nabla \cdot \vec{\zeta}^{\,(k)}). \end{cases} \tag{6.20}$$

Note that with this notation $\delta\vec{v}^{\,(k)}(x,0) = \delta\vec{\zeta}^{\,(k)}(x,t_f) = \vec{0}$ in Ω, and $\delta\vec{v}^{\,(k)}(x,t) = \delta\vec{\zeta}^{\,(k)}(x,t) = \vec{0}$ on $\partial\Omega \times (0,t_f)$.

Equations (6.19) constitute an Oseen approximation for instationary Navier–Stokes control, involving a coupled system of instationary convection–diffusion equations and divergence-free conditions. In order to discretize them, we employ either backward Euler or Crank–Nicolson in time. Further, in order to solve (6.19) we need to choose an initial guess $\boldsymbol{v}^{(1)}$ and $\boldsymbol{\zeta}^{(1)}$ for the state and the adjoint velocities and then iteratively solve a sequence of linearized problems. In our tests, $\boldsymbol{v}^{(1)}$ and $\boldsymbol{\zeta}^{(1)}$ are again the (velocity) solutions of the KKT conditions for the corresponding Stokes control problem.

We now derive the linear systems resulting from the time-stepping schemes. In order to simplify the notations, we will employ the $\bar{n} \times \bar{n}$ matrices $I_{\bar{n},1}$, $I_{\bar{n},2}$, $I_{\bar{n},3}$, and $I_{\bar{n},4}$ defined in (5.18).

Backward Euler for Instationary Navier–Stokes Control

In this section we introduce the backward Euler scheme for approximating (6.19)–(6.20), and then derive the resulting linear system. We discretize the interval $(0,t_f)$ into n_t subintervals of length $\tau = \frac{t_f}{n_t}$, denoting the grid points as $t_n = n\tau$, for $n = 0,1,\ldots,n_t$, and approximate all the functions as for instationary Stokes control with backward Euler in time. Specifically, our approximations of the solutions at the k-th step of the non-linear solver are given by $\boldsymbol{v}_n^{(k)} \approx \vec{v}(x,t_n)$, $\boldsymbol{\zeta}_n^{(k)} \approx \vec{\zeta}(x,t_n)$, for $n = 0,1,\ldots,n_t$, and $\boldsymbol{p}_{n+1}^{(k)} \approx p(x,t_{n+1})$, $\boldsymbol{\mu}_n^{(k)} \approx \mu(x,t_n)$, for $n = 0,1,\ldots,n_t-1$, for all $x \in \Omega$. We also introduce the following finite element matrices:

$$\mathbf{L}_n^{(k)} = \tau(\nu\mathbf{K} + \mathbf{N}_n^{(k)} + \mathbf{W}_n^{(k)}) + \mathbf{M}, \qquad \mathbf{T}_n^{(k)} = \tau(\nu\mathbf{K} - \mathbf{N}_n^{(k)} + \mathbf{W}_n^{(k)}) + \mathbf{M},$$
$$\bar{\mathbf{M}}^{\mathrm{BE}} = \tau\mathbf{M}, \qquad \bar{\mathbf{M}}_\beta^{\mathrm{BE}} = \tfrac{\tau}{\beta}\mathbf{M}, \qquad \bar{B} = \tau B,$$

where $\mathbf{W}_n^{(k)}$ is the stabilization matrix related to $\vec{v}_n^{\,(k)}$, and $\mathbf{N}_n^{(k)} = [(\vec{v}_n^{\,(k)} \cdot \nabla\vec{\phi}_l, \vec{\phi}_i)]$, with $\vec{v}_n^{\,(k)}$ the approximation to $\vec{v}$ at time t_n, at the k-th Oseen iteration. Note that the superscripts of $\mathbf{L}_0^{(k)}$, $\mathbf{T}_0^{(k)}$ are superfluous, as the initial condition on $\vec{v}$ is fixed; however we keep them for consistency. We then write the discrete Oseen iterate as

$$\begin{aligned} &\boldsymbol{v}_n^{(k+1)} = \boldsymbol{v}_n^{(k)} + \boldsymbol{\delta v}_n^{(k)}, \quad &&\boldsymbol{\zeta}_n^{(k+1)} = \boldsymbol{\zeta}_n^{(k)} + \boldsymbol{\delta\zeta}_n^{(k)}, \quad && n = 0,1,\ldots,n_t, \\ &\boldsymbol{p}_{n+1}^{(k+1)} = \boldsymbol{p}_{n+1}^{(k)} + \boldsymbol{\delta p}_{n+1}^{(k)}, \quad &&\boldsymbol{\mu}_n^{(k+1)} = \boldsymbol{\mu}_n^{(k)} + \boldsymbol{\delta\mu}_n^{(k)}, \quad && n = 0,1,\ldots,n_t-1, \end{aligned}$$

with $\boldsymbol{\delta v}_n^{(k)}$, $\boldsymbol{\delta \zeta}_n^{(k)}$, $\boldsymbol{\delta p}_n^{(k)}$, $\boldsymbol{\delta \mu}_n^{(k)}$ the solutions of the following discretization of (6.19):

$$\begin{cases} \bar{\mathbf{M}}^{\mathrm{BE}}\boldsymbol{\delta v}_n^{(k)} + \mathbf{T}_n^{(k)}\boldsymbol{\delta \zeta}_n^{(k)} - \mathbf{M}\,\boldsymbol{\delta \zeta}_{n+1}^{(k)} + \bar{B}^\top \boldsymbol{\delta \mu}_n^{(k)} = \boldsymbol{R}_{2,n}^{(k)}, \\ -\mathbf{M}\,\boldsymbol{\delta v}_n^{(k)} + \mathbf{L}_{n+1}^{(k)}\boldsymbol{\delta v}_{n+1}^{(k)} + \bar{B}^\top \boldsymbol{\delta p}_{n+1}^{(k)} - \bar{\mathbf{M}}_\beta^{\mathrm{BE}}\boldsymbol{\delta \zeta}_{n+1}^{(k)} = \boldsymbol{R}_{1,n}^{(k)}, \\ B\,\boldsymbol{\delta v}_{n+1}^{(k)} = \boldsymbol{r}_{1,n+1}^{(k)}, \\ B\,\boldsymbol{\delta \zeta}_n^{(k)} = \boldsymbol{r}_{2,n}^{(k)}, \end{cases} \tag{6.21}$$

for $n = 0, 1, ..., n_t - 1$, with $\boldsymbol{\delta v}_0^{(k)} = \mathbf{0}$, $\boldsymbol{\delta \zeta}_{n_t}^{(k)} = \mathbf{0}$. The discretized residuals are given by

$$\begin{cases} \boldsymbol{R}_{1,n}^{(k)} = \tau \boldsymbol{f}^{n+1} + \mathbf{M}\boldsymbol{v}_n^{(k)} - \mathbf{L}_{n+1}^{(k)}\boldsymbol{v}_{n+1}^{(k)} - \bar{B}^\top \boldsymbol{p}_{n+1}^{(k)} + \bar{\mathbf{M}}_\beta^{\mathrm{BE}}\boldsymbol{\zeta}_{n+1}^{(k)}, \\ \boldsymbol{r}_{1,n+1}^{(k)} = -B\,\boldsymbol{v}_{n+1}^{(k)}, \\ \boldsymbol{R}_{2,n}^{(k)} = \bar{\mathbf{M}}^{\mathrm{BE}}\boldsymbol{v}_d^n - \bar{\mathbf{M}}^{\mathrm{BE}}\boldsymbol{v}_n^{(k)} - \mathbf{T}_n^{(k)}\boldsymbol{\zeta}_n^{(k)} + \mathbf{M}\boldsymbol{\zeta}_{n+1}^{(k)} - \bar{B}^\top \boldsymbol{\mu}_n^{(k)} - \tau \boldsymbol{\omega}_n^{(k)}, \\ \boldsymbol{r}_{2,n}^{(k)} = -B\,\boldsymbol{\zeta}_n^{(k)}, \end{cases} \tag{6.22}$$

where $\boldsymbol{f}^{n+1} = \{(\vec{f}(x, t_{n+1}), \vec{\phi}_i)\}_{i=1}^{n_v}$, and $\boldsymbol{\omega}_n^{(k)} = \{\big((\nabla \vec{v}_n^{(k)})^\top \vec{\zeta}_n^{(k)}, \vec{\phi}_i\big)\}_{i=1}^{n_v}$, for $n = 0, 1, \ldots, n_t - 1$. Note that the non-linear residuals $\boldsymbol{R}_{1,0}^{(k)}$, $\boldsymbol{R}_{1,n_t-1}^{(k)}$, $\boldsymbol{R}_{2,0}^{(k)}$, $\boldsymbol{R}_{2,n_t-1}^{(k)}$ in (6.22) take into account the initial and the final conditions on $\vec{v}$ and $\vec{\zeta}$.

Even if the incompressibility constraints $B\,\boldsymbol{\delta v}_{n+1}^{(k)} = \mathbf{0}$ are solved exactly, for $n = 0, 1, \ldots, n_t - 1$, at each Oseen iteration the system described in (6.21) is not symmetric, due the conditions $\boldsymbol{\delta v}_0^{(k)} = \mathbf{0}$ and $\boldsymbol{\delta \zeta}_{n_t}^{(k)} = \mathbf{0}$. However, we work as for instationary Stokes control problems solved with backward Euler in time, and employ a solenoidal projection of the form (5.20). However, within the projection we include also a convection and a stabilization matrix. Specifically, given a vector $\bar{\boldsymbol{b}}$, we define its solenoidal projection at time $t = 0$ as $\boldsymbol{b}$, with

$$\begin{cases} \mathbf{L}_0^{(k)}\,\boldsymbol{b} + \bar{B}^\top \bar{\boldsymbol{p}} = \mathbf{L}_{\bar{\boldsymbol{b}}}\,\bar{\boldsymbol{b}}, \\ B\,\boldsymbol{b} = \mathbf{0}, \end{cases} \tag{6.23}$$

with $\mathbf{L}_{\bar{\boldsymbol{b}}} = \tau(\nu\mathbf{K} + \mathbf{N}_{\bar{\boldsymbol{b}}} + \mathbf{W}_{\bar{\boldsymbol{b}}}) + \mathbf{M}$, $\mathbf{N}_{\bar{\boldsymbol{b}}}$ and $\mathbf{W}_{\bar{\boldsymbol{b}}}$ being the vector-convection and stabilization matrices related to $\bar{\boldsymbol{b}}$. Similarly, we define the solenoidal projection of a vector $\bar{\boldsymbol{b}}$ at time $t = t_f$ as the vector $\boldsymbol{b}$ solution of (6.23), with the matrix $\mathbf{L}_0^{(k)}$ replaced by $\mathbf{T}_{n_t}^{(k)}$.

We apply the previous projections to the initial and final time conditions $\boldsymbol{\delta v}_0^{(k)} = \mathbf{0}$ and $\boldsymbol{\delta \zeta}_{n_t}^{(k)} = \mathbf{0}$. Then, by multiplying the incompressibility conditions by τ, the linear system of (6.21) can be rewritten as

$$\underbrace{\begin{bmatrix} \Phi_{\mathrm{BE}}^{(k)} & (\Psi_{\mathrm{BE}})^\top \\ \Psi_{\mathrm{BE}} & -\Theta_{\mathrm{BE}} \end{bmatrix}}_{\mathcal{A}_{\mathrm{BE}}^{(k)}} \begin{bmatrix} \boldsymbol{\delta v}^{(k)} \\ \boldsymbol{\delta \zeta}^{(k)} \\ \boldsymbol{\delta \mu}^{(k)} \\ \boldsymbol{\delta p}^{(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^{(k)} \\ \mathbf{b}_2^{(k)} \\ \mathbf{b}_3^{(k)} \\ \mathbf{b}_4^{(k)} \end{bmatrix}, \tag{6.24}$$

where the right-hand side accounts for the non-linear residual. Further,

$$\Phi_{\mathrm{BE}}^{(k)} = \begin{bmatrix} \mathcal{M}^{\mathrm{BE}} & \mathcal{L}_1^{\mathrm{BE},(k)} \\ \mathcal{L}_2^{\mathrm{BE},(k)} & -\mathcal{M}_\beta^{\mathrm{BE}} \end{bmatrix}, \quad \Psi_{\mathrm{BE}} = \begin{bmatrix} \mathcal{B}^{\mathrm{BE}} & 0 \\ 0 & \mathcal{B}^{\mathrm{BE}} \end{bmatrix}, \quad \Theta_{\mathrm{BE}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad (6.25)$$

with $\mathcal{M}^{\mathrm{BE}} = I_{n_t+1,1} \otimes \bar{\mathbf{M}}^{\mathrm{BE}}$, $\mathcal{M}_\beta^{\mathrm{BE}} = I_{n_t+1,2} \otimes \bar{\mathbf{M}}_\beta^{\mathrm{BE}}$, $\mathcal{B}^{\mathrm{BE}} = I_{n_t+1} \otimes \bar{B}$, and

$$\mathcal{L}_1^{\mathrm{BE},(k)} = \begin{bmatrix} \mathbf{T}_0^{(k)} & -\mathbf{M} & & \\ & \ddots & \ddots & \\ & & \mathbf{T}_{n_t-1}^{(k)} & -\mathbf{M} \\ & & & \mathbf{T}_{n_t}^{(k)} \end{bmatrix}, \quad \mathcal{L}_2^{\mathrm{BE},(k)} = \begin{bmatrix} \mathbf{L}_0^{(k)} & & & \\ -\mathbf{M} & \mathbf{L}_1^{(k)} & & \\ & \ddots & \ddots & \\ & & -\mathbf{M} & \mathbf{L}_{n_t}^{(k)} \end{bmatrix}.$$

Note that, in the case of the incompressibility conditions not being solved exactly, $\mathcal{L}_1^{\mathrm{BE},(k)} \neq (\mathcal{L}_2^{\mathrm{BE},(k)})^\top$; however, the system is symmetric if they are solved exactly.

We note that we can relax the incompressibility assumptions on $\vec{v}_0$ and modify the discretization of the Oseen problem (6.19)–(6.20) as follows. Suppose that $\vec{v}_0$ is not solenoidal. Then, for the first backward Euler step in (6.21) we can rewrite (6.23) as

$$\begin{cases} \mathbf{L}_0^{(k)}\, \boldsymbol{\delta v}_0^{(k)} + \bar{B}^\top \boldsymbol{\delta p}_0^{(k)} = \boldsymbol{R}_{1,-1}^{(k)}, \\ B\, \boldsymbol{\delta v}_0^{(k)} = \boldsymbol{r}_{1,0}^{(k)}, \end{cases}$$

where, given $\bar{\boldsymbol{v}}_0$ as an appropriate discretization of $\vec{v}_0$,

$$\begin{cases} \boldsymbol{R}_{1,-1}^{(k)} = \overline{\mathbf{L}}_0\, \bar{\boldsymbol{v}}_0 - \mathbf{L}_0^{(k)} \boldsymbol{v}_0^{(k)} - \bar{B}^\top \boldsymbol{p}_0^{(k)}, \\ \boldsymbol{r}_{1,0}^{(k)} = -B\, \boldsymbol{v}_0^{(k)}. \end{cases}$$

Here, $\overline{\mathbf{L}}_0 = \tau(\nu\mathbf{K} + \overline{\mathbf{N}}_0 + \overline{\mathbf{W}}_0) + \mathbf{M}$, where $\overline{\mathbf{N}}_0$ and $\overline{\mathbf{W}}_0$ are the vector-convection and stabilization matrices related to $\bar{\boldsymbol{v}}_0$, and the rest of the non-linear residuals are defined as in (6.22). Note that in this case we cannot substitute $\mathbf{M}\boldsymbol{v}_0^{(k)} = \mathbf{M}\bar{\boldsymbol{v}}_0$ into the non-linear residuals as $\bar{\boldsymbol{v}}_0$ is not incompressible. We note also that for $k = 0$ (meaning $\boldsymbol{v}_0^{(0)} = \mathbf{0}$) and with $\nu = 1$ from the above step, with $\overline{\mathbf{L}}_0 = \mathbf{L}_0^{(0)} = \tau\mathbf{K} + \mathbf{M}$, we recover the solenoidal projection (5.21) for the instationary Stokes control problem.

Crank–Nicolson for Instationary Navier–Stokes Control

In this short section we present the linear system arising upon employing Crank–Nicolson in time when solving (6.19)–(6.20). The starting point will again be the discretized optimality conditions for instationary Stokes control with Crank–Nicolson applied in time. We discretize the interval $(0, t_f)$ into n_t subintervals of length $\tau = \frac{t_f}{n_t}$, and approximate $\vec{v}$ and $\vec{\zeta}$ at the time points $t_n = n\tau$, $n = 0, 1, \ldots, n_t$, employing a staggered grid for p and μ, as in [13]. Specifically, our approximations of the solutions at the k-th non-linear iteration are given by $\boldsymbol{v}_n^{(k)} \approx \vec{v}(x, t_n)$, $\boldsymbol{\zeta}_n^{(k)} \approx \vec{\zeta}(x, t_n)$, for $n = 0, 1, \ldots, n_t$, and $\boldsymbol{p}_{n+\frac{1}{2}}^{(k)} \approx p(x, t_n + \frac{1}{2}\tau)$, $\boldsymbol{\mu}_{n+\frac{1}{2}}^{(k)} \approx \mu(x, t_n + \frac{1}{2}\tau)$, for $n = 0, 1, \ldots, n_t - 1$, for all $x \in \Omega$. In addition, we

introduce the following finite element matrices:

$$\mathbf{L}_n^{\pm,(k)} = \tfrac{\tau}{2}(\nu\mathbf{K} + \mathbf{N}_n^{(k)} + \mathbf{W}_n^{(k)}) \pm \mathbf{M}, \quad \mathbf{T}_n^{\pm,(k)} = \tfrac{\tau}{2}(\nu\mathbf{K} - \mathbf{N}_n^{(k)} + \mathbf{W}_n^{(k)}) \pm \mathbf{M},$$
$$\bar{\mathbf{M}}^{\mathrm{CN}} = \tfrac{\tau}{2}\mathbf{M}, \quad \bar{\mathbf{M}}_\beta^{\mathrm{CN}} = \tfrac{\tau}{2\beta}\mathbf{M},$$

with $\mathbf{W}_n^{(k)}$, $\mathbf{N}_n^{(k)}$ defined as for backward Euler. Then the discrete Oseen iterate is

$$\begin{aligned}
&\boldsymbol{v}_n^{(k+1)} = \boldsymbol{v}_n^{(k)} + \boldsymbol{\delta v}_n^{(k)}, && \boldsymbol{\zeta}_n^{(k+1)} = \boldsymbol{\zeta}_n^{(k)} + \boldsymbol{\delta\zeta}_n^{(k)}, && n = 0, 1, \ldots, n_t,\\
&\boldsymbol{p}_{n+\frac{1}{2}}^{(k+1)} = \boldsymbol{p}_{n+\frac{1}{2}}^{(k)} + \boldsymbol{\delta p}_{n+\frac{1}{2}}^{(k)}, && \boldsymbol{\mu}_{n+\frac{1}{2}}^{(k+1)} = \boldsymbol{\mu}_{n+\frac{1}{2}}^{(k)} + \boldsymbol{\delta\mu}_{n+\frac{1}{2}}^{(k)}, && n = 0, 1, \ldots, n_t - 1,
\end{aligned}$$

with $\boldsymbol{\delta v}_n^{(k)}$, $\boldsymbol{\delta\zeta}_n^{(k)}$, $\boldsymbol{\delta p}_{n+\frac{1}{2}}^{(k)}$, $\boldsymbol{\delta\mu}_{n+\frac{1}{2}}^{(k)}$ solutions of the following discretized version of (6.19):

$$\begin{cases}
\bar{\mathbf{M}}^{\mathrm{CN}}(\boldsymbol{\delta v}_n^{(k)} + \boldsymbol{\delta v}_{n+1}^{(k)}) + \mathbf{T}_n^{+,(k)}\boldsymbol{\delta\zeta}_n^{(k)} + \mathbf{T}_{n+1}^{-,(k)}\boldsymbol{\delta\zeta}_{n+1}^{(k)} + \bar{B}^\top\boldsymbol{\delta\mu}_{n+\frac{1}{2}}^{(k)} = \boldsymbol{R}_{2,n}^{(k)},\\
\mathbf{L}_n^{-,(k)}\boldsymbol{\delta v}_n^{(k)} + \mathbf{L}_{n+1}^{+,(k)}\boldsymbol{\delta v}_{n+1}^{(k)} + \bar{B}^\top\boldsymbol{\delta p}_{n+\frac{1}{2}}^{(k)} - \bar{\mathbf{M}}_\beta^{\mathrm{CN}}(\boldsymbol{\delta\zeta}_n^{(k)} + \boldsymbol{\delta\zeta}_{n+1}^{(k)}) = \boldsymbol{R}_{1,n}^{(k)},\\
B\boldsymbol{\delta v}_{n+1}^{(k)} = \boldsymbol{r}_{1,n+1}^{(k)},\\
B\boldsymbol{\delta\zeta}_n^{(k)} = \boldsymbol{r}_{2,n}^{(k)},
\end{cases}$$

for $n = 0, 1, ..., n_t - 1$, with $\boldsymbol{\delta v}_0^{(k)} = \mathbf{0}$, $\boldsymbol{\delta\zeta}_{n_t}^{(k)} = \mathbf{0}$. The discretized residuals are given by

$$\begin{cases}
\boldsymbol{R}_{1,n}^{(k)} = \tfrac{\tau}{2}(\boldsymbol{f}^n + \boldsymbol{f}^{n+1}) - \mathbf{L}_n^{-,(k)}\boldsymbol{v}_n^{(k)} - \mathbf{L}_{n+1}^{+,(k)}\boldsymbol{v}_{n+1}^{(k)} - \bar{B}^\top\boldsymbol{p}_{n+\frac{1}{2}}^{(k)}\\
\qquad\qquad + \bar{\mathbf{M}}_\beta^{\mathrm{CN}}(\boldsymbol{\zeta}_n^{(k)} + \boldsymbol{\zeta}_{n+1}^{(k)}),\\
\boldsymbol{r}_{1,n+1}^{(k)} = -B\,\boldsymbol{v}_{n+1}^{(k)},\\
\boldsymbol{R}_{2,n}^{(k)} = \bar{\mathbf{M}}^{\mathrm{CN}}(\boldsymbol{v}_d^n + \boldsymbol{v}_d^{n+1}) - \bar{\mathbf{M}}^{\mathrm{CN}}(\boldsymbol{v}_n^{(k)} + \boldsymbol{v}_{n+1}^{(k)}) - \mathbf{T}_n^{+,(k)}\boldsymbol{\zeta}_n^{(k)}\\
\qquad\qquad - \mathbf{T}_{n+1}^{-,(k)}\boldsymbol{\zeta}_{n+1}^{(k)} - \bar{B}^\top\boldsymbol{\mu}_{n+\frac{1}{2}}^{(k)} - \tfrac{\tau}{2}(\boldsymbol{\omega}_n^{(k)} + \boldsymbol{\omega}_{n+1}^{(k)}),\\
\boldsymbol{r}_{2,n}^{(k)} = -B\,\boldsymbol{\zeta}_n^{(k)},
\end{cases} \tag{6.26}$$

for $n = 0, 1, ..., n_t - 1$. Here, $\boldsymbol{f}^n$ and $\boldsymbol{\omega}_n^{(k)}$ are defined as for backward Euler, for $n = 0, 1, \ldots, n_t$. Note also that here the non-linear residuals $\boldsymbol{R}_{1,0}^{(k)}$, $\boldsymbol{R}_{1,n_t-1}^{(k)}$, $\boldsymbol{R}_{2,0}^{(k)}$, and $\boldsymbol{R}_{2,n_t-1}^{(k)}$ in (6.26) take into account the initial and final conditions on $\vec{v}$ and $\vec{\zeta}$.

In matrix form, after multipling the incompressibility constraints by τ, we write

$$\begin{bmatrix}
\bar{\mathcal{M}}^{\mathrm{CN}} & \bar{\mathcal{L}}_1^{\mathrm{CN},(k)} & (\bar{\mathcal{B}}_2^{\mathrm{CN}})^\top & 0\\
\bar{\mathcal{L}}_2^{\mathrm{CN},(k)} & -\bar{\mathcal{M}}_\beta^{\mathrm{CN}} & 0 & (\bar{\mathcal{B}}_1^{\mathrm{CN}})^\top\\
\bar{\mathcal{B}}_1^{\mathrm{CN}} & 0 & 0 & 0\\
0 & \bar{\mathcal{B}}_2^{\mathrm{CN}} & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\bar{\boldsymbol{\delta v}}^{(k)}\\ \bar{\boldsymbol{\delta\zeta}}^{(k)}\\ \bar{\boldsymbol{\delta\mu}}^{(k)}\\ \bar{\boldsymbol{\delta p}}^{(k)}
\end{bmatrix} =
\begin{bmatrix}
\bar{\mathbf{b}}_1^{(k)}\\ \bar{\mathbf{b}}_2^{(k)}\\ \bar{\mathbf{b}}_3^{(k)}\\ \bar{\mathbf{b}}_4^{(k)}
\end{bmatrix}, \tag{6.27}$$

where $\bar{\boldsymbol{\delta v}}^{(k)}$, $\bar{\boldsymbol{\delta\zeta}}^{(k)}$, $\bar{\boldsymbol{\delta\mu}}^{(k)}$, $\bar{\boldsymbol{\delta p}}^{(k)}$ are the k-th Oseen corrections, and the right-hand side accounts for the non-linear residual. The blocks in the previous matrix are

given by

$$\bar{\mathcal{L}}_1^{\mathrm{CN},(k)} = \begin{bmatrix} \mathbf{T}_0^{+,(k)} & \mathbf{T}_1^{-,(k)} & & \\ & \ddots & \ddots & \\ & & \mathbf{T}_{n_t-1}^{+,(k)} & \mathbf{T}_{n_t}^{-,(k)} \\ & & & \mathbf{M} \end{bmatrix}, \quad \bar{\mathcal{L}}_2^{\mathrm{CN},(k)} = \begin{bmatrix} \mathbf{M} & & & \\ \mathbf{L}_0^{-,(k)} & \mathbf{L}_1^{+,(k)} & & \\ & \ddots & \ddots & \\ & & \mathbf{L}_{n_t-1}^{-,(k)} & \mathbf{L}_{n_t}^{+,(k)} \end{bmatrix},$$

$$\bar{\mathcal{B}}_1^{\mathrm{CN}} = \begin{bmatrix} 0 & \bar{B} & & \\ & \ddots & & \\ & & \bar{B} \end{bmatrix}, \qquad \bar{\mathcal{B}}_2^{\mathrm{CN}} = \begin{bmatrix} \bar{B} & & \\ & \ddots & \\ & & \bar{B} \quad 0 \end{bmatrix},$$

and $\bar{\mathcal{M}}^{\mathrm{CN}} = (I_{n_t+1,1} + I_{n_t+1,3}) \otimes \bar{\mathbf{M}}^{\mathrm{CN}}$, $\bar{\mathcal{M}}_\beta^{\mathrm{CN}} = (I_{n_t+1,2} + I_{n_t+1,3}^\top) \otimes \bar{\mathbf{M}}_\beta^{\mathrm{CN}}$.

As for instationary Stokes control with Crank–Nicolson in time, the system (6.27) is not symmetric; however, we work as in [100] and in the previous chapter in order to transform the linear system above and make it as close to symmetric as possible. In fact, eliminating the initial and final-time conditions on $\vec{v}$ and $\vec{\zeta}$, we can rewrite

$$\begin{bmatrix} \widetilde{\mathcal{M}}^{\mathrm{CN}} & \widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)} & (\widetilde{\mathcal{B}}^{\mathrm{CN}})^\top & 0 \\ \widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} & -\widetilde{\mathcal{M}}_\beta^{\mathrm{CN}} & 0 & (\widetilde{\mathcal{B}}^{\mathrm{CN}})^\top \\ \widetilde{\mathcal{B}}^{\mathrm{CN}} & 0 & 0 & 0 \\ 0 & \widetilde{\mathcal{B}}^{\mathrm{CN}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta v}^{(k)} \\ \boldsymbol{\delta \zeta}^{(k)} \\ \boldsymbol{\delta \mu}^{(k)} \\ \boldsymbol{\delta p}^{(k)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^{(k)} \\ \mathbf{b}_2^{(k)} \\ \mathbf{b}_3^{(k)} \\ \mathbf{b}_4^{(k)} \end{bmatrix},$$

with $\boldsymbol{\delta v}^{(k)}$, $\boldsymbol{\delta \zeta}^{(k)}$, $\boldsymbol{\delta \mu}^{(k)}$, $\boldsymbol{\delta p}^{(k)}$ as well as the right-hand side modified accordingly. The matrices $\widetilde{\mathcal{M}}^{\mathrm{CN}} = I_{n_t,4}^\top \otimes \bar{\mathbf{M}}^{\mathrm{CN}}$, $\widetilde{\mathcal{M}}_\beta^{\mathrm{CN}} = I_{n_t,4} \otimes \bar{\mathbf{M}}_\beta^{\mathrm{CN}}$, $\widetilde{\mathcal{B}}^{\mathrm{CN}} = I_{n_t} \otimes \bar{B}$, and

$$\widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)} = \begin{bmatrix} \mathbf{T}_0^{+,(k)} & \mathbf{T}_1^{-,(k)} & & \\ & \ddots & \ddots & \\ & & \mathbf{T}_{n_t-2}^{+,(k)} & \mathbf{T}_{n_t-1}^{-,(k)} \\ & & & \mathbf{T}_{n_t-1}^{+,(k)} \end{bmatrix}, \quad \widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} = \begin{bmatrix} \mathbf{L}_1^{+,(k)} & & & \\ \mathbf{L}_1^{-,(k)} & \mathbf{L}_2^{+,(k)} & & \\ & \ddots & \ddots & \\ & & \mathbf{L}_{n_t-1}^{-,(k)} & \mathbf{L}_{n_t}^{+,(k)} \end{bmatrix}.$$

We now consider the linear transformation T defined as in (5.25), with T_1, T_2, T_3, and T_4 defined as in (5.26). Then, the previous system is equivalent to the following:

$$\underbrace{\begin{bmatrix} \Phi_{\mathrm{CN}}^{(k)} & (\Psi_{\mathrm{CN}})^\top \\ \Psi_{\mathrm{CN}} & -\Theta_{\mathrm{CN}} \end{bmatrix}}_{\mathcal{A}_{\mathrm{CN}}^{(k)}} \begin{bmatrix} \boldsymbol{\delta v}^{(k)} \\ \boldsymbol{\delta \zeta}^{(k)} \\ \boldsymbol{\delta \mu}^{(k)} \\ \boldsymbol{\delta p}^{(k)} \end{bmatrix} = T \begin{bmatrix} \mathbf{b}_1^{(k)} \\ \mathbf{b}_2^{(k)} \\ \mathbf{b}_3^{(k)} \\ \mathbf{b}_4^{(k)} \end{bmatrix}. \tag{6.28}$$

Here the matrix blocks are given by

$$\Phi_{\mathrm{CN}}^{(k)} = \begin{bmatrix} \mathcal{M}^{\mathrm{CN}} & \mathcal{L}_1^{\mathrm{CN},(k)} \\ \mathcal{L}_2^{\mathrm{CN},(k)} & -\mathcal{M}_\beta^{\mathrm{CN}} \end{bmatrix}, \quad \Psi_{\mathrm{CN}} = \begin{bmatrix} \mathcal{B}_1^{\mathrm{CN}} & 0 \\ 0 & \mathcal{B}_2^{\mathrm{CN}} \end{bmatrix}, \quad \Theta_{\mathrm{CN}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{6.29}$$

with

$$\begin{aligned}
&\mathcal{M}^{\mathrm{CN}} = T_1\widetilde{\mathcal{M}}^{\mathrm{CN}} = \left(I_{n_t,4}\; I_{n_t,4}^{\top}\right)\otimes\bar{\mathbf{M}}^{\mathrm{CN}}, \quad && \mathcal{B}_1^{\mathrm{CN}} = T_3\widetilde{\mathcal{B}}^{\mathrm{CN}} = I_{n_t,4}^{\top}\otimes\bar{B},\\
&\mathcal{M}_\beta^{\mathrm{CN}} = T_2\widetilde{\mathcal{M}}_\beta^{\mathrm{CN}} = \left(I_{n_t,4}^{\top}\; I_{n_t,4}\right)\otimes\bar{\mathbf{M}}_\beta^{\mathrm{CN}}, \quad && \mathcal{B}_2^{\mathrm{CN}} = T_4\widetilde{\mathcal{B}}^{\mathrm{CN}} = I_{n_t,4}\otimes\bar{B},\\
&\mathcal{L}_1^{\mathrm{CN},(k)} = T_1\widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)}, && \mathcal{L}_2^{\mathrm{CN},(k)} = T_2\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)}.
\end{aligned} \tag{6.30}$$

System (6.28) is still not symmetric in general, as $\mathcal{L}_1^{\mathrm{CN},(k)} \neq (\mathcal{L}_2^{\mathrm{CN},(k)})^{\top}$ due to the mismatch of the indices for the convection terms. We recall that the transformations $T_i,\ i = 1,2,3,4$, as well as their inverse operations are easy and cheap to apply, as they require only a sequence of block updates. In addition, the matrices $\mathcal{M}^{\mathrm{CN}}$ and $\mathcal{M}_\beta^{\mathrm{CN}}$ can be rewritten as in (5.30), with $\mathcal{M}_D^{\mathrm{CN}}$ and $\mathcal{M}_{D,\beta}^{\mathrm{CN}}$ defined as in (5.31). Finally, we recall that we may work efficiently with $\mathcal{M}^{\mathrm{CN}}$ and $\mathcal{M}_\beta^{\mathrm{CN}}$, by employing T_1, T_2, $\mathcal{M}_D^{\mathrm{CN}}$, $\mathcal{M}_{D,\beta}^{\mathrm{CN}}$, and that $\mathcal{M}^{\mathrm{CN}}$ and $\mathcal{M}_\beta^{\mathrm{CN}}$ are symmetric positive definite, since the same holds for $\mathcal{M}_D^{\mathrm{CN}}$ and $\mathcal{M}_{D,\beta}^{\mathrm{CN}}$.

As for instationary Stokes control with Crank–Nicolson applied in time, it is not straightforward to generalize the strategy presented here to the case where $\vec{v}_0$ is not incompressible, as in this case we must also solve an appropriate solenoidal projection. However, as we mentioned above, the projection cannot be solved along with the other equations, as our approach requires the elimination of the initial and final conditions on $\vec{v}$ and $\vec{\zeta}$. Therefore, before applying our solver we must solve the projection to a stricter tolerance than that required for the control problem.

6.4 Preconditioning Approach

We now devise preconditioners for the systems (6.16), (6.24), and (6.28) arising upon discretization of the optimality conditions for the problems under examination by making use of saddle-point theory. We will generalize the preconditioners derived in Section 5.5 for Stokes control problems. Each of the preconditioner below is of the form $\widehat{\mathcal{P}}_1$ as defined in (2.30): this requires us to (approximately) apply the inverse of the corresponding $(1,1)$-block of each matrix analysed; we accelerate this process by again employing an approximation of the form $\widehat{\mathcal{P}}_1$ as defined in (2.30). In the following, subscripts refer to the corresponding matrix we are considering; to simplify the notation, we drop the superscript referring to the non-linear iterate k.

6.4.1 Approximation of the $(1,1)$-Block

We now describe suitable approximations of the inverses of the $(1,1)$-blocks for the systems (6.16), (6.24), and (6.28). As noted after the discretization of the optimality conditions, each of these matrices is not symmetric if we solve the incompressibility constraints inexactly (for a Crank–Nicolson discretization the block is not symmetric even if those constraints are solved exactly). We thus use a fixed number of GMRES iterations to approximate the $(1,1)$-block, accelerated with the preconditioners described below, as opposed to Uzawa iteration for ex-

ample (see [43]) which may be symmetrized, and so has utility within a MINRES solver, for instance.

Stationary Navier–Stokes Control

Consider the $(1,1)$-block $\Phi_{\mathrm{S}}^{(k)}$ defined in (6.17). This matrix can be considered as the discretization of the optimality conditions for a stationary convection–diffusion control problem. Using saddle-point theory, a suitable preconditioner is given by

$$\mathcal{P}_{\Phi,\mathrm{S}} = \begin{bmatrix} \mathbf{M} & 0 \\ \mathbf{L}^{(k)} & -\mathbf{S}_{\Phi,\mathrm{S}} \end{bmatrix},$$

with $\mathbf{S}_{\Phi,\mathrm{S}} = \mathbf{M}_\beta + \mathbf{L}^{(k)}\mathbf{M}^{-1}\mathbf{L}_{\mathrm{adj}}^{(k)}$ the corresponding Schur complement. As described in [141], a potent preconditioner for $\mathcal{P}_{\Phi,\mathrm{S}}$ (optimal in the symmetric case) is given by

$$\widehat{\mathcal{P}}_{\Phi,\mathrm{S}} = \begin{bmatrix} \mathbf{M}_c & 0 \\ \mathbf{L}^{(k)} & -\widehat{\mathbf{S}}_{\Phi,\mathrm{S}} \end{bmatrix}.$$

Here, $\mathbf{M}_c$ represents a fixed number of steps of the Chebyshev semi-iterative method [63, 64, 181], and

$$\widehat{\mathbf{S}}_{\Phi,\mathrm{S}} = \big(\mathbf{L}^{(k)} + \mathbf{M}_{\sqrt{\beta}}\big)\mathbf{M}^{-1}\big(\mathbf{L}_{\mathrm{adj}}^{(k)} + \mathbf{M}_{\sqrt{\beta}}\big),$$

with $\mathbf{M}_{\sqrt{\beta}} = \frac{1}{\sqrt{\beta}}\mathbf{M}$ and the blocks $\mathbf{L}^{(k)} + \mathbf{M}_{\sqrt{\beta}}$ and $\mathbf{L}_{\mathrm{adj}}^{(k)} + \mathbf{M}_{\sqrt{\beta}}$ approximated by the action of a multigrid routine, for example. It is worth noting that, if the incompressibility constraints are solved exactly, $\Phi_{\mathrm{S}}^{(k)}$ is symmetric and the approximation $\widehat{\mathbf{S}}_{\Phi,\mathrm{S}}$ of the Schur complement $\mathbf{S}_{\Phi,\mathrm{S}}$ is optimal; in fact, it can be proved that $\lambda(\widehat{\mathbf{S}}_{\Phi,\mathrm{S}}^{-1}\,\mathbf{S}_{\Phi,\mathrm{S}}) \in \left[\frac{1}{2}, 1\right]$ [141].

Instationary Navier–Stokes Control with Backward Euler

We now derive a preconditioner for the matrix $\Phi_{\mathrm{BE}}^{(k)}$ defined in (6.25). As in the stationary case, the matrix can be considered as the discretization of the optimality conditions for an instationary convection–diffusion control problem with backward Euler in time. As the matrix $\mathcal{M}^{\mathrm{BE}}$ is not invertible, we seek a preconditioner of the form:

$$\widetilde{\mathcal{P}}_{\Phi,\mathrm{BE}} = \begin{bmatrix} \widetilde{\mathcal{M}}^{\mathrm{BE}} & 0 \\ \mathcal{L}_2^{\mathrm{BE},(k)} & -\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}} \end{bmatrix},$$

with $\widetilde{\mathcal{M}}^{\mathrm{BE}}$ an invertible approximation of $\mathcal{M}^{\mathrm{BE}}$, and the perturbed Schur complement $\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}} = \mathcal{M}_\beta^{\mathrm{BE}} + \mathcal{L}_2^{\mathrm{BE},(k)}\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1}\mathcal{L}_1^{\mathrm{BE},(k)}$. Since the $(1,1)$-block is the same as for instationary Stokes control with backward Euler in time, here we employ the same approximation $\widetilde{\mathcal{M}}^{\mathrm{BE}}$ of $\mathcal{M}^{\mathrm{BE}}$. Specifically, a suitable approximation of $\mathcal{M}^{\mathrm{BE}}$ is given by

$$\widetilde{\mathcal{M}}^{\mathrm{BE}} = \mathtt{blkdiag}(\bar{\mathbf{M}}^{\mathrm{BE}}, \ldots, \bar{\mathbf{M}}^{\mathrm{BE}}, \xi\bar{\mathbf{M}}^{\mathrm{BE}}),$$

with $0 < \xi \ll 1$. In addition, following the work in [139], we can derive that a good approximation for $\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}$ is the matrix

$$\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}} = \big(\mathcal{L}_2^{\mathrm{BE},(k)} + \mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}}\big)\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1}\big(\mathcal{L}_1^{\mathrm{BE},(k)} + \mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}}\big),$$

with

$$\mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}} = \frac{\tau}{\sqrt{\beta}}\mathtt{blkdiag}(0, \mathbf{M}, \ldots, \mathbf{M}, \sqrt{\xi}\,\mathbf{M}).$$

As for the stationary case, the blocks $\mathcal{L}_2^{\mathrm{BE},(k)} + \mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}}$ and $\mathcal{L}_1^{\mathrm{BE},(k)} + \mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}}$ are not inverted exactly, but rather we apply block-forward and block-backward substitution respectively, with each block on the diagonal approximated by the action of a multigrid process, for instance. Finally, a suitable approximation of the matrix $\widetilde{\mathcal{P}}_{\Phi,\mathrm{BE}}$ is given by

$$\widehat{\mathcal{P}}_{\Phi,\mathrm{BE}} = \begin{bmatrix} \widehat{\mathcal{M}}_c^{\mathrm{BE}} & 0 \\ \mathcal{L}_2^{\mathrm{BE},(k)} & -\hat{\mathcal{S}}_{\Phi,\mathrm{BE}} \end{bmatrix}, \qquad \widehat{\mathcal{M}}_c^{\mathrm{BE}} = \tau\,\mathtt{blkdiag}(\mathbf{M}_c, \ldots, \mathbf{M}_c, \xi\mathbf{M}_c).$$

It is worth noting that, if the incompressibility constraints are solved exactly, the Schur complement approximation $\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}}$ of $\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}$ derived here is optimal. In fact, following the work in Chapter 3 and in Chapter 4, it is possible to prove that $\lambda(\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}}^{-1}\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}) \in [\frac{1}{2}, 1]$, as follows. Suppose that the incompressibility constraints are solved exactly. We first observe that $(\mathbf{N}_n^{(k)})^\top = -\mathbf{N}_n^{(k)}$, which implies $\mathcal{L}_1^{\mathrm{BE},(k)} = (\mathcal{L}_2^{\mathrm{BE},(k)})^\top$. Then, since both the matrices $\widetilde{\mathcal{M}}^{\mathrm{BE}}$ and $\mathcal{M}_\beta^{\mathrm{BE}}$ are symmetric positive definite, from Theorem 1 we derive that $\frac{1}{2}$ is a lower bound for the eigenvalues of the matrix $\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}}^{-1}\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}$. In addition, we employ Theorem 2 in order to prove that 1 is an upper bound for the eigenvalues of $\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}}^{-1}\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}$. For this to hold, it is enough to prove that the matrix

$$\mathcal{X}^{(k)} = \mathcal{L}_2^{\mathrm{BE},(k)}\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1}\mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}} + \mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}}\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1}(\mathcal{L}_2^{\mathrm{BE},(k)})^\top$$

is positive semi-definite. Recalling that $\bar{\mathbf{M}}^{\mathrm{BE}} = \tau\mathbf{M}$, it is easy to derive that

$$\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1}\mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}} = \mathcal{M}_{\sqrt{\beta}}^{\mathrm{BE}}\big(\widetilde{\mathcal{M}}^{\mathrm{BE}}\big)^{-1} = \frac{1}{\sqrt{\beta}}\mathtt{blkdiag}(0, I_{n_v}, \ldots, I_{n_v}, (\sqrt{\xi})^{-1}\, I_{n_v}),$$

which implies that

$$\mathcal{X}^{(k)} = \frac{1}{\sqrt{\beta}}\left(\mathcal{X}_1^{(k)} + \mathcal{X}_2\right),$$

with $\mathcal{X}_1^{(k)} = 2\tau\mathtt{blkdiag}(0, \widetilde{\mathbf{L}}_1^{(k)}, \ldots, \widetilde{\mathbf{L}}_{n_t-1}^{(k)}, (\sqrt{\xi})^{-1}\widetilde{\mathbf{L}}_{n_t}^{(k)})$, where $\widetilde{\mathbf{L}}_n^{(k)} = \nu\mathbf{K} + \mathbf{W}_n^{(k)}$,

for $n = 1, 2, \ldots, n_t$, and

$$\mathcal{X}_2 = \underbrace{\begin{bmatrix} 0 & 0 & & & \\ 0 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & 2 & -1 \\ & & & -1 & \frac{2}{\sqrt{\xi}} \end{bmatrix}}_{\mathcal{T}} \otimes \mathbf{M}.$$

We observe that the matrix $\mathcal{T}$ can be rewritten as follows:

$$\mathcal{T} = \mathcal{T}_1^\top \mathcal{T}_1 + \begin{bmatrix} 0 & & & & \\ & 0 & & & \\ & & \ddots & & \\ & & & 0 & \\ & & & & \frac{2}{\sqrt{\xi}} - 1 \end{bmatrix},$$

where

$$\mathcal{T}_1 = \begin{bmatrix} 0 & & & & \\ 0 & 1 & & & \\ & -1 & \ddots & & \\ & & \ddots & 1 & \\ & & & -1 & 1 \end{bmatrix}.$$

From here, we derive that the matrix $\mathcal{T}$ is positive semi-definite, as it is the sum of two symmetric positive semi-definite matrices. Recalling that also $\mathbf{M}$ is positive definite and employing Theorem 3, we can infer that the matrix $\mathcal{X}_2$ is positive semi-definite. Similarly, since each $\widetilde{\mathbf{L}}_n^{(k)}$ is positive definite, for $n = 1, 2, \ldots, n_t$, we can also infer that the matrix $\mathcal{X}_1^{(k)}$ is positive semi-definite. Then, due to $\mathcal{X}_1^{(k)}$ and $\mathcal{X}_2$ being positive semi-definite we derive that the same holds for the matrix $\mathcal{X}$. Finally, the latter together with Theorem 2 implies that 1 is an upper bound for the eigenvalues of the matrix $\widehat{\mathcal{S}}_{\Phi,\mathrm{BE}}^{-1}\widetilde{\mathcal{S}}_{\Phi,\mathrm{BE}}$.

Instationary Navier–Stokes Control with Crank–Nicolson

Let us consider the linear system $\Phi_{\mathrm{CN}}^{(k)}$ defined in (6.29), arising from a Crank–Nicolson discretization. In order to devise a preconditioner for this system, as for the backward Euler case, we observe that this matrix can be considered as the (symmetrized) discretization of the optimality conditions for the control of the instationary convection–diffusion equation discretized when employing Crank–Nicolson in time. Again, a suitable preconditioner is the block triangular matrix

$$\mathcal{P}_{\Phi,\mathrm{CN}} = \begin{bmatrix} \mathcal{M}^{\mathrm{CN}} & 0 \\ \mathcal{L}_2^{\mathrm{CN},(k)} & -S_{\Phi,\mathrm{CN}} \end{bmatrix},$$

where $S_{\Phi,\mathrm{CN}} = \mathcal{M}_\beta^{\mathrm{CN}} + \mathcal{L}_2^{\mathrm{CN},(k)}(\mathcal{M}^{\mathrm{CN}})^{-1}\mathcal{L}_1^{\mathrm{CN},(k)}$. In order to find an approximation of $\mathcal{P}_{\Phi,\mathrm{CN}}$, we adapt the strategy used in [100] and discussed in Section 4.3.1 as

follows.

The $(1,1)$-block $\mathcal{M}^{\mathrm{CN}}$ is the same as that arising upon discretization of the optimality conditions for instationary Stokes control with Crank–Nicolson in time. Therefore, we may find an approximation by working as in the previous chapter. In fact, from (5.30)–(5.31), we can rewrite $\mathcal{M}^{\mathrm{CN}}$ as $\mathcal{M}^{\mathrm{CN}} = T_1 \mathcal{M}_D^{\mathrm{CN}} T_1^\top$, with $\mathcal{M}_D^{\mathrm{CN}}$ a block diagonal matrix with each diagonal block a multiple of $\mathbf{M}$. Therefore, a good approximation of $\mathcal{M}^{\mathrm{CN}}$ is given by $\widehat{\mathcal{M}}^{\mathrm{CN}} = T_1 \widehat{\mathcal{M}}_D^{\mathrm{CN}} T_1^\top$, with $\widehat{\mathcal{M}}_D^{\mathrm{CN}} = \frac{\tau}{2} I_{n_t} \otimes \mathbf{M}_c$.

To derive an approximation of $S_{\Phi,\mathrm{CN}}$, we use (5.30) together with (6.30) to rewrite

$$S_{\Phi,\mathrm{CN}} = T_2 \underbrace{\big[\mathcal{M}_{D,\beta}^{\mathrm{CN}} + \big(\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)}\big)\big(\mathcal{M}^{\mathrm{CN}}\big)^{-1}\big(T_1 \widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)}\, T_1^{-1}\big)\big]}_{S_{\Phi,\mathrm{CN}}^{\mathrm{int}}} T_1, \tag{6.31}$$

recalling that $T_1 = T_2^\top$. We first seek an approximation $\widehat{S}_{\Phi,\mathrm{CN}}^{\mathrm{int}}$ for $S_{\Phi,\mathrm{CN}}^{\mathrm{int}}$ of the form

$$\widehat{S}_{\Phi,\mathrm{CN}}^{\mathrm{int}} = \big(\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}_2\big)\big(\mathcal{M}^{\mathrm{CN}}\big)^{-1}\big(T_1 \widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)}\, T_1^{-1} + \widehat{\mathcal{M}}_1\big),$$

such that

$$\widehat{\mathcal{M}}_2\big(\mathcal{M}^{\mathrm{CN}}\big)^{-1}\widehat{\mathcal{M}}_1 = \big(\widehat{\mathcal{M}}_2 T_2^{-1}\big)\big(\mathcal{M}_D^{\mathrm{CN}}\big)^{-1}\big(T_1^{-1}\widehat{\mathcal{M}}_1\big) = \mathcal{M}_{D,\beta}^{\mathrm{CN}}.$$

The previous expression is clearly satisfied with the choice

$$\widehat{\mathcal{M}}_2 T_2^{-1} = T_1^{-1}\widehat{\mathcal{M}}_1 = \frac{\tau}{2\sqrt{\beta}} I_{n_t} \otimes \mathbf{M}.$$

Then, our approximation of $S_{\Phi,\mathrm{CN}}^{\mathrm{int}}$ is given by

$$\begin{aligned}\widehat{S}_{\Phi,\mathrm{CN}}^{\mathrm{int}} &= \big(\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}\big)\big(\mathcal{M}^{\mathrm{CN}}\big)^{-1}\big(T_1 \widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)}\, T_1^{-1} + \widehat{\mathcal{M}}^{\top}\big)\\ &= \big(\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}\big) T_2^{-1}\big(\mathcal{M}_D^{\mathrm{CN}}\big)^{-1}\big(\widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)}\, T_1^{-1} + T_1^{-1}\widehat{\mathcal{M}}^{\top}\big),\end{aligned}$$

with $\widehat{\mathcal{M}} = \frac{\tau}{2\sqrt{\beta}} I_{n_t,4}^\top \otimes \mathbf{M}$. Finally, substituting $\widehat{S}_{\Phi,\mathrm{CN}}^{\mathrm{int}}$ into (6.31) and observing that $\widehat{\mathcal{M}}$ and T_1 commute, we obtain that our approximation of $S_{\Phi,\mathrm{CN}}$ is given by

$$\widehat{S}_{\Phi,\mathrm{CN}} = T_2\big(\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}\big) T_2^{-1}\big(\mathcal{M}_D^{\mathrm{CN}}\big)^{-1}\big(\widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}^{\top}\big).$$

As for the backward Euler method, we apply the inverse operators of the matrices $\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}$ and $\widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}^{\top}$ inexactly by employing a block-forward and block-backward substitution respectively, with the action of a multigrid process used to approximately apply the inverse of each block diagonal entry.

As opposed to the previous cases, it is not possible in general to prove bounds on eigenvalues for this preconditioner, derived for instationary Navier–Stokes control with Crank–Nicolson in time, as the Schur complement approximation $\widehat{S}_{\Phi,\mathrm{CN}}$ is in general non-symmetric. However, if we suppose that the velocity $\vec{v}$ is constant in time, then the matrix T_2 commutes with $\widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} + \widehat{\mathcal{M}}$, and the Schur complement approximation derived here reduces to the one we derived in Section

4.3.1 for time-dependent convection–diffusion control problems when the Crank–Nicolson method is applied in time. The last observation implies that the Schur complement approximation derived here is optimal in the simplified setting of the velocity $\vec{v}$ being constant in time, due to Assumption 1 trivially holding, since we are considering enclosed flow.

6.4.2 Approximation of Schur Complement

We now derive efficient approximations for each Schur complement of the systems (6.16), (6.24), and (6.28). As for the Stokes control problems, the $(2,1)$- and the $(1,2)$-blocks of these systems can be considered as a (negative) vector-divergence matrix and its transpose. Therefore, the starting points for our approximations are the commutator arguments employed for Stokes control problems described in Section 5.5.2. In addition to the Laplacian $-\nabla^2$, all the differential operators considered below will take into account also for a convection term. The discretization of the latter will include also a stabilization matrix, aside from the convection matrix. The latter will be quite an important component of our preconditioners, as it will allow more robustness with respect to the viscosity ν.

Stationary Navier–Stokes Control

Let us consider the Schur complement $S_{\mathcal{A},\mathrm{S}} = \Psi_{\mathrm{S}}(\Phi_{\mathrm{S}}^{(k)})^{-1}\Psi_{\mathrm{S}}^{\top}$ of the system (6.16), with $\Phi_{\mathrm{S}}^{(k)}$ and Ψ_{S} defined as in (6.17). As for stationary Stokes control, we apply the commutator argument to $\mathcal{E}_{\bar{m}}$ as defined in (5.9) with $\bar{m} = 2$, with the differential operator on the velocity space taking also into account a convection term. Specifically, the differential operator on the velocity space is defined as

$$\mathcal{D} = \begin{bmatrix} \mathrm{Id} & -\nu\nabla^2 - \vec{v}^{\,(k)} \cdot \nabla \\ -\nu\nabla^2 + \vec{v}^{\,(k)} \cdot \nabla & -\frac{1}{\beta}\mathrm{Id} \end{bmatrix},$$

and $\mathcal{D}_p$ the corresponding differential operator on the pressure space; we recall from Section 6.1.1 that Id represents the identity operator. Employing stable finite elements and working as in Section 5.3, we obtain the following expression for (5.11):

$$\widehat{S}_{\mathcal{A},\mathrm{S}} = \begin{bmatrix} K_p & 0 \\ 0 & K_p \end{bmatrix} \begin{bmatrix} M_p & L_{\mathrm{adj},p}^{(k)} \\ L_p^{(k)} & -M_{\beta,p} \end{bmatrix}^{-1} \begin{bmatrix} M_p & 0 \\ 0 & M_p \end{bmatrix} \approx S_{\mathcal{A},\mathrm{S}},$$

where we set

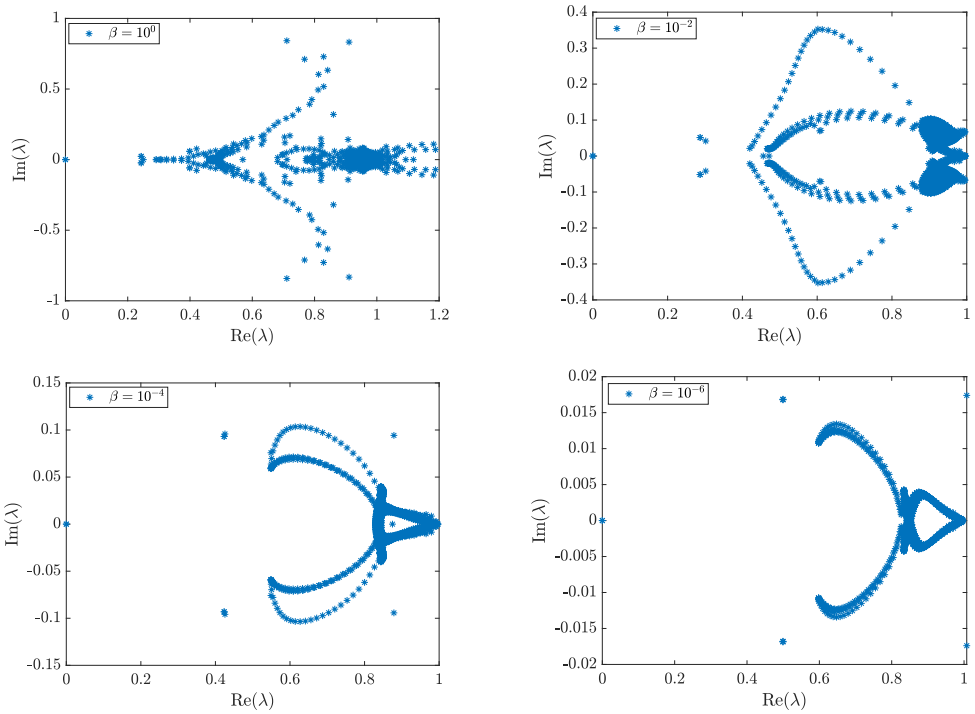
$$L_p^{(k)} = \nu K_p + N_p^{(k)} + W_p^{(k)}, \qquad L_{\mathrm{adj},p}^{(k)} = \nu K_p - N_p^{(k)} + W_p^{(k)},$$

and $M_{\beta,p} = \frac{1}{\beta}M_p$.

Before presenting the approximations of the Schur complements arising from the instationary case, as we did for the stationary Stokes control problem, we would like to show the effectiveness of our approach. In Figure 6.1, we report the eigenvalues of the matrix $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1}S_{\mathcal{A},\mathrm{S}}$, after three Oseen iterations of the test

problem defined in Section 6.5.1, for $\nu = \frac{1}{100}$ and level of refinement $\mathtt{l} = 5$, and for some range of β, where $\mathtt{l}$ represents a (spatial) uniform grid of mesh-size $h = 2^{1-\mathtt{l}}$ for $\mathbf{Q}_1$ basis functions, and $h = 2^{-\mathtt{l}}$ for $\mathbf{Q}_2$ elements, in each dimension. As for the stationary Stokes control problem, we "pin" the value of one of the nodes of the matrix K_p, for each K_p in $\mathtt{blkdiag}(K_p, K_p)$ in the factorization of $\widehat{S}_{\mathcal{A},\mathrm{S}}$.

Figure 6.1: Commutator approximation for the stationary Navier–Stokes control problem defined in Section 6.5.1. Eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ after three Oseen iterations, with $\Omega = (-1, 1)^2$ and $\nu = \frac{1}{100}$, for $\beta = 10^{-j}$, $j = 0, 2, 4, 6$, and $\mathtt{l} = 5$.



In Figure 6.1, we observe a strong cluster of eigenvalues around 0.5 and 1. However, as opposed to the stationary Stokes control problem, the real part of the eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ are not clustered between 0.2 and 1 for all the parameters β. In addition, the imaginary part of the eigenvalues of $\widehat{S}_{\mathcal{A},\mathrm{S}}^{-1} S_{\mathcal{A},\mathrm{S}}$ depends on the parameter β. Nonetheless, the approximation adopted seems quite effective.

Instationary Navier–Stokes Control with Backward Euler

We now consider the Schur complement $S_{\mathcal{A},\mathrm{BE}} = \Psi_{\mathrm{BE}} (\Phi_{\mathrm{BE}}^{(k)})^{-1} \Psi_{\mathrm{BE}}^{\top}$ of (6.24), and derive an efficient approximation by employing the commutator argument (5.9). As we mentioned for instationary Stokes control, the differential operator $\mathcal{D}$ is chosen in such a way that it mimics the blocks of a suitable matrix. In this case, we want to mimic the blocks of $\Phi_{\mathrm{BE}}^{(k)}$ defined in (6.25). Again, the starting point is the commutator argument for instationary Stokes control with backward Euler in time, with the differential operator $\mathcal{D}$ taking into account also a convection term.

Specifically, we consider (5.9) with $\bar{m} = 2(n_t + 1)$ and the differential operator:

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}_{\mathrm{BE}}^{1,1} & \mathcal{D}_{\mathrm{BE}}^{1,2} \\ \mathcal{D}_{\mathrm{BE}}^{2,1} & \mathcal{D}_{\mathrm{BE}}^{2,2} \end{bmatrix},$$

where $\mathcal{D}_{\mathrm{BE}}^{1,1} = \tau I_{n_t+1,1} \otimes \mathrm{Id}$, $\mathcal{D}_{\mathrm{BE}}^{2,2} = -\frac{\tau}{\beta} I_{n_t+1,2} \otimes \mathrm{Id}$, and

$$\mathcal{D}_{\mathrm{BE}}^{1,2} = \begin{bmatrix} \mathcal{D}_{0,\,\mathrm{adj}} & -\mathrm{Id} & & \\ & \ddots & \ddots & \\ & & \mathcal{D}_{n_t-1,\,\mathrm{adj}} & -\mathrm{Id} \\ & & & \mathcal{D}_{n_t,\,\mathrm{adj}} \end{bmatrix}, \quad \mathcal{D}_{\mathrm{BE}}^{2,1} = \begin{bmatrix} \mathcal{D}_0 & & & \\ -\mathrm{Id} & \mathcal{D}_1 & & \\ & \ddots & \ddots & \\ & & -\mathrm{Id} & \mathcal{D}_{n_t} \end{bmatrix},$$

with

$$\mathcal{D}_i = \tau(-\nu\nabla^2 + \vec{v}_i^{(k)} \cdot \nabla) + \mathrm{Id}, \qquad \mathcal{D}_{i,\,\mathrm{adj}} = \tau(-\nu\nabla^2 - \vec{v}_i^{(k)} \cdot \nabla) + \mathrm{Id}.$$

As above, we define $\mathcal{D}_p$ as the corresponding differential operator on the pressure space. Discretizing (5.9) and observing that $S_{\mathcal{A},\mathrm{BE}} = \tau^2 \vec{B}\, \mathbf{D}^{-1} \vec{B}^{\top}$, with $\mathbf{D}$ the discretization of the differential operator $\mathcal{D}$ and $\vec{B} = I_{2(n_t+1)} \otimes B$, we obtain the following approximation:

$$\widehat{S}_{\mathcal{A},\mathrm{BE}} = \tau^2 \mathcal{K}_p^{\mathrm{BE}} \begin{bmatrix} D_{p,\,\mathrm{BE}}^{1,1} & D_{p,\,\mathrm{BE}}^{1,2} \\ D_{p,\,\mathrm{BE}}^{2,1} & D_{p,\,\mathrm{BE}}^{2,2} \end{bmatrix}^{-1} \mathcal{M}_p^{\mathrm{BE}} \approx S_{\mathcal{A},\mathrm{BE}}.$$

Here, we set

$$\begin{aligned} \mathcal{K}_p^{\mathrm{BE}} &= I_{2(n_t+1)} \otimes K_p, & \mathcal{M}_p^{\mathrm{BE}} &= I_{2(n_t+1)} \otimes M_p, \\ \mathcal{D}_{p,\,\mathrm{BE}}^{1,1} &= \tau I_{n_t+1,1} \otimes M_p, & \mathcal{D}_{p,\,\mathrm{BE}}^{2,2} &= -\frac{\tau}{\beta} I_{n_t+1,2} \otimes M_p, \end{aligned}$$

and

$$D_{p,\,\mathrm{BE}}^{1,2} = \begin{bmatrix} T_{0,p}^{(k)} & -M_p & & \\ & \ddots & \ddots & \\ & & T_{n_t-1,p}^{(k)} & -M_p \\ & & & T_{n_t,p}^{(k)} \end{bmatrix}, \quad D_{p,\,\mathrm{BE}}^{2,1} = \begin{bmatrix} L_{0,p}^{(k)} & & & \\ -M_p & L_{1,p}^{(k)} & & \\ & \ddots & \ddots & \\ & & -M_p & L_{n_t,p}^{(k)} \end{bmatrix},$$

with

$$L_{i,p}^{(k)} = \tau(\nu K_p + N_{i,p}^{(k)} + W_{i,p}^{(k)}) + M_p, \qquad T_{i,p}^{(k)} = \tau(\nu K_p - N_{i,p}^{(k)} + W_{i,p}^{(k)}) + M_p.$$

Instationary Navier–Stokes Control with Crank–Nicolson

Finally, we move on to finding an approximation for the Schur complement $S_{\mathcal{A},\mathrm{CN}} = \Psi_{\mathrm{CN}}(\Phi_{\mathrm{CN}}^{(k)})^{-1}\Psi_{\mathrm{CN}}^{\top}$ of (6.28). As we have done for the Schur complement arising from the backward Euler discretization, the starting point is the commutator argument for the instationary Stokes control problem with Crank–

Nicolson in time. Again, we apply the commutator argument (5.9) to a suitable differential operator $\mathcal{D}$.

We first observe that the Schur complement $S_{\mathcal{A},\mathrm{CN}}$ can be rewritten as

$$S_{\mathcal{A},\mathrm{CN}} = \begin{bmatrix} T_3 & 0 \\ 0 & T_4 \end{bmatrix} \begin{bmatrix} \widetilde{\mathcal{B}}^{\,\mathrm{CN}} & 0 \\ 0 & \widetilde{\mathcal{B}}^{\,\mathrm{CN}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathcal{M}}^{\mathrm{CN}} & \widetilde{\mathcal{L}}_1^{\mathrm{CN},(k)} \\ \widetilde{\mathcal{L}}_2^{\mathrm{CN},(k)} & -\widetilde{\mathcal{M}}_\beta^{\mathrm{CN}} \end{bmatrix}^{-1} \begin{bmatrix} \widetilde{\mathcal{B}}^{\,\mathrm{CN}} & 0 \\ 0 & \widetilde{\mathcal{B}}^{\,\mathrm{CN}} \end{bmatrix}^{\top}.$$

Then, we consider (5.9) with $\bar{m} = 2n_t$ and the differential operator

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}_{\mathrm{CN}}^{1,1} & \mathcal{D}_{\mathrm{CN}}^{1,2} \\ \mathcal{D}_{\mathrm{CN}}^{2,1} & \mathcal{D}_{\mathrm{CN}}^{2,2} \end{bmatrix},$$

where $\mathcal{D}_{\mathrm{CN}}^{1,1} = \frac{\tau}{2} I_{n_t,4}^{\top} \otimes \mathrm{Id}$, $\mathcal{D}_{\mathrm{CN}}^{2,2} = -\frac{\tau}{2\beta} I_{n_t,4} \otimes \mathrm{Id}$, and

$$\mathcal{D}_{\mathrm{CN}}^{1,2} = \begin{bmatrix} \mathcal{D}_{0,\,\mathrm{adj}}^{+} & \mathcal{D}_{1,\,\mathrm{adj}}^{-} & & \\ & \ddots & \ddots & \\ & & \mathcal{D}_{n_t-2,\,\mathrm{adj}}^{+} & \mathcal{D}_{n_t-1,\,\mathrm{adj}}^{-} \\ & & & \mathcal{D}_{n_t-1,\,\mathrm{adj}}^{+} \end{bmatrix}, \quad \mathcal{D}_{\mathrm{CN}}^{2,1} = \begin{bmatrix} \mathcal{D}_1^{+} & & & \\ \mathcal{D}_1^{-} & \mathcal{D}_2^{+} & & \\ & \ddots & \ddots & \\ & & \mathcal{D}_{n_t-1}^{-} & \mathcal{D}_{n_t}^{+} \end{bmatrix},$$

with

$$\mathcal{D}_i^{\pm} = \frac{\tau}{2}(-\nu\nabla^2 + \vec{v}_i^{\,(k)} \cdot \nabla) \pm \mathrm{Id}, \qquad \mathcal{D}_{i,\,\mathrm{adj}}^{\pm} = \frac{\tau}{2}(-\nu\nabla^2 - \vec{v}_i^{\,(k)} \cdot \nabla) \pm \mathrm{Id}.$$

Denoting with $\mathcal{D}_p$ the corresponding differential operator on the pressure space and proceeding as above, we can derive the following approximation:

$$\widehat{S}_{\mathcal{A},\mathrm{CN}} = \tau^2 \begin{bmatrix} T_3 & 0 \\ 0 & T_4 \end{bmatrix} \mathcal{K}_p^{\mathrm{CN}} \begin{bmatrix} D_{p,\,\mathrm{CN}}^{1,1} & D_{p,\,\mathrm{CN}}^{1,2} \\ D_{p,\,\mathrm{CN}}^{2,1} & D_{p,\,\mathrm{CN}}^{2,2} \end{bmatrix}^{-1} \mathcal{M}_p^{\mathrm{CN}} \approx S_{\mathcal{A},\mathrm{CN}}.$$

Here, we set

$$\begin{aligned} \mathcal{K}_p^{\mathrm{CN}} &= I_{2n_t} \otimes K_p, & \mathcal{M}_p^{\mathrm{CN}} &= I_{2n_t} \otimes M_p, \\ \mathcal{D}_{p,\,\mathrm{CN}}^{1,1} &= \frac{\tau}{2} I_{n_t,4}^{\top} \otimes M_p, & \mathcal{D}_{p,\,\mathrm{CN}}^{2,2} &= -\frac{\tau}{2\beta} I_{n_t,4} \otimes M_p, \end{aligned}$$

and

$$D_{p,\,\mathrm{CN}}^{1,2} = \begin{bmatrix} T_{0,p}^{+,(k)} & T_{1,p}^{-,(k)} & & \\ & \ddots & \ddots & \\ & & T_{n_t-2,p}^{+,(k)} & T_{n_t-1,p}^{-,(k)} \\ & & & T_{n_t-1,p}^{+,(k)} \end{bmatrix}, \; D_{p,\,\mathrm{CN}}^{2,1} = \begin{bmatrix} L_{1,p}^{+,(k)} & & & \\ L_{1,p}^{-,(k)} & L_{2,p}^{+,(k)} & & \\ & \ddots & \ddots & \\ & & L_{n_t-1,p}^{-,(k)} & L_{n_t,p}^{+,(k)} \end{bmatrix},$$

with

$$L_{i,p}^{\pm,(k)} = \tfrac{\tau}{2}(\nu K_p + N_{i,p}^{(k)} + W_{i,p}^{(k)}) \pm M_p, \quad T_{i,p}^{\pm,(k)} = \tfrac{\tau}{2}(\nu K_p - N_{i,p}^{(k)} + W_{i,p}^{(k)}) \pm M_p.$$

6.5 Numerical Results

We now demonstrate the effectiveness of our preconditioners by presenting numerical results. In all our tests, $d = 2$ (that is, $x = [x_1, x_2]^\top$), and $\Omega = (-1, 1)^2$. All tests are run on MATLAB R2018b, using a 1.70GHz Intel quad-core i5 processor and 8 GB RAM on an Ubuntu 18.04.1 LTS operating system.

As our preconditioners are non-symmetric and require an inner solve for the $(1,1)$-block, for the outer solver we apply flexible GMRES [155] restarted every 10 iterations, up to a tolerance 10^{-6} on the relative residual; we make this choice as, at each step, we require the Oseen linearization to be solved to a stricter tolerance than that of the non-linear residual reduction we wish to achieve. Our implementation is based on the flexible GMRES routine in the TT-Toolbox [127]. To apply the approximate inverse of the $(1,1)$-block, we take 5 iterations of the GMRES routine implemented in MATLAB. We apply 20 steps of Chebyshev semi-iteration to mass matrices (on the velocity or pressure space); we apply 4 V-cycles of the AGMG routine [118, 121, 122, 123] for other matrices constructed on the velocity space[8], while employing 2 V-cycles (with 2 symmetric Gauss–Seidel iterations for pre-/post-smoothing) of the `HSL_MI20` solver [22] for stiffness matrices on the pressure space within our Schur complement approximation.

Regarding the non-linear iteration for solving the Navier–Stokes control problem, we allow 20 Oseen iterations, specifying as a stopping criterion a reduction of 10^{-5} on the (non-linear) relative residual; the initial residual is the right-hand side of the corresponding Stokes control problem, with $\nu = 1$ (for the instationary case with Crank–Nicolson, we evaluate the residual before applying T). For each problem below, the first Oseen iterate is employed for the Stokes control solve, whose solutions $\boldsymbol{v}^{(1)}$, $\boldsymbol{\zeta}^{(1)}$, $\boldsymbol{p}^{(1)}$, $\boldsymbol{\mu}^{(1)}$ are then used as the initial guess. We use inf–sup stable Taylor–Hood $\mathbf{Q}_2$–$\mathbf{Q}_1$ finite elements in the spatial dimensions, with level of refinement `l` representing a (spatial) uniform grid of mesh-size $h = 2^{1-\mathtt{l}}$ for $\mathbf{Q}_1$ basis functions, and $h = 2^{-\mathtt{l}}$ for $\mathbf{Q}_2$ elements, in each dimension. All CPU times below are reported in seconds.

6.5.1 Stationary Navier–Stokes Control

We first test our solver on the stationary Navier–Stokes control problem (6.1)–(6.2). We set $\vec{f} = \vec{0}$, $\vec{v}_d = \vec{0}$, and

$$\vec{g} = \begin{cases} [1,0]^\top & \text{on } \partial\Omega_1 := (-1,1) \times \{1\}\,, \\ [0,0]^\top & \text{on } \partial\Omega \backslash \partial\Omega_1. \end{cases}$$

We report the average number of GMRES iterations, together with the average CPU time per GMRES solve, for the stationary Navier–Stokes control problem in Tables 6.1–6.3, and in Table 6.4 we state the total degrees of freedom (DoF) together with the total number of Oseen iterations required. We provide results for different levels of refinement `l`, values of β, and viscosities ν.

[8]Note that, since each of these blocks contains a different convection matrix, the multigrid routine cannot be recycled, as the prolongation and the restriction operators have to be computed again.

Table 6.1: Average GMRES iterations and CPU times for stationary Navier–Stokes control problem, for $\nu = \frac{1}{20}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
3	21	0.40	19	0.37	15	0.24	12	0.12	11	0.17	10	0.13	9	0.13
4	22	1.14	20	1.19	18	0.92	15	0.74	12	0.44	11	0.57	10	0.52
5	24	4.81	21	4.06	20	3.62	17	2.73	15	2.37	12	1.55	12	1.59
6	26	24.2	25	22.8	20	18.0	18	16.1	17	14.1	16	12.4	13	8.43
7	31	112	25	89.6	23	83.3	20	68.6	17	57.9	16	52.0	16	50.8
8	40	665	32	526	28	457	22	360	19	304	18	281	16	257

Table 6.2: Average GMRES iterations and CPU times for stationary Navier–Stokes control problem, for $\nu = \frac{1}{100}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
3	38	0.79	24	0.30	13	0.20	11	0.19	11	0.18	9	0.13	9	0.14
4	31	1.84	24	1.28	18	0.96	12	0.43	11	0.61	11	0.59	10	0.50
5	29	5.47	23	4.23	20	3.23	16	2.32	12	1.60	11	1.58	11	1.64
6	31	28.0	27	23.6	22	18.5	18	14.6	15	11.1	12	8.05	11	8.14
7	32	116	27	93.6	24	83.9	20	69.6	17	58.7	15	46.7	13	35.9
8	38	627	32	528	27	437	22	351	19	298	17	276	15	236

Table 6.3: Average GMRES iterations and CPU times for stationary Navier–Stokes control problem, for $\nu = \frac{1}{500}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
3	77†[9]	1.46†	23	0.44	13	0.25	11	0.24	10	0.19	9	0.14	9	0.14
4	86†	4.18†	48†	2.02†	18	1.01	12	0.68	11	0.67	10	0.57	10	0.55
5	74	15.4	49	8.02	28	4.13	14	2.08	12	1.87	11	1.73	10	1.50
6	57	55.1	39	34.2	27	22.0	20	14.6	13	8.93	11	8.86	11	8.63
7	54	192	32	111	27	90.6	21	67.3	17	49.4	12	32.5	12	37.2
8	53	878	34	561	29	472	23	369	19	292	16	232	13	172

Tables 6.1–6.3 demonstrate the robustness of our proposed preconditioner. The numbers of iterations show only a mild dependence on the viscosity ν, and a slight increase only for large values of β. The CPU time scales approximately linearly with respect to the dimension of the systems, with a marginal increase for very fine grids; in this case we observe that the AGMG multigrid routine does

not scale exactly linearly. Table 6.4 shows that the number of Oseen iterations strongly depends on the viscosity ν, as expected as the non-linear term becomes increasingly significant for smaller ν; however, as the grid is refined the number of outer iterations decreases. We also note that the number of linear and non-linear iterations increases for larger values of β and coarser grids.

Table 6.4: Degrees of freedom (DoF) and number of Oseen iterations required for stationary Navier–Stokes control problem. In each cell are the Oseen iterations for the given `l`, ν, and $\beta = 10^{-j}$, $j = 0, 1, ..., 6$.

`l`	DoF	$\nu = \frac{1}{20}$	$\nu = \frac{1}{100}$	$\nu = \frac{1}{500}$
3	1062	5 5 5 5 4 4 3	13 9 7 5 4 4 3	† 20 7 5 4 4 3
4	4422	5 5 4 4 4 4 4	8 6 6 5 4 4 4	† † 9 5 4 4 4
5	18,054	4 4 4 4 4 4 3	7 5 5 4 4 4 3	16 10 8 6 4 4 3
6	72,966	4 4 4 3 3 3 3	6 4 4 4 4 4 3	11 6 5 5 4 4 3
7	293,382	4 3 3 3 3 3 3	5 4 3 3 3 3 3	8 4 4 4 4 4 3
8	1,176,582	3 3 3 3 3 3 3	4 3 3 3 3 3 3	5 3 3 3 3 3 3

6.5.2 Instationary Navier–Stokes Control

We now test our solver on the instationary Navier–Stokes control problem (6.3)–(6.4), where we set $t_f = 2$, $\vec{f}(x,t) = \vec{0}$, the initial condition $\vec{v}_0(x) = \vec{0}$, and boundary conditions

$$\vec{g}(x,t) = \begin{cases} [t,0]^\top & \text{on } \partial\Omega_1 \times (0,1), \\ [1,0]^\top & \text{on } \partial\Omega_1 \times [1,t_f), \\ [0,0]^\top & \text{on } (\partial\Omega \backslash \partial\Omega_1) \times (0,t_f), \end{cases}$$

with $\partial\Omega_1$ defined as in Section 6.5.1. We present results obtained by employing backward Euler and Crank–Nicolson discretizations in time. Setting

$$c_1 = 1 - \sqrt{(\tfrac{100}{49}(x_1 - \tfrac{1}{2}))^2 + (\tfrac{100}{99}x_2)^2},$$
$$c_2 = 1 - \sqrt{(\tfrac{100}{49}(x_1 + \tfrac{1}{2}))^2 + (\tfrac{100}{99}x_2)^2},$$

we seek the (divergence-free) desired state:

$$\vec{v}_d(x,t) = \begin{cases} c_1 \cos(\tfrac{\pi t}{2}) \, [(\tfrac{100}{99})^2 x_2, -(\tfrac{100}{49})^2 (x_1 - \tfrac{1}{2})]^\top & \text{if } c_1 \geqslant 0, \\ c_2 \cos(\tfrac{\pi t}{2}) \, [-(\tfrac{100}{99})^2 x_2, (\tfrac{100}{49})^2 (x_1 + \tfrac{1}{2})]^\top & \text{if } c_2 \geqslant 0, \\ [0,0]^\top & \text{otherwise.} \end{cases}$$

[9] † means that the outer (Oseen) iteration did not converge in 20 iterations. The average number of GMRES iterations and CPU time is evaluated over the first 10 Oseen iterations.

Backward Euler for Instationary Navier–Stokes Control

We first report the results obtained when employing the backward Euler scheme in time. We provide the average number of GMRES iterations together with the average elapsed CPU time in Tables 6.5–6.7, and in Table 6.8 the total dimensions of the systems solved and the Oseen iterations required, for different levels of refinements `l`, values of β, and viscosities ν. Here, we choose the time-step $\tau = 0.05$ (that is, $n_t = 40$), while the level of refinement `l` refers to a spatial uniform grid constructed as above.

Table 6.5: Average GMRES iterations and CPU times for instationary Navier–Stokes control problem, with backward Euler in time ($\tau = 0.05$), for $\nu = \frac{1}{20}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	17	6.74	16	6.18	13	5.24	11	4.42	11	4.11	12	4.89	20	7.75
3	18	9.36	18	9.71	16	8.6	14	9.84	12	10.1	14	10.2	22	17.5
4	19	40	19	41	18	46.2	17	42.5	15	30.2	16	41.3	24	65.1
5	19	195	20	206	19	176	17	153	17	142	17	135	24	194
6	23	1096	22	1053	20	970	18	843	17	771	18	828	25	995

Table 6.6: Average GMRES iterations and CPU times for instationary Navier–Stokes control problem, with backward Euler in time ($\tau = 0.05$), for $\nu = \frac{1}{100}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	17	6.96	14	5.70	11	4.47	11	4.21	10	4.08	12	4.93	21	8.54
3	22	18.8	19	17.7	14	13.2	11	10.9	11	10.6	13	11.8	21	19.7
4	23	45.3	22	45.4	18	39.0	14	37.5	13	35.0	15	41.2	23	61.4
5	22	190	22	196	19	169	17	148	15	126	16	136	25	220
6	25	1153	24	1099	21	979	18	809	17	729	17	685	25	1080

As for the stationary case, Tables 6.5–6.7 show robustness of the proposed preconditioner with respect to all the parameters involved. We note that the number of iterations increases slightly for small viscosities and large values of β. The elapsed CPU time scales almost linearly with the dimension of the system, except for very fine grids. We see from Table 6.8 that the number of Oseen iterations increases for small values of ν and large values of β when employing a coarse grid; however, as the grid is refined, the number of non-linear iterations decreases.

Table 6.7: Average GMRES iterations and CPU times for instationary Navier–Stokes control problem, with backward Euler in time ($\tau = 0.05$), for $\nu = \frac{1}{500}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	16†	6.66†	14†	5.55†	11†	4.43†	10†	3.92†	10	3.84	12	4.91	22	8.87
3	26†	28.9†	20†	25.4†	14	17.2	11	13.6	11	13.3	13	15.1	22	27.9
4	43†	130†	34	120	17	64.5	13	50.6	12	47.8	14	53.6	24	82.7
5	54	467	43	417	28	294	16	160	15	164	16	187	24	263
6	39	1723	35	1552	27	1209	20	842	17	742	18	830	26	1165

Table 6.8: Degrees of freedom (DoF) and number of Oseen iterations required for instationary Navier–Stokes control problem, with backward Euler in time ($\tau = 0.05$). In each cell are the Oseen iterations for the given `l`, ν, and $\beta = 10^{-j}, j = 0, 1, ..., 6$.

`l`	DoF	$\nu = \frac{1}{20}$							$\nu = \frac{1}{100}$							$\nu = \frac{1}{500}$						
2	10,086	6	6	5	5	5	5	5	15	8	6	5	5	5	5	†	†	†	†	10	8	8
3	43,542	5	5	5	5	4	4	4	9	8	6	5	5	5	5	†	†	7	6	6	6	6
4	181,302	5	5	4	4	4	4	4	6	6	6	5	5	5	5	†	14	8	6	6	6	6
5	740,214	4	4	4	4	4	3	3	5	5	5	4	4	4	4	8	8	7	5	5	5	5
6	2,991,606	4	4	4	3	3	3	3	4	4	4	4	4	3	3	6	5	5	5	4	4	4

Crank–Nicolson for Instationary Navier–Stokes Control

We now report the results obtained when applying Crank–Nicolson in time. We report the average number of GMRES iterations together with the average elapsed time for instationary Navier–Stokes control with Crank–Nicolson in time in Tables 6.9–6.11, and in Table 6.12 the total dimensions of the systems solved and the numbers of Oseen iterations, for different levels of refinements `l`, values of β, and viscosities ν. In addition, in Table 6.13 we report the average number of GMRES iterations, the average elapsed time, and the number of Oseen iterations `Os` with very small viscosity ν, for different levels of refinement `l`, and for some values of β. Here, for level of refinement `l` we divide the time interval into subintervals of length $2^{1-\mathtt{l}}$ and consider a spatial uniform grid of refinement level `l`. In Figure 6.2 we show the numerical solutions of the state and adjoint velocities $\vec{v}$ and $\vec{\zeta}$, at time $t = 1$, and of the pressure p, at time $t = 1.0625$, for $\nu = \frac{1}{100}$, $\beta = 10^{-1}$, and $\mathtt{l} = 4$.

From Tables 6.9–6.11 we observe that the number of iterations required for reaching a prescribed accuracy is, again, roughly constant, increasing only for small ν and large β. The dependence on the viscosity ν is more evident from Table 6.13, as in fact the number of iterations increases also for more moderate values of β, but remains low considering the dimensions of the systems solved. It is worth

Table 6.9: Average GMRES iterations and CPU times for instationary Navier–Stokes control problem, with Crank–Nicolson in time ($\tau = h$), for $\nu = \frac{1}{20}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	16	0.73	15	0.68	12	0.53	10	0.44	9	0.39	9	0.37	8	0.36
3	18	3.40	17	3.23	15	2.14	12	1.68	10	1.93	10	1.56	9	1.55
4	18	22.7	19	22.9	18	21.2	15	17.4	12	11.9	11	12.6	10	11.8
5	19	170	19	173	18	162	17	151	15	122	13	98.4	11	85.0
6	21	1948	21	1898	21	1848	18	1587	17	1448	15	1295	13	1022

Table 6.10: Average GMRES iterations and CPU times for instationary Navier–Stokes control problem, with Crank–Nicolson in time ($\tau = h$), for $\nu = \frac{1}{100}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	16	0.74	13	0.64	11	0.51	10	0.45	9	0.40	9	0.38	8	0.38
3	21	3.80	19	3.72	13	2.80	10	2.20	10	2.20	9	1.77	9	1.87
4	23	22.6	22	22.2	18	18.5	12	15.4	11	13.6	10	13.3	10	12.2
5	22	187	21	184	19	166	16	135	12	103	11	87.7	11	89.7
6	24	2141	24	2087	22	1922	18	1507	15	1272	12	973	11	913

Table 6.11: Average GMRES iterations and CPU times for instationary Navier–Stokes control problem, with Crank–Nicolson in time ($\tau = h$), for $\nu = \frac{1}{500}$ and a range of `l` and β.

	$\beta = 10^0$		$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-3}$		$\beta = 10^{-4}$		$\beta = 10^{-5}$		$\beta = 10^{-6}$	
`l`	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU	`it`	CPU
2	16†	0.75†	14†	0.64†	10†	0.46†	10	0.43	9	0.40	8	0.38	8	0.37
3	26†	6.49†	20†	5.70†	13	3.78	10	2.90	9	2.60	9	2.19	9	2.19
4	47	64.6	33	53.5	17	28.2	11	20.3	10	17.7	10	15.8	9	14.6
5	54	476	44	417	28	291	15	148	11	119	11	109	10	98.2
6	44	3728	37	3140	29	2472	20	1701	14	1157	11	983	11	968

noting that the preconditioner struggles only for viscosity $\nu = \frac{1}{2000}$, at which stage the problem becomes increasingly non-linear and convection-dominated. As experienced above, the CPU time scales approximately linearly with the size of the system, except for very fine grids. Regarding the non-linear iteration, as above we note in Table 6.12 that the number of Oseen iterations is decreasing as the grid is refined, while it is increasing for small values of ν and large values of β. From

Table 6.12: Degrees of freedom (DoF) and number of Oseen iterations required for instationary Navier–Stokes control problem, with Crank–Nicolson in time ($\tau = h$). In each cell are the Oseen iterations for the given `l`, ν, and $\beta = 10^{-j}$, $j = 0, 1, ..., 6$.

`l`	DoF	$\nu = \frac{1}{20}$							$\nu = \frac{1}{100}$							$\nu = \frac{1}{500}$						
2	984	6	6	6	6	5	4	3	12	7	7	7	5	4	4	†	†	†	10	7	5	4
3	8496	5	5	5	6	5	4	4	8	8	6	7	6	4	4	†	†	7	7	7	5	4
4	70,752	4	4	4	4	4	4	3	6	6	5	5	4	4	3	18	14	6	5	6	5	4
5	577,728	4	4	4	3	3	3	3	5	4	4	4	4	4	3	9	8	6	5	4	4	3
6	4,669,824	3	3	3	3	3	3	3	4	4	4	3	3	3	3	5	5	5	4	4	3	3

Table 6.13: Average GMRES iterations, average CPU times, and number of Oseen iterations (`Os`) for instationary Navier–Stokes control problem, with Crank–Nicolson in time ($\tau = h$), for $\nu = \frac{1}{1000}$ and $\nu = \frac{1}{2000}$, and for a range of `l` and β.

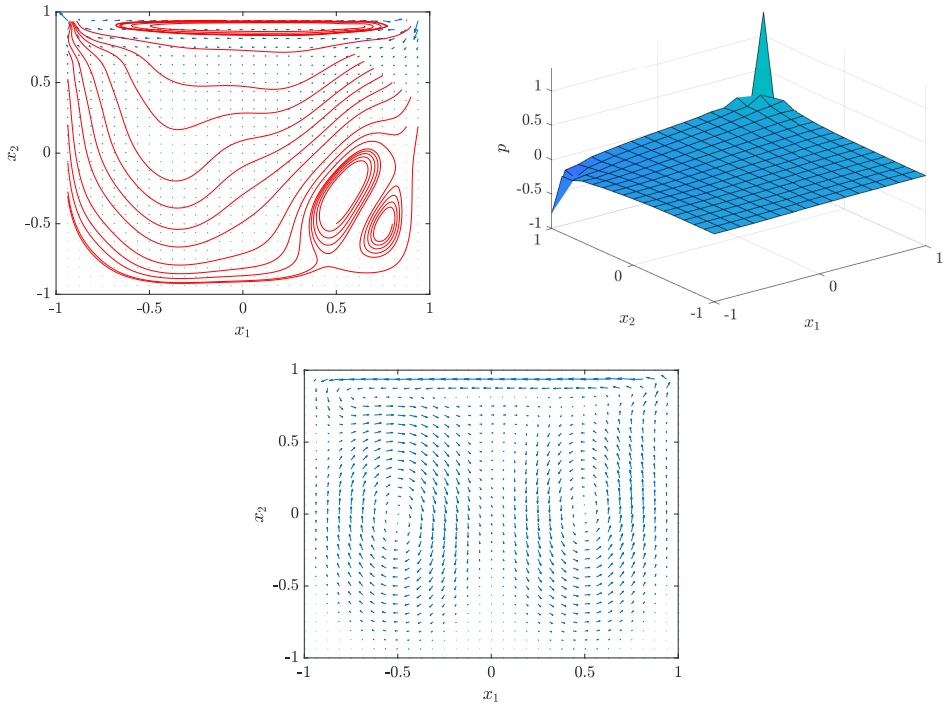
	$\nu = \frac{1}{1000}$									$\nu = \frac{1}{2000}$								
	$\beta = 10^{-2}$			$\beta = 10^{-4}$			$\beta = 10^{-6}$			$\beta = 10^{-2}$			$\beta = 10^{-4}$			$\beta = 10^{-6}$		
`l`	`it`	CPU	`Os`	`it`	CPU	`Os`	`it`	CPU	`Os`	`it`	CPU	`Os`	`it`	CPU	`Os`	`it`	CPU	`Os`
2	11†	0.51†	†	9	0.41	7	8	0.37	4	11†	0.54†	†	9	0.42	7	8	0.35	4
3	14	4.19	8	10	2.72	7	9	2.26	4	14	4.38	9	9	2.79	8	9	2.31	4
4	18	37.2	8	10	18.8	7	9	15.1	4	22	45.8	8	10	19.4	8	9	15.7	4
5	32	453	8	12	148	4	10	111	3	34†	568†	†	11	161	5	10	121	4
6	46	4833	6	13	1348	4	11	1046	3	77†	10505†	†	13	1548	4	11	1176	3

Table 6.13 we observe that for very small viscosities (e.g., $\nu = \frac{1}{2000}$) the number of Oseen iterations starts to increase for a wider range of regularization paramenters, due to the stronger non-linearity in the problem. From here, we envisage the utility of developing a robust solver for Newton's method for (stationary and instationary) Navier–Stokes control problems, which is a topic of future work and would be likely to mitigate effects of very small viscosities on the convergence of the outer iteration.

6.6 Summary and Comments

In this chapter, we presented mesh- and parameter-robust preconditioners for distributed Navier–Stokes control problems, of both stationary and instationary type, coupled with an Oseen linearization. For the instationary case, we employed either the backward Euler or the Crank–Nicolson discretization in the time variable. The preconditioners made use of most of the techniques presented so far in this work. Specifically, we employed saddle-point theory in conjuction with the commutator argument presented in the previous chapter for Stokes control, in

Figure 6.2: Solution plots for the instationary Navier–Stokes control problem, for $\nu = \frac{1}{100}$, $\beta = 10^{-1}$, and $\mathtt{l} = 4$. Top left: velocity $\vec{v}$ at $t = 1$. Top right: pressure p at $t = 1.0625$. Bottom: adjoint velocity $\vec{\zeta}$ at $t = 1$.



order to approximate the Schur complement of the corresponding discrete Oseen problems. In addition, we employed well known preconditioners for convection–diffusion control problems to accelerate the process of approximately applying the inverse operators of the corresponding $(1,1)$-block of the systems considered. The outer preconditioners were applied within the flexible GMRES algorithm. Numerical results demonstrated the versatility and effectiveness of this approach when solving a range of huge-scale linear systems.

As we did at the end of the previous chapter, before moving on to the next one we would like to devote some discussion to the technique described here. Specifically, we would like to discuss the utility of the stabilization matrix within our commutator argument.

In Section 6.4.2, we made use of a commutator argument for deriving effective approximations of the Schur complements of the discrete Oseen problems considered above. The discretizations of the commutator arguments were also taking into account a stabilization matrix, aside from the discretization of the convection term. As we mentioned, this has been done in order to give more robustness within the preconditioners with respect to the viscosity ν. We will explain this idea by considering the Oseen approximations of the stationary Navier–Stokes control problem.

In order to find an approximation of a critical point for (6.1)–(6.2), we iteratively solved the Oseen linearization (6.14). Upon discretization, we had to solve a sequence of saddle-point systems of the form (6.16), with the blocks defined as in (6.17). In order to understand the importance of the stabilization matrices, for

now we consider the case when no stabilization is used. In this case, the system we have to solve at each Oseen iteration is given in (6.16), with the blocks Ψ_{S} and Θ_{S} defined as in (6.17), and the $(1,1)$-block $\Phi_{\mathrm{S}}^{(k)}$ given by

$$\Phi_{\mathrm{S}}^{(k)} = \begin{bmatrix} \mathbf{M} & \nu\mathbf{K} - \mathbf{N}^{(k)} \\ \nu\mathbf{K} + \mathbf{N}^{(k)} & -\mathbf{M}_{\beta} \end{bmatrix}.$$

Here, we suppose that if a commutator argument can be used for approximating each block of the inverse of $\Phi_{\mathrm{S}}^{(k)}$, then we may be able to derive that a similar argument holds for the Schur complement $S_{\mathcal{A},\mathrm{S}}$ of $\mathcal{A}_{\mathrm{S}}$. In particular, let us consider the Schur complement of $\Phi_{\mathrm{S}}^{(k)}$ arising when no stabilization is employed. In this case, we have

$$\begin{aligned} \mathbf{S}_{\Phi,\mathrm{S}} &= \mathbf{M}_{\beta} + (\nu\mathbf{K} + \mathbf{N}^{(k)})\mathbf{M}^{-1}(\nu\mathbf{K} - \mathbf{N}^{(k)}) \\ &= \mathbf{M}_{\beta} + \nu^2\mathbf{K}\mathbf{M}^{-1}\mathbf{K} + \nu(\mathbf{N}^{(k)}\mathbf{M}^{-1}\mathbf{K} - \mathbf{K}\mathbf{M}^{-1}\mathbf{N}^{(k)}) - \mathbf{N}^{(k)}\mathbf{M}^{-1}\mathbf{N}^{(k)}. \end{aligned}$$

From here, we may be able to understand the complexity of finding preconditioners for Navier–Stokes control problems that are robust with respect to the viscosity ν, as the dependence of the Schur complement $\mathbf{S}_{\Phi,\mathrm{S}}$ on ν is not only linear, but (hiddenly) quadratic. In particular, fixing the mesh-size h and letting ν tend to 0 and β tend to infinity, we find the following approximation of the Schur complement $\mathbf{S}_{\Phi,\mathrm{S}}$:

$$\mathbf{S}_{\Phi,\mathrm{S}} \approx -\mathbf{N}^{(k)}\mathbf{M}^{-1}\mathbf{N}^{(k)}.$$

Finally, the latter approximation tells us that the commutator argument presented in this chapter may be not completely robust when fixing h and letting ν tend to 0 and β tend to infinity. However, if we include a stabilization $\mathbf{W}^{(k)}$ when discretizing the convection term, we can derive the following approximation of the Schur complement $\mathbf{S}_{\Phi,\mathrm{S}}$ for this parameter regime:

$$\mathbf{S}_{\Phi,\mathrm{S}} \approx \mathbf{W}^{(k)}\mathbf{M}^{-1}\mathbf{W}^{(k)} + (\mathbf{N}^{(k)}\mathbf{M}^{-1}\mathbf{W}^{(k)} - \mathbf{W}^{(k)}\mathbf{M}^{-1}\mathbf{N}^{(k)}) - \mathbf{N}^{(k)}\mathbf{M}^{-1}\mathbf{N}^{(k)}.$$

Now, as we mentioned above, the stabilization $\mathbf{W}^{(k)}$ is chosen to enhance coercivity of the discretization. The latter may be translated in terms of numerical linear algebra as: the stabilization $\mathbf{W}^{(k)}$ is such that the real part of the eigenvalues of the discretized differential operator is shifted further into the right half-plane. In particular, the stabilization matrix we employed is positive semi-definite, and can be considered as a shifted discrete diffusion operator associated with the streamline direction defined by the approximation of the velocity $\vec{v}^{(k)}$, see Section 4.1.1. Thus, it is not so surprising that we are able to recover robustness within our preconditioners, also for very small viscosity. Similar arguments can be made for the control of the instationary Navier–Stokes equations, either when employing backward Euler or Crank–Nicolson in time.

Chapter 7

Preconditioning Fractional Differential Equation Constrained Optimization Problems

"-Non ti disunire, Fabio.
-Mi chiamano tutti Fabietto.
-È ora ca' t' faje chiammà Fabio. Non ti disunire.
-Ma che significa?
-L'hê cap... hê capì tu sul'. [...] Non ti disunire, Schisa. Non ti disunire mai!"

["-Don't come apart, Fabio.
-Everybody calls me Fabietto.
-Well now it's time they called you Fabio. Don't come apart, Fabio.
-But what does it mean?
-You hav... you have to understand it by yourself. [...] Don't come apart, Schisa. Don't ever come apart!"]

– Paolo Sorrentino, *È stata la mano di Dio*

We are entering now the last main chapter of this work. Here, we consider the optimal control of *Fractional Differential Equations (FDEs)*, with additional algebraic constraints on the state and the control variables. As opposed to the classical derivatives treated so far, which are *local operators*, FDEs describe *non-local* properties: given a function f defined on a domain Ω, the *fractional derivative* of f at a given point of Ω is related to the effects of f on the whole domain. In particular, problems with non-local properties can frequently be modeled accurately using FDEs. Among other processes, FDEs have been used to model viscoelasticity (e.g., [94]), anomalous transport (e.g., [114]), and flow in porous media (e.g., [15]), with applications to biology (e.g., [2]), electrochemistry (e.g., [124]), electrical circuits (e.g., [144]), and in finance (e.g., [159]).

As for the problems considered in the previous chapter, in order to obtain an approximation of a solution of the problems considered here we have to run

a non-linear process due to the presence of the box constraints on the variables. As opposed to all the previous chapters, in which we employed an optimize-then-discretize approach, here we adopt a discretize-then-optimize solution strategy. We employ this strategy here in order to avoid having to derive an infinite-dimensional adjoint for a fractional operator. The convex quadratic optimization problem resulting from the discretization step consists of a quadratic cost functional to be minimized subject to a (very dense) system of linear equations (that presents a very specific structure), aside from the box constraints on the state and control variables. In order to preserve the structure of the linear system arising at each non-linear iteration, we separate inequality from equality constraints, with the latter solved by employing an efficient and robust preconditioner within a suitable Krylov subspace method (although we are splitting inequality from equality constraints, in this case it is important that we "don't come apart" so as to preserve the structure of the latter).

This chapter is structured as follows. In Section 7.1, we introduce the notions of *fractional integrals* and fractional derivatives. In Section 7.2, we describe how to discretize a fractional derivative, and show the linear system that arises upon discretization. Then, in Section 7.3, we describe the preconditioning technique employed for approximating the discretization of forward FDEs. In Section 7.4, we introduce the problem we consider in this chapter, that is, the optimal control of FDEs with additional box constraints on the state and the control variables, describing the strategy employed for finding an approximation of the solution in Section 7.5. The latter is based on an *Alternating Direction Method of Multipliers (ADMM)*, which allows us to separate the equality from the inequality constraints, solving the equality constraints, and then updating the current solutions. In Section 7.6, we present the preconditioners we employ at each non-linear iteration. Finally, in Section 7.7, we provide numerical results that show the robustness of our approach.

This chapter is based on the work in [146], which was a joint work with Spyros Pougkakiotis, John Pearson, and Jacek Gondzio at the University of Edinburgh. Where the author was not involved in the work discussed, we omit detailed discussion and the paper [146] is referred to, and we provide some auxiliary results which motivated the preconditioning strategy in Appendix A.

7.1 Fractional Calculus

In this section we introduce the notion of a fractional integral and fractional derivative. We follow the work in [66]. For a detailed discussion on the fractional calculus, see, for example, [116, 125, 143, 158].

7.1.1 Fractional Integral

We start our description with the Riemann–Liouville integral of a holomorphic function f defined on the interval $[0, t_f]$. This integral is a generalization of the

Cauchy formula [143, Section 2.3.1]

$$f^{(-n)}(t) = \frac{1}{(n-1)!}\int_0^t (t-\tau)^{n-1} f(\tau)\,\mathrm{d}\tau, \quad t > 0, \quad n \in \mathbb{N},$$

where $f^{(-n)}$ denotes the n-fold primitive of the function f. Following [66], we call the previous expression $J^n\, f(t)$. We note that $f^{(-n)}(t)$ vanishes at $t = 0$ with its derivatives of order $1, 2, \ldots, n-1$. For convention it is required that $f(t)$ is a *causal* function, that is identically vanishing for $t < 0$.

The way to generalize the Cauchy formula is to consider a positive number $\alpha > 0$ (not anymore constrained to be an integer) and then find an expression that reduces to the formula above when α is an integer. The only formal issue with this generalization is the definition of the factorial for a general positive number α. The natural way to do that is to consider the *Gamma function*. In fact, it is well known that $\Gamma(n) = (n-1)!$ whenever $n \in \mathbb{N}$. With that in mind, it is possible to define the *fractional integral of order* α as

$$J^\alpha\, f(t) := \frac{1}{\Gamma(\alpha)}\int_0^t (t-\tau)^{\alpha-1} f(\tau)\,\mathrm{d}\tau, \quad t > 0, \quad \alpha > 0,$$

and for the sake of being fully comprehensive we can set $J^0 := \mathrm{Id}$, with Id the identity operator, implying that $J^0\, f(t) = f(t)$.

One may prove the following semigroup property [66]:

$$J^\alpha\, J^\gamma = J^{\alpha+\gamma}, \quad \alpha, \gamma \geqslant 0,$$

which implies the commutative property $J^\alpha\, J^\gamma = J^\gamma\, J^\alpha$, and that the following equality holds [66]:

$$J^\alpha\, t^\gamma = \frac{\Gamma(\gamma+1)}{\Gamma(\gamma+1+\alpha)} t^{\gamma+\alpha}, \quad \alpha > 0, \quad \gamma > -1, \quad t > 0.$$

The above properties are clearly a generalization of the properties of the classical (integer order) integral, and the proofs are based on both the Gamma and *Beta* functions [116, 125]

$$\Gamma(z) = \int_0^\infty e^{-x} x^{z-1}\,\mathrm{d}x, \qquad \mathcal{R}e(z) > 0,$$

$$B(z_1, z_2) = \int_0^1 (1-x)^{z_1-1} x^{z_2-1}\,\mathrm{d}x, \qquad \mathcal{R}e(z_1),\, \mathcal{R}e(z_2) > 0,$$

where with $\mathcal{R}e(z) > 0$ we place a constraint on the real part of z.

7.1.2 Fractional Derivative

Once we have defined the fractional integral of order α, it is natural to seek the fractional derivative for a general real value of α.

If we denote by D^n, $n \in \mathbb{N}$, the classical derivative of order n, we observe that

$$D^n\, J^n = \mathrm{Id}, \quad J^n\, D^n \neq \mathrm{Id}, \qquad n \in \mathbb{N},$$

that is, D^n is a left-inverse but not a right-inverse to the corresponding integral operator J^n. In fact, we have that

$$J^n\, D^n\, f(t) = f(t) - \sum_{i=0}^{n-1} f^{(i)}(0^+) \frac{t^{\,i}}{i\,!}, \quad t > 0.$$

Therefore, it is reasonable to define D^α as left-inverse to J^α. Introducing the positive integer n such that $n - 1 < \alpha \leqslant n$, one can define the *(left-sided Riemann–Liouville) fractional derivative of order* α as

$${}^{\mathrm{RL}}_{\mathrm{L}}D^\alpha\, f(t) = D^n\, J^{n-\alpha}\, f(t),$$

namely

$${}^{\mathrm{RL}}_{\mathrm{L}}D^\alpha\, f(t) := \begin{cases} \dfrac{1}{\Gamma(n-\alpha)} \dfrac{\mathrm{d}^n}{\mathrm{d}t^n} \displaystyle\int_0^t \frac{f(\tau)}{(t-\tau)^{\alpha+1-n}}\, \mathrm{d}\tau, & n-1 < \alpha < n, \\ \dfrac{\mathrm{d}^n}{\mathrm{d}t^n} f(t), & \alpha = n, \end{cases}$$

and again for completeness one may set ${}^{\mathrm{RL}}_{\mathrm{L}}D^0 = J^0 = \mathrm{Id}$. From here, it is clear that

$${}^{\mathrm{RL}}_{\mathrm{L}}D^\alpha\, J^\alpha = \mathrm{Id}, \quad \alpha \geqslant 0,$$

and that

$${}^{\mathrm{RL}}_{\mathrm{L}}D^\alpha\, t^{\,\gamma} = \frac{\Gamma(\gamma+1)}{\Gamma(\gamma+1-\alpha)} t^{\gamma-\alpha}, \quad \alpha > 0, \quad \gamma > -1, \quad t > 0. \tag{7.1}$$

Again, we find a generalization of the properties of the classical (integer order) derivative. However, we realize that if $f(t) \equiv 1$ then ${}^{\mathrm{RL}}_{\mathrm{L}}D^\alpha\, f(t) \neq 0$ if $\alpha \notin \mathbb{N}$. In fact, using the expression above with $\gamma = 0$ we obtain

$${}^{\mathrm{RL}}_{\mathrm{L}}D^\alpha\, 1 = \frac{t^{-\alpha}}{\Gamma(1-\alpha)}, \quad \alpha \geqslant 0, \quad t > 0,$$

that is of course identically equal to zero when $\alpha \in \mathbb{N}$, due to the poles of the Gamma function at the points $0, -1, -2, \ldots$.

In a similar way, we can define the *(right-sided Riemann–Liouville) fractional*

derivative of order α as

$$
{}^{\mathrm{RL}}_{\mathrm{R}}D^{\alpha}\, f(t) := \begin{cases} \dfrac{(-1)^n}{\Gamma(n-\alpha)} \dfrac{\mathrm{d}^n}{\mathrm{d}t^n} \displaystyle\int_t^{t_f} \frac{f(\tau)}{(\tau - t)^{\alpha+1-n}}\, \mathrm{d}\tau, & n-1 < \alpha < n, \\ \dfrac{\mathrm{d}^n}{\mathrm{d}t^n} f(t), & \alpha = n. \end{cases}
$$

In addition, from the right- and left-sided Riemann–Liouville fractional derivative we can define the *symmetric Riesz derivative* as follows [143, 158]:

$$
{}^{\mathrm{R}}D^{\alpha}\, f(t) := \frac{-1}{2\cos(\frac{\alpha\pi}{2})} \Big({}^{\mathrm{RL}}_{\mathrm{L}}D^{\alpha}\, f(t) + {}^{\mathrm{RL}}_{\mathrm{R}}D^{\alpha}\, f(t) \Big).
$$

The previous expression is not defined if $\alpha = 1$, due to the cosine vanishing for this value of α. For this reason, in the following we consider only symmetric Riesz derivatives of order $\alpha \in (1, 2]$. For $\alpha \leqslant 1$, we require a different fractional derivative, which leads us to the following definition.

We can find also another way for defining the fractional derivative, the so-called *Caputo fractional derivative of order* $\alpha > 0$, defined as

$$
D^{\alpha}_{*}\, f(t) = J^{n-\alpha}\, D^n\, f(t),
$$

with $n - 1 < \alpha \leqslant n$, that is

$$
D^{\alpha}_{*}\, f(t) := \begin{cases} \dfrac{1}{\Gamma(n-\alpha)} \displaystyle\int_0^{t} \frac{f^{(n)}(\tau)}{(t - \tau)^{\alpha+1-n}}\, \mathrm{d}\tau, & n-1 < \alpha < n, \\ \dfrac{\mathrm{d}^n}{\mathrm{d}t^n} f(t), & \alpha = n. \end{cases}
$$

The last definition is more restrictive than the previous one, since it requires the absolute integrability of the derivative of order n. Moreover, from the two definitions it is clear that

$$
D^{\alpha}\, f(t) := D^n\, J^{n-\alpha}\, f(t) \neq J^{n-\alpha}\, D^n\, f(t) := D^{\alpha}_{*}\, f(t)
$$

unless the function $f(t)$ along with its first $n - 1$ derivatives vanishes at $t = 0^+$ [66]. In fact, assuming that the passage of the n-derivative under the integral is valid, we have that for $n - 1 < \alpha < n$ and $t > 0$:

$$
D^{\alpha}\, f(t) = D^{\alpha}_{*}\, f(t) + \sum_{i=0}^{n-1} \frac{t^{\,i-\alpha}}{\Gamma(i - \alpha + 1)}\, f^{(i)}(0^+), \tag{7.2}
$$

whereupon recalling (7.1) implies

$$
D^{\alpha} \left(f(t) - \sum_{i=0}^{n-1} \frac{t^{\,i}}{i\,!}\, f^{(i)}(0^+) \right) = D^{\alpha}_{*}\, f(t).
$$

From here, it can be easily recognised that $D_*^\alpha 1 \equiv 0$, $\alpha > 0$.

7.2 Discretizing a Fractional Derivative

In this section, we discuss how discretize a fractional derivative of general order $\alpha > 0$. For an overview on discretizations for fractional differential equations, see, for instance, [143, 158].

Let $f(t)$ be defined on $\Omega = [0, 1]$. We consider both the Riemann–Liouville fractional derivative $D^\alpha f(t)$ and the Caputo fractional derivative $D_*^\alpha f(t)$. In order to find a numerical approximation of these differential operators, having taken $n_t \in \mathbb{N}$, we construct the grid

$$t_i = i\,\tau, \quad i = 0, 1, \ldots, n_t, \qquad \tau = \frac{1}{n_t}.$$

Then, we approximate the fractional derivatives through the formula of Grünwald and Letnikov [143]. In particular, for the Riemann–Liouville fractional derivative we use the shifted Grünwald-Letnikov formula

$$D^\alpha f(t) \approx \frac{1}{\tau^\alpha} \sum_{i=0}^{n_t} g_{\alpha,i}\, f(t - (i-1)\tau), \tag{7.3}$$

where the coefficients $g_{\alpha,i}$ are given by

$$g_{\alpha,i} = \frac{\Gamma(i-\alpha)}{\Gamma(-\alpha)\Gamma(i+1)} = (-1)^i \binom{\alpha}{i}.$$

These can be computed by setting $g_{\alpha,0} = 1$, and then using the following recurrence formula:

$$g_{\alpha,i} = \left(1 - \frac{\alpha+1}{i}\right) g_{\alpha,i-1},$$

for $i = 1, 2, \ldots, n_t$ [143]. It is possible to prove that (7.3) is first order accurate, see [111, 112, 113].

Rewriting (7.3) in matrix form, we obtain that the discretization of the Riemann–Liouville fractional derivative is defined by

$$\mathfrak{L}_\alpha \boldsymbol{f},$$

with the vector $\boldsymbol{f}$ containing the evaluations of the function f at the grid points,

and the matrix $\mathfrak{L}_\alpha$ given by

$$\mathfrak{L}_\alpha = \frac{1}{\tau^\alpha} \begin{bmatrix} g_{\alpha,1} & g_{\alpha,0} & 0 & & & \\ g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & g_{\alpha,1} & g_{\alpha,0} & 0 \\ \vdots & & & g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} \\ g_{\alpha,n_t} & \cdots & \cdots & g_{\alpha,3} & g_{\alpha,2} & g_{\alpha,1} \end{bmatrix}. \tag{7.4}$$

We observe that the matrix $\mathfrak{L}_\alpha$ is a *Toeplitz matrix*, as defined in the next section.

We now consider the Caputo fractional derivative $D_*^\alpha f(t)$ of $f(t)$. Again, by using the Grünwald–Letnikov formula in its non-shifted form [143, 158], we obtain the following approximation:

$$D_*^\alpha f(t) \approx \frac{1}{\tau^\alpha} \sum_{i=0}^{n_t-1} g_{\alpha,i}\, f(t - i\,\tau).$$

As above, rewriting the previous expression in matrix form leads to the lower triangular Toeplitz matrix $\mathfrak{C}_\alpha$

$$\mathfrak{C}_\alpha = \frac{1}{\tau^\alpha} \begin{bmatrix} g_{\alpha,0} & 0 & & & \\ g_{\alpha,1} & g_{\alpha,0} & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ \vdots & & g_{\alpha,1} & g_{\alpha,0} & 0 \\ g_{\alpha,n_t-1} & \cdots & g_{\alpha,2} & g_{\alpha,1} & g_{\alpha,0} \end{bmatrix}. \tag{7.5}$$

We would like to note that we employ the non-shifted Grünwald–Letnikov formula as an approximation of the Caputo fractional derivative in order to take into account the discrepancy between the Caputo and the Riemann–Liouville fractional derivatives given in (7.2).

From (7.4) and from (7.5), we can realize that the matrices arising upon discretization of a fractional differential operator are (usually) dense. Since we are looking for a fine discretization in order to reduce the numerical error, one is usually dealing with very large, and very dense linear systems (that often cannot even be stored in computer memory). However, the discretized systems have a structure which we may exploit: as we mentioned above, the type of matrix we obtain upon this discretization of fractional derivatives of these types is a Toeplitz matrix. In the following section we will introduce the definition of Toeplitz matrix, and then generalize this notion to block-Toeplitz matrices whose blocks are again Toeplitz matrices. Then, we will derive optimal preconditioners for those type of matrices, and we will employ this method in the problems considered below.

7.3 Preconditioners for FDEs

In this section, we discuss the preconditioning techniques employed for approximating the discrete fractional differential operators described above. The preconditioners are derived by making use of the *Generalized Locally Toeplitz (GLT)* theory. For an overview on the latter, we refer the reader to [56, 57]. In addition, we refer the reader to Appendix A for a formal definition of the notions below.

In the following, we employ the notion of *multi-index* (or *d-index*), defined as a row vector $\mathbf{i} \in \mathbb{Z}^d$ with components $i_1, i_2, \ldots, i_d$, where $d \in \mathbb{N}$. Below, we denote $N(\mathbf{i}) = \prod_{j=1}^{d} i_j$, and write $\mathbf{i} \to \infty$ to indicate that $\min(\mathbf{i}) \to \infty$. In addition, any operation involving d-indices that has no meaning in the vector space $\mathbb{Z}^d$ will be interpreted in a componentwise sense.

We begin with the definition of d-level matrix. Given $d \in \mathbb{N}$, a matrix A of dimension n_A is a d-level matrix with level orders $n_1, n_2, \ldots, n_d$ if $n_A = n_1 n_2 \cdots n_d$, and it is partitioned into n_1^2 square blocks of size $\frac{n_A}{n_1}$, each of which is partitioned into n_2^2 blocks of size $\frac{n_A}{n_1 n_2}$, and so on until the last n_d^2 blocks of size 1.

We now give the definition of Toeplitz matrix. Given $2n_A - 1$ numbers a_i, $i = -n_A + 1, -n_A, \ldots, -1, 0, 1, \ldots, n_A - 2, n_A - 1$, the matrix A of dimension $n_A \times n_A$ such that

$$A_{i,l} = a_{i-l}, \quad i, l = 1, 2, \ldots, n_A$$

is called a Toeplitz matrix, where with $A_{i,l}$ we have denoted the element of position (i, l) of the matrix A. Specifically, we have

$$A = \begin{bmatrix} a_0 & a_{-1} & \ldots & a_{-n_A+1} \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{-1} \\ a_{n_A-1} & \ldots & a_1 & a_0 \end{bmatrix}.$$

From the definition, we have that a Toeplitz matrix is not necessarily square.

Then, given $d \in \mathbb{N}$, we can recursively define a *d-level Toeplitz matrix*. A matrix A is said to be a d-level Toeplitz if it can be written as

$$A = A_1 \otimes A_2,$$

where A_1 is a Toeplitz matrix, and A_2 is a $(d-1)$-level Toeplitz matrix. From the definition, a d-level Toeplitz matrix can be partitioned in blocks (each of which is a $(d-1)$-level Toeplitz matrix) that are constant along each diagonal. In the GLT theory, it is possible to associate a d-level Toeplitz matrix A to a function f, which is called the *generating function*; in this case, we say that the matrix A is *generated* by f.

From the definition of d-level Toeplitz matrix we can also define another class of matrices, namely, the *d-level circulant matrices*. We start with the definition of unilevel circulant matrix, and then generalize the notion to the multilevel case.

Given n_A numbers a_i, $i = 0, 1, \ldots, n_A - 1$, the matrix A of dimension $n_A \times n_A$ such that

$$A_{i,l} = a_{(i-l) \bmod n_A}, \quad i, l = 1, 2, \ldots, n_A$$

is called a circulant matrix. All circulant matrices are therefore Toeplitz, with an additional cyclic permutation property, namely

$$A = \begin{bmatrix} a_0 & a_{n_A-1} & \dots & a_1 \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n_A-1} \\ a_{n_A-1} & \dots & a_1 & a_0 \end{bmatrix}.$$

In addition, as for the Toeplitz case, we can define a matrix A to be a d-level circulant matrix if it can be written as

$$A = A_1 \otimes A_2,$$

where A_1 is a circulant matrix, and A_2 is a $(d-1)$-level circulant matrix. As for a d-level Toeplitz matrix, a d-level circulant matrix can be partitioned in blocks (each of which is a $(d-1)$-level circulant matrix) that are constant along each diagonal with an additional cyclic permutation property. In the following, we will employ particular circulant and d-level circulant matrices in our analysis. For this reason, given $\bar{n} \in \mathbb{N}$ we define the $\bar{n} \times \bar{n}$ matrix $C_{\bar{n}} = [c_{i,l}]_{i,l=1}^{\bar{n}}$, with

$$c_{i,l} = \begin{cases} 1 & \text{if } (i-l) \bmod \bar{n} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

In addition, given the vectors $\mathbf{n}_A \in \mathbb{N}^d$ and $\mathbf{j} \in \mathbb{Z}^d$, we define the d-level circulant matrix $C_{\mathbf{n}_A}^{\mathbf{j}} = C_{n_1}^{j_1} C_{n_2}^{j_2} \cdots C_{n_d}^{j_d}$, where $C_{n_i}^{j_i}$ is the j_i-th power of the matrix C_{n_i} defined above.

Circulant and d-level circulant matrices are important classes of matrices in relation to finding optimal preconditioners for systems arising upon discretization of FDEs. In particular, it is possible to prove that every d-level circulant matrix A can be written as a linear combination of the matrices $C_{\mathbf{n}_A}^{\mathbf{j}}$ defined above, see [57, Section 3.4]. As a consequence, we have that the set of all d-level circulant matrix is a commutative ring under matrix addition and multiplication. In addition, it is possible to prove that every d-level circulant matrix A is diagonalizable by the discrete Fourier transform (a very important property from the point of view of preconditioning), see [57, Section 3.4].

As we mentioned at the beginning of this section, we employ GLT theory to motivate our preconditioners for the problems considered here. An important notion in the GLT theory is that of a *matrix-sequence*. A (d-level) matrix-sequence is a sequence of square matrices $\{A_j\}_j$ (respectively, $\{A_{\mathbf{j}}\}_j$), whose size n_{A_j} (respectively, $N(\mathbf{j})$ with $\mathbf{j} = \mathbf{j}(j)$) tends to infinity as $j \to \infty$.

In order to find suitable preconditioners for the systems we consider below, we have to introduce also the notion of *approximating class of sequences (a.c.s.)*. Given a matrix-sequence $\{A_j\}_j$, a sequence of matrix-sequences $\{\{\widetilde{A}_{j,m}\}_j\}_m$ is an approximating class of sequences for $\{A_j\}_j$ if the difference between A_j and $\widetilde{A}_{j,m}$ is the sum of a matrix of low rank and a matrix of small norm. A formal definition of approximating class of sequences is given in Definition 1 in Appendix A.

As mentioned above, approximating classes of sequences are employed in order

to build preconditioners for the problems examined below. However, we need some sort of "measure" in order to understand how good the preconditioners are. For this reason, we will find useful the notion of *clusters*. Given a sequence of matrices $\{A_j\}_j$ and a subset X of the complex plane, we say that $\{A_j\}_j$ is *weakly clustered* at X if the number of the eigenvalues of A_j that are not "close enough" to X is bounded above by the dimension n_{A_j} of A_j.

An important class of matrix sequences is a *d-level GLT sequence*. The sequences belonging to this class are such that they can be associated in a certain sense to a measurable function κ, which is called the *symbol* of the sequence. A formal definition of d-level GLT sequence is given in Definition 4 in Appendix A.

All the previous notions can be used in order to find suitable approximations of Toeplitz and multilevel Toeplitz matrices. In particular, in the following we are going to employ multilevel circulant preconditioners as an approximation of multilevel Toeplitz matrices. A preconditioner of this type can be derived by firstly finding a unilevel circulant approximation of an arbitrary unilevel Toeplitz matrix. Given a unilevel Toeplitz matrix $A \in \mathbb{R}^{n_A \times n_A}$, we employ the circulant approximation proposed for the first time in [32] (also called the T. Chan preconditioner for A). More specifically, we define the optimal circulant approximation of A, as the solution of the following optimization problem:

$$C_1(A) = \min_{C_{\bar{n}} \in \mathcal{C}_{\bar{n}}} \|C_{\bar{n}} - A\|_F,$$

where $\mathcal{C}_{\bar{n}}$ is the the set of all $n_A \times n_A$ circulant matrices, and $\|\cdot\|_F$ the *Frobenius norm*. It turns out that the previous problem admits the following closed form solution:

$$c_i = \frac{(n_A - i)\, a_i + i\, a_{-n_A+i}}{n}, \quad i \in \{0, 1, \ldots, n_A - 1\}.$$

Then, we can write $C_1(A) = F_{n_A}^* \Lambda_{n_A} F_{n_A}$, where F_{n_A} is the (scaled) discrete Fourier transform of size n_A and Λ_{n_A} is a diagonal matrix containing the eigenvalues of $C_1(A)$, which can be computed as $\Lambda_{n_A} = \mathrm{diag}(F_{n_A} \mathbf{c}_1)$, where $\mathbf{c}_1$ is the first column of $C_1(A)$. Other unilevel circulant approximations are possible, such as those proposed in [30, 31, 176], however, the T. Chan preconditioner seems (empirically) to behave better for the problem under consideration.

From the unilevel circulant preconditioner we can then find a preconditioner for multilevel Toeplitz matrices. In fact, recalling that a d-level Toeplitz matrix can be defined as a Kronecker product of unilevel and $(d-1)$-level Toeplitz matrices, we can recursively define the preconditioner of a d-level Toeplitz matrix $A = A_1 \otimes A_2 \otimes \ldots \otimes A_d$ as

$$\begin{aligned} A &\approx C_1(A_1) \otimes C_1(A_2) \otimes \ldots \otimes C_1(A_d) \\ &= (F_{n_{A_1}}^* \otimes F_{n_{A_2}}^* \otimes \ldots \otimes F_{n_{A_d}}^*) \Lambda_{\mathbf{d}} (F_{n_{A_1}} \otimes F_{n_{A_2}} \otimes \ldots \otimes F_{n_{A_d}}) =: C_d(A), \end{aligned}$$

where $\Lambda_{\mathbf{d}}$ is the following diagonal matrix:

$$\Lambda_{\mathbf{d}} = \Lambda_1 \otimes \Lambda_2 \otimes \ldots \otimes \Lambda_d.$$

The previous approximation of a d-level Toeplitz matrix will be employed in the

following as an optimal preconditioner for the problems considered in this chapter.

In order to show how to derive the previous expression, we consider the case of a 2-level Toeplitz matrix A. In this case, we have

$$A = \begin{bmatrix} A_0 & A_{-1} & \dots & A_{-m_1+1} & A_{-m_1} \\ A_1 & A_0 & \ddots & & A_{-m_1+1} \\ \vdots & A_1 & \ddots & \ddots & \vdots \\ A_{m_1-1} & & \ddots & \ddots & A_{-1} \\ A_{m_1} & A_{m_1-1} & \dots & A_1 & A_0 \end{bmatrix},$$

where each A_i, for $i = -m_1, -m_1+1, \dots, m_1-1, m_1$, is a unilevel Toeplitz matrix. Then, if we employ the T. Chan preconditioner $C_1(A_i) = F_{m_1}^* \Lambda_i F_{m_1}$ in order to approximate each block A_i, for $i = -m_1, -m_1+1, \dots, m_1-1, m_1$, we have the following approximation:

$$A \approx \begin{bmatrix} F_{m_1}^* \Lambda_0 F_{m_1} & F_{m_1}^* \Lambda_{-1} F_{m_1} & \dots & F_{m_1}^* \Lambda_{-m_1} F_{m_1} \\ F_{m_1}^* \Lambda_1 F_{m_1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & F_{m_1}^* \Lambda_{-1} F_{m_1} \\ F_{m_1}^* \Lambda_{m_1} F_{m_1} & \dots & F_{m_1}^* \Lambda_1 F_{m_1} & F_{m_1}^* \Lambda_0 F_{m_1} \end{bmatrix},$$

or equivalently

$$A \approx (I_{m_1} \otimes F_{m_1}^*) \begin{bmatrix} \Lambda_0 & \Lambda_{-1} & \dots & \Lambda_{-m_1} \\ \Lambda_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Lambda_{-1} \\ \Lambda_{m_1} & \dots & \Lambda_1 & \Lambda_0 \end{bmatrix} (I_{m_1} \otimes F_{m_1}),$$

where I_{m_1} is the identity matrix of dimension m_1. Recalling that each matrix Λ_i, for $i = -m_1, -m_1+1, \dots, m_1$, is diagonal, we can employ a suitable permutation P and rewrite

$$A \approx (I_{m_1} \otimes F_{m_1}^*) P^\top \begin{bmatrix} \bar{A}_1 & & & \\ & \bar{A}_2 & & \\ & & \ddots & \\ & & & \bar{A}_{m_2} \end{bmatrix} P \, (I_{m_1} \otimes F_{m_1}),$$

where each block $\bar{A}_i$, for $i = 1, 2, \dots, m_2$, is a unilevel Toeplitz matrix. Therefore, by approximating again each block with the T. Chan preconditioner, we can rewrite:

$$A \approx (I_{m_1} \otimes F_{m_1}^*) P^\top (I_{m_2} \otimes F_{m_2}^*) \begin{bmatrix} \bar{\Lambda}_1 & & & \\ & \bar{\Lambda}_2 & & \\ & & \ddots & \\ & & & \bar{\Lambda}_{m_2} \end{bmatrix} (I_{m_2} \otimes F_{m_2}) P \, (I_{m_1} \otimes F_{m_1}).$$

Then, by observing that $(I_{m_2} \otimes F_{m_2}) P = P^\top (F_{m_2} \otimes I_{m_2})$, we derive the following

approximation:

$$A \approx (I_{m_1} \otimes F^*_{m_1})(F^*_{m_2} \otimes I_{m_2})\Lambda(F_{m_2} \otimes I_{m_2})(I_{m_1} \otimes F_{m_1}),$$

where Λ is a diagonal matrix obtained permuting the elements on the diagonal of the previous approximation. Finally, observing that $(F_{m_2} \otimes I_{m_2})(I_{m_1} \otimes F_{m_1}) = F_{m_2} \otimes F_{m_1}$, we obtain the T. Chan-based 2-level circulant approximation.

From here, by recursion one can easily generalize the previous strategy to d-level Toeplitz matrices.

We would like to note that, although a unilevel circulant approximation of a unilevel Toeplitz matrix is an optimal preconditioner, in the multilevel case this is not true, see for instance [163].

7.4 Optimal Control of FDEs

In this section we introduce the optimal control of a fractional differential equation. This problem is defined by (1.2)–(1.3) where the differential operator $\mathcal{D}$ is a fractional differential equation. In our study we will focus on the control of the fractional diffusion equation [39], so our problem reads as

$$\min_{v,u} \quad \frac{1}{2}\int_0^{t_f}\int_\Omega |v - v_d|^2 \,\mathrm{d}\Omega\mathrm{d}t + \frac{\beta}{2}\int_0^{t_f}\int_\Omega |u|^2 \,\mathrm{d}\Omega\mathrm{d}t \tag{7.6}$$

subject to

$$\begin{cases} ({}^tD^\alpha_* - {}^{x_1}_{\mathrm{R}}D^{\gamma_1} - {}^{x_2}_{\mathrm{R}}D^{\gamma_2})\, v(x_1,x_2,t) + u(x_1,x_2,t) = 0 & \text{in } \Omega \times (0,t_f), \\ v(x_1,x_2,t) = g(x_1,x_2,t) & \text{on } \partial\Omega \times (0,t_f), \\ v(x_1,x_2,0) = v_0(x_1,x_2) & \text{in } \Omega, \end{cases} \tag{7.7}$$

where $\Omega = \prod_{i=1}^2 (a_i, b_i) \subset \mathbb{R}^2$, and $t_f > 0$ the final time. Here, ${}^tD^\alpha_*$ is the Caputo fractional derivative of order α, with $\alpha \in (0,1]$, and ${}^{x_i}_{\mathrm{R}}D^{\gamma_i}$ is the symmetric Riesz derivative of order $\gamma_i \in (1,2]$, for $i = 1,2$, with the subscripts x_i denoting the variable we are differentiating with respect to. The functions g and v_0 are known. The differential operator considered here is a specific problem, but we devise methods that are readily generalized to more complex FDEs.

As opposed to all the previous chapters, in the problems we consider here we also include constraints on the state and control variables. Specifically, the problems we want to solve here is given by (7.6)–(7.7), with the following additional constraints:

$$\begin{cases} v_{\min}(x,t) \leqslant v(x,t) \leqslant v_{\max}(x,t), \\ u_{\min}(x,t) \leqslant u(x,t) \leqslant u_{\max}(x,t), \end{cases} \tag{7.8}$$

where the functions $v_{\min}$, $v_{\max}$, $u_{\min}$, and $u_{\max}$ are given.

Due to the presence of additional algebraic constraints on the variables, in order to find an approximation of the solution of (7.6)–(7.8) we have to employ a non-linear iteration. Before showing the method employed for solving the problems considered, we have to derive the discrete first-order optimality conditions, that will be the topic of the next section. We want to mention that the problems

considered here are only two-dimensional in space, but it is perfectly reasonable to consider problems in different numbers of spatial dimensions; for this reason, we would like to emphasize that the methodology in this chapter could be readily tailored to such problems in higher dimensions.

7.4.1 Discretize-Then-Optimize Approach

We derive now the first-order optimality conditions of the problem (7.6)–(7.8). As opposed to the previous chapters, here we adopt a discretize-then-optimize strategy, described in Section 1.3.2.

Let n_t, n_{x_1}, n_{x_2} be the number of points on the time interval, the x_1-axis and the x_2-axis respectively. We discretize the L^2-norm using the trapezoidal rule, the Caputo derivative with the matrix $\mathfrak{C}_\alpha$ in (7.5), and ${}^{x_i}_{\mathrm{R}}D^{\gamma_i}$, $i = 1, 2$, with the matrix

$$\mathfrak{L}^{\mathrm{R}}_{\gamma_i} = \frac{-1}{2\cos(\frac{\alpha\pi}{2})}(\mathfrak{L}_{\gamma_i} + \mathfrak{L}^{\top}_{\gamma_i}),$$

with $\mathfrak{L}_{\gamma_i}$ defined by (7.4). With this notation we have

$${}^{x_1}_{\mathrm{R}}D^{\gamma_1} + {}^{x_2}_{\mathrm{R}}D^{\gamma_2} \approx \mathfrak{L}^{\mathrm{R}}_{\gamma_1} \otimes I_{n_{x_2}} + I_{n_{x_1}} \otimes \mathfrak{L}^{\mathrm{R}}_{\gamma_2} =: \mathfrak{L}_{\gamma_1,\gamma_2},$$

where the matrix $I_{n_{x_i}}$ denotes the identity matrix of size n_{x_i}, for $i = 1, 2$; in addition, we have

$${}^{t}D^{\alpha}_{*} - {}^{x_1}_{\mathrm{R}}D^{\gamma_1} - {}^{x_2}_{\mathrm{R}}D^{\gamma_2} \approx \mathfrak{C}_\alpha \otimes I_N - I_{n_t} \otimes \mathfrak{L}_{\gamma_1,\gamma_2} =: \mathfrak{B},$$

where $N = n_{x_1} n_{x_2}$ is the total number of points in the spatial domain. Finally, the discrete formulation of (7.6)–(7.8) reads as

$$\min_{\boldsymbol{v},\boldsymbol{u}} \quad \frac{1}{2}\left(\boldsymbol{v} - \boldsymbol{v}_d\right)^{\top} M_1 \left(\boldsymbol{v} - \boldsymbol{v}_d\right) + \frac{1}{2}\boldsymbol{u}^{\top} M_2 \boldsymbol{u} \tag{7.9}$$

subject to

$$\begin{cases} \mathfrak{B}\boldsymbol{v} + \boldsymbol{u} = \boldsymbol{0}, \\ \boldsymbol{v}_{\min} \leqslant \boldsymbol{v} \leqslant \boldsymbol{v}_{\max}, \quad \boldsymbol{u}_{\min} \leqslant \boldsymbol{u} \leqslant \boldsymbol{u}_{\max}, \end{cases} \tag{7.10}$$

where the matrices M_1, M_2 contain the weights of the quadrature rule, so they are given by

$$M_1 = \begin{bmatrix} I_N & & & \\ & \ddots & & \\ & & I_N & \\ & & & \frac{1}{2}I_N \end{bmatrix}, \qquad M_2 = \beta M_1.$$

Note that we have eliminated the initial condition in our discretization, otherwise the first diagonal-block of the matrix M_1 would be multiplied by a factor of $\frac{1}{2}$ for the trapezoidal rule.

It is worth noting that the first-order optimality condition of the problem

(7.9)–(7.10) without box constraints would read in matrix form as follows:

$$\begin{bmatrix} M_1 & 0 & \mathfrak{B}^\top \\ 0 & M_2 & I_{\bar{N}} \\ \mathfrak{B} & I_{\bar{N}} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{u} \\ \boldsymbol{\zeta} \end{bmatrix} = \begin{bmatrix} M_1 \boldsymbol{v}_d \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

where $I_{\bar{N}}$ is the identity matrix of dimension $\bar{N}$, with $\bar{N} = Nn_t$. Then, the corresponding Schur complement reads as:

$$S_{\text{unc}} = \mathfrak{B} M_1^{-1} \mathfrak{B}^\top + M_2^{-1}. \tag{7.11}$$

We recall that, in our case, the matrix $\mathfrak{B}$ is a 3-level Toeplitz matrix. In particular, since the matrix $\mathfrak{B}$ is dense, the previous Schur complement will be also dense. However, the strategy devised in Section 7.6 for the constrained problem can be readily adapted in order to solve a system involving the Schur complement S_{unc} of the unconstrained problem.

In the analysis below, we will suppose that the d-level matrices $\mathfrak{B}$, M_1, and M_2 of dimension $\bar{N} \times \bar{N}$ have a spectral norm uniformly bounded with respect to $\bar{N}$, and are a d-level GLT sequences, with symbols κ, κ_1, and κ_2, respectively. We refer the reader to Assumption 2 in Appendix A for the formal assumptions used in [146] to prove the effectiveness of the preconditioner described below. We would like to mention that the assumptions on the matrix $\mathfrak{B}$ can be ensured by multiplying the discretized FDE constraint operator by a factor depending on grid size.

As we mentioned above, due to the algebraic constraints, in order to find an approximate solution of the problem we have to run a non-linear process. In Section 1.4, we discussed some techniques that can be employed for solving a problem of the form (7.9)–(7.10). Specifically, we discussed Active Set (AS) methods and Interior Point methods (IPMs). However, problems of the form (7.9)–(7.10) are usually highly structured, and this structure must be exploited, given that the problem size increases indefinitely as one refines the discretization. As we discussed in Section 1.4, at each AS iteration only a subset of the constraints (7.10) is considered. This in turn implies that we would lose the Toeplitz structure. In fact, any optimization method whose sub-problems arise by projecting the variables of the problem in a subspace would face this issue.

On the other hand, IPMs deal with the inequality constraints by introducing logarithmic barriers in the objective. Then, at every IPM iteration, one approximately solves the optimality conditions of the barrier sub-problem using a Newton method. However, the sequence of Hessian matrices arising from the logarithmic barriers is not a GLT sequence, see [146, Section 3]. As a consequence, the system matrix of the optimality conditions of each barrier sub-problem, within the IPM, will not be in the GLT class.

We therefore must consider a strategy that preserves the multilevel Toeplitz structure in order for our preconditioning approach to be effective. The following section will describe the strategy employed for finding a solution of (7.9)–(7.10).

7.5 Alternating Direction Method of Multipliers

In order to overcome the previous issues, we employ an Alternating Direction Method of Multipliers (ADMM) (originally proposed in the work [55, 60]), which separates the equality from the inequality constraints, thus allowing us to preserve the Toeplitz structure of the equality constraints in (7.10). For a more comprehensive discussion on ADMM, we refer the reader to [21, 129]. We would like to mention that, although we are able to preserve the structure of the problem by employing ADMM, the convergence of the method can be relatively slow (see, e.g., [21]). In fact, one can prove linear convergence of ADMM under certain assumptions on the problem under consideration (such as strong convexity, see [38] and the references therein). For this reason, this method is not suitable for finding very accurate solutions. Nevertheless, a 4-digit accurate solution can generally be found in reasonable CPU time. In addition, since the linear system solved at each ADMM iteration is unchanged, if we find a suitable preconditioner that exploits the problem structure, we only need to compute it once.

In order to present the ADMM algorithm, we introduce some auxiliary variables $\boldsymbol{z}_v$ and $\boldsymbol{z}_u$, and rewrite (7.9)–(7.10) as follows:

$$\min_{\boldsymbol{v},\boldsymbol{u},\boldsymbol{z}_v,\boldsymbol{z}_u} \quad \frac{1}{2}\left(\boldsymbol{v}-\boldsymbol{v}_d\right)^\top M_1\left(\boldsymbol{v}-\boldsymbol{v}_d\right)+\frac{1}{2}\boldsymbol{u}^\top M_2\boldsymbol{u} \tag{7.12}$$

subject to

$$\begin{cases} \mathfrak{B}\boldsymbol{v}+\boldsymbol{u}=\boldsymbol{0}, \\ \boldsymbol{v}=\boldsymbol{z}_v, \qquad \boldsymbol{u}=\boldsymbol{z}_u, \\ \boldsymbol{v}_{\min}\leqslant\boldsymbol{z}_v\leqslant\boldsymbol{v}_{\max}, \quad \boldsymbol{u}_{\min}\leqslant\boldsymbol{z}_u\leqslant\boldsymbol{u}_{\max}. \end{cases} \tag{7.13}$$

Next, we introduce the adjoint variables $\boldsymbol{\zeta}$, $\boldsymbol{w}_v$, and $\boldsymbol{w}_u$ for each equality constraint in (7.13), and consider the following augmented Lagrangian function

$$\begin{aligned} \mathcal{L}_\delta(\boldsymbol{v},\boldsymbol{u},\boldsymbol{z}_v,\boldsymbol{z}_u,\boldsymbol{\zeta},\boldsymbol{w}_v,\boldsymbol{w}_u) = {} & \frac{1}{2}\left(\boldsymbol{v}-\boldsymbol{v}_d\right)^\top M_1\left(\boldsymbol{v}-\boldsymbol{v}_d\right)+\frac{1}{2}\boldsymbol{u}^\top M_2\boldsymbol{u} \\ & +\boldsymbol{\zeta}^\top(\mathfrak{B}\boldsymbol{v}+\boldsymbol{u})+\boldsymbol{w}_v^\top(\boldsymbol{v}-\boldsymbol{z}_v)+\boldsymbol{w}_u^\top(\boldsymbol{u}-\boldsymbol{z}_u) \\ & +\frac{1}{2\delta}(\|\mathfrak{B}\boldsymbol{v}+\boldsymbol{u}\|_2^2+\|\boldsymbol{v}-\boldsymbol{z}_v\|_2^2+\|\boldsymbol{u}-\boldsymbol{z}_u\|_2^2), \end{aligned}$$

with $\delta>0$. An ADMM algorithm applied to solve problem (7.12)–(7.13) is given in Algorithm 15. As we see from Algorithm 15, the method fixes some of the variables and minimizes the problem only for two of them (first and second steps), and then updates the remaining variables with a recurrence formula (third and fourth steps). In practice, ADMM splits the problem into smaller subproblems, solves the subproblems with respect to some of the variables, and then updates the remaining ones. We omit further details of the algorithm. The reader is referred to [21] for a basic proof of convergence of Algorithm 15, as well as a detailed overview of ADMM. We should mention that the step-length ρ in Algorithm 15 plays an important role in the convergence behavior of ADMM. In fact, convergence of Algorithm 15 is guaranteed for any $\rho\in(0,\frac{\sqrt{5}+1}{2})$ (see [59]).

Algorithm 15 Alternating Direction Method of Multipliers

Choose $\boldsymbol{v}^{(0)}$, $\boldsymbol{u}^{(0)}$, $\boldsymbol{z}_v^{(0)}$, $\boldsymbol{z}_u^{(0)}$, $\boldsymbol{\zeta}^{(0)}$, $\boldsymbol{w}_v^{(0)}$, $\boldsymbol{w}_u^{(0)}$

for $k = 1$ **until** convergence, **do**

Solve $(\boldsymbol{v}^{(k+1)}, \boldsymbol{u}^{(k+1)}) = \underset{\boldsymbol{v},\boldsymbol{u}}{\operatorname{argmin}} \mathcal{L}_\delta(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{z}_v^{(k)}, \boldsymbol{z}_u^{(k)}, \boldsymbol{\zeta}^{(k)}, \boldsymbol{w}_v^{(k)}, \boldsymbol{w}_u^{(k)})$

Solve $(\boldsymbol{z}_v^{(k+1)}, \boldsymbol{z}_u^{(k+1)}) = \underset{\substack{\boldsymbol{z}_v \in [\boldsymbol{v}_{\min}, \boldsymbol{v}_{\max}], \\ \boldsymbol{z}_u \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}]}}{\operatorname{argmin}} \mathcal{L}_\delta(\boldsymbol{v}^{(k+1)}, \boldsymbol{u}^{(k+1)}, \boldsymbol{z}_v, \boldsymbol{z}_u, \boldsymbol{\zeta}^{(k)}, \boldsymbol{w}_v^{(k)}, \boldsymbol{w}_u^{(k)})$

Set $\boldsymbol{\zeta}^{(k+1)} = \boldsymbol{\zeta}^{(k)} + \dfrac{\rho}{\delta}(\mathfrak{B}\boldsymbol{v}^{(k+1)} + \boldsymbol{u}^{(k+1)})$

Set $(\boldsymbol{w}_v^{(k+1)}, \boldsymbol{w}_u^{(k+1)}) = (\boldsymbol{w}_v^{(k)}, \boldsymbol{w}_u^{(k)}) + \dfrac{\rho}{\delta}\left(\boldsymbol{v}^{(k+1)} - \boldsymbol{z}_v^{(k+1)}, \boldsymbol{u}^{(k+1)} - \boldsymbol{z}_u^{(k+1)}\right)$

end for

One can observe that the most challenging step of Algorithm 15, is that of solving the following problem:

$$(\boldsymbol{v}^{(k+1)}, \boldsymbol{u}^{(k+1)}) = \underset{\boldsymbol{v},\boldsymbol{u}}{\operatorname{argmin}} \mathcal{L}_\delta(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{z}_v^{(k)}, \boldsymbol{z}_u^{(k)}, \boldsymbol{\zeta}^{(k)}, \boldsymbol{w}_v^{(k)}, \boldsymbol{w}_u^{(k)}). \tag{7.14}$$

The optimality conditions of (7.14), at iteration k, read as follows:

$$\begin{bmatrix} M_1 + \frac{1}{\delta}(\mathfrak{B}^\top\mathfrak{B} + I_{\bar{N}}) & \frac{1}{\delta}\mathfrak{B}^\top \\ \frac{1}{\delta}\mathfrak{B} & M_2 + \frac{2}{\delta}I_{\bar{N}} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{b}}_1^{(k)} \\ \bar{\mathbf{b}}_2^{(k)} \end{bmatrix}, \tag{7.15}$$

where $I_{\bar{N}}$ is the identity matrix of dimension $\bar{N} = Nn_t$, and $\bar{\mathbf{b}}_1^{(k)}$ and $\bar{\mathbf{b}}_2^{(k)}$ take into account the non-linear residuals.

Solving the previous system directly is not a good idea in our case, since its coefficient matrix is not expected to be cheap or convenient to work with. Instead, we can merge the first and the third steps in Algorithm 15 to obtain a more flexible saddle point system. More specifically, we substitute $\boldsymbol{\zeta} = \boldsymbol{\zeta}^{(k)} + \dfrac{\rho}{\delta}(\mathfrak{B}\boldsymbol{v} + \boldsymbol{u})$ into (7.15), and rewrite the optimality conditions for the first and third lines of Algorithm 15 as follows:

$$\begin{bmatrix} \rho(M_1 + \frac{1}{\delta}I_{\bar{N}}) & 0 & \mathfrak{B}^\top \\ 0 & \rho(M_2 + \frac{1}{\delta}I_{\bar{N}}) & I_{\bar{N}} \\ \mathfrak{B} & I_{\bar{N}} & -\frac{\delta}{\rho}I_{\bar{N}} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{u} \\ \boldsymbol{\zeta} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^{(k)} \\ \mathbf{b}_2^{(k)} \\ \mathbf{b}_3^{(k)} \end{bmatrix}, \tag{7.16}$$

where as above the right-hand side accounts for the non-linear residuals. Specifically, we have

$$\begin{aligned} \mathbf{b}_1^{(k)} &= \rho(M_1\boldsymbol{v}_d - \boldsymbol{w}_v^{(k)} + \tfrac{1}{\delta}\boldsymbol{z}_v^{(k)}) + (1-\rho)\mathfrak{B}^\top\boldsymbol{\zeta}^{(k)}, \\ \mathbf{b}_2^{(k)} &= \rho(-\boldsymbol{w}_u^{(k)} + \tfrac{1}{\delta}\boldsymbol{z}_u^{(k)}) + (1-\rho)\boldsymbol{\zeta}^{(k)}, \\ \mathbf{b}_3^{(k)} &= -\tfrac{\delta}{\rho}\boldsymbol{\zeta}^{(k)}. \end{aligned}$$

System (7.16) presents a saddle-point structure. However, instead of employ-

ing the saddle-point theory described in Section 2.10, since the blocks on the main diagonal are easy to invert (as each block on the main diagonal is a scaled identity matrix) we rather form the normal equations, and then derive a preconditioner for the system so obtained to be employed within preconditioned CG. Specifically, pivoting the second and then the third block equation of this system, yields:

$$\begin{aligned} \boldsymbol{u} &= \left(\rho\left(M_2 + \tfrac{1}{\delta} I_{\bar{N}}\right)\right)^{-1} \left(-\boldsymbol{\zeta} - \rho \boldsymbol{w}_u^{(k)} + \tfrac{\rho}{\delta} \boldsymbol{z}_u^{(k)} + (1-\rho)\boldsymbol{\zeta}^{(k)}\right), \\ \boldsymbol{\zeta} &= \left(\left(\rho M_2 + \tfrac{\rho}{\delta} I_{\bar{N}}\right)^{-1} + \tfrac{\delta}{\rho} I_{\bar{N}}\right)^{-1} (\mathfrak{B}\boldsymbol{v} + \mathbf{b}_4^{(k)}), \end{aligned} \tag{7.17}$$

where

$$\mathbf{b}_4^{(k)} = \frac{\delta}{\rho}\boldsymbol{\zeta}^{(k)} - \left(\rho\left(M_2 + \frac{1}{\delta} I_{\bar{N}}\right)\right)^{-1} \left(-\rho \boldsymbol{w}_u^{(k)} + \frac{\rho}{\delta}\boldsymbol{z}_u^{(k)} + (1-\rho)\boldsymbol{\zeta}^{(k)}\right).$$

Then, we can write the normal equations as follows:

$$\begin{aligned} S\boldsymbol{v} = \; & \rho\left(M_1 \boldsymbol{v}_d - \boldsymbol{w}_v^{(k)} + \frac{1}{\delta}\boldsymbol{z}_v^{(k)}\right) + (1-\rho)\mathfrak{B}^\top \boldsymbol{\zeta}^{(k)} \\ & -\mathfrak{B}^\top \left(\left(\rho M_2 + \frac{\rho}{\delta} I_{\bar{N}}\right)^{-1} + \frac{\delta}{\rho} I_{\bar{N}}\right)^{-1} \mathbf{b}_4^{(k)}, \end{aligned} \tag{7.18}$$

where the Schur complement S is given by

$$S = \rho\left(M_1 + \frac{1}{\delta} I_{\bar{N}}\right) + \mathfrak{B}^\top \left(\left(\rho M_2 + \frac{\rho}{\delta} I_{\bar{N}}\right)^{-1} + \frac{\delta}{\rho} I_{\bar{N}}\right)^{-1} \mathfrak{B}. \tag{7.19}$$

By solving the system (7.18) and then updating the current approximations of $\boldsymbol{u}$ and $\boldsymbol{\zeta}$ by employing (7.17), one is able to find approximations of the solutions of the first and third step in Algorithm 15. As the fourth step in Algorithm 15 is trivial, the only remaining step to solve is the following minimization problem:

$$(\boldsymbol{z}_v^{(k+1)}, \boldsymbol{z}_u^{(k+1)}) = \underset{\substack{\boldsymbol{z}_v \in [\boldsymbol{v}_{\min}, \boldsymbol{v}_{\max}], \\ \boldsymbol{z}_u \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}]}}{\operatorname{argmin}} \mathcal{L}_\delta(\boldsymbol{v}^{(k+1)}, \boldsymbol{u}^{(k+1)}, \boldsymbol{z}_v, \boldsymbol{z}_u, \boldsymbol{\zeta}^{(k)}, \boldsymbol{w}_v^{(k)}, \boldsymbol{w}_u^{(k)}).$$

However, we would like to mention that the previous problem has a closed form solution. More specifically, we perform the optimization by ignoring the box constraints and then projecting the solution onto the box.

As we discussed above, the most difficult task in Algorithm 15 is solving the normal equations (7.18). In the following, we are going to derive an optimal preconditioner for the system to be solved at each ADMM iteration by employing the circulant preconditioner discussed in Section 7.3.

7.6 Preconditioning Approach

In this section, we describe the strategy adopted in order to solve the system (7.18), with the Schur complement S defined as in (7.19). In order to find a preconditioner for this system, we employ GLT theory, as follows. For sake of clarity, we avoid all the technicalities, and refer the reader to [146] for a rigorous derivation of the proposed preconditioners.

We recall that, for the problem under examination, the matrix $\mathfrak{B}$ is a 3-level Toeplitz matrix. We observe that this is because the spatial domain Ω is a subset of $\mathbb{R}^2$. More generally, if the spatial domain Ω is of dimension d, the matrix $\mathfrak{B}$ will be a $(d+1)$-level Toeplitz matrix. Then, under suitable assumption, it is possible to find three sequences of GLT sequences

$$\{\widetilde{\mathfrak{B}}_{\mathbf{j}}\}_{\bar{N}} \xrightarrow{\text{a.c.s.}} \{\mathfrak{B}\}_{\bar{N}}, \qquad \{\widetilde{M}_{1_{\mathbf{j}}}\}_{\bar{N}} \xrightarrow{\text{a.c.s.}} \{M_1\}_{\bar{N}}, \qquad \{\widetilde{M}_{2_{\mathbf{j}}}\}_{\bar{N}} \xrightarrow{\text{a.c.s.}} \{M_2\}_{\bar{N}},$$

with symbols $\{\widetilde{\mathfrak{B}}_{\mathbf{j}}\}_{\bar{N}} \sim_{GLT} \kappa_{\bar{N}}$, $\{\widetilde{M}_{1_{\mathbf{j}}}\}_{\bar{N}} \sim_{GLT} \kappa_{1_{\bar{N}}}$, $\{\widetilde{M}_{2_{\mathbf{j}}}\}_{\bar{N}} \sim_{GLT} \kappa_{2_{\bar{N}}}$ such that $\kappa_{\bar{N}} \to \kappa$, $\kappa_{1_{\bar{N}}} \to \kappa_1$, and $\kappa_{2_{\bar{N}}} \to \kappa_2$, in measure, see [146, Proposition 3.2].

Then, given the three sequences of GLT sequences $\{\widetilde{\mathfrak{B}}_{\mathbf{j}}\}_{\bar{N}}$, $\{\widetilde{M}_{1_{\mathbf{j}}}\}_{\bar{N}}$, $\{\widetilde{M}_{2_{\mathbf{j}}}\}_{\bar{N}}$, we are able to find a suitable approximation of the matrix S, as follows. Starting from the expression of S in (7.19), we approximate each matrix in this expression with the corresponding GLT sequences above, and obtain the following approximation for the Schur complement S:

$$\widetilde{S} = \rho\left(\widetilde{M}_{1_{\mathbf{j}}} + \frac{1}{\delta} I_{\bar{N}}\right) + \widetilde{\mathfrak{B}}_{\mathbf{j}}^{\top}\left(\left(\rho\widetilde{M}_{2_{\mathbf{j}}} + \frac{\rho}{\delta} I_{\bar{N}}\right)^{-1} + \frac{\delta}{\rho} I_{\bar{N}}\right)^{-1} \widetilde{\mathfrak{B}}_{\mathbf{j}}. \tag{7.20}$$

It is possible to prove that the approximation $\widetilde{S}$ is weakly clustered at 1, and that the eigenvalues of the matrix $\widetilde{S}^{-1}S$ lie in an interval of the form $[\frac{1}{c_S}, c_S]$, where c_S is a positive constant uniformly bounded with respect to the size of the problem $\bar{N}$, see [146, Theorem 3.3]. From here, we can imply that, if we employ $\widetilde{S}$ as a preconditioner, the number of PCG iterations required for convergence is independent of the grid size (but it may depend on other parameters of the problem).

Remark 9. *The previous strategy is readily tailored in order to solve FDE-constrained optimization problems of the form* (7.6)–(7.7)*, when no additional box constraint are imposed. In fact, given the GLT sequences* $\{\widetilde{\mathfrak{B}}_{\mathbf{j}}\}_{\bar{N}}$, $\{\widetilde{M}_{1_{\mathbf{j}}}\}_{\bar{N}}$, $\{\widetilde{M}_{2_{\mathbf{j}}}\}_{\bar{N}}$*, if we substitute these sequences in place of the corresponding matrix in* (7.11)*, we still have a good approximation of* S_{unc}*.*

It is worth noting that our assumptions hold for a wide range of problems. In addition, it is possible to find easy-to-invert sequences $\{\widetilde{\mathfrak{B}}_{\mathbf{j}}\}_{\bar{N}}$, $\{\widetilde{M}_{1_{\mathbf{j}}}\}_{\bar{N}}$, $\{\widetilde{M}_{2_{\mathbf{j}}}\}_{\bar{N}}$. As we mentioned above, for such sequences the matrix $\widetilde{S}$ defined as in (7.20) will result in a preconditioned matrix $\widetilde{S}^{-1}S$ with a weak cluster at 1. However, in general the matrix $\widetilde{S}$ may be not easy to invert nor apply. However, if we employ as sequences of GLT sequences the multilevel circulant approximations described in Section 7.3, then the matrix $\widetilde{S}$ itself will be a multilevel circulant

matrix. In fact, as we mentioned above, the set of all d-level circulant matrix is a commutative ring under the matrix addition and multiplication, and each d-level circulant matrix is diagonalizable by the discrete Fourier transform. In particular, by employing the discrete Fourier transform one can easily compute matrix–vector and matrix–matrix operations involving d-level circulant matrices, keeping the storage requirements of order $O(\bar{N})$. In fact, one has only to evaluate and store the eigenvalues of a given d-level circulant matrix. Then, any time one has to perform matrix–vector or matrix–matrix operations involving d-level circulant matrices, one has only to work with the diagonal matrix containing the eigenvalues. As the computational cost of the fast Fourier transform is of order $O(\bar{N}\log\bar{N})$, it is clear that the cost of applying a preconditioner based on a d-level circulant approximation will be $O(\bar{N}\log\bar{N})$.

We conclude this section by showing the preconditioner that will be employed in our numerical results. As we mentioned, we employ the T. Chan preconditioner for a given unilevel Toeplitz matrix. In addition, the matrices M_1 and M_2 can be approximated by a scaled identity. Then, our approximation $\widetilde{S}$ is given by (7.20), with $\widetilde{M}_{1_{\mathbf{j}}}$ and $\widetilde{M}_{2_{\mathbf{j}}}$ given by those scaled identities, respectively, and $\widetilde{\mathfrak{B}}_{\mathbf{j}}$ given by the following T. Chan preconditioner for $\mathfrak{B}$:

$$\begin{aligned} C_3(\bar{\mathfrak{B}}) &= C_1(\mathfrak{C}_\alpha)\otimes I_N - I_{n_t}\otimes\left(C_1(\mathfrak{L}^{\mathrm{R}}_{\gamma_1})\otimes I_{n_{x_2}} + I_{n_{x_1}}\otimes C_1(\mathfrak{L}^{\mathrm{R}}_{\gamma_2})\right)\\ &= \left(F_{n_{x_1}}\otimes F_{n_{x_2}}\otimes F_{n_t}\right)^* \Lambda_{\bar{N}} \left(F_{n_{x_1}}\otimes F_{n_{x_2}}\otimes F_{n_t}\right), \end{aligned}$$

where the diagonal matrix $\Lambda_{\bar{N}}$ is given by

$$\Lambda_{\bar{N}} = \Lambda_\alpha\otimes I_N - I_{n_t}\otimes\left(\Lambda_{\gamma_1}\otimes I_{n_{x_2}} + I_{n_{x_1}}\otimes\Lambda_{\gamma_2}\right),$$

with Λ_α, Λ_{γ_1}, and Λ_{γ_2} being the diagonal matrices containing the eigenvalues of the T. Chan approximations of the matrices $\mathfrak{C}_\alpha$, $\mathfrak{L}^{\mathrm{R}}_{\gamma_1}$, and $\mathfrak{L}^{\mathrm{R}}_{\gamma_2}$, respectively.

7.7 Numerical Results

We now present numerical results that show the effectiveness of the proposed strategy. We consider a problem of the type (7.6)–(7.8), with $\Omega\times(0,t_f) = (0,1)^2\times(0,1)$. The desired state is given by

$$v_d(x_1,x_2,t) = 10\cos(10x_1)\sin(x_1x_2)(1-e^{-5t}),$$

as in [39, Section 5.1], with homogeneous boundary and initial conditions. We present results for three types of problems, that is, problems with box constraints only on the state v, problems with box constraints only the control u, and problems with box constraints on both the variables. As expected, the last type of problem is the most challenging one. For this reason, we focus our attention on this class of problems, and present only few experiments on problems of the other types. For problems with box constraints, we employ the convention that the discretized restricting functions are of the form $\boldsymbol{v}_{\max} = -\boldsymbol{v}_{\min} = c\mathbf{1}$ (or $\boldsymbol{u}_{\max} = -\boldsymbol{u}_{\min} = c\mathbf{1}$), where $\mathbf{1}$ is the vector of all ones and $c > 0$. For this reason, we present only the value of the entries of $\boldsymbol{v}_{\min}$ ($\boldsymbol{u}_{\min}$, respectively). For all the

problems presented, we construct a uniform grid in space and in time, and set $n_{x_1} = n_{x_2} = n_t = n$, for some $n \in \mathbb{N}$. The overall size of the discretized state vector is given by $\bar{N} = n_{x_1} \cdot n_{x_2} \cdot n_t = n^3$. As an indicator of performance of the numerical method, we employ the discrete L^2-norm of the discrepancy between the state and the desired state. Specifically, by applying the trapezoidal rule, we define

$$\texttt{trap}_{L^2}(v - v_d) \approx \|v - v_d\|_{L^2}.$$

We would like to note that the previous measure is not expected to converge to zero. In fact, the functions v and v_d are not equal on the boundary $\partial\Omega$, and we expect that the approximate discrepancy measure slightly increases as we refine the grid.

For all the tests considered, we run the MATLAB function `pcg` as the linear solver, and set the tolerance dynamically as

$$\text{Krylov Tol.} = 0.05 \cdot \min\{\|\mathfrak{B}\boldsymbol{v}^{(k)} - \boldsymbol{u}^{(k)}\|_\infty, \|\boldsymbol{v}^{(k)} - \boldsymbol{z}_v^{(k)}\|_\infty, \|\boldsymbol{u}^{(k)} - \boldsymbol{z}_u^{(k)}\|_\infty, 10^{-4}\},$$

at every non-linear iteration k, presenting the average number of PCG iterations required for reaching convergence.

Since we are only interested in showing the viability of the proposed approach, we employ a standard 2-Block ADMM for solving problems of the form of (7.12)–(7.13), with additional box constraints on the variables. It is worth mentioning that various potential acceleration strategies for ADMMs have been studied in the literature (see for example [21, 61]), and the approach presented here could also be employed within those versions of ADMM. We choose the step-size $\rho = 1.618$. We choose the penalty parameter $\delta \in \{0.1, 0.4, 2, 10, 100\}$, as for those values the method behaves reasonably well. We would like to note that one could tune this parameter for each problem instance and obtain significantly better results. However, as this is not practical, we restrict ourselves to a small set of possible values. Finally, the termination criteria of the ADMM are summarized as follows:

$$(\|\mathcal{A}\boldsymbol{v}^{(k)} - \boldsymbol{u}^{(k)}\|_\infty \leqslant 10^{-4}) \wedge (\|\boldsymbol{v}^{(k)} - \boldsymbol{z}_v^{(k)}\|_\infty \leqslant 10^{-4}) \wedge (\|\boldsymbol{u}^{(k)} - \boldsymbol{z}_u^{(k)}\|_\infty \leqslant 10^{-4}).$$

We would like to note that we do not require a specific tolerance for the dual infeasibility in order to avoid unnecessary computations. Instead, we report the dual infeasibility at the accepted optimal point.

All tests are run on MATLAB R2019a using a 2.2 GHz Intel (hexa-) core i7 processor, run under the Windows 10 operating system. All CPU times are reported in seconds.

7.7.1 Box Constraints on the State v

We first test our solver on problems with box constraints on the state variable, while the control is left free, that is $v_{\min} \leqslant v \leqslant v_{\max}$, $-\infty \leqslant u \leqslant \infty$. We report the results in Table 7.1. All fixed parameters are provided at the title of the respective table.

[10] † means that the solution coincides with the equality constrained solution; all the variables lie strictly within the restriction bounds.

Table 7.1: Inequalities on the state: varying restriction bounds (with $\bar{N} = 50^3$, $\gamma_1 = \gamma_2 = 1.3$, $\alpha = 0.7$, $\beta = 10^{-4}$, $\delta = 0.1$).

$v_{\min}$	$\texttt{trap}_{L^2}(v - v_d)$	Dual Inf.	PCG	ADMM	CPU
-7	5.60e-1†[10]	2.13e-3	9	75	142.31
-5	5.80e-1	3.14e-3	10	105	206.73
-3	7.88e-1	4.11e-3	10	100	193.77
-1	1.38e0	8.74e-4	10	86	173.49

From Table 7.1, we can observe the robustness of our preconditioner, as the average number of PCG iterations is roughly constant, with the total CPU time mainly depending on the number of ADMM iterations. Regarding the latter, we observe that, even for very tight box constraints, the number of ADMM iterations stays roughly the same.

7.7.2 Box Constraints on the Control u

We now focus on the case with $-\infty \leqslant v \leqslant \infty$, $u_{\min} \leqslant u \leqslant u_{\max}$. By employing similar arguments as in [41] we can prove existence of an optimal solution. We run the method for different inequality bounds on the control u. We report the results in Table 7.2, stating all the values of the parameters used to perform the experiment in the respective caption.

Table 7.2: Inequalities on the control: varying restriction bounds (with $\bar{N} = 50^3$, $\gamma_1 = \gamma_2 = 1.3$, $\alpha = 0.7$, $\beta = 10^{-4}$, $\delta = 0.4$).

$u_{\min}$	$\texttt{trap}_{L^2}(v - v_d)$	Dual Inf.	PCG	ADMM	CPU
-400	5.60e-1†	8.43e-4	16	30	84.46
-300	5.65e-1	8.79e-4	19	22	72.77
-200	6.25e-1	4.39e-4	17	28	85.55
-100	8.90e-1	1.49e-4	18	65	205.69

From Table 7.2, we see that also for this case the average number of PCG iterations is roughly constant, even when decreasing the value of $u_{\min}$. As a consequence, the average CPU time is almost constant, with the overall CPU time depending only on the number of ADMM iterations. As we can see from Table 7.2, the number of ADMM iterations slightly depends on the lower bound for the control u, although it stays low compared to the dimension of the problem solved.

7.7.3 Box Constraints on Both Variables

Let us now consider the case where $v_{\min} \leqslant v \leqslant v_{\max}$, $u_{\min} \leqslant u \leqslant u_{\max}$. First, we present the runs of the method for different inequality bounds in Table 7.3. As one can observe from Table 7.3, the average number of PCG iterations slightly depends on the lower bounds on the state v and the control u; nonetheless, it is reasonably small compared to the dimension of the problems solved. In addition, as we expected, this class of problem is the most challenging one, and we can observe this by looking at the number of ADMM iterations, which clearly depends on the bounds on the variables.

Table 7.3: Inequalities on both variables: varying restriction bounds (with $\bar{N} = 50^3$, $\gamma_1 = \gamma_2 = 1.3$, $\alpha = 0.7$, $\beta = 10^{-4}$, $\delta = 0.4$).

$v_{\min}$	$u_{\min}$	$\texttt{trap}_{L^2}(v - v_d)$	Dual Inf.	PCG	ADMM	CPU
-7	-400	5.60e-1†	4.88e-3	10	36	74.20
-7	-200	5.94e-1	2.35e-3	11	38	80.14
-4	-350	6.45e-1	1.99e-4	18	126	412.66
-1	-400	1.38e0	2.56e-4	19	109	377.86

Next, we test our method when varying grid size, and report the results in Table 7.4. As one can observe in Table 7.4, the grid size does not affect the average number of inner PCG iterations. Nevertheless, as the size of the problem increases, we expect that also the number of ADMM iterations increases. Furthermore, we can observe the first-order convergence of the numerical method, as n is increased. In addition, we can observe that the CPU time scales approximately as $\bar{N} \log \bar{N}$, as expected.

Table 7.4: Inequalities on both variables: varying grid size (with $v_{\min} = -4$, $u_{\min} = -350$, $\gamma_1 = \gamma_2 = 1.3$, $\alpha = 0.7$, $\beta = 10^{-4}$).

$\bar{N}$	δ	$\texttt{trap}_{L^2}(v - v_d)$	Dual Inf.	PCG	ADMM	CPU
8^3	2	3.87e-1	5.23e-4	12	86	1.89
16^3	2	5.02e-1	8.68e-5	13	58	6.06
32^3	0.4	6.09e-1	2.94e-4	16	62	75.04
50^3	0.4	6.45e-1	1.99e-4	18	126	412.66
64^3	0.1	6.58e-1	3.34e-1	17	97	987.12
80^3	0.1	6.65e-1	4.31e-1	17	102	1,135.83
100^3	0.1	6.70e-1	4.91e-1	17	119	2,436.17
128^3	0.1	6.73e-1	3.49e-1	17	169	9,077.08

Subsequently, we test our method when varying the fractional derivative orders, and present the results in Table 7.5. From Table 7.5, we can observe that, as $\gamma = \gamma_1 = \gamma_2$ approaches 1, the constraint matrix becomes more ill-conditioned.

This is due to the scaling factor in the definition of the Riesz derivative (that is, $\frac{-1}{2\cos(\frac{\gamma\pi}{2})}$). As a consequence, we observe an increase of the PCG iterations in the case where $\gamma = 1.1$. Nonetheless, the average number of PCG iterations is reasonably small compared to the size of the problem solved. In addition, we can see that the number of ADMM iterations is roughly constant.

Table 7.5: Inequalities on both variables: varying fractional derivative orders (with $\bar{N} = 32^4$, $v_{\min} = -4$, $u_{\min} = -350$, $\beta = 10^{-4}$).

α	γ	δ	$\texttt{trap}_{L^2}(v - v_d)$	Dual Inf.	PCG	ADMM	CPU
0.1	1.3	0.4	6.46e-1	1.60e-4	17	126	380.75
0.3	1.3	0.4	6.46e-1	2.56e-4	17	126	385.96
0.5	1.3	0.4	5.12e-1	2.83e-4	18	126	408.47
0.9	1.3	0.4	6.44e-1	3.10e-4	19	125	419.46
0.7	1.1	0.4	6.48e-1	1.21e-3	30	100	508.59
0.7	1.5	0.1	7.79e-1	4.23e-4	15	96	275.03
0.7	1.7	0.4	1.04e0	2.46e-4	13	113	275.21
0.7	1.9	0.1	1.36e0	1.36e-3	8	108	180.85

Finally, in Table 7.6 we present the runs of our method when varying the regularization parameter β. It is worth noting that, as β is changed, the solution of the equality constrained problem is significantly altered. For this reason, we adjust the inequality constraints of the problem for each value of β, in order to ensure that the optimal solution will lie strictly within the bounds. This allows us to compare the convergence behaviour of ADMM, for instances with different regularization values, β.

Table 7.6: Inequalities on both variables: varying regularization (with $\bar{N} = 50^3$, $\alpha = 0.7$, $\gamma_1 = \gamma_2 = 1.3$).

β	$v_{\min}$	$u_{\min}$	δ	$\texttt{trap}_{L^2}(v - v_d)$	Dual Inf.	PCG	ADMM	CPU
10^{-2}	-2	-100	0.1	1.77e-0†	1.38e-3	11	87	190.99
10^{-4}	-7	-400	0.4	5.60e-1†	4.88e-3	10	36	74.20
10^{-6}	-9	$-2,800$	10	1.28e-1†	6.03e-4	8	47	71.13
10^{-8}	-9	$-4,000$	100	1.13e-1†	2.18e-4	6	32	44.70
10^{-10}	-9	$-4,000$	100	1.13e-1†	2.18e-4	5	32	40.77

Again, the average number of PCG iterations is roughly constant, with the overall CPU time depending mainly on the number of ADMM iterations. It is worth noting that, as β becomes smaller, the average number of PCG iterations decreases. In addition, as β decreases, also the number of ADMM iterations decreases. Again, the number of ADMM iterations slightly depends on the regularization parameter, although it stays low compared to the size of the problem being solved.

From the results presented in this section, we can conclude that the proposed approach is sufficiently robust with respect to the problems parameters. In fact, the number of PCG iterations required to solve each ADMM system is small compared to the size of the problem solved, for a wide range of parameter choices. In addition, ADMM is able to achieve a 4-digit accurate primal solution in a reasonable number of iterations, allowing us to solve a convex quadratic optimization problem with very dense equality constraints in a reasonable time. Finally, we believe that the method described here can be employed to solve even more complex FDE optimization problems.

7.8 Summary

In this chapter, we considered the optimal control of FDEs, with additional box constraints on the state and control variables. Previously, no robust solver for this class of problems with FDEs defined in dimension $d > 1$ was available, to our knowledge. By employing a discretize-then-optimize approach, we were required to solve a convex quadratic optimization problems with box constraints on the variables. We proposed the use of an Alternating Direction Method of Multipliers. This strategy allowed us to separate inequality from equality constraints, solve the latter in order to obtain the current approximations of the state, the control, and the adjoint variables, and then updating the multipliers by solving an unconstrained problem and projecting the solutions onto the box constraints. In this way, we preserved the structure of the equality constraints. In order to solve the latter, which presents a multilevel Toeplitz structure, we employed the theory of Generalized Locally Toeplitz sequences. This allowed us to find optimal preconditioners based on multilevel circulant approximation. Then, we employed the derived preconditioner within a suitable preconditioned Krylov subspace method in order to solve the system arising at each ADMM iteration. The circulant approximation of the equality constraints allowed us to employ the discrete Fourier transform in order to find the eigenvalues of the individual blocks of the proposed preconditioner, thus allowing us to keep the storage requirements to order of $\bar{N}$ (where $\bar{N}$ is the grid size), while requiring only $O(\bar{N} \log \bar{N})$ operations for every iteration of the Krylov solver. Numerical results showed the scalability, efficiency, and generality of our approach.

Chapter 8

Conclusion

"When I was at last by myself, a drowsy sensation fell on me; but before my eyes closed I endeavoured to reproduce the Third Dimension, and especially the process by which a Cube is constructed through the motion of a Square. It was not so clear as I could have wished; but I remembered that it must be "Upward, and yet not Northward," and I determined steadfastly to retain these words as the clue which, if firmly grasped, could not fail to guide me to the solution. So mechanically repeating, like a charm, the words, "Upward, yet not Northward," I fell into a sound refreshing sleep."

– Edwin Abbott Abbott, *Flatland: A Romance of Many Dimensions*

In this thesis, we developed preconditioned iterative methods for the solution of optimal control problems with differential operators as constraints. The majority of the problems considered above involved time-dependent PDEs as constraints, but we also tackled the optimal control of stationary (linear and non-linear) PDEs, as well as the optimal control of FDEs.

In order to derive the proposed preconditioners for the PDE-constrained optimization problems considered in this thesis, we devised tailored, easily invertible transformations, which allowed us to symmetrize the linear systems (or make them as close to symmetric as possible) arising from the discretization of the first-order optimality conditions. This strategy allowed us to employ saddle-point theory. Starting from ideal preconditioners (for which the inverse operator is almost as computationally expensive as solving the original system), we devised suitable approximations of the main blocks. In order to approximately invert the $(1,1)$-block, we employed an inner iteration. For some of the systems, the inner iteration required a fixed number of steps of a preconditioned iterative method applied to a block matrix that represents the discrete optimality conditions of a simpler PDE-constrained optimization problem. However, the most complex tasks have been to devise robust approximations of the corresponding Schur complements. In order to do so, we employed the matching strategy and a novel block commutator argument. The resulting preconditioners were tested on a large number of test problems, showing the robustness and the efficiency of our approaches.

In order to derive an optimal preconditioner for the FDE-constrained optimization problems with algebraic constraints on the state and/or the control

variables, we employed the GLT theory. Applying an ADMM to separate equality from inequality constraints, we were faced with solving a very large and dense linear system. By exploiting the (multilevel) Toeplitz structure of the discretized system and employing GLT theory, we were able to derive an optimal preconditioner for the linear system to be solved. Then, we tested our strategy on a wide range of problems, showing that the computaitonal cost scales as $\bar{N} \log \bar{N}$, where $\bar{N}$ is the dimension of the grid used. In addition, our strategy does not suffer from excessive memory requirements, as we are able to keep the storage requirement of order $\bar{N}$.

We expect the novel transformations, Schur complement approximations, formulation of the block commutator argument, and multilevel circulant preconditioning ideas could inform the efficient preconditioning of many problems beyond those discussed in this thesis. What is left to discuss is related to future work. As the Square in Abbott's Flatland was wondering about (and puzzled by) the extension of the Plane to the Third Dimension, we question ourself on the possible extensions of our work.

The problems considered in this thesis are aimed at describing real-life and physical problems. For this reason, one can extend the strategies devised above to problems with multiple differential operators as constraints, to problems with different cost functionals, or to problems with additional algebraic constraints on the variables.

As we have employed a Crank–Nicolson discretization in time for the parabolic PDEs considered above, one may also try to devise optimal preconditioners for time-dependent PDE- and FDE-constrained optimization problems when higher-order discretizations in time are used. For instance, the discussion at the end of Section 3.2.2 may be the starting point for developing preconditioned iterative methods for optimal control of time-dependent PDEs when employing time-stepping schemes other than Crank–Nicolson. In the same section we also discussed the possibility of applying the discretize-then-optimize strategy as an alternative approach to the one employed in this work. We have noted that the linear system arising from the discretize-then-optimize approach is symmetric from the start (so no transformation is required). Further, we have noted that in this case we may apply a similar preconditioning strategy to the one described in Section 3.3, and that, if such a preconditioner is optimal, one may apply it to the other problems decribed in this thesis as well. However, we did not address this question here. For this reason, future work can be related to prove that our strategy may be applied also to the case of a discretize-then-optimize approach, and prove the optimality of the derived preconditioner.

As we noted above, one of the main drawback of the all-at-once approach when solving optimal control of time-dependent PDEs is that one has to store global-in-time solutions. For this reason, when employing very fine discretization computers may easily run out of memory. An alternative may be exploiting the Kronecker structure of the linear system arising upon discretization, and employ low-rank approximations of the problem considered. For this reason, future work may be coupling the preconditioning strategy devised in this work with a low-rank solver for the problems we have considered above.

As in practical industrial problems it is not always possible to apply the

control within the whole domain, one has to introduce the control on its boundary, resulting thus in Dirichlet or Neumann boundary control problems, or only on a part of the domain, resulting thus in subdomain control problems. A future study can be related to extend the optimal preconditioners derived here to the boundary control of the Navier–Stokes equations. Regarding these equations, we observed in Chapter 6 that, for very small viscosity, the numbers of Oseen iterations required to reach a prescribed reduction on the non-linear residual were increasing for a wider range of regularization parameters. For this reason, we noted the importance of a robust and efficient solver also for the Newton approximation of the Navier–Stokes control problems, which could be the topic of future work. Finally, we believe that it would be of interest to investigate on the effectiveness of the block commutator argument devised for solving the Stokes control problems. We believe the combination of these new ideas would allow the solution of a range of problems beyond those discussed in this thesis.

Appendix A

An Overview of the GLT Theory

In this Appendix, we describe some important notions employed in the GLT theory.

We begin with some fundamental properties of d-level circulant matrices. It is possible to prove the following theorem, the proof of which can be found in [57, Section 3.4]:

Theorem 8. *The d-level circulant matrix A admits the following expression:*

$$A = \sum_{\mathbf{j}=\mathbf{1}}^{\mathbf{n}_A-\mathbf{1}} a_{\mathbf{j}} C_{\mathbf{n}_A}^{\mathbf{j}},$$

with $C_{\mathbf{n}_A}^{\mathbf{j}}$ defined as in Section 7.3. In addition, given $c_{-\mathbf{i}}, c_{-\mathbf{i}+\mathbf{1}}, \ldots, c_{\mathbf{i}} \in \mathbb{C}$, with $\mathbf{i} \in \mathbb{N}^d$, we have that any linear combination of the form $\sum_{\mathbf{j}=-\mathbf{i}}^{\mathbf{i}} c_{\mathbf{j}} C_{\mathbf{n}_A}^{\mathbf{j}}$ is a d-level circulant matrix. Then,

$$\sum_{\mathbf{j}=-\mathbf{i}}^{\mathbf{i}} c_{\mathbf{j}} C_{\mathbf{n}_A}^{\mathbf{j}} = F_{\mathbf{n}_A}^* \left(\operatorname*{diag}_{\mathbf{j}=\mathbf{0},\mathbf{1},\ldots,\mathbf{n}_A-\mathbf{1}} c\left(\frac{2\pi\mathbf{j}}{\mathbf{n}_A}\right) \right) F_{\mathbf{n}_A},$$

where $c(\mathbf{z}) = \sum_{\mathbf{j}=-\mathbf{i}}^{\mathbf{i}} c_{\mathbf{j}} e^{i\langle \mathbf{j},\, \mathbf{z}\rangle}$, with $\langle \mathbf{j},\, \mathbf{z}\rangle = \sum_{i=1}^{d} j_i\, z_i$, and $F_{\mathbf{n}_A} = F_{n_1} \otimes F_{n_2} \otimes \ldots \otimes F_{n_d}$ is the multilevel discrete Fourier transform, with F_{n_i} denoting the unitary discrete Fourier transform of order n_i. In the previous expression, $F_{\mathbf{n}_A}^*$ denotes the inverse of the multilevel discrete Fourier transform. Moreover, $\sum_{\mathbf{j}=-\mathbf{i}}^{\mathbf{i}} c_{\mathbf{j}} C_{\mathbf{n}_A}^{\mathbf{j}}$ is a normal matrix whose spectrum is given by:

$$\lambda\left(\sum_{\mathbf{j}=-\mathbf{i}}^{\mathbf{i}} c_{\mathbf{j}} C_{\mathbf{n}_A}^{\mathbf{j}} \right) = \left\{ c\left(\frac{2\pi\mathbf{j}}{\mathbf{n}_A}\right) : \mathbf{j} = \mathbf{0}, \mathbf{1}, \ldots, \mathbf{n}_A - \mathbf{1} \right\}.$$

As a consequence of the previous theorem, we have that the set of all d-level circulant matrix is a commutative ring under matrix addition and multiplication. Further, we also have that a d-level circulant matrix A is diagonalizable by the discrete Fourier transform.

An important notion in the GLT theory is that of a matrix-sequence. A matrix-sequence is a sequence of the form $\{A_j\}_j$, where j varies over some infinite

subset of $\mathbb{N}$, A_j is a square matrix of size n_{A_j}, and $n_{A_j} \to \infty$ as $j \to \infty$. In particular, a d-level matrix sequence is a sequence of the form $\{A_{\mathbf{j}}\}_j$, where $A_{\mathbf{j}}$ is a matrix of size $N(\mathbf{j}) \times N(\mathbf{j})$, j varies over some infinite subset of $\mathbb{N}$, and $\mathbf{j} = \mathbf{j}(j) \in \mathbb{N}^d$ is such that $\mathbf{j} \to \infty$, as $j \to \infty$.

An approximating class of sequences is a sequence of matrix-sequences that is able to approximate the eigenvalues or the singular values of another given matrix-sequence. Specifically, we have the following formal definition [57, Section 2.7]:

Definition 1. *Let $\{A_j\}_j$ be a matrix-sequence, with A_j of size $n_{A_j} \times n_{A_j}$, and let $\{\{\widetilde{A}_{j,m}\}_j\}_m$ be a sequence of matrix-sequences, with $\widetilde{A}_{j,m}$ of size $n_{A_j} \times n_{A_j}$. We say that $\{\{\widetilde{A}_{j,m}\}_j\}_m$ is an approximating class of sequences (a.c.s.) for $\{A_j\}_j$ if for every m, there exists j_m such that, for all $j \geqslant j_m$, we can write:*

$$A_j = \widetilde{A}_{j,m} + R_{j,m} + N_{j,m}, \qquad \operatorname{rank}(R_{j,m}) \leqslant c(m)d_j, \quad \|N_{j,m}\| \leqslant \omega(m),$$

where j_m, $c(m)$, and $\omega(m)$ depend only on m, and are such that:

$$\lim_{m\to\infty} c(m) = \lim_{m\to\infty} \omega_m = 0.$$

In that case, we write $\{\{\widetilde{A}_{j,m}\}_j\}_m \xrightarrow{\text{a.c.s.}} \{A_j\}_j$.

From the previous definition, we have that $\{\{\widetilde{A}_{j,m}\}_j\}_m$ is an approximating class of sequences for $\{A_j\}_j$ if the difference between A_j and $\widetilde{A}_{j,m}$ is the sum of a matrix of low rank and a matrix of small norm.

We now introduce the definitions of clusters.

Definition 2. *Let $\{A_j\}_j$ be a sequence of matrices, with A_j of size $n_{A_j} \times n_{A_j}$, and let $X \subseteq \mathbb{C}$ be a non-empty subset of $\mathbb{C}$. We say that $\{A_j\}_j$ is strongly clustered at X (in the sense of eigenvalues) if $\forall \epsilon > 0$ we have:*

$$\#\{i \in \{1, 2, \ldots, n_{A_j}\} : \lambda_i(A_j) \notin D(X, \epsilon)\} = O(1),$$

and weakly clustered at X if $\forall \epsilon > 0$,

$$\#\{i \in \{1, 2, \ldots, n_{A_j}\} : \lambda_i(A_j) \notin D(X, \epsilon)\} = o(n_{A_j}).$$

Here, given $\epsilon > 0$, we denote with $D(X, \epsilon)$ the ϵ-expansion of X, defined as $D(X, \epsilon) = \bigcup_{z\in X} D(z, \epsilon)$, where $D(z, \epsilon)$ is the the disk with center z and radius ϵ. Further, with $\#X$ we denote the cardinality of a set X.

Given a function $f : [0, 1]^d \to \mathbb{C}$, we define the $\mathbf{j}$-*th diagonal sampling matrix generated by* f as the following $n_{D_{\mathbf{j}}} \times n_{D_{\mathbf{j}}}$ diagonal matrix:

$$D_{\mathbf{j}}(f) = \operatorname*{diag}_{\mathbf{i}=\mathbf{1},\mathbf{2},\ldots,\mathbf{j}} f\left(\frac{\mathbf{i}}{\mathbf{j}}\right).$$

In addition, given $j, m \in \mathbb{N}$, $g : [0, 1] \to \mathbb{C}$, and $f \in L^1([-\pi, \pi])$, we define the 1-*level locally Toeplitz operator* as the following $n_{LT_j} \times n_{LT_j}$ matrix:

$$LT_j^m(g, f) = \big(D_m(g) \otimes T_{\lfloor j/m \rfloor}(f)\big) \oplus O_{j \bmod m},$$

where $D_m(g)$ is a diagonal sampling matrix generated by g, $T_{\lfloor j/m \rfloor}(f)$ a Toeplitz matrix generated by f, and $O_{j \bmod m}$ a zero matrix. Similarly, given also $\mathbf{j}, \mathbf{m} \in N^d$, $g : [0,1]^d \to \mathbb{C}$, and $f \in L^1([-\pi,\pi]^d)$, the *d-level locally Toeplitz operator* is recursively defined as the following $N(\mathbf{j}) \times N(\mathbf{j})$ matrix:

$$LT_{\mathbf{j}}^{\mathbf{m}}(g, f_1 \otimes f_2 \otimes \ldots \otimes f_d) = LT_{j_1,j_2,\ldots,j_d}^{m_1,m_2,\ldots,m_d}(g(x), f_1 \otimes f_2 \otimes \ldots \otimes f_d).$$

We can now define the *Locally Toeplitz sequences*, that can be generalized to define the notion of *GLT sequences*. This theory was originally developed in [172], and we refer the reader to [56, 57] for a complete derivation and overview of this class.

Definition 3. *Let $\{A_{\mathbf{j}}\}_j$ be a d-level matrix-sequence, let $g : [0,1]^d \to \mathbb{C}$ be Riemann-integrable, and let $f \in L^1([-\pi,\pi]^d)$. We say that $\{A_{\mathbf{j}}\}_j$ is a (d-level) locally Toeplitz (LT) sequence with symbol $g \otimes f$, and we write $\{A_{\mathbf{j}}\}_j \sim_{LT} g \otimes f$, if:*

$$\{LT_{\mathbf{j}}^{\mathbf{m}}(g, f)\}_j \xrightarrow{\text{a.c.s.}} \{A_{\mathbf{j}}\}_j, \quad \text{as } m \to \infty.$$

The previous definition can be generalized to define the *Generalized Locally Toeplitz sequences* [57, Section 5.1].

Definition 4. *Let a d-level matrix-sequence $\{A_{\mathbf{j}}\}_j$, and a measurable function $\kappa : [0,1]^d \times [-\pi,\pi]^d \to \mathbb{C}$ be given. Suppose that $\forall \epsilon > 0$ there exists a finite number of d-level LT sequences $\{A_{\mathbf{j}}^{(i,\epsilon)}\}_j \sim_{LT} g_{i,\epsilon} \otimes f_{i,\epsilon}$, $i = 1, 2, \ldots, N_\epsilon$, such that as $\epsilon \to 0$:*

$$\sum_{i=1}^{N_\epsilon} g_{i,\epsilon} \otimes f_{i,\epsilon} \to \kappa$$

in measure, and that

$$\left\{ \sum_{i=1}^{N_\epsilon} A_{\mathbf{j}}^{(i,\epsilon)} \right\}_j \xrightarrow{\text{a.c.s.}} \{A_{\mathbf{j}}\}_j.$$

Then $\{A_{\mathbf{j}}\}_j$ is a d-level GLT sequence with symbol κ, and we write $\{A_{\mathbf{j}}\}_j \sim_{GLT} \kappa$.

The above definitions were used to motivate the multilevel circulant preconditioner derived in Chapter 7, and were used in [146] to prove the effectiveness of the proposed preconditioning strategy.

In Chapter 7, we supposed the following assumptions hold true. Where the first assumption does not hold automatically, it can be ensured by multiplying the discretized PDE constraint by a constant depending on the mesh-sizes for a given problem.

Assumption 2. *The matrices $\mathfrak{B}$, M_1, and M_2 of dimension $\bar{N} \times \bar{N}$ are such that:*

- *The sequence $\{\mathfrak{B}\}_{\bar{N}}$ is a d-level matrix sequence with spectral norm uniformly bounded with respect to $\bar{N}$, i.e. there exists a constant $c_{\mathfrak{B}}$ such that $\|\mathfrak{B}\| \leqslant c_{\mathfrak{B}}$ for all $\bar{N}$. Furthermore, there exists a measurable function $\kappa : [0,1]^d \times [-\pi,\pi]^d \to \mathbb{C}$, which is the symbol of $\{\mathfrak{B}\}_{\bar{N}}$, so that $\{\mathfrak{B}\}_{\bar{N}} \sim_{GLT} \kappa$.*

- *The sequences* $\{M_1\}_{\bar{N}}$ *and* $\{M_2\}_{\bar{N}}$ *are two* d*-level matrix sequences, with uniformly bounded spectral norms with respect to* $\bar{N}$*. Furthermore, there exist two measurable functions* $\kappa_1, \kappa_2 : [0,1]^d \times [-\pi,\pi]^d \to \mathbb{R}$*, such that* $\kappa_i \geqslant 0$*, and* $\{M_i\}_{\bar{N}} \sim_{GLT} \kappa_i$*, for* $i = 1, 2$*.*

Bibliography

[1] Abraham F., Behr M., Heinkenschloss M.: *Shape Optimization in Steady Blood Fow: A Numerical Study of Non-Newtonian Effects*, Comput. Method. Biomech. **8**, 127–137, 2005

[2] Anastasio T. J.: *The Fractional-Order Dynamics of Brainstem Vestibulo-Oculomotor Neurons*, Biol. Cybernet. **72**, 69–79, 1994

[3] Angenent S., Haker S., Tannenbaum A.: *Minimizing Flows for the Monge–Kantorovich Problem*, SIAM J. Matrix Anal. Appl. **35**, 61–97, 2003

[4] Apel T., Flaig T. G.: *Crank–Nicolson Schemes for Optimal Control Problems with Evolution Equations*, SIAM J. Numer. Anal. **50**, 1484–1512, 2012

[5] Arridge S. R.: *Optical Tomography in Medical Imaging*, Inverse Problems **15**, R41–R93, 1999

[6] Arrow K., Hurwicz L., Uzawa H.: *Studies in Nonlinear Programming*, Stanford University Press, 1958

[7] Axelsson O., Blaheta R., Kohut, R.: *Preconditioning Methods for High-Order Strongly Stable Time Integration Methods with an Application for a DAE Problem*, Numer. Linear Algebra Appl. **22**, 930–949, 2015

[8] Axelsson O., Farouq S., Neytcheva M.: *A Preconditioner for Optimal Control Problems, Constrained by Stokes Equation with a Time-Harmonic Control*, J. Comp. Appl. Math. **310**, 5–18, 2017

[9] Axelsson O., Neytcheva M.: *Eigenvalue Estimates for Preconditioned Saddle Point Matrices*, Numer. Linear Algebra Appl. **13**, 339–360, 2006

[10] Barthel W., John C., Tröltzsch F.: *Optimal Boundary Control of a System of Reaction Diffusion Equations*, ZAMM **90**, 966–982, 2010

[11] Becker R., Braack M.: *A Finite Element Pressure Gradient Stabilization for the Stokes Equations Based on Local Projections*, Calcolo **38**, 173–199, 2001

[12] Becker R., Vexler B.: *Optimal Control of the Convection–Diffusion Equation Using Stabilized Finite Element Methods*, Numer. Math. **106**, 349–367, 2007

[13] Bell J. B., Colella P., Glaz H. M.: *A Second-Order Projection Method for the Incompressible Navier–Stokes Equations*, J. Comput. Phys. **85**, 257–283, 1989

[14] Benamou J. D., Brenier Y.: *A Computational Fluid Mechanics Solution to the Monge–Kantorovich Mass Transfer Problem*, Numer. Math. **84**, 375–393, 2000

[15] Benson D. A.: *The Fractional Advection-Dispersion Equation: Development and Application*, D.Phil. Thesis, University of Nevada, 1998

[16] Benzi M.: *Preconditioning Techniques for Large Linear Systems: A Survey*, J. Comput. Phys. **182**, 418–477, 2002

[17] Benzi M., Golub G. H., Liesen J.: *Numerical Solution of Saddle Point Problems*, Acta Numerica **14**, 1–137, 2005

[18] Bergounioux M., Haddou M., Hintermüller M., Kunisch, K.: *A Comparison of a Moreau–Yosida-Based Active Set Strategy and Interior Point Methods for Constrained Optimal Control Problems*, SIAM J. Optim. **11**, 495–521, 2000

[19] Bergounioux M., Ito K., Kunisch K.: *Primal-Dual Strategy for Constrained Optimal Control Problems*, SIAM J. Control Optim. **37**, 1176–1194, 1999

[20] Bouchouev I., Isakov V.: *Uniqueness, Stability and Numerical Methods for the Inverse Problem that Arises in Financial Markets*, Inverse Problems **15**, R95–R116, 1999

[21] Boyd S., Parikh N., Chu E., Peleato B., Eckstein J.: *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, Found. Trends Mach. Learn. **3**, 1–122, 2010

[22] Boyle J., Mihajlović M., Scott J.: *HSL_MI20: an Efficient AMG Preconditioner for Finite Element Problems in 3D*, Int. J. Numer. Meth. Eng. **82**, 64–98, 2010

[23] Braack M., Burman E.: *Local Projection Stabilization for the Oseen Problem and its Interpretation as a Variational Multiscale Method*, SIAM J. Numer. Anal. **43**, 2544–2566, 2006

[24] Bramble J. H., Pasciak J. E., Vassilev A. T.: *Analysis of the Inexact Uzawa Algorithm for Saddle Point Problems*, SIAM J. Numer. Anal. **34**, 1072–1092, 1997

[25] Brandt A.: *Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems*, 3rd International Conference on Numerical Methods in Fluid Mechanics, 82–89, Springer, 1973

[26] Brenner S. C., Scott L. R.: *The Mathematical Theory of Finite Element Methods*, Springer, Berlin, 3rd Edition, 2008

[27] Briggs W. L., Emden Henson V., McCormick S. F.: *A Multigrid Tutorial*. SIAM, 2nd Edition, 2000

[28] Brooks A. N., Hughes T. J. R.: *Streamline Upwind/Petrov–Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier–Stokes Equations*, Comput. Methods Appl. Mech. Eng. **32**, 199–259, 1982

[29] Casas E.: *Control of an Elliptic Problem with Pointwise State Constraints*, SIAM J. Control Optim. **24**, 1309–1318, 1986

[30] Chan R. H., Strang G.: *Toeplitz Equations by Conjugate Gradient with Circulant Preconditioner*, SIAM J. Sci. Stat. Comp. **10**, 104–119, 1989

[31] Chan R. H., Wong C. K.: *Best-Conditioned Circulant Preconditioners*, Linear Alg. Appl. **218**, 205–211, 1995

[32] Chan T. F.: *An Optimal Circulant Preconditioner for Toeplitz Systems*, SIAM J. Sci. Stat. Comp. **9**, 766–771, 1988

[33] Cheney M., Isaacson D., Newell J. C.: *Electrical Impedance Tomography*, SIAM Review **41**, 85–101, 1999

[34] Choi H., Hinze M., Kunisch, K.: *Instantaneous Control of Backward-Facing Step Flows*, Appl. Numer. Math. **31**, 133–158, 1999

[35] Collis S. S., Heinkenschloss M.: *Analysis of the Streamline Upwind/Petrov Galerkin Method Applied to the Solution of Optimal Control Problems*, Technical Report CAAM TR02-01, Dept. of Computational and Applied Math., Rice University, Houston, 2002

[36] De los Reyes J. C.: *Numerical PDE-Constrained Optimization*, Springer, Berlin, 2015

[37] De los Reyes J. C., Schönlieb C.-B.: *Image Denoising: Learning the Noise model Via Nonsmooth PDE-Constrained Optimization*, Inverse Probl. Imag. **7**, 1183–1214, 2013

[38] Deng W., Yin W.: *On Global and Linear Convergence of the Generalized Alternating Direction Method of Multipliers*, J. Sci. Comp. **66**, 889–916, 2016

[39] Dolgov S., Pearson J. W., Savostyanov D. V., Stoll M.: *Fast Tensor Product Solvers for Optimization Problems with Fractional Differential Equations as Constraints*, Appl. Math. Comp. **273**, 604–623, 2016

[40] Dolgov S., Stoll M.: *Low-Rank Solution to an Optimization Problem Constrained by the Navier–Stokes Equations*, SIAM J. Sci. Comput. **39**, A255–A280, 2016

[41] Durastante F., Cipolla S.: *Fractional PDE Constrained Optimization: Box and Sparse Constrained Problems*. In: Falcone M., Ferretti R., Grune L., McEneaney W. M. (eds.), *Numerical Methods for Optimal Control Problems*, 111–135, Springer, 2018

[42] Egger H., Engl H. W.: *Tikhonov Regularization Applied to the Inverse Problem of Option Pricing: Convergence Analysis and Rates*, Inverse Problems **21**, 1027–1045, 2005

[43] Elman H. C., Golub G. H.: *Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems*, SIAM J. Numer. Anal. **31**, 1645–1661, 1994

[44] Elman H. C., Silvester D. J., Wathen A. J.: *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2nd Edition, 2014

[45] Engel M., Griebel M.: *A Multigrid Method for Constrained Optimal Control Problems*, J. Comput. Appl. Math. **235**, 4368–4388, 2011

[46] Evans L. C.: *Partial Differential Equations*, American Mathematical Society, 2nd Edition, 2010

[47] Ewing R. E., Lin T.: *A Class of Parameter Estimation Techniques for Fluid Flow in Porous Media*, Adv. Water Resources **14**, 89–97, 1991

[48] Farrell P. E., Knepley M. G., Mitchell L., Wechsung F.: *PCPATCH: Software for the Topological Construction of Multigrid Relaxation Methods*, ACM Trans. Math. Softw., **47**, 1–22, 2021

[49] Farrell P. E., Mitchell L., Scott L. R., Wechsung F.: *Robust Multigrid Methods for Nearly Incompressible Elasticity Using Macro Elements*, IMA J. Numer. Anal., 2022

[50] Farrell P. E., Mitchell L., Wechsung F.: *An Augmented Lagrangian Preconditioner for the 3D Stationary Incompressible Navier–Stokes Equations at High Reynolds Number*, SIAM J. Sci. Comput., **41**, A3073–A3096, 2019

[51] Fletcher R.: *Conjugate Gradient Methods for Indefinite Systems*, Proc. Dundee Biennial Conf. Numer. Anal., Watson G. A. (editor), Lecture Notes in Mathematics **506**, 73–89, Springer Verlag, Berlin, 1976

[52] Fortin M., Glowinski R.: *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary Value Problems*, North Holland, 1983

[53] Franca L. P., Frey S. L.: *Stabilized Finite Element Methods: II. The Incompressible Navier–Stokes Equations*, Comput. Methods Appl. Mech. Eng. **99**, 209–233, 1992

[54] Freund R. W., Golub G. H., Nachtigal N. M.: *Iterative Solution of Linear Systems*, Acta Numerica **1**, 57–100, 1992

[55] Gabay D., Mercier B.: *A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation*, Comp. Math. Appl. **2**, 17–40, 1976

[56] Garoni C., Serra-Capizzano S.: *Generalized Locally Toeplitz Sequences: Theory and Applications (Volume I)*, Springer, 2017

[57] Garoni C., Serra-Capizzano S.: *Generalized Locally Toeplitz Sequences: Theory and Applications (Volume II)*, Springer, 2018

[58] Gelhard T., Lube G., Olshanskii M. A., Starcke J. H.: *Stabilized Finite Element Schemes with LBB-Stable Elements for Incompressible Flows*, J. Comput. Appl. Math. **177**, 243–267, 2005

[59] Glowinski R.: *Numerical Methods for Non-Linear Variational Problems*, Springer Series Comput. Phys., 1984

[60] Glowinski R., Marrocco A.: *Sur l'Approximation, par Élements Finis d'Ordre un, et la Résolution, par Pénalisation-Dualité, d'une Classe de Problèms de Dirichlet non Linéares*, Revue Française d'Automatique, Informatique, et Recherche Opérationelle **9**, 41–76, 1975

[61] Goldstein T., O'Donoghue B., Setzer S., Baraniuk R.: *Fast Alternating Direction Optimization Methods*, SIAM J. Imag. Sci. **7**, 1588–1623, 2014

[62] Golub G. H., van Loan C. F.: *Matrix Computations*, The Johns Hopkins University Press, 4th edition, 1996

[63] Golub G. H., Varga R. S.: *Chebyshev Semi-Iterative Methods, Successive Over-Relaxation Iterative Methods, and Second Order Richardson Iterative Methods, Part I*, Numer. Math. **3**, 147–156, 1961

[64] Golub G. H., Varga R. S.: *Chebyshev Semi-Iterative Methods, Successive Over-Relaxation Iterative Methods, and Second Order Richardson Iterative Methods, Part II*, Numer. Math. **3**, 157–168, 1961

[65] Gondzio J.: *Interior Point Methods 25 Years Later*, European J. Oper. Res. **218**, 587–601, 2012

[66] Gorenflo R., Mainardi F.: *Fractional Calculus: Integral and Differential Equations of Fractional Order*. In: Carpinteri A., Mainardi F. (eds.), *Fractals and Fractional Calculus in Continuum Mechanics*, 223–276, Springer, 1997

[67] Götschel S., Minion M. L.: *An Efficient Parallel-in-Time Method for Optimization with Parabolic PDEs*, SIAM J. Sci. Comput. **41**, C603–C626, 2019

[68] Greenbaum A.: *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997

[69] Greenbaum A., Pták V., Strakoš Z.: *Any Nonincreasing Convergence Curve is Possible for GMRES*, SIAM J. Matrix Anal. Appl. **17**, 465–469, 1996

[70] Gresho P. M., Sani R. L.: *Incompressible Flow and the Finite Element Method. Volume 1: Advection-Diffusion and Isothermal Laminar Flow*, Wiley, New York, 1998

[71] Griesse R., Volkwein S.: *A Primal-Dual Active Set Strategy for Optimal Boundary Control of a Nonlinear Reaction-Diffusion System*, SIAM J. Control Optim. **44**, 467–494, 2005

[72] Gunzburger M. D.: *Perspectives in Flow Control and Optimization*, SIAM, 2002

[73] Güttel S., Pearson J. W.: *A Rational Deferred Correction Approach to Parabolic Optimal Control Problems*, IMA J. Numer. Anal. **38**, 1861–1892, 2017

[74] Haber E., Hanson L.: *Model Problems in PDE-Constrained Optimization*, Tech. Rep. TR-2007-09, Emory University, 2007

[75] Hackbusch W.: *Multigrid Methods and Applications*, Springer Series in Computational Mathematics, Volume 4, Springer-Verlag, Berlin, 1985

[76] Hasan A., Foss B., Sagatun S.: *Flow Control of Fluids through Porous Media*, Appl. Math. Comput. **219**, 3323–3335, 2012

[77] Heidel G., Wathen A.: *Preconditioning for Boundary Control Problems in Incompressible Fluid Dynamics*, Numer. Linear Algebra Appl. **26**, e2218, 2019

[78] Heinkenschloss M.: *A Time-Domain Decomposition Iterative Method for the Solution of Distributed Linear Quadratic Optimal Control Problems*, J. Comput. Appl. Math. **173**, 169–198, 2005

[79] Heinkenschloss M., Leykekhman D.: *Local Error Estimates for SUPG Solutions of Advection-Dominated Elliptic Linear–Quadratic Optimal Control Problems*, SIAM J. Numer. Anal. **47**, 4607–4638, 2010

[80] Hestenes M. R., Stiefel E.: *Methods of Conjugate Gradients for Solving Linear Systems*, J. Res. Nat. Bur. Stand. **49**, 409–436, 1952

[81] Hintermüller M., Ito K., Kunisch K.: *The Primal-Dual Active Set Strategy as a Semismooth Newton Method*, SIAM J. Optim. **13**, 865–888, 2002

[82] Hintermüller M., Hinze M.: *A SQP-Semismooth Newton-Type Algorithm Applied to Control of the Instationary Navier–Stokes System Subject to Control Constraints*, SIAM J. Optim. **16**, 1177–1200, 2006

[83] Hinze M.: *Optimal and Instantaneous Control of the Instationary Navier–Stokes Equations*, Habilitation Thesis, Technische Universität Berlin, 2000

[84] Hinze M., Köster M., Turek S.: *A Space–Time Multigrid Method for Optimal Flow Control*. In: Leugering G., Engell S., Griewank A., Hinze M., Rannacher R., Schulz V., Ulbrich M., Ulbrich S. (eds.), *Constrained Optimization and Optimal Control for Partial Differential Equations*, 147–170, Springer, Basel, 2012

[85] Hinze M., Pinnau R.: *A Second Order Approach to Optimal Semiconductor Design*, J. Optim. Th. Appl. **133**, 179–200, 2007

[86] Hinze M., Pinnau R., Ulbrich M., Ulbrich S.: *Optimization with PDE Constraints*, Springer-Verlag, New York, 2008

[87] Hughes T. J. R., Brooks A.: *A Multidimensional Upwind Scheme with no Crosswind Diffusion*. In: Hughes T. J. R. (ed.), *Finite Element Methods for Convection Dominated Flows*, AMD **34**, 19–35, ASME, New York, 1979

[88] Ipsen I. C. F.: *A Note on Preconditioning Nonsymmetric Matrices*, SIAM J. Sci. Comput. **23**, 1050–1051, 2001

[89] Ito K., Kunisch K.: *Augmented Lagrangian Methods for Nonsmooth, Convex Optimization in Hilbert Spaces*, Nonlinear Anal. **41**, 591–616, 2000

[90] Ito K., Kunisch K.: *Semi-Smooth Newton Methods for State-Constrained Optimal Control Problems*, Systems Control Lett. **50**, 221–228, 2003

[91] Johnson C., Saranen J.: *Streamline Diffusion Methods for the Incompressible Euler and Navier–Stokes Equations*, Math. Comp. **47**, 1–18, 1986

[92] Kay D., Loghin D., Wathen A.: *A Preconditioner for the Steady-State Navier–Stokes Equations*, SIAM J. Sci. Comput. **24**, 237–256, 2002

[93] Klibanov M. V., Lucas T. R.: *Numerical Solution of a Parabolic Inverse Problem in Optical Tomography Using Experimental Data*, SIAM J. Appl. Math. **59**, 1763–1789, 1999

[94] Koeller R. C.: *Applications of Fractional Calculus to the Theory of Viscoelasticity*, J. Appl. Mech. **51**, 299–307, 1984

[95] Kopteva N., O'Riordan E.: *Shishkin Meshes in the Numerical Solution of Singularly Perturbed Differential Equations*, Int. J. Numer. Anal. Mod. **7**, 393–415, 2010

[96] Krendl W., Simoncini V., Zulehner W.: *Efficient Preconditioning for an Optimal Control Problem with the Time-Periodic Stokes Equations*. In: Abdulle A., Deparis S., Kressner D., Nobile F., Picasso M. (eds.), *Numerical Mathematics and Advanced Applications*, 479–487, Springer International Publishing, 2015

[97] Kuznetsov Y. A.: *Efficient Iterative Solvers for Elliptic Problems on Nonmatching Grids*, Russ. J. Numer. Anal. Math. M. **10**, 187–211, 1995

[98] Laub A. J.: *Matrix Analysis for Scientists and Engineers*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2004

[99] Leveque S., Pearson J. W.: *Parameter-Robust Preconditioning for Unsteady Stokes Control Problems*, PAMM **21**, e202100131, 2021

[100] Leveque S., Pearson J. W.: *Fast iterative solver for the optimal control of time-dependent PDEs with Crank–Nicolson discretization in time*, Numer. Linear Algebra Appl. **29**, e2419, 2022

[101] Leveque S., Pearson J. W.: *Parameter-Robust Preconditioning for Oseen Iteration Applied to Stationary and Instationary Navier–Stokes Control*, to appear in SIAM J. Sci. Comput., arXiv:2108.00282

[102] Li B., Liu J., Xiao M.: *A New Multigrid Method for Unconstrained Parabolic Optimal Control Problems*, J. Comp. Appl. Math. **326**, 358–373, 2017

[103] Lindenstrauss J., Preiss D.: *On Fréchet Differentiability of Lipschitz Maps between Banach Spaces*, Annals of Mathematics **157**, 257–288, 2003

[104] Lions J. L.: *Optimal Control of Systems Governed by Partial Differential Equations*, Springer, Berlin, 1971

[105] Maday Y., Turinici G.: *A Parareal in Time Procedure for the Control of Partial Differential Equations*, C. R. Math. **335**, 387–392, 2002

[106] Manteuffel T. A.: *The Tchebychev Iteration for Nonsymmetric Linear Systems*, Numer. Math. **28**, 307–327, 1977

[107] Manteuffel T. A.: *Adaptive Procedure for Estimation of Parameter for the Non-Symmetric Tchebychev Iteration*, Numer. Math. **28**, 187–208, 1978

[108] Mason J. C., Handscomb D. C.: *Chebyshev Polynomials*, CRC Press, 2003

[109] Mathew T. P., Sarkis M., Schaerer C. E.: *Analysis of Block Parareal Preconditioners for Parabolic Optimal Control Problems*, SIAM J. Sci. Comput. **32**, 1180–1200, 2010

[110] Matthies G., Tobiska L.: *Local Projection Type Stabilization Applied to Inf–Sup Stable Discretizations of the Oseen Problem*, IMA J. Numer. Anal. **35**, 239–269, 2015

[111] Meerschaert M., Scheffler H. P., Tadjeran C.: *Finite Difference Methods for Two-Dimensional Fractional Dispersion Equation*, J. Comp. Phys. **211**, 249–261, 2006

[112] Meerschaert M., Tadjeran C.: *Finite Difference Approximations for Fractional Advection Dispersion Equations*, J. Comp. Appl. Math. **172**, 65–77, 2004

[113] Meerschaert M., Tadjeran C.: *Finite Difference Approximations for Two-Sided Space-Fractional Partial Differential Equations*, Appl. Num. Math. **56**, 80–90, 2004

[114] Metzler R., Klafter J.: *The Restaurant at the End of the Random Walk: Recent Developments in the Description of Anomalous Transport by Fractional Dynamics*, J. Phys. A **37**, R161–R208, 2004

[115] Meurant G.: *Computer Solution of Large Linear Systems*, North Holland, 1999

[116] Miller K. S., Ross B.: *An Introduction to the Fractional Calculus and Fractional Differential Equations*, John Wiley & Sons, 1993

[117] Murphy M. F., Golub G. H., Wathen A. J.: *A Note on Preconditioning for Indefinite Linear Systems*, SIAM J. Sci. Comput. **21**, 1969–1972, 2000

[118] Napov A., Notay Y.: *An Algebraic Multigrid Method with Guaranteed Convergence Rate*, SIAM J. Sci. Comput. **34**, A1079–A1109, 2012

[119] Nocedal J., Wright S.: *Numerical Optimization*, Springer, New York, 2nd Edition, 2006

[120] Norburn S., Silvester D. J.: *Stabilised vs. Stable Mixed Methods for Incompressible Flow*, Comput. Methods Appl. Mech. Eng. **166**, 131–141, 1998

[121] Notay Y.: *An Aggregation-Based Algebraic Multigrid Method*, Electron. Trans. Numer. Anal. **37**, 123–146, 2010

[122] Notay Y.: *Aggregation-Based Algebraic Multigrid for Convection–Diffusion Equations*, SIAM J. Sci. Comput. **34**, A2288–A2316, 2012

[123] Notay Y.: *AGMG Software and Documentation*; see `http://agmg.eu/index.html`

[124] Oldham K. B., Spanier J.: *The Replacement of Fick's Laws by a Formulation Involving Semidifferentiation*, J. Electroanal. Chem. and Interfacial Electrochem. **26**, 331–341, 1970

[125] Oldham K. B., Spanier J.: *The Fractional Calculus*, Academic Press, New York, London, 1974

[126] Orozco C. E., Ghattas O. N.: *Massively Parallel Aerodynamic Shape Optimization*, Comput. Syst. Eng. **3**, 311–320, 1992

[127] Oseledets I. V. *et al.*: *TT-Toolbox Software*; see `https://github.com/oseledets/TT-Toolbox`

[128] Paige C. C., Saunders M. A.: *Solution of Sparse Indefinite Systems of Linear Equations*, SIAM J. Numer. Anal. **12**, 617–629, 1975

[129] Parikh N., Boyd S.: *Proximal Algorithms*, Found. Trends Optim. **1**, 127–239, 2014

[130] Pearson J. W.: *Fast Iterative Solvers for PDE-Constrained Optimization Problems*, D.Phil. Thesis, University of Oxford, 2013

[131] Pearson J. W.: *On the Development of Parameter-Robust Preconditioners and Commutator Arguments for Solving Stokes Control Problems*, Electron. Trans. Numer. Anal. **44**, 53–72, 2015

[132] Pearson J. W.: *Preconditioned Iterative Methods for Navier–Stokes Control Problems*, J. Comput. Phys. **292**, 194–207, 2015

[133] Pearson J. W.: *PDE-Constrained Optimization Models for Scientific Processes*, PAMM **18**, e201800253, 2018

[134] Pearson J. W., Gondzio J: *Fast Interior Point Solution of Quadratic Programming Problems Arising from PDE-Constrained Optimization*, Numer. Math. **137**, 959–999, 2017

[135] Pearson J. W., Gondzio J: *On Block Triangular Preconditioners for the Interior Point Solution of PDE-Constrained Optimization Problems*. In: Bjørstad P. E., Brenner S. C., Halpern L., Kim H. H., Kornhuber R., Rahman T., Widlund O. B. (eds.), *Domain Decomposition Methods in Science and Engineering XXIV*, 503–510, Springer, 2018

[136] Pearson J. W., Pestana J.: *Preconditioners for Krylov Subspace Methods: An Overview*, GAMM-Mitteilungen **43**, e202000015, 2020

[137] Pearson J. W., Porcelli M., Stoll M.: *Interior-Point Methods and Preconditioning for PDE-Constrained Optimization Problems Involving Sparsity Terms*, Numer. Linear Algebra Appl. **27**, 2020

[138] Pearson J. W., Stoll M.: *Fast Iterative Solution of Reaction–Diffusion Control Problems Arising from Chemical Processes*, SIAM J. Sci. Comput. **35**, B987–B1009, 2013

[139] Pearson J. W., Stoll M., Wathen A. J.: *Regularization-Robust Preconditioners for Time-Dependent PDE-Constrained Optimization Problems*, SIAM J. Matrix Anal. Appl. **33**, 1126–1152, 2012

[140] Pearson J. W., Wathen A. J.: *A New Approximation of the Schur Complement in Preconditioners for PDE-Constrained Optimization*, Numer. Linear Algebra Appl. **19**, 816–829, 2012

[141] Pearson J. W., Wathen A. J.: *Fast Iterative Solvers for Convection–Diffusion Control Problems*, Electron. Trans. Numer. Anal. **40**, 294–310, 2013

[142] Pearson J. W., Wathen A.: *Matching Schur Complement Approximations for Certain Saddle-Point Systems*. In: Dick J., Kuo F. Y., Woźniakowski H. (eds.), *Contemporary Computational Mathematics – A Celebration of the 80th Birthday of Ian Sloan*, 1001–1016, Springer, 2018

[143] Podlubny I.: *Fractional Differential Equations: an Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of their Solution and some of their Applications*, Academic Press, 1999

[144] Podlubny I., Petráš I., Vinagre B. M., O'leary P., Dorčák L'.: *Analogue Realizations of Fractional-Order Controllers*, Nonlinear Dynam. **29**, 281–296, 2002

[145] Pošta M., Roubíček T.: *Optimal Control of Navier–Stokes Equations by Oseen Approximation*, Comp. Math. Appl. **53**, 569–581, 2007

[146] Pougkakiotis S., Pearson J. W., Leveque S., Gondzio J.: *Fast Solution Methods for Convex Quadratic Optimization of Fractional Differential Equations*, SIAM J. Matrix Anal. Appl. **41**, 1443–1476, 2020

[147] Qiu Y., van Gijzen M. B., van Wingerden J.-W., Verhaegen M., Vuik C.: *Preconditioning Navier–Stokes Control Using Multilevel Sequentially Semiseparable Matrix Computations*, Numer. Linear Algebra Appl. **28**, e2349, 2021

[148] Quarteroni A., Rozza G.: *Optimal Control and Shape Optimization of Aorto-Coronaric Bypass Anastomoses*, Math. Models Methods Appl. Sci. **13**, 1801–1823, 2003

[149] Quarteroni A., Valli A.: *Numerical Approximation of Partial Differential Equations*, Springer, Berlin, 1997

[150] Ramage A.: *A Multigrid Preconditioner for Stabilised Discretisations of Advection–Diffusion Problems*, J. Comp. Appl. Math. **110**, 187–203, 1999

[151] Rees T.: *Preconditioning Iterative Methods for PDE Constrained Optimization*, D. Phil. Thesis, University of Oxford, 2010

[152] Rees T., Dollar H. S., Wathen A. J.: *Optimal Solvers for PDE-Constrained Optimization*, SIAM J. Sci. Comput. **32**, 271–298, 2010

[153] Rees T., Wathen A. J.: *Preconditioning Iterative Methods for the Optimal Control of the Stokes Equations*, SIAM J. Sci. Comput. **33**, 2903–2926, 2011

[154] Richardson L. F.: *The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam*, Philos. Trans. Roy. Soc. London Ser. A **210**, 307–357, 1911

[155] Saad Y.: *A Flexible Inner–Outer Preconditioned GMRES Algorithm*, SIAM J. Sci. Comput. **14**, 461–469, 1993

[156] Saad Y.: *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996

[157] Saad Y., Schultz M. H.: *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput. **7**, 856–869, 1986

[158] Samko S. G., Kilbas A. A., Marichev O. O. I.: *Fractional Integrals and Derivatives*, Gordon and Breach Science Publishers Yverdon, 1993

[159] Scalas E., Gorenflo R., Mainardi F.: *Fractional Calculus and Continuous-Time Finance*, Physica A **284**, 376–384, 2000

[160] Schenk C., Schulz V., Rosch A., von Wallbrunn C.: *Less Cooling Energy in Wine Fermentation – A Case Study in Mathematical Modeling, Simulation and Optimization*, Food and Bioproducts Processing **103**, 131–138, 2017

[161] Schiela A., Weiser M.: *Superlinear Convergence of the Control Reduced Interior Point Method for PDE Constrained Optimization*, Comput. Optim. Appl. **39**, 369–393, 2008

[162] Schöberl J., Zulehner W.: *Symmetric Indefinite Preconditioners for Saddle Point Problems with Applications to PDE-Constrained Optimization Problems*, SIAM J. Matrix Anal. Appl. **29**, 752–773, 2007

[163] Serra-Capizzano S., Tyrtyshnikov E.: *Any Circulant-Like Preconditioner for Multilevel Matrices is Not Superlinear*, SIAM J. Matrix Anal. Appl. **21**, 431–439, 2000

[164] Silvester D., Wathen A.: *Fast Iterative Solution of Stabilised Stokes Systems. Part II: Using General Block Preconditioners*, SIAM J. Numer. Anal. **31**, 1352–1367, 1994

[165] Simoncini V., Szyld D. B.: *Flexible Inner-Outer Krylov Subspace Methods*, SIAM J. Numer. Anal. **40**, 2219–2239, 2003

[166] Stoll M.: *One-Shot Solution of a Time-Dependent Time-Periodic PDE-Constrained Optimization Problem*, IMA J. Numer. Anal. **10**, 1554–1577, 2014

[167] Stoll M., Benner P., Onwunta A., Dolgov S.: *Low-Rank Solvers for Unsteady Stokes–Brinkman Optimal Control Problem with Random Data*, Comput. Methods Appl. Mech. Eng. **304**, 26–54, 2016

[168] Stoll M., Breiten T.: *A Low-Rank in Time Approach to PDE-Constrained Optimization*, SIAM J. Sci. Comput. **37**, B1–B29, 2014

[169] Stoll M., Pearson J. W., Maini P. K.: *Fast Solvers for Optimal Control Problems from Pattern Formation*, J. Comput. Phys. **304**, 27–45, 2016

[170] Stoll M., Wathen A.: *All-at-Once Solution of Time-Dependent Stokes Control*, J. Comput. Phys. **232**, 498–515, 2013

[171] Sudaryanto B., Yortsos Y. C.: *Optimization of Fluid Front Dynamics in Porous Media Using Rate Control. I. Equal Mobility Fluids*, Phys. Fluids **12**, 1656–1670, 2000

[172] Tilli P.: *Locally Toeplitz Sequences: Spectral Properties and Applications*, Linear Alg. Appl. **278**, 91–120, 1998

[173] Tobiska L., Lube G.: *A Modified Streamline Diffusion Method for Solving the Stationary Navier–Stokes Equations*, Numer. Math. **59**, 13–29, 1991

[174] Trefethen L. N., Bau D.: *Numerical Linear Algebra*, SIAM, 1997

[175] Tröltzsch F.: *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, Graduate Series in Mathematics, American Mathematical Society, 2010

[176] Tyrtyshnikov E. E.: *Optimal and Superoptimal Circulant Preconditioners*, SIAM J. Matrix Anal. Appl. **13**, 459–473, 1992

[177] Ulbrich M., Ulbrich S.: *Primal-Dual Interior-Point Methods for PDE-Constrained Optimization*, Math. Program. **117**, 435–485, 2009

[178] Varga R. S.: *Matrix Iterative Analysis*, Prentice Hall, 1962

[179] Wathen A. J.: *Realistic Eigenvalue Bounds for the Galerkin Mass Matrix*, IMA J. Numer. Anal. **7**, 449–457, 1987

[180] Wathen A. J.: *Preconditioning*, Acta Numerica **24**, 329–376, 2015

[181] Wathen A., Rees T.: *Chebyshev Semi-Iteration in Preconditioning for Problems Including the Mass Matrix*, Electron. Trans. Numer. Anal. **34**, 125–135, 2009

[182] Wathen A., Silvester D.: *Fast Iterative Solution of Stabilised Stokes Systems. Part I: Using Simple Diagonal Preconditioners*, SIAM J. Numer. Anal. **30**, 630–649, 1993

[183] Weiser M.: *Interior Point Methods in Function Space*, SIAM J. Control Optim. **44**, 1766–1786, 2005

[184] Weiser M., Schiela A.: *Function Space Interior Point Methods for PDE Constrained Optimization*, PAMM **4**, 43–46, 2004

[185] Wesseling P.: *An Introduction to Multigrid Methods*, John Wiley & Sons, 1992

[186] Yücel H., Stoll M., Benner P.: *Adaptive Discontinuous Galerkin Approximation of Optimal Control Problems Governed by Transient Convection–Diffusion Equations*, Electron. Trans. Numer. Anal. **48**, 407–434, 2018

[187] Zeng M., Zhang G.: *A New Preconditioning Strategy for Solving a Class of Time-Dependent PDE-Constrained Optimization Problems*, J. Comput. Math. **32**, 215–232, 2014

[188] Zhou G.: *How Accurate is the Streamline Diffusion Finite Element Method?*, Math. Comp. **66**, 31–44, 1997

[189] Zulehner W.: *Analysis of Iterative Methods for Saddle Point Problems: A Unified Approach*, Math. Comp. **71**, 479–505, 2001

[190] Zulehner W.: *Nonstandard Norms and Robust Estimates for Saddle Point Problems*, SIAM J. Matrix Anal. Appl. **32**, 536–560, 2011