A Hierarchical Flow Solver for Optimisation with PDE Constraints

---

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Michael Köster

A Hierarchical Flow Solver for Optimisation with PDE Constraints

Michael Köster

Dissertation eingereicht am: 20. 07. 2011
Tag der mündlichen Prüfung: 23. 11. 2011

Mitglieder der Prüfungskommission:

Prof. Dr. Stefan Turek (1. Gutachter, Betreuer)
Prof. Dr. Christian Meyer (2. Gutachter)
Prof. Dr. Joachim Stöckler
Prof. Dr. Rudolf Scharlau
Dr. Matthias Möller

'There are only two mistakes one can make along the road to truth;
not going all the way, and not starting.'

Buddha

Acknowledgements

This PhD thesis has been created in many years of research with a lot of highlighting discussions, interesting talks, visits at conferences all over Europe and frustrating periods of time in the development of the underlying code basis. During all this time, I met a lot of people who supported me in the one or the other way. The most important of them I would like to mention here.

First, I appreciate Prof. Dr. Stefan Turek from the TU Dortmund, my main supervisor. He gave me the opportunity to develop the underlying methods, always had an open ear and time for questions and discussions and gratefully supported me as a member of a big group of young scientists with trust and expertise as well as humanity, good mood and humour. In all these years, he was exceptionally patient with all the difficulties and setbacks that arise in the development of a finite element code due to to the large amount of technical and implementational details.

Second, I am grateful to Prof. Dr. Michael Hinze from the university of Hamburg. He provided greatest knowledge about theory and implementation of optimal control problems involving the nonstationary Navier–Stokes equations, patiently introduced the basic concepts, explained unknown details that appeared during the derivation of the underlying KKT systems and gave excellent support during the publication of articles and preprints about this topic. Together with Prof. Turek, he initiated this work as a part of a project in the SPP1253, see below.

Third, I am very much thankful to Prof. Dr. Christian Meyer. When he started his professorship 2011 in the TU Dortmund, he provided highly valuable support during the final phase of this PhD thesis.

Fourth, I would like to thank the main FEAT2 group, namely Peter Zajac, Matthias Möller and Raphael Münster for their great commitment in the development of the FEAT2 kernel. Due to their support, it was possible to develop all the efficient core routines in the finite element package FEAT2 upon which the KKT system solver has been set up.

Fifth, I appreciate Dominik Göddeke, Matthias Möller, Robert Strehl, Raphael Münster and Shafquat Hussain for proofreading this work. In particular, Dominik's comments allowed to greatly improve the structure of the whole thesis.

Sixth, I would like to thank Prof. Dr. Friedhelm Schieweck from the university of Magdeburg for enlightening and fruitful discussions about the Crank–Nicolson scheme, Jaroslav Hron (currently working at the Charles University in Czech) for his insight about finite

elements, Hogenrich Damanik for helping me to get insight to globalisation schemes together with Jaroslav Hron, and finally Abderrahim Ouazzi for his support concerning the edge oriented stabilisation technique.

This thesis has been created as part of the project 'Hierarchical Solution Concepts for Flow Control Problems' as part of the priority program SPP1253, funded by the DFG (Deutsche Forschungsgemeinschaft, TU 102/24-1+2). This program was a great opportunity for me to be part of a large community of scientists working in the field of optimisation and optimal control, and I am very grateful to the DFG that I was able to be part of it. Furthermore, I also like to express my gratitude to the NRW Graduate School of Production Engineering and Logistics.

For the final notes, I would like to switch to the german language.

Abschließend möchte ich noch meiner Familie danken, für eure Geduld und euer Verständnis wegen der vielen Stunden, die ich wegen dieser Arbeit nicht mit euch verbringen konnte. Ein besonderer Dank gilt dabei meinen Eltern, die mich während all der Jahre unterstützt haben wo immer es nötig wurde, und die während all dieser Zeit so unglaublich großen persönlichen Einsatz geleistet haben, um meinem Bruder und mir eine so gute Ausbildung zu ermöglichen.

Dortmund, July 20, 2011

Michael Köster

Contents

1

Introduction

In the beginning, there was an engineer coming to a mathematician.

> 'I have a problem, my device does not work, it sometimes even explodes. I cannot look inside. Can you create an accurate computer program that simulates my device so that I can see what happens?'
>
> The trouble begun. After hours, days, months or sometimes even years, the mathematician said: 'Ok, now I can simulate your problem and I can show you what's wrong.'
>
> And the engineer replied: 'That's great. But now, I'd like to ask you for a little favour. As you can see, the device is not very effective. Can you tell your program to optimise it for me?'
>
> And then, the *real* trouble begun...

This small example demonstrates two important issues in the world of industrial engineering and simulation:

- Optimisation is the most desirable and most natural extension of simulation and experiment. Whenever a physical process is simulated or an experiment is set up, the intention is to do something existing in a better way.
- As natural as optimisation is an extension to simulation and experiment, as challenging and time consuming the optimisation usually is.

The term 'challenge' is meant in a very general sense in this context and does not necessarily restrict to a high computational effort. Usually, problems like high memory requirements, ill-conditioned physical operators in underlying equations or ill-posed optimisation problems are faced, such that standard optimisation algorithms do not compute suitable solutions. In fact, every kind of optimisation problem has its own challenges.

As a special example, the PDE-constrained optimisation, or more precisely, the optimisation with partial differential equations as constraints, is one of the most challenging problems in the world of modern numerics. With increasing compute power in modern computers, this class attracts more and more attention in the industry. It allows target-oriented, automatic, computer-aided optimisation of physical processes without the need to construct too many samples. For example, an engineer designing a new sheet metal roller with specific physical requirements can ask a computer for the design instead of building the device five to ten times in a trial and error style until the requirements are met. This

is a big saving as the price for one such a unit used for production in the industry is about 10 000 to 500 000 dollar.[1]

And the application field of PDE-constrained optimisation is large — at least as large as the application field of simulation, i. e., it can be applied everywhere where PDEs are involved. Some famous classes where this method is successfully applied are:

- *Distributed Control*: A force term inside of a domain should be controlled. In industrial applications, this can be used for example...
  - ... to find an optimal heat source for a plate such that some regions are heated while others stay cold; for a thin plate, a proper heater can be used, while complex 3D domains (for example, a tumour region in a body [54]) can be heated up, e. g., by microwaves.
  - ... to find optimal doping profiles for MOSFET semiconductors; the doping profile describes the concentration of impurity in a pure semiconductor material (by 'foreign' atoms) and controls the electrical properties [88, 92]. Important, e. g., for the chip design in the hardware of modern computers.
  - ... to find an optimal magnetic field stirring a melt; used, e. g., in crystal growth for the production of semiconductors to keep a molten crystal in a homogeneous flow [11, 47, 67, 71].
- *Boundary control*: Force terms acting on the boundary of a domain should be controlled. Used for example...
  - ... to control the cooling of a tissue in medicine without destroying the internal structures [25, 94].
  - ... to control the heating of certain parts of a human body with a special probe; used in the *RF-ablation* to destroy tumour cells, cf. [4].
  - ... in optimal glass cooling, to prevent cracks due to an improper cooling process [88, 92].
  - ... to control a suction or injection into a flow; used, e. g., in aircraft design to reduce the friction, to control separation effects and to reduce turbulence around an airfoil [30, 136].
- *Shape optimisation*: The shape of an object or a domain is to be deformed to fulfil certain criteria. Used for example...
  - ... to optimise the shape of an airfoil, to reduce its drag or increase its lift at a given velocity [27, 125, 133, 134].
  - ... to optimise the shape of a racing yacht in order to reduce the drag/increase the speed [28].
  - ... to optimise the design of bypass shapes in order to reduce the shear rate in artificial grafts around occlusions of blood vessels [127].
  - ... to optimise the geometric structure of trusses, e. g., to reduce the weight while maintaining highest stability w. r. t. the load acting on the truss [44]. Important, e. g., for the construction of bridges, hip roofs or frame structures in buildings.

[1] Determined by searching for 'hydraulic roller bending machine' at `http://www.alibaba.com`.

1.1. Why is optimisation so much more complicated than simulation?

For the application of PDE-constrained optimisation, the knowledge of different fields has to be combined to be successful: A physical background to understand the nature of a partial differential equation that describes a physical process, a mathematical background to properly simplify and discretise the equations and a background in computer science to realise highly optimised computer programs that are able to solve the discrete problems with minimal resources in terms of CPU time and memory requirements.

Optimal control and optimal design is an active field of research in recent years, see for example [5, 7, 11, 43, 69, 92, 92, 94, 112–114, 117, 130] and many more. For an understanding of some of the difficulties arising in this field from an algorithmic point of view, it is instructive to consider some examples.

Low dimensional parameter optimisation The following example is an adaption of the shape optimisation of a NACA0012 wing profile. The example is only roughly sketched here for simplicity.

Consider a 2D domain $\Omega \subset \mathbb{R}^2$ with an embedded NACA airflow profile $\Omega_{\mathrm{NACA}} = \Omega_{\mathrm{NACA}}(\gamma, h)$, fixed at point M. For a fixed parabolic inflow with maximum velocity $Y_{\max}$, the lift of the airfoil should be maximised. The angle of attack $\gamma \in [0, \pi)$ as well as the height $h > 0$ can be modified, see Figure 1.1.

A possible formulation in a laminar Reynolds number regime involving the Navier–Stokes equations reads as follows. Let $\tilde{\Omega}(\gamma, h) := \Omega \setminus \overline{\Omega_{\mathrm{NACA}}}$ denote the domain without the airfoil profile. Furthermore, let $F_L(y, p)$ denote the lift coefficient of the airfoil (i. e., the lift force integrated over the wing profile, cf. [142]), depending on a velocity field $y : \Omega \to \mathbb{R}^2$ and a pressure $p : \Omega \to \mathbb{R}$. Both unknowns must fulfil the Navier–Stokes equations on the domain $\tilde{\Omega}(\gamma, h)$. Thus, the optimisation problem can be formulated as

$$J(\gamma, h) := F_L(y, p) \quad \longrightarrow \quad \max!$$

$$\begin{aligned} \text{s.t.} \quad -\nu\Delta y + y\nabla y + \nabla p &= 0 \qquad \text{in } \tilde{\Omega}(\gamma, h), \\ -\operatorname{div} y &= 0 \qquad \text{in } \tilde{\Omega}(\gamma, h), \end{aligned}$$

for $\nu > 0$, complemented by appropriate boundary conditions.

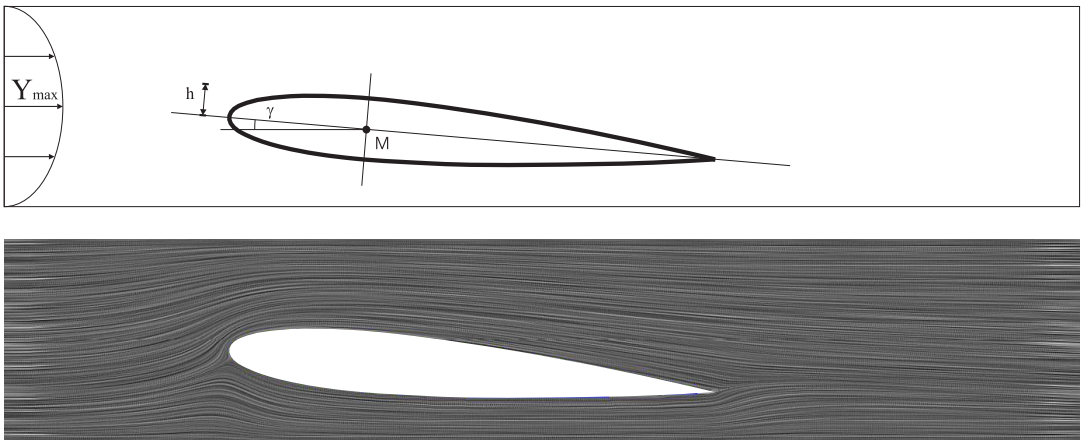

Figure 1.1: Two-parameter optimisation of an airfoil profile. Height h and angle of attack γ are unknown. Top: Geometry. Bottom: Streamlines of a possible flow around the airfoil.

Algorithmic issues This is an example for a 'low-dimensional' parameter optimisation with a PDE as constraint. The main flow — and thus, the functional to optimise — depends on a small number of parameters. A standard, black-box, derivative-free optimisation algorithm like Compass-Search or Nelder–Mead usually does a good job here. Such algorithms optimise the values of a target functional just by evaluating its values for different configurations (cf. [52, 107, 123]).

For the application of derivative-free methods, a method has to be provided that evaluates the target functional to be optimised. In the above example this means that a flow solver for the Navier–Stokes equations has to be provided as well as a routine that calculates $J(\cdot)$. This can be done in a black-box manner: Any arbitrary flow solver can be used as long as it is able to calculate the value of the functional $J(\gamma, h)$ based on the geometry parameters γ and h.

However, derivative-free optimisation algorithms are known to be efficient only if the number of design parameters is 'small', for example, less than three or five, depending on the application. Up to exponential growth in the total numerical costs can be expected if the number of design parameters is increased (see, e. g., the discussion about level-0 or 'black-box NAND' algorithms in [153]).

Medium and higher dimensional parameter optimisation Consider the optimal distributed control of the Poisson equation as described, e. g., in [139]. For a domain $\Omega \subset \mathbb{R}^{\dim}$ with $\dim = 2$ or $= 3$, this problem reads (in a short form, ignoring boundary conditions)

$$\begin{aligned} J(y,u) := \frac{1}{2}||y-z||^2_{L^2(\Omega)} + \frac{\alpha}{2}||u||^2_{L^2(\Omega)} \quad &\longrightarrow \quad \min! \\ \text{s.t.} \qquad -\Delta y = u \qquad &\text{in } \Omega \end{aligned} \tag{1.1}$$

for a relaxation parameter $\alpha > 0$ which is typically small, e. g., 10^{-3}. The variable $y : \Omega \to \mathbb{R}$ can be interpreted as a temperature distribution. The aim of this problem is to find a heat source $u : \Omega \to \mathbb{R}$ such that the temperature y matches a target temperature $z : \Omega \to \mathbb{R}$. For a 2D domain, the heat source can be realised, e. g., as a heater, for a 3D domain, a microwave heat source or similar can be used. As an example, Figure 1.2 illustrates[2], on a rectangular 2D domain, a target temperature z, the best possible approximation y and the control u used to generate y.

For the optimisation, nonlinear programming methods can be applied. In the first step, the problem is usually discretised and reformulated as a minimisation problem without constraints. One typical approach is to use penalty or barrier functions, cf. [12, Chapter 9]. A possible reformulation after applying the finite difference or finite element method reads

$$\tilde{J}_1(\vec{y},\vec{u}) := \frac{1}{2}||\vec{y}-\vec{z}||^2_{l^2} + \frac{\alpha}{2}||\vec{u}||^2_{l^2} + \mu\left(\frac{1}{2}\vec{y}^{\mathsf{T}}A\vec{y} - \vec{y}^{\mathsf{T}}\vec{u}\right) \quad \longrightarrow \quad \min!$$

with $\vec{y}$, $\vec{u}$, $\vec{z}$ referring to the vectors of degrees of freedom for y, u and z and A identifying the discrete Laplacian. The constant $\mu > 0$ is typically chosen large, e. g., 10^6. Note that the heat source u is specified in every point of the domain and usually discretised in the same way as the temperature y. This approach treats y and u independent of each other, which results in a large number of unknowns, and the constraint $-\Delta y = u$ is only satisfied approximately.

[2] In this figure, small over- and undershoots in y can be recognised. These are natural in this example: The temperature y is discretised with an unstabilised Q_1 finite element approach and approximates a discontinuous function z.

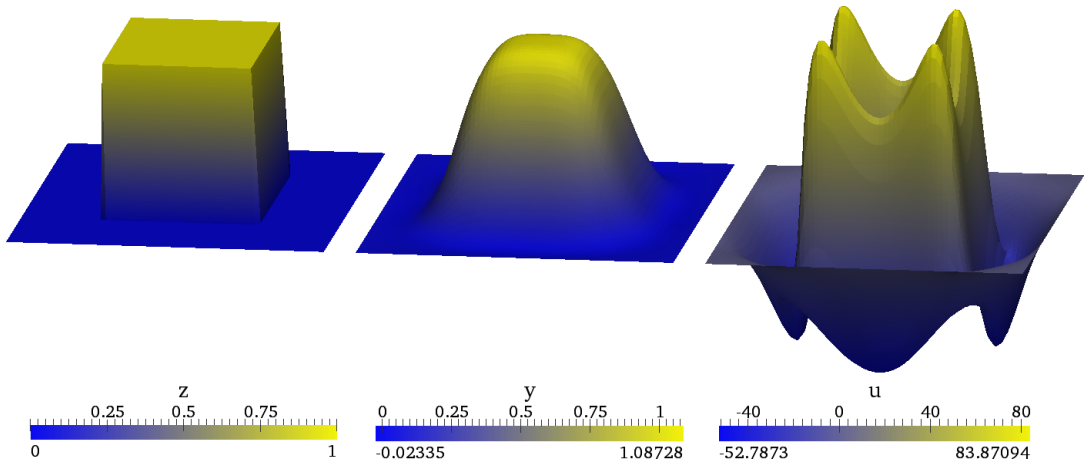

Figure 1.2: Optimal distributed control of the Poisson equation on a domain $\Omega = (0,1)^2$ in 2D on a regular 128×128 mesh. $\alpha = 10^{-5}$. Left: Target temperature z. Centre: Optimal approximation y. Right: Corresponding control u.

An even more common alternative in the context of optimal control (since it reduces the number of unknowns $J(\cdot)$ depends on, and which automatically guarantees the constraint $-\Delta y = u$ to be satisfied) is to define the reduced functional

$$\tilde{J}_2(\vec{u}) := \frac{1}{2}||\mathcal{S}(\vec{u}) - \vec{z}||_{l^2}^2 + \frac{\alpha}{2}||\vec{u}||_{l^2}^2 \quad \longrightarrow \quad \text{min!}$$

for the solution operator $\mathcal{S}(\vec{u}) = A^{-1}\vec{u}$. The temperature can be obtained from $y = \mathcal{S}(\vec{u})$ if necessary.

In the next step, an optimisation algorithm has to be applied. Derivative-free optimisation algorithms as used in the previous example can usually not be used anymore due to the size of the problem (see below). A more efficient way is to apply gradient methods of first and second order, see also [80] for an overview. First-order methods exploit the explicit knowledge of $DJ_*(\cdot)$. Second order methods exploit either the existence or even the explicit knowledge of the Hessian $\nabla^2 J_*(\cdot)$. Typical examples for first order methods are the steepest descent method or the conjugate gradient ('CG') method. Common second order methods are the Newton mathod and Quasi-Newton methods. A widely known representative from the group of Quasi-Newton methods is for example the BFGS algorithm ('Broydon–Fletcher–Goldfarb–Shanno'), see [12, 18, 110, 123, 125].

Algorithmic issues Unfortunately, all methods share a common crucial disadvantage: For a finer mesh resolution, either the optimisation runs into severe memory issues or the efficiency of the algorithm suffers:

- The steepest descent method as well as the conjugate gradient method require the explicit knowledge of the first derivative $\nabla \tilde{J}_*$ (which is usually available) and provide first order convergence. The CG method converges faster than the steepest descent method in practice. However, the convergence rate of both methods is known to depend on the condition of the operator in the target functional. As a consequence, in the above example, the efficiency of the iteration suffers with increasing refinement level due to the presence of the (discrete) Laplace operator in $\tilde{J}_*$. In the 2D case, as a rule of thumb, the number of iterations doubles with every mesh refinement.

- Quasi-Newton methods like the BFGS method are more advanced but also need a higher regularity of the problem. BFGS for example requires the knowledge of the gradient $\nabla \tilde{J}_*(\cdot)$ and assumes the existence of the Hessian. It stores an approximate Hessian which is calculated using $\nabla \tilde{J}_*$. As an advantage in comparison to the steepest descent or CG method, BFGS provides superlinear convergence, which is in particular useful for nonlinear problems. However, storing the recovery information leads to high memory requirements: For N unknowns, the algorithm internally stores a non-sparse, $N \times N$ matrix, i. e., the memory requirement is quadratically in the number of unknowns. The algorithm is therefore only applicable for a low to medium number of degrees of freedom. Software packages like MATHEMATICA [129] apply BFGS up to 250 unknowns, but there are examples where BFGS has been applied up to 10 000 unknowns.

  Example I Consider the above heat equation example on a domain $\Omega := (0,1)^2 \subset \mathbb{R}^2$, y and u discretised with a Q_1 finite element on a regular mesh with 8×8 cells. This results in ≈ 80 degrees of freedom for u (and y) and BFGS is applicable. In this form, the problem can be associated to the class of 'medium-dimensional' problems.

  Example II Consider the above heat equation example on a domain $\Omega := (0,1)^3 \subset \mathbb{R}^3$, y and u discretised with a Q_1 finite element on a regular mesh with $128 \times 128 \times 128$ cells. This results in $\approx 2 \cdot 10^6$ unknowns in u (and y) and the problem should be called a 'higher-dimensional' optimisation problem. BFGS can usually not be applied anymore.

- To cope with this issue, there are limited memory variants available like L-BFGS which store none or only a limited amount of recovery information. However, the general convergence behaviour for limited memory Quasi-Newton methods is similar to a CG algorithm, cf. [12, Chapter 8].

- The Newton method finally requires the explicit knowledge of the Hessian $\nabla^2 J_*(\cdot)$ — or at least a way to apply it. For nonlinear problems, it provides quadratic convergence, and for linear problems like above, it needs only one iteration. However, during the iteration, a sequence of linear systems has to be solved for the control u. This can be done, e. g., with standard linear solvers like GMRES or BiCGStab — which again depend on the condition of the operator, and their performance can be expected to degrade with increasing refinement level.

Conclusion The main issue which introduces problems to the above examples is the size of the discrete problem in combination with the analytic properties of the partial differential equation which is used as constraint. Even for problems with a higher regularity (i. e., where second order gradient information is available, which is assumed in the following), standard nonlinear optimisation algorithms usually either run out-of-memory or their performance degrades from a certain point on. The degrade is reasoned in mesh-dependent condition issues with the underlying discrete operator.

Unfortunately, many important optimisation problems belong to the class of medium and higher dimensional problems with ill-conditioned constraints, so black-box nonlinear programs are in general not applicable. The situation even worsens for time dependent optimal control which is illustrated later. At this point, specialised algorithms are needed that fully exploit the mathematical structure of the problem in order to be successful. In this context, the Lagrange multiplier technique has turned out to be a flexible tool to regain

some kind of black-box character. It allows to reformulate many minimisation problems as a well-structured set of equations. Solving minimisation problems can in this way be reduced to solving a set of linear or nonlinear equations. This allows completely different solution techniques to be applied.

1.2. The SQP approach — a remedy up to a certain extent

The Sequential Quadratic Programming ('SQP') approach has advanced to one of the standard methods in PDE-constrained optimisation in recent years. Using the Lagrange multiplier technique and exploiting second order gradient information, PDE-constrained minimisation problems can be reformulated as nonlinear equations which can be processed by appropriate solvers.

Considering for example the above minimisation problem (1.1), the Lagrange multiplier technique can be applied (cf. [139]) to derive an associated set of equations,

$$\begin{aligned} -\Delta y &= u, \\ -\Delta \lambda &= y - z, \\ u &= -\frac{1}{\alpha}\lambda \end{aligned}$$

for an auxiliary 'dual' temperature $\lambda : \Omega \to \mathbb{R}^{\dim}$. For simplicity, the boundary conditions are neglected. This set of equations is called 'Karush-Kuhn-Tucker' or short 'KKT' system. It defines the set of first order necessary optimality conditions, or 'KKT conditions', of a special Lagrange functional. For a nonlinear PDE as constraint, the KKT system is of course nonlinear, and solving the KKT system with a Newton method is referred to as 'SQP approach' in the literature.

The variable u can be eliminated in this example, which leads to the system

$$\begin{aligned} -\Delta y + \frac{1}{\alpha}\lambda &= 0, \\ -\Delta \lambda - \quad y &= -z. \end{aligned}$$

The variable u can be calculated in a postprocessing step.

What has been gained? The advantage of this approach becomes clear if, e. g., a finite element discretisation is applied. This leads to a linear system

$$\begin{pmatrix} A & \frac{1}{\alpha}M \\ -M & A \end{pmatrix} \begin{pmatrix} \vec{y} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \vec{z} \end{pmatrix} \tag{1.2}$$

with A denoting the discrete Laplacian, M the mass matrix and $\vec{y}$, $\vec{\lambda}$, $\vec{z}$ the vectors containing the degrees of freedom corresponding to y, λ and the right-hand side z. This allows to apply all kinds of solvers for linear systems. Smaller problems (e. g., Example I on a regular 2D mesh with 128×128 cells, corresponding to $\approx 30\,000$ unknowns) can usually be solved with a sparse LU decomposition solver. For larger problems, e. g., a preconditioned CG algorithm can be applied. With a proper preconditioner, this algorithm is advantageous to the unpreconditioned CG method and thus, superior to the Quasi-Newton methods known from nonlinear programming. Other possible solvers are for example the popular (preconditined) GMRES or BiCGStab method. All in all, it can be expected that problems with a size similar to Example II from above can be solved. Memory is not an issue as well.

The size of the system is only twice the size of a standard Poisson problem, and if solvers like the CG or BiCGStab are applied, the memory consumption increases linearly with the problem size.

What still remains Unfortunately, this is not the end of the story. The system matrix in (1.2) contains the Laplacian on the diagonal, and the condition of this matrix cannot be expected to be better than the condition of the discrete Laplace matrix itself. As a consequence, the convergence behaviour of usual linear solvers like CG or GMRES still degenerates with the problem size and lead to infeasible computational times for larger problems. And although the size of Example II seems to be large, in the context of PDE-constrained optimisation, this problem is small.

Example III Consider the optimal distributed control of the time dependent Stokes equations as described, e. g., in [84]. This problem reads (in short form, ignoring, e. g., the boundary conditions)

$$\begin{aligned} J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad &\longrightarrow \quad \min! \\ \text{s.t.} \quad y_t - \nu\Delta y + \nabla p &= u, \\ -\operatorname{div} y &= 0, \\ y(0,\cdot) &= y^0 \end{aligned}$$

for $\nu > 0$, an initial condition $y^0 : \Omega \to \mathbb{R}^{\dim}$ and is defined on a space-time cylinder $\mathcal{Q} = (0,T) \times \Omega$. The variable $T > 0$ corresponds to a maximum time and $\Omega \subset \mathbb{R}^{\dim}$ with $\dim = 2$ or $= 3$ defines a domain in space. Furthermore, $y : \mathcal{Q} \to \mathbb{R}^{\dim}$ describes a velocity, $u : \mathcal{Q} \to \mathbb{R}^{\dim}$ a right-hand side force term and $p : \mathcal{Q} \to \mathbb{R}$ a pressure term. This problem aims at introducing a force u (which is realised in practice, e. g., by a magnetic field) such that the flow y matches a prescribed flow $z : \mathcal{Q} \to \mathbb{R}^{\dim}$ on a given time interval $(0,T)$. In a modified form, such a problem can be used, e. g., to control a flow in a crystal growth process [11]. By modifying the flow, the quality of crystals growing from a liquid can be controlled.

Now, the Lagrange multiplier technique can be applied. The KKT system associated with the above minimisation problem reads

$$\begin{aligned} y_t - \nu\Delta y + \nabla p &= -\frac{1}{\alpha}\lambda, \\ -\lambda_t - \nu\Delta\lambda + \nabla\xi &= y - z, \\ -\operatorname{div} y = -\operatorname{div}\lambda &= 0, \\ y(0,\cdot) &= y^0, \\ \lambda(T,\cdot) &= 0. \end{aligned}$$

with a 'dual velocity' $\lambda : \mathcal{Q} \to \mathbb{R}^{\dim}$ and a 'dual pressure' $\xi : \mathcal{Q} \to \mathbb{R}$, u eliminated as above.

Algorithmic issues In comparison to the Poisson example (1.1) from above, a crucial difference can immediately be recognised: The KKT system contains two equations, where one is given forward in time and the other one backward in time. Both are coupled via the right-hand side. Thus, y and λ are fully coupled on the space-time cylinder $\mathcal{Q}$. This has severe consequences for a discretisation. Plainly speaking, the time acts as additional

dimension. For an optimisation of the nonstationary Stokes equation in 3D, the optimisation effectively has to deal with a 4D discretisation. This quickly leads to a tremendously large number of degrees of freedom for finer meshes with a lot of timesteps.

Example III (cont.) Consider Example III on the domain $\Omega = (0,1)^3 \subset \mathbb{R}^3$ and the time interval $[0,T] = [0,1]$. The spatial domain can be discretised, e. g., on a regular $128 \times 128 \times 128$ mesh with the Q_2 finite element for the velocity and the P_1^{disc} element for the pressure. A discretisation in time with 128 timesteps based on the implicit Euler timestepping scheme leads to a total number of unknowns of $\approx 1.4 \cdot 10^{10}$.

For problems of this size or larger, the application of standard linear solver techniques is barely reasonable: The condition of the operator in the above primal/dual system cannot be expected to be better than the condition of the standard Laplace operator. As a consequence, the convergence rate of any usual linear solver (like, e. g., the (preconditioned) CG or GMRES method) suffers to such an extent that the problem cannot be computed in reasonable time. This aspect is independent of whether a serial or even a parallel computer system is used, it is in the nature of the mathematical algorithm.

Furthermore, problems are usually nonlinear and unstable, which induces additional difficulties in the design of algorithms and the underlying discretisation. A nonlinearity in the PDE induces a nonlinearity in the KKT system, and stability problems, e. g., by dominant convection, are an issue in the KKT system as well. A simple example reads as follows:

Example IV As an extension to the optimal control of the Stokes equation in Example III, consider the optimal distributed control of the time dependent Navier–Stokes equations, which reads (in simplified form, ignoring, e. g., the boundary conditions)

$$\begin{aligned} J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad &\longrightarrow \quad \min! \\ \text{s.t.} \quad y_t - \nu\Delta y + y\nabla y + \nabla p &= u, \\ -\operatorname{div} y &= 0, \\ y(0,\cdot) &= y^0. \end{aligned}$$

The corresponding KKT system has the form

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p &= -\frac{1}{\alpha}\lambda, \\ -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi &= y - z, \\ -\operatorname{div} y = -\operatorname{div}\lambda &= 0, \\ y(0,\cdot) &= y^0, \\ \lambda(T,\cdot) &= 0. \end{aligned}$$

fot $\nu > 0$. The PDE from the constraint (including the nonlinearity) is inherited by the KKT system. In particular, stability issues for the Navier–Stokes equations can be expected to be stability issues for the KKT system as well.

Conclusion and main challenges

PDE-constrained optimisation is a field in modern numerics which is characterised by two major difficulties from the algorithmic point of view:

- The optimisation inherits all difficulties that appear also during the simulation of the underlying PDE; a nonlinearity in the PDE induces a nonlinearity in the optimal optimisation, stability issues known for the underlying PDE are usually stability issues for the optimal control, and ill-conditioned operators in the PDE are ill-conditioned operators in the optimisation as well.

- Already rather small problems result in a tremendously large number of degrees of freedom, at least in the time dependent case. Standard methods for nonlinear programming are usually not applicable. The SQP approach is a remedy to some extent: Using the Lagrange multiplier technique allows to reformulate the underlying minimisation problem as a set of equations which can be solved by linear and nonlinear solvers. As a result, the memory requirement increases linearly with the problem size and preconditioning techniques can be applied to accelerate convergence. However, for high dimensional problems, the convergence of the underlying solvers usually suffers due to the problem size and the nature of the underlying PDE to such an extent that the problem is unsolvable in reasonable time.

Altogether, one of the main challenges in dealing with really large-scale problems from PDE-constrained optimisation is the loss of efficiency for increasing problem size. This is a very disappointing point because for the simulation of a PDE, the problem is solved in many cases, in particular for CFD problems:

Computational effort in nonstationary simulations Consider the following example.

Example V The simulation of the nonstationary Stokes equations reads

$$\begin{aligned} y_t - \Delta y + \nabla p &= f, \\ -\operatorname{div} y &= 0, \\ y(0,\cdot) &= y^0, \end{aligned}$$

(complemented by the boundary conditions) for some right-hand side $f : \mathcal{Q} \to \mathbb{R}^{\dim}$, $y^0 : \Omega \to \mathbb{R}^{\dim}$ denoting an initial condition. Applying, e. g., the implicit Euler time discretisation and a discretisation in space with finite elements, the discrete system

$$\begin{aligned} y_n^h - k\Delta y_n^h + k\nabla p_n^h &= k f_n^h + y_{n-1}^h, \\ -\operatorname{div} y_n^h &= 0, \\ y_0^h &= y^0, \end{aligned} \tag{1.3}$$

is obtained, $n = 1, 2, 3, ...$, with y_0^h, y_1^h,... and p_1^h, p_2^h,... referring to the discrete velocity/pressure solutions at the different timesteps and $f_1^h, f_2^h, ...$ denoting the corresponding right-hand sides.

The time discretisation is decoupled from the space discretisation. Doubling the number of timesteps therefore essentially results in a doubling of the computational time. Similarly, a refinement in space (in 2D) would in the optimal case lead to an increase of the CPU time by a factor of four (since a spatial refinement leads approximately to four times

more unknowns). However, it is not straightforward to reach this CPU time increase with standard numerical methods.

The Stokes operator in space is ill-conditioned, or more precisely, the condition is mesh-dependent and therefore influences the convergence properties of numerical solvers. At the first glance, this is unsatisfactory: In time interval n, Equation (1.3) induces a linear system in space which is usually solved with standard iterative solution schemes like GMRES or BiCGStab. As a consequence, a refinement of the spatial mesh would lead to a degrade of the solver convergence behaviour and a disproportional increase of the total CPU time.

This had been a problem for a long time but it was fortunately solved. The idea has been to apply a hierarchical ('multigrid') solution strategy which exploits the elliptic character of the Laplace operator on a hierarchy of refinement levels. In the beginning, this approach has been introduced for the Poisson equation [9] and has later been generalised to the Stokes and Navier–Stokes equations, see for example [31, 75]. The main advantage is that the solver performance does not degrade with the refinement level. Hence, problems in space can be solved with a numerical effort (in terms of CPU time and memory requirements) proportional to the number of unknowns. In advanced software packages [68], this technique has been established as standard method for the simulation on fine meshes.

All in all, refining the spatial mesh and doubling the number of timesteps, the problem size and the CPU time in contemporary CFD simulation packages increase by a factor of eight (in 2D, a factor of 16 in 3D). The complexity of the simulation is therefore $\mathcal{O}(n)$ in terms of memory and CPU time, n denoting the number of degrees of freedom in space and time.

Design aim for optimal control solvers Of course, an optimisation cannot be expected to be carried out faster than a simulation of the underlying PDE — but the optimisation should also not take much longer. It should have at least the same complexity. This can be expressed as follows.

Defining the fastest possible simulation of a PDE as a reference, the main aim in the design of a solver strategy for PDE-constrained optimisation is to create a solver that solves the underlying KKT system with computational costs similar to the simulation. More precisely, the solver should fulfil

$$\frac{\text{effort for the optimisation}}{\text{effort for the simulation}} \leq C$$

where *effort for the simulation* is measured by the best available numerical methods. The constant $C > 0$ should have a 'moderate' size which is independent of the refinement of the underlying problem. In practice, such a relationship is highly favourable and can hardly be achieved. It gives the end user the ability to plan the resources for a specific (optimisation) task in terms of memory usage and CPU time — and in the industry, the term 'resources' is equivalent to 'money'. Just by measuring the resources needed for a small-scale simulation, those necessary for a large scale optimisation can be estimated. Thus, the end user is able to determine in advance whether or not the available resources are sufficient to reach a required accuracy.

In particular for the Stokes example from above, a KKT system solver should be able to solve the underlying KKT system with linear complexity, i. e., with CPU time (and memory) requirements $\mathcal{O}(n)$, n denoting the number of degrees of freedom in space *and* time. Without exploiting additional mathematical structure, this is usually not the case. As indicated above in the notes to Example III, the performance of standard linear solvers

applied to the discrete KKT system will degrade on higher mesh levels due to the ill-conditioning of the KKT system for increasing refinement level.

However, the methods from the simulation can serve as as source of inspiration for the design of a KKT system solver with an improved complexity: Solving linear systems in space during the simulation (of the heat equation or the Stokes equations, e. g.) has been accelerated by a hierarchical approach, exploiting the elliptic character of the Laplace operator. Similar to the Laplace operator in space, the operator that defines the left-hand side of the KKT system has an elliptic character on the space-time cylinder as well. This key property, which is also illustrated in this work, can be exploited by a hierarchical algorithm.[3]

1.3. The story about hierarchical solution concepts

To the best of the author's knowledge, one of the first milestones for the application of hierarchical solution strategies in the field of optimal control was set by Hackbusch [72]. Based on his work, Büttner analysed in [34] a hierarchical solver for the heat equation based on finite elements in space and proved mesh independent convergence. Independent of this work, Borzì created linear and nonlinear space-time multigrid techniques [21–24] based on a finite difference discretisation in space and time, and some of his Time-Splitted-Gauß–Seidel techniques are the root of the time-smoothing algorithms in this work.

The aim of the work The present work demonstrates and analyses one possibility to generalise hierarchical solution methods from the heat equation to the much more general case of the nonstationary Navier–Stokes equations. Instead of using an optimisation algorithm on top of an existing flow solver (which is often the only possibility if commercial software is used), a fully integrated KKT system solver is designed and implemented from the scratch. Existing code and concepts from a modern, highly efficient open-source flow solver are changed and adapted to simultaneously process primal and dual variables. The approach therefore falls into the highest, most intrusive and most sophisticated category of level-6 or 'Full-Newton SAND' methods in the solver classification introduced in [153].

For high efficiency, robustness and flexibility, most advanced flow solver techniques known from the CFD simulations are the core of the algorithmic approach. These techniques allow to solve problems from PDE-constrained flow control with computational costs of the order of magnitude similar to today's fastest numerical flow solvers. This brings PDE-constrained optimisation of fluid flow one step closer to an application:

- The usage of so-called *local pressure Schur complement techniques* in space allows to deal with a large variety of equations, of standard as well as of saddle point type. This renders the method capable of being later applied to more complex problems like non-Newtonian or non-isothermal fluids — the first point important for instance in biomathematics/hemodynamics (blood flow), the second one being of special interest in the field of crystal growth [11, 71].

[3] An alternative approach which also aims at a small ratio between the computational costs for optimisation and simulation was developed in [77]. The method acts iteratively and treats primal and dual variables monolithically during the iteration as well. It was successfully used, e. g., in the one-shot parameter optimisation of a nonlinear climate model realised in [110] or in the one-shot shape optimisation techniques in [125]. However, to the best of the author's knowledge, it has only been applied to problems with a stationary, low-dimensional design parameter set u, thus no comparison can be made to the methods in this work regarding efficiency and robustness.

- The combination of local pressure Schur complement techniques with monolithic multigrid methods in space results in highly efficient preconditioning techniques which converge with level-independent convergence rates. The computational costs scale linearly with the problem size.
- For flexibility in the solution process of space and space-time subproblems, a large variety of preconditioning techniques is introduced, from weak and fast for simple problems up to strong and more expensive for more ill-conditioned tasks.
- The discretisation in space is carried out with finite elements. Due to this approach, very general unstructured meshes can be used. Furthermore, the finite element approach provides in a straightforward way the possibility to use higher order discretisations. For smooth problems, this provides a high accuracy for a small number of unknowns in space. For less smooth problems which possibly have to be stabilised, stabilisation techniques like the edge-oriented jump stabilisation are applied and work well in the numerical tests.
- On the other hand, a large variety of time discretisation techniques are applicable for the discretisation in time. The work demonstrates how to realise not only the first order implicit Euler but also the second order Crank–Nicolson scheme. Basically, the method itself is capable also to deal with higher order methods like Fractional-Step or the cGP(m) schemes introduced in [96]. For solutions which are sufficiently smooth in time, this allows to drastically reduce the number of timesteps.

The final step, the application of monolithic multigrid methods on the whole space-time cylinder in combination with a Newton solver for the nonlinearity, is the key to efficiency. All in all, a robust and fast solver strategy is presented which is able to solve the underlying KKT system of an optimisation problem in such a way that

$$\frac{\text{effort for the optimisation}}{\text{effort for the simulation}} \leq C$$

with a moderate constant C which is about $C \approx 20$–50 in most numerical tests. In many examples (including the optimal control of a nonstationary flow at Re=100), there is even $C < 30$. It has to be kept in mind that today's best numerical methods apply multigrid algorithms in space and solve a simulation with complexity $\mathcal{O}(n)$ (n the number of degrees of freedom in space and time), i. e., the CPU time grows linearly with the spatial refinement and the number of timesteps. In particular, because of the above relation, also the optimisation scales linearly with the problem size which is the major challenge from above.

1.4. Thesis contribution

To the best of the author's knowledge, no hierarchical solution concept for the optimal control of the Stokes and Navier–Stokes equations has been realised up to now. In [34], a hierarchical concept has been realised for the heat equation and in [11], the discretisation strategy applied in Chapter 2 has been proposed for Stokes and Navier–Stokes-like equations. This work closes the gap and presents for the first time a realisation of a hierarchical concept for the Stokes and Navier–Stokes equations with an extensive numerical analysis.

The definition and realisation follows an 'uncommon' strategy compared to the methods used by many authors in the literature. Instead of discretising the control u, the primal and dual variables are discretised in a monolithic way, and the control is implicitly used

without being discretised. This allows to define robust preconditioners that reduce all global operations on the space-time cylinder to local linear systems in space. These local systems combine primal and dual variables, have a structure similar to the local systems solved during a simulation and are solved with similar techniques. More precisely, a multigrid solver in space is applied. A difficulty in this context is that the local systems exhibit a saddle-point structure which necessitates the definition of appropriate smoothers. The concept of *local pressure Schur complement* ('Vanka-type') smoothers known from Computational Fluid Dynamics ('CFD') applications is adapted. To the best of the author's knowledge, this concept has never been used by other authors for combined primal-dual systems in the context of optimal control for the Navier–Stokes equations, and numerical tests illustrate the robustness of this approach.

Another issue addressed in this work is the time discretisation of nonstationary optimal control problems with the Crank–Nicolson scheme. The idea to modify this scheme in such a way that the dual variables are located in the midpoints of the time intervals is rather new and to the best of the author's knowledge only used by a very few people. This work presents for the first time how this approach can be applied to discretise the KKT system for the optimal control of the Navier–Stokes equations. In particular, this involves the derivation of the corresponding Newton system in order to apply the Newton algorithm. Furthermore, this work illustrates how to create appropriate prolongation/restriction operators in the multigrid context. For this purpose, the concept of 'discrete abstract functions' is introduced which allows to adapt notations from 1D multigrid schemes for finite difference discretisations in space to functions on the space-time cylinder.

Finally, the numerical analysis in this work contains an overview about the applicability of the edge oriented stabilisation technique. This technique has mainly been used for the simulation of the nonstationary Navier–Stokes equations, but to the best of the author's knowledge, it has not yet been applied in the context of optimal control of transient flows.

1.5. Thesis outline

This work is divided into two main parts. The first part, Chapter 2 to Chapter 4, contains descriptions of model problems, discretisation strategies and solver components.

Chapter 2 formulates a set of fundamental model problems for the heat equation as well as the Stokes and Navier–Stokes equations. Applying the Lagrange multiplier technique, the model problems are reformulated as sets of equations, and it is demonstrated that these exhibit an elliptic character if simultaneously considered in space and time. In the following, a special time discretisation recipe is applied to discretise the equations in time with the implicit Euler and general θ-schemes. In combination with an additional finite element discretisation in space, fully discrete systems are derived that couple all degrees of freedom. This discretisation strategy serves as a basis for the algorithm in Chapter 3.

Chapter 3 focuses on a detailed description of the solver methodology and all components that have to interact with each other. This chapter is the first core of this work. In the beginning, the discretisation scheme from Chapter 2 is applied to a hierarchy of meshes in space and time. This way, a hierarchy of discrete problems is defined. On the finest level, a nonlinear space-time solver is set up to solve for the nonlinearity. In the nonlinear loop, a monolithic space-time multigrid solver takes care of linear subsystems which have to be solved for the preconditioning of the nonlinear system. The chapter introduces all necessary solver components, defines appropriate

prolongation/restriction operators to traverse the problem hierarchy and describes how solvers for linear block systems can be adapted to create space-time smoothers. In the end, all space-time preconditioning operations are reduced to local linear systems in space that can be processed with a monolithic multigrid solver based on a hierarchy of finite element spaces. An important part in this context, in particular for the control of the Stokes and Navier–Stokes equations, is the description of *local pressure Schur complement smoothers*. This technique allows to apply monolithic smoothing to local saddle-point subsystems in space.

Chapter 4 is an appendix chapter for the description of the discretisation and the solver. The chapter introduces some extensions to common, more general minimisation problems than the model problems in Chapter 2, e. g., more general boundary conditions and constraints in the control. This allows more flexibility in numerical tests.

Chapters 5 to 8 form the second core of this work: The solver methodology derived in the previous chapters is applied to numerous example problems ranging from the optimal control of the heat equation to the control of the nonstationary Navier–Stokes equations. The chapters analyse the efficiency of the solver and its components in various situations, give advice concerning the choice of problem/solver parameters and highlight the advantages and disadvantages of different discretisation techniques in relation to the accuracy of the solution.

Chapter 5 starts with a numerical analysis of the linear space-time solver and its components. More precisely, this chapter is restricted to linear, analytical test examples based on the heat equation and the Stokes equations. In a single-grid, two-grid and multigrid setting, efficiency and robustness of the different solver components (single-grid solvers, smoothers, prolongation/restriction operators,...) are discussed. The tests include low and higher order discretisations, and a final numerical example illustrates the influence of the choice of the multigrid cycle to the complexity/CPU time of the solver. In particular it is shown that some combinations of space-time hierarchy and multigrid cycle have to be avoided.

Chapter 6 is an intermediate chapter that deals with the discretisation. In the beginning, the error in the solution is analysed with respect to the choice of the finite element space and the time discretisation scheme. The chapter demonstrates that the space and the time discretisation are related and gives advice about the choice of the coupling between the spatial mesh to the time mesh. Finally, the chapter discusses the accuracy of the modified Crank–Nicolson scheme and demonstrates its accuracy in comparison to the 'traditional' Crank–Nicolson scheme. In particular, the second order accuracy is numerically validated.

Chapter 7 extends the numerical analysis of the solver components from Chapter 5 to non-analytical test examples of the 'Driven–Cavity' type known from CFD benchmarks. The chapter contains an in-depth discussion of efficiency and robustness of the linear and nonlinear solver with focus on the Stokes and Navier–Stokes equations. The convergence properties of the linear solver are analysed, e. g., with respect to regularisation parameters and anisotropies in the space-time mesh. The second order convergence of the Newton solver is verified, and a CPU time comparison reveals that for the considered example, the optimisation is by a factor $C \approx 10$–12 more expensive than the simulation. Finally, the superlinear convergence of the semismooth Newton method for the case of constrained control is verified.

Chapter 8 forms the concluding chapter of the numerical tests. In the beginning, test and benchmark examples of the well-known 'Flow–Around–Cylinder' type at Re=20 and Re=100 are defined. The aim of this chapter is to find first quantitative benchmark reference results for distributed control flow solvers and to demonstrate the relation between the accuracy of the discretisation and the numerical effort that has to be invested to solve the underlying discrete systems. With the help of drag and lift coefficients representing physical forces, the influence of different refinement levels, different space and time discretisations and different regularisation parameters are shown. A solver discussion illustrates the relation between the choice of the discretisation and the invested numerical effort and demonstrates the impact of higher order discretisations on accuracy and efficiency. Furthermore, a CPU time comparison shows that for the considered Re=100 case, the optimisation is by a factor $C \approx 20$–50 more expensive than a corresponding simulation. The chapter concludes with a discussion of the applicability of the edge oriented stabilisation technique in optimal control. Examples demonstrate the influence of the stabilisation onto the efficiency of the solver and the accuracy of the discretisation.

Chapter 9 gives a conclusion and a discussion of further extensions of the methodology. The work closes with some appendix chapters. In Appendix A, the proposed solver approach is applied to further well known model problems which are important in practice. Appendix B demonstrates the incorporation of common globalisation strategies into the proposed solver algorithm. These aim for increased robustness and global convergence. Appendix C briefly compares the proposed solution strategy to other existing hierarchical methods from the literature and Appendix D motivates the special Crank–Nicolson discretisation that was chosen in Chapter 2. Finally, Appendix E gives an overview about the definition of parametric and nonparametric finite elements that are used in the numerical tests for the discretisation in space.

Computational resources

Chapters 5 to 8 contain numerous numerical tests. The computations have been carried out at the LiDOng cluster of the TU Dortmund using Intel Xeon E5450 processors with 3.00GHz and 16GB RAM. The flow solver was written in Fortran 90 based on the FEAT2 library developed at Lehrstuhl III für Angewandte Mathematik und Numerik, Fakultät für Mathematik, TU Dortmund. For compiling the code, the Intel Fortran compiler 12 was used in combination with the GOTO BLAS for linear algebra processing [36] and the UMFPACK library for solving linear systems of moderate size [48].

1.6. Publications of the author

This work was created during the research in the DFG project 'Hierarchical Solution Concepts for Flow Control Problems' which is a part of the priority program SPP1253. During the development of this project, a number of articles and preprints were published or submitted for publication by the author of this thesis. This work contains detailed descriptions and updated results from these papers, in particular also results which did not exist at the time of publication/submission or had to be skipped due to page limitations.

The first two preprints, [89] and [90] have been published in the course of the second application to extend the grants after a first project period. [89] gives a first introduction in the solution strategy for linear equations while [90] extends the methods to nonlinear equations. A variant of these articles was submitted for publication to [91]. Finally in 2010,

a first survey of the method was accepted for publication in [93] as part of a proceedings book covering the first period of the project.

Apart from articles in the context of optimal control, the author contributed a section about non-parametric finite elements to [109]. Appendix E of this thesis contains an introduction to this topic with more details that were able to be published in this article. In particular, this appendix introduces all types of finite elements used in the numerical tests, e. g., the element $\tilde{Q}_1$.

1.7. General terms and definitions

In this work, a couple of symbols and definitions are used, which are generally introduced when they are needed. However, some general notations that belong to the mathematical standard are given here for reference in advance.

The underlying domain With $\Omega \subset \mathbb{R}^{\dim}$ ($\dim = 2$ or $= 3$), a bounded domain with smooth boundary $\Gamma := \partial\Omega$ is denoted. For $T > 0$, the interval $[0, T]$ defines a time interval under consideration and $\mathcal{Q} := (0, T) \times \Omega$ a space-time domain with boundary $\Sigma := (0, T) \times \Gamma$.

Standard operators The symbol Δ refers to the Laplace operator in space and ∇ to the usual gradient, i.e. $\Delta q = \partial_{x_1 x_1} q + \partial_{x_2 x_2} q + ...$ for a smooth function $q : \mathbb{R}^{\dim} \to \mathbb{R}$ and $\nabla q = (\partial_{x_1} q,\, \partial_{x_2} q,\, ...)^{\mathsf{T}}$.

For a smooth function $y : \mathbb{R}^{\dim} \to \mathbb{R}^{\dim}$, Δy denotes the Laplace operator in space applied to every component. Furthermore for two differentiable functions $v, w : \mathbb{R}^{\dim} \to \mathbb{R}^{\dim}$, the operators $v\nabla w$, $(\nabla v)^{\mathsf{T}} w$ are defined in the usual way; in particular for $v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$ and $w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ in the case $\dim = 2$, these operators yield

$$v\nabla w = (\nabla w)v = \begin{pmatrix} v_1 \partial_{x_1} w_1 + v_2 \partial_{x_2} w_1 \\ v_1 \partial_{x_1} w_2 + v_2 \partial_{x_2} w_2 \end{pmatrix},$$
$$(\nabla v)^{\mathsf{T}} w = w(\nabla v)^{\mathsf{T}} = \begin{pmatrix} \partial_{x_1} v_1 w_1 + \partial_{x_1} v_2 w_2 \\ \partial_{x_2} v_1 w_1 + \partial_{x_2} v_2 w_2 \end{pmatrix}.$$

For functions $y, v, w : \mathcal{Q} \to \mathbb{R}^{\dim}$ and $q : \mathcal{Q} \to \mathbb{R}$, the above operators are meant to be applied in space, which is sometimes denoted by a subscript x to avoid ambiguity, e. g., $\Delta y = \Delta_x y = \Delta_x y(t, \cdot)$ for a fixed $t \in (0, T)$.

The operator ∂_t denotes the time derivative; it is usually abbreviated by a subscript t on the variable to differentiate, i. e., $\partial_t y = y_t$.

Standard Hilbert spaces The standard Hilbert space of square integrable functions on Ω is referred to as $L^2(\Omega)$ and equipped with the standard scalar product

$$(u, v)_\Omega := (u, v)_{L^2(\Omega)} := \int_\Omega u(x)\, v(x)\, dx$$

for all $u, v \in L^2(\Omega)$. The corresponding norm is denoted by $||\cdot||_\Omega = ||\cdot||_{L^2(\Omega)}$. The same notation in the scalar product and norm is used for vector valued functions $v, w \in (L^2(\Omega))^n$,

$n \in \mathbb{N}$. The standard Sobolev space for functions with weak first derivatives is referred to by $H^1(\Omega) \subset L^2(\Omega)$, and $H_0^1(\Omega) \subset H^1(\Omega)$ is the subspace with weak zero boundary conditions.

Dual spaces and operators in abstract spaces For a Hilbert space V, V^* refers to its dual and $(\cdot,\cdot)_{V^*,V}$ to the corresponding dual pairing. The (Fréchet) derivative of an operator $J \in V^*$ is denoted by $DJ(\cdot)$. If J is twice differentiable, $\nabla^2 J(\cdot)$ stands for the associated Hessian.

Spaces on the space-time cylinder The space $L^2(\mathcal{Q})$ is the space of all square-integrable functions on $\mathcal{Q}$, equipped with the scalar product

$$(u,v)_{L^2(\mathcal{Q})} := \int_0^T \int_\Omega u(t,x)v(t,x)\,dx\,dt$$

for all $u, v \in L^2(\mathcal{Q})$. The associated norm is denoted by $||\cdot||_{L^2(\mathcal{Q})}$.

Abstract functions on a space-time cylinder A convenient formulation for functions on the space-time cylinder $\mathcal{Q}$ is the concept of 'abstract functions', cf. [92, 139]. For real valued Hilbert space X, the standard space of abstract functions on a time interval $(0,T)$ with values in X is denoted by $L^2(0,T;X)$ with associated scalar product

$$(v,w)_{L^2(0,T;X)} := \int_0^T (v(t),w(t))_X dt \qquad \forall\, v,w \in L^2(0,T;X)$$

which implies the norm

$$||v||_{L^2(0,T;X)} = \left(\int_0^T (v(t),v(t))_X\,dt\right)^{1/2}, \qquad v \in L^2(0,T;X).$$

Spaces of abstract functions are closely related to functions on the space-time cylinder. It is possible to identify

$$L^2(\mathcal{Q}) \;\cong\; L^2(0,T;L^2(\Omega))$$

via an appropriate isomorphism. The scalar products and norms for both spaces are the same. In this context, a function $v \in L^2(\mathcal{Q})$ is interpreted as $v \in L^2(0,T;L^2(\Omega))$ using the identification

$$[v(t)](x) := v(t,x), \qquad t \in (0,T),\ x \in X,$$

pointwise almost everywhere. The corresponding construction can be found, e. g., in [57, 82]. For simplicity, the notation in this work does not distinguish between these spaces.

Functions differentiable in time Important spaces for this work which allow to deal with functions differentiable in time are

$$H^1(0,T;X) = \{y \in L^2(0,T;X) \,|\, y_t \in L^2(0,T;X)\}$$

and

$$H^{1,1}(\mathcal{Q}) := L^2(0,T;H^1(\Omega)) \cap H^1(0,T;L^2(\Omega)),$$

see also [139].

Multiple components in space For functions with multiple components in space, product spaces are used. An additional superscript denotes the dimension. For example, with $n \in \mathbb{N}$,

$$\begin{aligned} L^2(\Omega)^n &:= (L^2(\Omega))^n, \\ H^1(\Omega)^n &:= (H^1(\Omega))^n, \\ L^2(\mathcal{Q})^n &:= L^2(0,T;L^2(\Omega)^n), \\ H^{1,1}(\mathcal{Q})^n &:= L^2(0,T;H^1(\Omega)^n) \cap H^1(0,T;L^2(\Omega)^n). \end{aligned}$$

The associated scalar products (and norms) are appropriately defined by a sum of the scalar products of the components. For example, there is

$$(u,v)_{L^2(\mathcal{Q})^n} = \int_0^T \int_\Omega u_1 v_1 + ... + u_n v_n \, dx \, dt.$$

and the associated norm $||\cdot||_{L^2(\mathcal{Q})^n}$.

Simplified notation To simplify the notation involving L^2 scalar products and norms,

$$(u,v)_{\mathcal{Q}}$$

refers to $(u,v)_{L^2(\mathcal{Q})}$ or $(u,v)_{L^2(\mathcal{Q})^{\dim}}$, depending on the context. The associated norm is identified by

$$||\cdot||_{\mathcal{Q}}.$$

2

Problem formulation and discretisation

In this first introductory chapter, the mathematical background of (distributed) optimal control problems with PDE constraints is highlighted and appropriate discretisation techniques are presented. With the help of a set of model problems of nonstationary distributed control type, the Lagrange multiplier framework is used to demonstrate how a PDE-constrained minimisation problem can be reformulated as a set of partial differential equations, the 'KKT system'. In the following, a special discretisation recipe is applied to discretise this system in space and time. This discretisation is the core for the algorithms in Chapter 3, since it can be applied in a hierarchical way on the space-time cylinder. In a natural way, all degrees of freedom in the discrete KKT system are coupled, which reflects the fact that the KKT system exhibits a globally elliptic character — the core property that the hierarchical approach relies on.

Outline

Section 2.1 introduces a set of basic model problems which are used throughout the whole work. All of these model problems are minimisation problems with nonstationary partial differential equations as constraints. In order to reformulate these as sets of equations, Section 2.2 demonstrates how the Lagrange multiplier method has to be applied. In the following, Section 2.3 formulates the KKT systems associated with the model problems in a detailed way.

Section 2.4 forms the theoretical core of the whole methodology. This section motivates that the KKT systems are equivalent to biharmonic systems which are second order in time, fourth order in space and which exhibit an elliptic character if considered as global problems on the whole space-time cylinder. This was already proven for a variant of the heat equation and can be derived in a formal way for the Stokes and Navier–Stokes equations as well.

Finally, the KKT systems of the model problems have to be discretised. This is done in two steps, at first in time, then in space. Section 2.5 and Section 2.6 describe the time discretisation, on the one hand based on the implicit Euler timestepping scheme, on the other hand based on the Crank–Nicolson method. Finally Section 2.7 applies an additional space discretisation based on the finite element approach to derive fully discrete KKT systems.

The complete discretisation follows a special discretisation recipe which imitates the way, the continuous KKT system has been derived in Section 2.2. In the end, this leads to the commutation of the *First-Optimise-Then-Discretise* approach with the *First-Discretise-Then-Optimise* approach — a desirable property of the discretisation. Section 2.8 discusses

this topic. With this discretisation strategy at hand, Chapter 3 will then be able to formulate a hierarchical solver strategy.

2.1. Model problems

This work considers three basic model problems, namely the optimal distributed control of the heat equation, the nonstationary Stokes equations and the nonstationary Navier–Stokes equations. These problems are formally defined as follows:

Optimal control of the heat equation

For $U := L^2(\mathcal{Q})$, $V := H^{1,1}(\mathcal{Q})$, $u, z \in U$ and $y \in V$, formally consider the problem

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min! \tag{2.1}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \Delta y &= u && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma. \end{aligned}$$

The function $g : \Sigma \to \mathbb{R}$ specifies the Dirichlet boundary conditions, u denotes the control, y a 'temperature' to be computed and z a given target 'temperature' for y. Finally, $\alpha > 0$ denotes a regularisation parameter. For simplicity, no restrictions are assumed on the control u.

This problem was already analysed in [34] in a multigrid setting and is included here for reference purposes. The equation has to be interpreted in the weak form on the space-time cylinder. A possible formulation (cf. [26, 62]) needs the space

$$H_0^{1,1}(\mathcal{Q}) := L^2(0,T;H_0^1(\Omega)) \cap H^1(0,T;L^2(\Omega)).$$

The function y is a solution of the above heat equation if it has the form $y = \tilde{y} + \tilde{g}$ with a function $\tilde{g} \in V$, $\tilde{g}_{|\Sigma} = g$ and a function $\tilde{y} \in H_0^{1,1}(\mathcal{Q})$ such that

$$\begin{aligned} (\tilde{y}_t, v)_{\mathcal{Q}} + (\nabla\tilde{y}, \nabla v)_{\mathcal{Q}} &= (u - \tilde{g}_t, v)_{\mathcal{Q}} - (\nabla\tilde{g}, \nabla v)_{\mathcal{Q}} && \forall\, v \in L^2(0,T;H_0^1(\Omega)), \\ \tilde{y}(0,\cdot) &= y^0 && \text{a. e. in } \Omega. \end{aligned}$$

In particular, $y_{|\Sigma} = \tilde{y}_{|\Sigma} + \tilde{g}_{|\Sigma} = 0 + \tilde{g}_{|\Sigma} = g$ satisfies the boundary conditions. Under mild conditions (cf. [58, Chapter 7]), solutions exist and are unique.

Optimal control of the nonstationary Stokes equations

For $U := L^2(\mathcal{Q})^{\dim}$, $V := H^{1,1}(\mathcal{Q})^{\dim}$, $Z_0 := \{q \in L^2(\Omega) : \int_\Omega q\,dx = 0\}$, $Z := L^2(0,T;Z_0)$, $u, z \in U$, $y \in V$ and $p \in Z$, formally consider the problem

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min! \tag{2.2}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \nu\Delta y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma. \end{aligned}$$

This formulation is a simplification of the optimal distributed control of the Navier–Stokes equations which were considered, e. g., in [84]. The function $g : \Sigma \to \mathbb{R}^{\dim}$ specifies the Dirichlet boundary conditions, u denotes the control, y the velocity vector to be computed, p the pressure and z a given target velocity field for y. The parameter $\nu > 0$ defines the viscosity which is assumed to be constant. Furthermore, $\alpha > 0$ again denotes a constant regularisation parameter. For simplicity, no restrictions are assumed on the control u.

The above equation has to be interpreted in the weak sense on the space-time cylinder. With $H_0^{1,1}(\mathcal{Q})$ defined as above, the pair (y, p) is a solution of the above Stokes equations if y has the form $y = \tilde{y} + \tilde{g}$ with a function $\tilde{g} \in V$, $\tilde{g} = g$ on Σ and a function $\tilde{y} \in H_0^{1,1}(\mathcal{Q})^{\dim}$ such that

$$\begin{aligned}
(\tilde{y}_t, v)_{\mathcal{Q}} + (\nu \nabla \tilde{y}, \nabla v)_{\mathcal{Q}} - (p, \operatorname{div} v)_{\mathcal{Q}} & \\
= (u - \tilde{g}_t, v)_{\mathcal{Q}} - (\nu \nabla \tilde{g}, \nabla v)_{\mathcal{Q}} & \qquad \forall\, v \in L^2(0,T; H_0^1(\Omega)^{\dim}), \\
(-\operatorname{div} \tilde{y}, q)_{\mathcal{Q}} = (\operatorname{div} \tilde{g}, q)_{\mathcal{Q}} & \qquad \forall\, q \in L^2(\mathcal{Q}), \\
y(0, \cdot) = y^0 & \qquad \text{a. e. in } \Omega.
\end{aligned}$$

The above problem can be extended to the control of the Navier–Stokes equations as follows:

Optimal control of the nonstationary Navier–Stokes equations

With the same spaces and notations as in 2.), formally consider the problem

$$J(y,u) := \frac{1}{2}||y - z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min! \tag{2.3}$$

$$\begin{aligned}
\text{s.t.} \qquad y_t - \nu \Delta y + y \nabla y + \nabla p &= u && \text{in } \mathcal{Q}, \\
-\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\
y(0, \cdot) &= y^0 && \text{in } \Omega, \\
y &= g && \text{at } \Sigma,
\end{aligned}$$

which also has to be interpreted in the weak sense, similar to the Stokes equations as above. This problem was considered, e. g., in [1, 61, 84].

2.2. The *First-Optimise-Then-Discretise* calculus. Step one: The optimisation

The *First-Optimise-Then-Discretise* calculus is a common way to discretise large-scale PDE-constrained minimisation problems, cf. [92, 139]. It contains two steps:

a) In the first step, a set of optimality conditions is derived for the minimisation problem under consideration. For this purpose, the formal Lagrange multiplier approach is applied on the continuous level. An auxiliary Lagrange functional is formulated and minimised without constraints. The first order necessary optimality conditions of this Lagrange functional, also called 'Karush-Kuhn-Tucker' or 'KKT' conditions, characterise a set of critical points which are candidates for local extrema of the original minimisation problem.

b) In a second step, the KKT system (which is now a set of equations on the continuous level) is discretised and solved.

While it is a problem of its own to analyse and prove under which conditions a critical point forms a global minimum (see [157]), this work concentrates on setting up and solving the underlying KKT system.

This section focuses on step a). The formal Lagrange multiplier approach is exemplarily applied to the optimal control of the Navier–Stokes equations. The aim is to derive a compact form for the associated KKT system. This form plays a central role in the whole work. It basically contains a set of two equations, a primal and a dual equation, which are both fully coupled on the space-time cylinder. The whole discretisation and solver concept is based on this form.

In Section 2.5 and the following sections, a special discretisation recipe is applied to discretise the KKT system in space and time. This realises step b). The operators used in the recipe are constructed as discrete counterparts to the operators in the continuous KKT system. Later, Chapter 3 will create a multilevel based solver strategy based on this discretisation scheme.

Simplified optimal control of the Navier–Stokes equations All following notations and formal transformations have to be interpreted in the weak sense. Consider the optimal distributed control of the Navier–Stokes equations (2.3) on page 23. For simplicity, the initial and boundary conditions are assumed to be homogeneous, i. e., $y^0 = 0$ and $g = 0$. Even more, boundary and initial conditions are treated as explicit constraints and are not included into the Lagrange functional; for a proper derivation of the KKT system, this would have to be done, but for the pure motivation of the discretisation recipe used in later chapters, these quantities are neglected for the moment. With these settings, equation (2.3) reads

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min! \tag{2.4}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \nu\Delta y + y\nabla y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= 0 && \text{in } \Omega, \\ y &= 0 && \text{at } \Sigma, \end{aligned}$$

with $U = L^2(\mathcal{Q})$, $V = H^{1,1}(\mathcal{Q})^{\dim}$, $Z_0 = \{q \in L^2(\Omega) : \int_\Omega q = 0\}$, $Z := L^2(0,T;Z_0)$, $y \in V$, $u \in U$ and $p \in Z$. To simplify the notation, the spaces $X := V \times Z$, the variables $x := (y,p) \in X$, $\bar{x} := (\bar{y},\bar{p}) \in X$ and the operators

$$\begin{aligned} \tilde{J}(x,u) &:= J(y,u), \\ \mathcal{H}(x)\bar{x} &:= \begin{pmatrix} \bar{y}_t - \nu\Delta\bar{y} + y\nabla\bar{y} + \nabla\bar{p}, \\ -\operatorname{div}\bar{y} \end{pmatrix} \end{aligned}$$

are introduced. With this notation, the optimal control problem can be written in the form

$$\tilde{J}(x,u) \quad \longrightarrow \quad \min! \tag{2.5}$$

$$\begin{aligned} \text{s.t.} \qquad \mathcal{H}(x)x &= \begin{pmatrix} u \\ 0 \end{pmatrix} && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= 0 && \text{in } \Omega, \\ y &= 0 && \text{at } \Sigma. \end{aligned}$$

The formal Lagrange multiplier approach The next step applies the formal Lagrange multiplier technique to the above equation. A description of this approach can be found, e. g., in [92, 139], see also [11]. For a dual variable $\psi := (\lambda, \xi) \in X$, the Lagrange functional

$$L(x, u, \psi) := J(x, u) + \left(\begin{pmatrix} u \\ 0 \end{pmatrix} - \mathcal{H}(x)x, \psi \right)_{\mathcal{Q}}$$

is introduced; initial/boundary conditions are neglected for simplicity. The scalar product is the natural scalar product over $\mathcal{Q}$, i. e., for all $v = (v_y, v_p) \in X$ and $w = (w_y, w_p) \in X$, there is

$$(v, w)_{\mathcal{Q}} = (v_y, w_y)_{\mathcal{Q}} + (v_p, w_p)_{\mathcal{Q}}.$$

A critical point of $L(\cdot)$ is a candidate for a minimiser of (2.4). The critical points are characterised by

$$DL(x, u, \psi)(\bar{x}, \bar{u}, \bar{\psi}) = 0 \tag{2.6}$$

for all $(\bar{x}, \bar{u}, \bar{\psi})$ with $\bar{x} := (\bar{y}, \bar{p}) \in X$, $\bar{\psi} := (\bar{\lambda}, \bar{\xi}) \in X$ and $\bar{u} \in U$. With $DL(\cdot)$, the Fréchet derivative of $L(\cdot)$ is denoted. Equation (2.6) reads in detail

$$\begin{aligned} 0 = \quad & \left(\begin{pmatrix} y - z \\ 0 \end{pmatrix}, \bar{x} \right)_{\mathcal{Q}} && + (\alpha u, \bar{u})_{\mathcal{Q}} \\ & + \left(\begin{pmatrix} u \\ 0 \end{pmatrix} - \mathcal{H}(x)x, \bar{\psi} \right)_{\mathcal{Q}} && + (\psi, D\mathcal{H}\bar{x})_{\mathcal{Q}} \\ & + \left(\begin{pmatrix} u \\ 0 \end{pmatrix}, \bar{\psi} \right)_{\mathcal{Q}} && + \left(\psi, \begin{pmatrix} \bar{u} \\ 0 \end{pmatrix} \right)_{\mathcal{Q}} \end{aligned} \tag{2.7}$$

with $D\mathcal{H} = D\mathcal{H}(x)$ being the Fréchet derivative of $x \mapsto \mathcal{H}(x)x$, i. e.,

$$D\mathcal{H}(x)\bar{x} = \begin{pmatrix} \bar{y}_t - \nu\Delta\bar{y} + (\bar{y}\nabla y + y\nabla\bar{y}) + \nabla\bar{p} \\ -\operatorname{div}\bar{y} \end{pmatrix}. \tag{2.8}$$

Equation (2.7) leads to the so called 'Karush-Kuhn-Tucker' or 'KKT' system

$$\left.\begin{aligned} \mathcal{H}(x)x &= \begin{pmatrix} u \\ 0 \end{pmatrix} \\ D\mathcal{H}^*(x)\psi &= \begin{pmatrix} y - z \\ 0 \end{pmatrix} \\ u &= -\frac{1}{\alpha}\lambda. \end{aligned}\right\} \tag{2.9}$$

The last equation can be eliminated by plugging it into the first one. The result is the KKT system in abstract form with two equations, which is formulated in the next corollary. For an even more compact notation,

$$\mathcal{B}\psi := \begin{pmatrix} -\frac{1}{\alpha}\lambda \\ 0 \end{pmatrix} \qquad \text{and} \qquad \tilde{\mathcal{B}}x := \begin{pmatrix} y - z \\ 0 \end{pmatrix}$$

are used.

2.1 Corollary. With the above notations, the KKT system associated with the minimisation problem (2.4) reads

$$\mathcal{H}(x)x = \mathcal{B}\psi, \tag{2.10a}$$

$$D\mathcal{H}^*(x)\psi = \tilde{\mathcal{B}}x. \tag{2.10b}$$

The operator $D\mathcal{H}^* = D\mathcal{H}^*(x)$ is the adjoint operator of the Fréchet derivative $D\mathcal{H} = D\mathcal{H}(x)$ of the Navier–Stokes operator $\mathcal{H}(x)x$, i. e., for all $v, w \in X$ there is

$$(v, D\mathcal{H}w)_{\mathcal{Q}} = (D\mathcal{H}^*v, w)_{\mathcal{Q}}. \tag{2.11}$$

Without loss of generality, (2.10) can be assumed to include the initial/end time/boundary conditions, they can be added via additional terms in the Lagrange functional.

2.3. The KKT systems in detail

Equation (2.10) denotes in a general way the first order necessary optimality conditions and holds in a similar way for all minimisation problems to which the Lagrange multiplier approach is applied. Of course, the above formulation still lacks a proper definition of the initial/boundary conditions in the primal and dual equation; these can be obtained by including the initial/boundary conditions of the primal equation into the Lagrange functional as well. Applying the whole calculus, the following detailed KKT systems are obtained, which all have to be interpreted in the weak sense on the space-time cylinder (cf. [84, 139]):

Heat equation

With $U = L^2(\mathcal{Q})$ and $V = H^{1,1}(\mathcal{Q})$, there are $y, \lambda \in V$ and $u \in U$ to be found such that for $z \in U$ there is

$$\left.\begin{aligned} y_t - \Delta y &= u && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma, \end{aligned}\right\} \tag{2.12a}$$

$$\left.\begin{aligned} -\lambda_t - \Delta\lambda &= y - z && \text{in } \mathcal{Q}, \\ \lambda(T,\cdot) &= 0 && \text{in } \Omega, \\ \lambda &= 0 && \text{at } \Sigma, \end{aligned}\right\} \tag{2.12b}$$

$$u = -\frac{1}{\alpha}\lambda \quad \text{in } \mathcal{Q}. \tag{2.12c}$$

In this set of equations, λ denotes the 'dual temperature'. The control u can be eliminated, which leads to the coupled system

$$y_t - \Delta y = -\frac{1}{\alpha}\lambda, \tag{2.13a}$$

$$-\lambda_t - \Delta\lambda = y - z, \tag{2.13b}$$

complemented by the above initial/end time/boundary conditions. (2.13a) is called the *primal* and (2.13b) the *dual* equation.

Stokes equations

With $U = L^2(\mathcal{Q})^{\dim}$, $V = H^{1,1}(\mathcal{Q})^{\dim}$, $Z_0 = \{q \in L^2(\Omega) : \int_\Omega q\,dx = 0\}$, $Z := L^2(0,T;Z_0)$ there are $y, \lambda \in V$, $p, \xi \in Z$, $u \in U$ to be found such that for $z \in U$ there is

$$\left.\begin{aligned} y_t - \nu\Delta y + \nabla p &= u && \text{in } \mathcal{Q},\\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q},\\ y(0,\cdot) &= y^0 && \text{in } \Omega,\\ y &= g && \text{at } \Sigma, \end{aligned}\right\} \tag{2.14a}$$

$$\left.\begin{aligned} -\lambda_t - \nu\Delta\lambda + \nabla\xi &= y - z && \text{in } \mathcal{Q},\\ -\operatorname{div}\lambda &= 0 && \text{in } \mathcal{Q},\\ \lambda(T,\cdot) &= 0 && \text{in } \Omega,\\ \lambda &= 0 && \text{at } \Sigma, \end{aligned}\right\} \tag{2.14b}$$

$$u = -\frac{1}{\alpha}\lambda \qquad \text{in } \mathcal{Q}. \tag{2.14c}$$

Herein, λ denotes the 'dual velocity' and ξ the 'dual pressure'. Eliminating u, the reduced nonlinear KKT system reads

$$\begin{aligned} y_t - \nu\Delta y + \nabla p &= -\frac{1}{\alpha}\lambda, \\ -\operatorname{div} y &= 0, \end{aligned} \tag{2.15a}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda + \nabla\xi &= y - z, \\ -\operatorname{div}\lambda &= 0, \end{aligned} \tag{2.15b}$$

complemented by the above initial/end time/boundary conditions. (2.15a) is called the *primal* and (2.15b) the *dual* equation.

Navier–Stokes equations

With the same spaces as in the case of the Stokes equations, there are $y, \lambda \in V$, $p, \xi \in Z$, $u \in U$ to be found such that for $z \in U$ there is

$$\left.\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p &= u && \text{in } \mathcal{Q},\\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q},\\ y(0,\cdot) &= y^0 && \text{in } \Omega,\\ y &= g && \text{at } \Sigma, \end{aligned}\right\} \tag{2.16a}$$

$$\left.\begin{aligned} -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi &= y - z && \text{in } \mathcal{Q},\\ -\operatorname{div}\lambda &= 0 && \text{in } \mathcal{Q},\\ \lambda(T,\cdot) &= 0 && \text{in } \Omega,\\ \lambda &= 0 && \text{at } \Sigma, \end{aligned}\right\} \tag{2.16b}$$

$$u = -\frac{1}{\alpha}\lambda \qquad \text{in } \mathcal{Q}. \tag{2.16c}$$

Elimination of u leads the reduced KKT system

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p &= -\frac{1}{\alpha}\lambda, \\ -\operatorname{div} y &= 0, \end{aligned} \tag{2.17a}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi &= y - z, \\ -\operatorname{div}\lambda &= 0, \end{aligned} \tag{2.17b}$$

complemented by the above initial/end time/boundary conditions. For the exact analytical setting of these equations including a discussion about existence and uniqueness, see [92, 139, 157].

2.4. Ellipticity of the KKT system

Consider for the moment a modified optimal control problem for the heat equation, which reads

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min!$$

$$\left.\begin{aligned} \text{s.t.} \qquad y_t - \Delta y + cy &= u \quad \text{in } \mathcal{Q}, \\ \partial_\eta y &= g \quad \text{at } \Sigma, \\ y(0,\cdot) &= y^0 \quad \text{in } \Omega, \end{aligned}\right\} \tag{2.18}$$

with constants α, $c > 0$ and functions y, u, $z : \mathcal{Q} \to \mathbb{R}$. Here, $\eta : \Gamma \to \mathbb{R}^2$ denotes the outer unit normal vector of the domain $\Omega \subset \mathbb{R}^2$ and $\mathcal{Q} = (0,T) \times \Omega$ for $T > 0$. The first order necessary optimality conditions for this problem are given by

$$y_t - \Delta y + cy = u \qquad \text{in } \mathcal{Q}, \tag{2.19a}$$
$$-\lambda_t - \Delta\lambda + c\lambda = y - z \qquad \text{in } \mathcal{Q}, \tag{2.19b}$$
$$\partial_\eta y = g \qquad \text{at } \Sigma, \tag{2.19c}$$
$$\partial_\eta \lambda = 0 \qquad \text{at } \Sigma, \tag{2.19d}$$
$$y(0,\cdot) = y^0 \qquad \text{in } \Omega, \tag{2.19e}$$
$$\lambda(T,\cdot) = 0 \qquad \text{in } \Omega, \tag{2.19f}$$
$$\alpha u + \lambda = 0 \qquad \text{in } \mathcal{Q}, \tag{2.19g}$$

which have to be understood in the weak sense on the space-time cylinder $\mathcal{Q}$.

It is shown in [121] that under suitable conditions, this problem is equivalent to the following mixed harmonic (in time) / biharmonic (in space) problem

$$\begin{aligned} -\lambda_{tt} + \Delta^2\lambda - 2c\Delta\lambda + (c^2 + \frac{1}{\alpha})\lambda &= f \qquad \text{in } \mathcal{Q}, \\ \partial_\eta(\Delta\lambda) &= 0 \qquad \text{at } \Sigma, \\ \partial_\eta\lambda &= 0 \qquad \text{at } \Sigma, \\ -\lambda_t - \Delta\lambda + c\lambda &= 0 \qquad \text{at } \{0\} \times \Omega, \\ \lambda(T,\cdot) &= 0 \qquad \text{in } \Omega \end{aligned}$$

in the weak sense, for a properly defined $f : \mathcal{Q} \to \mathbb{R}^2$.

For a more precise interpretation of this statement, in [121], Neitzel et al. define the spaces

$$H^{2,1}(\mathcal{Q}) = L^2(0,T;H^2(\Omega)) \cap H^1(0,T;L^2(\Omega)), \tag{2.20}$$

$$V := \bar{H}^{2,1}(\mathcal{Q}) = \{y \in H^{2,1}(\mathcal{Q})) : \partial_\eta y = 0 \text{ on } \Sigma,\ y(T,\cdot) = 0 \text{ in } \Omega\}, \tag{2.21}$$

and the bilinear form $a_\alpha(\cdot,\cdot)$,

$$\begin{aligned} a_\alpha(v,w) = {} & (v_t, w_t) + (\Delta v, \Delta w) - (\Delta v, w_t) + (v_t, \Delta w) \\ & + 2c(\nabla v, \nabla w) + (c^2 + \frac{1}{\alpha})(v,w) \\ & + c(v(0,\cdot), w(0,\cdot))_{L^2(\Omega)} \end{aligned} \tag{2.22}$$

with $(\cdot,\cdot) = (\cdot,\cdot)_{L^2(\mathcal{Q})}$ for $v, w \in H^{2,1}(\mathcal{Q})$. The dual variable $\lambda \in V$ is the solution of a linear equation

$$a_\alpha(\lambda, w) = (f, w) \qquad \forall\, w \in V,$$

for a properly defined $f \in L^2(\mathcal{Q})$. The control u and the solution y can be calculated from λ in a postprocessing step using (2.18) and (2.19g). According to [121], the bilinear form $a_\alpha(\cdot,\cdot)$ is V-elliptic, i. e., there is a constant $\kappa > 0$ such that

$$a_\alpha(v,v) \geq \kappa ||v||^2_{H^{2,1}(\mathcal{Q})}$$

for all $v \in V$. Thus, the KKT system is equivalent to a V-elliptic problem on the space-time cylinder $\mathcal{Q}$.

Deriving the biharmonic operator in a formal way

Applying the same techniques as in [34], the mixed harmonic/biharmonic problem can also be derived in a formal way: Solving the dual equation

$$-\lambda_t - \Delta\lambda + c\lambda = y - z$$

for y and applying ∂_t leads to

$$y = -\lambda_t - \Delta\lambda + c\lambda + z, \tag{2.23}$$

$$y_t = -\lambda_{tt} - \Delta\lambda_t + c\lambda_t + z_t. \tag{2.24}$$

Inserting (2.23) and (2.24) into the primal equation

$$y_t - \Delta y + cy = -\frac{1}{\alpha}\lambda$$

and cancelling out all redundant terms, the mixed harmonic/biharmonic equation

$$-\lambda_{tt} + \Delta^2\lambda - 2c\Delta\lambda + (c^2 + \frac{1}{\alpha})\lambda = f$$

in λ is obtained, with $f := -(z_t - \Delta z + cz)$.

The Stokes equations The same technique can also be used for the Stokes equations. Setting formally

$$y,\, \lambda \in \{g : \mathcal{Q} \to \mathbb{R}^2 :\ \operatorname{div} g = 0\},$$

the steps in a) can be applied to the KKT system for the optimal distributed control of the Stokes equations,

$$\begin{aligned} y_t - \nu\Delta y &= -\frac{1}{\alpha}\lambda \qquad \text{in } \mathcal{Q} \\ -\lambda_t - \nu\Delta\lambda &= y - z \qquad \text{in } \mathcal{Q}, \end{aligned}$$

which formally leads to the system

$$-\lambda_{tt} + \nu^2\Delta^2\lambda + \frac{1}{\alpha}\lambda = f$$

in λ, with $f := -(z_t - \Delta z)$. This has the form of the mixed harmonic/biharmonic problem in a), so an elliptic character can be expected here.[1]

The Navier–Stokes equations In case of the Navier–Stokes equations, the KKT system allows to formally derive

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y &= -\frac{1}{\alpha}\lambda, \\ -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda - y &= -z, \qquad (2.25) \end{aligned}$$

$$\Rightarrow \qquad \lambda = -\alpha(y_t - \nu\Delta y + y\nabla y), \qquad (2.26)$$

$$\Rightarrow \qquad \lambda_t = -\alpha(y_{tt} - \nu\Delta y_t + y_t\nabla y + y\nabla y_t). \qquad (2.27)$$

Inserting (2.26) and (2.27) into (2.25) and cancelling out redundant terms leads to a mixed harmonic/biharmonic system for the primal variables of the form

$$-y_{tt} + \nu^2\Delta^2 y + \gamma(y_t, y) + y = z$$

for some function $\gamma(y_t, y)$ that contains at most third derivatives of y in space. If ν is large enough, the system can be seen as a disturbed Stokes problem, so an elliptic character similar to the case of the Stokes equations can be expected. Similar arguments also indicate ellipticity for the Fréchet derivative of the KKT system.

2.2 Remark. The elliptic character of the KKT system and its Fréchet derivative on the space-time cylinder is the key point for this work. The following chapters derive a solver for the KKT system that acts on the whole space-time cylinder. The solver itself is basically a Newton solver which in each iteration has to solve a linear subproblem on the space-time cylinder that represents the Fréchet derivative of the KKT system. The idea is now to apply a multigrid based algorithm for these linear subproblems. Multigrid based algorithms are known to show level-independent convergence rates for problems that have an elliptic character (cf. [72, 75]) and should therefore be able to solve the linear subproblems with linear complexity. Finally, an outer Newton method which solves for a possible nonlinearity is expected to converge independent of the refinement as well.

Before the solver methodology can be presented, a proper discretisation scheme must be applied to the minimisation problem. This is one part in the second step in the *First-Optimise-Then-Discretise* calculus. The KKT system is semi-discretised in time with a proper time discretisation scheme in a way that *imitates* the derivation of the KKT system in Section 2.2. In a last step, the system is discretised in space using finite elements.

[1] In this context, *ellipticity* refers to *V-ellipticity* for an appropriate space V on the space-time cylinder $\mathcal{Q}$; the term 'V' is omitted for convenience.

2.5. Time discretisation with the implicit Euler scheme

For stability and efficiency reasons (cf. [142]), the time discretisation of the KKT system is carried out with implicit time stepping techniques. The largest admissible timestep size for such schemes is not limited by stability constraints, and thus, it can be chosen only based on accuracy requirements. This is preferable, as the user might want to increase the accuracy of the spatial discretisation without having to increase the number of time intervals due to some kind of CFL condition.

Example The main aim for the algorithm presented in this work is that the algorithm should have *linear complexity*, i. e., the numerical complexity should be proportional to the number of unknowns on the space-time cylinder. If for example the accuracy of a computed solution induces the spatial mesh to be globally refined, an increase in memory and CPU time by a factor of four on a 2D mesh can be expected, as the number of degrees of freedom #dof grows by a factor of four. A CFL condition due to a non-implicit timestepping would drive the time mesh to be globally refined as well, which would finally lead to an increase of #dof, CPU time and memory by at least a factor of eight. Thus from the perspective of the end user, the algorithm would take twice as much CPU time as expected, which is undesirable.

The following paragraphs primarily focus on the standard first order implicit Euler scheme. Later, the concept is generalised to the Crank–Nicolson or general θ-scheme, respectively. All steps are directly applied to the Navier–Stokes equations; for the heat equation, only the final results are mentioned.

2.5.1. The Navier–Stokes equations

The derivation of a proper time discretisation scheme starts with the time discretisation of the primal equation (2.17a) on page 28,

$$\begin{aligned} \frac{y_n - y_{n-1}}{k} - \nu\Delta y_n + y_n\nabla y_n + \nabla p_n, &= -\frac{1}{\alpha}\lambda_n, \\ -\operatorname{div} y_n &= 0, \\ y_0 &= y^0, \end{aligned} \tag{2.28}$$

with $y_n, \lambda_n \in V$ and $p_n \in Z$ for $n = 1, ..., N$ with $N \in \mathbb{N}$ and $k = 1/N$. Next, the discretisation recipe from [11] is applied to (2.28) to derive a discrete couterpart to (2.17b) on page 28. The involved operators $\mathcal{I}$, $\mathcal{G}$, $\mathcal{D}$ and $\mathcal{C}_n$ are defined for arbitrary velocity vectors v and pressure functions q in space as follows:

$$\begin{aligned} \mathcal{I} : v \mapsto v, \qquad & \mathcal{G} : q \mapsto \nabla q, \\ \mathcal{C}_n = \mathcal{C}(y_n) : v \mapsto -\nu\Delta v + (y_n\nabla)v, \qquad & \mathcal{D} : v \mapsto -\operatorname{div} v. \end{aligned}$$

The initial solution $y^0 \in V$ is not necessarily solenoidal. The initial condition $y_0 = y^0$ is therefore realised by the following solenoidal projection,

$$\begin{aligned} \frac{1}{k}y_0 - \nu\Delta y_0 + y_0\nabla y_0 + \nabla p_0 &= \frac{1}{k}y^0 - \nu\Delta y^0 + y^0\nabla y^0, \\ -\operatorname{div} y_0 &= 0. \end{aligned}$$

This projection uses the same operations as the implicit Euler scheme and allows a more consistent notation. With $X_k := (V \times Z)^{N+1}$, using $x := (y_0, p_0, y_1, p_1, ..., y_N, p_N) \in X_k$,

this yields the nonlinear system of the primal equation,

$$\begin{aligned}\mathcal{H}^k x &:= \mathcal{H}^k(x)x \\ &= \left(\begin{array}{c|c|c|c} \frac{\mathcal{I}}{k}+\mathcal{C}_0 \quad \mathcal{G} & & & \\ \mathcal{D} & & & \\ \hline -\frac{\mathcal{I}}{k} & \frac{\mathcal{I}}{k}+\mathcal{C}_1 \quad \mathcal{G} & & \\ & \mathcal{D} & & \\ \hline & \ddots & \ddots \quad \ddots & \\ \hline & & -\frac{\mathcal{I}}{k} & \frac{\mathcal{I}}{k}+\mathcal{C}_N \quad \mathcal{G} \\ & & & \mathcal{D} \end{array}\right)\left(\begin{array}{c} y_0 \\ p_0 \\ y_1 \\ p_1 \\ \vdots \\ y_N \\ p_N \end{array}\right) \\ &= \left((\frac{\mathcal{I}}{k}+\mathcal{C}_0)y^0, 0, -\frac{\lambda_1}{\alpha}, 0, ..., -\frac{\lambda_N}{\alpha}, 0,\right)^{\mathsf{T}}\end{aligned}$$

which is equivalent to (2.28) if y^0 is solenoidal. This equation is the discrete counterpart to the primal equation (2.10a) on page 25 that appears during the derivation of the KKT system with the formal Lagrange principle.

In the second step, a discrete counterpart to the corresponding dual equation (2.10b) on page 25 has to be derived. Similar to the continuous case, the starting point is the Fréchet derivative $D\mathcal{H}$ of the Navier–Stokes equations, see (2.8) on page 25. The operator $D\mathcal{H}$ is discretised in time by the same timestepping scheme as the operator $\mathcal{H}$, so as to obtain the discrete Fréchet derivative $D\mathcal{H}^k(x)$ of the mapping $x \mapsto \mathcal{H}^k(x)x$. In particular, for a vector $\bar{x} := (\bar{y}_0, \bar{p}_0, \bar{y}_1, \bar{p}_1, ..., \bar{y}_N, \bar{p}_N) \in X_k$ the resulting scheme can be expressed in a matrix-vector notation over all timesteps,

$$\begin{aligned}D\mathcal{H}^k\bar{x} &:= D\mathcal{H}^k(x)\bar{x} \\ &= \left(\begin{array}{c|c|c|c} \frac{\mathcal{I}}{k}+\mathcal{N}_0 \quad \mathcal{G} & & & \\ \mathcal{D} & & & \\ \hline -\frac{\mathcal{I}}{k} & \frac{\mathcal{I}}{k}+\mathcal{N}_1 \quad \mathcal{G} & & \\ & \mathcal{D} & & \\ \hline & \ddots & \ddots \quad \ddots & \\ \hline & & -\frac{\mathcal{I}}{k} & \frac{\mathcal{I}}{k}+\mathcal{N}_N \quad \mathcal{G} \\ & & & \mathcal{D} \end{array}\right)\left(\begin{array}{c} \bar{y}_0 \\ \bar{p}_0 \\ \bar{y}_1 \\ \bar{p}_1 \\ \vdots \\ \bar{y}_N \\ \bar{p}_N \end{array}\right),\end{aligned}$$

with the additional operator

$$\mathcal{N}_n := \mathcal{N}(y_n) : v \mapsto -\nu\Delta v + (y_n\nabla)v + (v\nabla)y_n.$$

Corresponding to (2.11) on page 26, the time discretisation of the dual equation is defined as the adjoint $D\mathcal{H}^{k,*}$ of $D\mathcal{H}^k$,

$$(\psi, D\mathcal{H}^k\bar{x}) = (D\mathcal{H}^{k,*}\psi, \bar{x}),$$

whereby $\psi := (\lambda_0, \xi_0, \lambda_1, \xi_1, ..., \lambda_N, \xi_N) \in X_k$. Exploiting $\mathcal{D}^* = \mathcal{G}$ and the adjoint of $\mathcal{N}_n$ being given by

$$\mathcal{N}_n^* = \mathcal{N}^*(y_n) : v \mapsto -\nu\Delta v - (y_n\nabla)v + (\nabla y_n)^{\mathsf{T}}v$$

for all velocity vectors $v \in V$, this yields

$$
\begin{aligned}
D\mathcal{H}^{k,*}\psi &= D\mathcal{H}^{k,*}(x)\psi \\
&= \left(\begin{array}{c|c|c|c}
\frac{\mathcal{I}}{k} + \mathcal{N}_0^* \quad \mathcal{G} & -\frac{\mathcal{I}}{k} & & \\
\mathcal{D} & & & \\
\hline
 & \frac{\mathcal{I}}{k} + \mathcal{N}_1^* \quad \mathcal{G} & -\frac{\mathcal{I}}{k} & \\
 & \mathcal{D} & & \\
\hline
 & & \ddots \quad \ddots & \ddots \\
\hline
 & & & \frac{\mathcal{I}}{k} + \mathcal{N}_N^* \quad \mathcal{G} \\
 & & & \mathcal{D}
\end{array}\right)
\begin{pmatrix} \lambda_0 \\ \xi_0 \\ \lambda_1 \\ \xi_1 \\ \vdots \\ \lambda_N \\ \xi_N \end{pmatrix} \\
&= (0, 0,\ y_1 - z_1, 0,\ ..., y_N - z_N, 0)^{\mathsf{T}}.
\end{aligned}
$$

The right-hand side is chosen in such a way that the *First-Optimise-Then-Discretise* approach used here commutes with the *First-Discretise-Then-Optimise* approach, see below. Correspondingly, the time discretisation scheme of the dual equation (2.17b) on page 28 reads

$$\frac{\lambda_n - \lambda_{n+1}}{k} - \nu\Delta\lambda_n - y_n\nabla\lambda_n + (\nabla y_n)^{\mathsf{T}}\lambda_n + \nabla\xi_n = y_n - z_n, \tag{2.29a}$$

$$-\operatorname{div}\lambda_n = 0, \tag{2.29b}$$

$$\frac{\lambda_N}{k} - \nu\Delta\lambda_N - y_N\nabla\lambda_N + (\nabla y_N)^{\mathsf{T}}\lambda_N + \nabla\xi_N = y_N - z_N, \tag{2.29c}$$

where (2.29c) approaches the actual end time condition $\lambda(T,\cdot) = 0$ for $k \to 0$. With $w_n := (y_n, \lambda_n, p_n, \xi_n)$, the primal and dual solutions are combined in the vector

$$
\begin{aligned}
w &:= \big(\underbrace{y_0, \lambda_0, p_0, \xi_0}_{w_0}, \underbrace{y_1, \lambda_1, p_1, \xi_1}_{w_1}, \underbrace{y_2, \lambda_2, p_2, \xi_2}_{w_2}, \ ...\big)^{\mathsf{T}} \\
&\in (V \times V \times Z \times Z)^{N+1}.
\end{aligned}
$$

Shifting the terms with λ_n and y_n in (2.28) and (2.29) from the right-hand side to the left-hand side and mixing the two matrices stemming from $\mathcal{H}^k$ and $D\mathcal{H}^{k,*}$ yields the semi-discrete system

$$\mathbf{G}(w)w = f. \tag{2.30}$$

The right-hand side is given by

$$f = \big(\underbrace{(\mathcal{I}/k + \mathcal{C}_0)y^0, 0, 0, 0}_{f_0}, \underbrace{0, -z_1, 0, 0}_{f_1}, ..., \underbrace{0, -z_{N-1}, 0, 0}_{f_{N-1}}, \underbrace{0, -z_N, 0, 0}_{f_N}\big)^{\mathsf{T}}$$

and the operator on the left-hand side reads

$$\mathbf{G} = \mathbf{G}(w) = \left(\begin{array}{c|c|c|c|c}
\mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\
\hline
\breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\
\hline
 & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\
\hline
 & & \ddots & \ddots & \ddots \\
\hline
 & & & \breve{\mathbf{I}}_N & \mathbf{G}_N
\end{array}\right) \tag{2.31}$$

with

$$\mathbf{G}_0 = \begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{C}_0 & 0 & \mathcal{G} & 0 \\ 0 & \frac{\mathcal{I}}{k}+\mathcal{N}_0^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}, \qquad \mathbf{G}_n = \begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{C}_n & \frac{\mathcal{I}}{\alpha} & \mathcal{G} & 0 \\ -\mathcal{I} & \frac{\mathcal{I}}{k}+\mathcal{N}_n^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

$$\breve{\mathbf{I}}_n = \begin{pmatrix} -\frac{\mathcal{I}}{k} & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}, \qquad \hat{\mathbf{I}}_{n-1} = \begin{pmatrix} 0 & & & \\ & -\frac{\mathcal{I}}{k} & & \\ & & 0 & \\ & & & 0 \end{pmatrix},$$

for $n = 1, ..., N$.

2.5.2. The Newton system associated with (2.30)

With the semi-discrete system at hand, a Newton algorithm in space and time can be formulated. Written in defect correction form, the corresponding iteration reads

$$w_{i+1} := w_i + \mathbf{F}(w_i)^{-1}(f - \mathbf{G}(w_i)w_i), \qquad i \in \mathbb{N} \tag{2.32}$$

starting with an initial iterate $w_0 \in (V \times V \times Z \times Z)^{N+1}$. Here, $\mathbf{F}(w)$ is the Fréchet derivative of the mapping $w \mapsto \mathbf{G}(w)w$ which is given by the Newton matrix

$$\mathbf{F}(w) = \left(\begin{array}{c|c|c|c|c} \mathbf{F}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{F}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{F}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{F}_N \end{array}\right), \tag{2.33}$$

$$\mathbf{F}_0 = \begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{N}_0 & 0 & \mathcal{G} & 0 \\ \mathcal{R}_0 & \frac{\mathcal{I}}{k}+\mathcal{N}_0^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}, \ \mathbf{F}_n = \begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{N}_n & \frac{1}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -\mathcal{I}+\mathcal{R}_n & \frac{\mathcal{I}}{k}+\mathcal{N}_n^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}$$

for $n = 1, ..., N$. The additional operator $\mathcal{R}_n$ is defined as

$$\mathcal{R}_n := \mathcal{R}(\lambda_n) : v \mapsto -(v\nabla)\lambda_n + (\nabla v)^{\mathsf{T}}\lambda_n \qquad \text{for all } v \in V.$$

2.5.3. The heat equation

The presented derivation of the semi-discrete KKT system carries over to the heat equation (2.13) on page 26. Using $\mathcal{A} : V \to V^*$, $\mathcal{A} : v \mapsto -\Delta v$ for all $v \in V$, applying the above discretisation recipe leads to

$$\mathbf{G}w = f \tag{2.34}$$

with

$$f = \big(\underbrace{(\mathcal{I}/k + \mathcal{A})y^0, 0}_{f_0}, \underbrace{0, -z_1}_{f_1}, ..., \underbrace{0, -z_{N-1}}_{f_{N-1}}, \underbrace{0, -z_N}_{f_N}\big)^{\mathsf{T}},$$
$$w = (y_0, \lambda_0, y_0, \lambda_1, ..., y_N, \lambda_N)^{\mathsf{T}},$$

$$\mathbf{G} = \left(\begin{array}{c|c|c|c|c} \mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{G}_N \end{array}\right) \tag{2.35}$$

and the matrices

$$\mathbf{G}_0 = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & 0 \\ 0 & \frac{\mathcal{I}}{k} + \mathcal{A} \end{pmatrix}, \qquad \mathbf{G}_n = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & \frac{\mathcal{I}}{\alpha} \\ -\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A} \end{pmatrix},$$
$$\breve{\mathbf{I}}_n = \begin{pmatrix} -\frac{\mathcal{I}}{k} & 0 \\ 0 & 0 \end{pmatrix}, \qquad \hat{\mathbf{I}}_{n-1} = \begin{pmatrix} 0 & 0 \\ 0 & -\frac{\mathcal{I}}{k} \end{pmatrix},$$

for $n = 1, ..., N$. This corresponds to the (time-)discrete primal-dual system

$$\frac{y_n - y_{n-1}}{k} - \Delta y_n = -\frac{1}{\alpha}\lambda_n, \tag{2.36a}$$
$$\frac{\lambda_n - \lambda_{n+1}}{k} - \Delta \lambda_n = y_n - z_n, \tag{2.36b}$$
$$\frac{y_0}{k} - \Delta y_0 = \frac{y^0}{k} - \Delta y^0 \qquad \text{approximating } y_0 = y^0, \tag{2.36c}$$
$$\frac{\lambda_N}{k} - \Delta \lambda_N = y_N - z_N \qquad \text{approximating } \lambda_N = 0, \tag{2.36d}$$

accompanied by the boundary conditions.

An equivalent *First-Discretise-Then-Optimise* strategy The semi-discrete scheme (2.34) (including the right-hand side) can also be derived using a special *First-Discretise-Then-Optimise* discretisation strategy in time that maintains the structure of the optimisation problem. For brevity, this technique is illustrated for the optimal distributed control of the heat equation (2.1) on page 22,

$$J(y,u) := \frac{1}{2}||y - z||^2_{\mathcal{Q}} + \frac{\alpha}{2}||u||^2_{\mathcal{Q}} \quad \longrightarrow \quad \min! \tag{2.37}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \Delta y &= u && \text{in } Q, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma. \end{aligned}$$

The variable $y := (y_0, y_1, ..., y_N) \in V^{N+1}$ denotes the set of velocity vectors and $u := (u_0, u_1, ..., u_N) \in U^{N+1}$ the set of controls, $y_n = y(t_n)$, $u_n = u(t_n)$ for all n. The choice of the time discretisation for the norm $||\cdot||_{\mathcal{Q}}$ in the primal and dual space is crucial for the form of the resulting discrete KKT system. In particular, choosing the summed rectangular rule yields (including a projection in the first timestep which ensures, e. g., that boundary

conditions at $t = 0$ are satisfied)

$$J_{\mathrm{IE}}(y,u) := \frac{1}{2}k\sum_{n=1}^{N}||y_n - z_n||_\Omega^2 + \frac{\alpha}{2}k\sum_{n=1}^{N}||u_n||_\Omega^2 \quad \longrightarrow \quad \min! \tag{2.38}$$

$$\begin{aligned} \text{s.t.} \qquad (y_n - y_{n-1}) - k\Delta y_n &= ku_n && \text{in } \Omega,\ n = 1,...,N, \\ y_0 - k\Delta y_0 &= y^0 - k\Delta y^0 && \text{in } \Omega, \\ y_n &= g(t_n,\cdot) && \text{at } \Gamma,\ n = 0,...,N, \end{aligned}$$

where $J_{\mathrm{IE}}(\cdot)$ denotes the discrete counterpart to $J(\cdot)$ based on the implicit Euler time discretisation. Keeping the structure of the necessary optimality conditions of the *First-Optimise-Then-Discretise* approach in mind, the formal Lagrange multiplier technique can be applied to derive a discrete KKT system: Neglecting the boundary conditions for the moment, a corresponding Lagrange functional reads

$$\begin{aligned} L_{\mathrm{IE}}(y,u,\lambda) := \quad & J_{\mathrm{IE}}(y,u) \\ & + \sum_{n=1}^{N}\big(\lambda_n, ku_n - (y_n - y_{n-1} - k\Delta y_n)\big) \\ & + \big(\lambda_0, (y^0 + k\Delta y^0) - (y_0 - k\Delta y_0)\big) \end{aligned} \tag{2.39}$$

for a set of dual velocities $\lambda = (\lambda_0, \lambda_1, ..., \lambda_N) \in V^{N+1}$. System (2.34) follows from $DL_{\mathrm{IE}}(y,u,\lambda) = 0$.

2.6. Time discretisation with a general one-step θ-scheme

The following paragraphs generalise the discretisation techniques from the implicit Euler scheme to a general one step θ-scheme in time. For $\theta = 1$, the scheme reduces to the implicit Euler scheme from above. For $\theta = \frac{1}{2}$, a variant of the Crank–Nicolson scheme is obtained which is second order in time. The discretisation is again directly applied to the nonstationary Navier–Stokes equations; results for the heat equation are mentioned briefly.

2.6.1. The Navier–Stokes equations

Similar to the implicit Euler case, the time interval $[0,T]$ is divided into $N \in \mathbb{N}$ equidistant intervals of length $k = 1/N$. Based on the idea in [59], a special modified θ-scheme is used which stores the dual solution at points in time different from the primal solution, see also Appendix D. This idea allows a more consistent notation of the algorithm, reduces the number of couplings in time and exploits at $\theta = \frac{1}{2}$ the special central-difference nature of Crank–Nicolson. Consider the scheme

$$\begin{aligned} \frac{y_n - y_{n-1}}{k} &+ \theta(-\nu\Delta y_n + y_n\nabla y_n) \\ &+ (1-\theta)(-\nu\Delta y_{n-1} + y_{n-1}\nabla y_{n-1}) + \nabla p_{n-1+\theta} = u_{n-1+\theta} \\ & \qquad\qquad - \operatorname{div} y_n = 0 \\ & \qquad\qquad y_0 = y^0 \end{aligned} \tag{2.40}$$

for $\theta \in [\frac{1}{2}, 1]$ and $n = 1, ..., N$, y_n, $u_{n-1+\theta} \in V$, $p_{n-1+\theta} \in Z$.

The right hand side $u_{n-1+\theta}$ is interpreted as an approximation to $u(\theta t_n + (1-\theta)t_{n-1})$. The same holds for $p_{n-1+\theta}$. There is $u_{n-1+\theta} = u_n + \mathcal{O}(k)$ and $u_{n-\frac{1}{2}} = \frac{1}{2}(u_{n-1} + u_n) +$

$\mathcal{O}(k^2)$ which reflects the second order accuracy in time for $\theta = \frac{1}{2}$. In comparison, the traditional Crank–Nicolson scheme uses $\frac{1}{2}(u_n + u_{n-1})$ as right hand side with u_n interpreted as approximation to $u(t_n)$, see also Figure 2.1. For a more detailed discussion about the interpretation of the pressure and the right-hand side in the time stepping scheme, see Appendix D.

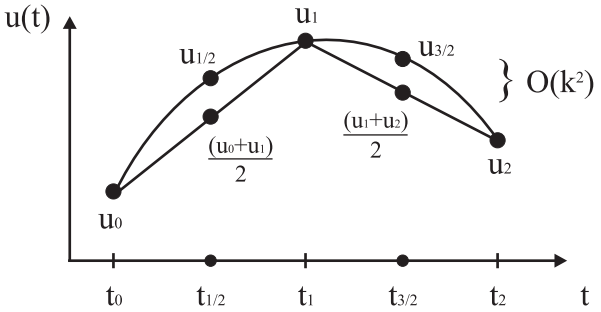

Figure 2.1: Difference in the right-hand side for a quadratic control u (in time). The traditional Crank–Nicolson uses $\frac{u_0+u_1}{2}$, $\frac{u_1+u_2}{2}$,... and modified Crank–Nicolson scheme $u_{\frac{1}{2}}$, $u_{\frac{3}{2}}$,...

In the next step, the discretisation recipe from [11] is applied. Additionally to the operators from Section 2.5, with $\tau \in \mathbb{R}$, the operator

$$\mathcal{C}_n^\tau := \mathcal{C}^\tau(y_n) : v \mapsto \tau(-\nu\Delta v + (y_n\nabla)v)$$

is used, for all velocity vectors $v \in V$. Using $X_k := (V \times Z)^{N+1}$ and $x := (y_0, p_{-1+\theta}, y_1, p_\theta, y_2, p_{1+\theta}..., y_N, p_{N-1+\theta}) \in X_k$, this yields the nonlinear system of the primal equation,

$$\begin{aligned}
\mathcal{H}^k x &:= \mathcal{H}^k(x)x := \mathcal{H}_\theta^k(x)x \\
&= \left(\begin{array}{c|c|c|c}
\frac{\mathcal{I}}{k} + \mathcal{C}_0^\theta \quad \mathcal{G} & & & \\
\mathcal{D} & & & \\
\hline
-\frac{\mathcal{I}}{k} + \mathcal{C}_0^{1-\theta} & \frac{\mathcal{I}}{k} + \mathcal{C}_1^\theta \quad \mathcal{G} & & \\
 & \mathcal{D} & & \\
\hline
 & \ddots & \ddots \quad \ddots & \\
\hline
 & & -\frac{\mathcal{I}}{k} + \mathcal{C}_{N-1}^{1-\theta} & \frac{\mathcal{I}}{k} + \mathcal{C}_N^\theta \quad \mathcal{G} \\
 & & & \mathcal{D}
\end{array}\right)
\left(\begin{array}{c} y_0 \\ p_{-1+\theta} \\ y_1 \\ p_\theta \\ \vdots \\ y_N \\ p_{N-1+\theta} \end{array}\right) \\
&= \left((\frac{\mathcal{I}}{k} + \mathcal{C}_0^\theta)y^0, 0, -\lambda_\theta/\alpha, 0, ..., -\lambda_{N-1+\theta}/\alpha, 0\right)^T
\end{aligned}$$

which is equivalent to (2.40) in case that y^0 is solenoidal because of a proper projection in the first timestep. This equation is the discrete counterpart to the primal equation (2.10a) on page 25 that appears during the derivation of the KKT system with the formal Lagrange principle.

In the second step, a discrete counterpart to the corresponding dual equation (2.10b) on page 25 is derived. The Fréchet derivative $D\mathcal{H}$ of the Navier–Stokes equations (see equation (2.8) on page 25) is discretised with the same timestepping scheme which has been used for $\mathcal{H}^k$. This leads to $D\mathcal{H}^k$, the discrete Fréchet derivative of the mapping $x \mapsto \mathcal{H}^k(x)x$. In particular, for a vector $\bar{x} := (\bar{y}_0, \bar{p}_{-1+\theta}, \bar{y}_1, \bar{p}_\theta, ..., \bar{y}_N, \bar{p}_{N-1+\theta}) \in X_k$ the

resulting scheme expressed in matrix-vector notation reads

$$D\mathcal{H}^k\bar{x} := D\mathcal{H}^k(x)\bar{x} := D\mathcal{H}^k_\theta(x)\bar{x} := \tag{2.41}$$

$$\left(\begin{array}{c|c|c|c} \frac{\mathcal{I}}{k}+\mathcal{N}_0^\theta \quad \mathcal{G} & & & \\ \mathcal{D} & & & \\ \hline -\frac{\mathcal{I}}{k}+\mathcal{N}_0^{1-\theta} & \frac{\mathcal{I}}{k}+\mathcal{N}_1^\theta \quad \mathcal{G} & & \\ & \mathcal{D} & & \\ \hline & \ddots & \ddots \quad \ddots & \\ \hline & & -\frac{\mathcal{I}}{k}+\mathcal{N}_{N-1}^{1-\theta} & \frac{\mathcal{I}}{k}+\mathcal{N}_N^\theta \quad \mathcal{G} \\ & & & \mathcal{D} \end{array}\right) \left(\begin{array}{c} \bar{y}_0 \\ \bar{p}_{-1+\theta} \\ \bar{y}_1 \\ \bar{p}_\theta \\ \vdots \\ \bar{y}_N \\ \bar{p}_{N-1+\theta} \end{array}\right)$$

with the additional operator

$$\mathcal{N}_n^\tau := \mathcal{N}^\tau(y_n) : v \mapsto \tau(-\nu\Delta v + (y_n\nabla)v + (v\nabla)y_n)$$

for all velocity vectors $v \in V$. Corresponding to (2.11) on page 26, the time discretisation of the dual equation is defined as the adjoint $D\mathcal{H}^{k,*}$ of $D\mathcal{H}^k$,

$$(\psi, D\mathcal{H}^k\bar{x}) = (D\mathcal{H}^{k,*}\psi, \bar{x}),$$

where $\psi := (\lambda_{-1+\theta}, \xi_0, \lambda_\theta, \xi_1, ..., \lambda_{N-1+\theta}, \xi_N) \in X_k$. With the adjoint of $\mathcal{N}_n^\tau$ being defined by

$$\mathcal{N}_n^{\tau,*} := \mathcal{N}^{\tau,*}(y_n) : v \mapsto \tau(-\nu\Delta v - (y_n\nabla)v + (\nabla y_n)^\mathsf{T} v)$$

for all velocity vectors $v \in V$, this reads

$$D\mathcal{H}^{k,*}\psi = D\mathcal{H}^{k,*}(x)\psi$$

$$= \left(\begin{array}{c|c|c|c} \frac{\mathcal{I}}{k}+\mathcal{N}_0^{\theta,*} \quad \mathcal{G} & -\frac{\mathcal{I}}{k}+\mathcal{N}_0^{1-\theta,*} & & \\ \mathcal{D} & & & \\ \hline & \frac{\mathcal{I}}{k}+\mathcal{N}_1^{\theta,*} \quad \mathcal{G} & -\frac{\mathcal{I}}{k}+\mathcal{N}_1^{1-\theta,*} & \\ & \mathcal{D} & & \\ \hline & & \ddots \quad \ddots & \ddots \\ \hline & & & \frac{\mathcal{I}}{k}+\mathcal{N}_N^{\theta,*} \quad \mathcal{G} \\ & & & \mathcal{D} \end{array}\right) \left(\begin{array}{c} \lambda_{-1+\theta} \\ \xi_0 \\ \lambda_\theta \\ \xi_1 \\ \vdots \\ \lambda_{N-1+\theta} \\ \xi_N \end{array}\right)$$

$$= (y_0 - z_0, 0, \ ..., \ y_{N-1} - z_{N-1}, 0, \ \theta(y_N - z_N), 0)^\mathsf{T}.$$

Similar to the implicit Euler case, the right-hand side is chosen in such a way that the *First-Optimise-Then-Discretise* approach commutes with the *First-Discretise-Then-Optimise* approach, see below. This corresponds to the time discretisation scheme

$$\begin{aligned} &\frac{\lambda_{n-1+\theta} - \lambda_{n+\theta}}{k} + \theta\big(-\nu\Delta\lambda_{n-1+\theta} - y_n\nabla\lambda_{n-1+\theta} + (\nabla y_n)^\mathsf{T}\lambda_{n-1+\theta}\big) \\ &\qquad +(1-\theta)\big(-\nu\Delta\lambda_{n+\theta} \quad - y_n\nabla\lambda_{n+\theta} \quad + (\nabla y_n)^\mathsf{T}\lambda_{n+\theta} \quad \big) + \nabla\xi_n = y_n - z_n \end{aligned} \tag{2.42a}$$

$$-\operatorname{div}\lambda_{n-1+\theta} = 0 \tag{2.42b}$$

$$\frac{\lambda_{N-1+\theta}}{k} + \theta\big(-\nu\Delta\lambda_{N-1+\theta} - y_N\nabla\lambda_{N-1+\theta} + (\nabla y_N)^\mathsf{T}\lambda_{N-1+\theta}\big) + \nabla\xi_N = \theta(y_N - z_N) \tag{2.42c}$$

applied to (2.17b) on page 28. Here, (2.42c) approaches the actual end time condition $\lambda(T,\cdot)=0$ for $k\to 0$.

Figure 2.2 highlights the special definition of the points in time where the primal and dual variables are located, here for $\theta=\frac{1}{2}$. The points in time for the primal pressure coincide with the points in time of the dual velocity and vice versa.

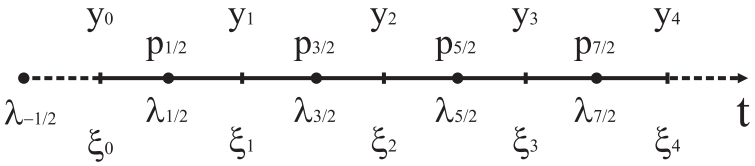

Figure 2.2: Distribution of the solutions on the time axis for $\theta=\frac{1}{2}$.

Shifting the terms with $\lambda_{n-1+\theta}$ and y_n in (2.40) and (2.42) from the right-hand side to the left-hand side and mixing the two matrices originating from $\mathcal{H}^k$ and $D\mathcal{H}^{k,*}$ once again results in a semi-discrete system

$$\mathbf{G}(w)w=f. \tag{2.43}$$

The global solution vector w has the form

$$\begin{aligned} w &:= \big(\underbrace{y_0,\lambda_{-1+\theta},p_{-1+\theta},\xi_0}_{w_0},\ \underbrace{y_1,\lambda_\theta,p_\theta,\xi_1}_{w_1},\ \underbrace{y_2,\lambda_{1+\theta},p_{1+\theta},\xi_2}_{w_2},\ \ldots\big)^{\mathsf{T}} \\ &\in (V\times V\times Z\times Z)^{N+1} \end{aligned}$$

with $w_n := (y_n,\lambda_{n-1+\theta},p_{n-1+\theta},\xi_n)$. The right-hand side is given by

$$f=\big(\underbrace{(\mathcal{I}/k+\theta\mathcal{C}_0)y^0,-(1-\theta)z_0,0,0}_{f_0},\ \underbrace{0,-z_1,0,0}_{f_1},\ \ldots,\ \underbrace{0,-z_{N-1},0,0}_{f_{N-1}},\ \underbrace{0,-\theta z_N,0,0}_{f_N}\big)^{\mathsf{T}}$$

and the left-hand side of the system reads

$$\mathbf{G}=\mathbf{G}^\theta(w)=\left(\begin{array}{c|c|c|c|c} \mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{G}_N \end{array}\right) \tag{2.44}$$

with

$$\mathbf{G}_0=\mathbf{G}_0^\theta=\begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{C}_0^\theta & 0 & \mathcal{G} & 0 \\ -(1-\theta)\mathcal{I} & \frac{\mathcal{I}}{k}+\mathcal{N}_0^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

$$\mathbf{G}_N=\mathbf{G}_N^\theta=\begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{C}_N^\theta & \frac{1}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -\theta\mathcal{I} & \frac{\mathcal{I}}{k}+\mathcal{N}_N^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

$$\mathbf{G}_n=\mathbf{G}_n^\theta=\begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{C}_n^\theta & \frac{1}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -\mathcal{I} & \frac{\mathcal{I}}{k}+\mathcal{N}_n^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}$$

for $n = 1, ..., N-1$ and

$$\breve{\mathbf{I}}_n = \breve{\mathbf{I}}_n^\theta = \begin{pmatrix} -\frac{\mathcal{I}}{k} + \mathcal{C}_{n-1}^{1-\theta} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \hat{\mathbf{I}}_{n-1} = \hat{\mathbf{I}}_{n-1}^\theta = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{\mathcal{I}}{k} + \mathcal{N}_{n-1}^{1-\theta,*} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

for $n = 1, ..., N$.

2.6.2. The Newton system associated with (2.43)

The application of the Newton algorithm (2.32) on page 34 requires the Fréchet derivative of the mapping $w \mapsto \mathbf{G}(w)w$. For the general one-step θ-scheme, it is given by the Newton matrix

$$\mathbf{F}(w) = \mathbf{F}^\theta(w) = \left(\begin{array}{c|c|c|c|c} \mathbf{F}_0 & \hat{\mathbf{J}}_0 & & & \\ \hline \breve{\mathbf{J}}_1 & \mathbf{F}_1 & \hat{\mathbf{J}}_1 & & \\ \hline & \breve{\mathbf{J}}_2 & \mathbf{F}_2 & \hat{\mathbf{J}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{J}}_N & \mathbf{F}_N \end{array}\right),$$

where

$$\mathbf{F}_0 = \mathbf{F}_0^\theta = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{N}_0^\theta & 0 & \mathcal{G} & 0 \\ -(1-\theta)\mathcal{I} + \mathcal{R}_0^\theta + \mathcal{R}_1^{1-\theta} & \frac{\mathcal{I}}{k} + \mathcal{N}_0^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

$$\mathbf{F}_N = \mathbf{F}_N^\theta = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{N}_N^\theta & \frac{\theta}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -\theta\mathcal{I} + \mathcal{R}_N^\theta & \frac{\mathcal{I}}{k} + \mathcal{N}_N^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

$$\mathbf{F}_n = \mathbf{F}_n^\theta = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{N}_n^\theta & \frac{\theta}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -\mathcal{I} + \mathcal{R}_n^\theta + \mathcal{R}_{n+1}^{1-\theta} & \frac{\mathcal{I}}{k} + \mathcal{N}_n^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

for $n = 1, ..., N-1$ and

$$\breve{\mathbf{J}}_n = \breve{\mathbf{J}}_n^\theta = \begin{pmatrix} -\frac{\mathcal{I}}{k} + \mathcal{N}_{n-1}^{1-\theta} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \hat{\mathbf{J}}_{n-1} = \hat{\mathbf{J}}_{n-1}^\theta = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{\mathcal{I}}{k} + \mathcal{N}_{n-1}^{1-\theta,*} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

for $n = 1, ..., N$ with the additional operator

$$\mathcal{R}_n^\tau := \mathcal{R}^\tau(\lambda_{n-1+\theta}) : v \mapsto \tau\big(-(v\nabla)\lambda_{n-1+\theta} + (\nabla v)^{\mathsf{T}}\lambda_{n-1+\theta}\big)$$

for all velocity vectors $v \in V$.

2.6.3. The heat equation

For the heat equation, the system reads

$$\mathbf{G}w = f \tag{2.45}$$

with

$$f = \big(\underbrace{(\mathcal{I}/k + \mathcal{A}^\theta)y^0, -(1-\theta)z_0}_{f_0},\ \underbrace{0, -z_1}_{f_1},\ ...,\ \underbrace{0, -z_{N-1}}_{f_{N-1}},\ \underbrace{0, -\theta z_N}_{f_N} \big)^\mathsf{T},$$

$$\mathbf{G} = \mathbf{G}^\theta = \left(\begin{array}{c|c|c|c|c} \mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{G}_N \end{array} \right), \tag{2.46}$$

whereby $\mathcal{A}^\tau : V \to V^*$, $\mathcal{A}^\tau : v \mapsto -\tau\Delta v$ for all $v \in V$. The submatrices read

$$\mathbf{G}_0 = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A}_0^\theta & 0 \\ -(1-\theta)\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A}_0^\theta \end{pmatrix}, \qquad \mathbf{G}_N = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{C}_N^\theta & \frac{1}{\alpha}\mathcal{I} \\ -\theta\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A}_N^\theta \end{pmatrix},$$

$$\mathbf{G}_n = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{C}_n^\theta & \frac{1}{\alpha}\mathcal{I} \\ -\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A}_n^\theta \end{pmatrix}$$

for $n = 1, ..., N-1$ and

$$\breve{\mathbf{I}}_n = \begin{pmatrix} -\frac{\mathcal{I}}{k} + \mathcal{A}_{n-1}^{1-\theta} & 0 \\ 0 & 0 \end{pmatrix}, \quad \hat{\mathbf{I}}_{n-1} = \begin{pmatrix} 0 & 0 \\ 0 & -\frac{\mathcal{I}}{k} + \mathcal{A}_{n-1}^{1-\theta} \end{pmatrix}$$

for $n = 1, ..., N$.

An equivalent *First-Discretise-Then-Optimise* strategy This system can also be obtained using a *First-Discretise-Then-Optimise* approach that imitates the *First-Optimise-Then-Discretise* approach. This approach is again presented for the optimal distributed control of the heat equation (2.37) on page 35.

For the semi-discretisation in time, the time integrals of the solution and the control are discretised with a scheme that respects their location. The first timestep is again projected. This yields

$$\begin{aligned} J_{\mathrm{TS}}(y,u) := \frac{1}{2}k\sum_{n=1}^{N}\Big((1-\theta)\,||y_{n-1} - z_{n-1}||_\Omega^2 + \theta\,||y_n - z_n||_\Omega^2 \Big) \\ + \frac{\alpha}{2}k\sum_{n=0}^{N} ||u_{n-1+\theta}||_\Omega^2 \end{aligned}$$

s.t.

$$\begin{aligned} (y_n - y_{n-1}) - k\big(\theta\Delta y_n + (1-\theta)\Delta y_{n-1}\big) &= ku_{n-1+\theta} && \text{in } \Omega,\ n = 1, ..., N \\ y_0 - k\big(\theta\Delta y_0\big) &= y^0 - k\theta\Delta y^0 && \text{in } \Omega \\ y_n &= g(t_n) && \text{at } \Gamma,\ n = 0, ..., N, \end{aligned}$$

where J_{TS} denotes the discrete counterpart to J based on the θ-scheme time discretisation. Note that for $\theta = 1/2$, the time integral of the primal equation y is discretised with the summed trapezoidal rule while the control u is discretised with the midpoint rule. Both schemes are second order in 1D which corresponds to the order of Crank–Nicolson discretisation in time.

Keeping the structure of the necessary optimality conditions of the *First-Optimise-Then-Discretise* approach in mind, the formal Lagrange multiplier technique can be applied to derive a discrete KKT system. Neglecting the boundary conditions at the moment, a corresponding Lagrange functional reads

$$\begin{aligned} L_{\mathrm{TS}}(y,u,\lambda) := {} & J_{\mathrm{TS}}(y,u) \\ & + \sum_{n=1}^{N} \Big(\lambda_{n-1+\theta}, k\, u_{n-1+\theta} - \big(y_n - y_{n-1} - k(\theta \Delta y_n + (1-\theta)\Delta y_{n-1})\big) \Big) \\ & + \big(\lambda_0, (y^0 + k\theta \Delta y^0) - (y_0 - k\theta \Delta y_0)\big) \end{aligned}$$

for $\lambda := (\lambda_{-1+\theta}, \lambda_\theta, ..., \lambda_{N-1+\theta}) \in X_k$. System (2.45) follows from $DL_{\mathrm{TS}}(y,u,\lambda) = 0$.

2.7. Discretisation in space — the fully discretised problem

The discretisation scheme which is applied in this work decouples the time discretisation from the space discretisation. In the previous two sections, the semi-discretisation in time of the continuous KKT system has been addressed. The resulting semi-discrete KKT system has been discrete in time and continuous in space. This section deals with the space discretisation of the resulting semi-discrete system and applies the finite element approach to derive a fully discrete counterpart.

Let $\Omega_h \subset \mathbb{R}^{\dim}$ be a triangulation of the domain Ω. The spaces V and (potentially) Z are replaced by appropriate finite element spaces V_h and (potentially) Z_h build upon the mesh Ω_h. The fully discretised version of the KKT system and its associated Fréchet derivative are defined by replacing the spatial operators $\mathcal{I}$, $\mathcal{C}$, $\mathcal{D}$, ... by their finite element counterparts $\mathcal{I}^h$, $\mathcal{C}^h$, $\mathcal{D}^h$, ... and by incorporating boundary conditions. For a convenient notation, the superscript $\sigma := (h,k)$ denotes a simultaneous discretisation in space and time, while single subscripts/superscripts h or k refer to a discretisation in space or in time, respectively. The nonlinear system for a discretisation with the implicit Euler scheme reads

$$G^\sigma(w^\sigma) w^\sigma = f^\sigma \tag{2.47}$$

whereas the one corresponding to the general θ-scheme is

$$G^\sigma(w^\sigma) w^\sigma = G^{\theta,\sigma}(w^\sigma) w^\sigma = f^\sigma. \tag{2.48}$$

Here, G^σ represents the fully discrete counterpart of the space-time matrix $\mathbf{G}$ that stems from the time discretisation. The vector w^σ has the form

$$w^\sigma = (w_0^h, w_1^h, ..., w_N^h) \in \mathcal{V}_h^{N+1}$$

with $N \in \mathbb{N}$ denoting the number of time intervals. The space $\mathcal{V}_h$ depends on the equation being discretised. For the heat equation, the space $\mathcal{V}_h$ is defined as $\mathcal{V}_h := V_h \times V_h$. The elements in w^σ are given by

$$w_n^h = (y_n^h,\ \lambda_n^h)$$

for the implicit Euler scheme or generally

$$w_n^h = (y_n^h,\ \lambda_{n-1+\theta}^h)$$

for the general θ-scheme, $n = 0, ..., N$. For the Stokes/Navier–Stokes equations, it is convenient to define $\mathcal{V}_h := V_h \times V_h \times Z_h \times Z_h$. The elements in w^σ have the form

$$w_n^h = (y_n^h,\ \lambda_n^h,\ p_n^h,\ \xi_n^h) \qquad \text{or} \qquad w_n^h = (y_n^h,\ \lambda_{n-1+\theta}^h,\ p_{n-1+\theta}^h,\ \xi_n^h),$$

respectively. The right-hand side is given by

$$f^\sigma = (f_0^h, f_1^h, ..., f_N^h) \in (\mathcal{V}_h^*)^{N+1}, \qquad f_n^h \in \mathcal{V}_h^*.$$

With the corresponding dual pairing $(\cdot,\cdot)_{\mathcal{V}_h^*,\mathcal{V}_h}$ in $\mathcal{V}_h$, the space $\mathcal{V}_h^*$ can be identified[2] with $\mathcal{V}_h$,

$$\mathcal{V}_h^* \cong \mathcal{V}_h,$$

which immediately induces

$$(\mathcal{V}_h^*)^{N+1} \cong \mathcal{V}_h^{N+1}.$$

via an appropriate dual pairing on $\mathcal{V}_h^{N+1}$. Therefore, f_n^h has a unique representation in $\mathcal{V}_h$ and f^σ has a unique representation in $\mathcal{V}_h^{N+1}$.

The Fréchet derivative of $w^\sigma \mapsto G^\sigma(w^\sigma)w^\sigma$ is denoted by $F^\sigma(w^\sigma)$ and $F^\sigma(w^\sigma) = F^{\sigma,\theta}(w^\sigma)$, respectively. F^σ is the discrete counterpart to $\mathbf{F}$. An important fact is that all matrices analogously feature a block tridiagonal form,

$$G^\sigma = G^\sigma(w^\sigma) = \left(\begin{array}{c|c|c|c} G_0 & \hat{M}_0 & & \\ \hline \breve{M}_1 & G_1 & \hat{M}_1 & \\ \hline & \ddots & \ddots & \ddots \\ \hline & & \breve{M}_N & G_N \end{array}\right), \quad F^\sigma = F^\sigma(w^\sigma) = \left(\begin{array}{c|c|c|c} F_0 & \hat{M}_0 & & \\ \hline \breve{M}_1 & F_1 & \hat{M}_1 & \\ \hline & \ddots & \ddots & \ddots \\ \hline & & \breve{M}_N & F_N \end{array}\right)$$

where $N \in \mathbb{N}$ denotes the number of time intervals. Thus, the solver for the optimal control problem reduces to a solver for a sparse block tridiagonal system, where the diagonal blocks $G_n = G_n(w^\sigma)$ correspond to the timesteps of the fully coupled KKT system.

2.3 Remarks. A solver for system (2.47) or (2.48) can be defined without the need to store the complete system in memory. Indeed, only space-time vectors have to be stored: Utilising defect correction algorithms reduces the solution process to a sequence of matrix vector multiplications in space. A matrix-vector multiplication of a solution w^h with the space-time matrix G^σ (or F^σ) reduces to $3N + 1$ local matrix-vector multiplications with approximately sparse matrices, three for each time interval with $\breve{M}_n$, G_n (or F_n) and $\hat{M}_n$. These local matrices can be created on-demand, so G^σ (or F^σ) does not have to be stored in its complete form. A more detailed description of this procedure will be given in the next chapter.

2.8. The *First-Discretise-Then-Optimise* strategy

The above approach uses a *First-Optimise-Then-Discretise* strategy: First, the continuous optimality system is derived. Second, an appropriate discretisation is applied. This yields systems (2.47) and (2.48), respectively.

[2] Since the spaces are discrete, this association is isomorphic.

The same system can also be derived by using a *First-Discretise-Then-Optimise* strategy. However, the term *First-Discretise-Then-Optimise* is not unique in the literature. Instead of following the Lagrange multiplier approach, starting from the minimisation problem, a discretisation in space and time leads to a fully discrete quadratic minimisation problem. As an example, the heat equation (2.1) on page 22 with homogeneous boundary conditions is considered. This reads

$$J(y,u) := \frac{1}{2}||y-z||_Q^2 + \frac{\alpha}{2}||u||_Q^2 \quad \longrightarrow \quad \min! \tag{2.49}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \Delta y &= u && \text{in } Q, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= 0 && \text{at } \Sigma \end{aligned}$$

for appropriate y, u, z, y^0 and g.

Discretising in space with finite elements and in time with the rectangular rule yields the discrete counterpart for the above system

$$J_{\mathrm{IE}}(y^\sigma, u^\sigma) := \frac{1}{2}k\sum_{n=1}^{N}||y_n - z_n||_\Omega^2 + \frac{\alpha}{2}k\sum_{n=1}^{N}||u_n||_\Omega^2 \quad \longrightarrow \quad \min! \tag{2.50a}$$

$$\left.\begin{aligned} \text{s.t.} \qquad \frac{y_n - y_{n-1}}{k} - \Delta y_n &= u_n && \text{in } Q, \\ y_0 - \Delta y_0^\sigma &= y^0 - \Delta y^0 && \text{in } \Omega, \\ y_n &= 0 && \text{at } \Sigma. \end{aligned}\right\} \tag{2.50b}$$

with $y^\sigma = (y_0, y_1, ..., y_N) \in V_h^{N+1}$, $u^\sigma = (u_0, u_1, ..., u_N) \in V_h^{N+1}$. There are basically two choices now:

i) The system is fully discrete in the degrees of freedom of y^σ and u^σ. Thus, standard nonlinear programming techniques can be applied to solve the system, see for example [17, 103, 123, 153]. For this purpose, it is a common way to define a reduced optimisation problem and an iteration for u^σ. At first, a solution operator $S_h : V_h^{N+1} \to V_h^{N+1}$ to (2.50b) is defined, with $S_h(u^\sigma) = y^\sigma$ for every $u^\sigma \in V_h^{N+1}$. Then, the minimisation problem (2.50) is equivalent to the reduced discrete quadratic minimisation problem

$$\begin{aligned} \tilde{J}_{\mathrm{IE}}(u^\sigma) &:= J_{\mathrm{IE}}(S_h(y^\sigma), u^\sigma) \\ &= \frac{1}{2}k\sum_{n=1}^{N}||(S_h(u^\sigma))_n - z_n||_\Omega^2 + \frac{\alpha}{2}k\sum_{n=1}^{N}||u_n||_\Omega^2 \quad \longrightarrow \quad \min!. \end{aligned} \tag{2.51}$$

At this point, standard methods from nonlinear programming can be applied, e. g., steepest descent or nonlinear CG method. The operator $S_h(\cdot)$ is used in a black box manner. As a disadvantage of this approach, the complete mathematical structure of the problem is 'hidden' in S_h and usually not exploited by the optimisation algorithm.

ii) As an alternative, the Lagrange multiplier method can be applied, cf. Section 2.5.3. Neglecting the boundary conditions for the moment, a Lagrange functional associated

with (2.50a) reads

$$\begin{aligned} L_{\mathrm{IE}}(y^\sigma, u^\sigma, \lambda^\sigma) &:= J_{\mathrm{IE}}(y^\sigma, u^\sigma) \\ &+ \sum_{n=1}^{N} \big(\lambda_n, ku_n - (y_n - y_{n-1} - k\Delta y_n)\big) \\ &+ \big(\lambda_0, (y^0 + k\Delta y^0) - (y_0 - k\Delta y_0)\big) \end{aligned}$$

with $\lambda^\sigma = (\lambda_0, \lambda_1, ..., \lambda_N) \in V_h^{N+1}$. This is the same formula as obtained by applying the space discretisation to (2.39) on page 36. Demanding

$$DL_{\mathrm{IE}}(y^\sigma, u^\sigma, \lambda^\sigma) = 0$$

and eliminating the control u^σ results in system (2.47). In the case of the general θ-scheme, the steps are the same and lead to (2.48). Thus, the *First-Optimise-Then-Discretise* approach commutes with the *First-Discretise-Then-Optimise* approach. Since the discrete system stems from a continuous one, the mathematical structure of the problem is still available and can be exploited by an appropriate solver.

As a summary, the proposed schemes (2.47) and (2.48) can be seen as the 'most effective' discretisation techniques in the sense that they result from a rather straightforward *First-Optimise-Then-Discretise* approach that commutes with *First-Discretise-Then-Optimise* approach. It can be said that 'the best from two worlds is combined': From the *First-Optimise-Then-Discretise* view, the so-called 'inconsistent gradients' are avoided which can lead to a premature stop in the iteration (cf. [69]) without having reached a meaningful optimum. From the *First-Discretise-Then-Optimise* view, these schemes aim at reflecting as much structure from the *First-Optimise-Then-Discretise* approach — i. e., from the continuous optimisation problem — as possible, see also [92, Chapter 3.2.4].

2.9. Summary and conclusions

This chapter has provided the first step in the design of a solver methodology for the optimal distributed control of nonstationary partial differential equations. In the beginning, formulations for the optimal distributed control of the heat equation, the Stokes equations and the Navier–Stokes equations have been introduced as a set of model problems. Based on the formal Lagrange multiplier technique, first order necessary optimality conditions have been formulated in a continuous setting.

In a second step, these KKT systems have been discretised in time. The discretisation has been carried out on the one hand based on the implicit Euler timestepping scheme, on the other hand based on the general one-step θ-scheme which contains the Crank–Nicolson scheme as special case. The θ-scheme uses a special interpretation of the functions on the time axis since the dual solution is not interpreted as being located at the endpoints of the time interval.

At the end, a discretisation in space with finite elements has been applied. The whole discretisation has followed a special discretisation recipe which guarantees that the so called *First-Optimise-Then-Discretise* strategy (i. e., first create the continuous optimality conditions, then discretise them) commutes with the so called *First-Discretise-Then-Optimise* strategy (i. e., first discretise the minimisation problem, then apply an optimisation strategy). This is a highly desirable property.

The *First-Optimise-Then-Discretise* strategy imposes a lot of structure into the discrete optimisation problem. The problem is reformulated as a set of partial differential equations

in the continuous sense which shows an elliptic character on the space-time cylinder. This is the key for efficiency. In the next chapter, Chapter 3, the continuous formulation of the KKT system will be exploited to generate a hierarchy of discrete KKT systems based on a set of meshes on the space-time cylinder. With the help of appropriate projection operators that allow to traverse this hierarchy, the multigrid framework will be applied. This approach promises level-independent convergence rates and thus, in combination with an additional multigrid solver for subproblems in space, linear complexity of the whole algorithm.

The *First-Discretise-Then-Optimise* strategy has the advantage to provide meaningful gradient information, i. e., descent directions with respect to the discrete functional to be minimised. Inconsistent gradients are one of the main disadvantages of the *First-Optimise-Then-Discretise* strategy for common mesh resolutions in practice [69, Section 2.9, Section 4.1] since they can lead to an iteration which does not decrease the functional $J(\cdot)$ — neither the continuous one, nor its discrete counterpart. In such a situation, the computed solution can be far away from the optimum. An approach that follows the *First-Discretise-Then-Optimise* strategy on the other hand does not suffer from inconsistent gradients. Loosely speaking, computed search directions are downhill directions and the computed result is a minimiser, at least for the discrete counterpart of the functional $J(\cdot)$.

The next chapter will concentrate on the design of the solver. Apart from the definition of the problem hierarchy and the general solver approaches, the two main issues will be the definition of appropriate projection operators to traverse the hierarchy — which are not obvious if the general θ-scheme is chosen for the time discretisation — and the definition of efficient one-level smoothers, preconditioners and solvers. Exploiting the special structure of the global space-time matrices, all operations will in the end reduce to global matrix-vector multiplications and local linear systems based on the diagonal blocks of global matrices.

3

The multigrid and the Newton solvers

This chapter builds up a solver methodology to solve the fully discrete KKT system which has been derived in Chapter 2. Basically, an outer Newton iteration processes the nonlinearity. The linear subproblems appearing during the Newton algorithm are solved with a space-time multigrid approach. This approach exploits the ellipticity in the underlying optimisation problem: The multigrid method is known to converge with level-independent convergence rates for a large class of elliptic problems. Since the KKT system also exhibits an elliptic character on the space-time cylinder (see Section 2.4 on page 28ff), this method is expected to work also in this situation.

The multigrid preconditioner is formulated based on a hierarchy of systems. The infinite dimensional KKT system is at first discretised with the methods from Chapter 2 on a hierarchy of space-time meshes. In a natural way, this leads to a hierarchy of problems and serves as a basis for a multigrid solver. On each level of the hierarchy, standard block-preconditioning techniques allow to formulate proper one-level solvers and smoothers which reduce global space-time problems to sequences of linear systems in space. In the end, these can be processed by a monolithic multigrid scheme in space that exploits the hierarchy of finite element spaces.

Outline

Section 3.1 describes how the discretisation techniques from Chapter 2 can be used to generate a hierarchy of discrete problems associated with any of the KKT systems of the model problems. In particular, this section introduces different methods for creating a hierarchy of meshes on the space-time cylinder. There is some freedom in the choice due to the fact that the space-discretisation is independent of the time-discretisation.

Section 3.2 and 3.3 formulate the basic Newton solver and the basic multigrid solver in the space-time framework. The multigrid solver is used as a preconditioner for the linear subproblems that appear during the nonlinear loop on the finest level. Both algorithms can be formulated in a straightforward way, although the multigrid method needs additional components to be defined properly.

The first couple of additional components needed by the multigrid algorithm are prolongation and restriction operators, which are described in Section 3.4. These operators are a combination of prolongation/restriction operators in space and in time. Unfortunately, due to the fact that a special variant of the Crank–Nicolson method is used for the time discretisation in this work, the definition of prolongation/restriction operators in time is not obvious. For a proper formulation, the concept of 'discrete abstract functions' is introduced here. This adapts the usual concept of abstract functions to the situation that a function is represented as set of discrete values in time, stemming from a discretisation

with finite differences. The prolongation in time is formulated as interpolation and the restriction as the adjoint of the prolongation with the help of a scalar product.

Furthermore, the multigrid algorithm needs the definition of iterative one-level smoothing operators and a one-level solver for the coarse grid. These operators are described in Section 3.5. For the definition, standard linear solvers for block systems (e. g., block Jacobi, block Gauß–Seidel) are adapted to the space-time case, every block corresponding to one solution in time. As a result, solving a linear system on the space-time cylinder is reduced to solving sequences of linear systems in space.

The linear systems in space can finally be solved with a monolithic multigrid method as well. However, this involves the definition of proper smoothing algorithms, which are not obvious for saddle-point problems like the Stokes or Navier–Stokes equations. Section 3.6 contains a detailed description of a special 'local Pressure-Schur complement' smoother which is adapted from CFD to the optimal control context.

The chapter closes with Section 3.7 which introduces the basic notation concerning the stopping criteria of all the solver components. Additionally, the section contains a description of the 'inexact Newton' method — a variant of the standard Newton method that adapts the stopping criteria of the inner solvers to save CPU time. This closes the basic description of the method. Of course, there are a couple of extensions possible, and some of them will be described in Chapter 4 and the appendices. The numerical analysis of the described solver strategy will start in Chapter 5.

3.1. Definition of hierarchies

In the following, the fully discrete KKT system is denoted by

$$G^\sigma(w^\sigma)w^\sigma = f^\sigma \tag{3.1}$$

with $\sigma = (h, k)$ standing for to a discretisation in space and time. $G^\sigma(\cdot)$ represents the nonlinear, fully discrete KKT operator from (2.47) or (2.48) on page 42. $w^\sigma \in \mathcal{V}_h^{N+1}$ refers to the space-time solution vector with $N+1$ solutions in time, $N \in \mathbb{N}$, discretised in space with finite elements. The underlying finite element space is called $\mathcal{V}_h$, build upon a mesh $\Omega_h \subset \mathbb{R}^{\dim}$. For convenience, this space is assumed to be a cross product of all involved finite element spaces, i. e., in the case of the Stokes or the Navier–Stokes equations, $\mathcal{V}_h = V_h \times V_h \times Z_h \times Z_h$ in the notation of the previous chapter.

The vector $f^\sigma \in \mathcal{V}_h^{N+1}$ identifies the right-hand side and $F^\sigma(w^\sigma)$ denotes the Fréchet derivative of $w^\sigma \mapsto G^\sigma(w^\sigma)w^\sigma$. The underlying mesh Ω_h that was used to define the finite element space is assumed to be the finest mesh from a hierarchy of triangulations, see below.

Hierarchies generated by refinement and coarsening There are different choices how to define a hierarchy of discretisations on a space-time cylinder. The simplest one starts with the definition of a coarse mesh in space, a coarse mesh in time and appropriate discretisations. Using simultaneous refinement in both space and time defines a hierarchy of levels. A more general approach is to separately set up a space hierarchy and a time hierarchy and create the space-time hierarchy by coarsening. These approaches read as follows.

3.1.1. Hierarchies in space and in time

For $L \in \mathbb{N}$, Ω_1, ..., Ω_L refers to a conforming hierarchy of triangulations of the domain Ω in the sense of [37] with $\Omega_L = \Omega_h$. The mesh Ω_1 is the basic coarse mesh and Ω_{l+1} stems from a regular refinement of Ω_l (i. e., new vertices, edges, faces and cells are generated by connecting opposite midpoints of edges/faces).

For each mesh Ω_l, $l = 1, ..., L$, a space discretisation with finite elements is carried out, cf. Section 2.7 on page 42f. This leads to a hierarchy

$$V^1, ..., V^L = \mathcal{V}_h$$

of spatial finite element spaces built upon these meshes.

For $M \in \mathbb{N}$, T^1, ..., T^M defines a hierarchy of decompositions of the time interval $[0, T]$ into ordered sequences, where each T^{m+1} is derived from T^m by a bisection of each time interval,

$$T^m := \left\{0, \frac{T}{N_m}, \frac{2T}{N_m}, ..., T\right\}, \quad N_m = 2^{m-1}N_1, \quad N_1 \in \mathbb{N}, \quad m = 1, ..., M. \tag{3.2}$$

Here, N_m refers to the number of time intervals on level m. To work with a general θ-scheme,

$$T_\theta^m := \left\{0, \frac{\theta T}{N_m}, \frac{(1+\theta)T}{N_m}, ..., \frac{(N_m - 1 + \theta)T}{N_m}\right\} \tag{3.3}$$

is defined, which coincides with T^m for $\theta = 1$ and which represents the arithmetic mean of neighbouring points in T_m for $\theta = \frac{1}{2}$.

3.1.2. Space-time hierarchies created by coarsening strategies

Based on the hierarchy of meshes in space and in time, a space-time hierarchy is created by coarsening. Each combination of a spatial mesh Ω_l, $l = 1, ..., L$, with a temporal mesh T_m, $m = 1, ..., M$, defines a possible space-time mesh. For every such a combination, the combined solution space

$$W^{l,m} := (V^l)^{N_m+1}$$

is defined. The finest possible combination of spatial and temporal mesh is used to define the fine grid space,

$$W^{\mathrm{NLMAX}} := W^{L,M},$$

for a 'maximum level' NLMAX $\in \mathbb{N}$.

In the next step, a hierarchy of spaces W^1, W^2,..., W^{NLMAX} is to be defined, and different strategies are possible here. A straightforward way to define such a hierarchy is to apply a coarsening strategy to W^{NLMAX}, based on the available spatial and temporal meshes, for example:

- Semi-coarsening in time:

$$(W^{\mathrm{NLMAX}}, W^{\mathrm{NLMAX}-1}, ..., W^1) := (W^{L,M}, W^{L,M-1}, W^{L,M-2}, ...)$$

- Usual 1:1 coarsening:

$$(W^{\mathrm{NLMAX}}, W^{\mathrm{NLMAX}-1}, ..., W^1) := (W^{L,M}, W^{L-1,M-1}, W^{L-2,M-2}, ...)$$

- Anisotropic coarsening in time, e. g.:

$$(W^{\mathrm{NLMAX}}, W^{\mathrm{NLMAX}-1}, ..., W^1) := (W^{L,M}, W^{L,M-1}, W^{L-1,M-2}, W^{L-1,M-3}, W^{L-2,M-4}, ...)$$

- Semi-coarsening in space:

$$(W^{\mathrm{NLMAX}}, W^{\mathrm{NLMAX}-1}, ..., W^1) := (W^{L,M}, W^{L-1,M}, W^{L-2,M}, ...)$$

or combinations of these. Figure 3.1 illustrates an example of a full space-time hierarchy obtained by 1:1 coarsening and Figure 3.2 the semi-coarsening in time. The choice of the hierarchy influences the solver stability, the total numerical costs of solving the system as well as theoretical properties of the solver. This issue will be discussed in later chapters for some selected numerical tests. In particular it will be shown that a wrong choice of the hierarchy can lead to an algorithm that loses the property of having linear complexity — in theory as well as in practice.

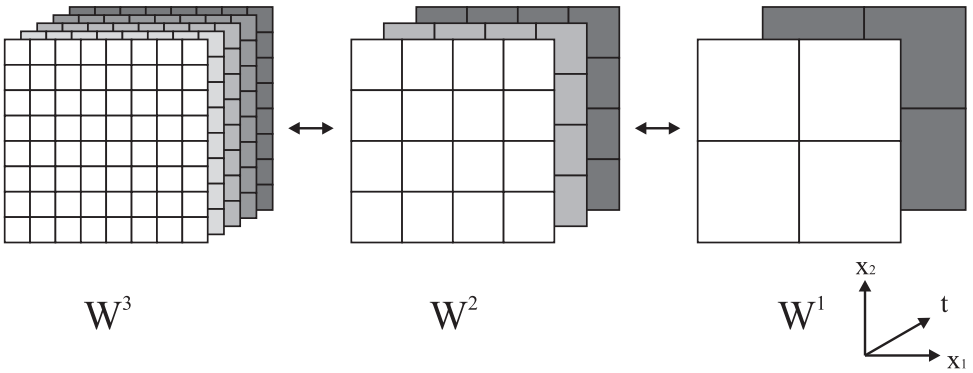

Figure 3.1: A space-time hierarchy on $\mathcal{Q} = (0,T) \times \Omega$ with $\Omega \subset \mathbb{R}^2$, obtained by 1:1 coarsening.

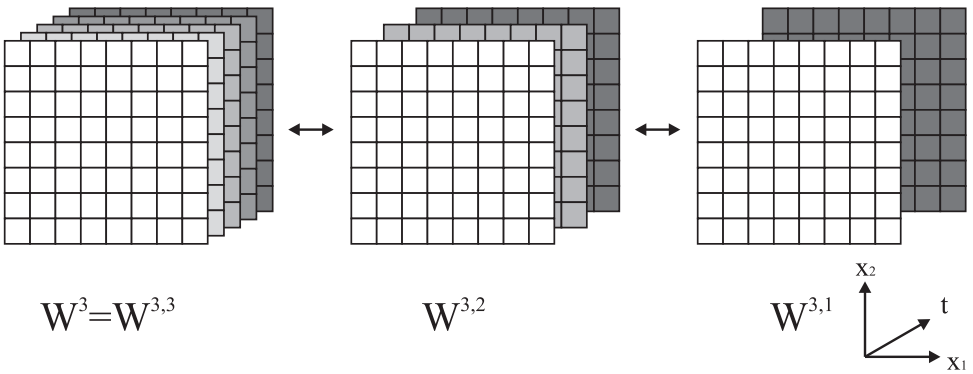

Figure 3.2: A space-time hierarchy on $\mathcal{Q} = (0,T) \times \Omega$ with $\Omega \subset \mathbb{R}^2$, obtained by semi-coarsening in time.

3.1.3. Problem hierarchies

The discretisation of the KKT system in space and time on space level l and time level m leads to the discrete space-time system which is denoted by

$$G^{lm}(w^{lm})w^{lm} = f^{lm}, \tag{3.4}$$

and

$$F^{lm}(w^{lm}) \tag{3.5}$$

refers to the Fréchet derivative of the operator $w \mapsto G^{lm}(w)w$ evaluated in $w = w^{lm}$. Here,

$$w^{lm} \in W^{l,m}, \qquad f^{lm} \in (W^{l,m})^{\times},$$

whereby

$$(W^{l,m})^{\times} := (V^{l,*})^{N_m+1} \cong (V^l)^{N_m+1}$$

refers to the space of right-hand sides, $V^{l,*} = (V^l)^*$. Using an appropriate dual pairing (see Section 2.7 on page 42f), this space is identified with $(V^l)^{N_m+1}$, which is expressed in the notation '$\cong$'.

3.1 Remark. The actual space for the right hand sides is $(W^{l,m})^*$. However, due to the fact that the spaces are discrete, there is

$$(W^{l,m})^* = \big((V^l)^{N_m+1}\big)^* \cong (V^{l,*})^{N_m+1} = (W^{l,m})^{\times}.$$

In this work, the notation $(W^{l,m})^{\times}$ is used to underline the fact that the right hand side consists of $N_m + 1$ components, one for each solution in time.

Simplified notation To simplify the definition of the multigrid method, it is convenient to assume $L = M = \mathrm{NLMAX}$. This corresponds to the above 1:1 coarsening. For each level $l = 1, ..., \mathrm{NLMAX}$, the solution and right-hand side spaces are abbreviated as

$$W^l := W^{l,l} = (V^l)^{N_l+1}, \qquad (W^l)^{\times} = (W^{l,l})^{\times} \cong (V^l)^{N_l+1}.$$

The corresponding space-time system

$$G^l(w^l)w^l = f^l \tag{3.6}$$

is of the form (3.1) for a solution vector $w^l \in W^l$, a right-hand side $f^l \in (W^l)^{\times}$ and the system operator $G^l : W^l \to (W^l)^{\times}$. In particular, on level NLMAX, the system reads

$$G^{\mathrm{NLMAX}}(w^{\mathrm{NLMAX}})w^{\mathrm{NLMAX}} = f^{\mathrm{NLMAX}} \tag{3.7}$$

with $w^{\mathrm{NLMAX}} = w^{\sigma}$, $f^{\mathrm{NLMAX}} = f^{\sigma}$ and $G^{\mathrm{NLMAX}} = G^{\sigma}$ representing the discrete right-hand side, the solution and system operator on the finest level, respectively.

3.2. The outer defect correction loop

The discrete system (3.7), which stems from the discretisation, is nonlinear. A typical way to solve such a nonlinear system is a nonlinear (preconditioned) defect correction loop, or 'fixed point iteration', which can be found, e. g., in [53, 104, 142]. For an initial guess $w_0^{\sigma} \in W^{\mathrm{NLMAX}}$, a fixed point iteration is typically defined in the form

$$w_{i+1}^{\sigma} := w_i^{\sigma} + C(w_i^{\sigma})^{-1}\big(f^{\sigma} - G^{\sigma}(w_i^{\sigma})w_i^{\sigma}\big), \qquad i \in \mathbb{N}. \tag{3.8}$$

Algorithm 3.1 illustrates this iteration in an algorithmic style. As usual, the update (3.8) is expressed in two steps, involving the solution of a linear system $C(w_i^h)g_i = d_i$, see (3.9). An efficient way to obtain a solution of this auxiliary system is the space-time multigrid method which is introduced in the next section.

Fixed point and Newton method A crucial choice in this defect correction loop is the preconditioner $C(w_i^\sigma)$. One possibility is to choose

$$C(w_i^\sigma) := G^\sigma(w_i^\sigma)$$

as preconditioner; the corresponding algorithm is simply called 'fixed point method' in the following. This iteration was used, e. g., in [142] in the context of simulation. Another common choice (cf. [53]) is the Fréchet derivative matrix $F^\sigma(w_i^\sigma)$ as defined in Section 2.7 on page 42ff,

$$C(w_i^\sigma) := F^\sigma(w_i^\sigma).$$

The corresponding algorithm defines a Newton iteration on the space-time cylinder.

Algorithm 3.1 Nonlinear defect correction loop

```
1: function NonlinearDefectCorrection(w_0^σ, f^σ)
2:     i ← 0
3:     while (w_i^σ not converged) do
4:         Solve C(w_i^σ) g_i = d_i := (f^σ − G^σ(w_i^σ) w_i^σ)        (3.9a)
5:         w_{i+1}^σ ← w_i^σ + g_i        ∈ W^NLMAX                    (3.9b)
6:         i ← i + 1
7:     end while
8:     return w_i^σ
9: end function
```

3.3. The inner multigrid solver

In each nonlinear step, the linear equation (3.9a) has to be solved, which reads

$$C(w_i^\sigma) g_i \;=\; d_i. \tag{3.10}$$

By exploiting the hierarchical structure of the space-time meshes, system (3.10) can be solved within a multigrid framework. A description of this framework can be found, e. g., in [9, 75, 118, 161]. It necessitates a couple of definitions, i. e., prolongation and restriction operators, a set of coarse grid matrices, a smoother on every level and a coarse grid solver:

a) For $l = 1, ..., \mathrm{NLMAX} - 1$, the operator $P^l : W^l \to W^{l+1}$ denotes a prolongation operator and $R^l : (W^{l+1})^\times \to (W^l)^\times$ a corresponding restriction operator.

b) For each level l, C^l refers to a representation of the operator $C(w_i^\sigma)$ on level l. The sequence $C^1, ..., C^{\mathrm{NLMAX}-1}$ forms a hierarchy of operators on $W^1, ..., W^{\mathrm{NLMAX}-1}$ approximating $C(w_i^\sigma)$. On level $l = \mathrm{NLMAX}$, $C^{\mathrm{NLMAX}} := C(w_i^\sigma)$ is used. For a proper definition of the coarse grid preconditioning operators, see Section 3.4.5.

c) For level $l = 2, ..., \mathrm{NLMAX}$, $S : W^l \to W^l$ defines a *smoothing* operator with $\mathrm{NSMpre}, \mathrm{NSMpost} \in \mathbb{N}_0$ referring to the numbers of pre- and postsmoothing steps, respectively.

d) For level $l = 1$, $(C^l)^{-1}$ refers to a coarse grid solver.

Appropriate definitions of all these operators follow in the next sections. Algorithm 3.2 describes a basic multigrid V-cycle to solve a system of the form

$$C^l w = f, \qquad w \in W^l,\ f \in (W^l)^\times,$$

and problem (3.9a) is solved by the call

$$g_i := \text{SpaceTimeMultigrid}(0; d_i; \text{NLMAX})).$$

For variations of this algorithm which use the W- or F-cycle, see [9, 75, 161].

Algorithm 3.2 Space-time multigrid

Predefined constant: NSMpre $\in \mathbb{N}_0$: number of presmoothing steps
Predefined constant: NSMpost $\in \mathbb{N}_0$: number of postsmoothing steps

function SpaceTimeMultigrid(w;f;l)
  if ($l = 1$) **then**
    return $(C^l)^{-1} f$ ▷ coarse grid solver
  end if
  while (not converged) **do**
    $w \leftarrow S(C^l, w, f, \text{NSMpre})$ ▷ presmoothing
    $d^{l-1} \leftarrow R^l(f - C^l w) \quad \in (W^{l-1})^\times$ ▷ restriction of the defect
    $g^{l-1} \leftarrow \text{SpaceTimeMultigrid}(0; d^{l-1}; l-1) \quad \in W^{l-1}$ ▷ coarse grid solution
    $w \leftarrow w + P^l(g^{l-1})$ ▷ coarse grid correction
    $w \leftarrow S(C^l, w, f, \text{NSMpost})$ ▷ postsmoothing
  end while
  return w ▷ solution
end function

3.4. Prolongation/Restriction/Coarse grid preconditioning operators

The discretisation of the KKT system is based on finite differences in time and finite elements in space. The operators for transferring solution/correction and right-hand side/defect vectors between the different levels can therefore be decomposed into a finite difference prolongation/restriction in time (see also [76]) and finite element prolongation/restriction in space.

3.4.1. Preliminaries

Let $l \in \{1, ..., \text{NLMAX}-1\}$ be a space-time level and $N = N_l$ the number of time intervals on that level. In the following, the vector

$$w = w^l = (w_0, ..., w_N) \in W^l$$

stands for a space-time vector on level l to be prolongated and

$$d^{l+1} = (d_0^{l+1}, ..., d_{2N}^{l+1}) \in (W^{l+1})^\times$$

for a space-time defect vector on level $l+1$ to be restricted. A prolongation

$$P^l : W^l \to W^{l+1}$$

and a corresponding restriction

$$R^l : (W^{l+1})^\times \to (W^l)^\times$$

is formulated as a combination of a finite difference prolongation/restriction in time combined with a finite element prolongation/restriction in space.

Prolongation/Restriction in space Appropriate prolongation/restriction operators in space are usually available via the finite element approach, see for example [75, 142]. In the following,

$$P_{\text{space}} = P^l_{\text{space}} : V^l \to V^{l+1}$$

denotes a finite element prolongation and

$$R_{\text{space}} = R^l_{\text{space}} : V^{l+1,*} \to V^{l,*}$$

the corresponding restriction. It is noted that using the appropriate dual pairing, it is possible to associate

$$V^{l,*} \cong V^l \qquad \text{and} \qquad V^{l+1,*} \cong V^{l+1}.$$

Thus, R_{space} has a representation as an operator $R_{\text{space}} = R^l_{\text{space}} : V^{l+1} \to V^l$. It is usually formulated as weighted mean of the degrees of freedom, while the prolongation is realised as an appropriate interpolation to the higher level.

Prolongation/Restriction in time The definition of the finite difference prolongation/restriction in time involves some special notation. In the end, it can be written down as matrix-vector product, where a prolongation/restriction matrix is applied to the subvectors in time. The choice of the matrix depends on the choice of the timestep scheme. For a convenient description, the following concept of 'discrete abstract functions' is introduced, which allows to adapt concepts of the 1D case in [75, 76] to the situation considered here. Sections 3.4.3 and 3.4.4 apply this concept to derive prolongation/restriction operators in time and in space-time.

3.4.2. Discrete abstract functions

In the following, Ξ denotes an ordered sequence of $N+1$ points, $N \in \mathbb{N}$, on the time interval $[0,T]$,

$$\Xi = \{\xi_0 < \xi_1 < ... < \xi_N \,|\, 0 \le \xi_n \le T,\, n = 0, ..., N\}.$$

Furthermore, X stands for a finite dimensional Hilbert space, e. g., $X = \mathbb{R}$ or the finite element space $X = V^l$ on level l. The space of discrete abstract functions from Ξ to X is defined as the set of equivalence classes of functions which have the same values in $\{\xi_i\}$. A more precise definition necessitates the notation

$$\bar{v} := (v_0, v_1, ..., v_N)^{\mathsf{T}} := (v(\xi_0), v(\xi_1) ..., v(\xi_N))^{\mathsf{T}} \in X^{N+1}$$

which represents a vector of point values of a function $v : [0,T] \to X$ in Ξ. Using

$$Y_0 := Y_0(\Xi, X) := \{v : [0,T] \to X \,|\, \bar{v} = 0\},$$

an equivalence relation on $\{v : [0,T] \to X\}$ is defined by

$$u \sim v \quad :\Leftrightarrow \quad u - v \in Y_0 \quad \Leftrightarrow \quad \bar{u} = \bar{v},$$

for all $u, v : [0,T] \to X$. The equivalence classes form the quotient space

$$Y(\Xi, X) := \{v : [0,T] \to X\} \,/\, Y_0$$

which is called the 'space of discrete abstract functions' from Ξ to X.

Scalar product on $Y(\Xi, X)$ For two functions $u, v \in L^2(0,T;X)$ in time with values in X, the corresponding scalar product on $L^2(0,T;X)$ reads

$$(u,v)_{L^2(0,T;X)} = \int_0^T (u(t),\, v(t))_X \, dt. \tag{3.11}$$

This scalar product motivates the following choice of a scalar product on the space $Y(\Xi, X)$. For u, $v \in Y(\Xi, X)$, a discrete weighted scalar product is defined as

$$(u,v)_\Xi := \frac{1}{N} \sum_{n=0}^{N} (u_n, v_n)_X. \tag{3.12}$$

This scalar product induces the norm

$$||u||_\Xi = \sqrt{(u,u)_\Xi}, \qquad u \in Y(\Xi, X).$$

Association with X^{N+1} The spaces $Y(\Xi, X)$ and X^{N+1} can be associated, which is in the following expressed by

$$Y(\Xi, X) \cong X^{N+1}. \tag{3.13}$$

For every $v \in Y(\Xi, X)$, there is a uniquely associated counterpart $\bar{v} \in X^{N+1}$ by definition. On the other hand, for $\bar{w} \in X^{N+1}$, it is possible to construct $w \in Y(\Xi, X)$ with $w_n = w(\xi_n) = \bar{w}_n$ using, e. g., a piecewise linear interpolation in time of the values $\bar{w}_n$.

3.2 Remarks. a) The concept of discrete abstract functions is motivated by the 1D multigrid theory in [75]. The scalar product is defined similar to the weighted scalar product for finite difference functions in [75, Section 3.5] but adapted to functions on the time axis. It is used for the definition of weighted restrictions in time.

b) If $u, v \in Y(\Xi, X)$, equation (3.12) is the l_2 scalar product of the values in all $t \in \Xi$ (up to the term $\frac{1}{N}(u_0, v_0)$) and as such an approximation to the L^2 scalar product in (3.11) for equidistant time stepping.

c) With the above scalar product, $Y(\Xi, X)$ forms a Hilbert space. The dual space of $Y(\Xi, X)$ is denoted by $Y(\Xi, X)^*$ and $\langle \cdot, \cdot \rangle$ refers to the dual pairing. Since the space is discrete, $Y(\Xi, X)$ can be identified with its dual space via the Riesz representation theorem,

$$Y(\Xi, X)^* \cong Y(\Xi, X).$$

d) Similar to (3.13), the spaces $Y(\Xi, X)^*$ and $(X^*)^{N+1}$ can be associated,

$$Y(\Xi, X)^* \cong (X^*)^{N+1}, \tag{3.14}$$

by applying the association $X^* \cong X$ to every component in time,

$$Y(\Xi, X)^* \;\cong\; Y(\Xi, X) \;\cong\; X^{N+1} \;\cong\; (X^*)^{N+1}.$$

3.3 Example a) For a sequence $\tilde{\Xi}$ of $N+1$ discrete points in time on the time interval $[0,T]$,

$$\tilde{\Xi} := \{0 =: t_0 < t_1 < ... < t_N := T \,|\, 0 \le t_n \le T,\, n = 0, ..., N\},$$

the space $Y(\tilde{\Xi}, \mathbb{R})$ represents the point values of functions $v : [0,T] \to \mathbb{R}$ in the discrete points $\tilde{\Xi}$ in time. There is

$$v \in Y(\tilde{\Xi}, \mathbb{R}) \quad \Rightarrow \quad \bar{v} = (v(t_0), v(t_1), ..., v(t_N))^{\mathsf{T}} \in \mathbb{R}^{N+1}.$$

b) On level l, if the time discretisation is carried out with the implicit Euler scheme, the space W^l is characterised by the sequence T^l of points in time and the finite element space V^l, see Section 3.1.1. In this case, W^l has a representation in terms of discrete abstract functions due to (3.13),

$$Y(T^l, V^l) \;\cong\; (V^l)^{N_l+1} \;=\; W^l,$$

and for the space of right-hand side functions, there is

$$Y(T^l, V^l)^* \;\cong\; (V^{l,*})^{N_l+1} \;=\; (W^l)^\times$$

due to (3.14).

Linear mappings of discrete abstract functions For $M, N \in \mathbb{N}$,

$$\Xi_1 := \{\xi_0 < \xi_1 < ... < \xi_M \,|\, 0 \le \xi_n \le T,\, n = 0, ..., M\} \quad \text{and}$$
$$\Xi_2 := \{\eta_0 < \eta_1 < ... < \eta_N \,|\, 0 \le \eta_n \le T,\, n = 0, ..., N\}$$

define two ordered sequences of points in time. Considering a linear mapping

$$K : Y(\Xi_1, X) \to Y(\Xi_2, X),$$

the action $v = Ku$ for $u \in Y(\Xi_1, X)$ and $v \in Y(\Xi_2, X)$ is described by a matrix-vector multiplication. Due to

$$Y(\Xi_1, X) \cong X^{M+1} \quad \text{and} \quad Y(\Xi_2, X) \cong X^{N+1},$$

without loss of generality, the mapping K can be expressed as a matrix

$$K \in \mathbb{R}^{(N+1)\times(M+1)}$$

with entries K_{ij}, $i = 0, ..., N$, $j = 0, ..., M$. For $\bar{u} = (u_0, ..., u_M)$, the entries of $\bar{v} = (v_0, ..., v_N)$ are given by

$$v_i = \sum_{j=0}^{M} K_{ij} u_j, \qquad i = 0, ..., N.$$

3.4 Remark. In particular, the linear mapping only affects the components in time and is independent of the space X.

3.4.3. The implicit Euler case

Using the notations in Section 3.4.1, in the case of the implicit Euler timestepping scheme, the prolongation $w^{l+1} := P^l w \in W^{l+1}$ of a vector $w \in W^l$ is formulated as

$$\begin{aligned} &1.) \quad \tilde{w} := (P_{\text{space}} w_0, ..., P_{\text{space}} w_N) && \in W^{l+1,l} \\ &2.) \quad w^{l+1} := P_{\text{time}} \tilde{w} && \in W^{l+1,l+1} = W^{l+1} \end{aligned}$$

with a time prolongation operator $P_{\text{time}} : W^{l+1,l} \to W^{l+1,l+1}$ representing an interpolation of the solutions in time. This operator can be formulated by means of discrete abstract functions: With $X := V^{l+1}$, due to (3.13), the spaces

$$W^{l+1,l} \cong Y(T^l, X) \quad \text{and} \quad W^{l+1,l+1} \cong Y(T^{l+1}, X) \tag{3.15}$$

can be identified.

Prolongation in time A prolongation $P_{\text{time}} = P^l_{\text{time}} : Y(T^l, X) \to Y(T^{l+1}, X)$ that computes intermediate solutions as arithmetic means is specified by the matrix

$$P_{\text{time}} = \begin{pmatrix} 1 & & \\ 1/2 & 1/2 & \\ & 1 & \\ & 1/2 & 1/2 \\ & & \ddots \\ & & & 1 \end{pmatrix} \in \mathbb{R}^{(2N+1)\times(N+1)},$$

see also [76]. Figure 3.3 illustrates the action of P_{time} onto a space-time vector. New values of time midpoints are obtained by interpolating the neighbouring solutions. Since the interpolation is linear, piecewise linear functions in time are interpolated exactly.

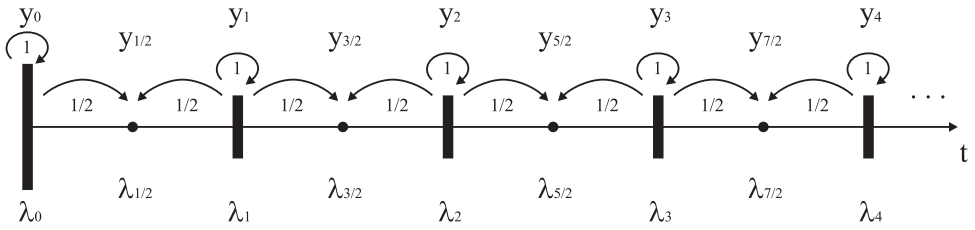

Figure 3.3: Weights in the prolongation for the implicit Euler. Primal solution y_n (top) and dual solution λ_n (bottom), with $w_n = (y_n, \lambda_n)$, $n = 0, 1, 2, ...$

Restriction in time Restriction operators are naturally defined as adjoint of prolongation operators, cf. [75, Section 3.5]. Here, a restriction is first applied in time, then in space. For the restriction in time, using (3.14), the spaces of the right-hand side functions are associated with duals of discrete abstract spaces,

$$(W^{l+1,l})^\times \cong Y(T^l, X)^* \quad \text{and} \quad (W^{l+1,l+1})^\times \cong Y(T^{l+1}, X)^*. \tag{3.16}$$

A restriction in time $R_{\text{time}} = R^l_{\text{time}} : Y(T^{l+1}, X)^* \to Y(T^l, X)^*$ follows as the adjoint of the prolongation via the dual pairing. For arbitrary $v' \in Y(T^{l+1}, X)^*$, its Riesz representative $v \in Y(T^{l+1}, X)$ and $w \in Y(T^l, X)$, considering

$$\begin{aligned} & \langle R_{\text{time}} v', w \rangle = \langle v, P_{\text{time}} w \rangle \\ \Leftrightarrow \quad & (R_{\text{time}} v, w)_{T^l} = (v, P_{\text{time}} w)_{T^{l+1}} \\ \Leftrightarrow \quad & \frac{1}{N} \sum_{n=0}^{N} ((R_{\text{time}} v)_n, w_n)_X = \frac{1}{2N} \sum_{n=0}^{2N} (v_n, (P_{\text{time}} w)_n)_X \end{aligned}$$

motivates the choice

$$R_{\text{time}} = \frac{1}{2} (P_{\text{time}})^{\mathsf{T}} \tag{3.17}$$

as expected from a finite difference restriction, see e. g. [76] and in particular [75, Section 2.3]. The restriction $R : (W^{l+1})^\times \to (W^l)^\times$ with $d^l := R^l d^{l+1}$ follows in the form

$$\begin{aligned} 1.) \quad & \tilde{d} := R_{\text{time}} d^{l+1} && \in (W^{l+1,l})^\times \\ 2.) \quad & d^l := (R_{\text{space}} \tilde{d}_0, ..., R_{\text{space}} \tilde{d}_N) && \in (W^{l,l})^\times = (W^l)^\times. \end{aligned}$$

3.4.4. The general θ-scheme case

In the case of a general θ-scheme, proper prolongation and restriction operators are slightly more involved. The location of the degrees of freedom in time should be respected. The crucial point is that the right-hand side functions are defined in different points in time than the solution vectors, and that the components in a solution vector are defined in different points in time according to their role: For the Stokes and Navier–Stokes equations as an example, y and ξ are located according to T^l, while λ and p are located according to T^l_θ. For the right-hand side, the roles are exactly opposite. This has immediate consequences on the choice of the prolongation and restriction operators.

Decomposition of solutions For space level $l \in \mathbb{N}$ and time level $m \in \mathbb{N}$, the space

$$W^{l,m} = (V^l)^{N+1}$$

is considered, with $N = N_m$ denoting the number of time intervals. Based on this space, the space-time system reads, cf. equation (3.5),

$$G(w)w = f$$

for some $w = (w_0, ..., w_N) \in W^{l,m}$, the space-time operator $G = G^{lm}$ and some right-hand side function $f = (f_0, ..., f_N) \in (W^{l,m})^\times$.

a) Concentrating on one element w_n of the solution, $n \in \{0, ..., N\}$, some components in w_n are located in time according to T^l, while others are located in time according to T^l_θ. Without loss of generality, it is possible (eventually after resorting) to assume a decomposition

$$w_n = (w_n^p, w_n^d)$$

with w_n^p being located in time according to T^l and w_n^d located in time according to T^l_θ.

b) Resorting the components in f_n in the same way as in a) allows to assume a decomposition

$$f_n = (f_n^p, f_n^d)$$

with f_n^p being the right-hand side corresponding to w_n^p and f_n^d being the right-hand side to w_n^d. It is important that due to the timestepping scheme, the roles in time change: f_n^p can be interpreted as being located in time according to T^l_θ while f_n^d corresponds to T^l.

c) The decomposition in a) can also be applied to the underlying finite element spaces. It is possible to assume that V^l is decomposed as follows,

$$V^l = V_p^l \times V_d^l,$$

such that $w_n = (w_n^p, w_n^d) \in V_p^l \times V_d^l$ and $f_n = (f_n^p, f_n^d) \in V_p^{l,*} \times V_d^{l,*} \cong V^{l,*}$.

d) With the decomposition in a) and the spaces in c), appropriate spaces on the space-time cylinder can be defined. The setting

$$W_p^{l,m} := (V_p^l)^{N_m+1}, \qquad W_d^{l,m} := (V_d^l)^{N_m+1}$$

suggests a decomposition of the solution w into

$$w_p := (w_0^p, ..., w_N^p) \in W_p^{l,m},$$
$$w_d := (w_0^d, ..., w_N^d) \in W_d^{l,m}.$$

Similarly, for the right-hand side,

$$(W_p^{l,m})^\times := (V_p^{l,*})^{N_m+1}, \qquad (W_d^{l,m})^\times := (V_d^{l,*})^{N_m+1}$$

denote the right-hand side spaces corresponding to $W_p^{l,m}$ and $W_d^{l,m}$ and suggests a decomposition of f in the form

$$f_p := (f_0^p, ..., f_N^p) \in (W_p^{l,m})^\times,$$
$$f_d := (f_0^d, ..., f_N^d) \in (W_d^{l,m})^\times.$$

3.5 Example. As an example, the Stokes or Navier–Stokes equations with solution w and right-hand side f is considered. Each entry in w and f has the form

$$w_n = (y_n, \lambda_{n-1+\theta}, p_{n-1+\theta}, \xi_n),$$
$$f_n = (f^y_{n-1+\theta}, f^\lambda_n, f^p_n, f^\xi_{n-1+\theta}).$$

The index at each component indicates the location of the component in time. For example, y_n and ξ_n are interpreted to be located according to T^l while $\lambda_{n-1+\theta}$ and $p_{n-1+\theta}$ are located according to T^l_θ. Without loss of generality, sorting the components according to their location in time, the vectors are rewritten in the form

$$w_n = (\underbrace{y_n, \xi_n}_{w_n^p}, \underbrace{\lambda_{n-1+\theta}, p_{n-1+\theta}}_{w_n^d}), \qquad f_n = (\underbrace{f^y_{n-1+\theta}, f^\xi_{n-1+\theta}}_{f_n^p}, \underbrace{f^\lambda_n, f^p_n}_{f_n^d}),$$

and the subvectors

$$w^p := (\underbrace{y_0, \xi_0}_{w_0^p}, ..., \underbrace{y_N, \xi_N}_{w_N^p}), \qquad w^d := (\underbrace{\lambda_{-1+\theta}, p_{-1+\theta}}_{w_0^d}, ..., \underbrace{\lambda_{N-1+\theta}, p_{N-1+\theta}}_{w_N^d})$$
$$f^p := (\underbrace{f^y_{-1+\theta}, f^\xi_{-1+\theta}}_{f_0^p}, ..., \underbrace{f^y_{N-1+\theta}, f^\xi_{N-1+\theta}}_{f_N^p}), \qquad f^d := (\underbrace{f^\lambda_0, f^p_0}_{f_0^d}, ..., \underbrace{f^\lambda_N, f^p_N}_{f_N^d})$$

are formulated.

The prolongation Using the notations in Section 3.4.1, $w \in W^l$ denotes be a solution vector at level l. The corresponding w^p contains solutions located at T^l, so they can be processed with the same prolongation as in the case of the implicit Euler scheme; this prolongation is second order in time (it realises a linear interpolation in each timestep) and can therefore be used for the implicit Euler and the general θ-scheme. Alike, w^d is located in-between the endpoints of the time intervals, and hence, a different interpolation matrix is required. The prolongation $w^{l+1} := P^l w \in W^{l+1}$ is defined as

$$\begin{aligned} &1.) \quad \tilde{w} := (\tilde{w}_1, ..., \tilde{w}_N) := (P_{\text{space}} w_0, ..., P_{\text{space}} w_N) && \in W^{l+1,l} \\ &2.) \quad (w_p^{l+1}, w_d^{l+1}) := (P^p_{\text{time}} \tilde{w}_p,\ P^d_{\text{time}} \tilde{w}_d) && \in (W_p^{l+1,l+1}, W_d^{l+1,l+1}) \end{aligned}$$

with some time prolongation operators

$$P^p_{\text{time}} = P^{l,p}_{\text{time}} : W_p^{l+1,l} \to W_p^{l+1,l+1} \quad \text{and}$$
$$P^d_{\text{time}} = P^{l,d}_{\text{time}} : W_d^{l+1,l} \to W_d^{l+1,l+1}.$$

Using $X_p := V_p^{l+1}$, $X_d := V_d^{l+1}$ and (3.13), the spaces

$$\begin{aligned} W_p^{l+1,l} &\cong Y(T^l, X_p), & W_p^{l+1,l+1} &\cong Y(T^{l+1}, X_p), \\ W_d^{l+1,l} &\cong Y(T_\theta^l, X_d), & W_d^{l+1,l+1} &\cong Y(T_\theta^{l+1}, X_d), \end{aligned}$$

are associated and above operators are interpreted in terms of discrete abstract functions,

$$P_{\text{time}}^p : Y(T^l, X_p) \to Y(T^{l+1}, X_p) \quad \text{and} \tag{3.18}$$

$$P_{\text{time}}^d : Y(T_\theta^l, X_d) \to Y(T_\theta^{l+1}, X_d). \tag{3.19}$$

The operators can be realised, e. g., as linear interpolations in time. In matrix form, these read

$$P_{\text{time}}^p = \begin{pmatrix} 1 & & & \\ 1/2 & 1/2 & & \\ & 1 & & \\ & 1/2 & 1/2 & \\ & & \ddots & \\ & & & 1 \end{pmatrix}, \quad P_{\text{time}}^d = \begin{pmatrix} 1 & & & & & \\ 0 & 1+\frac{1}{2}\theta & -\frac{1}{2}\theta & & & \\ & \frac{1}{2}+\frac{1}{2}\theta & \frac{1}{2}-\frac{1}{2}\theta & & & \\ & \frac{1}{2}\theta & 1-\frac{1}{2}\theta & & & \\ & & \frac{1}{2}+\frac{1}{2}\theta & \frac{1}{2}-\frac{1}{2}\theta & & \\ & & \frac{1}{2}\theta & 1-\frac{1}{2}\theta & & \\ & & \ddots & & & \\ & & & \frac{1}{2}+\frac{1}{2}\theta & \frac{1}{2}-\frac{1}{2}\theta & \\ & & & \frac{1}{2}\theta & 1-\frac{1}{2}\theta & \\ & & & \frac{1}{2}\theta-\frac{1}{2} & \frac{3}{2}-\frac{1}{2}\theta & \end{pmatrix}.$$

Figure 3.4 depicts the weights for the Crank–Nicolson case ($\theta = 1/2$).

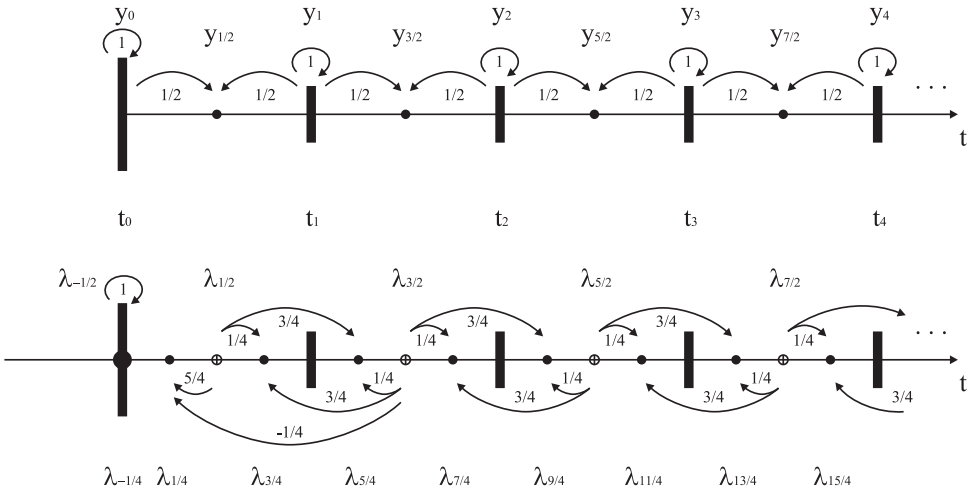

Figure 3.4: Weights in the prolongation of the primal (top) and dual (bottom) space in the case of the Crank–Nicolson scheme at the beginning of the time interval.

3.6 Remark. The different weights for the first and last solution in time (the first two rows and the last row of P_{time}^d) result from an extrapolation and constant prolongation in time: The last dual solution on the fine mesh is located *behind* the last dual solution on the coarse mesh and must therefore be extrapolated. The first dual solution on the other hand has no physical meaning as this solution is not used as a right-hand side for the primal equation. To prevent it from influencing the dual solution inside the temporal domain, it

is treated separately with a constant prolongation. As this does not influence the solutions in the other timesteps, this treatment does not destroy the global approximation order of the prolongation.

The restriction In the following, a restriction operator is defined which maps the space of right-hand sides to the coarser level,

$$R^l : (W^{l+1})^\times \to (W^l)^\times.$$

Similar to the implicit Euler case, this operator is constructed as the 'adjoint' of the prolongation, realised in two steps. First, a restriction in time

$$R_{\text{time}} : (W^{l+1})^\times = (W^{l+1,l+1})^\times \to (W^{l+1,l})^\times$$

is needed. Similar to the case of the prolongation, this can be decomposed into two restrictions, one for the solution components located in time according to T^l and one corresponding to the components located in time according to T^l_θ,

$$\begin{aligned} R^p_{\text{time}} &: (W_p^{l+1,l+1})^\times \to (W_p^{l+1,l})^\times, \\ R^d_{\text{time}} &: (W_d^{l+1,l+1})^\times \to (W_d^{l+1,l})^\times. \end{aligned}$$

On the other hand, a restriction in space must be carried out as the adjoint of the prolongation in space; the operator

$$R_{\text{space}} : V^{l+1,*} \to V^{l,*}$$

from the finite element space provides this task. Having defined these operators, a possible choice for the restriction $d^l := R^l d^{l+1}$ of a defect $d^{l+1} = (d_p, d_d) \in (W^{l+1})^\times$ reads

$$\begin{aligned} &1.) \quad (\tilde{d}_p, \tilde{d}_d) := (R^p_{\text{time}} d_p,\ R^d_{\text{time}} d_d) \\ &2.) \quad d^l := (R_{\text{space}}(\tilde{d}_0), ..., R_{\text{space}}(\tilde{d}_N)) \end{aligned}$$

with an auxiliary vector $\tilde{d} \in (W^{l+1,l})^\times$.

What remains is the definition of suitable restriction operators R^p_{time} and R^d_{time} in time. Intuitively, these would be created by transposing the corresponding prolongation matrices,

$$R^p_{\text{time}} = \frac{1}{2}(P^p_{\text{time}})^\mathsf{T}, \qquad R^d_{\text{time}} = \frac{1}{2}(P^d_{\text{time}})^\mathsf{T}, \tag{3.20}$$

but a closer look on the analysis reveals that this choice is *not* the optimal one: P^p_{time} and P^d_{time} have to be interchanged.

Similar to the prolongation, all spaces are interpreted in terms of discrete abstract functions. It is important that the right-hand side and defect vectors have to be considered in a way 'opposite' to the solution vectors used in the prolongation. With $X_p := V_p^{l+1}$ and $X_d := V_d^{l+1}$, using (3.14), the spaces

$$\begin{aligned} (W_p^{l+1,l})^\times &\cong Y(T^l_\theta, X_p)^*, & (W_p^{l+1,l+1})^\times &\cong Y(T^{l+1}_\theta, X_p)^*, \\ (W_d^{l+1,l})^\times &\cong Y(T^l, X_d)^*, & (W_d^{l+1,l+1})^\times &\cong Y(T^{l+1}, X_d)^*, \end{aligned}$$

are associated and the restriction operators are interpreted as

$$\begin{aligned} &R^p_{\text{time}} : Y(T^{l+1}_\theta, X_p)^* \to Y(T^l_\theta, X_p)^* \quad \text{and} \\ &R^d_{\text{time}} : Y(T^{l+1}, X_d)^* \to Y(T^l, X_d)^*. \end{aligned}$$

A possible restriction in time, R^p_{time}, follows as the adjoint of the prolongation via the dual pairing. For this purpose, a function $v' \in Y(T^{l+1}_\theta, X_p)^*$, its Riesz representative $v \in Y(T^{l+1}_\theta, X_p)$ and $w \in Y(T^l_\theta, X_p)$ are used. According to (3.19) and Remark 3.4, the prolongation in the space $Y(T^l_\theta, X_p)$ has the same matrix representation as P^d_{time}, thus the restriction is defined according to

$$\begin{aligned} & \langle R^p_{\text{time}} v', w\rangle = \langle v, P^d_{\text{time}} w\rangle \\ \Leftrightarrow \quad & (R^p_{\text{time}} v, w)_{T^l_\theta} = (v, P^d_{\text{time}} w)_{T^{l+1}_\theta} \\ \Leftrightarrow \quad & \frac{1}{N}\sum_{n=0}^{N} ((R^p_{\text{time}} v)_n, w_n)_{X_p} = \frac{1}{2N}\sum_{n=0}^{2N} \Big(v_n, (P^d_{\text{time}} w)_n\Big)_{X_p}. \end{aligned}$$

For R^d_{time}, the corresponding formulas are similar and involve (3.18). In conclusion, the above considerations suggest the following choice for the restriction in time, which exchanges the roles of the prolongation matrices if they are used for the definition of the restriction,

$$R^p_{\text{time}} = \frac{1}{2}(P^d_{\text{time}})^{\mathsf{T}}, \qquad R^d_{\text{time}} = \frac{1}{2}(P^p_{\text{time}})^{\mathsf{T}}. \tag{3.21}$$

3.7 Example. For the Navier–Stokes equations, a solution $w \in W^l$ at level l can be decomposed into the components

$$w \mapsto (w_y, w_\lambda, w_p, w_\xi)$$

with

$$\begin{aligned} w_y &= (y_0, ..., y_N), & w_\lambda &= (\lambda_{-1+\theta}, ..., \lambda_{N-1+\theta}), \\ w_p &= (p_{-1+\theta}, ..., p_{N-1+\theta}), & w_\xi &= (\xi_0, ..., \xi_N). \end{aligned}$$

The prolongation $w^{l+1} = P^l w$ is formulated in two steps,

$$\begin{aligned} &1.) \quad \tilde{w} := (\tilde{w}_0, ..., \tilde{w}_N) := (P_{\text{space}}(w_0), ..., P_{\text{space}}(w_N)) \\ &2.) \quad (w^{l+1}_y, w^{l+1}_\lambda, w^{l+1}_p, w^{l+1}_\xi) := (P^p_{\text{time}}\tilde{w}_y,\ P^d_{\text{time}}\tilde{w}_\lambda, P^d_{\text{time}}\tilde{w}_p,\ P^p_{\text{time}}\tilde{w}_\xi) \end{aligned}$$

with an auxiliary vector $\tilde{w} = (\tilde{w}_0, ..., \tilde{w}_N)$. In the above notation, the vector $\tilde{w}$ has an associated decomposition $(\tilde{w}_y, \tilde{w}_\lambda, \tilde{w}_p, \tilde{w}_\xi)$ and the vector w^{l+1} an associated decomposition $(w^{l+1}_y, w^{l+1}_\lambda, w^{l+1}_p, w^{l+1}_\xi)$ into the velocity/pressure components as above.

A defect vector $d^{l+1} \in (W^{l+1})^\times$ at level $l+1$ can be decomposed into the components

$$d^{l+1} \mapsto (d_y, d_\lambda, d_p, d_\xi)$$

with

$$\begin{aligned} d_y &= (d^y_{-1+\theta}, ..., d^y_{2N-1+\theta}), & d_\lambda &= (d^\lambda_0, ..., d^\lambda_{2N}) \\ d_p &= (d^p_0, ..., d^p_{2N}), & d_\xi &= (d^\xi_{-1+\theta}, ..., d^\xi_{2N-1+\theta}). \end{aligned}$$

The restriction $d^l = R^l d^{l+1}$ reads

$$\begin{aligned} &1.) \quad (\tilde{d}_y, \tilde{d}_\lambda, \tilde{d}_p, \tilde{d}_\xi) := \frac{1}{2}\big((P^d_{\text{time}})^{\mathsf{T}} d_y,\ (P^p_{\text{time}})^{\mathsf{T}} d_\lambda,\ (P^p_{\text{time}})^{\mathsf{T}} d_p,\ (P^d_{\text{time}})^{\mathsf{T}} d_\xi\big) \\ &2.) \quad d^l := (R_{\text{space}}(\tilde{d}_0), ..., R_{\text{space}}(\tilde{d}_N)) \end{aligned}$$

with an auxiliary vector $\tilde{d} = (\tilde{d}_0, ..., \tilde{d}_N)$ and associated decomposition $(\tilde{d}_y, \tilde{d}_\lambda, \tilde{d}_p, \tilde{d}_\xi)$ as above.

3.4.5. Coarse grid preconditioning operators

The multigrid algorithm needs a hierarchy of operators C^1,...,C^{NLMAX} corresponding to the levels $l = 1, ..., \mathrm{NLMAX}$. At the finest level, the equation

$$C(w_i^\sigma)g_i \;=\; d_i$$

has to be solved, see (3.10), on coarser levels systems of the form

$$C^l w \;=\; f, \qquad w \in W^l,\, f \in (W^l)^\times.$$

With the underlying hierarchy of space-time meshes, $C(w_i^\sigma)$ is interpreted as fine grid operator of a hierarchy of operators $C^1, ..., C^{\mathrm{NLMAX}}$ with $C^{\mathrm{NLMAX}} = C(w_i^\sigma)$.

The linear case In case of the heat equation or the Stokes equations, the state equation is linear, i. e.,

$$C(w_i^\sigma) = C = C^{\mathrm{NLMAX}}$$

independent of w_i^σ, and chosen as

$$C = G^\sigma = G^{\mathrm{NLMAX}}.$$

Due to equation (3.6), the set $G^1, ..., G^{\mathrm{NLMAX}-1}$ forms a hierarchy of operators approximating C on level $l = 1, ..., \mathrm{NLMAX} - 1$. Correspondingly, the preconditioning matrices for the multigrid algorithm are canonically defined by

$$C^l := G^l, \quad l = 1, ..., \mathrm{NLMAX}.$$

The nonlinear case In the nonlinear case, the operator $C := C^{\mathrm{NLMAX}} = C(w_i^\sigma)$ depends on the solution w_i^σ, and w_i^σ is only given at the finest level. Therefore, a hierarchy of preconditioning operators $C^1, ..., C^{\mathrm{NLMAX}-1}$ approximating C is not naturally given; it has to be created it via an appropriate approximation.

On level NLMAX, there is by definition

$$C^{\mathrm{NLMAX}} = G^{\mathrm{NLMAX}}(w_i^\sigma) = G^\sigma(w_i^\sigma) \quad \text{or} \quad C^{\mathrm{NLMAX}} = F^{\mathrm{NLMAX}}(w_i^\sigma) = F^\sigma(w_i^\sigma)$$

for the fixed point or the Newton algorithm, respectively. On level $l \in \{1, ..., \mathrm{NLMAX}-1\}$, an approximation $w_i^l \in W^l$ to w_i^σ is needed. The problem hierarchy (3.6) suggests that

$$G^l(w_i^l) \approx G^{\mathrm{NLMAX}}(w_i^\sigma).$$

The Fréchet derivative of $w \mapsto G^l(w)w$ in $w = w_i^l$ is given by $F^l(w_i^l)$, and thus,

$$F^l(w_i^l) \approx F^{\mathrm{NLMAX}}(w_i^\sigma).$$

Therefore, a canonical choice for the preconditioner C^l is given by

$$C^l := G^l(w_i^l) \quad \text{or} \quad C^l := F^l(w_i^l) \tag{3.22}$$

for the fixed point or the Newton algorithm, respectively. What remains is a definition of the approximation w_i^l to w_i^σ, which is obtained here by a recursive interpolation of w_i^σ to the coarser levels. Similar to a prolongation or restriction operator, an appropriate

interpolation operator is formulated using a combination of finite difference interpolation in time and finite element interpolation in space.

For convenience, the subscript i is dropped in the following, i.e., $w^l = w_i^l$. With $w := w^{l+1} \in W^{l+1}$ the current 'fine' grid solution on level $l+1$ is denoted, $w^{\mathrm{NLMAX}} = w_i^\sigma$. In space, there is a natural finite element interpolation available,

$$I_{\mathrm{space}} = I^l_{\mathrm{space}} : V^{l+1} \to V^l.$$

This is realised, e.g., by a standard L^2-projection in space or a simple interpolation operator that evaluates a finite element function on a lower level. An interpolation operator $I^l : W^{l+1} \to W^l$ on the space-time cylinder with $w^l = I^l w \in W^l$ is formulated in two steps,

$$\begin{aligned} &1.) \quad \tilde{w} = (\tilde{w}_0, ..., \tilde{w}_N) := I_{\mathrm{time}} w \qquad \in W^{l+1,l}, \\ &2.) \quad w^l := (I_{\mathrm{space}}\tilde{w}_0, ..., I_{\mathrm{space}}\tilde{w}_N) \quad \in W^{l,l} = W^l, \end{aligned}$$

with a time interpolation operator $I_{\mathrm{time}} : W^{l+1} \to W^{l+1,l}$. Defining $X := V^{l+1}$ and using (3.13) to identify

$$W^{l+1,l} \cong Y(T^l, X) \quad \text{and} \quad W^{l+1,l+1} \cong Y(T^{l+1}, X), \tag{3.23}$$

the time interpolation can be interpreted in terms of discrete abstract functions by

$$I_{\mathrm{time}} = I^l_{\mathrm{time}} : Y(T^{l+1}, X) \to Y(T^l, X).$$

A possible interpolation is given by the matrix

$$P_{\mathrm{time}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & 1 & ... & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & 0 & 0 & ... & 1 \end{pmatrix} \in \mathbb{R}^{(N+1)\times(2N+1)}.$$

This realises a constant interpolation in time: The solutions at the endpoints of the time intervals on the coarser time mesh are taken from the corresponding solutions of the time fine mesh. All other solutions of the time fine mesh are ignored, see Figure 3.5.

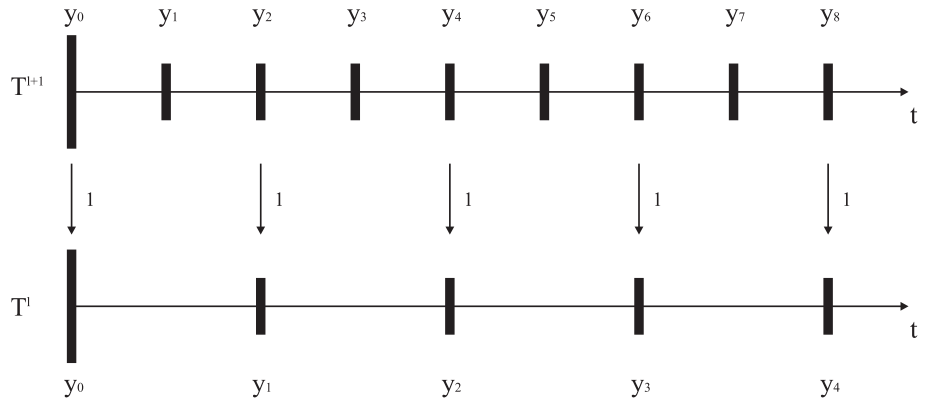

Figure 3.5: Constant interpolation of solutions onto a time coarse mesh.

3.8 Remarks. The above constant time interpolation can be seen as the 'best' choice in the case of the implicit Euler scheme since it exactly reproduces the solution of the time fine grid onto the time coarse grid. As long as there are no constraints used, this interpolation can also be used for the Crank–Nicolson scheme: The interpolation can ignore the dual equation. Without constraints, the nonlinearity is completely specified by the solution y in the state equation, there are no nonlinear terms involving the dual variable λ (not to mention the primal and dual pressure). Since the layout of the timesteps of y is the same for both time discretisation schemes, the constant time interpolation can also be used in both cases.

The situation slightly changes if additional constraints are used for the control, see Section 4.2 on page 87ff. The state equation depends in a nonlinear way on the dual variable λ, and thus, λ has to be interpolated to the lower level as well. In the θ-scheme case, the time interpolation for the dual variable would have to be modified, similar to the restriction. However, numerical tests in later sections neglect this fact. Always using a constant time interpolation for both, primal and dual variables, no considerable impact on the convergence speed is observed, see Section 7.3 on page 144ff.

3.5. Smoothing operators and the coarse grid solver

By definition, the global space-time preconditioner C^l on level l has the structure of a three-band block matrix (see Section 2.7 on page 42ff) with $N+1$ block columns and rows, $N = N_l$ denoting the number of time intervals on level l. This special matrix structure allows to define iterative smoothing operators based on the defect correction approach. In the following, a couple of basic block smoothing algorithms are introduced: FBJacSmoother, FBGsSmoother and FBSimSmoother. They realise a block Jacobi iteration, a block Gauß–Seidel iteration and a 'simulation' solver that carries out (linear) forward-backward simulations (with a fixed nonlinearity). These algorithms are modifications of common block algorithms (cf. [10]), adapted to the space-time case.

3.5.1. Standard block smoothers

For a proper formulation, the (linear) system on a space-time level l is denoted by

$$C^l w = f \tag{3.24}$$

for some a right-hand side vector f and a space-time vector w. Let w be an approximate to a solution w^* of (3.24) which should be smoothed. The matrix C^l can be decomposed into blocks corresponding to the time intervals as follows:

$$C^l x =: \left(\begin{array}{c|c|c|c} C_{0,0} & C_{0,1} & & \\ \hline C_{1,0} & C_{1,1} & C_{1,2} & \\ \hline & \ddots & \ddots & \ddots \\ \hline & & C_{N,N-1} & C_{N,N} \end{array}\right)$$

$$L := \left(\begin{array}{c|c|c|c} 0 & & & \\ \hline C_{1,0} & 0 & & \\ \hline & \ddots & \ddots & \\ \hline & & C_{N,N-1} & 0 \end{array}\right), \qquad R := \left(\begin{array}{c|c|c|c} 0 & C_{0,1} & & \\ \hline & 0 & C_{1,2} & \\ \hline & & \ddots & \ddots \\ \hline & & & 0 \end{array}\right)$$

$$D := \begin{pmatrix} C_{0,0} & & & \\ & C_{1,1} & & \\ & & \ddots & \\ & & & C_{N,N} \end{pmatrix}.$$

For a damping parameter $\omega \in \mathbb{R}$, the special matrix structure suggests the use of a damped Block-Jacobi method, see Algorithm 3.3.

Similar to a Block-Jacobi algorithm, it is possible to design a damped forward-backward block Gauß–Seidel algorithm for smoothing, see Algorithm 3.4. This algorithm decomposes into a forward-loop and a backward-loop. In contrast to Block-Jacobi, it exploits basic coupling in time without significant additional costs.

3.9 Remark. With some minor changes, the FBGSSmoother algorithm can also be modified to a forward-backward block SOR algorithm (see Algorithm 3.5 with $\omega_{\mathrm{SOR}} \in \mathbb{R}$ denoting a relaxation parameter), but this algorithm will not be in the focus of the numerical experiments in later chapters. It is noted that in line 3 of the algorithm, x_1 appears on the left as well as on the right side, but on the right side multiplied with L. This has to be interpreted as a forward sweep in the typical Gauß–Seidel like manner: The calculation of a new entry in x_1 depends on the previously calculated values of the new iterate. The same holds for line 4 which has to be interpreted as a backward sweep for x_2.

Reduction to linear systems in space The expensive part in all the algorithms is the application of the operator D^{-1}. Since D is a block diagonal system, this step can equivalently be expressed as a sequence of linear systems in space,

$$\begin{aligned} w = D^{-1} f \quad &\Leftrightarrow \quad Dw = f \\ &\Leftrightarrow \quad D_n w_n = f_n \qquad n = 0, ..., N_l, \\ &\Leftrightarrow \quad C_{n,n} w_n = f_n \qquad n = 0, ..., N_l, \end{aligned} \tag{3.25}$$

for $w \in W^l$, $f \in (W^l)^\times$. The diagonal matrix $D_n = C_{n,n}$ represents the fully coupled primal-dual system in each time interval, and (3.25) is a sequence of linear systems in space. Similar to a linear system in each timestep of a simulation, it can be solved with a multigrid approach in space. A spatial multigrid solver is expected to have linear complexity and provides the final key for the efficiency of the whole algorithm. However, the crucial point in the definition is again the smoother. In Section 3.6, this topic will be discussed in detail.

3.5.2. Forward-Backward simulation smoother

FBJacSmoother and FBGsSmoother treat the primal and dual solution in a coupled way. As an alternative, the solution components can be decoupled and processed by a (linear) forward simulation for the primal solution, followed by a (linear) backward simulation for the dual solution. This type of algorithm, which is called 'forward-backward simulation algorithm' here, abbreviated FBSimSmoother, is rather natural and is motivated by a primal-dual simulation strategy which was used also by other authors before (see, e. g., [71, 152]). It is expected to be a compromise between speed and stability: Fully coupled systems in space are avoided, so the computation of each timestep is faster. On the other hand, due to the reduced coupling, the convergence speed of the overall solver is expected to deteriorate in comparison to FBGsSmoother.

Algorithm 3.3 Space-time Block-Jacobi smoother

Predefined constant: $\omega > 0$: damping parameter

function FBJacSmoother (C^l,w,f,NSM)
  for istep = 1 to NSM **do**
    $w \leftarrow w + \omega D^{-1}(f - C^l w)$
  end for
  return w
end function

Algorithm 3.4 Forward-Backward Block-GS smoother

Predefined constant: $\omega > 0$: damping parameter

function FBGsSmoother (C^l,w,f,NSM)
  for istep = 1 to NSM **do**
    $x_1 \leftarrow w + (L + D)^{-1}(f - C^l w) \;=\; w + D^{-1}(f - (D + R)w - Lx_1)$
    $x_2 \leftarrow x_1 + (R + D)^{-1}(f - C^l x_1) \;=\; x_1 + D^{-1}(f - (D + L)x_1 - Rx_2)$
    $w \leftarrow (1 - \omega)w + \omega x_2$
  end for
  return w
end function

Algorithm 3.5 Forward-Backward Block-SOR smoother

Predefined constant: $\omega_{\mathrm{SOR}} \in \mathbb{R}$: relaxation parameter
Predefined constant: $\omega > 0$: damping parameter

function FBSorSmoother (C^l,w,f,NSM)
  for istep = 1 to NSM **do**
    $x_1 \leftarrow w + \omega_{\mathrm{SOR}} D^{-1}(f - (D + R)w - Lx_1)$
    $x_2 \leftarrow x_1 + \omega_{\mathrm{SOR}} D^{-1}(f - (D + L)x_1 - Rx_2)$
    $w \leftarrow (1 - \omega)w + \omega x_2$
  end for
  return w
end function

To formulate this smoother, a modification of D is used. The matrix

$$D^{\mathrm{dec}} := \left(\begin{array}{c|c|c|c} C_{0,0}^{\mathrm{dec}} & & & \\ \hline & C_{1,1}^{\mathrm{dec}} & & \\ \hline & & \ddots & \\ \hline & & & C_{N,N}^{\mathrm{dec}} \end{array}\right)$$

defines the decoupled part of this matrix D. The diagonal blocks $C_{n,n}^{\mathrm{dec}}$ stem from $C_{n,n}$ by removing the coupling matrices between the primal and the dual part. For example, in case of the heat equation discretised with the implicit Euler scheme, this means

$$C_{n,n} = \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{A}_h & \frac{\mathcal{I}_h}{\alpha} \\ -\mathcal{I}_h & \frac{\mathcal{I}_h}{k} + \mathcal{A} \end{pmatrix} \quad \Rightarrow \quad C_{n,n}^{\mathrm{dec}} = \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{A}_h & 0 \\ 0 & \frac{\mathcal{I}_h}{k} + \mathcal{A} \end{pmatrix},$$

and in the case of the Navier–Stokes equations, with $C = G(w^l)$,

$$C_{n,n} = \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{C}_{n,h} & \frac{\mathcal{I}_h}{\alpha} & \mathcal{G}_h & 0 \\ -\mathcal{I}_h & \frac{\mathcal{I}_h}{k} + \mathcal{N}_{n,h}^* & 0 & \mathcal{G}_h \\ \mathcal{D}_h & 0 & 0 & 0 \\ 0 & \mathcal{D}_h & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \quad C_{n,n}^{\mathrm{dec}} = \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{C}_{n,h} & 0 & \mathcal{G}_h & 0 \\ 0 & \frac{\mathcal{I}_h}{k} + \mathcal{N}_{n,h}^* & 0 & \mathcal{G}_h \\ \mathcal{D}_h & 0 & 0 & 0 \\ 0 & \mathcal{D}_h & 0 & 0 \end{pmatrix}.$$

Thus, the problem of solving a linear system with $C_{n,n}$ for each component in time decomposes into two smaller, independent linear systems, one for the forward and one for the backward equation.

Algorithm 3.6 is similar to FBGsSmoother but uses the matrix D^{dec} instead of D to formulate the smoothing iteration. The algorithm assumes a decomposition $x = (x_{\mathrm{primal}}, x_{\mathrm{dual}})$ of a space-time vector $x \in W^l$ into a primal and dual part. In a first step, line 4 of the algorithm, only the primal part is updated by a forward simulation using the dual part as right-hand side. In the second step, line 5 of the algorithm, only the dual part is updated using the primal part as right-hand side.

Algorithm 3.6 Forward-Backward simulation smoother

Predefined constant: $\omega > 0$: damping parameter
Notation: $x = (x_{\mathrm{primal}}, x_{\mathrm{dual}})$: decomposition into primal/dual part

1: **function** FBSimSmoother (C^l,w,f,NSM)
2: **for** istep = 1 to NSM **do**
3: $x \leftarrow w$
4: $x_{\mathrm{primal}} \leftarrow \left(x + (L + D^{\mathrm{dec}})^{-1}(f - C^l x)\right)_{\mathrm{primal}}$ ▷ Update primal part
5: $x_{\mathrm{dual}} \leftarrow \left(x + (R + D^{\mathrm{dec}})^{-1}(f - C^l x)\right)_{\mathrm{dual}}$ ▷ Update dual part
6: $w \leftarrow (1 - \omega)w + \omega x$
7: **end for**
8: **return** w
9: **end function**

3.5.3. Extensions: Smoothers, preconditioners and one-level solvers

The above smoothers define a set of basic defect correction based algorithms which can be applied to the discrete space-time problem. In their basic form, these algorithms are sometimes slow and unstable. A common approach to enhance the robustness is to combine them with other iterative schemes by applying preconditioning techniques. This leads in a natural way to a concept which is called 'cascased smoothers' here and allows to define rather general one-level solvers which can be used for smoothing and coarse grid solving.

Every smoother can work as a preconditioner Let xSMOOTHER identify any of the above smoothers. For an arbitrary initial guess w and right-hand side f, setting

$$\tilde{w} := \text{xSMOOTHER}(C^l, w, f, \text{NSM})$$

for an arbitrary NSM $\in \mathbb{N}$ produces a new approximate $\tilde{w}$ to a solution w^* with $C^l w^* = f$. In particular, the simplest setting

$$\tilde{w} := \text{xSMOOTHER}(C^l, 0, f, 1) \tag{3.26}$$

generates an approximation $\tilde{w}$ to w^*. This can be used to define a preconditioner: One step of a typical preconditioned defect correction loop with preconditioner $\tilde{C}$ has the form

$$w_{\text{new}} := w + \tilde{C}^{-1}(f - C^l w)$$

or equivalently

$$\begin{aligned} &1.) \quad \tilde{C}g = d := (f - C^l w) \\ &2.) \quad w_{\text{new}} := w + g. \end{aligned}$$

The preconditioner $\tilde{C}$ should be an approximation to C^l, i. e., solving $\tilde{C}g = d$ should produce an intermediate solution g which is an approximation to g^* with $C^l g^* = d$. Setting $f := d$, this is exactly what (3.26) does, so g can be defined by

$$\tilde{C}g = d \quad :\Leftrightarrow \quad g := \text{xSMOOTHER}(C^l, 0, d, 1).$$

For an arbitrary (defect) vector d, based on the smoothers above, the following preconditioners can therefore be defined:

$$\begin{aligned} &1.) \ \text{FBJACPREC}(d) &&:= \text{FBJACSMOOTHER}(C^l, 0, d, 1) \\ &2.) \ \text{FBSIMPREC}(d) &&:= \text{FBSIMSMOOTHER}(C^l, 0, d, 1) \\ &3.) \ \text{FBGSPREC}(d) &&:= \text{FBGSSMOOTHER}(C^l, 0, d, 1) \\ &4.) \ \text{FBSORPREC}(d) &&:= \text{FBSORSMOOTHER}(C^l, 0, d, 1) \end{aligned}$$

Cascaded smoothers With the above preconditioners defined, any block-based, iterative, preconditioned, defect correction based solver can be used as a smoother in a multigrid setting. This is done by performing NSM $\in \mathbb{N}$ solver steps with one of the above preconditioners for preconditioning. Such a technique was for example successfully applied in [108] in order to avoid damping parameters in iterative smoothers like the Jacobi smoother. However, it has to be noted that the underlying system is not symmetric: Already the different structure of the mass matrix block on the offdiagonals disturb the symmetry, not to mention a potential nonlinearity. Algorithms that assume symmetric matrices (like the standard CG method) should therefore be avoided.

For numerical tests in later chapters, the most prominent example of such a 'cascaded' algorithm is adapted from [108] to the space-time case: With BiCGStab($C^l, w, f,$ NITE), NITE steps of the BiCGStab algorithm [154] is identified, starting from an initial approximation w to w^* with $C^l w^* = f$. Furthermore, xPrec refers to any of the preconditioners in b). Using the preconditioned variant of BiCGStab, a *cascaded* smoother can be defined by

$$\text{BiCGStab}(\text{xPrec}, \text{NSM}).$$

As an example, BiCGStab(FBGSPrec, NSM) stands for applying NSM smoothing steps with a BiCGStab solver, BiCGStab($C^l, w, f,$ NSM), preconditioned by FBGSPrec.

The coarse grid solver Finally, every iterative smoother can also be used as a solver to solve an equation of the form $C^l w = f$ (assuming convergence of the smoother). For that purpose, the fixed number of iterations in the smoother has to be replaced by a stopping criterion depending on the residuum. This can be used to formulate a coarse grid solver for the multigrid iteration; for example BiCGStab(FBGSPrec) or a similar combination of algorithms can be used as solver for the coarse grid problems.

Generally, for a smoother xSmoother, the corresponding solver will in the following be denoted by xSolver. If the smoother is depending on a damping parameter, this is added to the notation. For example, FBJacSolver(ω) denotes the solver that corresponds to the damped Block Jacobi smoother. A preconditioned BiCGStab solver will be referred to as BiCGStab(xPrec), so in contrast to the smoother, the NSM is omitted in the notation.

3.6. Coupled multigrid solvers in space

Up to this point, the structure of the submatrices in the global space-time matrix did not matter. All algorithms have been formulated in an abstract matrix-vector form without exploiting the structure of the underlying state and adjoint equations in space. However, for an efficient preconditioning of the spatial subproblems, this plays an important role.

As mentioned above, for each component in time, linear systems of the form

$$D_n w_n = f_n \tag{3.27}$$

have to be solved. These can be interpreted as subproblems in space, with w_n and f_n denoting the n-th time component of the solution and right-hand side block vectors $w \in W^l$ and $f \in (W^l)^\times$, respectively. $D_n = C_{n,n}$ refers to the n-th diagonal block of the underlying space-time matrix C^l on level l. If the system stems from FBSimSmoother/FBSimPrec, there is $D_n = D_n^{\text{dec}} = C_{n,n}^{\text{dec}}$.

Since the operator D_n represents a continuous operator (like Poisson, Stokes, convection,...), e. g., a monolithic multigrid solver in space can be applied which is expected to converge with a (space-) level-independent convergence rate. This type of solver builds upon the hierarchy of spatial meshes which has been generated during the discretisation of the space-time problem. Basically, it needs a hierarchy of problems, a smoother on each level, a coarse grid solver and appropriate prolongation/restriction operators. Figure 3.6 illustrates this situation: For every solution component n (identified with time t_n here), a local spatial hierarchy is formed which serves as basis for an underlying multigrid in space.

The local hierarchy in space Equation (3.27) corresponds to level l in the space-time hierarchy. Due to the fact that $w \in W^l = W^{l,l}$ and $f \in (W^l)^\times = (W^{l,l})^\times$, there is

$$w_n \in V^l, \qquad f_n \in (V^l)^*.$$

A natural finite element hierarchy corresponding to $\Omega_1, ..., \Omega_l$ is

$$V^1, ..., V^l.$$

The operator D_n stems from a continuous operator and thus, there is a hierarchy of operators in space available,

$$D^1, ..., D^l = D_n, \quad D^m : V^m \to (V^m)^*, \quad D^m \approx D_n, \quad m = 1, ..., l.$$

Here, $D^m = D_n^m \approx D_n$ denotes that D^m is a representation of D_n on level m, created with techniques similar to the space-time case in Section 3.4.5.

Canonical definition of the operator hierarchy The canonical way to create the hierarchy $D^1, ..., D^l$ is sketched in the following. In the nonlinear case, according to (3.22), the canonical choice for C^l in $W^l = W^{l,l}$ is

$$C^l = G^l(w_i^l) \quad \text{and} \quad C^l = F^l(w_i^l),$$

respectively. Similarly, a canonical choice for a preconditioner in $W^{l,m}$ is

$$C^{l,m} := G^{l,m}(w_i^{l,m}) \quad \text{and} \quad C^{l,m} := F^{l,m}(w_i^{l,m}),$$

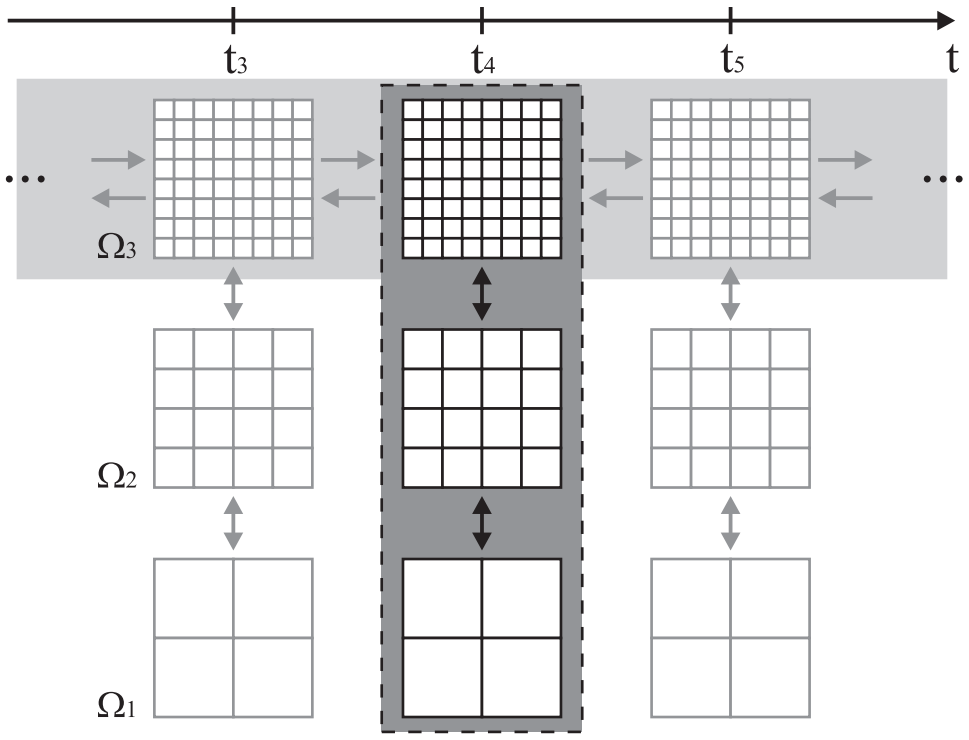

Figure 3.6: During the space-time preconditioning sweeps (topmost mesh row), a local mesh hierarchy is formed at each t_n to solve a local linear system with a spatial multigrid solver (mesh columns).

respectively, with $w_i^{l,m} \in W^{l,m}$ an approximation to w_i^σ created with the methods in Section 3.4.5 — however, due to the fact that the time level l is not modified, there is no restriction in time necessary.

Similarly, in the linear case, the canonical way for the preconditioner in $W^{l,m}$ is

$$C^{l,m} := G^{l,m}$$

In all cases, $C^{l,m}$ approximates $C^{l,l}$. Thus, $D^m = D_n^m$ is chosen as the n-th diagonal block of $C^{l,m}$.

Prolongation/Restriction/Coarse grid operators in space The required prolongation and restriction operators based on the applied finite element spaces are standard and well known (see for example [9, 32, 75, 142, 161]). Thus, the investigations are restricted to a short discussion of smoothing operators. For the coarse grid problems, either a direct solver can be used (for small problems) or a coarse grid solver can be derived from a smoothing operator by processing the smoothing iteration on the coarse mesh until convergence (see also Section 3.5.3).

Smoothing operators in space The definition of a smoothing operator in space is a rather delicate task. On the one hand, the smoother should be 'strong' enough to guarantee fast convergence. On the other hand, its application should be fast. For standard problems like the heat equation, there are a couple of iterative numerical algorithms available, e. g., (Block)-Jacobi, (Block)-Gauß Seidel, SSOR or ILU algorithms. All of them are applicable in the optimal control context as well. However, the block systems that arise during the discretisation of Stokes or Navier–Stokes problems cannot be handled with these algorithms due to the saddle-point character. As a possible remedy, the following paragraphs give an introduction into the pressure Schur complement ('PSC') approach known from CFD problems (see also [132, 155, 159]). The main property of this approach is that it treats all variables in a monolithic way, i. e., it acts simultaneously on the primal and dual variables.

For simplicity, the smoothing algorithms are only described here for space level l, i. e., if being applied to (3.27). In a full multigrid context, smoothing takes place at level $m = 2, ..., l$, and thus, the underlying operator $D_n = D^l$ has to be replaced by its counterpart D^m on level m.

The saddle point system in detail Equation (3.27) has an algebraic counterpart of the form

$$Ax = b$$

with a matrix $A \cong D_n$, a vector $x = \vec{w}_n \cong w_n$ and a right-hand side vector $b = \vec{f}_n \cong f_n$. The notation $A \cong D_n$ is used to express that A is the matrix corresponding to the operator D_n. Furthermore, $\vec{w}_n$ and $\vec{f}_n$ refer to the vectors with degrees of freedom of w_n and f_n, respectively, which is expressed as $\vec{w}_n \cong w_n$ and $\vec{f}_n \cong f_n$.

In the case of the Stokes– and Navier–Stokes equations, on the background of optimal distributed control, this system can be written in the form

$$\begin{pmatrix} A^{\text{primal}} & M^{\text{dual}} & B & 0 \\ M^{\text{primal}} & A^{\text{dual}} & 0 & B \\ B^{\mathsf{T}} & 0 & 0 & 0 \\ 0 & B^{\mathsf{T}} & 0 & 0 \end{pmatrix} \begin{pmatrix} y \\ \lambda \\ p \\ \xi \end{pmatrix} = \begin{pmatrix} b_y \\ b_\lambda \\ b_p \\ b_\xi \end{pmatrix}.$$

In the special case that the primal and dual variables are decoupled (if FBSIMSMOOTHER/-FBSIMPREC is used), this system reduces to

$$\begin{pmatrix} A^{\text{primal}} & 0 & B & 0 \\ 0 & A^{\text{dual}} & 0 & B \\ B^{\mathsf{T}} & 0 & 0 & 0 \\ 0 & B^{\mathsf{T}} & 0 & 0 \end{pmatrix} \begin{pmatrix} y \\ \lambda \\ p \\ \xi \end{pmatrix} = \begin{pmatrix} b_y \\ b_\lambda \\ b_p \\ b_\xi \end{pmatrix}$$

or in short to the two decoupled subsystems

$$\begin{pmatrix} A^{\text{primal}} & B \\ B^{\mathsf{T}} & 0 \end{pmatrix} \begin{pmatrix} y \\ p \end{pmatrix} = \begin{pmatrix} b_y \\ b_p \end{pmatrix}, \qquad \begin{pmatrix} A^{\text{dual}} & B \\ B^{\mathsf{T}} & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \xi \end{pmatrix} = \begin{pmatrix} b_\lambda \\ b_\xi \end{pmatrix}.$$

All systems are typical saddle point problems for primal and/or dual variables, with velocity submatrices A^{primal}, A^{dual}, the possible coupling matrices between the primal and dual velocity M^{primal} and M^{dual}, and the gradient/divergence matrices B and B^{T}.

The PSCSMOOTHER The idea of the PSCSMOOTHER is to apply a defect correction of the type

$$x_{j+1} := x_j + \omega C^{-1}_{\text{space}}(b - Ax_j), \qquad \omega > 0$$

element by element, with C_{space} an appropriate preconditioner in space. The underlying mesh of the finite element space on (space-) level l is again denoted by Ω_l. In an outer loop over all all elements $K \in \Omega_l$, a global defect $d_j := b - Ax_j$ is created. The components of the defect which do not belong to the current element are forced to zero. As a consequence, the global defect can be reduced to a local defect d^K, and the global preconditioner reduces to a local preconditioner as all rows and columns not belonging to that element can be eliminated. This gives a local linear system and a local update for the current element. A more formal description reads as follows:

Let $I(K)$ identify a list of all degrees of freedom that can be found on element K, containing numbers for the primal and/or dual velocity vectors in all spatial dimensions and for the primal and/or dual pressure. With this index set, $A_{I(K)}$ defines a (rectangular) matrix containing only those rows from A identified by the index set $I(K)$. In the same way, $x_{I(K)}$ and $b_{I(K)}$ refer to the subvectors of x and b containing only the entries identified by $I(K)$. Furthermore, $A_{I(K),I(K)}$ stands for the (square) matrix that stems from extracting only those rows and columns from A identified by $I(K)$.

This notation allows to formulate the basic PSCSMOOTHER in space, see Algorithm 3.7. The parameter $\omega > 0$ is a damping parameter. Of course, this formulation is not yet complete, as it lacks a proper definition of the local preconditioner C_K^{-1}. This is a small square matrix with as many rows and columns as indices in $I(K)$.

Local element-preconditioners Two basic approaches for this preconditioner are commonly used. The first approach, which is entitled by PSCSMOOTHERFULL, results in the simple choice of $C_K := A_{I(K),I(K)}$ and applies C_K^{-1} by invoking a LU decomposition (e. g., with the LAPACK package [122]) or a local Schur complement decomposition. This approach is rather robust and still feasible as the system is small; for the $\tilde{Q}_1/Q_0$ space (see [142]) that is used in a number of numerical tests in later chapters, the system has 9 or 18 unknowns, respectively.

Algorithm 3.7 PSCSmoother for smoothing an approximate solution to $Ax = b$

Predefined constant: $\omega > 0$: damping parameter

1: **function** PSCSmoother(A,x,b,NSM)
2: **for** ism = 1 to NSM **do** ▷ NSM smoothing sweeps
3: **for all** $K \in \Omega_l$ **do** ▷ loop over the elements
4: $x_{I(K)} \leftarrow x_{I(K)} + \omega C_K^{-1}(b_{I(K)} - A_{I(K)}x)$ ▷ local correction
5: **end for**
6: **end for**
7: **return** x ▷ solution
8: **end function**

The second approach, which is referred to by PSCSmootherDiag, employs a different subset of the matrix A for the construction of C_K. To describe this approach, the following submatrix of A is used,

$$\hat{A} := \begin{pmatrix} \operatorname{diag}(A^{\text{primal}}) & 0 & B & 0 \\ 0 & \operatorname{diag}(A^{\text{dual}}) & 0 & B \\ B^{\mathsf{T}} & 0 & 0 & 0 \\ 0 & B^{\mathsf{T}} & 0 & 0 \end{pmatrix},$$

where 'diag(·)' refers to the operator taking only the diagonal of a given matrix. The local preconditioner can then be formulated as $C_K := \hat{A}_{I(K),I(K)}$. This approach decouples the primal and dual variables in the local system. Applying $\hat{A}^{-1}_{I(K),I(K)}$ decomposes into two independent subproblems. This reduces the stability but leads to a much faster solution procedure. Similar to the choice $C_K := A_{I(K),I(K)}$, the local systems are of saddle point type, so either a direct solver or a Schur complement decomposition can be invoked.

3.10 Remark. The choice $C_K := A_{I(K),I(K)}$ is the most natural one and motivated by the local linear system behind the correction. The defect in line 4 of Algorithm 3.7 can be interpreted in the following way:

$$b_{I(K)} - A_{I(K)}x \;=\; \tilde{b}_K - A_{I(K),I(K)}\, x_{I(K)} \tag{3.28}$$

with

$$\tilde{b}_K := b_{I(K)} - A_{I(K)}\, x + A_{I(K),I(K)}\, x_{I(K)}.$$

One directly verifies that $\tilde{b}_K$ does not depend on $x_{I(K)}$, the corresponding contributions from x and $x_{I(K)}$ cancel out. Therefore, (3.28) creates a defect for the local linear system

$$A_{I(K),I(K)}\, x_{I(K)} = \tilde{b}_K.$$

This system only solves for the degrees of freedom in K, all others are treated as 'Dirichlet values', i. e., as known. The optimal preconditioner in a defect correction loop is C_K^{-1} with $C_K := A_{I(K),I(K)}$.

3.11 Example. The PSCSmoother approach is a rather general approach and can be applied to a large variety of systems. To demonstrate the action of this smoother, a simple Poisson problem of the form

$$-\Delta y = f$$

is considered on the domain $\Omega = (0,1)^2$, $y, f : \Omega \to \mathbb{R}$. If this problem is discretised with the Q_1 finite element approach on a regular mesh with 9 cells, the associated discrete system reads

$$Ax = b \tag{3.29}$$

with $A \in \mathbb{R}^{16\times 16}$ a system matrix and $x, b \in \mathbb{R}^{16}$ the vectors with the degrees of freedom of the solution y and the right-hand side f, respectively.

For a given approximation x to a solution x^* of (3.29), Algorithm 3.7 applies an update-loop over all elements. Figure 3.7 illustrates a situation in which the elements K_1,...,K_4 are already processed. The formula to update the degrees of freedom corresponding to element K_5 with PSCSmoother reads

$$\begin{pmatrix} x_6 \\ x_7 \\ x_{10} \\ x_{11} \end{pmatrix} := \begin{pmatrix} x_6 \\ x_7 \\ x_{10} \\ x_{11} \end{pmatrix} + \omega C_5^{-1} \left(\begin{pmatrix} b_6 \\ b_7 \\ b_{10} \\ b_{11} \end{pmatrix} - \begin{pmatrix} a_{6,1} & \dots & a_{6,16} \\ a_{7,1} & \dots & a_{7,16} \\ a_{10,1} & \dots & a_{10,16} \\ a_{11,1} & \dots & a_{11,16} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{16} \end{pmatrix} \right)$$

whereby

$$C_5 = \begin{pmatrix} a_{6,6} & a_{6,7} & a_{6,10} & a_{6,11} \\ a_{7,6} & a_{7,7} & a_{7,10} & a_{7,11} \\ a_{10,6} & a_{10,7} & a_{10,10} & a_{10,11} \\ a_{11,6} & a_{11,7} & a_{11,10} & a_{11,11} \end{pmatrix} \quad \text{or} \quad C_5 = \begin{pmatrix} a_{6,6} & & & \\ & a_{7,7} & & \\ & & a_{10,10} & \\ & & & a_{11,11} \end{pmatrix}$$

for the PSCSmootherFull or PSCSmootherDiag approach, respectively. The entry $a_{i,j}$ corresponds to the i-th row and j-th column of the matrix A. The old entries x_6, x_7, x_{10} and x_{11} are overwritten and the new values are used for all remaining cells K_j, $j = 6, 7, 8, 9$.

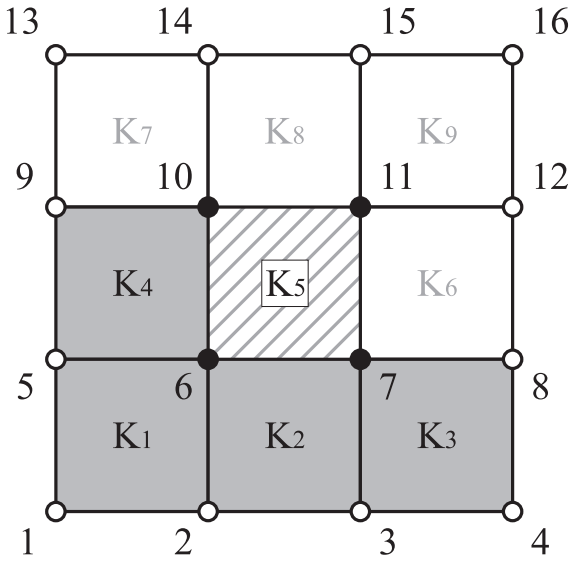

Figure 3.7: For element K_5, the PSCSmoother approach updates x_6, x_7, x_{10} and x_{11}.

3.12 Remarks. a) PSCSmootherDiag and PSCSmootherFull can also be used as a preconditioner in an outer iteration if applied with NSM = 1 and start vector $x = 0$

to a defect vector. The corresponding algorithms are called PSCPRECDIAG and PSCPRECFULL and are used in numerical tests as preconditioners in a BICGSTAB algorithm. Such 'cascaded' algorithms are denoted in a similar way to previous sections: The term BICGSTAB(PSCPRECDIAG,NSM=4) for example refers to a BICGSTAB smoother with four smoothing steps, preconditioned by PSCPRECDIAG.

b) Most of the numerical tests in later chapters are carried out using BICGSTAB(PSCPRECDIAG,NSM=4) as smoother in space unless otherwise indicated. As coarse grid solver, BICGSTAB(PSCPRECDIAG) is usually used.

c) If spatial problems turn out to be very difficult, it is possible to increase the stability by applying this approach to certain patches of cells (cf. [132]) but this concept is beyond the scope of the investigations here.

3.7. Stopping criteria and the inexact Newton algorithm

The above descriptions of iterative solution algorithms still lack the definition of proper stopping criteria. Different choices are possible here. The following list gives an overview about the stopping criteria of all the solver components that interact with each other or are used in numerical tests. Generally speaking, all stopping criteria provide a 'relative' control of the residual, i. e., they aim at damping the norm of a given initial residual by a number of digits. All calculations are carried out in double precision. An optional 'absolute' error bound stops the iteration prematurely if the norm of the residual drops below a certain value. This is meant as a kind of fallback strategy if the norm of the residual drops below machine accuracy and cannot be reduced anymore. There is no control of the iteration in the sense that the norm of the residual must drop below an defined value or that the relative change in the solution has to be less than a minimum tolerance. Such variants can be seen as possible extensions but are beyond the scope of this work.

3.7.1. Basic stopping criteria

The l_2-norm of a vector containing degrees of freedom is denoted by $||\cdot||_{l_2}$. Furthermore, $\varepsilon_{\mathrm{OptNL}}$, $\varepsilon_{\mathrm{OptMG}}$, $\varepsilon_{\mathrm{CoarseMG}}$, $\varepsilon_{\mathrm{SpaceMG}}$, $\varepsilon_{\mathrm{SimNL}}$, $\varepsilon_{\mathrm{SimMG}} > 0$ define 'relative' error bounds. Appropriate choices for these parameters are given in later sections based on numerical studies.

1.) In a full space-time Newton algorithm, a couple of solvers interact with each other. The outer Newton solver is a nonlinear solver which produces in the n-th nonlinear iteration a residual r_n^{OptNL}. The solver stops the iteration if

$$||r_n^{\mathrm{OptNL}}||_{l_2} < \varepsilon_{\mathrm{OptNL}}||r_0^{\mathrm{OptNL}}||_{l_2} \qquad \text{or} \qquad ||r_n^{\mathrm{OptNL}}||_{l_2} < 10^{-14}$$

is fulfilled. The solver detect divergence of the iteration if the initial residuum is increased by more than five digits. Except where noted differently, the iteration is stopped prematurely if more than 20 iterations are carried out.

2.) In each nonlinear iteration, the linear space-time multigrid solver produces in its n-th step the residual r_n^{OptMG}. The solver stops the iteration once

$$||r_n^{\mathrm{OptMG}}||_{l_2} < \varepsilon_{\mathrm{OptMG}}||r_0^{\mathrm{OptMG}}||_{l_2} \qquad \text{or} \qquad ||r_n^{\mathrm{OptMG}}||_{l_2} < 10^{-14}$$

is fulfilled. The solver detect divergence of the iteration if the initial residuum is increased by more than three digits. Except where noted differently, the iteration is stopped prematurely if more than ten iterations are carried out.

3.) On the space-time coarse mesh, a linear one-level solver is used. This is an iterative solver (usually a preconditioned version of BiCGStab) which produces in its n-th step the residual r_n^{CoarseMG}. The solver stops the iteration if

$$||r_n^{\mathrm{CoarseMG}}||_{l_2} < \varepsilon_{\mathrm{CoarseMG}}||r_0^{\mathrm{CoarseMG}}||_{l_2} \qquad \text{or} \qquad ||r_n^{\mathrm{CoarseMG}}||_{l_2} < 10^{-14}$$

is fulfilled. For the coarse grid solver, $\varepsilon_{\mathrm{CoarseMG}} = \varepsilon_{\mathrm{OptMG}}$ is always used, except where noted. The solver detect divergence of the iteration if the initial residuum is increased by more than three digits. Except where noted differently, the iteration is stopped prematurely if more than 15 iterations are carried out.

4.) For solving and preconditioning subproblems in space, a linear multigrid solver is applied. The n-th step of this solver produces a residual r_n^{SpaceMG}. The solver stops if

$$||r_n^{\mathrm{SpaceMG}}||_{l_2} < \varepsilon_{\mathrm{SpaceMG}}||r_0^{\mathrm{SpaceMG}}||_{l_2} \qquad \text{or} \qquad ||r_n^{\mathrm{SpaceMG}}||_{l_2} < 10^{-14}$$

is fulfilled. The solver detect divergence of the iteration if the initial residuum is increased by more than three digits. Except where noted differently, the iteration is stopped prematurely if more than 100 iterations are carried out.

5.) Sometimes, a comparison between an optimisation and a simulation is done. To keep the solvers comparable, the simulation solver always uses the same kind of preconditioners in space as the optimisation (usually both use PSCSmootherDiag). The nonlinear simulation solver in space produces in its n-th step the residual r_n^{SimNL} and stops if

$$||r_n^{\mathrm{SimNL}}||_{l_2} < \varepsilon_{\mathrm{SimNL}}||r_0^{\mathrm{SimNL}}||_{l_2} \qquad \text{or} \qquad ||r_n^{\mathrm{SimNL}}||_{l_2} < 10^{-14}$$

is fulfilled. In each nonlinear iteration, a spatial multigrid solver is used for the linear subproblems. This solver produces in its n-th step the residual r_n^{SimMG} and stops if the relative stopping criterion

$$||r_n^{\mathrm{SimMG}}||_{l_2} < \varepsilon_{\mathrm{SimMG}}||r_0^{\mathrm{SimMG}}||_{l_2} \qquad \text{or} \qquad ||r_n^{\mathrm{SimMG}}||_{l_2} < 10^{-14}$$

is fulfilled.

3.7.2. The inexact Newton algorithm

Based on the above stopping criteria, a variant of Newton's algorithm is formulated to reduce numerical costs during the calculation. The idea is to adapt the accuracy (stopping criterion) of the inner linear solver in relation to the residual of the outer nonlinear Newton solver in such a way that quadratic convergence is maintained. Heuristically, this strategy invests exactly as much effort into the linear subproblems as necessary for the global nonlinear iteration, thus reducing the overall CPU time in a quasi-optimal way. A description of this strategy can be found, e. g., in [84, 85] or [92, Chapter 4.3.2]. For an analysis of inexact Newton methods, see [104, Chapter 6].

The n-th nonlinear residual of the space-time Newton is again denoted by r_n^{OptNL} and r_m^{OptMG} refers to the m-th residual in the linear space-time-solver during the n-th nonlinear iteration. The inexact Newton algorithm is defined as the standard Newton algorithm with a modified stopping criterion for the linear solver. In detail, the linear solver terminates in its m-th iteration if

$$\frac{||r_m^{\mathrm{OptMG}}||_{l_2}}{||r_0^{\mathrm{OptMG}}||_{l_2}} \leq \min\left\{ \left(\frac{||r_n^{\mathrm{OptNL}}||_{l_2}}{||r_0^{\mathrm{OptNL}}||_{l_2}} \right)^{q-1}, \varepsilon_{\mathrm{OptIN}} \right\} \tag{3.30a}$$

holds. If zero is used as initial guess of the linear solver, an equivalent formulation reads

$$\frac{||r_m^{\mathrm{OptMG}}||_{l_2}}{||r_0^{\mathrm{OptNL}}||_{l_2}} \leq \min\left\{\left(\frac{||r_n^{\mathrm{OptNL}}||_{l_2}}{||r_0^{\mathrm{OptNL}}||_{l_2}}\right)^q, \varepsilon_{\mathrm{OptIN}}\left(\frac{||r_n^{\mathrm{OptNL}}||_{l_2}}{||r_0^{\mathrm{OptNL}}||_{l_2}}\right)\right\} \tag{3.30b}$$

due to the fact that $r_0^{\mathrm{OptMG}} = r_n^{\mathrm{OptNL}}$. The constant $\varepsilon_{\mathrm{OptIN}} > 0$ specifies the minimum residual reduction in each nonlinear iteration and can be set, e. g., to $\varepsilon_{\mathrm{OptIN}} := 10^{-2}$. On the other hand, $q > 0$ specifies the convergence order. For $q := 2$, the algorithm converges quadratically near the optimum similar to the standard Newton algorithm. Setting $1 < q < 2$ results in superlinear convergence.

3.8. Summary and conclusions

In this chapter, the whole solver methodology has been described. The key for the design is that the underlying optimisation problem has been formulated as a continuous set of partial differential equations, the KKT system, on the space-time cylinder. Following the discretisation introduced in the previous chapter, this KKT system has been discretised on a hierarchy of meshes.

A nonlinear defect correction solver has been applied on the finest mesh which realises in a special case a nonlinear Newton solver.

A linear space-time multigrid solver has been formulated which exploits the problem hierarchy. The purpose of this solver has been to act as a preconditioner during the above nonlinear iteration. The main difficulties in the definition of a multigrid solver have been the definition of prolongation/restriction operators and the definition of efficient smoothers/preconditioners/coarse grid solvers which are usually only efficient if created in a problem dependent way.

Prolongation/restriction operators have been formulated as a combination of finite difference prolongation/restriction operators in time and finite element prolongation/restriction operators in space. The most crucial choice in this context has been an appropriate definition of the restriction operator for the general θ-scheme due to the special interpretation of the discrete dual solutions in time. For the formulation of the operators, the concept of 'discrete abstract functions' has been introduced which has allowed to adapt the standard multigrid notation for 1D finite difference discretisations to the time axis.

Smoothers/preconditioners/coarse grid solvers have been derived based on the special matrix form of the discrete KKT system. The most important point in this context is that all space-time matrices feature a block tridiagonal form. This has allowed to adapt iterative block-based solution algorithms like the block Jacobi or block Gauß–Seidel algorithm as smoother, preconditioner and coarse grid solver. With the help of other iterative schemes like BiCGStab, complex, cascaded smoothers/coarse grid solvers have been formulated which aim at avoiding the choice of damping parameters. At the end of the day, the solution of linear systems on the space-time cylinder has been reduced to global matrix-vector multiplications and sequences of local linear systems, based on the diagonal blocks of the global matrices.

Local linear systems in space have been processed by a standard monolithic multigrid solver, based on a hierarchy of finite element discretisations in space. At this point,

the definition of *local pressure Schur complement* solvers has come into play which have allowed to reduce linear systems in space to small linear systems on each cell of the finite element mesh. Finally, this is the key for efficient smoothers in space, also for linear systems with saddle-point structure which appear during the discretisation of the Stokes and the Navier–Stokes equations.

This completes the description of the main solver components. Of course, there are several possible extensions which have been omitted for simplicity. For example, the technique can be applied to more complicated model problems including, e. g., end time observation or constraints in the control. The next chapter will illustrate some of these extensions; globalisation strategies to enhance the robustness of the solver can be found in Appendix B for completeness. Chapter 5 will continue with the numerical analysis of the proposed solver strategy. Starting with an analysis of the heat equation and proceeding to the Navier–Stokes equation, the analysis will cover all main proposed solver components as well as some of the extensions and study their efficiency and robustness.

4

Extended systems and additional discretisation strategies

Based on a set of model problems, the two previous chapters have introduced on the one hand a discretisation strategy for the underlying KKT systems on the space-time cylinder, on the other hand a hierarchical solution strategy which aims at solving the underlying KKT systems with linear complexity.

This chapter provides an overview of important extensions to the method. The model problems from Chapter 2 are modified, e. g., to more general boundary conditions and constraints in the control. The previous model problems are special cases of the new ones. These modifications provide additional flexibility for numerical tests in later chapters.

Outline

Sections 4.1 to 4.4 describe a set of common extensions of the method to more general model problems. At first, Section 4.1 introduces an additional end time observation at $t = T$. Section 4.2 illustrates constraints in the control and Section 4.3 generalises the boundary conditions to support outflow boundary conditions in the context of optimal control of the Stokes and Navier–Stokes equations. Finally, Section 4.4 switches the focus from model problems to discretisation strategies. This section describes the semi-explicit time discretisation which is used by other authors and highlights advantages and disadvantages in comparison to the implicit Euler and Crank–Nicolson time discretisation.

4.1. The end time observation

The formulations of the considered optimisation problems can be extended by an end time observation. This allows the user to influence the behaviour of the solution at the end of the time interval. In the following, the end time observation is at first introduced using the heat equation and later generalised to the Stokes and Navier–Stokes equations. Although the definition is easily introduced into the infinite dimensional problem, the choice of an appropriate discretisation is more delicate if it comes to higher order time discretisation schemes such as the Crank–Nicolson scheme.

With a parameter $\gamma \geq 0$, adapting the notation from Chapter 2, the distributed control problems involving end time observation have the following, form. cf. [61, 84]. For the heat equation, $y, u : \mathcal{Q} \to \mathbb{R}$ have to be computed with

$$J(y,u) := \frac{1}{2}||y - z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 + \frac{\gamma}{2}||y(T,\cdot) - z(T,\cdot)||_{\Omega}^2 \quad \longrightarrow \quad \min! \tag{4.1}$$

such that

$$\begin{aligned} y_t - \Delta y &= u && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma. \end{aligned}$$

For the Stokes/Navier–Stokes equations on the other hand, $y, u : \mathcal{Q} \to \mathbb{R}^{\dim}$ and $p : \mathcal{Q} \to \mathbb{R}$ have to be computed with

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 + \frac{\gamma}{2}||y(T,\cdot)-z(T,\cdot)||_{\Omega}^2 \quad \longrightarrow \quad \min! \tag{4.2}$$

such that

$$\left\{ \begin{aligned} y_t - \nu\Delta y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma, \end{aligned} \right\} \quad \text{or} \quad \left\{ \begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma, \end{aligned} \right\}$$

respectively. Similar to Chapter 2, y^0 and g define the initial and boundary conditions. Applying the usual Lagrange multiplier technique, the corresponding infinite dimensional KKT systems are derived. These are identical to those in Chapter 2 with the exception that the end time condition

$$\lambda(T,\cdot) = 0 \qquad \text{in } \Omega$$

is replaced by

$$\lambda(T,\cdot) = \gamma(y(T,\cdot) - z(T,\cdot)) \qquad \text{in } \Omega. \tag{4.3}$$

To derive an appropriate discretisation in terms of an implicit time stepping scheme, the *First-Discretise-Then-Optimise* strategy based on the Lagrange multiplier approach is used in the following.

4.1.1. End time observation for the implicit Euler scheme

The implicit Euler timestepping scheme allows to work with values at the end of the time intervals. As a consequence, the discretisation of the end time condition can be done in a straightforward way.

The heat equation At first, the optimal control of the heat equation is considered. Similar to Section 2.5 on page 31ff, a discrete form of the functional $J(\cdot,\cdot)$ is defined as follows.

$$J_{IET}(y,u) := \frac{1}{2}k\sum_{n=1}^{n}||y_n - z_n||_{\Omega}^2 + \frac{\alpha}{2}k\sum_{n=1}^{n}||u_n||_{\Omega}^2 + \frac{\gamma}{2}||y_N - z_N||_{\Omega}^2 \quad \longrightarrow \quad \min! \tag{4.4}$$

$$\begin{aligned} \text{s.t.} \qquad (y_n - y_{n-1}) - k\Delta y_n &= ku_n && \text{in } \Omega,\ n = 1,...,N, \\ y_0 - k\Delta y_0 &= y^0 - k\Delta y^0 && \text{in } \Omega, \\ y_n &= g && \text{at } \Sigma,\ n = 0,...,N. \end{aligned}$$

In the next step, the Lagrange functional is set up,

$$\begin{aligned} L_{IET}(y,u,\lambda) &:= J_{IET}(y,u) \\ &+ \sum_{n=1}^{N} \big(\lambda_n, ku_n - (y_n - y_{n-1} - k\Delta y_n)\big) \\ &+ \big(\lambda_0, (y^0 + k\Delta y^0) - (y_0 - k\Delta y_0)\big), \end{aligned}$$

with λ_0, ..., $\lambda_N : \mathbb{R}^{\dim} \to \mathbb{R}$. Requiring $DL_{IET}(y,u,\lambda) = 0$ and eliminating the control u results in the discrete KKT system (2.36) on page 35 with the discrete end time condition replaced by

$$\frac{\lambda_N}{k} - \Delta\lambda_N = -(1 + \frac{\gamma}{k})(y_N - z_N). \tag{4.5}$$

Using $\mathcal{A} : V \to V^*$, $\mathcal{A} : v \mapsto -\Delta v$ for all $v \in V$, the KKT system can be written in matrix-vector notation on the space-time cylinder,

$$\mathbf{G}w = f,$$

for $w_n := (y_n, \lambda_n)$, $w := (w_0, w_1, ...)$. The right hand side is given by

$$f = \Big(\underbrace{(\mathcal{I}/k + \mathcal{A})y^0, 0}_{f_0}, \underbrace{0, -z_1}_{f_1}, ..., \underbrace{0, -z_{N-1}}_{f_{N-1}}, \underbrace{0, -(1+\gamma/k)z_N}_{f_N} \Big),$$

which is the same as in Section 2.5.3 on page 34f except for the last entry. The system matrix reads

$$\mathbf{G} = \left(\begin{array}{c|c|c|c|c} \mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{G}_N \end{array} \right) \tag{4.6}$$

with the submatrices defined as in Section 2.5.3. The only exception is the last diagonal block which is replaced by

$$\mathbf{G}_N = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & \frac{\mathcal{I}}{\alpha} \\ -(1 + \frac{\gamma}{k})\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A} \end{pmatrix}.$$

The Stokes/Navier–Stokes equations For the Stokes and the Navier–Stokes equations, the modifications are exactly the same. In the case of the Navier–Stokes equations, the discretisation of the end time condition is given by

$$\begin{aligned} \frac{\lambda_N}{k} + \mathcal{N}_N^* \lambda_N + \nabla \xi_N &= -(1 + \frac{\gamma}{k})(y_N - z_N) \\ -\operatorname{div} \lambda_N &= 0 \end{aligned}$$

with $\mathcal{N}_N^*$ defined as in Section 2.5. The whole space-time system reads again

$$\mathbf{G}(w)w = f,$$

with the right-hand side given by

$$f = \Big(\underbrace{(\mathcal{I}/k + \mathcal{C}_0)y^0, 0, 0, 0}_{f_0}, \underbrace{0, -z_1, 0, 0}_{f_0}, \\ ..., \underbrace{0, -z_{N-1}, 0, 0}_{f_{N-1}}, \underbrace{0, -(1+\gamma/k)z_N, 0, 0}_{f_N} \Big)^{\mathsf{T}}$$

and the system matrix

$$\mathbf{G} = \mathbf{G}(w) = \left(\begin{array}{c|c|c|c|c} \mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{G}_N \end{array} \right). \tag{4.7}$$

The submatrices are the same as in Section 2.5, only the last diagonal matrix block changes to

$$\mathbf{G}_N = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{C}_n & \frac{\mathcal{I}}{\alpha} & \mathcal{G} & 0 \\ -(1+\frac{\gamma}{k})\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{N}_n^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}.$$

This has a slight influence to the Fréchet derivative which is used for the Newton iteration. Similar to Section 2.5.2 on page 34, the corresponding matrix has the form

$$\mathbf{F}(w) = \left(\begin{array}{c|c|c|c|c} \mathbf{F}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{F}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{F}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{F}_N \end{array} \right).$$

with the same submatrices as in Section 2.5.2, except for

$$\mathbf{F}_N = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{N}_N & \frac{1}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -(1+\frac{\gamma}{k})\mathcal{I} + \mathcal{R}_N & \frac{\mathcal{I}}{k} + \mathcal{N}_N^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}.$$

4.1.2. End time observation for the general θ-scheme

For a time discretisation with a general θ-scheme in the proposed way, the pointwise restriction

$$\lambda(T, \cdot) = \gamma(y(T, \cdot) - z(T, \cdot))$$

cannot be formulated in terms of y_N, λ_N and z_N since there is no λ_N, but $\lambda_{N-1+\theta}$ for $0 < \theta < 1$. For the definition of a proper discrete functional $J_{TST}(y, u)$, the term $\frac{\gamma}{2}||y(T, \cdot) -$

$z(T,\cdot)||_\Omega^2$ in the functional $J(y,u)$ has to be discretised in an appropriate way. Different choices are possible here. One possibility is for example to use the approximation

$$||y(T,\cdot)-z(T,\cdot)||_\Omega^2 = \frac{1}{K}\int_{T-K}^{T}\big(y(t)-z(t)\big)^2\,dt + \mathcal{O}(K) \tag{4.8}$$

for $K>0$ small and to replace $||y(T,\cdot)-z(T,\cdot)||_\Omega^2$ by $\frac{1}{K}\int_{T-K}^{T}\big(y(t)-z(t)\big)^2\,dt$. The advantage is that $\int_{T-K}^{T}\big(y(t)-z(t)\big)^2\,dt$ can be discretised with the trapezoidal rule in the same way as $||y-z||_Q^2$. The corresponding scheme would be of order $\mathcal{O}(K)$ in the (real) end time condition, provided that k is small enough, i. e., $0<k\ll K$. For a fixed K and $\theta=1/2$, this scheme would be second order accurate in time and in the (modified) end time condition.

A modification of this scheme is preferred in this work. Setting $K:=k$, an end time condition is obtained which converges with first order to $||y(T,\cdot)-z(T,\cdot)||_\Omega^2$. For the heat equation and the Stokes/Navier–Stokes equations, this approach reads as follows.

The heat equation Using a (modified) trapezoidal rule for the last timestep results in the discrete functional

$$\begin{aligned} J_{TST}(y,u) := {}& \frac{1}{2}k\sum_{n=1}^{n}\Big((1-\theta)\,||y_{n-1}-z_{n-1}||_\Omega^2+\theta\,||y_n-z_n||_\Omega^2\Big)\\ &+\frac{\alpha}{2}k\sum_{n=0}^{n}||u_{n-1+\theta}||_\Omega^2\\ &+\frac{\gamma}{2}\Big((1-\theta)\,||y_{N-1}-z_{N-1}||_\Omega^2+\theta\,||y_N-z_N||_\Omega^2\Big) \end{aligned} \tag{4.9}$$

which has to be minimised under the constraint of the heat equation or the Stokes/Navier–Stokes equations.

To derive the appropriate scheme, once again, a proper discrete Lagrange functional $L_{TST}(\cdot)$ is defined. From $DL_{TST}(\cdot)=0$, the discrete KKT system is obtained. This can be formulated in matrix-vector notation on the space-time cylinder,

$$\mathbf{G}w=f,$$

with

$$\mathbf{G}=\mathbf{G}^\theta=\left(\begin{array}{c|c|c|c|c} \mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{G}_N \end{array}\right). \tag{4.10}$$

In the particular case of the heat equation, the solution w has the form

$$w:=(w_0,w_1,...) \qquad \text{with} \qquad w_n:=(y_n,\lambda_{n-1+\theta}),$$

and the submatrices are the same as in Section 2.6.3 on page 41 except for the last two diagonal blocks; these read

$$\mathbf{G}_{N-1}=\begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{A}_{N-1}^\theta & \frac{1}{\alpha}\mathcal{I}\\ -(1+(1-\theta)\frac{\gamma}{k})\mathcal{I} & \frac{\mathcal{I}}{k}+\mathcal{A}_{N-1}^\theta \end{pmatrix}, \qquad \mathbf{G}_N=\begin{pmatrix} \frac{\mathcal{I}}{k}+\mathcal{A}_N^\theta & \frac{1}{\alpha}\mathcal{I}\\ -(\theta+\theta\frac{\gamma}{k})\mathcal{I} & \frac{\mathcal{I}}{k}+\mathcal{A}_N^\theta \end{pmatrix}.$$

The right-hand side is given by

$$f = \Big(\underbrace{(\mathcal{I}/k + \mathcal{A}_0^\theta) y^0, -(1-\theta) z_0}_{f_0}, \underbrace{0, -z_1}_{f_1}, \ ..., \underbrace{0, -z_{N-2}}_{f_{N-2}}, \\ \underbrace{0, -\left(1 + (1-\theta)\frac{\gamma}{k}\right) z_{N-1}}_{f_{N-1}}, \underbrace{0, -\left(\theta + \theta\frac{\gamma}{k}\right) z_N}_{f_N} \Big)$$

and coincides with the right-hand side in Section 2.6.3 except for the last two subvectors. In an explicit form, the end time condition reads

$$\frac{\lambda_{N-2+\theta} - \lambda_{N-1+\theta}}{k} - \theta \Delta\lambda_{N-2+\theta} - (1-\theta)\Delta\lambda_{N-1+\theta} = \left(1 + (1-\theta)\frac{\gamma}{k}\right)(y_{N-1} - z_{N-1}), \\ \frac{\lambda_{N-1+\theta}}{k} - \theta\Delta\lambda_{N-1+\theta} = \left(\theta + \theta\frac{\gamma}{k}\right)(y_N - z_N).$$

For $\theta = 1$, this reduces to (4.5).

The Stokes/Navier–Stokes equations A similar modification can also be applied for the Stokes and Navier–Stokes equations. In particular in the Navier–Stokes case, the Laplace operator changes to the corresponding nonlinear operator, thus the end time condition in explicit form reads

$$\frac{\lambda_{N-2+\theta} - \lambda_{N-1+\theta}}{k} - \mathcal{N}_{N-1}^{\theta,*}\lambda_{N-2+\theta} - \mathcal{N}_{N-1}^{1-\theta,*}\lambda_{N-1+\theta} = \left(1 + (1-\theta)\frac{\gamma}{k}\right)(y_{N-1} - z_{N-1}), \\ \frac{\lambda_{N-1+\theta}}{k} - \mathcal{N}_N^{\theta,*}\lambda_{N-1+\theta} = \left(\theta + \theta\frac{\gamma}{k}\right)(y_N - z_N), \\ -\operatorname{div}\lambda_{N-2+\theta} = -\operatorname{div}\lambda_{N-1+\theta} = 0.$$

Therefore, the right-hand side changes to

$$f = \Big(\underbrace{(\mathcal{I}/k + \theta\mathcal{C}_0) y^0, -(1-\theta) z_0, 0, 0}_{f_0}, \underbrace{0, -z_1, 0, 0}_{f_1}, \ ..., \\ \underbrace{0, -\left(1 + (1-\theta)\frac{\gamma}{k}\right) z_{N-1}, 0, 0}_{f_{N-1}}, \underbrace{0, -\left(\theta + \theta\frac{\gamma}{k}\right) z_N, 0, 0}_{f_N} \Big)$$

and, in contrast to the scheme in Section 2.6, the last two diagonal blocks read

$$\mathbf{G}_{N-1} = \mathbf{G}_{N-1}^\theta = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{C}_{N-1}^\theta & \frac{1}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -(1 + (1-\theta)\frac{\gamma}{k})\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{N}_{N-1}^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

$$\mathbf{G}_N = \mathbf{G}_N^\theta = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{C}_N^\theta & \frac{1}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -(\theta + \theta\frac{\gamma}{k})\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{N}_N^{\theta,*} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}.$$

This also has a direct influence to the matrix of the corresponding Fréchet derivative which is used for the Newton iteration. The Fréchet derivative $\mathbf{F}(w)$ of the mapping $w \mapsto \mathbf{G}(w)w$ is the matrix

$$\mathbf{F}(w) = \mathbf{F}^\theta(w) = \left(\begin{array}{c|c|c|c|c} \mathbf{F}_0 & \hat{\mathbf{J}}_0 & & & \\ \hline \breve{\mathbf{J}}_1 & \mathbf{F}_1 & \hat{\mathbf{J}}_1 & & \\ \hline & \breve{\mathbf{J}}_2 & \mathbf{F}_2 & \hat{\mathbf{J}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{J}}_N & \mathbf{F}_N \end{array}\right),$$

where all submatrices are the same as in Section 2.6.2, except for the last two diagonal blocks. These read

$$\mathbf{F}_{N-1} = \mathbf{F}^\theta_{N-1} = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{N}^\theta_{N-1} & \frac{\theta}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -(1+(1-\theta)\frac{\gamma}{k})\mathcal{I} + \mathcal{R}^\theta_{N-1} + \mathcal{R}^{1-\theta}_N & \frac{\mathcal{I}}{k} + \mathcal{N}^{\theta,*}_{N-1} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix},$$

$$\mathbf{F}_N = \mathbf{F}^\theta_N = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{N}^\theta_N & \frac{\theta}{\alpha}\mathcal{I} & \mathcal{G} & 0 \\ -(\theta+\theta\frac{\gamma}{k})\mathcal{I} + \mathcal{R}^\theta_N & \frac{\mathcal{I}}{k} + \mathcal{N}^{\theta,*}_N & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}.$$

4.2. Constrained Control

The following paragraphs focus on a variant of the distributed control of the nonstationary Navier–Stokes equations, introducing constraints in the control. This problem can formally be expressed in the minimisation problem

$$J(y,u) := \frac{1}{2}||y-z||^2_{\mathcal{Q}} + \frac{\alpha}{2}||u||^2_{\mathcal{Q}} \quad \longrightarrow \quad \min! \tag{4.11}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \nu\Delta y + y\nabla y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } \Sigma, \\ \\ a \le u &\le b && \text{in } \mathcal{Q}, \end{aligned}$$

for $y, z, u : \mathcal{Q} \to \mathbb{R}^{\dim}$, $p : \mathcal{Q} \to \mathbb{R}$ and two functions $a, b : \mathcal{Q} \to \mathbb{R}^{\dim}$, $a \le b$ a. e. in $\mathcal{Q}$.

4.2.1. The projection operator

The following three steps define a standard projection operator onto a closed interval in $\mathbb{R}$ and generalise it in a straightforward way to abstract functions on the space-time cylinder. With the help of this projection operator, the above problem can be translated into a standard KKT system.

The scalar projection operator For $\eta_1, \eta_2 \in \mathbb{R}$, a pointwise projection operator $P_{[\eta_1,\eta_2]} : \mathbb{R} \to \mathbb{R}$ is defined by

$$P_{[\eta_1,\eta_2]}(\tau) := \max\big\{\eta_1, \min\{\eta_2, \tau\}\big\}, \qquad \tau \in \mathbb{R}.$$

Extension to vectors For $m \in \mathbb{N}$, $\eta_1, \eta_2 \in \mathbb{R}^m$, $\eta_1 \leq \eta_2$, the projection $P_{[\eta_1,\eta_2]} : \mathbb{R}^m \to \mathbb{R}^m$ is defined by applying P to each component,

$$P_{[\eta_1,\eta_2]}(\tau) := \begin{pmatrix} P_{[\eta_1^1,\eta_2^1]}(\tau_1) \\ \vdots \\ P_{[\eta_1^m,\eta_2^m]}(\tau_m) \end{pmatrix}, \quad \tau \in \mathbb{R}^m.$$

Extension to functions in space or space-time For $m \in \mathbb{N}$ and $\eta_1, \eta_2 \in L^2(\mathcal{Q})^m$, $\eta_1 \leq \eta_2$ almost everywhere, using the definition in b), the projection $P_{[\eta_1,\eta_2]} : L^2(\mathcal{Q})^m \to L^2(\mathcal{Q})^m$ is pointwise defined by

$$P_{[\eta_1,\eta_2]}(f)(t,x) := P_{[\eta_1(t,x),\eta_2(t,x)]}\big(f(t,x)\big), \qquad f \in L^2(\mathcal{Q})^m$$

for almost all $(t,x) \in \mathcal{Q}$.

Reformulation of the problem as KKT system With the projection operator at hand, the KKT system corresponding to the above problem reads (cf. [84, 152]),

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y &= g && \text{at } \Sigma, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \end{aligned}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^\mathsf{T}\lambda + \nabla\xi &= y - z && \text{in } \mathcal{Q}, \\ -\operatorname{div}\lambda &= 0 && \text{in } \mathcal{Q}, \\ \lambda &= 0 && \text{at } \Sigma, \\ \lambda(T,\cdot) &= 0 && \text{in } \Omega, \end{aligned}$$

$$u = P_{[a,b]}\left(-\frac{1}{\alpha}\lambda\right) \qquad \text{in } \mathcal{Q}.$$

After elimination of u (and ignoring the boundary conditions for simplicity), the KKT system reads

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p &= P_{[a,b]}\left(-\frac{1}{\alpha}\lambda\right), \\ -\operatorname{div} y &= 0, \end{aligned} \tag{4.12a}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^\mathsf{T}\lambda + \nabla\xi &= y - z, \\ -\operatorname{div}\lambda &= 0. \end{aligned} \tag{4.12b}$$

The right-hand side of the primal equation (4.12a) is obviously not smooth anymore, which necessitates some changes in the solution algorithm. In particular, the Newton algorithm has to be replaced by a semismooth Newton algorithm (cf. [151, 152]) in order to achieve superlinear convergence. The changes in the discretisation and in the algorithm are as follows.

4.2.2. Discretisation in time

The time discretisation is carried out in the usual way, similar to Section 2.5, 2.6 or 4.1. The only difference is that the operator $-\frac{1}{\alpha}\lambda_n$ has everywhere to be replaced by $P_{[a_n,b_n]}(-\frac{1}{\alpha}\lambda_n)$, with $a_n = a(t_n,\cdot)$ and $b_n = b(t_n,\cdot)$. In the case of the implicit Euler time discretisation scheme, this yields the nonlinear system (2.30) on page 33, where the diagonal submatrices of the matrix $\mathbf{G}$ in (4.7) are replaced by

$$\mathbf{G}_n = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{C}_n & -P_n^\alpha & \mathcal{G} & 0 \\ -\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{N}_n^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}, \tag{4.13}$$

with

$$P_n^\alpha \lambda_n := P_{[a_n,b_n]}\left(-\frac{\lambda_n}{\alpha}\right) \tag{4.14}$$

for $n = 1, ..., N$. If the general θ-scheme is used for the time discretisation, the changes in the system are the same, but here the operator reads

$$P_n^\alpha \lambda_{n-1+\theta} := P_{[a_{n-1+\theta},b_{n-1+\theta}]}\left(-\frac{\lambda_{n-1+\theta}}{\alpha}\right)$$

with

$$a_{n-1+\theta} = a\big((1-\theta)t_{n-1} + \theta t_n\big), \quad b_{n-1+\theta} = b\big((1-\theta)t_{n-1} + \theta t_n\big).$$

For simplicity, the following descriptions concentrate on the implicit Euler case.

4.2.3. The semismooth Newton method

To process nonlinear terms in the KKT system, a Newton or Newton-like method similar to Chapter 3.2 should be used. The standard Newton algorithm consists of two main steps, see Algorithm 3.1 on page 52:

a) Create the nonlinear defect.

b) Apply a linear space-time solver to the defect to calculate an update for the solution.

Introducing the projection operator into the nonlinear system necessitates some changes in the underlying algorithm. The modifications for step a) are rather straightforward, and Section 4.2.4 on page 91 gives an overview about the techniques to create the nonlinear defect. However, step b) is slightly more complicated. The operator P_n^α is nonlinear and nonsmooth and thus, it does not have a Fréchet derivative. Therefore, the assembly of the preconditioner $C(w_i^\sigma)$ in Algorithm 3.1 is not trivial:

a) In the discrete space, the action of the nonlinear operator P_n^α cannot be formulated as matrix-vector product with a linear matrix. As a consequence, the action of

the preconditioner $C(w_i^\sigma) = G^\sigma(w_i^\sigma)$ cannot be formulated as matrix-vector product with a linear matrix, too. The standard fixed point method can therefore not be generalised in a straightforward way to the constrained case. A possible remedy is to apply P_n^α only during the calculation of the nonlinear defect, but to ignore it in setting up $C(w_i^\sigma)$. If the iteration converges, the solution is still correct due to the defect correction scheme.

b) In contrast to the fixed point method in a), a generalisation of the Newton method to constrained control is available. In the discrete space, the action of the preconditioner $C(w_i^\sigma) = F_i^\sigma(w_i^\sigma)$ can be formulated as a matrix-vector product with a linear matrix. However, due to the fact that $w^\sigma \mapsto G^\sigma(w^\sigma)w^\sigma$ is nonsmooth, $F_i^\sigma(w_i^\sigma)$ is not a Fréchet derivative anymore. An appropriate definition and interpretation of $F_i^\sigma(w_i^\sigma)$ involves the concept of *semismoothness* which leads to the 'semismooth Newton' approach, cf. [151, 152].

The following paragraphs give a brief overview about the implementation of the semismooth Newton method in b); for details about the underlying theory, see [151]. Due to the lack of smoothness, this modification of Newton's method does usually not show quadratic convergence anymore; however, numerical tests in later sections show that superlinear convergence can be expected in practice.

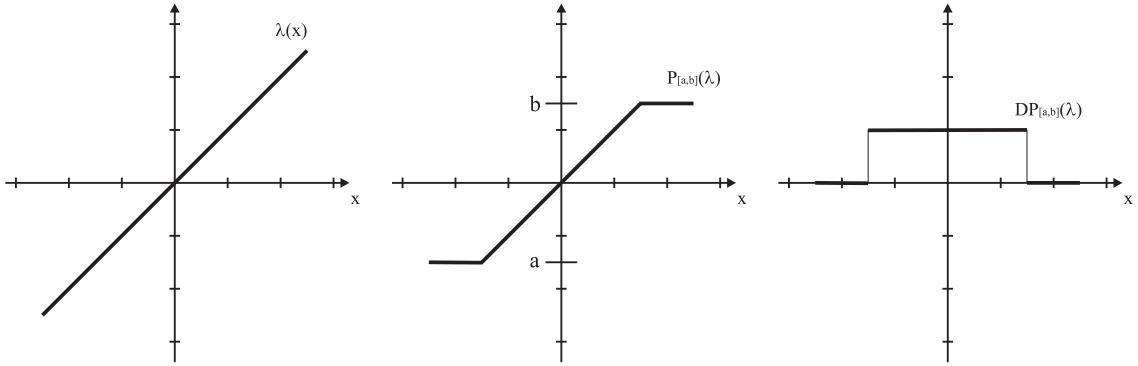

Figure 4.1: Left: A function λ in 1D. Centre: $P_{[a,b]}(\lambda)$. Right: $DP_{[a,b]}(\lambda)$.

At first, it is noted that the projection operator $P_{[\eta_1,\eta_2]}(\cdot)$ is smooth almost everywhere; it is either piecewise linear or constant, see also Figure 4.1. An generalised derivative DP_n^α of the operator P_n^α in timestep $n = 1, ..., N$ and component $i = 1, ..., \dim$ is defined by

$$\left(DP_n^\alpha\right)^i = \left(DP_n^\alpha(\lambda_n)\right)^i = \begin{cases} -\frac{1}{\alpha}\mathcal{I} & \text{, where } a_n^i \le -\frac{1}{\alpha}\lambda_n^i \le b_n^i \\ 0 & \text{, elsewhere,} \end{cases} \tag{4.15}$$

which is one representative of the generalised Newton derivative (cf. [39, 92]). For a fixed n, the operator DP_n^α can be discretised in the nonlinear loop based on λ_n. In the case of the implicit Euler time discretisation scheme, the generalised derivative $\mathbf{F}(w)$ of the mapping $w \mapsto \mathbf{G}(w)w$ is therefore given by matrix (2.33) on page 34, where the diagonal blocks $\mathbf{F}_n$, $n = 1, ..., N$, are replaced by

$$\tilde{\mathbf{F}}_n := \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{N}_n & -DP_n^\alpha & \mathcal{G} & 0 \\ -\mathcal{I} + \mathcal{R}_n & \frac{\mathcal{I}}{k} + \mathcal{N}_n^* & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}. \tag{4.16}$$

The defect correction loop (3.8) on page 51 with this modified generalised derivative matrix used as preconditioner is called 'semismooth Newton method'. In the case of the general θ-scheme, the corresponding change in the derivative matrix is similar, but the coupling matrix of the dual to the primal space reads $DP^{\alpha}_{n-1+\theta}$, $n = 1, ..., N$.

4.1 Remarks. In the case of the heat equation and the Stokes equations, the nonsmoothness of the operator P^{α}_{n} introduces a kind of nonlinearity into the system. Therefore, although the equation is basically linear, the corresponding discrete space-time system has to be embedded into a nonlinear loop. As an example, the heat equation is considered, discretised in time with the implicit Euler scheme, see Section 2.5.3 on page 34. The space-time system (2.34) on page 34 transforms to

$$\mathbf{G}(w)w = f \tag{4.17}$$

which is a nonlinear system. The system matrix $\mathbf{G}(w)$ has still the structure (2.35) on page 35 but the diagonal blocks $\mathbf{G}_n$ are replaced by

$$\tilde{\mathbf{G}}_n := \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & -P^{\alpha}_{n} \\ -\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A} \end{pmatrix} \tag{4.18}$$

for $n = 1, ..., N$. The generalised derivative $\mathbf{F}(w)$ of the mapping $w \mapsto \mathbf{G}(w)w$ is the matrix $\mathbf{G}$ from (2.35) with the diagonal blocks $\mathbf{G}_n$ replaced by

$$\tilde{\mathbf{F}}_n := \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & -DP^{\alpha}_{n} \\ -\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A} \end{pmatrix}. \tag{4.19}$$

4.2.4. Discretisation in space

For the discretisation of the operators P^{α}_{n} and DP^{α}_{n} in space, there are basically two different methods available. On the one hand, a cut-off function is applied during the integration in every cubature point. On the other hand, a postprocessing strategy can be used to change finite element vectors and mass matrices based on λ_n. The latter method is only applicable for finite elements of Lagrangian type while the first one can be applied to general finite element functions integrating a nonsmooth function on each element. A short overview about these methods is given in the following.

Method 1 This method uses the operator P^{α}_{n} pointwise, i. e., in every cubature point, basically ignoring all nonsmoothness. The approach is demonstrated for simplicity based on the heat equation (without end time observation) for an $n \in \{1, ..., N\}$.

As starting point, $y_n, \lambda_n \in V_h$ denotes a given pair of finite element functions on the underlying finite element space V_h. A matrix-vector product of this pair with the discrete counterpart $\tilde{\mathbf{G}}^{h}_{n}$ to $\tilde{\mathbf{G}}_n$ in (4.18) reads

$$d_n := \tilde{\mathbf{G}}^{h}_{n} \begin{pmatrix} y_n \\ \lambda_n \end{pmatrix} = \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{A}_h & (-P^{\alpha}_{n})_h \\ -\mathcal{I}_h & \frac{\mathcal{I}_h}{k} + \mathcal{A}_h \end{pmatrix} \begin{pmatrix} y_n \\ \lambda_n \end{pmatrix} \in V_h^* \times V_h^*.$$

It has been created by replacing the continuous operators $\mathcal{A}, \mathcal{I}, ...$ by their discrete counterparts $\mathcal{A}_h, \mathcal{I}_h, ...$

The operator $(-P^{\alpha}_{n})_h$ cannot be modelled as a matrix in the linear algebra sense since it is nonlinear and implicitly given. Thus, a nonlinear loop is required — even for a

linear equation as PDE constraint — which applies the operator during the creation of the nonlinear defect d_n. To describe the action of this operator in terms of finite element functions, $\varphi_1, \varphi_2, ...$ denotes a basis of the underlying finite element space V_h. The above matrix-vector product can be expressed in the form

$$d_n = \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{A}_h & 0 \\ -\mathcal{I}_h & \frac{\mathcal{I}_h}{k} + \mathcal{A}_h \end{pmatrix} \begin{pmatrix} y_n \\ \lambda_n \end{pmatrix} + \begin{pmatrix} \mu_n \\ 0 \end{pmatrix}$$

with the auxiliary functional $\mu_h \in V_h^*$ given by

$$\mu_n : v \mapsto \left((-P_n^\alpha)_h \lambda_n,\ v\right)_{\Omega_h} \qquad \text{for all } v \in V_h,$$

whereby

$$(-P_n^\alpha)_h \lambda_n = \sum_j \mu^j \varphi_j \in V_h$$

is specified by the coefficients (i. e., degrees of freedom of the finite element function)

$$\mu^j := (-P_n^\alpha \lambda_n, \varphi_j)_{\Omega_h} = \int_{\Omega_h} -P_n^\alpha \lambda_n(x) \varphi_j(x)\, dx.$$

As usual, Ω_h refers to the underlying mesh. Thus, on the linear algebra level, a matrix-vector multiplication of $(y_n, \lambda_n)^\mathsf{T}$ with $\tilde{\mathbf{G}}_n^h$ involves the assembly of the auxiliary 'right-hand side' function μ_n represented by the right-hand side vector $(\mu^1, \mu^2, ...)^\mathsf{T}$. For the Stokes and Navier–Stokes equations, this can be done in the same way by applying the projection to every component of λ_n.

To create a discrete counterpart to DP_n^α, it is noted that similar to (4.15), DP_n^α can be defined as

$$DP_n^\alpha = DP_n^\alpha(\lambda_n) = \begin{cases} -\frac{1}{\alpha}\mathcal{I} & \text{, where } a_n \le -\frac{1}{\alpha}\lambda_n \le b_n \\ 0 & \text{, elsewhere.} \end{cases} \tag{4.20}$$

A characteristic function $\chi_n : \Omega \to \mathbb{R}$ of the ('inactive') set

$$\left\{ x \in \Omega \;\middle|\; a_n(x) \le -\frac{1}{\alpha}\lambda_n(x) \le b_n(x) \right\}$$

is pointwise defined for $x \in \Omega$ by

$$\chi_n(x) := \chi_{[a_n,b_n]}^{\lambda_n,\alpha}(x) := \begin{cases} 1 & \text{, if } a_n(x) \le -\frac{1}{\alpha}\lambda_n(x) \le b_n(x), \\ 0 & \text{, else,} \end{cases} \tag{4.21}$$

The discretisation of DP_n^α yields a matrix M_n^α of mass matrix type. The entries $m_{ij} := (M_n^\alpha)_{ij}$ are defined as integrals over all cells K of the mesh Ω_h,

$$m_{ij} = \int_{\Omega_h} \left(-\frac{\chi_n}{\alpha}\, \varphi_j\, \varphi_i\right)(x)\, dx = \sum_{K \in \Omega_h} \int_K -\frac{\chi_n(x)}{\alpha}\, \varphi_j(x)\, \varphi_i(x)\, dx. \tag{4.22}$$

They can be calculated by usual cubature formulas. In particular,

$$\int_K -\frac{\chi_n}{\alpha}\, \varphi_j(x)\, \varphi_i(x)\, dx \approx \sum_{k=1}^{M} -\frac{\chi_n(x_k)}{\alpha}\, \omega_k\, \varphi_j(x_k)\, \varphi_i(x_k) \tag{4.23}$$

for $M \in \mathbb{N}$ cubature points $x_1, ..., x_M \in \mathbb{R}^{\dim}$ on element $K \in \Omega_h$ and their associated cubature weights $\omega_1, ..., \omega_M \in \mathbb{R}$. The order of the approximation in (4.23) is basically determined by the order of the cubature formula. However, it has to be noted that on some elements, χ_n is discontinuous which leads to a loss in the order of the cubature, see also the remarks below (in particular, Remarks 4.2c)).

In the case of the Stokes and Navier–Stokes equations, the above method has to be applied to every component of λ_n. The discrete counterpart to DP_n^α in (4.15) is a block matrix M_n^α of the form

$$M_n^\alpha = \begin{pmatrix} M_{n,1}^\alpha & & \\ & \ddots & \\ & & M_{n,d}^\alpha \end{pmatrix}, \tag{4.24}$$

with $M_{n,i}^\alpha$ created as above based on λ_n^i, $i = 1, ..., \dim$.

Method 2 For simplicity, the following description again restricts to the heat equation case. The underlying finite element space V_h is assumed to be of Lagrangian type, i. e., for every finite element function $v_h \in V_h$ there is a representation

$$v_h = \sum_j v^j \varphi_j \qquad \text{with} \qquad v^j = v(x_j) \in \mathbb{R} \tag{4.25}$$

for $j \in \mathbb{N}$ on a finite set of vertices $\{x_j\} \subset \Omega$.

For $n \in \{1, ..., N-1\}$, $y_n, \lambda_n \in V_h$ refers to a given pair of finite element functions on the underlying finite element space V_h. A matrix-vector product of this pair with the discrete counterpart $\tilde{\mathbf{G}}_n^h$ to $\tilde{\mathbf{G}}_n$ in (4.18) reads

$$\begin{aligned} d_n := \tilde{\mathbf{G}}_n^h \begin{pmatrix} y_n \\ \lambda_n \end{pmatrix} &= \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{A}_h & (-P_n^\alpha)_h \\ -\mathcal{I}_h & \frac{\mathcal{I}_h}{k} + \mathcal{A}_h \end{pmatrix} \begin{pmatrix} y_n \\ \lambda_n \end{pmatrix} \\ &= \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{A}_h & 0 \\ -\mathcal{I}_h & \frac{\mathcal{I}_h}{k} + \mathcal{A}_h \end{pmatrix} \begin{pmatrix} y_n \\ \lambda_n \end{pmatrix} + \begin{pmatrix} 0 & -\mathcal{I}_h \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ u_n \end{pmatrix} \qquad (4.26a) \\ &= \begin{pmatrix} \frac{\mathcal{I}_h}{k} + \mathcal{A}_h & 0 \\ -\mathcal{I}_h & \frac{\mathcal{I}_h}{k} + \mathcal{A}_h \end{pmatrix} \begin{pmatrix} y_n \\ \lambda_n \end{pmatrix} + \begin{pmatrix} \tilde{\mu}_n \\ 0 \end{pmatrix}, \qquad (4.26b) \end{aligned}$$

with $\tilde{\mu}_n = -\mathcal{I}_h u_n$, $\mathcal{I}_h : V_h \to V_h^*$ denoting the discrete identity. The discrete 'control' u_n, is defined by a 'nodal projection' of $-\frac{1}{\alpha}\lambda_n$ as follows. Due to (4.25), λ_n has the representation

$$\lambda_n = \sum_j \lambda^j \varphi_j.$$

The control $u_n \in V_h$ is calculated by projecting every node in the vector of degrees of freedoms of λ_n, i. e.,

$$u_n = \sum_j u^j \varphi_j, \quad u^j = P_{[a_n(x_j), b_n(x_j)]}\left(-\frac{1}{\alpha}\lambda^j\right).$$

In the Stokes and Navier–Stokes case, this method has to be applied to every component of λ_n.

The discrete counterpart to DP_n^α is again denoted by M_n^α. This matrix can be set up with a similar node-wise approach: M_n^α is defined by the scaled mass matrix $-\frac{1}{\alpha}M_h$ of the underlying finite element space V_h with the rows

$$\left\{ j \in \mathbb{N} \;\middle|\; -\frac{1}{\alpha}\lambda^j < a_n(x_j) \;\text{ or }\; -\frac{1}{\alpha}\lambda^j > b_n(x_j) \right\}$$

replaced by zero. For the Stokes/Navier–Stokes case, this has to be done for every component, resulting again in a block-diagonal matrix M_n^α of the form (4.24).

4.2 Remarks. a) Although the integral mean value based finite element $\tilde{Q}_1$ is not of Lagrangian type, the degrees of freedom can be interpreted as approximation to the values in the edge midpoints and method 2 can be applied. As numerical tests in later chapters show, the resulting approximation still generates an acceptable solution.

b) The main difference between method 1 and method 2 is the discrete identity $\mathcal{I}_h : V_h \to V_h^*$ in front of the auxiliary function u_n for method 2:

- Method 1 calculates the defect d_n using a function $\mu_n \in V_h^*$ which is implicitly defined by λ_n and computed by cubature.
- Method 2 calculates u_n and incorporates it into the defect in the form $\tilde{\mu}_n = -\mathcal{I}_h u_n \in V_h^*$, see (4.26). On the discrete level, this results in a mass matrix multiplied with the coefficient vector of the auxiliary control u_n. If applicable, this strategy is usually faster than method 1 since no integration has to be performed.

In the optimisation context, the difference between the two methods can be interpreted as a different handling of the control: Method 2 discretises the control $u_n = P_{[\cdot,\cdot]}(-\frac{1}{\alpha}\lambda_n)$, while method 1 directly models the action of u_n without discretising it as a function of the underlying finite element space V_h.

c) Method 1 applies the projection based on the cubature points in integral terms. This method has the disadvantage that the term below the integral is nonsmooth on each element that crosses the active set of the constraints. Therefore, it is generally not sufficient to use higher order cubature formulas in order to better resolve the integral on such elements, especially for higher order finite element spaces. There are basically two solutions:

On the one hand, it is possible to use an adapted mesh around the interface where the constraints get active (e. g., adapted by grid deformation [63–65] or h-adaptivity). On the other hand, if an element crosses the border of the active set of the control (i. e., the part of Ω where the constraints are active), adaptive integration can be used. This leads to a piecewise integration on different parts of such an element [45, 49, 102, 138, 156]. A third approach that can be considered as a modification of adaptive integration can be found in [92], where the active set of the constraints is approximated by dividing the elements that cross the border of the active set. Then, the different parts of each element are integrated independently of each other.

All these modifications improve the accuracy of the discretisation but are not expected to fundamentally change the behaviour of the solver. Therefore, the nonsmoothness is not considered further in this work.

d) A third method to deal with P_n^α, which is given here for the sake of reference, was introduced in [121]. Neitzel et al. replaced the nonsmooth projection operator by a regularised, smooth counterpart and showed convergence of the approximate solution for decreasing regularisation parameter.

4.3. Do-nothing and outflow boundary conditions

So far, all distributed optimal control problems have been formulated with pure Dirichlet boundary conditions. Formally, the systems can also be extended to the case that a part of the boundary is characterised by Neumann boundary conditions (in the case of the heat equation) or outflow boundary conditions (in the case of the Stokes/Navier–Stokes equations). Such boundary parts are typically realised by the so-called 'do-nothing' boundary conditions. Although only limited information about this type of boundary conditions can be found in the literature (see for example [66, 81]) — in particular concerning optimal distributed control — the numerical analysis in later chapters will show confirm this type of boundary condition usually works fine.

To give a small introduction, the optimal control of the heat equation, the Stokes equations and the Navier–Stokes equations is considered on a domain Ω. The boundary $\Gamma := \partial\Omega$ is decomposed into two parts, $\Gamma =: \Gamma_d \cup \Gamma_n$ with Γ_d describing the Dirichlet boundary and Γ_n the Neumann/outflow boundary. According to [26, 81, 84, 139, 142], the corresponding KKT systems that include 'do-nothing' boundary conditions at Γ_n read (in strong formulation)

1.) Optimal control of the heat equation

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min! \tag{4.27}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \Delta y &= u && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } (0,T)\times\Gamma_d, \\ \partial_\eta y &= 0 && \text{at } (0,T)\times\Gamma_n, \end{aligned}$$

2.) Optimal control of the nonstationary Stokes equations

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min! \tag{4.28}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \nu\Delta y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && (0,T)\times\Gamma_d, \\ \nu\partial_\eta y - p\eta &= 0 && (0,T)\times\Gamma_n, \end{aligned}$$

3.) Optimal control of the nonstationary Navier–Stokes equations

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min! \tag{4.29}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \nu\Delta y + y\nabla y + \nabla p &= u && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \\ y &= g && \text{at } (0,T)\times\Gamma_d, \\ \nu\partial_\eta y - p\eta &= 0 && \text{at } (0,T)\times\Gamma_n. \end{aligned}$$

In all of the three systems, the same definitions for y, p, u and g hold as in Chapter 2. Here, $\eta : \Gamma \to \mathbb{R}^{\dim}$ stands for the outer unit normal vector of the domain Ω.

Formally, it is possible to derive the corresponding KKT systems using the Lagrange multiplier approach. These read (cf. [61, 84, 137])

1.) Heat equation

$$\begin{aligned} y_t - \Delta y + \frac{1}{\alpha}\lambda &= 0 && \text{in } \mathcal{Q}, \\ y &= g && \text{at } (0,T)\times\Gamma_d, \\ \partial_\eta y &= 0 && \text{at } (0,T)\times\Gamma_n, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \end{aligned}$$

$$\begin{aligned} -\lambda_t - \Delta\lambda - y &= -z && \text{in } \mathcal{Q}, \\ \lambda &= 0 && \text{at } (0,T)\times\Gamma_d, \\ \partial_\eta \lambda &= 0 && \text{at } (0,T)\times\Gamma_n, \\ \lambda(T,\cdot) &= 0 && \text{in } \Omega, \end{aligned}$$

2.) Stokes equations

$$\begin{aligned} y_t - \nu\Delta y + \nabla p + \frac{1}{\alpha}\lambda &= 0 && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y &= g && \text{at } (0,T)\times\Gamma_d, \\ \nu\partial_\eta y - p\eta &= 0 && \text{at } (0,T)\times\Gamma_n, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \end{aligned}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda + \nabla\xi - y &= -z && \text{in } \mathcal{Q}, \\ -\operatorname{div}\lambda &= 0 && \text{in } \mathcal{Q}, \\ \lambda &= 0 && \text{at } (0,T)\times\Gamma_d, \\ \nu\partial_\eta\lambda - \xi\eta &= 0 && \text{at } (0,T)\times\Gamma_n, \\ \lambda(T,\cdot) &= 0 && \text{in } \Omega, \end{aligned}$$

3.) Navier–Stokes equations

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p + \frac{1}{\alpha}\lambda &= 0 && \text{in } \mathcal{Q}, \\ -\operatorname{div} y &= 0 && \text{in } \mathcal{Q}, \\ y &= g && \text{at } (0,T)\times\Gamma_d, \\ \nu\partial_\eta y - p\eta &= 0 && \text{at } (0,T)\times\Gamma_n, \\ y(0,\cdot) &= y^0 && \text{in } \Omega, \end{aligned}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi - y &= -z && \text{in } \mathcal{Q}, \\ -\operatorname{div}\lambda &= 0 && \text{in } \mathcal{Q}, \\ \lambda(t,\cdot) &= 0 && \text{at } (0,T)\times\Gamma_d, \\ \nu\partial_n\lambda - \xi n + (y\,\eta)\lambda &= 0 && \text{at } (0,T)\times\Gamma_n, \\ \lambda(T,\cdot) &= 0 && \text{in } \Omega. \end{aligned}$$

4.3 Remarks. a) For all equations, Dirichlet boundary conditions in the primal equation imply homogeneous Dirichlet boundary conditions in the dual equation.

b) For the heat equation and the Stokes equations, 'do-nothing' boundary conditions in the primal equations imply 'do-nothing' boundary conditions for the dual equation.

c) For the Navier–Stokes equations, the situation is similar. However, the 'do-nothing' boundary conditions in the primal equations imply the additional term $(y\,\eta)\lambda$ in the 'do-nothing' boundary conditions of the dual equation, rising up from a derivation of the term $-y\nabla\lambda$. This boundary term must be assembled into the matrices using a boundary integral. It depends on the velocity y, so during a nonlinear iteration, it can be set up, e. g., with the help of the last nonlinear iterate for y.

d) The nonlinear boundary condition appearing in the optimal control of the Navier–Stokes equations induces a modified Newton operator in the Newton iteration: The above KKT system corresponding to the Navier–Stokes case can formally be written in an operator form,

$$\mathcal{G}(w)w = f,$$

with

$$w := (y, p, \lambda, \xi)^{\mathsf{T}},\ f := (0, 0, 0, -z, 0, 0)^{\mathsf{T}}$$

and

$$\mathcal{G}(w)w := \begin{pmatrix} y_t - \nu\Delta y + y\nabla y + \nabla p + \frac{1}{\alpha}\lambda \\ -\operatorname{div} y \\ \nu\partial_\eta y - p\eta \\ -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi - y \\ -\operatorname{div}\lambda \\ \nu\partial_n\lambda - \xi n + (y\,\eta)\lambda \end{pmatrix},$$

with the third and sixth component applied at the boundary. Hence, the corresponding Fréchet derivative $\mathcal{F}(w)$ of the mapping $w \mapsto \mathcal{G}(w)w$ formally reads

$$\mathcal{F}(w)\bar{w} := \begin{pmatrix} \bar{y}_t - \nu\Delta\bar{y} + y\nabla\bar{y} + \bar{y}\nabla y + \nabla\bar{p} + \frac{1}{\alpha}\bar{\lambda} \\ -\operatorname{div}\bar{y} \\ \nu\partial_\eta\bar{y} - \bar{p}\eta \\ -\bar{\lambda}_t - \nu\Delta\bar{\lambda} - \bar{y}\nabla\lambda - y\nabla\bar{\lambda} + (\nabla\bar{y})^{\mathsf{T}}\lambda + (\nabla y)^{\mathsf{T}}\bar{\lambda} + \nabla\bar{\xi} - \bar{y} \\ -\operatorname{div}\bar{\lambda} \\ \nu\partial_n\bar{\lambda} - \bar{\xi}n + (\bar{y}\,\eta)\lambda + (y\,\eta)\bar{\lambda} \end{pmatrix}$$

for $\bar{w} = (\bar{y}, \bar{p}, \bar{\lambda}, \bar{\xi})$. The additional term $(\bar{y}\,\eta)\lambda$ in the last part of the operator stems from the nonlinearity in the outflow boundary condition and acts only on the outflow. Numerical tests in later chapters skip this term during the assembly of the discrete Newton operator. That way, the KKT system and its Fréchet derivative have the same boundary conditions which simplifies the assembly. In practice, this simplification does not destroy the quadratic convergence of the Newton iteration.

4.4. Semi-explicit time discretisation

As an alternative to the fully implicit time discretisation with the implicit Euler scheme, this section highlights a variant, namely the semi-explicit discretisation of the Navier–Stokes equations in time. This approach was used, e. g., in [11]: The diffusion term is treated implicitly while the nonlinearity is treated explicitly.

For an overview about this method, the optimal control of the nonstationary Navier–Stokes equation is considered, which is expressed in the KKT system

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p &= -\frac{1}{\alpha}\lambda, \\ -\operatorname{div} y &= 0, \end{aligned} \tag{4.30a}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi &= y - z, \\ -\operatorname{div}\lambda &= 0, \end{aligned} \tag{4.30b}$$

complemented by initial/end time/boundary conditions, see also (2.17) on page 28. To derive the corresponding discrete KKT system, the discretisation recipe from [11] can be applied. This involves at first a discretisation in time of the primal equation (4.30a). Using the semi-explicit Euler timestepping scheme, this reads

$$\begin{aligned} \frac{y_n}{k} - \nu\Delta y_n + \nabla p_n &= -\frac{1}{\alpha}\lambda_n + \frac{y_{n-1}}{k} - y_{n-1}\nabla y_{n-1} \\ -\operatorname{div} y_n &= 0 \\ y_0 &= y^0 \end{aligned} \tag{4.31}$$

which is linear for every $n = 1, ..., N$ with $N \in \mathbb{N}$ denoting the number of time intervals and $k = 1/N$. Similar to Section 2.5 on page 31, the remaining steps of the discretisation recipe can be applied to obtain a semi-discrete system. This leads to a system of the form

$$\mathbf{G}(w)w = f$$

with

$$\mathbf{G} = \mathbf{G}(w) = \left(\begin{array}{c|c|c|c|c} \mathbf{G}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{I}}_1 & \mathbf{G}_1 & \hat{\mathbf{I}}_1 & & \\ \hline & \breve{\mathbf{I}}_2 & \mathbf{G}_2 & \hat{\mathbf{I}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{I}}_N & \mathbf{G}_N \end{array}\right) \tag{4.32}$$

and the submatrices defined by

$$\mathbf{G}_0 = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & 0 & \mathcal{G} & 0 \\ 0 & \frac{\mathcal{I}}{k} + \mathcal{A} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}, \qquad \mathbf{G}_n = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & \frac{\mathcal{I}}{\alpha} & \mathcal{G} & 0 \\ -\mathcal{I} & \frac{\mathcal{I}}{k} + \mathcal{A} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}$$

for $n = 1, ..., N$ and

$$\breve{\mathbf{I}}_n = \begin{pmatrix} -\frac{\mathcal{I}}{k} + \mathcal{K}_{n-1} & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}, \qquad \hat{\mathbf{I}}_{n-1} = \begin{pmatrix} 0 & & & \\ & -\frac{\mathcal{I}}{k} - \mathcal{K}_{n-1} + \overline{\mathcal{K}}^*_{n-1} & & \\ & & 0 & \\ & & & 0 \end{pmatrix}.$$

The additional operators needed here and in the following are defined as

$$\begin{aligned}\mathcal{A} &: v \mapsto -\nu\Delta v\\ \mathcal{K}_n = \mathcal{K}(y_n) &: v \mapsto (y_n\nabla)v\\ \overline{\mathcal{K}}_n = \overline{\mathcal{K}}(y_n) &: v \mapsto (v\nabla)y_n\\ \overline{\mathcal{K}}_n^* = \overline{\mathcal{K}}^*(y_n) &: v \mapsto (\nabla y_n)^{\mathsf{T}} v\end{aligned}$$

For the initial condition, the projection scheme has to be changed slightly, so the right-hand side reads

$$f = \Big(\underbrace{(\mathcal{I}/k + \mathcal{A})y^0, 0, 0, 0}_{f_0}, \underbrace{0, -z_1, 0, 0}_{f_1}, \\ ..., \underbrace{0, -z_{N-1}, 0, 0}_{f_{N-1}}, \underbrace{0, -z_N, 0, 0}_{f_N} \Big).$$

The corresponding Fréchet derivative $\mathbf{F}(w)$ of the mapping $w \mapsto \mathbf{G}(w)w$ reads

$$\mathbf{F}(w) = \left(\begin{array}{c|c|c|c|c} \mathbf{F}_0 & \hat{\mathbf{I}}_0 & & & \\ \hline \breve{\mathbf{J}}_1 & \mathbf{F}_1 & \hat{\mathbf{J}}_1 & & \\ \hline & \breve{\mathbf{J}}_2 & \mathbf{F}_2 & \hat{\mathbf{J}}_2 & \\ \hline & & \ddots & \ddots & \ddots \\ \hline & & & \breve{\mathbf{J}}_N & \mathbf{F}_N \end{array}\right), \tag{4.33}$$

with

$$\mathbf{F}_0 = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & 0 & \mathcal{G} & 0 \\ \mathcal{R}_1 & \frac{\mathcal{I}}{k} + \mathcal{A} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}, \qquad \mathbf{F}_n = \begin{pmatrix} \frac{\mathcal{I}}{k} + \mathcal{A} & \frac{\mathcal{I}}{\alpha} & \mathcal{G} & 0 \\ -\mathcal{I} + \mathcal{R}_{n+1} & \frac{\mathcal{I}}{k} + \mathcal{A} & 0 & \mathcal{G} \\ \mathcal{D} & 0 & 0 & 0 \\ 0 & \mathcal{D} & 0 & 0 \end{pmatrix}$$

for $n = 1, ..., N$ and

$$\breve{\mathbf{J}}_n = \begin{pmatrix} -\frac{\mathcal{I}}{k} + \mathcal{K}_{n-1} + \overline{\mathcal{K}}_{n-1} & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{pmatrix}, \quad \hat{\mathbf{J}}_{n-1} = \begin{pmatrix} 0 & & & \\ & -\frac{\mathcal{I}}{k} - \mathcal{K}_{n-1} + \overline{\mathcal{K}}_{n-1}^* & & \\ & & 0 & \\ & & & 0 \end{pmatrix}.$$

The systems for the case of the general θ-scheme can be derived in a similar way.

4.4 Remarks. The advantage of the semi-explicit scheme is the higher numerical stability of the local systems in space since the diagonal blocks do not contain any nonlinear terms. Pressure correction schemes like those introduced in [11] can also easily be applied for instance in smoothers like the FBSimSmoother. On the other hand, the explicit treatment of the nonlinearity usually introduces some kind of CFL condition to the timestep size which necessitates small timesteps for stability reasons, cf. [119]. This fact contradicts the paradigm to use higher order timestep schemes with large timestep sizes in order to reduce the numerical costs by reducing the number of timesteps.

The scheme is of interest because of a possible numerical tradeoff: The advantage of faster or more stable solvers in space (due to the missing nonlinearity in the matrix blocks on the diagonal) faces the disadvantage of smaller timesteps. On the other hand, as it is shown in the numerical tests in later chapters, the timestep is anyway coupled to the space discretisation due to accuracy reasons. Therefore, faster CPU times of the solver for the same numerical accuracy can be expected. Numerical tests in later chapters analyse this effect of the semi-explicit timestepping.

5

Basic numerical analysis of the solver: Heat equation and Stokes equations

The following chapter is the first in a row of four chapters dealing with the numerical analysis of the proposed discretisation and solver strategy. In general, the analysis is structured to cover a large variety of parameters and test cases with increasing complexity. Together with Chapters 6, this chapter focuses on linear equations, namely the heat equation and the Stokes equations, and is restricted to analytical test examples. Chapter 7 and 8 will extend the analysis to non-analytical test problems and nonlinear solver techniques.

The main aim of all the numerical analysis is as follows: *Determine a combination of discretisation and solver configuration maximising the accuracy while at the same time minimising the numerical effort.* This is a minimisation problem of its own and basically influenced by two different choices: The parameter setting of the solver and the choice of the underlying discretisation. In the beginning, these two are treated independently. This chapter fixes different discretisation and focuses on finding proper solver configurations that reduce the numerical effort. In a second step, Chapter 6 concentrates on the discretisation, completely ignoring efficiency issues. Later chapters will treat efficiency and accuracy requirements in a combined way.

Outline

The analysis starts with the KKT system for the heat equation. Section 5.1 defines basic analytical test examples. The different smoothing/preconditioning techniques from Chapter 3 (block Jacobi, block Gauß–Seidel,...) are applied in a one-level, two-level and multilevel fashion to the underlying KKT system for a fixed discretisation. The different approaches are compared with respect to efficiency and robustness and good choices for certain basic solver parameters are determined.

Section 5.2 extends the analysis to more general discretisations, introducing higher order discretisations in space and time. In particular, this involves the analysis of the special prolongation/restriction operators derived for the θ-scheme discretisation. Section 5.3 verifies the experiences of the previous sections also in the case of the Stokes equations. It is shown that there are no fundamental differences to the results obtained for the heat equation, although the stability is slightly more sensitive to the choice of regularisation parameters.

The chapter closes with Section 5.4. This section focuses on the choice of the space-time hierarchy and the multigrid cycle. As described in Section 3.1 on page 48ff, the user has some freedom in the choice of the hierarchy, and this section demonstrates that for the wrong choice of hierarchy and multigrid cycle, the linear complexity of the multigrid

algorithm is lost.

5.1. Basic solver analysis for the heat equation and the Stokes equations

The numerical analysis starts with the definition of simple model problems which are tested for a number of model parameters and solver configurations to verify the basic applicability of the proposed linear solver algorithms.

5.1 Heat equation example 1 Consider the optimal control of the heat equation and focus on the following modified KKT system,

$$\left.\begin{aligned} y_t - \Delta y &= f - \frac{1}{\alpha}\lambda, \\ -\lambda_t - \Delta\lambda &= y - z, \\ \lambda(T) &= \gamma(y(T) - z(T)). \end{aligned}\right\} \tag{5.1}$$

The original KKT system (2.12) on page 26 is a special case of this more general form. The additional function $f : \mathcal{Q} \to \mathbb{R}$ is introduced to allow a simpler definition of analytical reference functions. An additional end time observation introduces the option to control $y(T)$, see also Chapter 4.1 on page 81. The following reference functions are defined in terms of an eigenfunction $w : \mathbb{R}^2 \to \mathbb{R}$, $w(x) := \sin(\pi x_1)\sin(\pi x_2)$, of the Laplace operator in space.

$$\begin{aligned} \bar{y}(t,x) &:= \sin(t\pi/2)\, w(x_1,x_2), \\ \bar{\lambda}(t,x) &:= \big(\sin(t\pi/2) - 1\big)\, w(x_1,x_2). \end{aligned}$$

The corresponding right-hand side functions are calculated by substituting $\bar{y} = y$ and $\bar{\lambda} = \lambda$ in the KKT system, i. e.,

$$\begin{aligned} f &:= \bar{y}_t - \Delta\bar{y} + \frac{1}{\alpha}\bar{\lambda}, \\ z &:= \bar{\lambda}_t + \Delta\bar{\lambda} + \bar{y}. \end{aligned}$$

The domain under consideration is $\Omega := (0,1)^2$ and the time interval $[0,T] := [0,1]$. The initial condition is $y(0,\cdot) = 0$. On the boundary, Dirichlet boundary conditions $y(\cdot,x) = 0$ for all $x \in \partial\Omega$ are prescribed.

5.2 Stokes equations example 1 Consider the optimal control of the Stokes equations and focus on the following modified KKT system,

$$\left.\begin{aligned} y_t - \Delta y + \nabla p &= f - \frac{1}{\alpha}\lambda, \\ -\lambda_t - \Delta\lambda + \nabla\xi &= y - z, \\ \lambda(T) &= \gamma(y(T) - z(T)) \\ -\operatorname{div} y &= \delta_y, \\ -\operatorname{div}\lambda &= \delta_\lambda. \end{aligned}\right\} \tag{5.2}$$

Similar to the previous heat equation example, the additional right-hand side functions $f : \mathcal{Q} \to \mathbb{R}^2$, $\delta_y, \delta_\lambda : \mathcal{Q} \to \mathbb{R}$ allow a simpler definition of analytical reference functions. The parameter γ for the end time condition allows again to control $y(T)$, see also Section 4.1 on page 81ff. Using the eigenfunction $w_2 : \mathbb{R}^2 \to \mathbb{R}$,

$w_2(x) := \sin(\frac{\pi}{2}x_1)\sin(\pi x_2)$ of the Laplace operator in space, the following reference functions are defined.

$$\bar{y}(t,x) := \sin(t\pi/2)\begin{pmatrix} w_2(x_1,x_2) \\ w_2(x_1,x_2)\end{pmatrix},$$
$$\bar{\lambda}(t,x) := \big(\sin(t\pi/2)-1\big)\begin{pmatrix} w_2(x_1,x_2) \\ w_2(x_1,x_2)\end{pmatrix},$$
$$\bar{p}(t,x) := \bar{\xi}(t,x) := 0.$$

Substituting $y = \bar{y}$, $\lambda = \bar{\lambda}$, $p = \bar{p}$ and $\xi = \bar{\xi}$ in (5.2) the associated right-hand side functions f, z, δ_y and δ_λ are computed.

The spatial domain and the underlying time interval are the same as in Example 5.1. At $t = 0$, the initial condition $y(0,\cdot) = 0$ is used, and on the boundary, Dirichlet boundary conditions $y(\cdot,x) = 0$ are prescribed for all $x \in \partial\Omega\backslash\Gamma_2$ with Γ_2 being the right edge. On Γ_2, do-nothing boundary conditions are used. It is noted that for w_2 there holds $\partial_\eta w_2(x) = 0$ for $x \in \Gamma_2$, thus for the reference functions, the natural boundary conditions $\partial_\eta y - p\eta = 0$ for the primal equation and $\partial_\eta\lambda - \xi\eta = 0$ for the dual equation are fulfilled on Γ_2, $\eta : \partial\Omega \to \mathbb{R}^2$ denoting the outer normal unit vector.

5.1.1. Basic single grid solver analysis

The numerical solver analysis starts with linear single-grid solvers. The three solvers FBJacSolver, FBSimSolver and FBGsSolver are used, at first without damping; there is no extra analysis for the FBSorSolver carried out. The stopping criterion of the space-time solver is set to $\varepsilon_{\text{CoarseMG}} = 10^{-10}$, the space discretisation is done with Q_1 and time-discretisation with the implicit Euler scheme. The linear solver in space is an optimised Gaussian elimination solver (cf. [48]). This is suitable since the problem size is small, and it helps to avoid errors that possibly influence the convergence of the space-time solver. (However, the influence of the error in the spatial solver is rather small; this will be shown later.)

Focusing on Example 5.1 involving the heat equation, the underlying coarse mesh in space is the cell $[0,1]\times[0,1]$, the underlying coarse mesh in time is one time interval $[0,T] = [0,1]$. This basic space-time mesh is completely isotropic and simultaneously refined in space and time. On each level, the number of iterations #ite and the convergence rate ρ of the solver are measured, where the latter one is defined as

$$\rho := \left(\frac{\|r_{\#\text{ite}}\|}{\|r_0\|}\right)^{\frac{1}{\#\text{ite}}}.$$

Here, $\|r_i\|$ denotes the l_2 norm of the vector of degrees of freedom of the i-th residual r_i. Table 5.1 shows the results for different regularisation parameters α and γ. In the very simple setting $\alpha = 1.0$, $\gamma = 0.0$, all solvers work fine for every level. For FBJacSolver the number of iterations increases with every additional level which is typical for this kind of solver. FBSimSolver and FBGsSolver on the other hand show even level-independent convergence rates. For the more restrictive setting $\alpha = 0.001$, $\gamma = 1$, FBJacSolver and FBGsSolver still work fine, but the number of iterations increases from level to level. On higher levels, FBGsSolver fails. FBSimSolver does not work at all.

The convergence behaviour changes with the damping parameter. In Table 5.2, a damping parameter of $\omega = 0.5$ is used. Both, FBJacSolver(ω) and FBGsSolver(ω)

			FBJacSolver		FBSimSolver		FBGsSolver	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 1.0$	1/16	1/16	25	3.85E-01	5	3.69E-03	4	7.87E-04
$\gamma = 0.0$	1/32	1/32	43	5.79E-01	5	3.77E-03	4	1.58E-03
	1/64	1/64	78	7.41E-01	5	3.82E-03	4	2.47E-03
	1/128	1/128	147	8.54E-01	5	3.88E-03	5	3.18E-03
$\alpha = 0.001$	1/16	1/16	21	3.28E-01	div	div	10	8.61E-02
$\gamma = 1.0$	1/32	1/32	36	5.20E-01	div	div	18	2.76E-01
	1/64	1/64	69	7.12E-01	div	div	437	9.48E-01
	1/128	1/128	136	8.43E-01	div	div	div	div

Table 5.1: FBJacSolver, FBSimSolver and FBGsSolver on different levels, no damping.

			FBJacSolver		FBSimSolver		FBGsSolver	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 0.001$	1/16	1/16	50	6.28E-01	div	div	33	4.93E-01
$\gamma = 1.0$	1/32	1/32	73	7.29E-01	div	div	33	4.94E-01
	1/64	1/64	129	8.36E-01	div	div	33	4.94E-01
	1/128	1/128	263	9.15E-01	div	div	37	5.30E-01

Table 5.2: FBJacSolver, FBSimSolver and FBGsSolver on different levels, damping with $\omega = 0.5$.

are now unconditionally stable in this example whereas FBSimSolver(ω) still does not work. It is noted that FBGsSolver converges with level-independent convergence rates on all levels. This is remarkable, since it shows that this solver is so strong that for simple test problems, there is no multigrid necessary to achieve level-independent convergence rates. Nevertheless, the behaviour of the solver is level dependent for harder problems which will be shown in later examples.

In a next step, all the three solvers are embedded as preconditioners into a BiCGStab solver, see Table 5.3. Using this technique, the manual choice of a damping parameter is avoided, cf. [108]. The solvers in the next tests are BiCGStab(FBJacPrec), BiCGStab(FBSimPrec) and BiCGStab(FBGsPrec) without damping for $\alpha = 0.001$ and $\gamma = 1$. The effect of this embedding is remarkable: All solvers are stable and converge in only a few number of iterations. BiCGStab(FBJacPrec) shows the worst convergence behaviour, the number of iterations is still doubled with every refinement level. BiCGStab(FBGsPrec) has the best convergence behaviour and BiCGStab(FBSimPrec) is somewhere in the middle. The number of iterations grows slightly with every higher level for these solvers.

Finally, Table 5.3 also illustrates results for the setting $\gamma = 1000$. In that case, BiCGStab(FBJacPrec) and BiCGStab(FBGsPrec) are still stable and BiCGStab(FBGsPrec) converges very rapidly, very similar to the case $\gamma = 1$. BiCGStab(FBSimPrec) fails for higher levels. All in all, it can concluded that BiCGStab(FBGsPrec) is the strongest solver and — together with BiCGStab(FBJacPrec) — a good choice for a coarse grid solver in a multigrid setting. The quality of these solvers if being used as smoothers will be analysed next.

			FBJacPrec		FBSimPrec		FBGsPrec	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 0.001$	1/16	1/16	13	1.36E-01	11	1.22E-01	4	1.48E-03
$\gamma = 1.0$	1/32	1/32	23	3.51E-01	13	1.48E-01	6	1.72E-02
	1/64	1/64	49	6.19E-01	14	1.85E-01	8	3.89E-02
	1/128	1/128	114	8.12E-01	16	2.14E-01	10	8.22E-02
$\alpha = 0.001$	1/16	1/16	13	1.33E-01	70	8.31E-01	4	1.49E-03
$\gamma = 1000$	1/32	1/32	23	3.46E-01	443	9.49E-01	6	2.03E-02
	1/64	1/64	50	6.25E-01	div	div	8	4.01E-02
	1/128	1/128	112	8.11E-01	div	div	10	9.01E-02

Table 5.3: Preconditioned BiCGStab space-time solver. The column headline denotes the preconditioner.

5.1.2. Basic two grid solver analysis

The second set of tests focuses for the first time on a two-grid solver. The test problem under consideration is Example 5.1 involving the heat equation, the solver in space is a Gaussian elimination and the space-time coarse grid solver is BiCGStab(FBGsPrec). The coarse grid solver and the two-grid solver are both configured to reduce the norm of the residual by ten digits, $\varepsilon_{\text{OptMG}} = \varepsilon_{\text{CoarseMG}} = 10^{-10}$. On the fine grid of the two-grid solver, there are four postsmoothing steps of the FBJacSmoother, FBSimSmoother and FBGsSmoother applied, in the first test without damping. Presmoothing is not used. Table 5.4 shows the number of iterations #ite and the convergence rate ρ of the two-grid solver for different refinement levels.

The behaviour is very similar to the one-level case. For the regularisation parameters $\alpha = 1.0$ and $\gamma = 0.0$, the solvers work quite well, but using $\alpha = 0.001$, $\gamma = 1.0$, most of the solver configurations break down. The convergence is also not level-independent, so it is clear that damping has to be used here.

For Table 5.5, the previous test is repeated with a damping of $\omega = 0.5$ for all smoothers. The solvers behave much better. Using FBJacSmoother, the number of iterations still grows slightly (mainly because the refinement level is still too low), but the convergence is clearly not deteriorating anymore. The FBGsSmoother based solver shows a completely level-independent convergence behaviour, very stable for all levels. The FBSimSmoother based solver does not work in this situation.

The third test introduces the preconditioned BiCGStab based smoothers, see Table 5.6. This smoother type does not use any damping parameter. The smoothers in this test example are BiCGStab(FBJacPrec,NSM=4), BiCGStab(FBGsPrec,NSM=4), and BiCGStab(FBGsPrec,NSM=4). All solvers work very well and rather level-independently, converging in a few iterations. The only problem is encountered for the FBSimPrec preconditioner in combination with BiCGStab which fails for a too strong setting of the regularisation parameter γ.

5.1.3. Basic multigrid solver analysis

Multigrid can be seen as a perturbation of the two-grid method. Thus, using a full space-time multigrid approach, the results are be expected to be similar to the two-grid case. In the following, the problem under consideration is again Example 5.1 involving the heat equation, using a BiCGStab(FBGsPrec) space-time coarse grid solver, the Gaussian

			FBJacSm.		FBSimSm.		FBGsSm.	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 1.0$	1/16	1/16	5	9.15E-03	1	3.21E-12	1	6.38E-15
$\gamma = 0.0$	1/32	1/32	9	6.25E-02	1	1.29E-12	1	3.19E-14
	1/64	1/64	16	2.27E-01	1	5.70E-13	1	9.36E-14
	1/128	1/128	29	4.49E-01	1	3.74E-13	1	2.78E-13
$\alpha = 0.001$	1/16	1/16	5	5.49E-03	div	div	2	8.38E-06
$\gamma = 1.0$	1/32	1/32	7	3.28E-02	div	div	4	1.33E-03
	1/64	1/64	13	1.42E-01	div	div	10	8.86E-02
	1/128	1/128	22	3.50E-01	div	div	div	div

Table 5.4: Two grid solver on different levels, smoothing with FBJacSmoother, FBSimSmoother and FBGsSmoother, no damping.

			FBJacSm.		FBSimSm.		FBGsSm.	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 0.001$	1/16	1/16	8	5.15E-02	div	div	8	4.01E-02
$\gamma = 1.0$	1/32	1/32	9	7.23E-02	div	div	8	4.00E-02
	1/64	1/64	12	1.32E-01	div	div	8	4.01E-02
	1/128	1/128	14	1.91E-01	div	div	8	4.01E-02

Table 5.5: Two grid solver on different levels, smoothing with FBJacSmoother, FBSimSmoother and FBGsSmoother, damping with $\omega = 0.5$.

			FBJacPrec		FBSimPrec		FBGsPrec	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 0.001$	1/16	1/16	3	3.15E-05	5	3.26E-03	1	6.16E-14
$\gamma = 1.0$	1/32	1/32	4	1.38E-03	5	3.16E-03	2	2.64E-08
	1/64	1/64	7	2.63E-02	5	4.56E-03	2	3.37E-06
	1/128	1/128	8	5.46E-02	4	1.09E-03	3	2.18E-05
$\alpha = 0.001$	1/16	1/16	2	8.75E-06	div	div	1	1.18E-14
$\gamma = 1000$	1/32	1/32	4	7.33E-04	div	div	2	1.25E-08
	1/64	1/64	6	1.66E-02	div	div	2	4.13E-07
	1/128	1/128	7	3.40E-02	div	div	2	4.41E-06

Table 5.6: Two grid solver on different levels, smoothing with a preconditioned BiCGStab smoother. The column headlines entitle the preconditioner.

			FBJacSm.		FBSimSm.		FBGsSm.	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 1.0$	1/16	1/16	5	9.15E-03	1	3.21E-12	1	6.56E-15
$\gamma = 0.0$	1/32	1/32	9	6.26E-02	1	1.29E-12	1	3.20E-14
	1/64	1/64	16	2.27E-01	1	5.70E-13	1	9.37E-14
	1/128	1/128	29	4.49E-01	1	3.74E-13	1	2.78E-13
$\alpha = 0.001$	1/16	1/16	5	5.49E-03	div	div	2	8.38E-06
$\gamma = 1.0$	1/32	1/32	7	3.26E-02	div	div	4	1.33E-03
	1/64	1/64	13	1.42E-01	div	div	10	9.03E-02
	1/128	1/128	22	3.50E-01	div	div	div	div

Table 5.7: Multigrid solver on different levels, smoothing with FBJacSmoother, FBSimSmoother and FBGsSmoother, no damping.

			FBJacSm.		FBSimSm.		FBGsSm.	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 0.001$	1/16	1/16	8	5.20E-02	div	div	8	4.01E-02
$\gamma = 1.0$	1/32	1/32	9	7.41E-02	div	div	8	4.01E-02
	1/64	1/64	12	1.33E-01	div	div	8	4.03E-02
	1/128	1/128	14	1.90E-01	div	div	8	4.03E-02

Table 5.8: Multigrid solver on different levels, smoothing with FBJacSmoother, FBSimSmoother and FBGsSmoother, damping with $\omega = 0.5$.

elimination as solver for the subproblems in space and stopping the coarse grid and multigrid iteration with a relative stopping criterion of $\varepsilon_{\text{OptMG}} = \varepsilon_{\text{CoarseMG}} = 10^{-10}$. The coarse grid is fixed at level 2 which corresponds to $h = k = 1/2$. The first set of tests uses again the FBJacSmoother, FBSimSmoother and FBGsSmoother smoothers in a V-cycle multigrid and analyses the behaviour of the solver for $\alpha = 1.0$, $\gamma = 0$ and $\alpha = 0.001$, $\gamma = 1.0$, respectively. Table 5.7 contains results for the case of no damping and Table 5.8 for the case of damping with $\omega = 0.5$.

The behaviour of the solver is almost indistinguishable to the two-grid case. For $\alpha = 1.0$, $\gamma = 0.0$, the solver work perfectly, while for $\alpha = 0.001$, $\gamma = 1.0$, the smoother has to be damped. FBSimSmoother fails, while FBGsSmoother features a stable, level-independent convergence if damping is used.

Using a preconditioned BiCGStab solver instead, the convergence behaviour gets more stable, see Table 5.9. All solvers converge nicely as long as the regularisation parameter γ is not too large — in the latter case, BiCGStab(FBSimPrec,...) fails as a smoother (the problem is too hard). The BiCGStab(FBGsPrec,...) smoother is again the most stable, the solver converges in only two iterations.

5.1.4. Basic analysis: Inexact solvers in space

The tests in the previous subsections have all used a Gaussian elimination solver for the subproblems in space, which is appropriate as long as the spatial problems are 'small'. An alternative for 'larger' problems is the use of a multigrid solver in space which asymptotically solves spatial problems with linear complexity. This offers the additional possibility to solve spatial problems inexactly: As long as the convergence behaviour of an outer defect correction scheme does not suffer, solving inner problems only up to a certain accuracy is

			FBJacPrec		FBSimPrec		FBGsPrec	
	h	k	#ite	ρ	#ite	ρ	#ite	ρ
$\alpha = 0.001$	1/16	1/16	3	3.15E-05	5	3.45E-03	1	6.16E-14
$\gamma = 1.0$	1/32	1/32	4	1.38E-03	5	3.36E-03	2	2.64E-08
	1/64	1/64	8	4.42E-02	4	2.47E-03	2	3.37E-06
	1/128	1/128	8	5.58E-02	4	1.91E-03	3	2.18E-05
$\alpha = 0.001$	1/16	1/16	2	8.75E-06	div	div	1	1.18E-14
$\gamma = 1000$	1/32	1/32	4	7.33E-04	div	div	2	1.26E-08
	1/64	1/64	7	2.93E-02	div	div	2	4.13E-07
	1/128	1/128	7	3.48E-02	div	div	2	4.41E-06

Table 5.9: Multigrid solver on different levels, smoothing with a preconditioned BiCGStab smoother. The column headlines entitle the preconditioner.

a simple way to reduce numerical costs.

The following numerical test concentrates on Example 5.1 involving the heat equation. Based on a space-time hierarchy defined as in Table 5.10, a space-time multigrid solver is applied to solve the underlying KKT system with the following four configurations.

a) $\alpha = 1.0$, $\gamma = 0.0$, space-time smoother is FBGsSmoother($\omega = 1.0$, NSM $= 1$).

b) $\alpha = 1.0$, $\gamma = 0.0$, space-time smoother is FBGsSmoother($\omega = 0.5$, NSM $= 1$).

c) $\alpha = 0.001$, $\gamma = 1.0$, space-time smoother is FBGsSmoother($\omega = 0.5$, NSM $= 1$).

d) $\alpha = 0.001$, $\gamma = 1.0$, space-time smoother is BiCGStab(FBGsPrec, NSM $= 1$).

The subsolver in space is on the one hand a Gaussian elimination solver [48] and on the other hand a multigrid solver. The latter one uses a BiCGStab smoother with $\text{NSM}_{\text{space}} = 4$ postsmoothing steps, preconditioned by SSOR. To analyse the influence of solving the spatial subproblems inexactly, the spatial multigrid uses the relative stopping criteria $\varepsilon_{\text{SpaceMG}} = 10^{-1}$, 10^{-2} and 10^{-6}. The results can be seen in Tables 5.11 to 5.14.

It is remarkable that the stopping criterion of the linear solver in space has almost no influence to the global convergence. Although the setting $\varepsilon_{\text{SpaceMG}} = 10^{-1}$ is too weak in some cases, using $\varepsilon_{\text{SpaceMG}} = 10^{-2}$ produces nearly as good convergence results as using the Gaussian elimination. The difference in the convergence behaviour between $\varepsilon_{\text{SpaceMG}} = 10^{-2}$ and $\varepsilon_{\text{SpaceMG}} = 10^{-6}$ is negligible. Therefore, $\varepsilon_{\text{SpaceMG}} = 10^{-2}$ can be seen as the best compromise between stability and efficiency in all solver configurations.

lv.	h	k
1	1/4	1/4
2	1/8	1/8
3	1/16	1/16
4	1/32	1/32
5	1/64	1/64
6	1/128	1/128

Table 5.10: Space-time hierarchy for tests with an inexact solver in space.

	a)		b)		c)		d)	
lv.	#ite	ρ	#ite	ρ	#ite	ρ	#ite	ρ
2	5	8.83E-03	34	4.98E-01	29	4.48E-01	4	1.12E-03
3	5	8.08E-03	34	4.98E-01	29	4.45E-01	4	1.23E-03
4	5	7.75E-03	34	4.98E-01	29	4.45E-01	6	1.30E-02
5	5	7.59E-03	34	4.98E-01	29	4.48E-01	7	2.85E-02
6	5	7.50E-03	34	4.99E-01	29	4.49E-01	7	2.32E-02

Table 5.11: Linear solver in space: Multigrid with stopping criterion $\varepsilon_{\text{SpaceMG}} = 10^{-1}$.

	a)		b)		c)		d)	
lv.	#ite	ρ	#ite	ρ	#ite	ρ	#ite	ρ
2	3	2.05E-04	33	4.93E-01	29	4.46E-01	3	6.34E-05
3	3	1.04E-04	33	4.93E-01	29	4.41E-01	4	7.83E-04
4	3	1.04E-04	33	4.93E-01	29	4.42E-01	5	6.22E-03
5	3	1.44E-04	33	4.94E-01	29	4.45E-01	7	2.85E-02
6	3	2.02E-04	33	4.94E-01	29	4.46E-01	6	1.87E-02

Table 5.12: Linear solver in space: Multigrid with stopping criterion $\varepsilon_{\text{SpaceMG}} = 10^{-2}$.

	a)		b)		c)		d)	
lv.	#ite	ρ	#ite	ρ	#ite	ρ	#ite	ρ
2	3	6.00E-05	33	4.93E-01	29	4.46E-01	3	6.34E-05
3	3	6.83E-05	33	4.93E-01	29	4.41E-01	4	7.52E-04
4	3	8.77E-05	33	4.93E-01	29	4.42E-01	5	6.23E-03
5	3	1.07E-04	33	4.94E-01	29	4.45E-01	7	2.85E-02
6	3	1.63E-04	33	4.94E-01	29	4.46E-01	6	1.88E-02

Table 5.13: Linear solver in space: Multigrid with stopping criterion $\varepsilon_{\text{SpaceMG}} = 10^{-6}$.

	a)		b)		c)		d)	
lv.	#ite	ρ	#ite	ρ	#ite	ρ	#ite	ρ
2	3	5.93E-05	33	4.93E-01	29	4.46E-01	3	6.35E-05
3	3	6.12E-05	33	4.93E-01	29	4.41E-01	4	7.52E-04
4	3	5.03E-05	33	4.93E-01	29	4.42E-01	5	6.23E-03
5	3	4.40E-05	33	4.93E-01	29	4.45E-01	7	2.85E-02
6	3	6.46E-05	33	4.93E-01	29	4.46E-01	6	1.88E-02

Table 5.14: Linear solver in space: Gaussian elimination.

5.2. Higher order discretisations: Q_2 and the Crank–Nicolson scheme

For a discretisation with the Crank–Nicolson scheme in time and/or a discretisation with Q_2 in space, the results are expected to be similar to a discretisation with Q_1 and the implicit Euler scheme. The following numerical results will confirm this. However, it will be shown that the smoother has to be strong enough or the time discretisation has to be fine enough, otherwise the level independent convergence is lost.

5.2.1. Basic multigrid solver analysis

Consider Example 5.1 involving the heat equation, set up with $\alpha = 0.001$ and $\gamma = 0.0$ and discretised with Q_1 in space and the implicit Euler as well as the Crank–Nicolson scheme in time. Table 5.15 shows the convergence properties of a space-time multigrid solver that uses a BiCGStab(FBGsPrec,NSM=2) smoother. The hierarchy in this example is build by 1:1 coarsening in space and time, starting from k and h in the table down to $h_{\text{coarse}} = k_{\text{coarse}} = 1/16$. Similar to the numerical tests in Section 5.1, the space-time solver reduces the norm of the residual by ten digits. A very rapid, level-independent convergence of the solver is visible and there is hardly any difference in the convergence of all the configurations.

However, the situation changes if a weaker smoother is used. Tables 5.16 and 5.17 depict the convergence behaviour of the same space-time multigrid solver on the same level hierarchy for a BiCGStab(FBJacPrec,NSM=2) smoother, once discretised in time with the implicit Euler, once with Crank–Nicolson scheme. As an additional parameter, $T_{\min}$ is introduced: The time interval under consideration is $[T_{\min}, 1]$, divided into #int time intervals. All previous tests adopted $T_{\min} = 0.0$ which was convenient.

Concerning the implicit Euler scheme, the solver converges rather well for $T_{\min} = 0.0$, but the efficiency reduces for $T_{\min} \to 1.0$. The level-independent convergence is nearly lost. This behaviour is a consequence of anisotropy in the space-time mesh (there is $k \ll h$), a more detailed analysis follows later. In the Crank–Nicolson case, the situation is slightly different. For $T_{\min} = 0.0$, the level-independent convergence is lost. It can be regained for $T_{\min} \to 1.0$, although the overall convergence slightly suffers, the closer $T_{\min}$ is to 1.0, which is then again similar to the test with the implicit Euler scheme.

Table 5.18 finally lists the results for a discretisation with Q_2 in space and Crank–Nicolson in time. The convergence behaviour of the solver is essentially the same as in the case of a discretisation with Q_1 in space. There is no fundamental difference visible in comparison to Table 5.17 which was also generated based on the Crank–Nicolson time discretisation.

5.2.2. Prolongation/restriction operators for the Crank–Nicolson scheme

As described in Section 2.6 on page 36, the special time discretisation in the Crank–Nicolson case induces modified prolongation/restriction operators. In practice however, different choices are possible, and the following numerical tests should draw a picture about their applicability.

Similar to the previous tests, the optimal control of the heat equation (cf. Example 5.1) on the unit space-time cylinder is considered. The discretisation is carried out using the Q_1 element in space and the Crank–Nicolson scheme in time. For different refinement levels in space and time, Table 5.19 illustrates the number of iterations and the convergence rate for the following choices of prolongation/restriction operators in time:

	Discr.:	Q_1, impl. Euler		Q_1, Crank–Nic.		Q_2, Crank–Nic.	
h	k	#ite	ρ	#ite	ρ	#ite	ρ
1/32	1/32	2	6.94E-06	2	3.33E-06	2	1.33E-06
1/64	1/64	3	1.70E-04	3	6.93E-05	3	4.02E-05
1/128	1/128	4	5.81E-04	3	2.86E-04	3	1.74E-04

Table 5.15: Space-time multigrid with BiCGStab(FBGsPrec,NSM=2) smoother. Q_1 and Q_2 in space, implicit Euler and Crank–Nicolson scheme in time.

		$T_{\min} = 0.0$		$T_{\min} = 1 - 1/128$		$T_{\min} = 1 - 1/1024$	
h	#int	#ite	ρ	#ite	ρ	#ite	ρ
1/32	32	7	2.20E-02	15	2.07E-01	17	2.46E-01
1/64	64	9	6.20E-02	19	2.94E-01	23	3.55E-01
1/128	128	11	1.16E-01	31	4.75E-01	29	4.47E-01

Table 5.16: Space-time multigrid with BiCGStab(FBJacPrec,NSM=2) smoother, Q_1 in space and the implicit Euler scheme in time. Different $T_{\min}$.

		$T_{\min} = 0.0$		$T_{\min} = 1 - 1/128$		$T_{\min} = 1 - 1/1024$	
h	#int	#ite	ρ	#ite	ρ	#ite	ρ
1/32	32	32	4.73E-01	13	1.44E-01	15	1.90E-01
1/64	64	49	6.24E-01	14	1.75E-01	21	3.23E-01
1/128	128	82	7.48E-01	17	2.44E-01	19	2.95E-01

Table 5.17: Space-time multigrid with BiCGStab(FBJacPrec,NSM=2) smoother, Q_1 in space and the Crank–Nicolson scheme in time. Different $T_{\min}$.

		$T_{\min} = 0.0$		$T_{\min} = 1 - 1/128$		$T_{\min} = 1 - 1/1024$	
h	#int	#ite	ρ	#ite	ρ	#ite	ρ
1/32	32	29	4.47E-01	13	1.48E-01	14	1.84E-01
1/64	64	38	5.45E-01	14	1.85E-01	14	1.77E-01
1/128	128	47	5.97E-01	22	3.47E-01	15	2.11E-01

Table 5.18: Space-time multigrid with BiCGStab(FBJacPrec,NSM=2) smoother, Q_2 in space and the Crank–Nicolson scheme in time. Different $T_{\min}$.

		a)		b)		c)	
h	k	#ite	ρ	#ite	ρ	#ite	ρ
1/8	1/64	8	4.63-02	6	2.12E-02	7	3.07E-02
1/16	1/128	9	6.92-02	6	2.11E-02	9	6.81E-02
1/32	1/256	16	2.33-01	6	1.85E-02	13	1.70E-01
1/64	1/512	>50	—	6	1.54E-02	34	5.07E-01

Table 5.19: Convergence for different prolongation/restriction operators in time.

a) Linear prolongation/full weighted restriction similar to the implicit Euler case (see Section 3.4.3 on page 56). The special time discretisation is ignored.

b) Linear prolongation/full weighted restriction as defined for the θ-scheme in Section 3.4.4 on page 58. Natural restriction: The roles of the prolongation matrices are changed for the definition of the restriction, see (3.21) on page 62.

c) Linear prolongation/full weighted restriction as defined for the θ-scheme in Section 3.4.4. The restriction uses the 'intuitive' approach that the restriction matrices are created by transposing the prolongation matrices *without* changing the roles, see (3.20) on page 61.

To show the effect of the modified time prolongation/restriction more clearly, the results in Table 5.19 are computed with a pure two-grid algorithm in time: The columns entitled 'k' and 'h' again define the resolution of the fine grid on the space-time cylinder, and the coarse grid is generated by one pure time coarsening. The coarse grid solver is adjusted to gain five digits and on the fine grid, FBGSSMOOTHER($\omega = 0.5$, NSM $= 4$) is used for postsmoothing; presmoothing is not applied. The other parameters in these tests are chosen as $\alpha = 0.001$, $\gamma = 0.0$.

It can be seen from Table 5.19 that all methods perform rather well for lower levels. For higher refinement levels however, the special prolongation/restriction operators that exchange the roles of the prolongation matrices (variant b)) are advantageous. It is the only method in this example which preserves the level-independent convergence for all considered refinement levels.

5.3. From the heat equation to the Stokes equations

Consider Example 5.2 on page 102 involving the Stokes equations. The corresponding KKT system is discretised on the one hand with $\tilde{Q}_1/Q_0$ in space and the implicit Euler scheme in time, on the other hand with Q_2/P_1^{disc} in space and the Crank–Nicolson scheme in time. For the basic coarse mesh, $(h,k)_{\text{coarse}} = (1/8, 1/8)$ is used. The regularisation parameters[1] are chosen as $\alpha = 1.0$, $\gamma = 0$ as well as $\alpha = 0.01$, $\gamma = 10.0$. Although this type of problem exhibits a saddle-point character in space and must therefore be treated with PSCSMOOTHER type smoothers, the equations are still linear and comparable to the heat equation; for that reason, the solver behaviour is expected to be similar as well.

The Tables 5.20 and 5.21 summarise the basic solver behaviour. For this test, a BICGSTAB(FBJACPREC,NSM=2) and a BICGSTAB(FBGSPREC,NSM=2) smoother is used

[1] Note that for a stronger setting of α and γ, the solver starts to get instable, which can also be seen in the results in later chapters (see in particular Section 7.1.2 at page 133). Therefore, the tests in this chapter restrict to values for these regularisation parameters which are not too strong but still show a characteristic solver behaviour.

for postsmoothing on every level. Presmoothing is not used. For the $\tilde{Q}_1/Q_0$/implicit Euler discretisation, all smoothers show a rather level-independent convergence behaviour, and the BiCGStab(FBGsPrec,NSM=2) smoother outperforms the BiCGStab(FBJacPrec,NSM=2) smoother similar to the heat equation case. For the Q_2/P_1^{disc}/Crank–Nicolson discretisation on the other hand, the BiCGStab(FBJacPrec,NSM=2) smoother is not strong enough, the convergence is not level-independent. The BiCGStab(FBGsPrec,NSM=2) smoother on the other hand shows a perfect, level-independent convergence behaviour. All in all, the convergence properties of the proposed solver methodology in the Stokes case is similar to the heat equation case.

		$\alpha = 1.0, \gamma = 0$				$\alpha = 0.01, \gamma = 10$			
		FBJacPrec		FBGsPrec		FBJacPrec		FBGsPrec	
h	k	#ite	ρ	#ite	ρ	#ite	ρ	#ite	ρ
1/16	1/16	4	2.39E-03	2	6.85E-07	5	3.63E-03	2	1.89E-06
1/32	1/32	6	1.22E-02	2	9.31E-07	6	1.43E-02	2	8.50E-06
1/64	1/64	7	3.49E-02	2	1.80E-06	8	4.33E-02	3	1.00E-04
1/128	1/128	9	6.88E-02	2	4.53E-06	10	8.39E-02	4	7.72E-04

Table 5.20: Convergence of the space-time multigrid method for the Stokes equations. $\tilde{Q}_1/Q_0$/implicit Euler discretisation. BiCGStab smoother.

		$\alpha = 1.0, \gamma = 0$				$\alpha = 0.01, \gamma = 10$			
		FBJacPrec		FBGsPrec		FBJacPrec		FBGsPrec	
h	k	#ite	ρ	#ite	ρ	#ite	ρ	#ite	ρ
1/16	1/16	17	2.46E-01	1	8.80E-11	43	5.80E-01	3	2.19E-04
1/32	1/32	27	4.13E-01	2	1.68E-06	65	7.01E-01	3	1.06E-04
1/64	1/64	42	5.75E-01	1	9.59E-11	104	7.97E-01	3	1.58E-04
1/128	1/128	76	7.42E-01	2	6.99E-06	212	8.96E-01	4	1.68E-03

Table 5.21: Convergence of the space-time multigrid method for the Stokes equations. Q_2/P_1^{disc}/Crank–Nicolson discretisation. BiCGStab smoother.

5.4. The choice of the multigrid cycle

One key point of a general multigrid scheme is the linear complexilty of the algorithm, i. e., the numerical effort for solving a linear system should be proportional to the number of unknowns. Given a mesh hierarchy with levels $1, 2, 3, ..., \text{NLMAX} =: n$ this means that the total CPU time $T_{\text{total}}(n)$ which is necessary to solve the system at level $n \in \mathbb{N}$ is proportional to the number of unknowns $\text{NEQ}(n)$ on that level, i. e., there are some $c_1, c_2 > 0$ with[2]

$$c_1 \ \text{NEQ}(n) \leq T_{\text{total}}(n) \leq c_2 \ \text{NEQ}(n) \qquad \forall\, n \in \mathbb{N}.$$

The time $T_{\text{total}}(n)$ can be decomposed into

[2] The number of unknowns is usually large, already for moderate mesh resolutions. As an example, consider the optimal distributed control of the Stokes equations, Example 5.2 on page 102. On a regular 2D mesh with 128×128 cells in space and 128 time intervals, if the problem is discretised with Q_2/P_1^{disc} in space and the implicit Euler scheme in time, there are $\text{NEQ} \approx 46 \cdot 10^6$ fully coupled unknowns on the space-time cylinder. In general, c_1 and c_2 should therefore be small to be able to solve such a problem in reasonable time.

- number of iterations #ite, multiplied with
- the time $T_{\text{smooth}}(2,...,n)$ necessary for smoothing on all levels, plus
- the time T_{coarse} necessary for coarse grid solving,

which gives the formula

$$T_{\text{total}}(n) = \#\text{ite } (T_{\text{smooth}}(2,...,n) + T_{\text{coarse}}).$$

For simplicity, the following idealised scenario is assumed: The number of iterations #ite of the multigrid solver is constant, independent of the refinement level. Furthermore, the time T_{coarse} for coarse grid solving is constant. The total solver time is therefore determined by the time necessary for smoothing, $T_{\text{smooth}}(2,...,n)$. $T_{\text{smooth}}(2,...,n)$ finally depends on the number of smoothing steps on each level, the type of the cycle that is used in multigrid and the time that is necessary to perform one smoothing step.

The number of smoothing steps is denoted by NSM, which is assumed to be constant for all levels, and T_s^l refers to the mean time necessary for NSM smoothing step on level $l = 2,...,n$. Then, the time for smoothing on level n depends on the cycle; for the special case of the V- and W-cycle, there is

$$\text{a) V-cycle:} \qquad T_{\text{smooth}}(2,...,n) = T_s^n + T_s^{n-1} + T_s^{n-2} + ... + T_s^2 \tag{5.3}$$

$$\text{b) W-cycle:} \qquad T_{\text{smooth}}(2,...,n) = T_s^n + 2T_s^{n-1} + 4T_s^{n-2} + ... + 2^{n-2}T_s^2. \tag{5.4}$$

The time for smoothing T_s^n can be assumed to grow linearly in n; this is reasonable since usually, all operations in the smoother work with approximately sparse matrices, e. g., matrix-vector multiplications. It follows that the algorithm has linear complexity if

$$T_{\text{smooth}}(2,...,n) \leq c\, T_s^n$$

is fulfilled for some constant $c > 0$ and $n \in \mathbb{N}$ arbitrary. Unfortunately, in the case of the space-time multigrid scheme discussed here, this is not always the case.

5.4.1. Multigrid cycle analysis – in theory

The space-time multigrid scheme proposed in the previous chapters allows some freedom in the choice of the space-time hierarchy, see Section 3.1.2 on page 49. With increasing space-time level, the definition of the hierarchy below the maximum level can be created by different coarsening strategies. Keeping the spatial mesh fixed for all space-time levels, e. g., allows to define a pure time-multigrid algorithm, but one can also think about other variants like one space-coarsening every two time-coarsenings or similar.

In the following, the term 'time-multigrid' stands for a space-time multigrid with semi-coarsening in time, while the term 'full space-time multigrid' denotes the case that the hierarchy is created by 1:1 coarsening. Furthermore, 'spatial space-time multigrid' refers to the situation where the hierarchy is created by semi-coarsening in space, see also Section 3.1.2 at page 49f.

The choice of the coarsening has direct influence to the time needed to perform one space-time smoothing step. At first, the time-multigrid and the full space-time multigrid are considered. In this case, level n in the hierarchy has twice as many timesteps as level $n-1$ (due to the regular refinement/coarsening in time). Depending on the dimension of

the spatial mesh, T_s^{n-1} can therefore roughly be estimated in terms of T_s^n as follows:

a) Time-multigrid, 2D and 3D: $T_s^{n-1} \approx 1/2\ T_s^n$
b) Full space-time multigrid, 2D: $T_s^{n-1} \approx 1/8\ T_s^n$
c) Full space-time multigrid, 3D: $T_s^{n-1} \approx 1/16\ T_s^n$

On the other hand, in the case of spatial space-time multigrid, which keeps the time level fixed, T_s^{n-1} can be estimated by

d) Spatial space-time multigrid, 2D: $T_s^{n-1} \approx 1/4\ T_s^n$
e) Spatial space-time multigrid, 3D: $T_s^{n-1} \approx 1/8\ T_s^n$

This allows to estimate the sums in (5.3) and (5.4):

a) Pure time multigrid, 2D and 3D:

$$\begin{aligned} \text{V-cycle:} \quad & (T_s^n + T_s^{n-1} + T_s^{n-2} + ... + T_s^2) \\ & \approx (1 + 1/2 + 1/4 + ...)T_s^n \qquad \leq 2\ T_s^n, \end{aligned}$$

$$\begin{aligned} \text{W-cycle:} \quad & (T_s^n + 2T_s^{n-1} + 4T_s^{n-2} + ... + 2^{n-1}T_s^2) \\ & \approx (1 + 1 + ... + 1)T_s^n \qquad = (n-1)\ T_s^n. \end{aligned}$$

In the same way, there is:

b) Full space-time multigrid, 2D:

$$\begin{aligned} \text{V-cycle:} \quad & (T_s^n + T_s^{n-1} + T_s^{n-2} + ... + T_s^2) \quad \lessapprox 8/7\ T_s^n, \\ \text{W-cycle:} \quad & (T_s^n + 2T_s^{n-1} + 4T_s^{n-2} + ... + 2^{n-1}T_s^2) \quad \lessapprox 4/3\ T_s^n \end{aligned}$$

c) Full space-time multigrid, 3D:

$$\begin{aligned} \text{V-cycle:} \quad & (T_s^n + T_s^{n-1} + T_s^{n-2} + ... + T_s^2) \quad \lessapprox 16/15\ T_s^n, \\ \text{W-cycle:} \quad & (T_s^n + 2T_s^{n-1} + 4T_s^{n-2} + ... + 2^{n-1}T_s^2) \quad \lessapprox 8/7\ T_s^n \end{aligned}$$

d) Spatial space-time multigrid, 2D:

$$\begin{aligned} \text{V-cycle:} \quad & (T_s^n + T_s^{n-1} + T_s^{n-2} + ... + T_s^2) \quad \lessapprox 4/3\ T_s^n, \\ \text{W-cycle:} \quad & (T_s^n + 2T_s^{n-1} + 4T_s^{n-2} + ... + 2^{n-1}T_s^2) \quad \lessapprox 2\ T_s^n \end{aligned}$$

e) Spatial space-time multigrid, 3D:

$$\begin{aligned} \text{V-cycle:} \quad & (T_s^n + T_s^{n-1} + T_s^{n-2} + ... + T_s^2) \quad \lessapprox 8/7\ T_s^n, \\ \text{W-cycle:} \quad & (T_s^n + 2T_s^{n-1} + 4T_s^{n-2} + ... + 2^{n-1}T_s^2) \quad \lessapprox 4/3\ T_s^n. \end{aligned}$$

Consequently, the time-multigrid in combination with the W-cycle does *not* have linear complexity as the coefficient in front of T_s^n increases with the number of grid levels. Pure time multigrid in combination with the V-cycle is theoretically the most expensive one (from those solver configurations that have linear complexity), the full multigrid costs twice as much time as the smoothing on the finest level. In case of the full space-time multigrid, the choice of the cycle is less significant.

The following numerical test shows that the bounds are indeed sharp and can also be observed in practice. The case of the time-multigrid and full space-time multigrid serve as an example.

5.4.2. Multigrid cycle analysis – in practice

Consider Example 5.1 involving the heat equation, which is an example in 2D. The coarse mesh in space consists of only one cell $[0,1] \times [0,1]$ and the coarse mesh in time consists of one time interval $[0,T] = [0,1]$. Two mesh hierarchies are generated by coarsening according to Table 5.22. Hierarchy a) represents a mesh generated by coarsening only in time with a fixed fine grid in space, while hierarchy b) stands for a full space-time coarsening. The solver in this example is the proposed space-time multigrid where the stopping criterion is chosen to reduce the norm of the initial residual by ten digits, $\varepsilon_{\text{OptMG}} = 10^{-10}$. The smoother in this test is BiCGStab(FBGsPrec,NSM) with NSM=2 (post-)smoothing steps. The problems in space are so small that it is possible to use a optimised Gaussian elimination solver (cf. [48]) without running into the region of superlinear runtime.

	Hierarchy a) for time-MG				Hierarchy b) for full space-time MG			
lv.	space-lv.	time-lv.	#intervals	#cells	space-lv.	time-lv	#intervals	#cells
1	8	2	2	16384	4	2	2	64
2	8	3	4	16384	5	3	4	256
3	8	4	8	16384	6	4	8	1024
4	8	5	16	16384	7	5	16	4096
5	8	6	32	16384	8	6	32	16384

Table 5.22: Space-time hierarchies for the multigrid cycle test. For every space-time level 'lv.': Corresponding space-level, time-level, number of cells in space and number of time intervals.

	Time-MG				Full space-time MG			
	V-cycle		W-cycle		V-cycle		W-cycle	
	#ite	CPU	#ite	CPU	#ite	CPU	#ite	CPU
$T_s^5 = T_{\text{smooth}}(5,...,5)$	3	455	3	457	3	455	3	457
$T_{\text{smooth}}(4,...,5)$	3	690	3	925	3	495	3	532
$T_{\text{smooth}}(3,...,5)$	3	814	3	1321	3	502	3	544
$T_{\text{smooth}}(2,...,5)$	3	889	3	1636	3	502	3	549
$\frac{T_{\text{smooth}}(2,...,5)}{T_{\text{smooth}}(5,...,5)} \approx$	2		$(n-1) = 4$		8/7		4/3	

Table 5.23: CPU time in seconds used for smoothing in the multigrid cycle test.

The complete hierarchy in this test contains five levels, the space-time fine mesh at level 5 is fixed. For $l = 2,...,5$, a space-time multigrid is applied on levels $l-1,...,5$, i. e., on level $l-1$, the coarse grid solver is applied and from level l on the smoothers. The test aims at computing total CPU time in seconds used for smoothing on all levels of this hierarchy, $T_{\text{smooth}}(l,...,5)$. The time for solving coarse grid problems is neglected.

Table 5.23 illustrates the number of multigrid iterations and the total CPU time for smoothing in the case of the time-multigrid and the full space-time multigrid, where in both cases the V-cycle and the W-cycle are tested. The solver needs 3 iterations in all situations independent of whether the coarse grid stems from a pure time coarsening or coarsening in space and time. Furthermore, the predicted computational time is very well reproduced in practice: For the pure time multigrid using the W-cycle the CPU time increases by $T_{\text{smooth}}(5,...,5) \approx 400...450$ seconds with every additional level in the hierarchy. The

V-cycle on the other hand remains bounded by $2 \cdot 450$ seconds. In the case of the full space-time multigrid, the CPU time for the V-cycle remains bounded by $8/7 \cdot 450$ seconds and for the W-cycle by $4/3 \cdot 450$ seconds.

5.5. Summary and conclusions

This chapter has concentrated on the basic analysis of the linear solver. Based on model problems for the heat equation and the Stokes equations, the different space-time solvers, smoothers and preconditioners have been analysed with respect to their robustness in a one-level, two-level and multilevel context.

- As expected, by a proper setting of the solver parameters, the multigrid solver strategy converges with level independent-convergence rates. The proposed smoothing and preconditioning algorithms work well and robust for a large variety of model problems, and the BiCGStab(FBGsPrec,...) smoother turned out to be the strongest one.
- Local problems in space can be solved inexactly without disturbing the global convergence. This allows to significantly reduce the CPU time.
- Concerning higher order discretisations it has been shown that the correct choice of the prolongation/restriction operator is crucial. In particular for the general θ-scheme, the restriction in time has to be modified as suggested in Section 3.4 on page 53ff.

But although the multigrid solver converges with level-independent convergence rates, it does not necessarily converge with linear complexity. It has been demonstrated that improper combinations of mesh hierarchy and multigrid cycle destroy the linear complexity of the algorithm.

The next chapter will continue with the numerical analysis of the heat equation and the Stokes equations. In contrast to this chapter, the discretisation will be in the focus of Chapter 6. It will be demonstrated that the space discretisation has to be coupled to the time discretisation in order to increase the global accuracy upon refinement of the mesh.

6

Numerical analysis of the discretisation: Heat equation and Stokes equations

The previous chapter has started the numerical analysis and concentrated on the solver and its components. For fixed space-time discretisations, the solver has been analysed with respect to efficiency and robustness. This chapter continues the numerical analysis, but in contrast, all solver issues are neglected. The main focus here is the proper choice of the discretisation. Advantages and disadvantages of different combinations of the discretisation techniques proposed in Chapter 2 are emphasised. Similar to Chapter 5, the analysis is again based on analytical test examples involving the heat equation and the Stokes equations.

The aim of this chapter is twofold: On the one hand, the correctness of the discretisation is verified. In particular it is shown with the help of numerical examples that the modified Crank–Nicolson time discretisation introduced in Chapter 2 is second order accurate. On the other hand, there are a couple of important facts repeated from the literature to give a practitioner some advice how to set up a discretisation and a proper space-time mesh. This should help to quantify, e. g., the advantages of higher order discretisations in space and time.

Outline

The chapter provides three sections. After the definition of an additional analytic test example in Section 6.1, Section 6.2 concentrates on the accuracy of the space-time discretisation. For the heat equation as well as for the Stokes equations it is verified that the accuracy of the space discretisation must be coupled to the accuracy of the time discretisation in order to guarantee global convergence. This is a well known fact from the literature. However, to the best of the author's knowledge, the error bounds have not been verified in numerical examples for the optimal control of the Stokes equations. In particular it is shown in this section that the modified Crank–Nicolson time discretisation is second order accurate.

The remaining Section 6.3 focuses on the modified Crank–Nicolson discretisation and compares the accuracy of this scheme to the traditional Crank–Nicolson scheme, i. e., the Crank–Nicolson scheme with the dual solution located at the endpoints of the time interval. Both schemes are second order accurate, but the analysis also shows that for the same resolution of the space-time mesh, the solutions are similarly 'close' to each other, i. e., not only in the limit for $h, k \to 0$.

6.1. Notations and additional test examples

For a combination of h and k, there is $\sigma := (h, k)$ and v^σ the discrete counterpart to a function $v \in L^2(\mathcal{Q})$ under a given space-time discretisation.

The following test example is similar to Example 5.2 on page 102, but uses a nonconstant, time-dependent pressure:

6.1 Stokes equations example 2 Consider the optimal distributed control of the Stokes equations (without end time observation) for $\nu = 1$, expressed in the following modified KKT system,

$$\begin{aligned} y_t - \Delta y + \nabla p &= f - \frac{1}{\alpha}\lambda, \\ -\lambda_t - \Delta\lambda + \nabla\xi &= y - z, \\ \lambda(T) &= 0, \\ -\operatorname{div} y &= \delta_y, \\ -\operatorname{div}\lambda &= \delta_\lambda, \end{aligned}$$

with δ_y, $\delta_\lambda : \mathcal{Q} \to \mathbb{R}$ and $f : \mathcal{Q} \to \mathbb{R}^2$. Similar to Example 5.2, the underlying domain is the unit square $\Omega = (0,1)^2$ on the time interval $[0, T] = [0, 1]$. The following reference functions are defined in terms of an eigenfunction $w_2 : \mathbb{R}^2 \to \mathbb{R}$, $w_2(x) := \sin(\frac{\pi}{2}x_1)\sin(\pi x_2)$, of the Laplace operator in space:

$$\begin{aligned} y(t,x) &:= \sin(t\pi/2)\begin{pmatrix} w_2(x_1,x_2) \\ w_2(x_1,x_2)\end{pmatrix}, \\ \lambda(t,x) &:= \big(\sin(t\pi/2) - 1\big)\begin{pmatrix} w_2(x_1,x_2) \\ w_2(x_1,x_2)\end{pmatrix}, \\ p(t,x) &:= \xi(t,x) := \sin(t\pi/2)\, w_2(x_1,x_2). \end{aligned}$$

The corresponding right-hand side functions are calculated using the KKT system,

$$\begin{aligned} f &:= y_t - \Delta y + \nabla p + \frac{1}{\alpha}\lambda, \\ z &:= \lambda_t + \Delta\lambda - \nabla\xi + y, \\ \delta_y &:= -\operatorname{div} y, \\ \delta_\lambda &:= -\operatorname{div}\lambda. \end{aligned}$$

Boundary conditions are defined as in Example 5.2.

6.2. Coupling of the space and the time discretisation

Following [115, 116], the choice of the space and time discretisation for a space-time problem should be coupled. By (y, λ), a solution of the continuous KKT system (2.1) on page 22 for the heat equation is denoted, and $(y^\sigma, \lambda^\sigma)$ is the solution of its discrete counterpart. In the special case of linear and quadratic finite elements in space and a time discretisation with the implicit Euler or Crank–Nicolson scheme, the following error estimates can be derived under proper assumptions:

a) Q_1 in space, implicit Euler in time:

$$||y - y^\sigma||_{\mathcal{Q}} = \mathcal{O}(k) + \mathcal{O}(h^2), \quad ||\lambda - \lambda^\sigma||_{\mathcal{Q}} = \mathcal{O}(k) + \mathcal{O}(h^2)$$

b) Q_1 in space, Crank–Nicolson in time:

$$||y - y^\sigma||_\mathcal{Q} = \mathcal{O}(k^2) + \mathcal{O}(h^2), \quad ||\lambda - \lambda^\sigma||_\mathcal{Q} = \mathcal{O}(k^2) + \mathcal{O}(h^2)$$

c) Q_2 in space, implicit Euler in time:

$$||y - y^\sigma||_\mathcal{Q} = \mathcal{O}(k) + \mathcal{O}(h^3), \quad ||\lambda - \lambda^\sigma||_\mathcal{Q} = \mathcal{O}(k) + \mathcal{O}(h^3)$$

d) Q_2 in space, Crank–Nicolson in time:

$$||y - y^\sigma||_\mathcal{Q} = \mathcal{O}(k^2) + \mathcal{O}(h^3), \quad ||\lambda - \lambda^\sigma||_\mathcal{Q} = \mathcal{O}(k^2) + \mathcal{O}(h^3)$$

Similar estimates can be expected for the Stokes equations as well, but due to the fact that the trial space for the pressure is one order lower (tests use $\tilde{Q}_1/Q_0$ and Q_2/P_1^{disc}, respectively, for velocity and pressure), the order of the error in space should be one order lower. The following examples show that this coupling between the space and the time error can indeed be seen in practice as well.

6.2.1. Space-time error for the heat equation

Consider at first Example 5.1 involving the heat equation, discretised with Q_1/implicit Euler, Q_1/Crank–Nicolson and Q_2/Crank–Nicolson in space and time, respectively.[1] The regularisation parameters are chosen as $\alpha = 0.001$ and $\gamma = 0.0$ and the coarse mesh is defined as a fully isotropic quadrilateral space-time mesh with $(h, k)_{\text{coarse}} = (1/16, 1/16)$.

For the test, the coarse mesh is refined at first only in space, second only in time and third in space and time according to the theoretical between h and k. For a discretisation with Q_1/implicit Euler for example, the simultaneous refinement in space and time would apply two refinements in time per refinement in space, which corresponds to $h \sim k^2$.

Figures 6.1 to 6.3 depict the error $||y - y^\sigma||_\mathcal{Q}$ and $||\lambda - \lambda^\sigma||_\mathcal{Q}$ for the different refinement strategies. Obviously, pure space refinement or pure time refinement is not the appropriate way to decrease the error: From a certain point on, the error stagnates or even increases. Refining in both space and time however, the error decreases as predicted.

6.2.2. Space-time error for the Stokes equations

Consider the Stokes equations example 2, Example 6.1, with the regularisation parameter set to $\alpha = 0.01$. Figures 6.4 and 6.5 illustrate the errors $||y - y^\sigma||_\mathcal{Q}$, $||\lambda - \lambda^\sigma||_\mathcal{Q}$, $||p - p^\sigma||_\mathcal{Q}$ and $||\xi - \xi^\sigma||_\mathcal{Q}$ for a discretisation with $\tilde{Q}_1/Q_0$ in space and the Crank–Nicolson scheme in time. The coarse mesh is again defined as a fully isotropic space-time mesh with $(h, k)_{\text{coarse}} = (1/16, 1/16)$ and properly refined: For measuring the error in the velocity, the refinement is done according to $h \sim k$ as an error reduction of $\mathcal{O}(h^2) + \mathcal{O}(k^2)$ is expected. For the errors in the pressure on the other hand, a reduction of $\mathcal{O}(h) + \mathcal{O}(k^2)$ is expected, so this test uses a set of meshes based on the relation $h \sim k^2$ to compensate for the different order in the reduction of the error in space and time. The behaviour of the error is similar to the results in Section 6.2.1 for the heat equation: Refining only in space or only in time, the error stagnates, while refining in both space and time according to the theoretical relation between k and h, the error reduces as expected.

[1] Results for the Q_2/implicit Euler discretisation are not presented. There are no fundamental differences to the Q_1/implicit Euler discretisation except for the memory consumption: c) implies that one refinement in space would have to be accompanied by three refinements in time. This would lead to an extremely high number of timesteps and very high memory requirements on higher levels (see also Section 6.2.3), rendering the discretisation unattractive in practice.

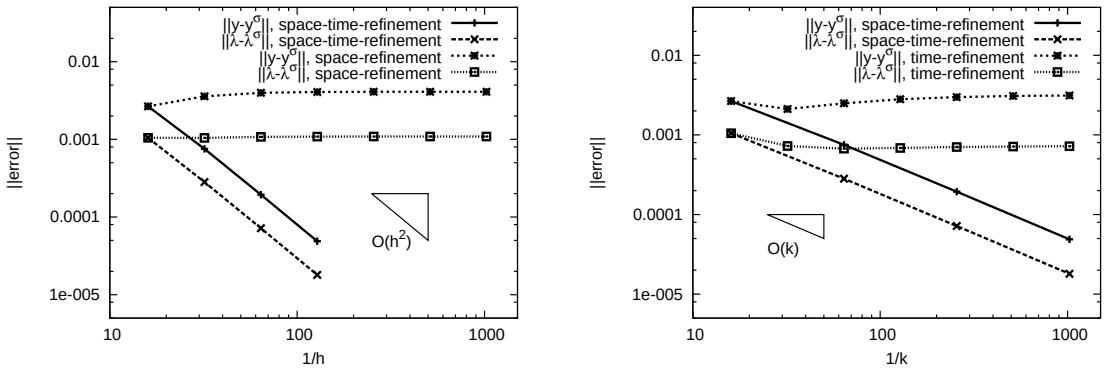

Figure 6.1: Heat equation, discretisation with Q_1/implicit Euler, error reduction. Logarithmic scale on the x- and y-axis. Left: Space and space-time refinement. Right: Time and space-time refinement.

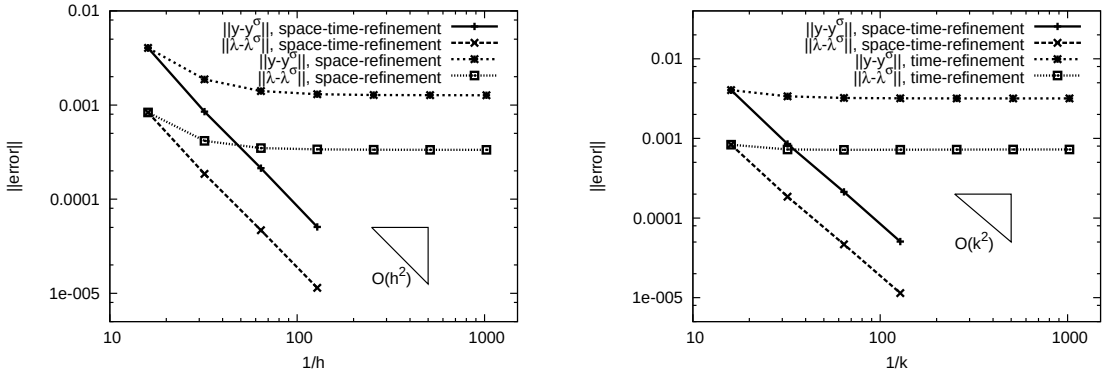

Figure 6.2: Heat equation, discretisation with Q_1/Crank–Nicolson, error reduction. Logarithmic scale on the x- and y-axis. Left: Space and space-time refinement. Right: Time and space-time refinement.

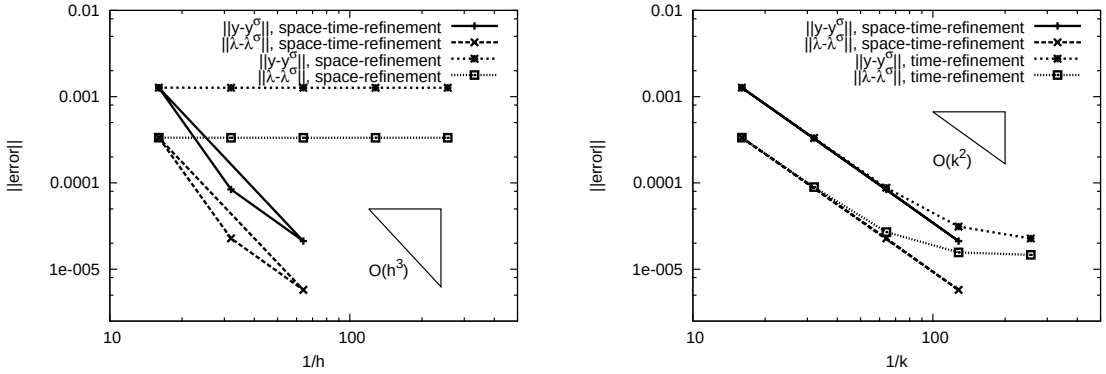

Figure 6.3: Heat equation, discretisation with Q_2/Crank–Nicolson, error reduction. Logarithmic scale on the x- and y-axis. Left: Space and space-time refinement. Right: Time and space-time refinement.

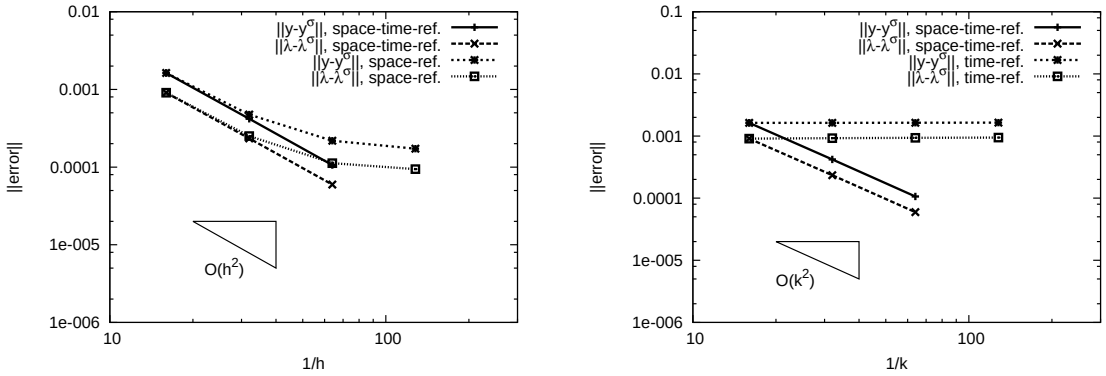

Figure 6.4: Stokes equations, discretisation with $\tilde{Q}_1/Q_0$/Crank–Nicolson, error reduction in the velocity. Logarithmic scale on the x- and y-axis. Left: Space and space-time refinement. Right: Time and space-time refinement.

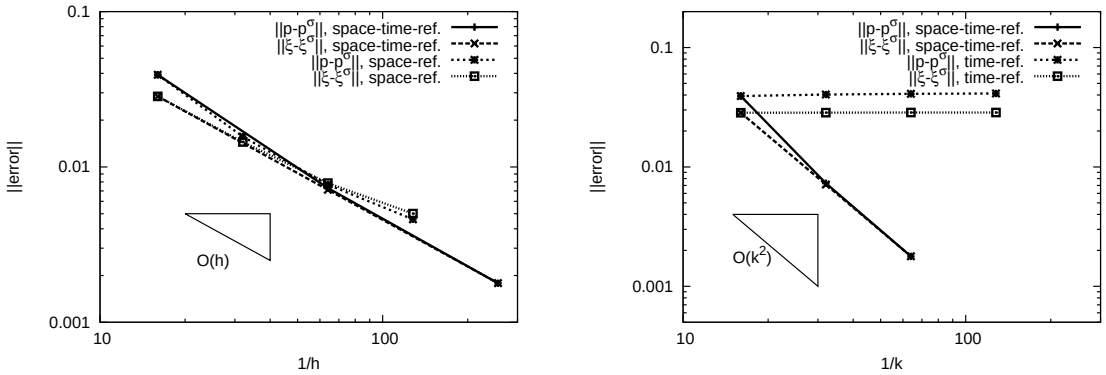

Figure 6.5: Stokes equations, discretisation with $\tilde{Q}_1/Q_0$/Crank–Nicolson, error reduction in the pressure. Logarithmic scale on the x- and y-axis. Left: Space and space-time refinement. Right: Time and space-time refinement.

6.2.3. Concluding remarks

Due to the strong coupling, it is advisable to equilibrate the accuracy of the space and time discretisation to avoid high computational cost in practical situations: Linear finite elements in space should be combined with a second order time stepping scheme like Crank–Nicolson, quadratic finite elements with a third order time stepping scheme. Other combinations can quickly lead to high memory requirements due to a very high number of timesteps:

Consider the heat equation test in Section 6.2.1, discretised with Q_2/implicit Euler. If the lowest level uses a resolution of $(h,k)_{\text{coarse}} = (1/16, 1/16)$ and if the mesh is refined three times based on the relationship $k \sim h^3$, there is $(h,k)_{\text{fine}} = (1/128, 1/8192)$ — so 8 192 timesteps had to be used, very quickly growing with every refinement level. As h is typically small, higher order time discretisations are therefore highly favourable. An example illustrating the construction of such schemes is given in [96]. Hussain et al. interpret the Crank–Nicolson time stepping as a special case of the so called cGP(1) method. Timestepping schemes of higher order are constructed by using a finite element approach in time with degrees of freedoms in the Gaussian points in time in combination with a nonlinear iteration for multiple solutions in a time interval.

6.3. The traditional θ-scheme time discretisation

In Section 2.6 on page 36ff, a special discretisation for the Crank–Nicolson scheme has been proposed. This scheme has been applied with the aim that the *First-Optimise-Then-Discretise* approach commutes with the *First-Discretise-Then-Optimise* approach. It has been derived by discretising the KKT system in such a way that the discrete KKT system of the *First-Optimise-Then-Discretise* approach matched the discrete KKT system of the *First-Discretise-Then-Optimise* approach.

However, the primal and dual equations of the KKT system can also be discretised 'traditionally', i. e., using the standard θ-scheme. This type of discretisation is easier to use in practice, e. g., there is no special prolongation/restriction in time necessary and the implementation of a possible end time observation is similar to the implicit Euler case. However, matrix-vector multiplications with the space-time matrix are slightly more expensive due to additional operators on the off-diagonal matrix bands. In the following it is shown that — at least for analytical test problems — the results are acceptable as well and show the expected convergence properties.

6.3.1. Heat equation

For simplicity, the heat equation is considered at first, see (2.1) on page 22. On the one hand, the KKT system is discretised with the scheme proposed in Section 2.6, on the other hand following [34] with the traditional θ-scheme. The first approach from Section 2.6, which is called 'derived' approach here, reads

$$\begin{aligned}
\frac{y_n - y_{n-1}}{k} \qquad - \theta\Delta y_n \qquad - (1-\theta)\Delta y_{n-1} &= -\frac{1}{\alpha}\lambda_{n-1+\theta},\\
\frac{\lambda_{n-1+\theta} - \lambda_{n+\theta}}{k} - \theta\Delta\lambda_{n-1+\theta} - (1-\theta)\Delta\lambda_{n+\theta} &= y_n - z_n,\\
y_0 = y^0, \quad \frac{\lambda_N}{k} - \theta\Delta\lambda_N &= \theta(y_N - z_N),
\end{aligned}$$

and the second, called 'traditional' approach,

$$\begin{aligned}
\frac{y_{n+1} - y_n}{k} - \theta\Delta y_{n+1} - (1-\theta)\Delta y_n \quad &= -\frac{\theta}{\alpha}\lambda_{n+1} \qquad - \frac{1-\theta}{\alpha}\lambda_n,\\
\frac{\lambda_n - \lambda_{n+1}}{k} - \theta\Delta\lambda_n \quad - (1-\theta)\Delta\lambda_{n+1} &= -\frac{\theta}{\alpha}(y_n - z_n) - \frac{1-\theta}{\alpha}(y_{n+1} - z_{n+1}),\\
y_0 = y^0, \quad \lambda_N = 0,&
\end{aligned}$$

with $n = 0, ..., N-1$, $k = 1/N$, complemented by the boundary conditions. The initial condition in the traditional approach can alternatively be realised, e. g., with a projection which mimics the timestep operator of the θ-scheme,

$$(\frac{\mathcal{I}}{k} - \theta\Delta)y_0 = (\frac{\mathcal{I}}{k} - \theta\Delta)y^0.$$

This type of projection ensures, e. g., correct boundary conditions into y_0. For the Stokes and Navier–Stokes equations, the projection is even more important. It has the additional effect to impose the additional condition '$-\operatorname{div} y_0 = 0$' into the initial solution in case it is not solenoidal.

Now, the analytical test Example 5.1 is considered, with $\alpha = 0.001$, $\gamma = 0.0$. The KKT system is discretised with Q_1 in space and the Crank–Nicolson scheme in time, once with the derived and once with the traditional approach. Table 6.1 lists the errors $||y - y^\sigma||_{\mathcal{Q}}$

and $||\lambda - \lambda^\sigma||_Q$ for different levels on refinement. Although both approaches show similar results and reduce the error with the expected order for increasing refinement level, the error for the traditional Crank–Nicolson scheme is even slightly smaller in this example.

		CN derived		CN traditional	
h	k	$\|\|y - y^\sigma\|\|$	$\|\|\lambda - \lambda^\sigma\|\|$	$\|\|y - y^\sigma\|\|$	$\|\|\lambda - \lambda^\sigma\|\|$
1/8	1/8	4.79E-02	1.95E-03	3.84E-02	1.57E-03
1/16	1/16	1.30E-02	5.19E-04	9.99E-03	4.52E-04
1/32	1/32	3.38E-03	1.34E-04	2.57E-03	1.21E-04
1/64	1/64	8.58E-04	3.43E-05	6.48E-04	3.17E-05
1/128	1/128	2.15E-04	8.68E-06	1.62E-04	8.10E-06

Table 6.1: Heat equation. Error for the derived and traditional Crank–Nicolson discretisation.

6.3.2. Stokes equations

Applying the two Crank–Nicolson schemes to the Stokes equations, the results are similar to those from the heat equation. Example 6.1 is considered, and the KKT system is discretised with $\tilde{Q}_1/Q_0$ and the Crank–Nicolson scheme. The corresponding errors are shown in Figure 6.6. For the velocity graphs (Figure 6.6 left), the space-time meshes use the relationship $h = k$. An error reduction corresponding to $\mathcal{O}(h^2) + \mathcal{O}(k^2)$ is expected which can be seen in the graph. For the pressure graphs on the other hand, a different test is performed which does two refinements in space per refinement in time to compensate for the reduced order of the pressure in space, i. e., $h = k^2$. Figure 6.6 right shows that the error reduces with the expected order $\mathcal{O}(h) + \mathcal{O}(k^2)$.

All in all, it can therefore be stated that the traditional approach as well as the derived approach provide similar results, at least for simple test examples. The situation can be different in the case of the full Navier–Stokes equations in combination with a stabilisation for the convective terms, but this analysis is beyond the scope of this work.

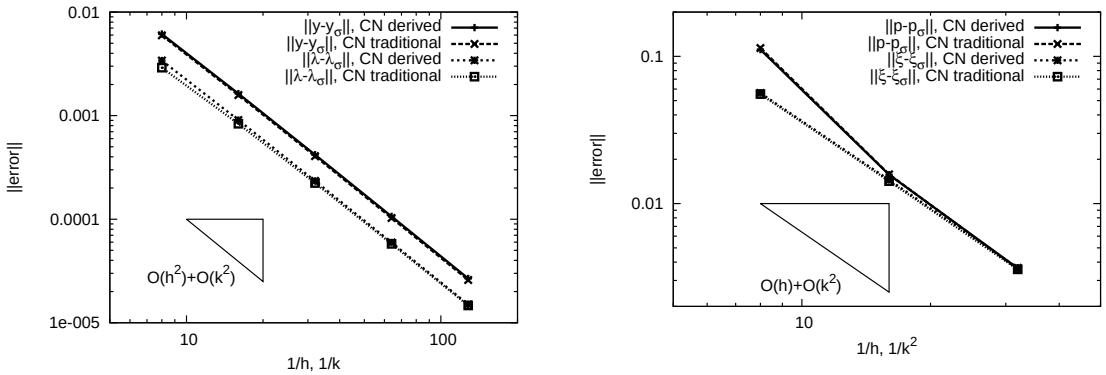

Figure 6.6: Stokes equations. Error for the derived and traditional Crank–Nicolson discretisation. Logarithmic scale on the x- and y-axis. Left: Primal and dual velocity. Right: Primal and dual pressure.

6.4. Summary and conclusions

Chapter 5 and Chapter 6 have provided the first part of the numerical analysis of the proposed discretisation and solver strategy.

At first, Chapter 5 has concentrated on a basic analysis of the linear solver. Numerical examples were created based on the heat equation and the Stokes equations. The focus of this chapter has been the choice of solver parameters and the robustness/efficiency of the different solver components.

Independently of the solver, Chapter 6 has analysed important properties of the space-time discretisation. The most important point is that the space discretisation has to be coupled to the time discretisation. This is a well known fact which can be seen in the error upon refinement of the space-time mesh and which has been verified for both the heat equation and the Stokes equations, for first and higher order discretisations. In addition, the chapter has analysed the accuracy of the modified Crank–Nicolson time discretisation preferred in this work. The error for this discretisation strategy has been shown to be similar to the error of the 'traditional' Crank–Nicolson time discretisation which interprets all solutions as being located at the endpoints of the time interval. In particular, the modified Crank–Nicolson scheme provides a second order accurate discretisation in time.

The next two chapters, Chapter 7 and Chapter 8, will apply a deeper numerical analysis. The focus will change to the optimal control of the Stokes and the Navier–Stokes equations, i. e., to problems with a saddle-point structure. All underlying model problems will be defined based on model and benchmark problems from CFD instead of analytical formulas. The overall aim will be a) to demonstrate the applicability of the solver methodology also for non-analytical test problems and b) to give an advice concerning the choice of discretisation and solver parameters to find a compromise between efficiency and robustness.

7

Numerical analysis of the solver: Stokes and Navier–Stokes equations

Chapter 5 and 6 have started the numerical analysis based on analytical test examples. The main focus of these chapters has been the heat equation as well as the Stokes equations. First tests have given a general overview about the efficiency and robustness of the different solver components introduced in Chapter 3, and a numerical error analysis has illustrated the influence of the coupling between the space and the time mesh to the accuracy of the solution.

Chapter 5 has briefly demonstrated at the end that the proposed discretisation and solver strategy is well applicable to the optimal control of the nonstationary Stokes equations, for first order as well as for higher order space and time discretisations. Based on the experiences acquired there, this chapter continues with an extended analysis of the proposed solver strategy, this time with the focus on non-analytical test problems from fluid dynamics. The first half of the chapter deals with the optimal distributed control of the Stokes equations and illustrates, e. g., the influence of regularisation parameters or the choice of the mesh and the mesh hierarchy to the convergence properties of the solver. The second half concentrates on the application of nonlinear solver techniques to distributed control problems based on the Navier–Stokes equations, with and without constraints in the control.

The solver is numerically analysed with respect to the following key points:

- Robustness: How does the solver react to different settings of the regularisation parameters? How does anisotropy in the space-time mesh influence the solver? How does the applied discretisation in space and time influence the solver stability?

- Efficiency: How efficient is the optimisation solver in comparison to a solver for a simulation? Which parameters influence the solver efficiency and how do they have to be chosen (e. g., stopping criteria of linear and nonlinear solvers)?

In particular, this chapter compares the numerical effort (CPU time) of optimisation and simulation to illustrate the linear complexity of the method and to give an overview about the additional costs of the optimisation compared to a simulation. However, similar to Chapter 5, the chapter does not analyse any relation between the accuracy of a chosen discretisation and the numerical effort that has to be invested to solve the underlying KKT system. This final link will be part of the last chapter concerning numerical tests, Chapter 8.

Outline

Section 7.1 extends the analysis of the space-time multigrid solver from Chapter 5. A nonstationary distributed control problem for the Stokes equations is defined which is based on the so called 'Driven–Cavity' configuration. This is used to compare the numerical effort in terms of CPU time for optimisation and simulation, to determine the influence of the regularisation parameters in the KKT system in more detail and to analyse the effects of anisotropy in the space-time mesh on the efficiency of the solver.

In the second part, Section 7.2, the focus is turned to nonlinear problems. Similar to Section 7.1, an underlying nonstationary distributed control problem of 'Driven–Cavity' type is defined, this time for the Navier–Stokes equations. The section concentrates at first on the different nonlinear solver techniques introduced in Chapter 3 to find a configuration that solves a given nonlinear problem with the smallest possible CPU time. It turns out that although the smallest CPU time can be achieved by a manual choice of the stopping criteria, the adaptive Newton scheme gives the best compromise between efficiency of the solver and simplicity for the end user.

The section continues with the analysis of the influence of regularisation parameters to the nonlinear solver and closes with a comparison of the CPU time between simulation and optimisation. It is shown that for the example considered there, the optimisation is less than ten times more expensive than a corresponding simulation.

Finally in Section 7.3, the effects of constraints on the control are analysed. The section applies the different discretisation techniques from Section 4.2 on page 87ff to realise the constraints and illustrates the efficiency of the semismooth Newton method for different space-time discretisations.

7.1. Analysis of the multigrid solver

As a starting point the following linear test problem is introduced, based on the optimal control of the nonstationary Stokes equations. An initial stationary Stokes flow should be controlled over time to a stationary Navier–Stokes flow.

7.1 Stokes equations, 'Driven–Cavity' example Consider the optimal control of the Stokes equations including an end time observation, see Equation (4.2) page 82. The underlying domain is $\Omega = (0,1)^2$. On the four boundary edges

$$\begin{aligned}
\Gamma_1 &:= \{0\} \times (0,1) &&= \{(0,x_2) \in \mathbb{R}^2 \,|\, x_2 \in (0,1)\},\\
\Gamma_2 &:= [0,1] \times \{0\} &&= \{(x_1,0) \in \mathbb{R}^2 \,|\, x_1 \in [0,1]\},\\
\Gamma_3 &:= \{1\} \times (0,1) &&= \{(1,x_2) \in \mathbb{R}^2 \,|\, x_2 \in (0,1)\},\\
\Gamma_4 &:= [0,1] \times \{1\} &&= \{(x_1,1) \in \mathbb{R}^2 \,|\, x_1 \in [0,1]\},
\end{aligned}$$

the Dirichlet boundary conditions $y(x,t) = (0,0)$ for $x \in \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$ and $y(x,t) = (1,0)$ for $x \in \Gamma_4$ are used. The coarse grid consists of only one square element. The time interval for this test case is $[0,T]$ with $T = 1$, the viscosity parameter is set to $\nu = 1/400$. The initial flow y^0 is the fully developed, stationary Stokes flow, while the target flow z is chosen as the fully developed, stationary Navier–Stokes flow at $\nu = 1/400$.

Table 7.1 depicts an overview about the problem size on different mesh levels. Figure 7.1 shows a picture of the streamlines of the target and the initial flow with the corresponding velocity magnitude in the background. For better visualisation, the positive and negative

		simulation		optimisation	
h	k	#dof(space)	#dof(total)	#dof(space)	#dof(total)
1/16	1/16	1 344	22 848	2 688	45 696
1/32	1/32	5 248	167 936	10 496	335 872
1/64	1/64	20 736	1 347 840	41 472	2 695 680
1/128	1/128	82 432	10 551 296	164 864	21 102 592

Table 7.1: 'Driven–Cavity' example, problem size. Number of degrees of freedom in space ('#dof(space)') and on the whole space-time domain including the initial condition ('#dof(total)'). Discretisation with $\tilde{Q}_1/Q_0$ in space.

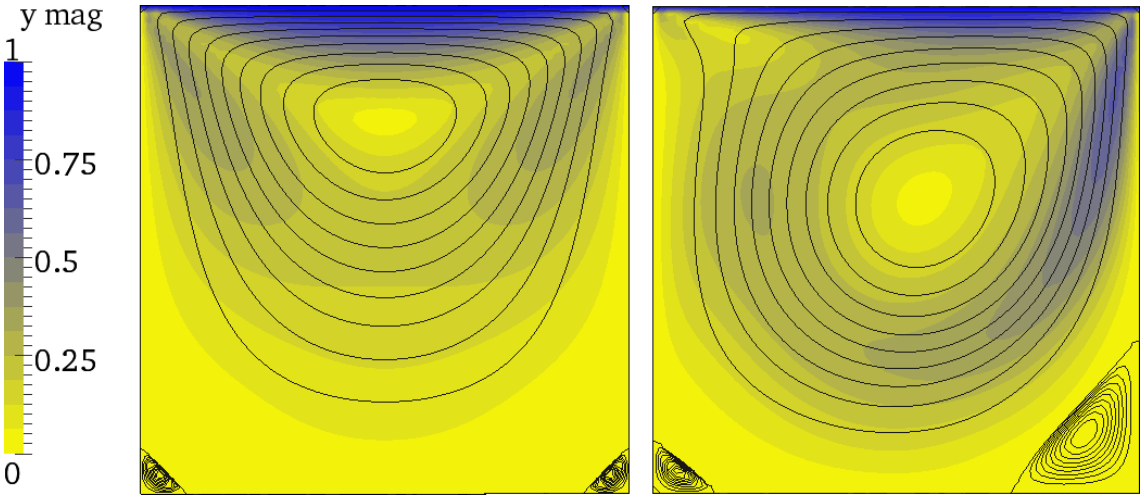

Figure 7.1: 'Driven–Cavity' example, Streamlines of the stationary Stokes (initial) flow (left) and stationary Navier–Stokes (target-) flow (right). Velocity magnitude in the background.

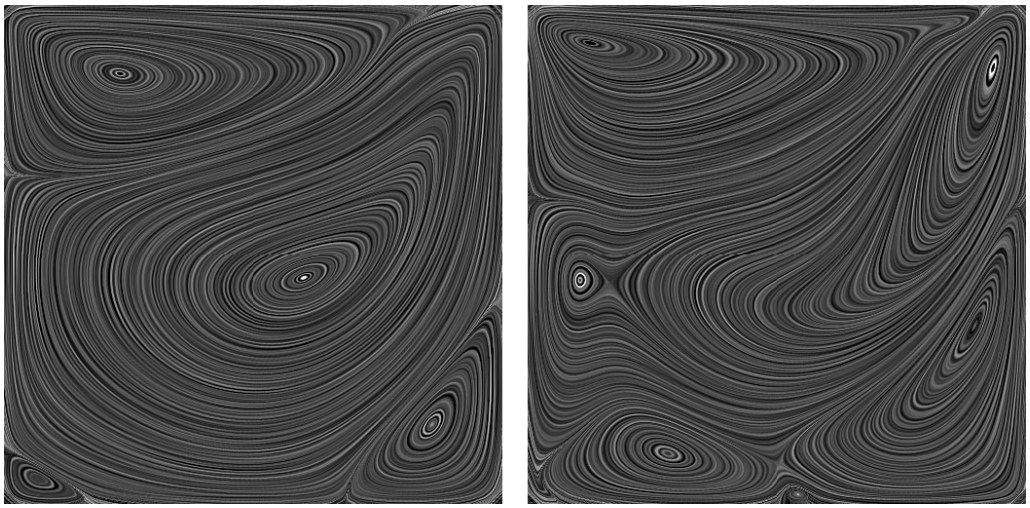

Figure 7.2: 'Driven–Cavity' example, Surface-LIC representation of u at $t = 0.0625$ (left) and $t = 0.5$ (right).

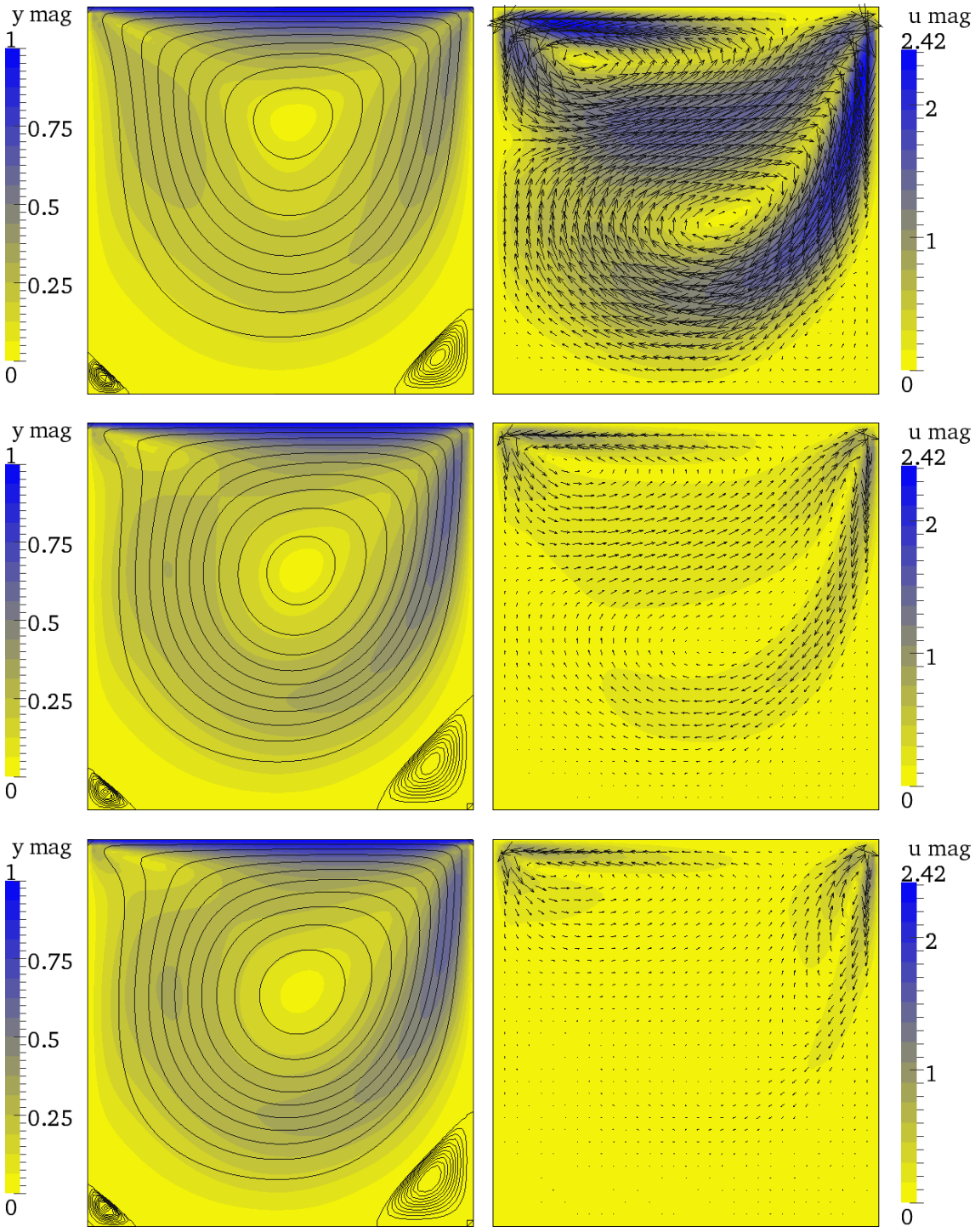

Figure 7.3: 'Driven–Cavity' example, controlled flow at $t = 0.0625$ (top), $t = 0.25$ (centre) and $t = 0.5$ (bottom). Left: Streamlines with primal velocity magnitude in the background. Right: Control magnitude.

streamlines are shown with a different resolution, so the big and small vortices can be seen more clearly. Figure 7.2 depicts the control u at $t = 0.0625$ and $t = 0.5$ using the Surface-LIC ('Line-Integral-Convolution') representation and Figure 7.3 the controlled flow and the control at $t = 0.0625$, $t = 0.25$ and $t = 0.5$ for $\alpha = 0.01$ and $\gamma = 0.0$. Similar to the streamline representation, the Surface-LIC representation illustrates the path of particles in the flow. Two main vortices in the control u can be observed which 'push' the Stokes flow to the Navier–Stokes flow at the beginning of the time interval. For $t = 0.5$, the vortices are still visible but much weaker; the flow at $t = 0.5$ is visibly close to the desired flow.

7.1.1. Basic multigrid performance

The first analysis concentrates on the behaviour of the solver if being applied to the non-analytical test problem from above. To be more specific, the purpose of the following paragraphs is two-fold. On the one hand, the convergence behaviour is compared to the results of Chapter 5. The solver behaviour is not expected to change much if the solver is applied to a non-analytical test problem. On the other hand, the convergence analysis involves CPU time measurements and comparisons to simulations. The purpose is here to show that the desired relationship

$$\frac{\text{effort for optimisation}}{\text{effort for simulation}} \leq C$$

for a moderate constant $C > 0$ holds also in terms of CPU time.

Measurement of the numerical effort In the following numerical tests, the numerical effort for the optimisation is compared to the numerical effort of a 'corresponding' simulation. The term 'numerical effort' is always measured in terms of CPU time. All such tests are executed on the same computer architecture in the following way:

1.) The optimal control problem is solved for a specific target flow z as defined in the above test example. The stopping criteria are always set to $\varepsilon_{\text{OptMG}} = 10^{-10}$ for the space-time multigrid solver, $\varepsilon_{\text{CoarseMG}} = 10^{-10}$ for the space-time coarse grid solver and $\varepsilon_{\text{SpaceMG}} = 10^{-2}$ for the multigrid solver in space, based on the result of Section 5.1.4 on page 107ff. The space-time coarse grid solver is BiCGStab(FBGsPrec). This step generates a control u.

2.) A pure simulation of the nonstationary Stokes–equations is calculated. The simulation uses the same discretisation in space and time as the previous optimal control problem and takes the computed control u from step 1 as right-hand side. In each timestep the norm of the residual is reduced by ten digits, $\varepsilon_{\text{SimMG}} = 10^{-10}$.

Basic comparison of simulation and optimisation For the first couple of tests, the regularisation parameters are fixed to $\alpha = 0.01$ and $\gamma = 0.0$, and the underlying KKT system is discretised with $\tilde{Q}_1/Q_0$ in space and the implicit Euler scheme in time. In Table 7.2, results from optimisation and simulation are compared. The optimisation uses a full space-time hierarchy which is set up by full space-time coarsening down to $(h, k)_{\text{coarse}} = (1/16, 1/16)$. The space-time multigrid solver uses a V-cycle and applies a BiCGStab(FBGsPrec,NSM=4) smoother; from the numerical analysis in Section 5.1 on page 102ff, this smoother is known to be the most stable one in this work, thus showing representative results for all introduced space-time smoothers. On the one hand, T_{opt} describes the time needed for the optimisation, ITE_{opt} and ρ_{opt} the number of iterations and

convergence rate of the space-time multigrid. On the other hand, T_{sim} denotes the time for the simulation. The optimisation solver converges rapidly and level-independent, although the convergence is slightly slower than it was for the analytic test problem in Chapter 5. Compared to the simulation, it can be seen that the solver needs $\approx$ 10–12 times more CPU time. Furthermore, the CPU time for, both simulation and optimisation, grows by a factor of ≈ 8 in correspondence to the problem size which confirms the linear complexity of both algorithms.

A similar behaviour can also be observed in the Tables 7.3 and 7.4. Here, the time discretisation is carried out with the Crank–Nicolson scheme, the space discretisation once using the $\tilde{Q}_1/Q_0$ and once using the Q_2/P_1^{disc} finite element pair. The $\tilde{Q}_1/Q_0$/Crank–Nicolson shows a factor $\approx$ 10–12 as well while the Q_2/P_1^{disc}/Crank–Nicolson discretisation has an even a better ratio of $\approx$ 5..7 in this example.

All in all, the effort for solving the KKT system grows linearly with the problem size. For the test problem above, the factor between the optimisation and the simulation is $C \approx$ 5..12. Of course, this factor depends on the problem and on the setting of the regularisation parameters; an example for this will be given in the next section.

h	k	ITE_{opt}	ρ_{opt}	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
1/32	1/32	5	5.30E-03	620	57	10.7
1/64	1/64	6	1.85E-02	4 266	415	10.2
1/128	1/128	7	2.53E-02	35 128	3 104	11.3

Table 7.2: Simulation and optimisation for the control of the Stokes equations. 'Driven–Cavity' example. Discretisation with $\tilde{Q}_1/Q_0$ in space and the implicit Euler scheme in time.

h	k	ITE_{opt}	ρ_{opt}	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
1/32	1/32	4	2.25E-03	608	53	11.4
1/64	1/64	7	2.41E-02	5 253	418	12.5
1/128	1/128	7	3.61E-02	38 898	3 371	11.5

Table 7.3: Simulation and optimisation for the control of the Stokes equations. 'Driven–Cavity' example. Discretisation with $\tilde{Q}_1/Q_0$ in space and the Crank–Nicolson scheme in time.

h	k	ITE_{opt}	ρ_{opt}	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
1/32	1/32	4	8.8E-04	2 119	429	4.9
1/64	1/64	4	7.2E-04	12 615	2 171	5.8
1/128	1/128	4	9.0E-04	93 671	13 786	6.8

Table 7.4: Simulation and optimisation for the control of the Stokes equations. 'Driven–Cavity' example. Discretisation with Q_2/P_1^{disc} in space and the Crank–Nicolson scheme in time.

7.1.2. Influence of the regularisation parameters

For the optimal distributed control of the Stokes equations including an end time observation, the corresponding functional $J(\cdot)$ to minimise depends on two regularisation parameters, namely α and γ (cf. Equation (4.2) page 82). While α penalises too high costs in the control, the parameter γ models the influence of the end time observation. An 'anisotropic choice' of these parameters, i. e., lower values for α and higher values of γ, is expected to have a negative impact on the efficiency of the solver. This was already briefly discussed for the heat equation in Section 5.1.4 on page 107ff and will be shown here for the Stokes equations in a more detailed way.

The 'Driven–Cavity' example for different regularisation parameters Consider the 'Driven cavity' example for the Stokes equations, Example 7.1, based on a space-time mesh with $(h, k) = (1/64, 1/64)$ and a hierarchy with two levels, the coarse mesh defined by $(h, k)_{\text{coarse}} = (1/32, 1/32)$. The solver in the following test is a two-grid solver using a BiCGStab(FBGsPrec,NSM=4) smoother for (postsmoothing (as above) and a BiCGStab(FBGsPrec) coarse grid solver (which is the strongest proposed one-level solver). The local systems in space are solved with a spatial multigrid solver that uses a PSCSmootherDiag/PSCSmootherFull smoother combination: Wherever possible, the proposed PSCSmootherDiag is used which provides fast monolithic smoothing for the spatial subproblems. If the spatial multigrid solver does not converge, the computation is repeated using the stronger (but slower) PSCSmootherFull smoother. This strategy helps to improve convergence in the last one or two time intervals, where the end time observation introduces numerical difficulties for large values of γ. The other solver parameters are adjusted as described in Section 7.1.1.

Based on this solver configuration, the convergence properties are measured for different α and γ, see Tables 7.5 to 7.7. The tables differ in the underlying discretisation: For Table 7.5, the problem is discretised with $\tilde{Q}_1/Q_0$ and the implicit Euler scheme, for Table 7.6 with $\tilde{Q}_1/Q_0$ and the Crank–Nicolson scheme and for Table 7.7 with Q_2/P_1^{disc} and the Crank–Nicolson scheme. The general behaviour can be summarised as follows: For smaller parameter α and larger parameter γ, the solver faces more numerical difficulties, thus the number of iterations increases. For too strong parameter settings, the solver does not even converge anymore (indicated by 'div'). A discretisation with the implicit Euler scheme in time is more stable than a discretisation with the Crank–Nicolson scheme in time. Switching from $\tilde{Q}_1/Q_0$ to Q_2/P_1^{disc} in space hardly influences the stability.

Comparing the results with those obtained for the heat equation in Section 5.1 on page 102ff it can be said that the regularisation parameters have a stronger influence to the numerical stability than for the heat equation: For the heat equation, the solver converged perfectly even for $\alpha = 0.001$, $\gamma = 1000$. In case of the Stokes equation on the other hand, stronger values than $\alpha = 0.01$, $\gamma = 10$ necessitate a stronger solver in space or may even lead to a complete breakdown of the solver.

7.1.3. Anisotropic space-time meshes and coarsening strategies

It is well known (cf. [75, Chapter 10], [6], [141, Chapter 5.1] that anisotropic meshes have an influence on the convergence behaviour of a solver. Consider for example the standard

α:	1.0		0.1		0.01	
γ	#ite	ρ	#ite	ρ	#ite	ρ
0.0	5	5.01E-03	5	5.39E-03	5	9.17E-03
1.0	4	1.96E-03	6	1.56E-02	6	2.04E-02
10.0	8	5.39E-02	9	7.04E-02	9	5.94E-02
100.0	6	1.83E-02	7	2.83E-02	div	div
1000.0	7	2.52E-02	div	div	div	div

Table 7.5: 'Driven–Cavity' example. Convergence of the space-time multigrid solver depending on α and γ. Discretisation with $\tilde{Q}_1/Q_0$ in space and the implicit Euler scheme in time. 'div' indicates a solver breakdown.

α:	1.0		0.1		0.01	
γ	#ite	ρ	#ite	ρ	#ite	ρ
0.0	5	5.57E-03	5	8.02E-03	8	5.41E-02
1.0	4	1.27E-03	8	4.52E-02	8	4.64E-02
10.0	5	9.31E-03	11	1.14E-01	9	6.93E-02
100.0	12	1.40E-01	14	1.81E-01	div	div
1000.0	21	3.20E-01	div	div	div	div

Table 7.6: 'Driven–Cavity' example. Convergence of the space-time multigrid solver depending on α and γ. Discretisation with $\tilde{Q}_1/Q_0$ in space and Crank–Nicolson in time. 'div' indicates a solver breakdown.

α:	1.0		0.1		0.01	
γ	#ite	ρ	#ite	ρ	#ite	ρ
0.0	2	4.45E-06	3	1.36E-04	4	1.35E-03
1.0	3	4.51E-05	6	1.21E-02	8	5.06E-02
10.0	5	8.80E-03	12	1.34E-01	9	6.84E-02
100.0	14	1.78E-01	16	2.21E-01	div	div
1000.0	20	3.09E-01	div	div	div	div

Table 7.7: 'Driven–Cavity' example. Convergence of the space-time multigrid solver depending on α and γ. Discretisation with Q_2/P_1^{disc} in space and Crank–Nicolson in time. 'div' indicates a solver breakdown.

anisotropic Poisson equation

$$\begin{aligned} -\text{div } \mathcal{D}\nabla y &= f \quad \text{on } \Omega, \\ y &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{7.1}$$

on a domain $\Omega \subset \mathbb{R}^2$ for a regular 2×2 diffusion tensor $\mathcal{D}$ and $f : \Omega \to \mathbb{R}^2$, $g : \partial\Omega \to \mathbb{R}$. If this problem is discretised with a finite element or finite difference approach, most standard solvers work well as long as $\mathcal{D} = \mathcal{I}$ is the identity tensor, while they deteriorate if $\mathcal{D}$ describes an anisotropic operator, e. g. $\mathcal{D} = \begin{pmatrix} 1/\varepsilon^2 & 0 \\ 0 & 1 \end{pmatrix}$, $\varepsilon > 0$ small.

The reason is that the anisotropy in the continuous operator on an isotropic mesh introduces an anisotropy in the discrete operator due to the mapping between the reference and the real element. However, a similar anisotropy is introduced by an isotropic operator on an anisotropic mesh: The above problem is essentially equivalent to the problem

$$\begin{aligned} -\text{div } \nabla y &= \tilde{f} \quad \text{on } \tilde{\Omega} \\ y &= 0 \quad \text{on } \partial\tilde{\Omega} \end{aligned}$$

on $\tilde{\Omega} = (0, \varepsilon) \times (0, 1)$ with $\tilde{f}(x_1, x_2) = f(\varepsilon x_1, x_2)$, cf. [6]. To deal with such an anisotropy, an appropriate smoother has to be used. Following [6], a smoother is well designed for an anisotropic problem if it focuses on the 'tight connections' of the degrees of freedoms (which can be interpreted as the 'direction of main stiffness'), i. e., if it is a line smoother for those degrees of freedom which are 'close' to each other.

This insight can be carried over to the field of optimal control problems. The solver methodology introduced in Chapter 3 basically processes all degrees of freedom in a monolithic way, but it treats the degrees of freedom in time in a different way than those in space. For smoothers like FBGsSmoother or FBJacSmoother, a local spatial solver is applied, so all degrees of freedom in space are 'tightly' coupled by the smoother. On the other hand, the degrees of freedom in time are 'more loosely' coupled, they are processed by a weaker Block-Jacobi or Block-Gauß–Seidel approach. The space-time smoothers from Chapter 3 can therefore be seen as *line smoothers in space*. As a consequence the convergence of the solver should improve if the timestep size is increased and deteriorate for smaller timesteps. This statement is verified in the next numerical test.

'Driven–Cavity' with anisotropic space-time meshes The basic underlying optimal control problem is the 'Driven cavity' example for the Stokes equations, Example 7.1. The regularisation parameters are chosen as $\alpha = 0.01$, $\gamma = 0.0$. A fully isotropic mesh with $(h, k)_{\text{iso}} = (1/32, 1/32)$ is chosen as initial fine mesh. The mesh is gradually refined, once only in space up to $(h, k) = (1/256, 1/32)$, once only in time up to $(h, k) = (1/32, 1/256)$. This gives a couple of fine meshes with different levels of anisotropy.

From each fine mesh, a mesh hierarchy for the space-time multigrid solver is generated by coarsening until either $h = 1/8$ or $k = 1/8$ is reached. This is done with different coarsening strategies, i. e., 1:1 coarsening, pure time coarsening and anisotropic coarsening in time (which coarsens two levels in time per one space coarsening), see Section 3.1 on page 48ff. Table 7.8 gives an example of the two mesh hierarchies based on the isotropic fine mesh with $(h, k)_{\text{iso}} = (1/32, 1/32)$.

Finally, the discretisation of the KKT systems is carried out with $\tilde{Q}_1/Q_0$ in space and the Crank–Nicolson and implicit Euler timestepping scheme in time. The resulting linear space-time systems are solved with a space-time multigrid using a BiCGStab(FBGsPrec,NSM=2) smoother.

coarsening:	space-time		only time		2× time/space	
lv.	h	k	k	h	k	h
1	1/8	1/8	1/8	1/32	1/8	1/16
2	1/16	1/16	1/16	1/32	1/16	1/32
3	1/32	1/32	1/32	1/32	1/32	1/32

Table 7.8: Space-time hierarchy with $(h,k)_{\text{iso}} = (1/32, 1/32)$. Hierarchy created by space-time coarsening and pure time coarsening.

Discretisation with the Crank–Nicolson scheme The first result column in Table 7.9, entitled 'space-time', lists the convergence behaviour of the solver for different degrees of anisotropy in the space-time mesh if the space-time hierarchy is created by full space-time coarsening. For the time discretisation, the Crank–Nicolson scheme is used. The convergence of the solver deteriorates for $h \gg k$. For $h \ll k$ on the other hand, the solver behaves well. The proposed solver methodology from Chapter 3 is therefore well suited for higher order timestep techniques that allow large timesteps for solutions that are smooth in time.

The situation is slightly different if the space-time hierarchy is set up in a different way. The next column in Table 7.9, entitled 'only time', shows the convergence behaviour of the solver if the space-time hierarchy is created by pure time coarsening. The solver hardly deteriorates for small k. This is reasonable: The algorithm can be interpreted as pure 1D time-multigrid for an equidistant mesh in time. Such a situation is similar to a 1D Laplace problem discretised with a finite difference scheme, for which multigrid is known to converge with level-independent convergence rates (cf. [75]). For very small timesteps, semi-coarsening in time should be chosen; this is more stable than full space-time coarsening.

An alternative to pure semi-coarsening in time is the strategy to coarsen the time-mesh twice per space coarsening — or more precisely, to coarsen the spatial mesh every two time coarsenings. The last column in Table 7.9, entitled '2×time/space', depicts the convergence results of the solver in this case. A similar strategy was proposed in [75, §10.5] for general anisotropic meshes to reduce the anisotropy from level to level. The convergence is rather similar to the pure time-coarsening case although the number of unknowns in space is reduced. Since the total number of unknowns for this strategy is between the total number of unknowns for the time coarsening and the full space-time coarsening, this strategy can be used as a compromise between robustness and problem size.

fine grid		space-time		only time		2×time/space	
h	k	#ite	ρ	#ite	ρ	#ite	ρ
1/256	1/32	6	1.78E-02	7	2.40E-02	7	2.38E-02
1/128	1/32	8	5.61E-02	7	2.67E-02	6	2.13E-02
1/64	1/32	7	3.62E-02	6	1.73E-02	6	1.85E-02
1/32	1/32	7	3.18E-02	6	2.09E-02	6	1.86E-02
1/32	1/64	11	1.09E-01	6	2.00E-02	6	1.70E-02
1/32	1/128	15	1.89E-01	7	2.38E-02	8	4.43E-02
1/32	1/256	20	3.06E-01	7	3.59E-02	10	9.40E-02

Table 7.9: Solver convergence for different degrees of anisotropy. Discretisation with $\tilde{Q}_1/Q_0$/Crank–Nicolson.

fine grid		space-time		only time		2×time/space	
h	k	#ite	ρ	#ite	ρ	#ite	ρ
1/256	1/32	6	1.79E-02	6	1.65E-02	6	1.64E-02
1/128	1/32	6	1.63E-02	6	2.14E-02	6	2.01E-02
1/64	1/32	6	1.63E-02	6	2.14E-02	6	2.01E-02
1/32	1/32	7	2.94E-02	7	2.29E-02	7	1.94E-02
1/32	1/64	11	9.67E-02	8	3.70E-02	8	3.64E-02
1/32	1/128	14	1.76E-01	10	7.97E-02	9	7.14E-02
1/32	1/256	19	2.84E-01	8	4.32E-02	9	7.49E-02

Table 7.10: Solver convergence for different degrees of anisotropy. Discretisation with $\tilde{Q}_1/Q_0$/implicit Euler.

Discretisation with the implicit Euler scheme Table 7.10 finally lists the results in the case that the implicit Euler scheme is used for the time discretisation. The results are quite similar, but in this case, the reduced order of the time discretisation has an influence to the results computed with pure time coarsening: The number of iterations is slightly larger for smaller time intervals, which has not been the case for the Crank–Nicolson scheme. Coarsening twice in time per space-level gives satisfactory results as well, similar to the previous test.

7.2. Basic analysis of the nonlinear solver

The following example extends the 'Driven cavity' example for the Stokes equations, Example 7.1 on page 128, to the Navier–Stokes equations. Basically, the setting is turned around: Instead of controlling a Stokes flow to a Navier–Stokes flow, a Navier–Stokes flow is controlled to a Stokes flow.

7.2 Navier–Stokes equations, 'Driven–Cavity' example Consider the optimal control of the Navier–Stokes equations including an end time observation, see Section 4.1 at page 81ff. The underlying domain is $\Omega = (0,1)^2$. On the four boundary edges

$$\begin{aligned}
\Gamma_1 &:= \{0\} \times (0,1) &&= \{(0,x_2) \in \mathbb{R}^2 \,|\, x_2 \in (0,1)\},\\
\Gamma_2 &:= [0,1] \times \{0\} &&= \{(x_1,0) \in \mathbb{R}^2 \,|\, x_1 \in [0,1]\},\\
\Gamma_3 &:= \{1\} \times (0,1) &&= \{(1,x_2) \in \mathbb{R}^2 \,|\, x_2 \in (0,1)\},\\
\Gamma_4 &:= [0,1] \times \{1\} &&= \{(x_1,1) \in \mathbb{R}^2 \,|\, x_1 \in [0,1]\},
\end{aligned}$$

the Dirichlet boundary conditions $y(x,t) = (0,0)$ for $x \in \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$ and $y(x,t) = (1,0)$ for $x \in \Gamma_4$ are defined. The coarse grid consists of only one square element. The time interval for this test case is $[0,T]$ with $T = 1$, the viscosity parameter is set to $\nu = 1/400$. The initial flow y^0 is the stationary fully developed Navier–Stokes flow at $\nu = 1/400$, while the target flow z is chosen as the fully developed Stokes flow.

A stationary counterpart of this example was analysed in [3]. In [152], Ulbrich analysed this problem under constraints, see also [83, 84]. Figure 7.4 depicts the streamlines of the target flow and the initial flow with the corresponding velocity magnitude in the background.

For better visualisation, the positive and negative streamlines are shown with a different resolution, so the big and small vortices can be seen more clearly. Figure 7.5 depicts the control u at $t = 0.0625$ and $t = 0.5$ using the Surface-LIC representation and Figure 7.6 the controlled flow with corresponding control at $t = 0.0625$, $t = 0.25$ and $t = 0.5$. The regularisation parameters are chosen as $\alpha = 0.01$ and $\gamma = 0.0$. Two main vortices in the control u can be identified at the beginning of the time interval which 'push' the Navier–Stokes flow to the Stokes flow. These nearly disappear when the Navier–Stokes flow reaches the 'optimal' state, or more precisely, they change to large 'conservation' vortices which keep the Navier–Stokes flow in shape.

7.2.1. Nonlinear solver comparison

In a first test, the convergence properties and the efficiency of the following three nonlinear solvers are analysed:

- Fixed point iteration with fixed ε_{OptMG},
- Newton iteration with fixed ε_{OptMG},
- inexact Newton (with $q = 2$ and $\varepsilon_{\mathrm{OptIN}} = 10^{-2}$ in (3.30b), see page 78).

The following well known behaviour from the algorithms is expected:

a) The standard fixed point iteration is expected to converge linearly and almost independently of ε_{OptMG}.

b) The Newton iteration converges quadratically if ε_{OptMG} is small enough. However, the residual is reduced at most by ε_{OptMG} in every iteration, thus the quadratic convergence reduces to (fast) linear convergence as soon as the residual is small enough.

c) The inexact Newton converges quadratically; as ε_{OptMG} is automatically chosen, the user does not have to define this parameter.

Nonlinear solver convergence test Consider Example 7.2 with $\alpha = 0.01$ and $\gamma = 0.0$. The corresponding KKT system is discretised with $\tilde{Q}_1/Q_0$/implicit Euler on a mesh with $(h, k) = (1/64, 1/64)$ and solved with the above three nonlinear algorithms using the stopping criterion $\varepsilon_{\mathrm{OptNL}} = 10^{-8}$ — i. e., the nonlinear solver reduces the norm of the residual by eight digits. For the inner space-time multigrid solver, the stopping criterion is set to $\varepsilon_{OptMG} = 10^{-1}$, 10^{-2} and 10^{-10}, respectively. Multigrid itself operates as 3-level solver with V-cycle on a space-time hierarchy created by full space-time coarsening down to $(h, k)_{\mathrm{coarse}} = (1/16, 1/16)$. For smoothing, a BiCGStab(FBGsPrec,NSM=4) smoother is chosen, similar to Section 7.1.

Figure 7.7 depicts the convergence of the different solvers. The standard fixed point iteration (left picture) shows a linear reduction of the residual. There is hardly any difference between the curves from $\varepsilon_{OptMG} = 10^{-1}$ to $\varepsilon_{OptMG} = 10^{-10}$, so the solver is quite independent of ε_{OptMG} as expected.

The right part of Figure 7.7 shows the convergence of the Newton and inexact Newton algorithms. For $\varepsilon_{OptMG} = 10^{-1}$, the solver converges linearly and for $\varepsilon_{OptMG} = 10^{-10}$ quadratically as expected. The inexact Newton solver converges much the same way as the standard Newton solver for $\varepsilon_{OptMG} = 10^{-2}$; both converge quadratically except for the last step: The Newton solver with $\varepsilon_{OptMG} = 10^{-2}$ stops the quadratic convergence

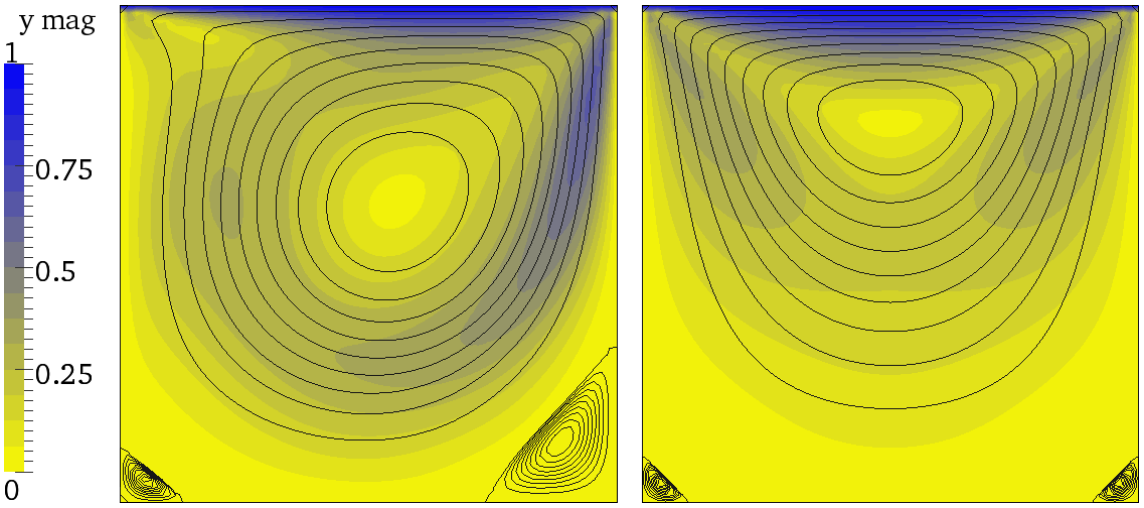

Figure 7.4: 'Driven–Cavity' example, Streamlines with velocity magnitude in the background. Left: Stationary Navier–Stokes (initial) flow. Right: Stationary Stokes (target) flow.

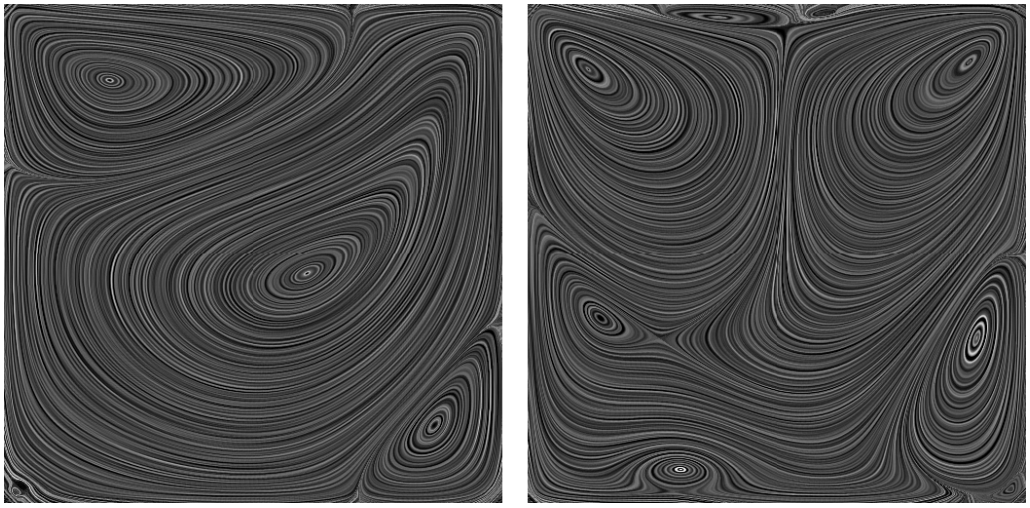

Figure 7.5: 'Driven–Cavity' example, Surface-LIC representation of u at $t = 0.0625$ (left) and $t = 0.5$ (right).

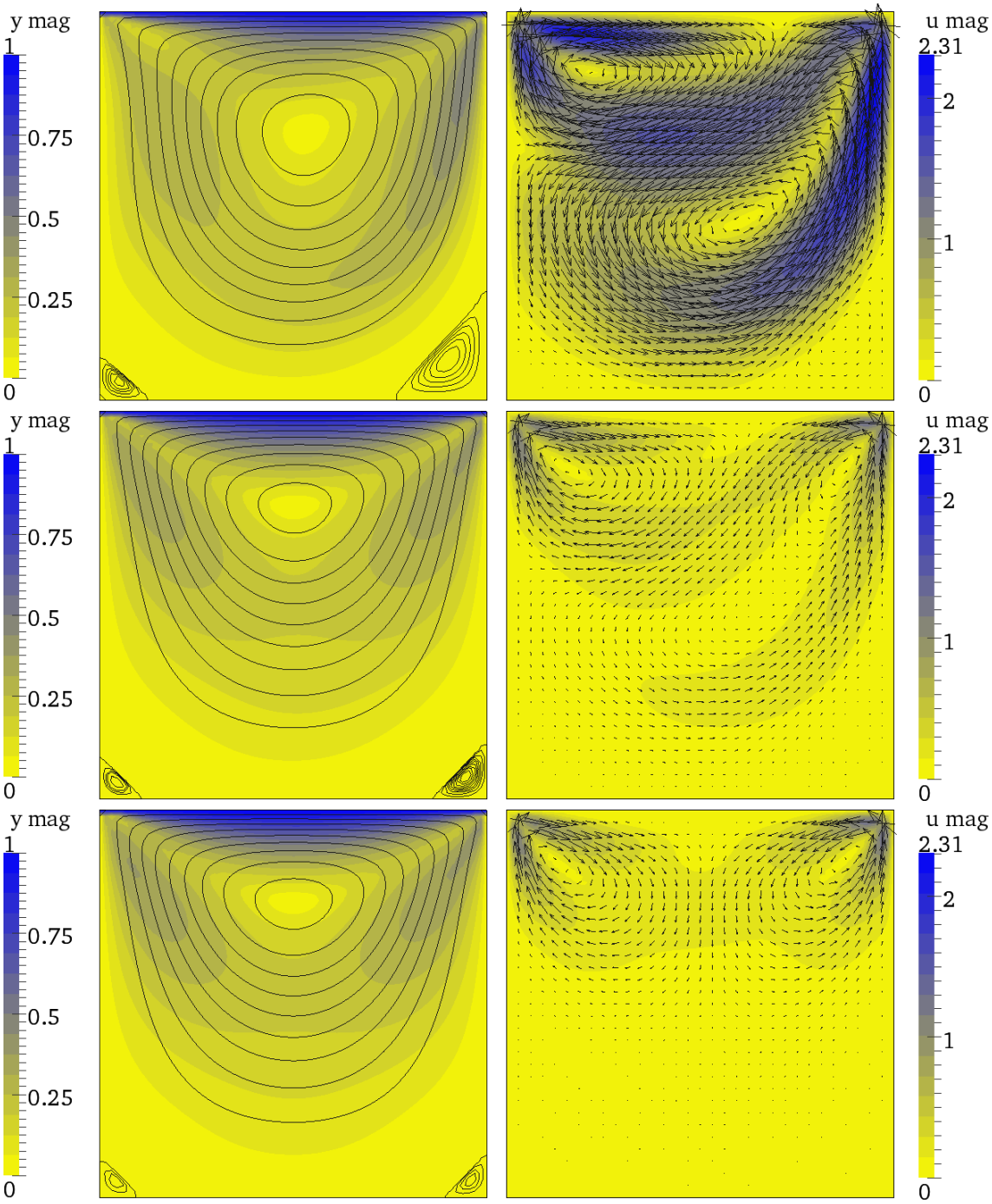

Figure 7.6: 'Driven–Cavity' example, controlled flow at $t = 0.0625$ (top), $t = 0.25$ (centre) and $t = 0.5$ (bottom). Left: Streamlines with primal velocity magnitude in the background. Right: Control.

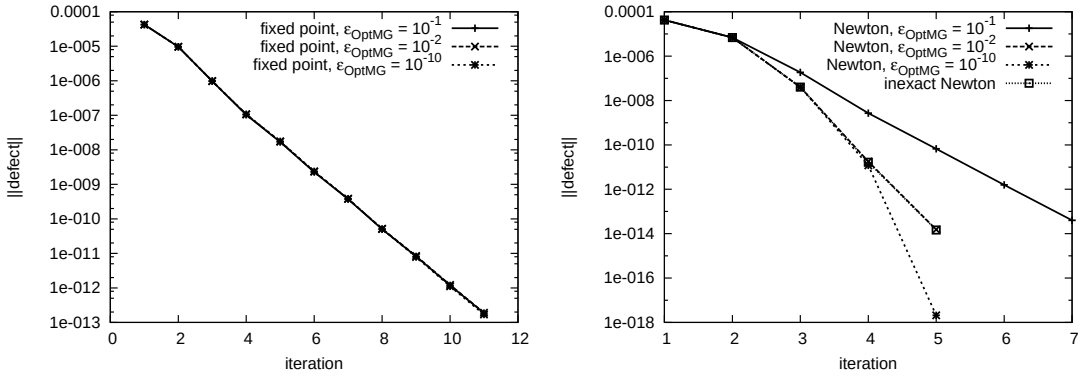

Figure 7.7: Convergence of the nonlinear solver for different ε_{OptMG}. Logarithmic scale on the y-axes. Left: standard fixed point iteration. Right: Newton and inexact Newton.

because the residual was small enough to reach the region of linear convergence. The inexact Newton solver stops the quadratic convergence for a different reason: the residual was small enough, so ε_{OptMG} is chosen in such a way that the nonlinear solver reaches its stopping criterion with one last step. A stronger nonlinear stopping criterion had led to quadratic convergence as well.

Table 7.11 compares the number of iterations for the nonlinear solver ('#NL'), the total number of iterations of the linear solver ('Σ#MG', summed up over all nonlinear iterations) and the total CPU time (in seconds) of the different algorithms. It is noted that the critical measure for the CPU time is the number of linear steps: Since the whole methodology does not set up the global matrix, a change in the nonlinearity does not lead to a new global matrix which is assembled in advance — a technique which is common for a simulation. Instead, the linear space-time solver assembles the local matrices in each timestep on demand. The actual update in the nonlinear loop is a rather cheap vector update, the expensive step is the solution of the linear system.

The table reveals the following facts: Due to the fast convergence of the linear solver, a fixed stopping criterion $\varepsilon_{OptMG} = 10^{-2}$ with a Newton iteration is sufficient. In this example, the lowest CPU time is obtained by choosing the Newton iteration with $\varepsilon_{OptMG} = 10^{-1}$ (as it requires the fewest linear iterations). For harder problems however, where the linear solver is not that strong, this stopping criterion is not expected to lead to the lowest CPU time. On the other hand, using a stronger stopping criterion than $\varepsilon_{OptMG} = 10^{-2}$ does not make sense, it only increases the number of linear iterations and the CPU time without reducing the number of nonlinear steps. However, this optimal choice depends on

	Def.Corr.			Newton		
ε_{OptMG}	#NL	Σ#MG	time	#NL	Σ#MG	time
10^{-1}	10	10	5229	6	6	4433
10^{-2}	10	11	5712	4	8	5780
10^{-10}	10	96	39442	4	34	21460
inexact Newton				4	8	5914

Table 7.11: Convergence behaviour of the standard fixed point and Newton iteration (top) as well as the inexact Newton (bottom). The inexact Newton does not depend on ε_{OptMG}.

the stopping criterion ε_{OptNL} of the nonlinear solver: For a lower value of ε_{OptNL}, lower values of ε_{OptMG} can make sense due to a faster convergence in the last couple of steps.

In this context, the inexact Newton gives a good compromise between convergence speed and simplicity. This algorithm automatically chooses ε_{OptMG} in a quasi-optimal way, thus leading to the same convergence[1] as a manually chosen $\varepsilon_{OptMG} = 10^{-2}$. From a practical point of view, this type of solver is therefore the most preferable as the choice of ε_{OptMG} is avoided without any considerable increase in the CPU time.

7.2.2. Influence of the regularisation parameters

As it was shown in Section 7.1.2, the choice of the regularisation parameters α and γ has an influence to the convergence properties of the linear solver. The nonlinear solver on the other hand depends on the solutions of the linear solver. However, if the Newton solver is used, the convergence of the nonlinear solver is not expected to (indirectly) depend on the regularisation parameters since Newton should converge quadratically as soon as the region of quadratic convergence is reached. This is verified next.

Convergence test for different regularisation parameters Consider Example 7.2 with the KKT system discretised using the $\tilde{Q}_1/Q_0$ element pair and the implicit Euler timestepping scheme. The fine mesh is defined by a regular mesh with $(h, k) = (1/64, 1/64)$. A mesh hierarchy with three levels is created by 1:1 coarsening down to $(h, k)_{\text{coarse}} = (1/16, 1/16)$. To process the nonlinearity, an inexact Newton iteration is used with $\varepsilon_{\text{OptNL}} = 10^{-8}$ characterising the stopping criterion. The inner space-time multigrid preconditioner solves the linear systems using a BiCGStab(FBGsPrec,NSM=4) smoother.

Depending on different parameters α and γ, the number of nonlinear iterations ('#NL') and the total number of linear iterations ('Σ#MG', summed up over all nonlinear iterations) is measured, see Table 7.12. A behaviour similar to Section 7.1.2 for the linear solver is observed here: The number of linear iterations increases with decreasing α and increasing γ. However, the nonlinear solver does not show any dependence on the regularisation parameters; the number of nonlinear iterations remains essentially constant for all combinations of α and γ which is in line with the expectations.

α	1.0		0.1		0.01	
γ	#NL	Σ#MG	#NL	Σ#MG	#NL	Σ#MG
0.0	3	4	4	6	4	8
1.0	4	5	4	7	4	9
10.0	4	7	4	8	4	9

Table 7.12: Convergence behaviour of the inexact Newton iteration for different α and γ.

7.2.3. Optimisation and simulation

From a practical point of view, the total costs of the optimisation are important. In Section 7.1.1 it has already been experimentally shown that the solver methodology introduced here has linear complexity, and that the CPU time was about a factor $5-12$ more expensive

[1] Although the number of iterations is the same for the inexact Newton as for the manual choice $\varepsilon_{OptMG} = 10^{-2}$, the CPU time is slightly higher. This slight increase stems from the stopping criterion on the coarse grid, which is also chosen adaptively for the inexact Newton.

than a simulation — at least in the linear case. The same should also hold for the nonlinear case (possibly with a different ratio of the simulation and the optimisation), and this is fulfilled if the number of linear and nonlinear iterations is independent of the refinement level. The focus of this section will therefore be on a comparison of the optimisation with a corresponding simulation for different levels of refinement.

CPU time comparison test Consider Example 7.2 with the regularisation parameters of the optimal control problem set to $\alpha = 0.01$ and $\gamma = 0.0$. The KKT system is discretised with $\tilde{Q}_1/Q_0$ and Q_2/P_1^{disc} in space and implicit Euler and Crank–Nicolson timestepping scheme in time. To process the nonlinearity, a standard Newton solver[2] is chosen. Similar to previous examples, the stopping criterion of the nonlinear solver is characterised by $\varepsilon_{\text{OptNL}} = 10^{-8}$. The linear subproblems are solved with a multigrid solver using a BiCGStab(FBGsPrec, NSM=4) smoother, reducing the norm of the residual by two digits, i. e., $\varepsilon_{\text{OptMG}} = 10^{-2}$. From a given fine mesh, the mesh hierarchy is created by 1:1 coarsening down to $(h,k)_{\text{coarse}} = (1/16, 1/16)$.

h	k	$\#\text{NL}_{\text{opt}}$	$\Sigma\#\text{MG}_{\text{opt}}$	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
1/32	1/32	5	5	583	96	6.0
1/64	1/64	4	8	5 790	633	9.1
1/128	1/128	4	8	38 784	4 296	9.0

Table 7.13: 'Driven–Cavity' example, simulation and optimisation for the optimal control of the Navier–Stokes equations. Discretisation with $\tilde{Q}_1/Q_0$/implicit Euler.

h	k	$\#\text{NL}_{\text{opt}}$	$\Sigma\#\text{MG}_{\text{opt}}$	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
1/32	1/32	4	7	930	170	5.4
1/64	1/64	4	7	6 627	598	11.0
1/128	1/128	4	7	43 667	4 060	10.7

Table 7.14: 'Driven–Cavity' example, simulation and optimisation for the optimal control of the Navier–Stokes equations. Discretisation with $\tilde{Q}_1/Q_0$/Crank–Nicolson.

h	k	$\#\text{NL}_{\text{opt}}$	$\Sigma\#\text{MG}_{\text{opt}}$	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
1/32	1/32	4	5	4 090	602	6.7
1/64	1/64	4	4	20 811	2 717	7.7
1/128	1/128	4	6	202 074	15 451	13.1

Table 7.15: 'Driven–Cavity' example, simulation and optimisation for the optimal control of the Navier–Stokes equations. Discretisation with Q_2/P_1^{disc}/Crank–Nicolson.

For the fine grids $(h,k) = (1/32, 1/32)$, $(1/64, 1/64)$ and $(1/128, 1/128)$, Tables 7.13 to 7.15 compare the convergence properties of the optimisation solver with those of a corresponding simulation solver. The procedure is the same as in Section 7.1.1: In a first step, the solution of the KKT system is calculated, which gives a control u. In a second step, the calculated control u from the KKT system is used as right-hand side in a nonstationary simulation. The tables contain on the one hand the number of nonlinear

[2]The inexact Newton solver is also possible in this test but would lead to slightly higher CPU times.

iterations ('$\#\mathrm{NL}_{\mathrm{opt}}$') as well as the total number of linear iterations ('$\Sigma\#\mathrm{MG}_{\mathrm{opt}}$', summed up over all nonlinear iterations) of the optimisation solver. On the other hand, they depict the required CPU time for the optimisation T_{opt} and the simulation T_{sim}. The number of linear and nonlinear iterations is rather independent of the refinement level. As a result of this, the factor $C = \frac{T_{\mathrm{opt}}}{T_{\mathrm{sim}}}$ between the optimisation and the simulation is bounded. In this simple test problem, the bound amounts to $C \approx 10$–12.

7.3. Constrained Control

In the case of constrained control, the solver is expected to converge slower in general, but still, the convergence should be superlinear, cf. [151, 152]. Using the proposed solver methodology, the convergence should be level-independent for a fixed choice of α and γ.

Constrained control convergence test Consider the 'Driven–Cavity' example for the Navier–Stokes equations, Example 7.2 on page 137, with $\alpha = 0.01$, $\gamma = 0.0$ and the additional constraint

$$a \leq u_i(t,x) \leq b, \qquad i = 1,2,$$

with the two values $a := -0.5$ and $b := 0.5$ (for simplicity, the same values are used for all components of u). In the unconstrained case, the components of the control u approximately take values in the range $[-2, 3.5]$, so this really defines a limit for the values, in particular at the beginning of the time interval. The corresponding KKT system is discretised with different discretisations in space and time on a fine mesh defined by $(h,k) = (1/32, 1/32)$, $(1/64, 1/64)$ and $(1/128, 1/128)$. Figure 7.8 depicts the control u at $t = 0.0625$ and $t = 0.5$ using the Surface-LIC representation and Figure 7.9 the flow and the computed control at the time $t = 0.0625$, 0.25 and 0.5. The thick line surrounds the area where at least one constraint is active, while in the region surrounded by the thin line, both controls have to be projected. Especially at the beginning of the time interval, large areas can be observed where the control is restricted. These regions nearly vanish while the flow approaches the desired flow state.

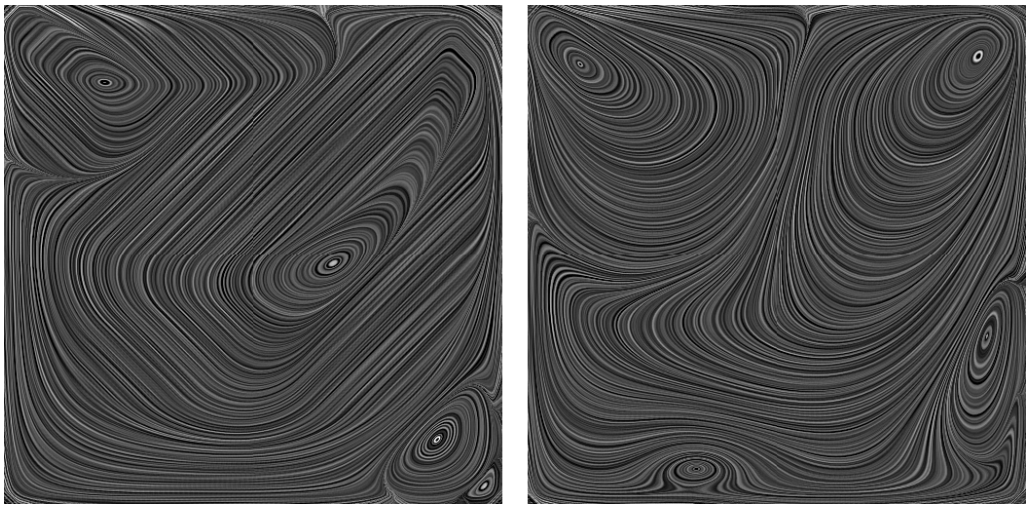

Figure 7.8: 'Driven–Cavity' example, Surface-LIC representation of u at $t = 0.0625$ (left) and $t = 0.5$ (right). Constrained control.

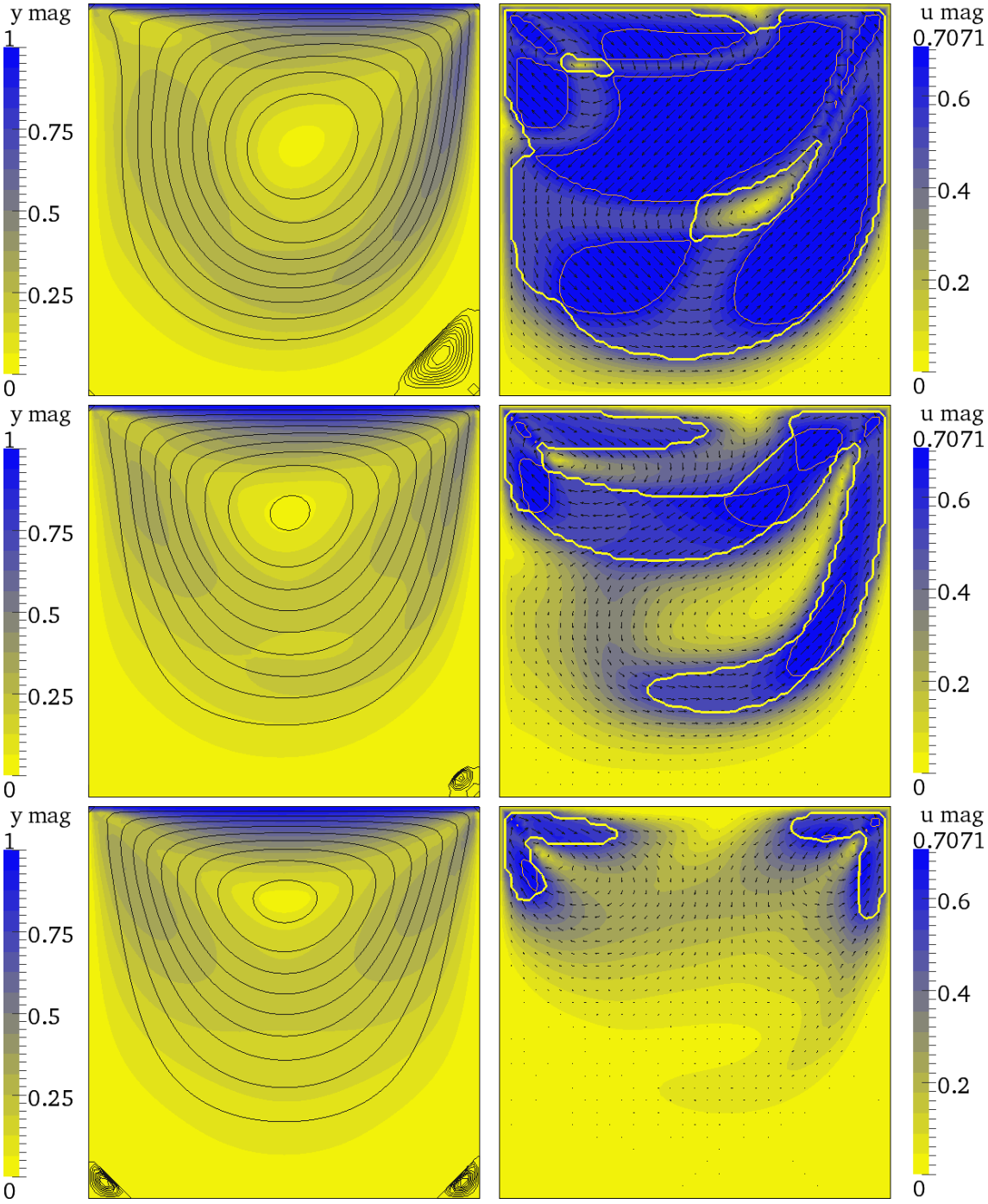

Figure 7.9: 'Driven–Cavity' example, controlled flow at $t = 0.0625$ (top), $t = 0.25$ (centre) and $t = 0.5$ (bottom). Constrained case. Left: Streamlines with primal velocity magnitude in the background. Right: Control.

For the solver test, the semismooth Newton method is chosen with the stopping criterion defined by $\varepsilon_{\mathrm{OptNL}} = 10^{-8}$. The inner space-time multigrid solver uses a fixed stopping criterion $\varepsilon_{OptMG} = 10^{-2}$, applies a BiCGStab(FBGsPrec,NSM=4) smoother and is based on a space-time hierarchy created by full space-time coarsening down to $(h,k)_{\mathrm{coarse}} = (1/16, 1/16)$.

Figure 7.10 illustrates the convergence behaviour of the nonlinear solver in the case $(h,k) = (1/64, 1/64)$. The first curve in these diagrams represents the convergence behaviour in the unconstrained case and serves as a reference for quadratic convergence. The other two curves correspond to the two methods introduced in Section 4.2 on page 87ff: 'Cub. pt. based' stands for the cubature point based discretisation method 1 and 'DOF based' for the discretisation method 2 which is based on a projection of the degrees of freedom. The left diagram uses a discretisation with $\tilde{Q}_1/Q_0$/implicit Euler, the right with $\tilde{Q}_1/Q_0$/Crank–Nicolson. Although not showing quadratic convergence as in Section 7.2.1, the solver still converges superlinearly for both discretisation methods. Comparing the two cases, the reduction of the norm of the initial residual is similar in this example. Hence, the convergence of the semismooth Newton method is rather independent of the method of discretisation.

Tables 7.16 to 7.18 summarise the convergence behaviour of the solver, on the one hand for the two projection methods, on the other hand for the three space-time discretisations $\tilde{Q}_1/Q_0$/implicit Euler, $\tilde{Q}_1/Q_0$/Crank–Nicolson and Q_2/P_1^{disc}/Crank–Nicolson. For different levels of refinement, the number of nonlinear ('#NL') and total number of linear ('Σ#MG') iterations remain rather constant, although the convergence is slightly faster on very coarse levels. Concerning the discretisation method, the tables confirm that there is no considerable difference in the convergence properties visible whether the 'Cub. pt. based' or the 'DOF based' method is used.

The number of multigrid steps per nonlinear step (Σ#MG / #NL, not depicted in the tables) is approx. two if the space discretisation is carried out with the $\tilde{Q}_1/Q_0$ finite element pair and approx. one if the Q_2/P_1^{disc} finite element pair is used (Table 7.18). This behaviour is similar to the unconstrained case (cf. Tables 7.13 to 7.15), thus, the projection operators in the constrained case do not disturb the convergence of the space-time multigrid method.

The CPU time ('time') increases by a factor ≈ 8 per level which fits to the experiences from the uncontrolled case. As expected, the 'Cub. pt. based' method is slightly more expensive than the 'DOF based' method due to the additional assembly time. This can be seen in the higher CPU time per multigrid step (time / Σ#MG, not depicted in the tables).

7.4. Summary and conclusions

This chapter has provided an analysis based on the 'Driven–Cavity' example. Starting with the optimal control of the Stokes equations, the space-time multigrid solver has been considered at first. The robustness of the solver has been analysed with respect to the choice of the regularisation parameters, the choice of the multigrid hierarchy and the anisotropy in the space-time mesh. It has been shown that the solver converges fast for a large variety of regularisation parameters. Concerning the multigrid hierarchy, the pure time coarsening strategy and the strategy to coarsen in space once per two time coarsenings have been demonstrated to be usually more robust than full space-time coarsening. Furthermore, the anisotropy tests have indicated that the choice of large timesteps in relation to the resolution of the spatial mesh are advantageous. Thus, higher order timestepping techniques are highly favourable: Without significant loss in the accuracy, such schemes allow to re-

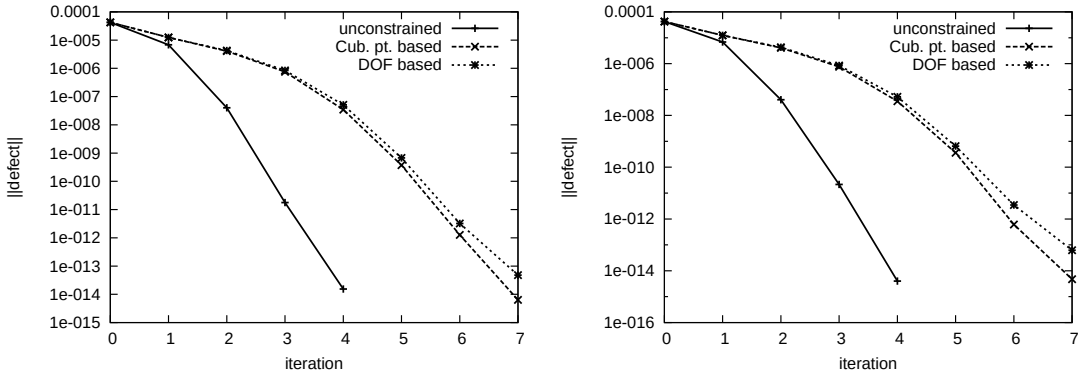

Figure 7.10: Convergence of the nonlinear solver for different projection methods in the constrained case. Logarithmic scale on the y-axes. Left: Discretisation with $\tilde{Q}_1/Q_0$/implicit Euler. Right: Discretisation with $\tilde{Q}_1/Q_0$/Crank–Nicolson.

		Cub. pt. based			DOF based		
h	k	#NL	Σ#MG	time	#NL	Σ#MG	time
1/32	1/32	7	9	1 103	8	10	1 100
1/64	1/64	7	12	9 594	7	12	8 417
1/128	1/128	7	15	84 168	7	14	67 285

Table 7.16: 'Driven–Cavity' example. Convergence of the solver for constrained control. Discretisation with $\tilde{Q}_1/Q_0$/implicit Euler.

		Cub. pt. based			DOF based		
h	k	#NL	Σ#MG	time	#NL	Σ#MG	time
1/32	1/32	7	7	1 006	8	8	1 036
1/64	1/64	7	12	11 824	7	9	8 259
1/128	1/128	7	13	94 234	7	12	76 646

Table 7.17: 'Driven–Cavity' example. Convergence of the solver for constrained control. Discretisation with $\tilde{Q}_1/Q_0$/Crank–Nicolson.

		Cub. pt. based			DOF based		
h	k	#NL	Σ#MG	time	#NL	Σ#MG	time
1/32	1/32	7	7	4 864	7	7	4 759
1/64	1/64	7	9	38 621	7	7	28 876
1/128	1/128	6	7	218 065	7	8	241 522

Table 7.18: 'Driven–Cavity' example. Convergence of the solver for constrained control. Discretisation with Q_2/P_1^{disc}/Crank–Nicolson.

duce the number of timesteps, which not only reduces the memory consumption but also increases the efficiency of the solver. Finally, in the considered example, the KKT system solver has been proven to be only five to twelve times more expensive than a corresponding simulation in terms of CPU time, independent of the refinement level. This verifies the linear complexity of the proposed solver strategy and underlines its efficiency.

In a second part, this chapter has illustrated the efficiency of the nonlinear solver being applied to the optimal control of the Navier–Stokes equations. The Newton solver has provided quadratic convergence, independent of the regularisation parameters. For constraints in the control, the solver has been shown to converge superlinearly. In the unconstrained case, the adaptive Newton scheme has numerically proven to be a good alternative to the standard Newton scheme with fixed stopping criterion for the linear preconditioner. A user can apply this type of solver in a black-box manner, there are no parameters to be adapted. Finally, it has been demonstrated that also the nonlinear solver converges with a constant number of iterations and that in the considered test example, the CPU time for the optimisation has only been by a factor ten to twelve higher than the CPU time of a corresponding simulation.

The next chapter will continue with the numerical analysis of the discretisation and the solver based on a more complex example. Benchmark problems of the 'Flow-around-cylinder' type known from CFD will be defined and used for an analysis of the accuracy of the discretisation and the efficiency and robustness of the solver.

8

The KKT solver in practice

This chapter is the concluding chapter for the numerical tests. Chapter 5 and 6 have started with the numerical analysis of the solver for the optimal control of the heat equation and the Stokes equations and illustrated the influence of the space-time discretisation to the global error. Chapter 7 has continued with the solver analysis for the optimal control of the Stokes and Navier–Stokes equations for more general example problems from CFD.

So far, all three chapters have discussed the solver configuration independently of the discretisation. This chapter combines the experiences from these chapters to find a compromise between the accuracy of the discretisation and the effort that has to be invested to solve the underlying KKT system. Basically, advices should be given to questions like:

- In how far can the described methods be applied also to more general problems?
- In how far does the choice of the discretisation quantitatively influence the solution?
- How does the choice of the regularisation parameters quantitatively influence the solution and in comparison, what is the numerical effort that has to be invested in the numerical solver to solve the underlying systems?

Generally said, in the course of the analysis, a rough picture should be drawn of what can be solved with the proposed solver methodology, how the solutions look like, which amount of available resources (e. g., in terms of CPU time) leads to which accuracy in the solution and how to find a compromise.

Outline

The first two sections, Section 8.1 and Section 8.2, are of introductory nature. A basic 'Flow–Around–Cylinder' test at Re=20 is defined, and the influence of the regularisation parameters in the KKT system is analysed with respect to the discretisation and the robustness of the solver. As a basic measure for the accuracy on different refinement levels and for the influence of parameters, drag and lift coefficients based on the forces acting on the cylinder are introduced.

The next two sections, Sections 8.3 and 8.4, contain the main discussion of this chapter. In the beginning of Section 8.3, a nonstationary benchmark problem of the 'Flow–Around–Cylinder' type at Re=100 with an oscillating initial condition is defined. With the help of an accurate space-time discretisation, reference values for the drag and lift coefficients are computed. These quantify errors introduced by different space-time discretisations and refinement levels.

In a second step, Section 8.4 contains a discussion about the efficiency and robustness of the solver, in particular, in relation to the accuracy of the underlying discretisation. The

section compares the computational time of optimisation and simulation for different discretisations. It is shown that for this more general benchmark problem, a factor of 20–50, in most cases even < 30, between optimisation and simulation is reached. Furthermore, the convergence behaviour of the solver is analysed with respect to the space-time discretisation, the coarsening strategy of the space-time mesh and the nonlinear solver. The main results obtained here are as follows: i) The implicit Euler or Crank–Nicolson scheme should be preferred to the semi-explicit Euler scheme for stability reasons, ii) simultaneous coarsening in space and time usually lead to the best CPU times and iii) the adaptive Newton gives again a good compromise between simplicity for the end user and efficiency of the solver. In a final step, the section expresses the error in the solution in relation to the CPU time needed to solve the underlying KKT system. This gives an advice to the balancing of accuracy and numerical effort by the choice of the discretisation.

The chapter closes with an appendix section which deals with stabilisation operators for the convection. Stabilisation is a crucial issue for the higher Reynolds number case since instabilities in convection dominated flows usually lead to wrong solutions and convergence problems in the solver. This section applies the edge-oriented stabilisation technique which can be found, e. g., in [124], and analyses its effect on the accuracy of the solution as well as on the efficiency of the solver. It is shown that this type of stabilisation is well suited for the application to optimal control problems in CFD.

8.1. Basic test configurations

One of the central test configurations used in this chapter is the following 'Flow–Around–Cylinder' example, based on a DFG benchmark [68, 128, 142, 146].

8.1 Navier–Stokes equations, 'Flow–Around–Cylinder' example Consider the optimal control of the Navier–Stokes equations including end time observation, see Section 4.1 on page 81ff. The spatial domain is described by a rectangle without an inner cylinder,

$$\Omega := (0, 2.2) \times (0, 0.41) \setminus \overline{B_r(0.2, 0.2)}, \quad r = 0.05,$$

see Figure 8.1. The boundary of this domain is decomposed into five parts:

$$\begin{aligned}
\Gamma_1 &:= \{0\} \times [0, 0.41] &&= \{(0, x_2) \in \mathbb{R}^2 \,|\, x_2 \in [0, 0.41]\},\\
\Gamma_2 &:= (0, 2.2] \times \{0\} &&= \{(x_1, 0) \in \mathbb{R}^2 \,|\, x_1 \in (0, 2.2]\},\\
\Gamma_3 &:= \{2.2\} \times (0, 0.41) &&= \{(2.2, x_2) \in \mathbb{R}^2 \,|\, x_2 \in (0, 0.41)\},\\
\Gamma_4 &:= (0, 2.2] \times \{0.41\} &&= \{(x_1, 0.41) \in \mathbb{R}^2 \,|\, x_1 \in (0, 2.2]\},
\end{aligned}$$

and

$$\Gamma_5 := \partial B_r(0.2, 0.2).$$

The boundary conditions are defined as follows: $y(x, t) := (0, 0)$ for $x \in \Gamma_2 \cup \Gamma_4 \cup \Gamma_5$, do-nothing boundary conditions on Γ_3 and a parabolic inflow profile with maximum velocity $U_{\max} := 0.3$ on Γ_1. The time interval for this test case is $[0, T]$ with $T = 0.5$. Similar to the 'Driven–Cavity'-example, the initial flow is the stationary Navier–Stokes flow at $\nu = 1/1000$ while the target flow is the stationary Stokes flow. The viscosity parameter in the optimisation is set to $\nu = 1/1000$ as well (resulting in a Re=20 optimisation). Figure 8.2 depicts the mesh, the initial condition and the target flow. Table 8.1 gives an overview of the problem size.

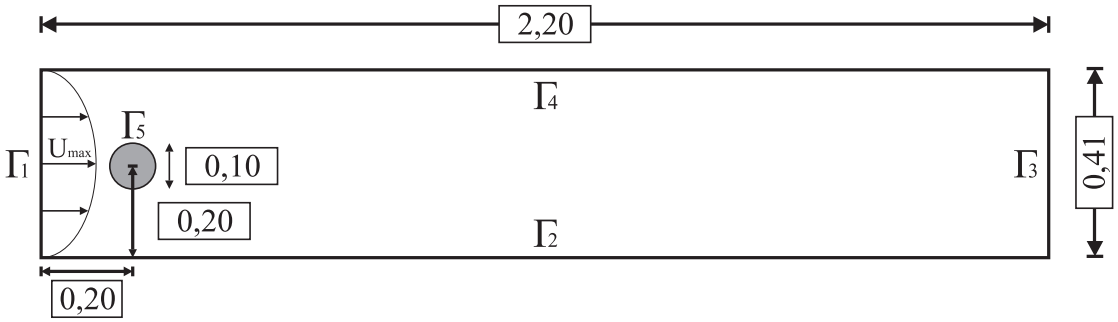

Figure 8.1: Benchmark configuration 'Flow–Around–Cylinder'.

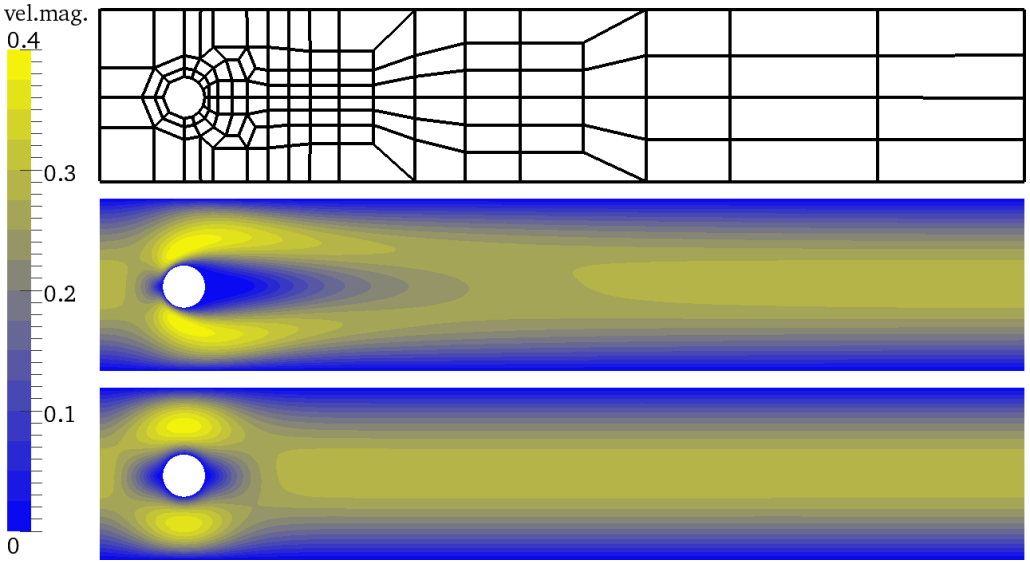

Figure 8.2: 'Flow–Around–Cylinder' control. Top: Coarse mesh, space-level one. Centre: Initial condition at $t = 0$, velocity magnitude, Navier–Stokes at Re $= 20$. Bottom: Target flow, velocity magnitude, Stokes.

Level	sp.lv.	#NEL	#dof(space)	k	#int.
1	2	520	5 408	1/32	16
2	3	2 080	21 216	1/64	32
3	4	8 320	84 032	1/128	64
4	5	33 280	334 464	1/256	128

Table 8.1: 'Flow–Around–Cylinder' control. Problem size for different refinement levels. Refinement level in space ('sp.lv.'), number of elements, number of degrees of freedom in space ('#dof(space)'), number of time intervals ('#int'). Space-discretisation with $\tilde{Q}_1/Q_0$.

8.2. Influence of the regularisation parameters

The following analysis gives a rough overview about the influence of changes in the regularisation parameters α and γ to the solution. Consider Example 8.1, discretised with $\tilde{Q}_1/Q_0$/implicit Euler on refinement level three in space-time (which corresponds to refinement level four in space and 64 time intervals).

Figure 8.3 shows the streamlines of the velocity field in the wake of the cylinder for the initial condition (left) and the target velocity field z (right). Figure 8.4 and 8.5 depict the primal velocity field of the solution of the KKT system at $t = 1/64$, 0.25, 0.375 and 0.5. Starting from the Navier–Stokes solution, the streamlines of the Stokes solution are nicely reproduced at $t = 0.25$ and 0.375 up to rather small details. Only at the end due to $\gamma = 0.0$, the accuracy of the approximation deteriorates slightly.

Tables 8.2 to 8.5 illustrate $||y_\sigma - z||_\mathcal{Q}$, $||u||_\mathcal{Q}$, $||y_\sigma(T) - z(T)||_\Omega$ and $J(y_\sigma, u_\sigma)$ for different values of α and γ. The discrete velocity is denoted by y_σ and the discrete control by u_σ. The behaviour of the errors and the values of the functionals can be summarised as follows.

- A reduction of α by the factor ten reduces the error $||y_\sigma - z||_\mathcal{Q}$ roughly by a factor two. A reduction of γ barely influences $||y_\sigma - z||_\mathcal{Q}$.
- For fixed α, doubling γ reduces $||y_\sigma(T) - z(T)||_\Omega$ roughly by a factor 1.5–2.
- For decreasing α and increasing γ, the amount of the control $||u||_\Omega$ increases. As expected, the costs for the control are dominated by the weight α; a reduction of α by a factor ten increases the costs by an order of magnitude. For 'large' values of α (around 1.0), a certain influence of the weight γ can be observed. This nearly disappears for small values of α.
- The functional $J(y_\sigma, u_\sigma)$ is dominated by the control costs $||u||_\Omega$ and has a similar behaviour. Only for small values of α (where $||u||_\Omega$ is rather constant), the influence of $||y_\sigma(T) - z(T)||_\Omega$ to $J(\cdot)$ can clearly be observed.

Table 8.6 finally gives an overview about the convergence behaviour of the Newton solver which is used for solving the nonlinear systems. The solver is configured to reduce the norm of the nonlinear residual by eight digits, $\varepsilon_{\mathrm{OptNL}} = 10^{-8}$, using a space-time multigrid solver that reduces the norm of the residual by two digits, $\varepsilon_{\mathrm{OptMG}} = 10^{-2}$. The preconditioner in space is BiCGStab(PSCPrecDiag,NSM=4) in most cases, but for $\alpha = 0.001$ the solver automatically repeats a couple of calculations with BiCGStab(PSCSmootherFull,NSM=4) since the spatial problems are difficult to solve.

The weight α has the strongest influence to the convergence properties of the solver. A smaller value of α leads to an increased number of iterations. For a fixed α, the value $\gamma = 0.5$ leads to the smallest number of iterations. For smaller or larger values of γ, the performance of the solver is usually worse. It is noted that this behaviour does not contradict to the results from Section 7.1.2 on page 133ff: The observation concentrates on γ in the range $[0, 2]$ in this test while in Section 7.1.2, the range $[0, 1000]$ was taken. Larger values for γ lead to a worse convergence behaviour, as indicated in the last row of Table 8.6. This row illustrates the solver behaviour for $\gamma = 100$. The number of iterations of the linear solver increases in comparison to $\gamma \leq 2$. For $\alpha = 0.001$, the solver diverged due to a breakdown in the convergence of the space-time multigrid solver (marked as 'div').

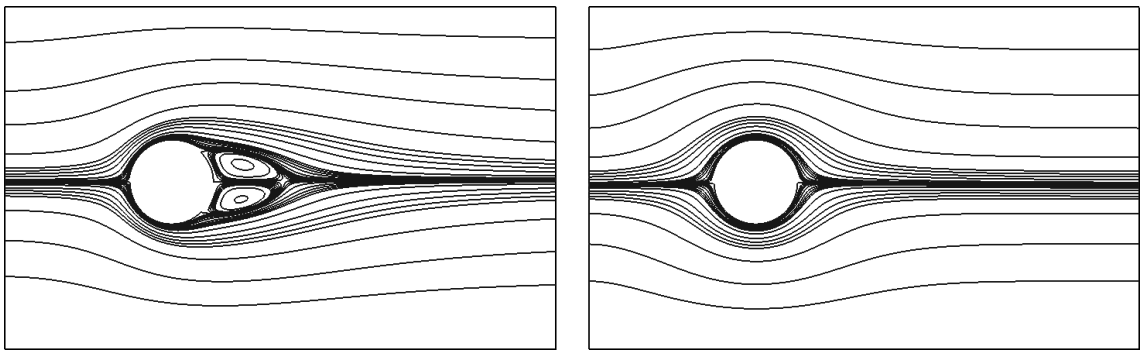

Figure 8.3: 'Flow–Around–Cylinder' control. Streamlines in the wake of the cylinder for the stationary reference flows. Left: Navier Stokes flow, initial condition. Right: Stokes flow, target flow z.

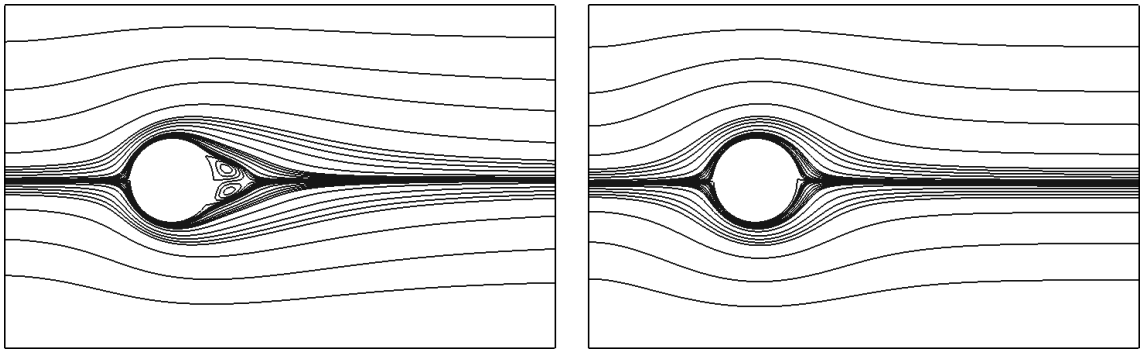

Figure 8.4: 'Flow–Around–Cylinder' control. Streamlines in the wake of the cylinder. Left: At $t = 1/64$. Right: At $t = 0.25$.

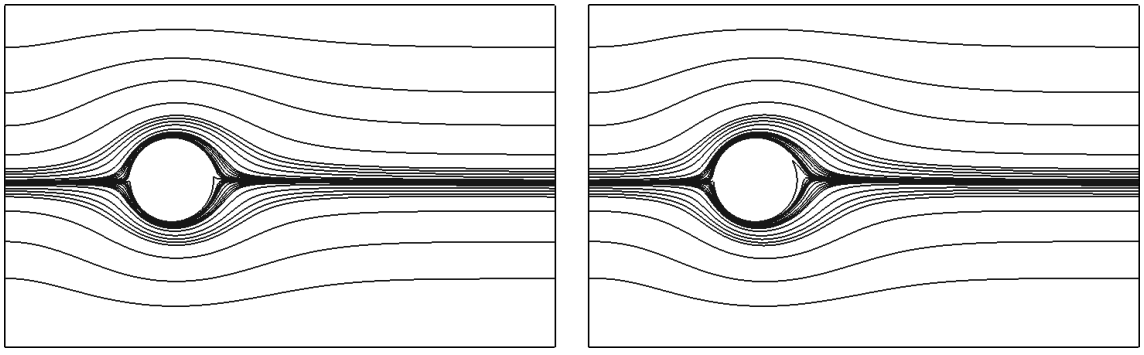

Figure 8.5: 'Flow–Around–Cylinder' control. Streamlines in the wake of the cylinder. Left: At $t = 0.375$. Right: At $t = 0.5$.

$\gamma \setminus \alpha$	1.00	0.10	0.01	0.001
0.0	1.97E-02	1.40E-02	7.01E-03	3.80E-03
0.1	1.94E-02	1.34E-02	6.93E-03	3.79E-03
0.5	1.85E-02	1.23E-02	6.89E-03	3.79E-03
1.0	1.76E-02	1.19E-02	6.88E-03	3.79E-03
2.0	1.65E-02	1.16E-02	6.87E-03	3.79E-03

Table 8.2: 'Flow–Around–Cylinder' control. $||y_\sigma - z||_\mathcal{Q}$ for different α and γ.

$\gamma \setminus \alpha$	1.00	0.10	0.01	0.001
0.0	2.72E-02	1.65E-02	5.42E-03	1.75E-03
0.1	2.64E-02	1.37E-02	3.10E-03	5.75E-04
0.5	2.35E-02	8.37E-03	1.23E-03	1.96E-04
1.0	2.08E-02	5.72E-03	7.42E-04	1.18E-04
2.0	1.69E-02	3.57E-03	4.38E-04	7.12E-05

Table 8.3: 'Flow–Around–Cylinder' control. $||y_\sigma(T) - z(T)||_\Omega$ for different α and γ.

$\gamma \setminus \alpha$	1.00	0.10	0.01	0.001
0.0	4.81E-03	2.98E-01	7.36E+00	1.27E+02
0.1	6.01E-03	3.31E-01	7.48E+00	1.27E+02
0.5	1.05E-02	4.07E-01	7.62E+00	1.27E+02
1.0	1.51E-02	4.49E-01	7.67E+00	1.27E+02
2.0	2.18E-02	4.85E-01	7.70E+00	1.27E+02

Table 8.4: 'Flow–Around–Cylinder' control. $||u||_\mathcal{Q}$ for different α and γ.

$\gamma \setminus \alpha$	1.00	0.10	0.01	0.001
0.0	2.05E-04	4.54E-03	2.71E-01	8.09E+00
0.1	2.42E-04	5.59E-03	2.80E-01	8.14E+00
0.5	3.66E-04	8.38E-03	2.91E-01	8.17E+00
1.0	4.88E-04	1.01E-02	2.94E-01	8.17E+00
2.0	6.62E-04	1.18E-02	2.96E-01	8.18E+00

Table 8.5: 'Flow–Around–Cylinder' control. $J(y_\sigma, u_\sigma)$ for different α and γ.

α	1.00		0.10		0.01		0.001	
γ	#NL	Σ#MG	#NL	Σ#MG	#NL	Σ#MG	#NL	Σ#MG
0.0	4	6	4	7	4	10	4	13
0.1	3	5	4	7	5	12	4	14
0.5	4	5	4	7	4	9	4	12
1.0	4	6	4	9	4	12	5	16
2.0	4	5	4	12	4	12	4	13
100.0	4	11	4	14	4	20	div	div

Table 8.6: 'Flow–Around–Cylinder' control. Number of nonlinear (#NL) and linear (Σ#MG) iterations of the solver.

Measuring drag and lift coefficients

Apart from the difference $||y - z||$ to a reference function in an integral norm, practitioners are often more interested in an accurate approximation y to z in terms of physical quantities. One example, which is common for 'Flow–Around–Cylinder' type benchmark configurations, is the set of forces that act on the cylinder, represented by the drag and lift coefficients. These are defined by

$$C_D = C_D(t) := \frac{2}{\rho L U_{\text{mean}}^2} \int_{\Gamma_5} \left(\rho\nu \frac{\partial y_{\tau_\Gamma}}{\partial \eta} \eta_y - p\eta_x \right) d\Gamma \tag{8.1}$$

and

$$C_L = C_L(t) := \frac{2}{\rho L U_{\text{mean}}^2} \int_{\Gamma_5} \left(\rho\nu \frac{\partial y_{\tau_\Gamma}}{\partial \eta} \eta_x - p\eta_y \right) d\Gamma \tag{8.2}$$

with $\eta = (\eta_x, \eta_y)^\mathsf{T}$ the unit normal vector on Γ_5 directing into Ω, $\tau_\Gamma = (\eta_y, -\eta_x)^\mathsf{T}$ the tangential vector on Γ_5, y_{τ_Γ} the tangential velocity, $\rho := 1$ the (by convention) constant density, L the diameter of the cylinder and U_{mean} the mean inflow velocity. In this case, $U_{\text{mean}} = 2/3\, U_{\max} = 0.2$ and $L = 0.1$ are chosen, cf. [99, 142, 146]. For small $||y - z||$, the drag and lift coefficients of y are usually expected to match those of z. This is not necessarily true as the following setting demonstrates.

Drag and lift coefficients for different regularisation parameters Figure 8.6 and 8.7 plot the drag and lift coefficients of the nonstationary simulation for different values of α and γ. The KKT system is discretised with $\tilde{Q}_1/Q_0$/implicit Euler. The crosses on the left on the diagrams indicate the drag and lift coefficients of the stationary Navier–Stokes flow, while the crosses on the right mark those of the stationary Stokes flow with $\nu = 1/1000$. At first, the behaviour of the drag is turned into focus, see Figure 8.6. For decreasing α, the whole drag approaches the drag of the stationary Stokes flow. Increasing γ influences the value $y(T, \cdot)$ and (due to the elliptic nature) also the values of the drag around $t = T$: The curve 'bends' towards the reference drag. The value $\gamma = 0.1$ can be seen as the nearly optimal value in this case as the flow remains rather stationary at the end of the time interval — a fact that was also observed in [69, 70].

For the lift coefficients in Figure 8.7 however, the situation is quite different. Decreasing α and increasing γ has the opposite effect, the lift values are 'torn away' from the reference lift. Only for the smaller value $\alpha = 0.01$, the weight $\gamma = 0.5$ for the end time observation has an effect and reduces $C_L(T)$ in direction of the reference as one would expect.[1]

Drag and lift coefficients for different refinement levels This effect is not caused by problems with the accuracy of the discretisation. Figure 8.8 and 8.9 plot the drag and lift values for different refinement levels and different parameters of α. The parameter γ was fixed to 0.0 in this test. For every setting of α, convergence with increasing refinement level can be observed, but the limit is different from the reference drag/lift.

Conclusion This example demonstrates that controlling the error $||y - z||$ in an integral norm over the whole space-time cylinder does not necessarily mean that local quantities

[1] For $\alpha = 0.10$ there is $C_L(\gamma = 0.0) < C_L(\gamma = 0.5)$; for $\alpha = 0.01$ the situation is opposite at the end, there is $C_L(\gamma = 0.0) > C_L(\gamma = 0.5)$.

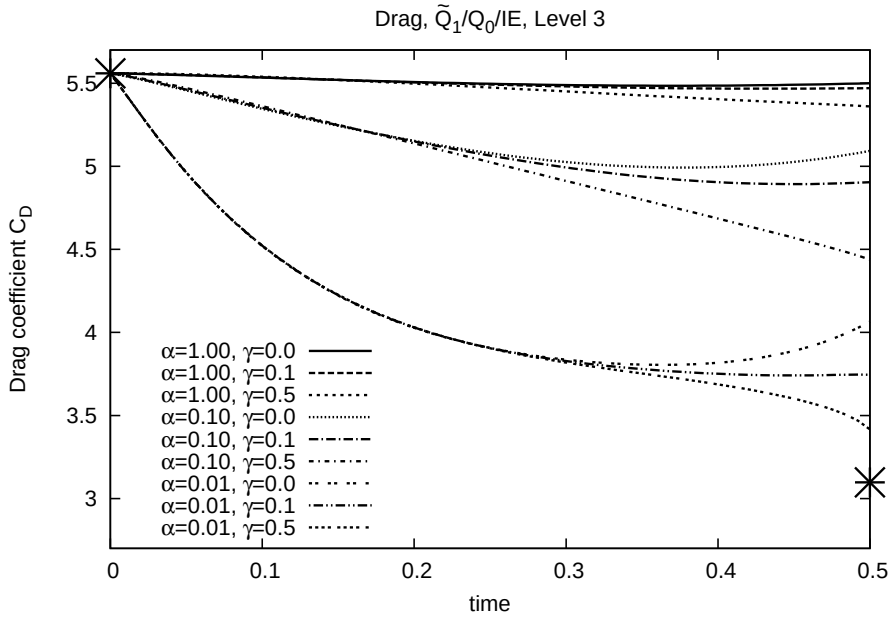

Figure 8.6: 'Flow–Around–Cylinder' control. Drag coefficients C_D for different α and γ.

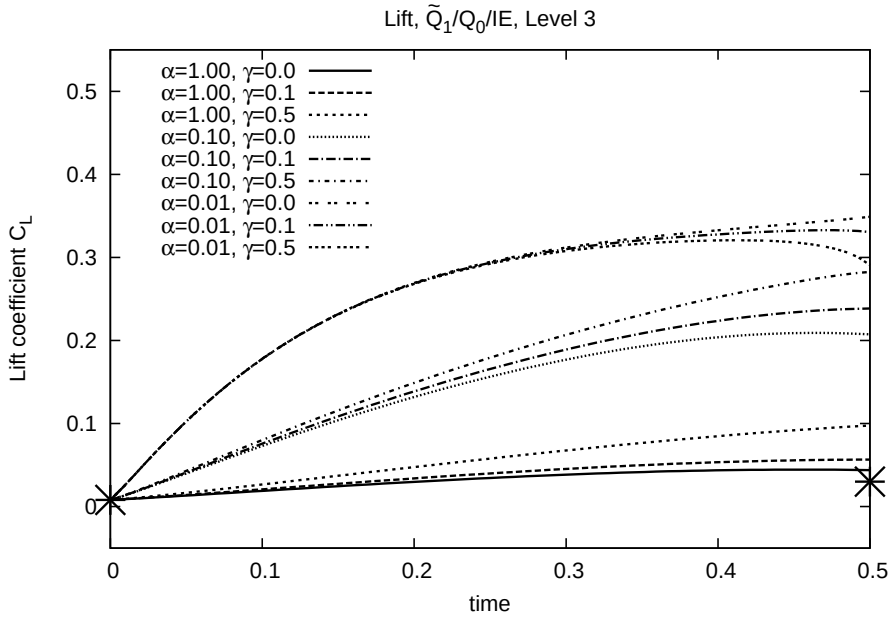

Figure 8.7: 'Flow–Around–Cylinder' control. Lift coefficients C_L for different α and γ.

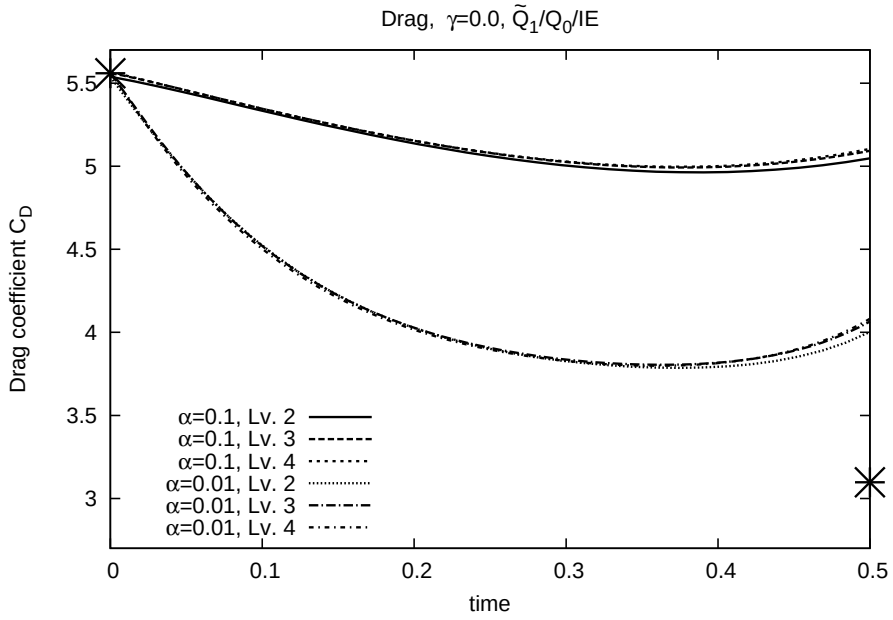

Figure 8.8: 'Flow–Around–Cylinder' control. Drag coefficients C_D for different α and refinement level.

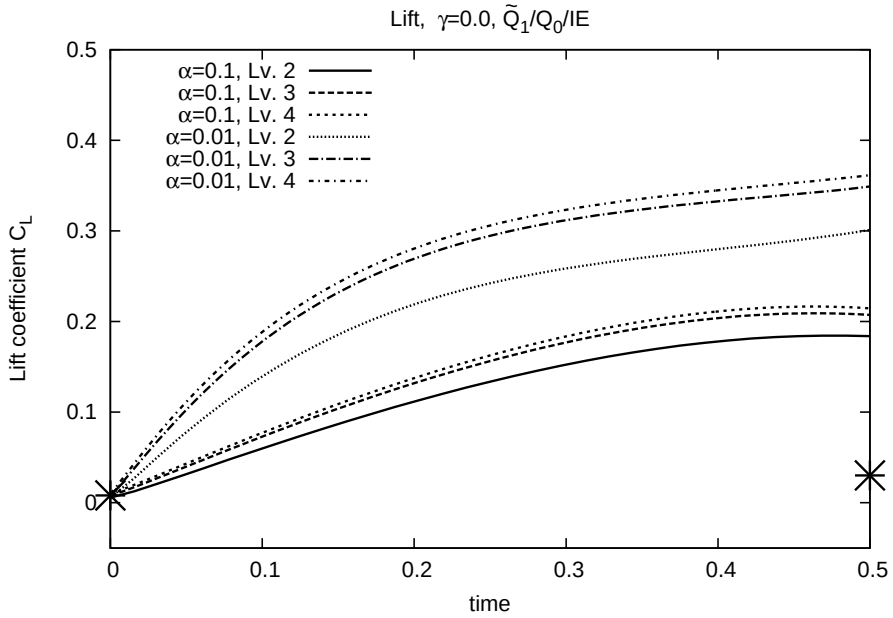

Figure 8.9: 'Flow–Around–Cylinder' control. Lift coefficients C_L for different α and refinement level.

depending on y can be controlled implicitly. The forces acting on the cylinder in this example mainly depend on the pressure acting on the cylinder. The lift coefficient in particular mainly depends on the pressure difference on the top and the bottom of the cylinder, the drag coefficient on the pressure difference on its front and rear. Figure 8.10 illustrates the pressure, the primal velocity field around the cylinder, the streamlines of the primal velocity and the Surface-LIC representation of the velocity field. The left column represents the target velocity field $z(t,\cdot)$ and the corresponding pressure, while the right column shows the controlled Navier–Stokes velocity field $y(t,\cdot)$ with the associated pressure at $t = 0.25$. Visually, the velocity y (centre/bottom right) is close to the target z (centre/bottom left) as it should be. However, the pressure corresponding to y (top right) is very different from the pressure corresponding to z (top left). Therefore, the expectation that physical quantities depending on the controlled velocity field y (like drag and lift values) match any expected physical quantity depending on the target velocity field z should be taken with care.

8.3. A nonstationary benchmark problem

While the previous chapters have dealt with controlling a stationary initial flow to a stationary target flow, the method is also designed for real nonstationary problems. One example is the control of a fully nonstationary flow with vortex shedding at Re=100 to a stationary target flow. This setting was originally introduced in [85] and is an adaption of the nonstationary 'Flow–Around–Cylinder' problem introduced in [146]. In the following, this example is formulated in a way which is suitable for benchmarking. The different discretisation techniques described and introduced in this work are applied to show quantitative results about their effectiveness.

The analysis starts with a proper benchmark configuration:

8.2 'Flow–Around–Cylinder' control for an oscillating initial condition Consider the basic 'Flow–Around–Cylinder' configuration from Example 8.1. The inflow at Γ_4 is defined as parabolic profile with maximum inflow velocity $U_{\max} = 1.5$; using $\nu = 1/1000$, this results in a Re=100 optimisation. The initial flow y^0 is the fully developed nonstationary Navier–Stokes flow (see below), the target flow is the stationary Stokes flow.

For the generation of the initial condition y^0, the following steps are carried out. At first, a fully nonstationary simulation at a high spatial level with small timestep length is calculated, possibly also with a high order finite element, until the oscillation in the solution is fully developed, cf. [146]. (In this example, the simulation is carried out with Q_2/P_1^{disc} and the Crank–Nicolson scheme on space-level 5 with $k = 1/1600$.) The L_2 projection is used to project this solution down to the finite element space/level where the optimisation is to be carried out. Using this solution as initial condition, $T_{\text{sim}} = 0.7$ seconds of simulation time are simulated which roughly corresponds to two oscillations in the flow. (In this work, a time discretisation with the Crank–Nicolson scheme at $k = 1/800$ is used.) The solution in the second half $[0.35, 0.7]$ of the simulated time interval which shows the smallest lift coefficient is used as initial condition y^0 for the optimisation.

The quality of the calculated solution is measured using the drag and lift coefficient along the time axis. For the optimisation, a time interval $[0, T]$ with $T = 0.35$ is chosen which roughly corresponds to one oscillation in the uncontrolled flow. The

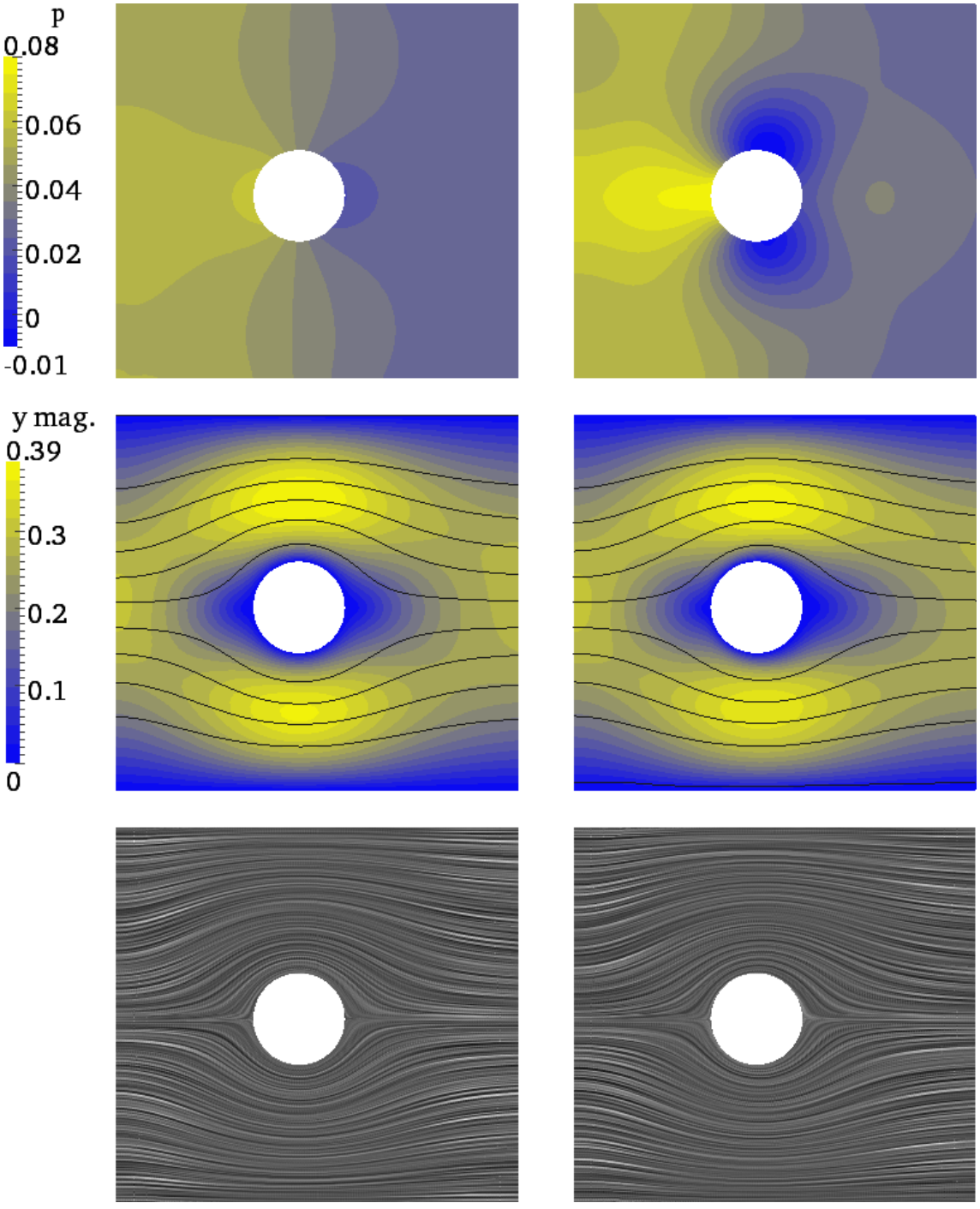

Figure 8.10: 'Flow–Around–Cylinder' control at t=0.25 for $\alpha = 0.01$, $\gamma = 0.0$. Left: Stationary Stokes flow. Right: Controlled Navier–Stokes flow. Top: Pressure. Centre: Magnitude of y and streamlines. Bottom: Surface-LIC representation of the velocity fields.

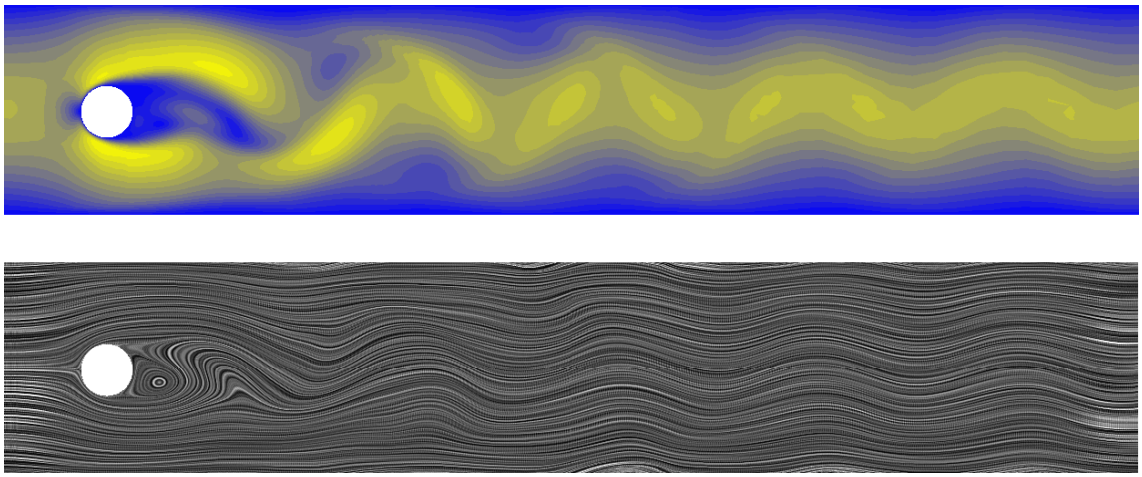

Figure 8.11: Nonstationary 'Flow–Around–Cylinder' control. Initial condition y^0. Velocity magnitude (top) and Surface-LIC representation (bottom)

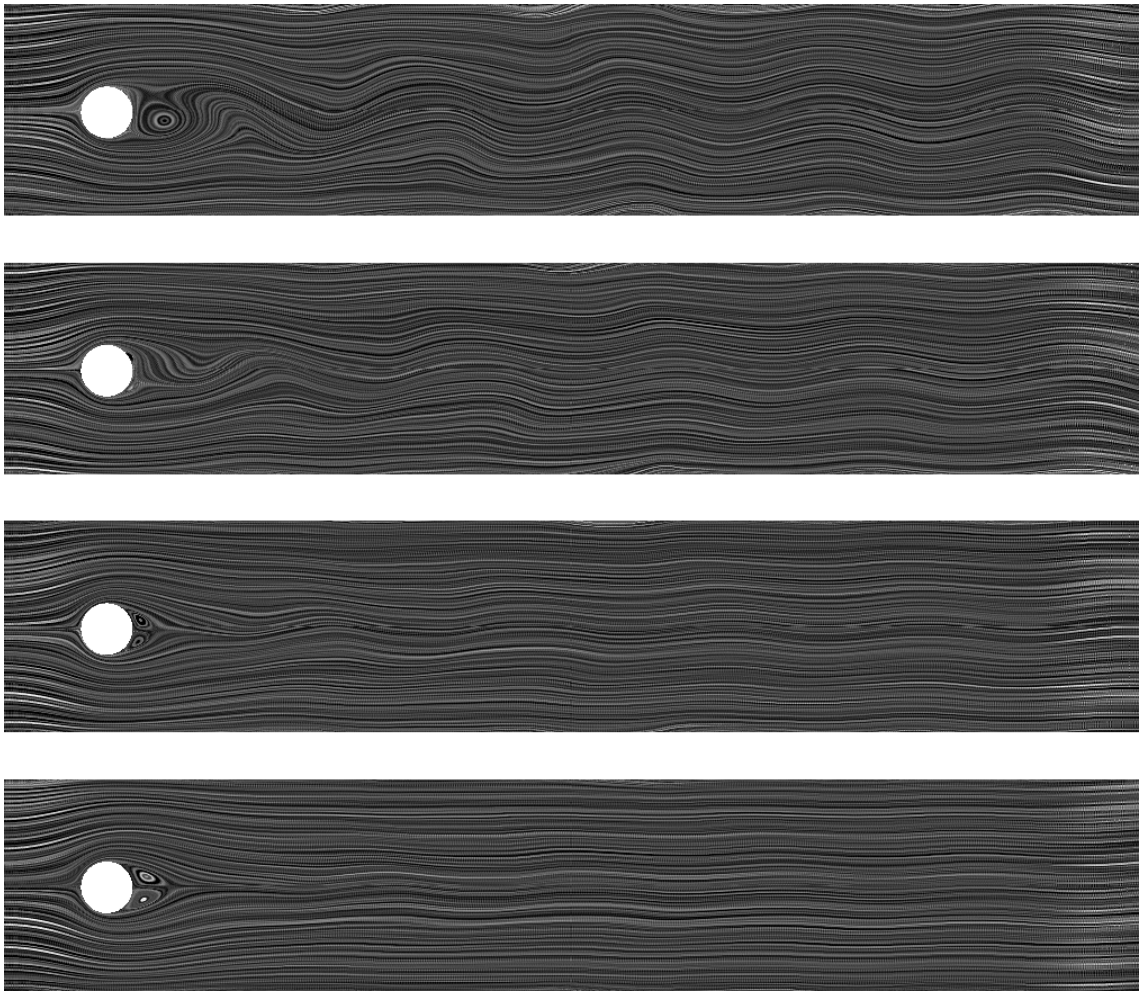

Figure 8.12: Nonstationary 'Flow–Around–Cylinder' control. Surface-LIC representation of the primal velocity field y at $t = 0.0175$, $t = 0.0875$, $t = 0.175$ and $t = 0.2625$ (from top to bottom).

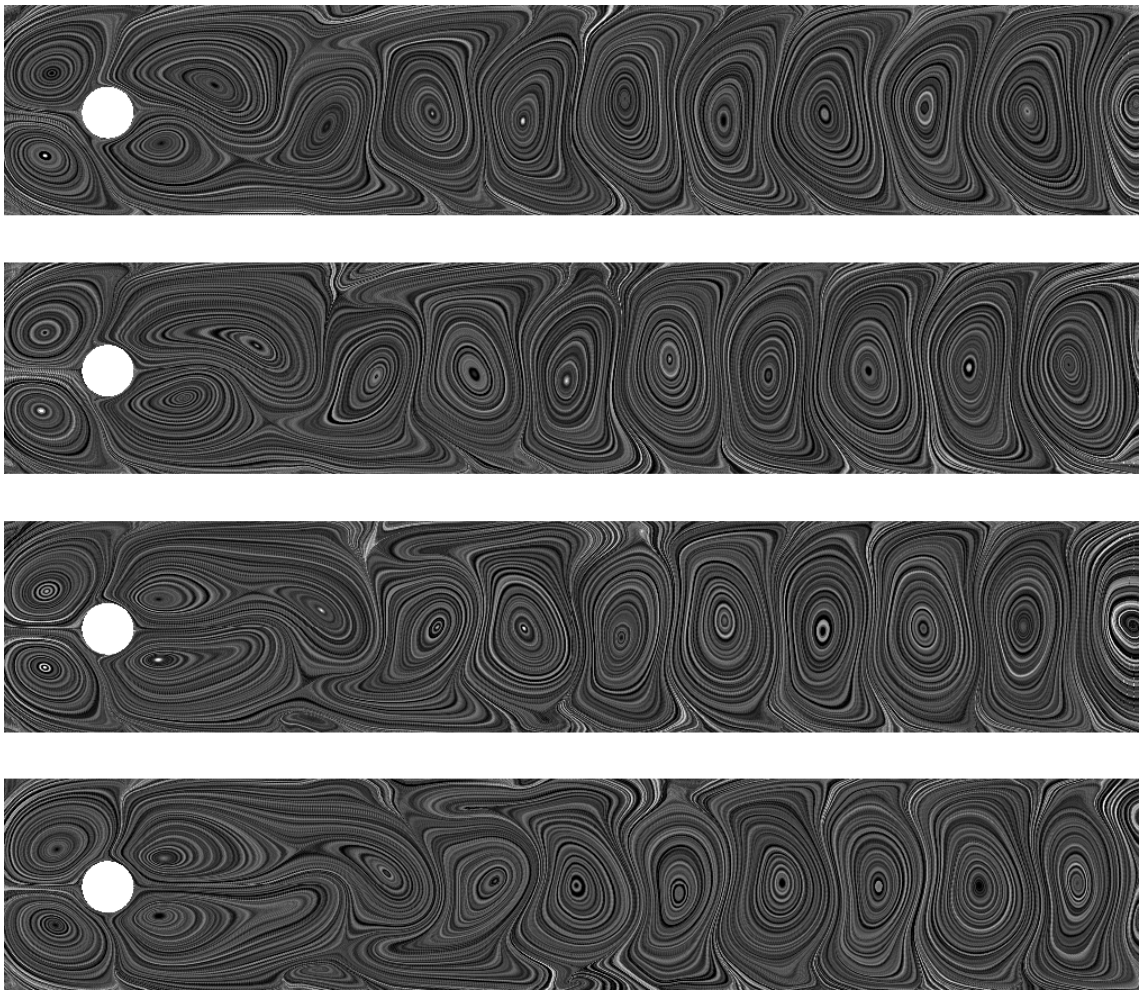

Figure 8.13: Nonstationary 'Flow–Around–Cylinder' control. Surface-LIC representation of the dual velocity field λ at $t = 0.0175$, $t = 0.0875$, $t = 0.175$ and $t = 0.2625$ (from top to bottom).

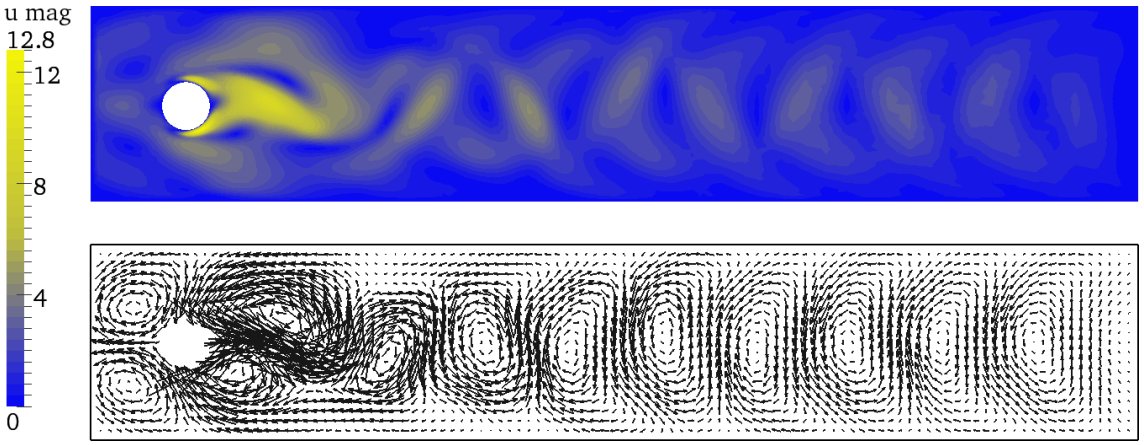

Figure 8.14: Nonstationary 'Flow–Around–Cylinder' control. Control u at $t = 0.0175$.

KKT solver starts with an initial iterate $w_0 = (y_0, p_0, \lambda_0, \xi_0)$ where (y_0, p_0) is the solution of a pure forward simulation starting from y^0 and $(\lambda_0, \xi_0) := (0, 0)$.

Figure 8.11 depicts the initial condition y^0, which shows the typical vortex shedding. Figure 8.12 and 8.13 show the Surface-LIC representation of the primal and dual velocity field, respectively, at different points in time. For these figures, the calculation was carried out with $\tilde{Q}_1/Q_0$ in space and the Crank–Nicolson scheme in time using a timestep of $k = 1/800$. In the Surface-LIC representation of the dual velocity, the typical vortices of the term $y\nabla y$ (which is approximated by λ or u, respectively) can be observed. Figure 8.14 finally depicts the control u at $t = 0.0175$.

All following tests use a Newton solver with an inner space-time multigrid solver for solving the linear subproblems. Similar to the other chapters, the Newton solver is configured to reduce the norm of the residual by eight digits, $\varepsilon_{\mathrm{OptNL}} = 10^{-8}$, and the space-time multigrid solver reduces the norm of the residual two digits per nonlinear step,[2] $\varepsilon_{\mathrm{OptMG}} = 10^{-2}$. The space-time smoother and coarse grid solver is BiCGStab(FBSimPrec,NSM=4) unless noted otherwise.[3] The space-time hierarchy is obtained from the finest space-time mesh by coarsening down to $k = 1/200$ or space-level 2. Table 8.7 gives an overview about the problem size for all refinement levels in space and in time that are used in the different numerical tests. For any combination of space and time level forming the fine mesh, the tests usually build a full space-time multigrid hierarchy with simultaneous coarsening in space and time. Those cases which apply a pure time-multigrid, i. e., without coarsening in space, are mentioned explicitly.

Space-lv.	#NEL	$\tilde{Q}_1/Q_0$ #dof(space)	Q_2/P_1^{disc} #dof(space)
2	520	5 408	11 856
3	2 080	21 216	46 592
4	8 320	84 032	184 704

time-lv.	k	#int.
1	1/200	70
2	1/400	140
3	1/800	280
4	1/1600	560

Table 8.7: 'Flow–Around–Cylinder' control. Problem size for different refinement levels in space (left) and time (right).

8.3.1. Reference calculation

In a first test, reference values for the drag and lift coefficients are determined. The spatial discretisation is carried out with the Q_2/P_1^{disc} element pair, which is expected to give a better approximation of the drag/lift coefficients than $\tilde{Q}_1/Q_0$. In time, the Crank–Nicolson discretisation (abbreviated by 'CN' for convenience) is chosen which divides the time interval of interest into 560 intervals (which corresponds to $k = 1/1600$). Figures 8.15 and 8.16 depict the drag and lift coefficients over time for different refinement levels in space, Figure 8.17 and 8.18 zoom in to subintervals in time. The finest space discretisation possible is refinement level four which consumes 12GB of memory out of the 16GB available on the machine. This calculation is taken as global reference for all future tests.

[2] Alternatively, the adaptive Newton could have been used. However, the static choice of the stopping criteria leads to slightly lower CPU times in this example, which is shown later.

[3] This is in slight contrast to the previous chapters which mostly use the more stable FBGsPrec based smoothers. On the one hand, this choice demonstrates that, although potentially being less stable, FBSimPrec based smoothers are applicable in practice as well. On the other hand, it helps to reduce the CPU time since some calculations in this chapter are extremely time consuming.

Table 8.8 illustrates these results in quantitative form. The table contains the minimum drag and the maximum lift values on the different space levels as well as the point in time where these values are taken. The last row of this table is used as a reference for later tests. The corresponding reference solution is illustrated illustrated as a thick black solid line later plozs.

discr.	sp.lv.	#int.	$t(C_D^{\min})$	$C_D^{\min}$	$t(C_L^{\max})$	$C_L^{\max}$
Q_2/P_1^{disc}/CN	2	560	0.1846875	2.41392	0.1746875	0.23401
Q_2/P_1^{disc}/CN	3	560	0.1721875	2.46917	0.1734375	0.24007
Q_2/P_1^{disc}/CN	4	560	0.1621875	2.51516	0.1734375	0.24325

Table 8.8: Minimum drag and maximum lift coefficients for different space refinements.

8.3.2. Influence of the time discretisation

In the next test, the influence of the time discretisation is analysed. For this purpose, the above nonstationary control problem is calculated for a fixed discretisation and refinement level in space ($\tilde{Q}_1/Q_0$, refinement level three, 21 216 unknowns in space) and varying time discretisation. The drag and lift coefficients are measured over time. In particular, this test uses a time discretisation with the implicit Euler (abbreviated by 'IE' for convenience) and the Crank–Nicolson scheme, each with 70 up to 560 intervals. Figures 8.19 and 8.20 depict the measured drag and lift coefficient values for the whole time interval, while Figures 8.21 and 8.22 zoom in to the time subintervals $[0.18, 0.22]$ and $[0.15, 0.21]$, respectively.

It is remarkable that independent of the timestep length, the drag and lift coefficients for the Crank–Nicolson discretisation visually match. A time discretisation with 560 intervals (black solid line) with this time discretisation scheme can therefore be used as reference values for an arbitrary discretisation in space, at least up to the level of accuracy which is considered in this work. In practice on the other hand, using a timestep $k = 1/200$ (corresponding to 70 intervals) would completely be enough to be close to the reference for a fixed space discretisation. This helps to reduce the CPU time by about 60–85 percent, see Section 8.4.

The time discretisation with the implicit Euler scheme is less accurate, which is the expected behaviour. Using a discretisation with only 70 intervals is far away from the reference calculated with the Crank–Nicolson scheme using the same space discretisation. For 560 intervals, the results are much better but still, the difference to the Crank–Nicolson discretisation can clearly be seen in the zoomed diagrams.

Table 8.9 illustrates these results in quantitative form. The table measures the minimum drag and maximum lift values for different time discretisations on different space levels as well as the point in time where these values are taken. The upper part of the table shows the result for a space discretisation with $\tilde{Q}_1/Q_0$, the last two rows give an overview about the reference result calculated with Q_2/P_1^{disc}. In the Crank–Nicolson case, the drag and lift coefficients on each space level do not change in the first three digits, independent of whether 70 or 560 timesteps are used. On space level four, the maximum lift values match even up to five digits. The accuracy of the implicit Euler time discretisation is in all cases between one and two digits lower.

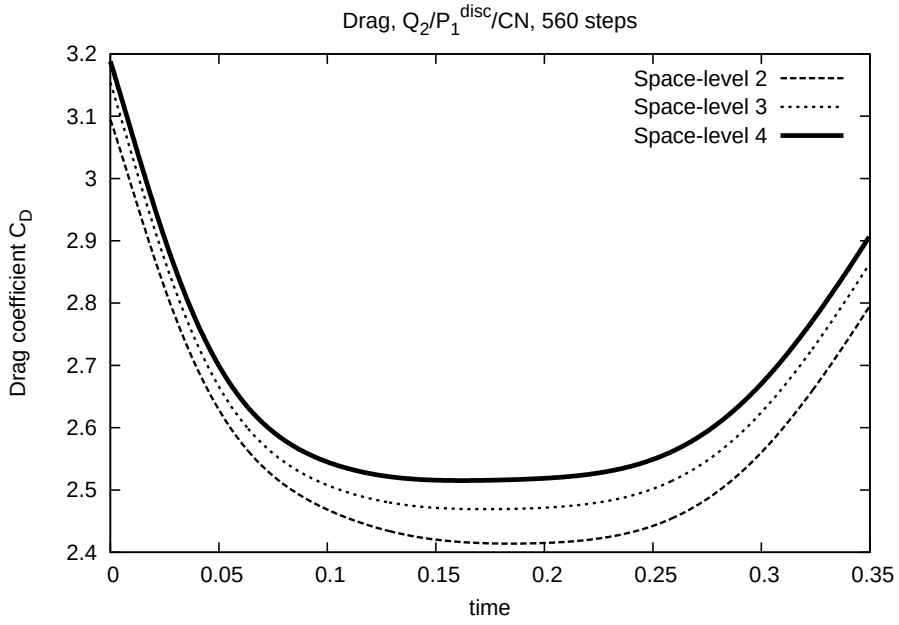

Figure 8.15: Drag coefficient C_D. $Q_2/P_1^{\text{disc}}/\text{CN}$ discretisation. Different space levels, 560 intervals.

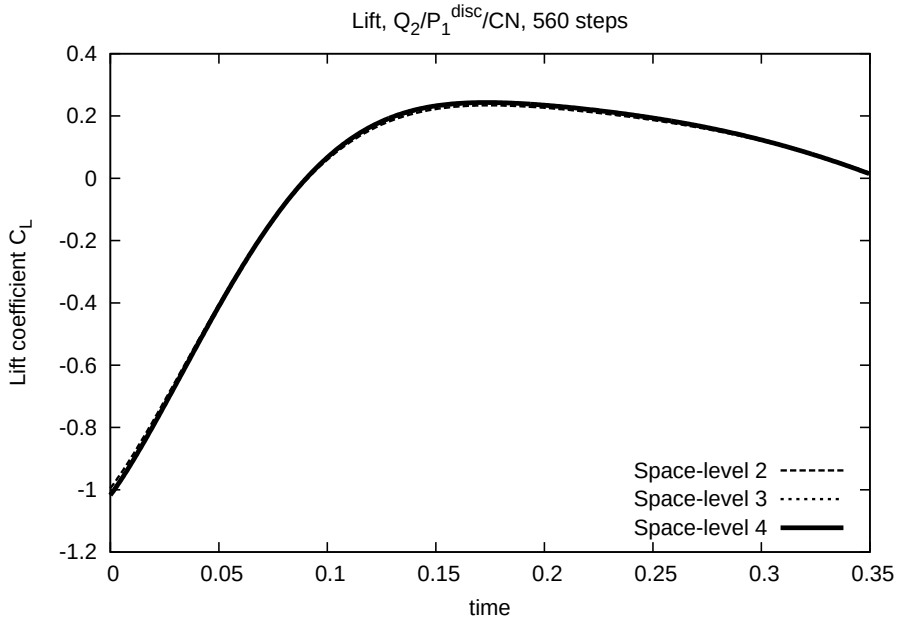

Figure 8.16: Lift coefficient C_D. $Q_2/P_1^{\text{disc}}/\text{CN}$ discretisation. Different space levels, 560 intervals.

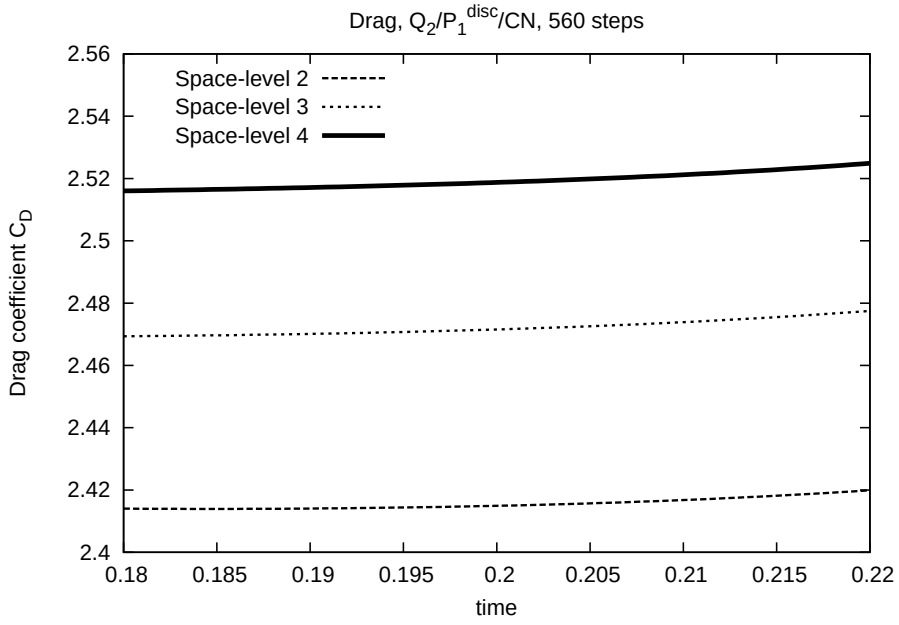

Figure 8.17: Drag coefficient C_D. $Q_2/P_1^{\text{disc}}/\text{CN}$ discretisation. Different space levels, 560 intervals. Zoom to the time interval $[0.18, 0.22]$.

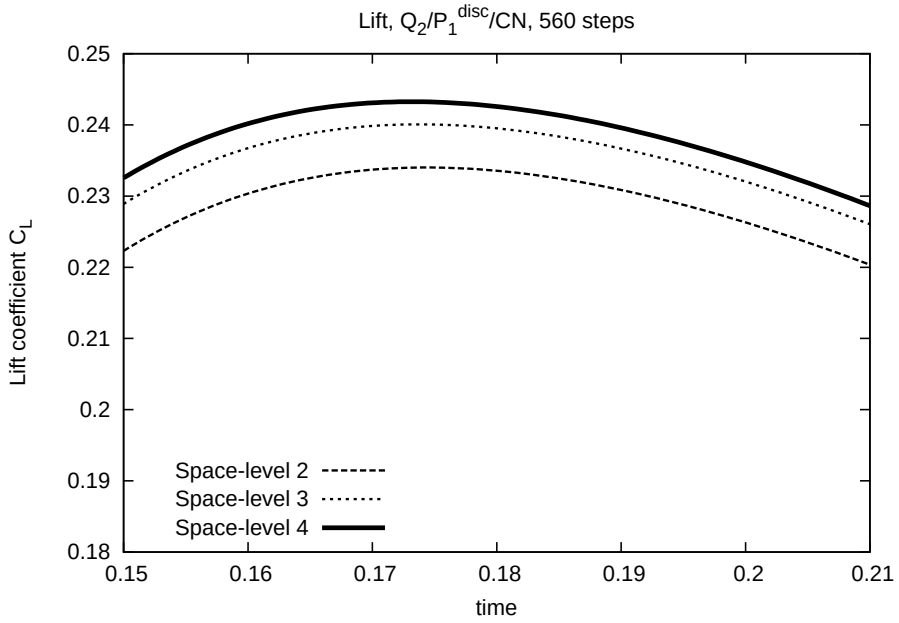

Figure 8.18: Lift coefficient C_D. $Q_2/P_1^{\text{disc}}/\text{CN}$ discretisation. Different space levels, 560 intervals. Zoom to the time interval $[0.15, 0.21]$.

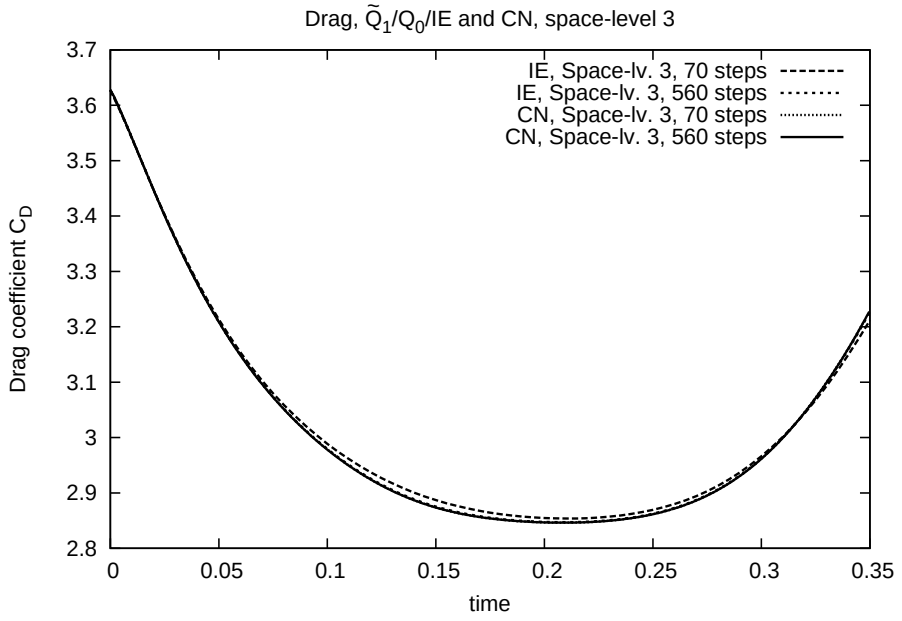

Figure 8.19: Drag coefficient C_D. Space-level three, time discretisation with the implicit Euler and Crank–Nicolson scheme.

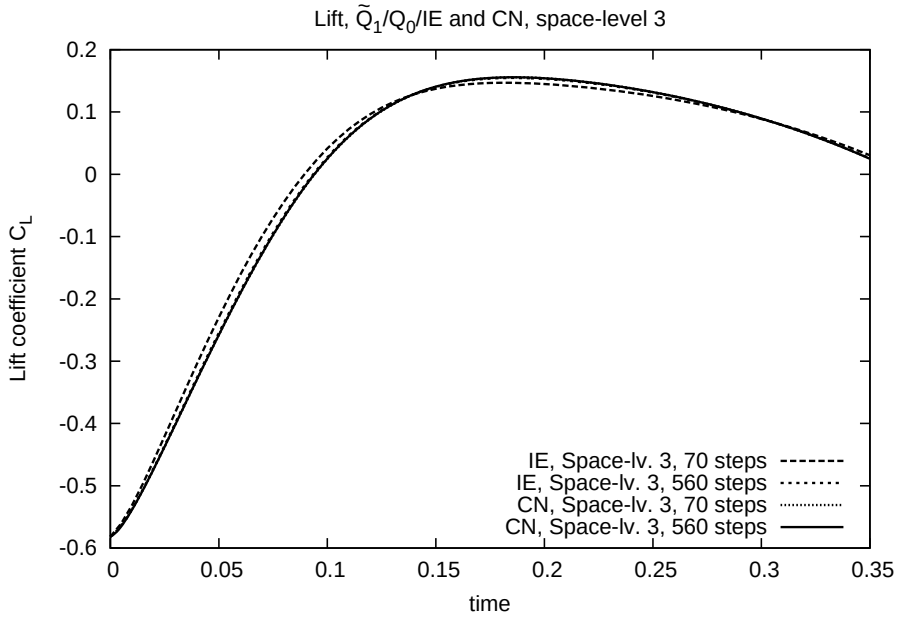

Figure 8.20: Lift coefficient C_L. Space-level three, time discretisation with the implicit Euler and Crank–Nicolson scheme.

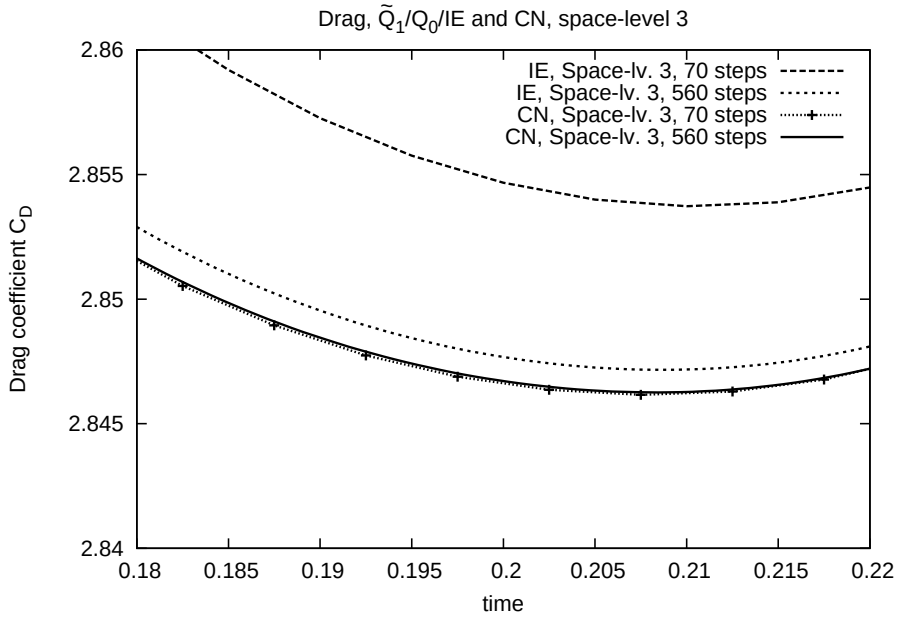

Figure 8.21: Drag coefficient C_D. Space-level three, time discretisation with the implicit Euler and Crank–Nicolson scheme. Zoom to the time interval $[0.18, 0.22]$.

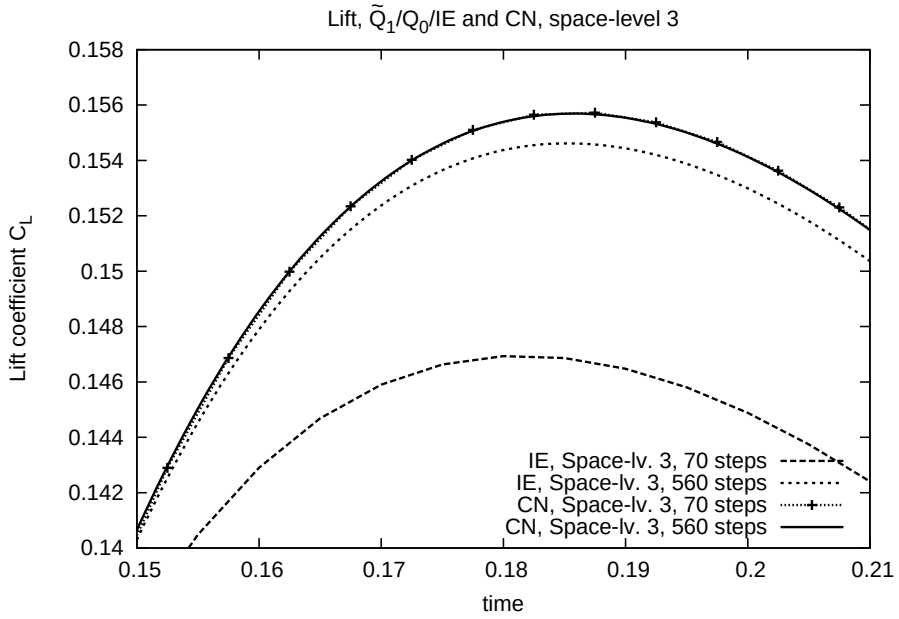

Figure 8.22: Lift coefficient C_L. Space-level three, time discretisation with the implicit Euler and Crank–Nicolson scheme. Zoom to the time interval $[0.15, 0.21]$.

discr.	sp.lv.	#int.	$t(C_D^{\min})$	$C_D^{\min}$	$t(C_L^{\max})$	$C_L^{\max}$
$\tilde{Q}_1/Q_0$/IE	3	140	0.2100000	2.849933	0.1850000	0.1513358
	3	560	0.2087500	2.847162	0.1850000	0.1546166
$\tilde{Q}_1/Q_0$/CN	3	70	0.2075000	2.846160	0.1875000	0.1557214
	3	140	0.2087500	2.846234	0.1862500	0.1557038
	3	560	0.2084375	2.846255	0.1859375	0.1556929
$\tilde{Q}_1/Q_0$/IE	4	280	0.1800000	2.725510	0.1725000	0.2147027
	4	560	0.1775000	2.724760	0.1731250	0.2167395
$\tilde{Q}_1/Q_0$/CN	4	70	0.1725000	2.723840	0.1725000	0.2187618
	4	280	0.1743750	2.723973	0.1743750	0.2187658
	4	560	0.1740625	2.723980	0.1740625	0.2187661
Q_2/P_1^{disc}/CN	4	70	0.1625000	2.515065	0.1725000	0.2432457
	4	560	0.1621875	2.515169	0.1734375	0.2432525

Table 8.9: Minimum drag and maximum lift coefficients for different discretisations.

8.3.3. Influence of the space discretisation

As explained in Chapter 6, the space and time discretisation are coupled in the optimal control case. Figures 8.23 and 8.24 give an overview about the influence of the space discretisation in the whole optimisation process. Figure 8.25 highlights the lift coefficients in a smaller time interval. The space discretisation in these tests is carried out with the $\tilde{Q}_1/Q_0$ and the Q_2/P_1^{disc} finite element pair on different levels of refinement. For the the time discretisation, the strongest available configuration is used, which is the Crank–Nicolson scheme with 560 intervals. Thus, it is assumed that the error in time is insignificant.

For increasing space level, the drag and lift coefficients using the $\tilde{Q}_1/Q_0$ element pair converge towards the reference computed with Q_2/P_1^{disc}, but for space-level four, a considerable discrepancy can still be seen. The Q_2/P_1^{disc} discretisation shows a much higher accuracy. The drag and lift plots on level two are visually much closer to the reference than those computed with $\tilde{Q}_1/Q_0$ on level four. This behaviour is also confirmed by Table 8.10 which lists the minimum drag coefficient, maximum lift coefficient and the time where these values are taken.

The computed drag and lift extrema are much closer to the reference for the Q_2/P_1^{disc} discretisation; roughly estimated, the Q_2/P_1^{disc} is about two to three space levels more exact than $\tilde{Q}_1/Q_0$ in this example. In combination with the experiences from the previous section, which states that it is enough to use only 70 timesteps, it can be concluded that a discretisation with Q_2/P_1^{disc}/CN on level 2 with 70 timesteps gives even better results

discr.	sp.lv.	#int.	$t(C_D^{\min})$	$C_D^{\min}$	$t(C_L^{\max})$	$C_L^{\max}$
$\tilde{Q}_1/Q_0$/CN	3	560	0.2084375	2.84625	1.859375	0.15569
$\tilde{Q}_1/Q_0$/CN	4	560	0.1740625	2.72398	1.740625	0.21876
Q_2/P_1^{disc}/CN	2	560	0.1846875	2.41392	1.746875	0.23401
Q_2/P_1^{disc}/CN	3	560	0.1721875	2.46917	1.734375	0.24007
Q_2/P_1^{disc}/CN	4	560	0.1621875	2.51516	1.734375	0.24325

Table 8.10: Minimum drag and maximum lift coefficients for different space discretisations.

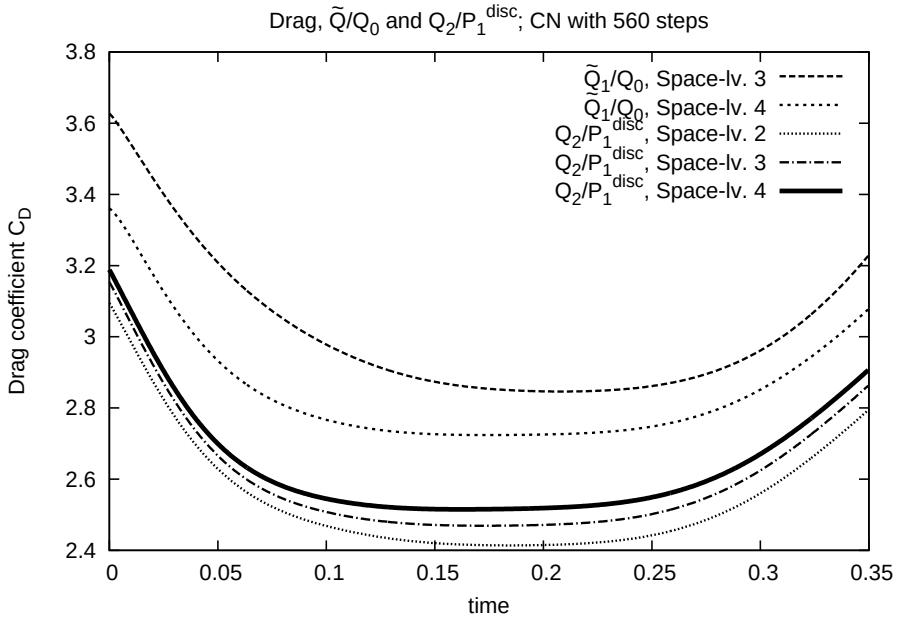

Figure 8.23: Drag coefficient C_D. Different space levels and space discretisations.

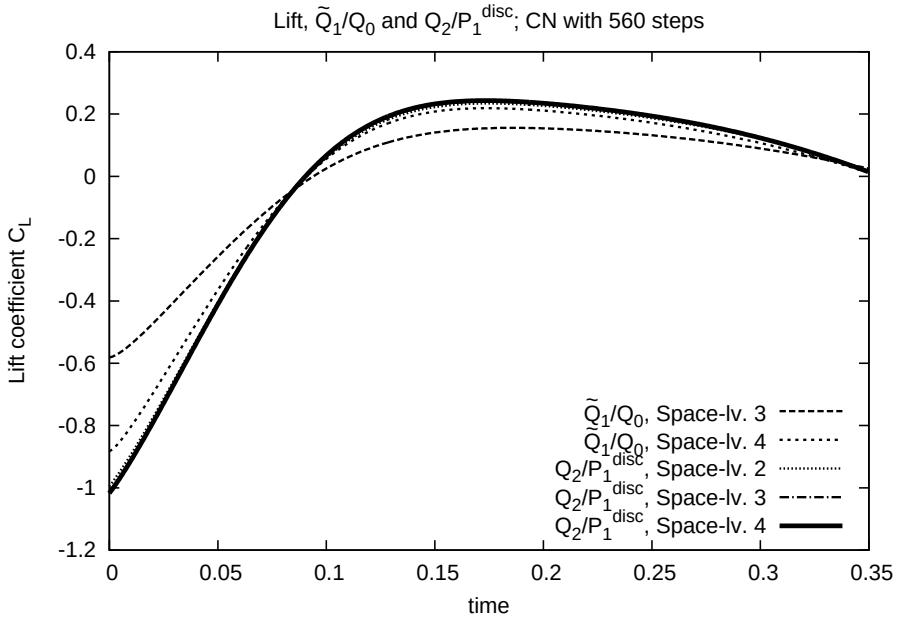

Figure 8.24: Lift coefficient C_L. Different space levels and space discretisations.

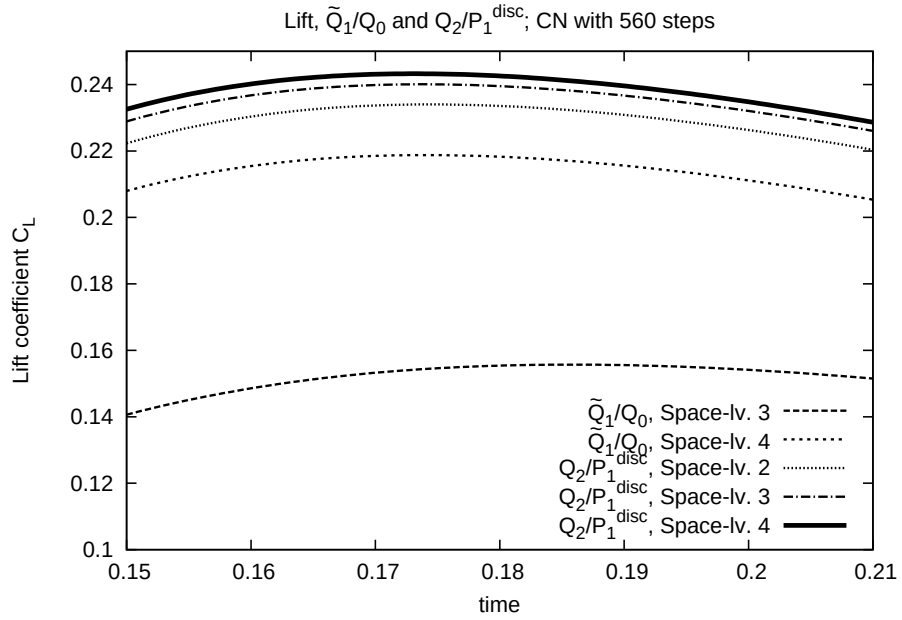

Figure 8.25: Lift coefficient C_L. Different space levels and space discretisations. Zoom to the time interval $[0.15, 0.21]$.

than a discretisation with $\tilde{Q}_1/Q_0$/IE on level four to five with 560 timesteps — at least in this example. For the overall CPU time, this means a reduction by approx. 95 percent, see Section 8.4.

8.3.4. Semi-explicit time discretisation

In Section 4.4 on page 97, the semi-explicit time discretisation has been introduced. The main property of this scheme is that the local problems in each time interval are linear. The following analysis gives a rough overview about the influence of this practice to the quality of the solution.

Similar to previous tests, the drag and lift coefficients are measured along the time axis. The space discretisation is fixed to refinement level three, using $\tilde{Q}_1/Q_0$. THe Figures 8.26 and 8.27 compare the Crank–Nicolson time discretisation (abbreviated again by 'CN') with the implicit Euler scheme (abbreviated by 'IE') and the semi-explicit Euler scheme (abbreviated by 'SEE') for 560 intervals. Figure 8.28 and 8.29 zoom in to the time intervals $[0.18, 0.22]$ and $[0.15, 0.21]$, respectively.

The curve corresponding to the Crank–Nicolson scheme can be seen as the reference (black solid line). Both schemes, the implicit as well as the semi-explicit Euler scheme, give an acceptable approximation. A closer comparisons reveals that the implicit Euler scheme 'smears' the solution more than the Crank–Nicolson scheme: The minimum drag values are larger, the maximum lift values smaller than the reference. For the semi-explicit Euler scheme, the situation is opposite, the scheme 'amplifies' the solution: The minimum drag value is smaller and the maximum lift value larger than the reference. However, the difference is small, in particular in case of the drag. Relative to the values computed with the Crank–Nicolson scheme, the difference in the minimum drag coefficient between the two Euler schemes is less than 0.1 percent and the difference in the maximum lift coefficient is about two percent.

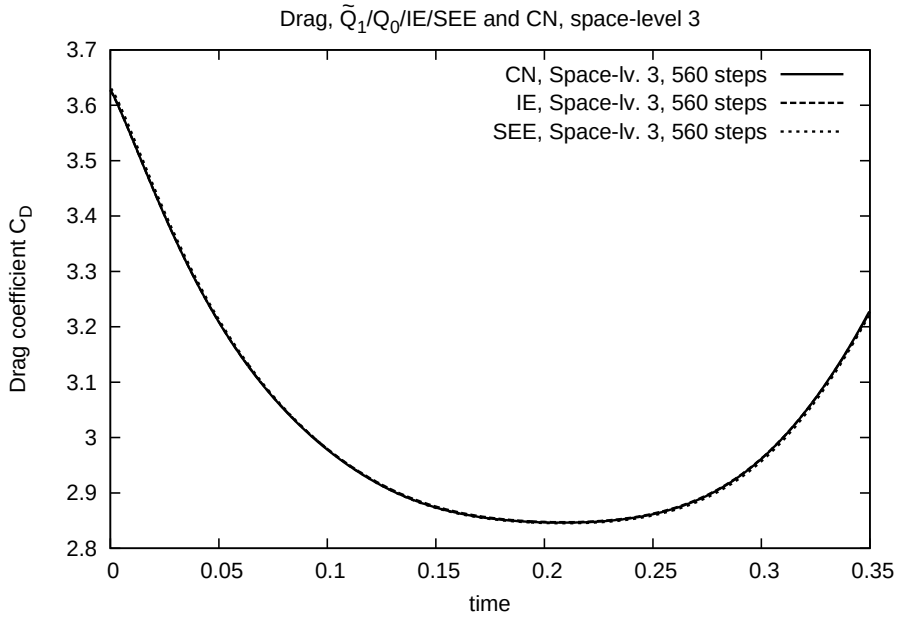

Figure 8.26: Drag coefficient C_D. Space-level three, time discretisation with the implicit/semi explicit Euler and Crank–Nicolson scheme.

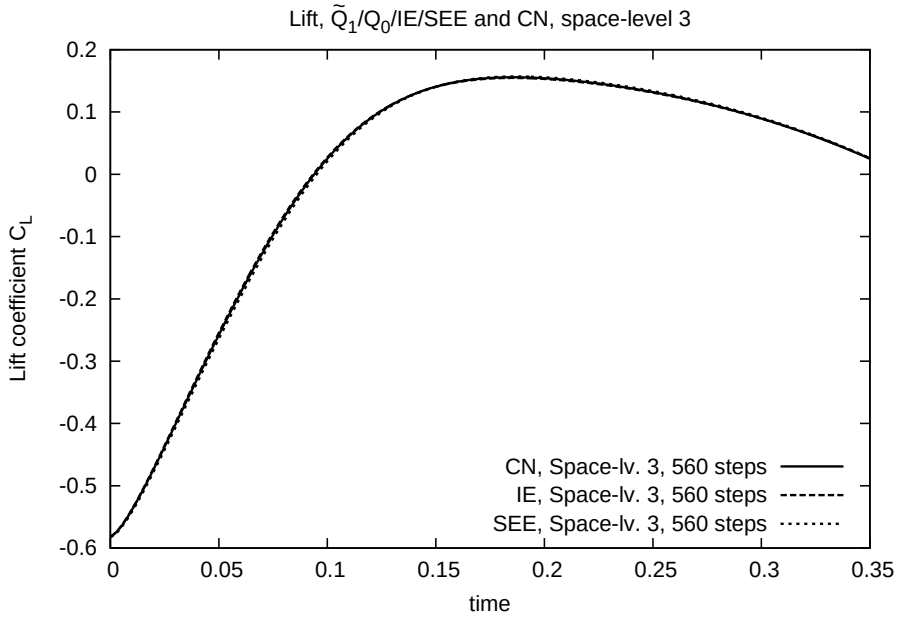

Figure 8.27: Lift coefficient C_L. Space-level three, time discretisation with the implicit/semi explicit Euler and Crank–Nicolson scheme.

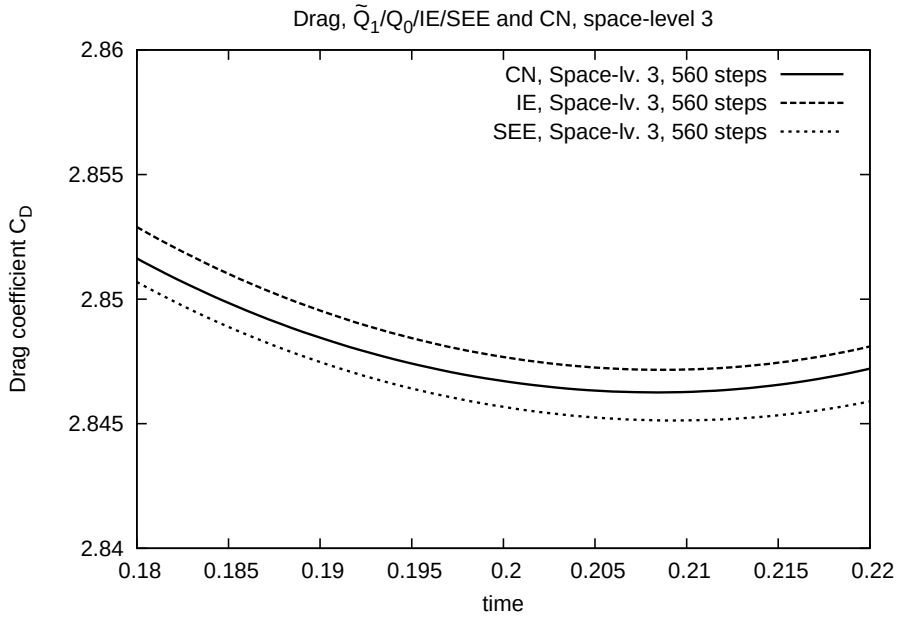

Figure 8.28: Drag coefficient C_D. Space-level three, time discretisation with the implicit/semi explicit Euler and Crank–Nicolson scheme. Zoom to the time interval $[0.18, 0.22]$.

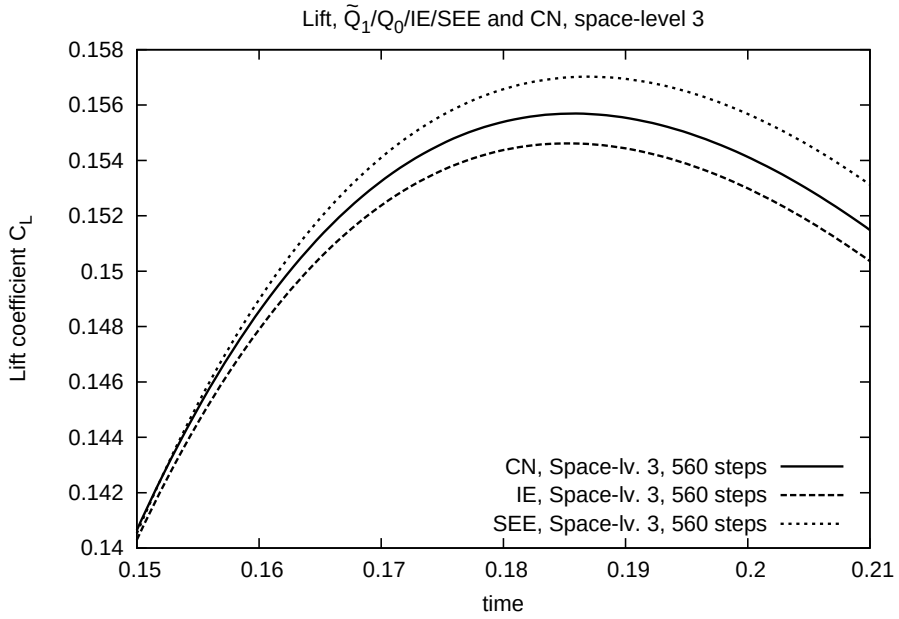

Figure 8.29: Lift coefficient C_L. Space-level three, time discretisation with the implicit/semi explicit Euler and Crank–Nicolson scheme. Zoom to the time interval $[0.15, 0.21]$.

Table 8.11 contains the results in quantitative form. All in all, the semi-explicit Euler scheme approximates the solution with an accuracy similar to the implicit Euler scheme for the same number of time intervals — if the time discretisation is fine enough. This behaviour is expected, since both schemes are of first order and designed in a similar way. However, the above analysis does not compare solutions for less than 560 time intervals. The reason is that it was not possible: If a discretisation with less than 560 timesteps based on the semi-explicit Euler scheme, the solver does not converge in less than 15 iterations or even diverges. This is due to the CFL condition; a more detailed overview will be given in the next section.

discr.	sp.lv.	#int.	$t(C_D^{\min})$	$C_D^{\min}$	$t(C_L^{\max})$	$C_L^{\max}$
$\tilde{Q}_1/Q_0$/IE	3	560	0.2087500	2.84716	0.1850000	0.15461
$\tilde{Q}_1/Q_0$/SEE	3	560	0.2093750	2.84512	0.1868750	0.15702
$\tilde{Q}_1/Q_0$/CN	3	560	0.2084375	2.84625	0.1859375	0.15569
$\tilde{Q}_1/Q_0$/IE	4	560	0.1775000	2.72476	0.1731250	0.21673
$\tilde{Q}_1/Q_0$/SEE	4	560	0.1731250	2.72264	0.1756250	0.22126
$\tilde{Q}_1/Q_0$/CN	4	560	0.1740625	2.72398	0.1740625	0.21876

Table 8.11: Minimum drag and maximum lift coefficients for different space discretisations. Implicit and semi-explicit Euler scheme.

8.4. A solver discussion

Building on top of the background of Chapter 7, the following analysis documents the solver behaviour for Example 8.2. While in Chapter 7, all test examples were of stationary nature, the convergence results documented here suggest that the proposed solver methodology can also successfully be applied to fully transient flow problems. Finally, a comparison between optimisation and simulation gives insight into the relationship between the CPU time of the KKT solver and the CPU time of a simulation solver in more practical situations.

Consider Example 8.2 discretised with either $\tilde{Q}_1/Q_0$ or Q_2/P_1^{disc} in space and implicit/semi-explicit Euler and Crank–Nicolson scheme in time, respectively. The regularisation parameters are set to $\alpha = 0.02$ and $\gamma = 0.0$. A hierarchy of space-time fine grids is defined according to Table 8.12.

Level	sp.lv.	#NEL	#int.	k
1	2	520	70	1/200
2	3	2 080	140	1/400
3	4	8 320	280	1/800

Table 8.12: Problem size of different fine grid meshes in space and time. 'sp.lv.' denotes the space-level, '#int' the number of timesteps.

From each fine grid, a corresponding multigrid hierarchy is obtained either by simultaneous coarsening in space and time ('full space-time multigrid') or by semi-coarsening in time ('pure time multigrid') until a coarse time mesh with 70 intervals is reached. The underlying Newton solver reduces the norm of the nonlinear residual by eight digits, $\varepsilon_{\text{OptNL}} = 10^{-8}$. The inner multigrid solver reduces the norm of the residual by two digits, $\varepsilon_{\text{OptMG}} = 10^{-2}$, using a BiCGStab(FBSimPrec,NSM=4) smoother and a V-cycle.

The implicit Euler and Crank–Nicolson schemes Tables 8.13 and 8.14 document the convergence properties of the solver for the different mesh levels, once using a full space-time multigrid for the linear subsystems, once a pure time multigrid. Similar to the results in Chapter 7, the solver converges in very few steps and independent of the level. The pure time-multigrid preconditioner needs less iterations, thus being more stable, but still needs more CPU time due to the larger coarse grid problems.

Discretisation with $\tilde{Q}_1/Q_0$/IE

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	7	18	5 879	303	19.4
3	4	280	6	15	39 193	1 571	24.9

Discretisation with $\tilde{Q}_1/Q_0$/CN

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	7	17	6 849	283	24.2
3	4	280	7	17	50 505	1 324	38.1

Discretisation with Q_2/P_1^{disc}/CN

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	7	12	21 628	5 034	4.2
3	4	280	7	13	200 580	13 655	14.6

Table 8.13: Nonstationary 'Flow–Around–Cylinder' control. Convergence statistics of the nonlinear and linear solver for different space and time discretisations on different mesh levels. Number of nonlinear ('#NL') and total number of linear ('Σ#MG') space-time iterations as well as CPU time. Full space-time multigrid. Comparison between optimisation and simulation.

The CPU time for solving the KKT system ('T_{opt}') increases by a factor of ≈ 8 per level which is in line with the increase of the problem size. The table also compares the CPU time for the optimisation with the CPU time of a 'corresponding' simulation ('T_{sim}'). This has been carried out similar to the previous examples: The control computed in the optimisation is used as right-hand side in a simulation that starts with the same initial condition on the same level with the same space and time discretisation. The simulation uses a Newton solver in each timestep that reduces the norm of the initial residual by eight digits, $\varepsilon_{\text{OptNL}} = 10^{-8}$. The Newton solver is preconditioned by a multigrid solver in space that reduces the norm of the residual by two digits per nonlinear iteration, using a BiCGStab(PSCSmootherDiag,NSM=4) smoother in space. A factor of $C \approx 10$–25 between the optimisation and the simulation can be seen, which is similar to the results of the 'Driven–Cavity' example in Section 7.2.3 on page 142ff. It is noted that T_{sim} grows by a factor < 8 per level which is most likely a consequence of cache effects for small problem sizes on modern computer architectures, cf. [148, 150]. A factor of ≈ 8 is expected for larger problems which approximately leads to a factor of $C \approx 20$–50 between the optimisation and the simulation.

8.3 Remark. One may recognise the extraordinary high CPU time marked with '(*)' in Table 8.14. In the pure time-multigrid case, the spatial problems turn out to be harder on finer levels. As a consequence, the spatial preconditioner breaks down for many subproblems. The solver automatically recalculates these subproblems with a multigrid solver that applies a PSCSmootherFull based smoother. This 'fallback strategy' is more ro-

Discretisation with $\tilde{Q}_1/Q_0$/IE

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	7	12	5 951	303	19.6
3	4	280	6	11	55 194	1 571	35.1

Discretisation with $\tilde{Q}_1/Q_0$/CN

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	7	12	7 848	283	27.7
3	4	280	6	10	56 644	1 324	42.7

Discretisation with Q_2/P_1^{disc}/CN

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	6	8	46 959	5 034	9.3
3	4	280	6	8	(*) 827 217	13 655	60.5

Table 8.14: Nonstationary 'Flow–Around–Cylinder' control. Comparison between optimisation and simulation similar to Table 8.13. Pure time multigrid.

bust but also computationally much more expensive, especially if many subproblems break down. Thus, the CPU time is much higher, although the number of nonlinear and linear iterations is the same on all refinement levels.

The adaptive Newton Table 8.15 demonstrates the efficiency of the adaptive Newton in comparison to the standard Newton algorithm. The column 'Coars.' denotes the coarsening strategy, which is either the full space-time coarsening or the pure time coarsening. The test restricts to a discretisation in space with the $\tilde{Q}_1/Q_0$ finite element pair and applies the implicit Euler as well as the Crank–Nicolson scheme. Similar to the previous chapter, the number of nonlinear and linear iterations obtained with the two types of solvers are rather close to each other or even the same, thus the adaptive Newton is a good alternative to the standard Newton algorithm with manually chosen, semi-optimal stopping criterion for the linear solver. It is noted that the CPU time for the adaptive Newton is again a bit higher due to the adaptive choice of the stopping criterion in the coarse grid solver of the multigrid preconditioner.

The semi-explicit Euler scheme Table 8.16 documents the convergence results of the solver if the semi-explicit Euler time discretisation scheme is used. The abbreviation 'not conv.' marks stagnation in the nonlinear iteration and entries containing 'div' could not be computed due to divergence. If the solver converges, the results are similar to the implicit Euler case. Nevertheless, the solver is less stable than if the implicit Euler scheme is used: For the standard mesh levels 3 and 4 in space with 140 and 280 intervals, respectively, as they are used for the implicit Euler scheme, the solver does not converge. If the number of intervals is larger, the convergence behaviour of the solver is better, but still, the solver fails to converge in many situations: Using pure time multigrid for the preconditioning of the linear system, the linear solver breaks down. This behaviour is in line with the expectations from Chapter 4.4 and is the results of a CFL condition which needs to be satisfied for not fully-implicit schemes. It is remarked that also a stronger space-time smoother does not necessarily help. Although not documented in a separate table, the

Discretisation with $\tilde{Q}_1/Q_0$/IE

				standard Newton			adaptive Newton		
Coars.	lv.	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	#NL	Σ#MG	T_{opt}
full	2	3	140	7	18	5 879	7	19	6 218
space-time	3	4	280	6	15	39 193	6	18	46 517
pure	2	3	140	7	12	5 951	7	12	6 870
time	3	4	280	6	11	55 194	6	11	59 322

Discretisation with $\tilde{Q}_1/Q_0$/CN

				standard Newton			adaptive Newton		
Coars.	lv.	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	#NL	Σ#MG	T_{opt}
full	2	3	140	7	17	6 849	7	19	7 830
space-time	3	4	280	7	17	50 505	6	16	48 128
pure	2	3	140	7	12	7 848	7	11	8 374
time	3	4	280	6	10	56 644	6	10	63 909

Table 8.15: Nonstationary 'Flow–Around–Cylinder' control. Convergence statistics for the standard Newton and the adaptive Newton.

best proposed smoother from this work has the same 'divergence behaviour' as the BiCGStab(FBSimPrec,...) smoother: BiCGStab(FBGsPrec,...) is also not strong enough, the solver does not converge for 140 or 280 time intervals on space level three and four, respectively.

Tables 8.17 and 8.18 compare the computational time in the implicit Euler and Crank–Nicolson cases to those of the semi-explicit Euler case in Table 8.16. Using the semi-explicit Euler scheme, the solver is up to 30 percent faster than using the implicit Euler or Crank–Nicolson scheme for the same level and the same number of intervals due to less numerical work in each interval. In the Crank–Nicolson case on the other hand, the discretisation needs only 70 to 140 intervals to have an even higher accuracy than if the implicit/semi-explicit Euler scheme with 560 intervals is used, see Section 8.3.4. Hence, in comparison to the Crank–Nicolson scheme, the semi-explicit Euler scheme does not pay off.

In summary, the implicit discretisations should be preferred over the semi-explicit discretisation. On the same space-time mesh, the semi-explicit Euler scheme is indeed faster than the implicit Euler scheme and provides the same accuracy. However, the implicit Euler as well as the Crank–Nicolson method are much more robust, the choice of the number of time intervals influences the stability of the underlying solver to a much smaller extent. Using the Crank–Nicolson method finally allows to reduce the number of time intervals without considerable loss in accuracy to such an extent that there is no benefit in using the semi-explicit method — at least in the examples examined here.

Numerical effort with respect to accuracy In a final comparison, the numerical effort for solving the underlying KKT systems is analysed with respect to the accuracy of the discretisation. Table 8.19 illustrates the CPU time of the solver for different refinement levels and different space-time discretisations in relation to the relative error in the minimum drag and maximum lift coefficients: As a reference, C_D^{ref} and C_L^{ref} denote the minimum drag and maximum lift coefficient computed on space-level four with 560 time intervals using the Q_2/P_1^{disc} finite element pair in space and the Crank–Nicolson scheme in time for

		space-time multigrid			pure time multigrid		
sp.lv.	#int.	#NL	Σ#MG	T_{opt}	#NL	Σ#MG	T_{opt}
3	140	div	div	div	not conv.	not conv.	not conv.
4	140	div	div	div	not conv.	not conv.	not conv.
3	280	div	div	div	7	25	18 988
4	280	div	div	div	not conv.	not conv.	not conv.
3	560	7	18	14 243	7	15	20 945
4	560	6	19	60 974	div	div	div

Table 8.16: Nonstationary 'Flow–Around–Cylinder' control. Convergence results for the semi-explicit Euler scheme. Full space-time multigrid as well as pure time multigrid preconditioner.

		space-time multigrid			pure time multigrid		
sp.lv.	#int.	#NL	Σ#MG	T_{opt}	#NL	Σ#MG	T_{opt}
3	140	7	18	5 879	7	12	5 951
4	280	6	15	39 193	6	11	55 194
3	560	7	18	20 515	7	11	21 565
4	560	7	16	75 024	7	16	137 870

Table 8.17: Nonstationary 'Flow–Around–Cylinder' control. Convergence results for the implicit Euler scheme. Full space-time multigrid as well as pure time multigrid preconditioner.

		space-time multigrid			pure time multigrid		
sp.lv.	#int.	#NL	Σ#MG	T_{opt}	#NL	Σ#MG	T_{opt}
3	140	7	17	6 849	7	12	7 848
4	280	7	17	50 505	6	10	56 644
3	560	7	17	24 578	7	7	17 469
4	560	6	13	74 655	6	8	84 611
4	140	6	15	26 793	7	14	45 098

Table 8.18: Nonstationary 'Flow–Around–Cylinder' control. Convergence results for the Crank–Nicolson time discretisation. Full space-time multigrid as well as pure time multigrid preconditioner.

Discretisation with $\tilde{Q}_1/Q_0$/IE

sp.lv.	#int.	$C_D^{\min}$	$C_L^{\max}$	$\mathrm{err}(C_D^{\min})$	$\mathrm{err}(C_L^{\max})$	#NL	Σ#MG	T_{opt}
3	140	2.8499	0.15133	0.1331	0.3778	7	18	5 879
3	560	2.8471	0.15461	0.1320	0.3643	7	18	20 515
4	560	2.7247	0.21673	0.0833	0.1090	7	16	75 024

Discretisation with $\tilde{Q}_1/Q_0$/CN

sp.lv.	#int.	$C_D^{\min}$	$C_L^{\max}$	$\mathrm{err}(C_D^{\min})$	$\mathrm{err}(C_L^{\max})$	#NL	Σ#MG	T_{opt}
3	140	2.8462	0.15570	0.1316	0.3599	7	17	6 849
3	560	2.8462	0.15569	0.1316	0.3599	7	17	24 578
4	560	2.7239	0.21876	0.0830	0.1006	6	13	74 655
3	70	2.8461	0.15572	0.1316	0.3598	10	(**) 92	3 410
4	70	2.7238	0.21877	0.0829	0.1006	13	(**) 149	24 548

Discretisation with Q_2/P_1^{disc}/CN

sp.lv.	#int.	$C_D^{\min}$	$C_L^{\max}$	$\mathrm{err}(C_D^{\min})$	$\mathrm{err}(C_L^{\max})$	#NL	Σ#MG	T_{opt}
2	70	2.4138	0.23394	0.0402	0.0382	8	(**) 85	2 964
2	560	2.4139	0.23401	0.0402	0.0379	9	55	80 804
4	560	2.5151	0.24325	reference		7	13	385 370

Table 8.19: CPU time comparison for different discretisations with respect to accuracy. The results marked with '(**)' have been computed with a one-level solver.

the discretisation (bottommost row in the table, see also Tables 8.8 to 8.10 on page 163ff). The relative errors are defined by

$$\mathrm{err}(C_D^{\min}) := \frac{|C_D^{\min} - C_D^{\mathrm{ref}}|}{C_D^{\mathrm{ref}}}, \qquad \mathrm{err}(C_L^{\max}) := \frac{|C_L^{\max} - C_L^{\mathrm{ref}}|}{C_L^{\mathrm{ref}}}. \tag{8.3}$$

Comparing the CPU time with respect to the relative error, in this example, more than 95 percent of the CPU time can be saved by a higher order discretisation. In relation to the reference solution, a discretisation with the Q_2/P_1^{disc} element pair in space and the Crank–Nicolson discretisation in time shows best relative errors, even on space level two with only 70 time intervals. (This is the coarsest time mesh at all, so a one-level solver was used for solving). With $\mathrm{err}(\cdot) \approx 0.04$, the errors are even much better than those computed with the $\tilde{Q}_1/Q_0$ element pair on level four with 560 time intervals ($\mathrm{relerr}(\cdot) \approx 0.10$), and the computation took only five percent of the CPU time.

On the other hand, if the space discretisation remains unchanged, a higher order time discretisation saves about 60–85 percent of the CPU time in this example. This can be seen, e. g., for a space discretisation with the $\tilde{Q}_1/Q_0$ finite element pair on a fixed spatial level. Using the Crank–Nicolson time discretisation with 70 time intervals, the relative errors are smaller than if the implicit Euler time discretisation with 560 time intervals is used. On space level four, this results in a decrease of CPU time from $\approx 20\,500$ seconds to $\approx 3\,500$ seconds without loss in the accuracy.

Of course, the above results do not hold for general problems, i. e., the problem must be smooth enough such that it makes sense to apply higher order schemes. However, for smooth problems, this example shows the potential of a proper discretisation scheme in terms of savings in CPU time in the optimal control context.

8.5. Appendix: About stabilisation in optimal control problems for fluid flow

One crucial point in many simulations is the part of the stabilisation. Whenever a numerical system turns out to be to hard or unstable, the mathematical formulation and/or discretisation of the underlying PDE is changed by introducing different kinds of stabilisation terms. Such stabilised formulations are properly defined if they simplify the solution process while keeping the disturbance of the solution as small as possible.

The CFD examples in the previous chapters have been computed without stabilisation. However, for higher Reynolds numbers, a proper stabilisation is crucial, as a pure Galerkin discretisation for convection-dominated flows would introduce numerical oscillations (cf. [97, 98, 100]). Many common stabilisation terms like Streamline Diffusion (see for example [142]) work in a nonlinear way, based on local Reynolds numbers or similar. In the optimal control framework, this introduces a quite annoying effect:

Stabilisation in general The general form of a KKT system (2.10) on page 25 can be written in short as a combination of a primal and dual equation,

$$\begin{aligned} \mathcal{H}(x)x &= \mathcal{B}\psi, \\ D\mathcal{H}^*(x)\psi &= \tilde{\mathcal{B}}x. \end{aligned}$$

During the space-time discretisation, all operators are replaced by their discrete counterparts, which results in a system of the form

$$\begin{aligned} \mathcal{H}^\sigma(x^\sigma)x^\sigma &= \mathcal{B}^\sigma\psi^\sigma, \\ D\mathcal{H}^{\sigma,*}(x^\sigma)\psi^\sigma &= \tilde{\mathcal{B}}^\sigma x^\sigma, \end{aligned}$$

for a space-time solution $x^\sigma \in W^{NLMAX}$ where the superscript $\sigma = (h, k)$ denotes discretisation in space and time, h the discretisation in space and k the discretisation in time. By $x^\sigma \mapsto s(x^\sigma)$, a general additive stabilisation term is denoted which is added to the primal equation. The corresponding KKT system changes to

$$\begin{aligned} \mathcal{H}^\sigma(x^\sigma)x^\sigma + s(x^\sigma) &= \mathcal{B}^\sigma\psi^\sigma \\ \big(D\mathcal{H}^{\sigma,*}(x^\sigma) + (Ds)^*(x^\sigma)\big)\psi^\sigma &= \tilde{\mathcal{B}}^\sigma x^\sigma \end{aligned}$$

with $(Ds)^*(x^\sigma)$ describing the adjoint of the Fréchet derivative of $x^\sigma \mapsto s(x^\sigma)$. At this point, the implementation becomes tedious: For a general nonlinear, locally defined operator $s(\cdot)$ like the Streamline Diffusion operator, the dual $(Ds)^*$ of its Fréchet derivative usually has a complicated form. For an overview about the effect of standard stabilisation techniques in optimal control problems, the interested reader is referred to [2, 40].

Symmetric stabilisation A remedy to this situation is the use of linear symmetric stabilisation techniques. One special representative which gives good results to the author's experience if applied to the Navier–Stokes equations is the 'edge-oriented jump stabilisation' technique described, e. g., in [124]. This stabilisation is defined purely in space. The underlying finite element space of the discretisation is in the following denoted by V_h. For an arbitrary $v \in V_h$, the action Sv of this stabilisation is defined by

$$(Sv, w) = \sum_{\text{edge E}} \max(\tilde{\kappa}\nu|E|, \kappa|E|^2) \int_E [\nabla v][\nabla w]\, ds \tag{8.4}$$

for all $w \in V_h$. The constants κ, $\tilde{\kappa} \geq 0$ define stabilisation parameters, ν the viscosity, $|E|$ the length of the edge E, and $[\cdot]$ the jump of a function over an edge. The above stabilisation $s(x^\sigma)$ is defined by applying S to every velocity component in every timestep. Setting $\tilde{\kappa} := 0$, an even simpler form of the edge-oriented stabilisation is obtained,

$$(Sv, w) = \sum_{\text{edge E}} \kappa |E|^2 \int_E [\nabla v][\nabla w]\, ds, \tag{8.5}$$

which was also used for numerical simulations in [55] and will be used in the following numerical tests.

Important features of this stabilisation are symmetry, linearity and consistency. As a consequence, there is $(Ds)^*(x^\sigma) = s$, independent of x^σ, which simplifies the KKT system to

$$\begin{aligned}(\mathcal{H}^\sigma(x^\sigma) + s)x^\sigma &= \mathcal{B}^\sigma \psi^\sigma \\ (D\mathcal{H}^{\sigma,*}(x^\sigma) + s)\psi^\sigma &= \tilde{\mathcal{B}}^\sigma x^\sigma.\end{aligned}$$

From the implementational point of view, this means adding a symmetric matrix (corresponding to S) to all velocity matrices. The disadvantage of this method is the increased matrix stencil which increases the computational time and memory consumption, cf. [124].

The stabilised benchmark example The following numerical test demonstrates that this type of stabilisation is also effective in the field of the optimal control. Consider the optimal distributed control of the nonstationary Navier–Stokes equations at Re=100, Example 8.2. The space discretisation is carried out with $\tilde{Q}_1/Q_0$ at space-level three and four. For the time discretisation, the Crank–Nicolson scheme is chosen with 140 intervals on space level three and 280 intervals on space-level four, respectively.[4] To measure the quality of the solution, the drag and lift coefficients C_D and C_L are measured as above. Figures 8.30 and 8.31 show the corresponding plots for the full time interval $[0, T]$, Figures 8.32 and 8.33 for a subinterval. The figures illustrate on the one hand the unstabilised results, on the other hand the stabilised counterparts using the edge-oriented jump stabilisation (denoted by 'EOJ(κ)') with a jump stabilisation parameter $\kappa = 0.02$. Finally, for reference purposes, the plots contain the unstabilised solution calculated with $Q_2/P_1^{\text{disc}}/\text{CN}$ on level four with 560 timesteps.

In comparison to the unstabilised case, the use of the stabilisation improves the solution. At space level 3, the plots of the stabilised solutions follow the reference plot, while the unstabilised counterparts show a much larger difference, especially in the drag coefficient C_D. The point in time at which C_D takes its minimum is much later in the unstabilised case than in the stabilised one. For level 4, the use of the edge-oriented jump stabilisation improves the drag and lift plots by roughly 1/4 to 1/2 level.

Figures 8.34 to 8.37 visualise the drag and lift coefficients for a fixed space-time level and different stabilisation parameters. The figures depict the drag and lift values over time for EOJ(0.1), EOJ(0.06) and EOJ(0.02). For reference purposes, the corresponding plots also contain the results in the unstabilised case, for a discretisation with $\tilde{Q}_1/Q_0/\text{CN}$ as well as with $Q_2/P_1^{\text{disc}}/\text{CN}$. Similar to the results for the simulation [124], the jump stabilisation is rather stable for a large range of stabilisation parameters (here $[0.02, 0.1]$). In this example, higher stabilisation parameters around ≈ 0.1 lead to better approximations of the drag and the lift.

[4] From the previous analysis, it can be expected that 70 intervals should already be enough. The refinement strategy chosen here starts with 140 intervals and refines in space and time simultaneously; this is the typical approach to ensure that the global space-time error reduces.

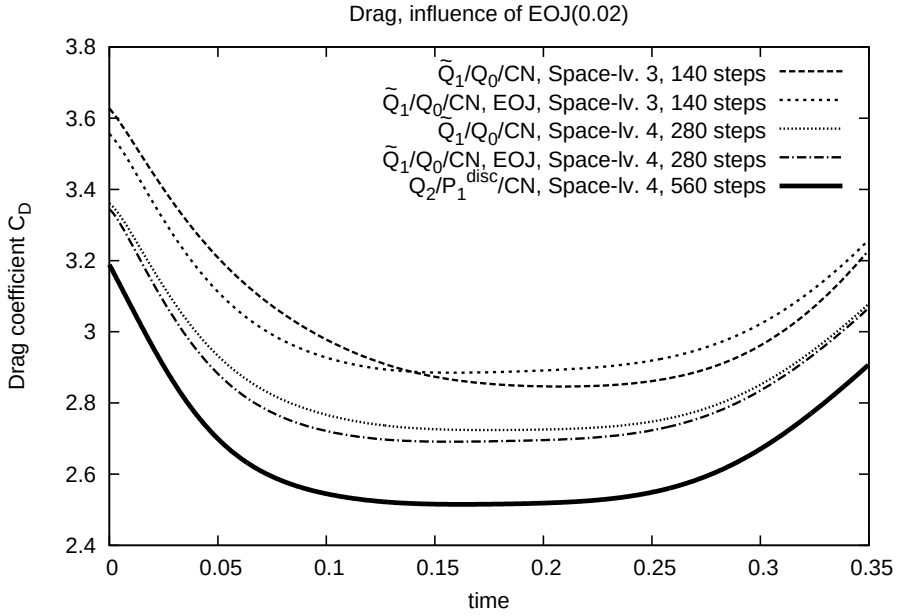

Figure 8.30: Drag coefficient C_D. Influence of the EOJ(0.02) stabilisation.

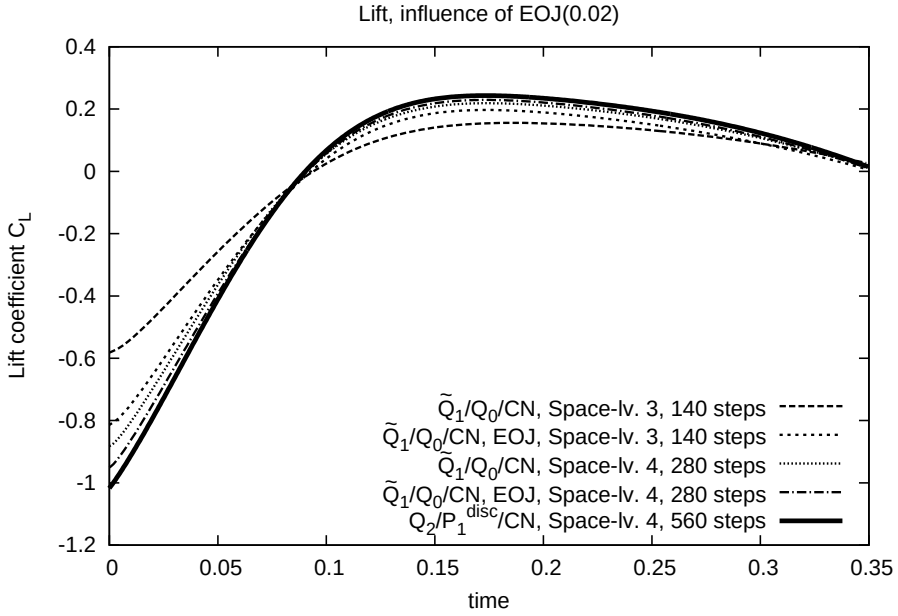

Figure 8.31: Lift coefficient C_L. Influence of the EOJ(0.02) stabilisation.

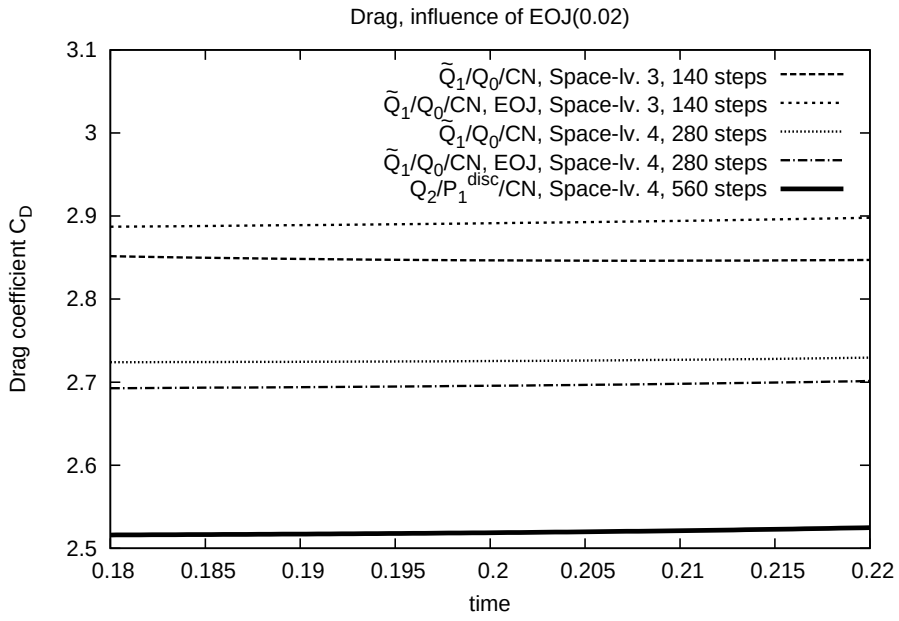

Figure 8.32: Drag coefficient C_D. Influence of the EOJ(0.02) stabilisation. Zoom to the time interval $[0.18, 0.22]$.

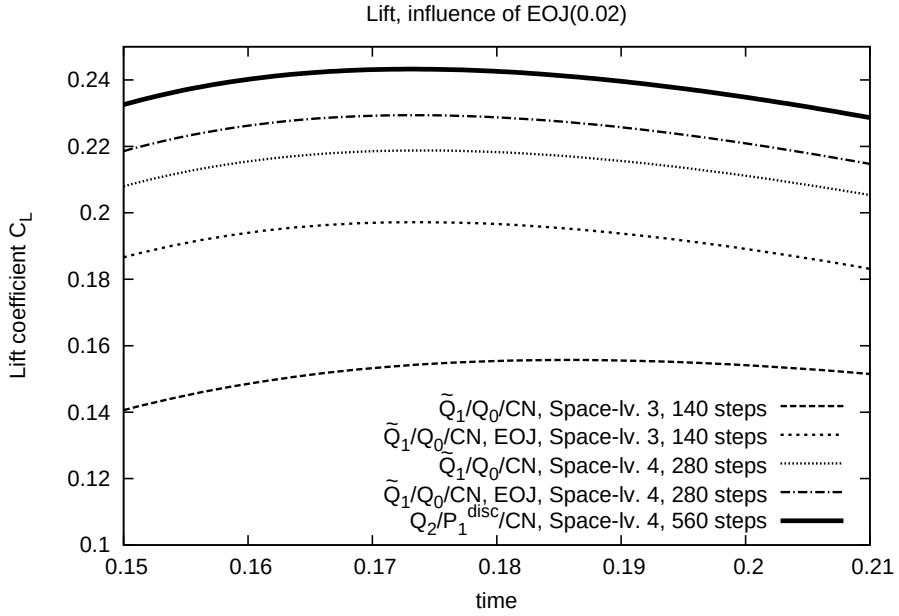

Figure 8.33: Lift coefficient C_L. Influence of the EOJ(0.02) stabilisation. Zoom to the time interval $[0.15, 0.21]$.

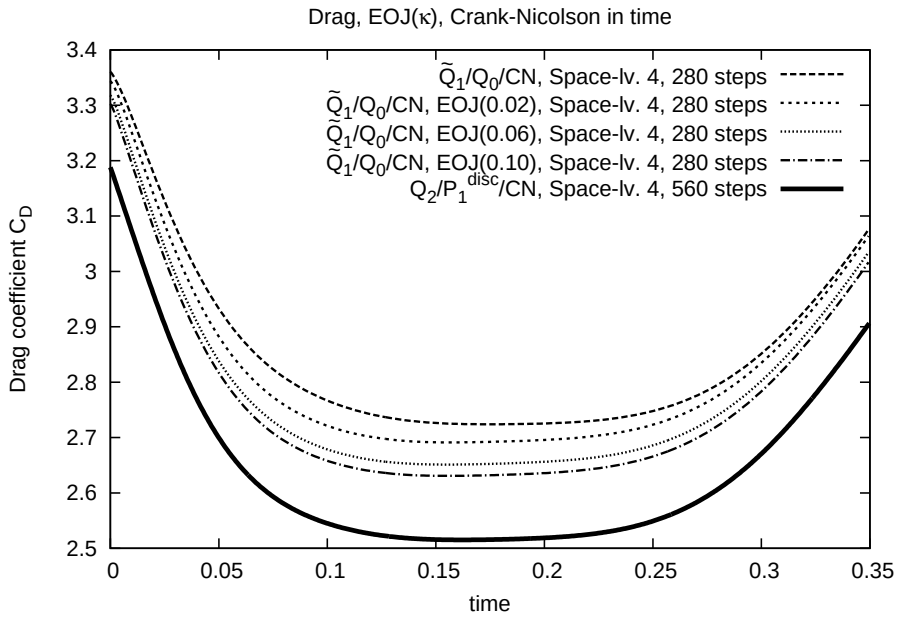

Figure 8.34: Drag coefficient C_D. Influence of the EOJ stabilisation for different stabilisation parameters on a fixed mesh.

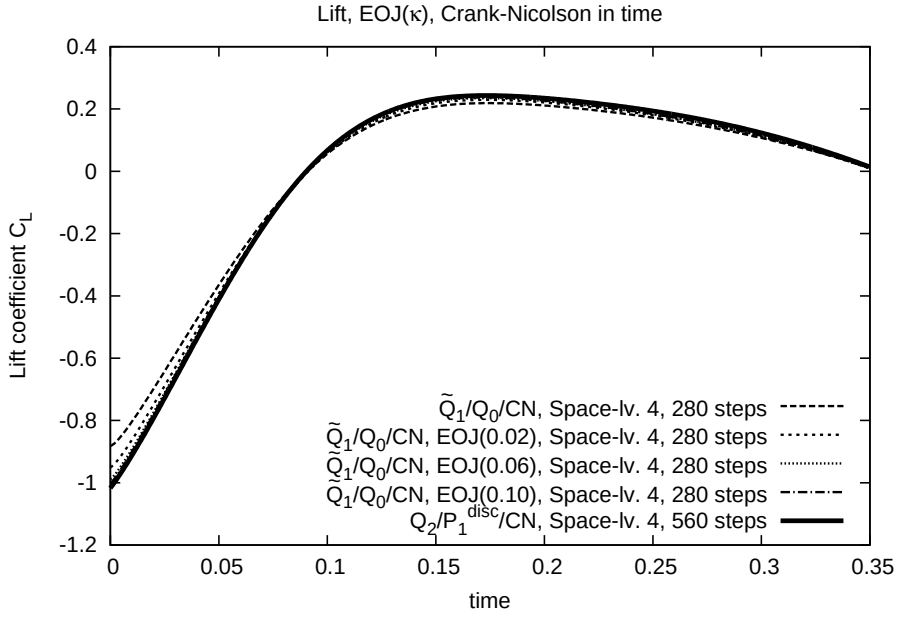

Figure 8.35: Lift coefficient C_D. Influence of the EOJ stabilisation for different stabilisation parameters on a fixed mesh.

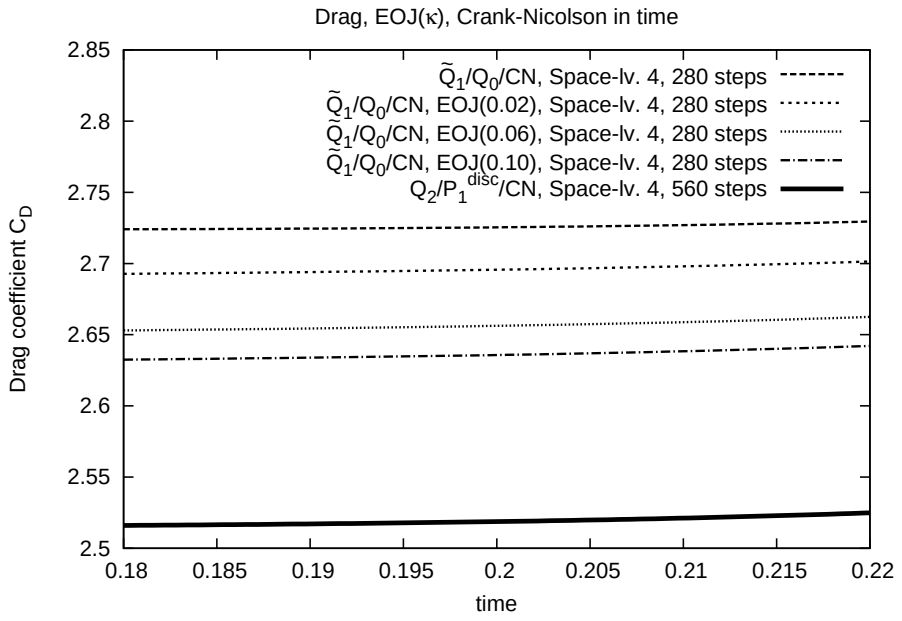

Figure 8.36: Drag coefficient C_D. Influence of the EOJ stabilisation for different stabilisation parameters on a fixed mesh. Zoom to the time interval $[0.15, 0.21]$.

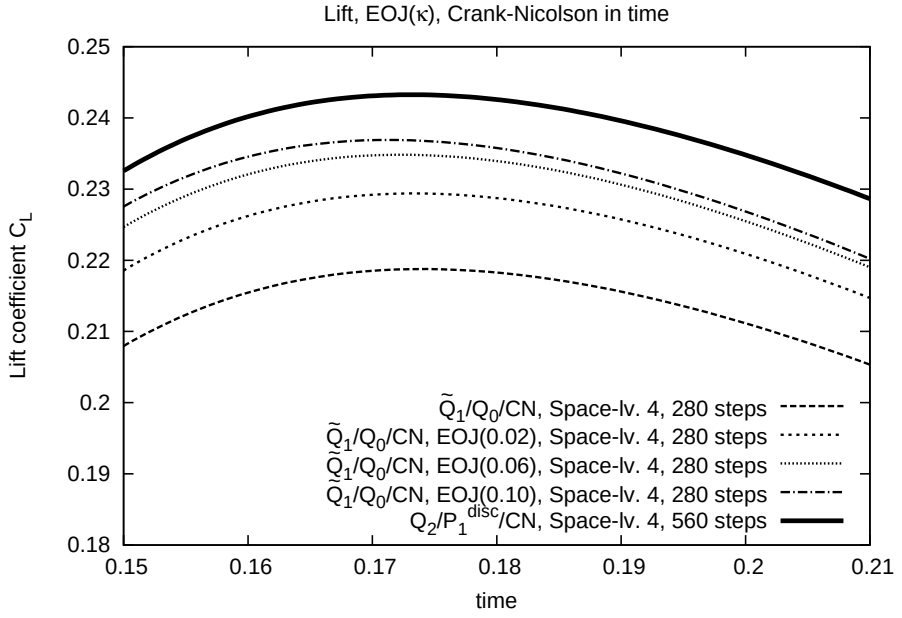

Figure 8.37: Lift coefficient C_D. Influence of the EOJ stabilisation for different stabilisation parameters on a fixed mesh. Zoom to the time interval $[0.15, 0.21]$.

Discretisation with $\tilde{Q}_1/Q_0$/IE, EOJ(0.02)

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	6	11	5 099	497	10.2
3	4	280	6	16	72 597	2 571	28.2

Discretisation with $\tilde{Q}_1/Q_0$/CN, EOJ(0.02)

Level	sp.lv.	#int.	#NL	Σ#MG	T_{opt}	T_{sim}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
2	3	140	6	13	7 246	503	14.4
3	4	280	7	14	75 130	2 499	30.0

Table 8.20: Nonstationary 'Flow–Around–Cylinder' control, stabilised with EOJ(0.02). Convergence statistics of the nonlinear and linear solver for different space and time discretisations on different mesh levels. Number of nonlinear ('#NL') and total number of linear ('Σ#MG') space-time iterations as well as CPU time. Full space-time multigrid. Comparison between optimisation and simulation.

Table 8.20 compares the CPU time for the optimisation with that for a corresponding simulation. Due to the increased matrix stencils, the computations generally take 1.5–2 times longer than those in the unstabilised case. However, the convergence results are similar, cf. Table 8.13. The number of nonlinear iterations hardly changes, thus the nonlinear solver works quite effectively. The number of linear iterations even reduces in most cases in comparison to the unstabilised case.

As a conclusion, the convergence of the proposed KKT solver methodology is maintained by the edge-oriented stabilisation and the results show good approximation properties. However, for a low/medium Re number example like the one presented in this work, it is possible to calculate accurate results even without any stabilisation. What happens for higher Re number examples is a completely different topic. Usually, it is not possible to compute results at all without any suitable stabilisation. For a standard CFD simulation, the edge-oriented oriented stabilisation technique already showed promising results in this case, and the results from this section indicate that this stabilisation likely works well for higher Re numbers in the context of optimal control.

8.6. Summary and conclusions

This chapter completes the numerical analysis of the discretisation and solver strategy described in Chapter 2 and 3. Together with Chapter 7, an extended analysis has been carried out to demonstrate the applicability of the methodology also in non-analytical cases. In this context, the focus of these chapters has been on benchmark examples from CFD.

The previous chapter has focused on the Stokes and Navier–Stokes examples and illustrated the convergence behaviour of the solver, in the unconstrained as well as in the constrained case. For test examples of the 'Driven–Cavity' type, the computational time of simulation and optimisation has been compared, and the optimisation has been shown to be only five to twelve times more expensive than a simulation.

This chapter has extended the analysis to a more general example, using a 'Flow–Around–Cylinder' configuration. Based on the forces acting on the cylinder, the influence of the choice of the regularisation parameters to the solution has been illustrated. Fur-

thermore, a benchmark example for the optimal control of the nonstationary distributed control of the Navier–Stokes equations has been defined, and the solution of the optimal control problem has only been by a factor 20–50 more expensive than a corresponding simulation. Furthermore, it has been demonstrated that the choice of the discretisation is simultaneously reflected in the accuracy of the solution and the efficiency of the solver. In particular, higher order discretisations are highly favourable (if applicable).

In a final appendix section, the edge-oriented stabilisation technique has been applied and analysed w.r.t. accuracy and efficiency. For the benchmark configuration defined in the beginning of the chapter, the linear complexity of the solver has been maintained and the solutions have shown an accuracy similar to the unstabilised case, often even better. This indicates that the edge-oriented stabilisation is likely applicable to the optimal distributed control of the Navier–Stokes equations with higher Reynolds numbers, where stabilisation is usually mandatory.

All in all, it has been illustrated that the proposed solver methodology is well applicable to nonstationary distributed optimal control problems. Even for the optimal control of a Re=100 flow, the costs for the optimisation have been of acceptable size in comparison to a simulation, and the numerical effort grows linearly with the problem size.

9

Further extensions, summary, conclusions and future work

This chapter is the concluding chapter of the complete work. It is divided into two parts. Section 9.1 gives a general overview about the methods that have been presented in this work. The main key points of the described discretisation and solver strategy are summarised and the main results from the numerical discussions are highlighted. Furthermore, Section 9.2 highlights some possible future extensions. These include on the one hand extensions to the discretisation and solver components (for example higher order space-time discretisations or globalisation strategies) and on the other hand, the adaption of the method to more general model problems (like boundary control and parametrised control spaces). The chapter closes with a discussion of possible algorithmic enhancements regarding parallelism, memory management, model reduction and adaptivity.

9.1. General summary and discussion

Optimal control with partial differential equations is one of the most advanced topics in applied mathematics. Many researchers focus on the optimisation of different physical phenomena, and a lot of fundamental research has been carried out in recent years to formulate optimality conditions, modelling and discretisation techniques as well as solution strategies (see for example [5, 7, 11, 43, 69, 92, 92, 94, 112–114, 117, 130]).

The biggest problem in bringing the theory to an application for practically relevant optimisation tasks is the size of the underlying discrete optimality system — and the fact that standard numerical algorithms for the solution of linear and nonlinear systems as well as general solvers for nonlinear programming scale badly with the problem size. This renders most problems uncomputable by standard methods, in particular in the time dependent case. Due to the elliptic character of time-dependent optimality systems, all unknowns in space and time are strongly coupled, and numerical algorithms must therefore exploit mathematical structures to be efficient. This is the point where hierarchical methods come into play, as such methods are typically effective for such problems.

9.1.1. Main key points of the hierarchical solution approach

This work has described a fully hierarchical approach for solving nonstationary optimal control problems from PDE-constrained optimisation based on the primal and dual variables. The key point from the mathematical point of view, exploited by the algorithms, has been the elliptic character of the KKT systems associated with common time dependent, PDE-constrained optimal control problems. Highly advanced algorithms and discretisa-

tions known from the simulation of the underlying PDE have been combined in a modular way to provide efficiency, robustness and flexibility. The approach has been demonstrated for the optimal distributed control of the heat equation, the Stokes equations and the Navier–Stokes equations. A comprehensive numerical analysis has been carried out to give advice to practitioners concerning the applicability of the approach, the choice of the discretisation, the hierarchy, the different solver components and necessary parameters. It has been demonstrated that the method is able to solve the underlying KKT system of a nonstationary optimal control problem problem in such a way that

$$\frac{\text{effort for the optimisation}}{\text{effort for the simulation}} \leq C$$

where *effort for the simulation* is measured by the best available numerical methods. The constant C has had a moderate size of < 30 in most numerical examples, including the optimal distributed control of the nonstationary Navier–Stokes equations at Re=100.

9.1.2. Discretisation concept and solver design

Chapters 2 and 3 have illustrated the general discretisation concept and presented the solver approach. Chapter 4 has described the extension to some more general optimal control problems and discretisation strategies for improved flexibility in the numerical tests.

Chapter 2 has focused on analytical properties of the KKT system. This system stems from the application of the Lagrange multiplier concept to the optimal control of a nonstationary partial differential equation, in particular, to the distributed control of the heat equation and the Stokes/Navier–Stokes equations. It has been demonstrated that the KKT system has an elliptic character in space-time which is the key for the hierarchical approach.

Next, the discretisation concept from [11] has been applied to discretise the optimal control problem in space and time simultaneously. This concept is characterised by the following key points:

- The approach applies the *First-Optimise-Then-Discretise* strategy. At first, the optimal control problem to be solved is reformulated as a KKT system similar to the SQP approach. The control u is eliminated. This step is advantageous for the construction of the proposed smoothers since the resulting KKT systems consist of exactly two equations of similar type. Furthermore, it reduces the number of unknowns and offers the possibility to solve the KKT system without a discretisation of the control u.
- The KKT system is discretised in time with a one-step scheme, in space with the finite element approach. Both discretisations are independent of each other. However, the approach applies the discretisation in such a way that the *First-Optimise-Then-Discretise* strategy commutes with a particular *First-Discretise-Then-Optimise* strategy which ensures the meaningfulness of gradients — and in consequence, of the dual variable — with respect to a discrete counterpart of the continuous functional to minimise.
- The special interpretation of the right-hand side (and the pressure) for the general one-step θ-scheme induces sparse, three-band block matrices in the time-discrete and fully discretised system. This form is the essential point exploited

by space-time smoothers, and the sparsity allows to perform matrix-vector multiplications with computational costs proportional to the number of unknowns in space and time.

Chapter 3 has applied the discretisation strategy from Chapter 2 to a hierarchy of space-time meshes. A Newton-Multigrid strategy has been applied to solve the system on the finest mesh. The nonlinear problem on the space-time cylinder has been reduced to sequences of local linear problems in space. The general method can be summarised as follows:

- A space-time Newton solver defines an outer iteration for the nonlinearity. This iteration is carried out on the finest level of a problem hierarchy.
- During the Newton iteration, linear systems in space and time have to be solved. For this purpose, a space-time multigrid is applied which exploits the defined problem hierarchy.
- There are different choices possible how to define the hierarchy. The choice of the hierarchy influences the prolongation/restriction operators. For the special θ-scheme time discretisation used in Chapters 2, appropriate prolongation/restriction operators are derived with the concept of discrete abstract functions.
- The block structure of the space-time matrices allows to apply block solver strategies as smoothers in the space-time multigrid. These reduce the space-time strategy to local linear systems in space. Finally, these local linear systems are solved with a 'standard' monolithic multigrid approach based on an underlying finite element hierarchy.
- The monolithic multigrid solver in space is applicable to general equations and primal/dual systems. The component responsible for this flexibility is the PSC-SMOOTHER approach which can be used for smoothing of general linear systems from partial differential equations.

Chapter 4 has been an appendix chapter to Chapters 2 and 3. Some common extensions to the model problems and the discretisation have been introduced. In particular, the end time observation, the do-nothing boundary conditions and the case of constraints in the control have been illustrated in detail. Finally, the semi-explicit time discretisation has been described. All these extensions have provided additional flexibility in the numerical analysis.

9.1.3. Numerical results

The numerical analysis of the proposed solution strategy has been carried out in Chapters 5 to 8. Basic properties and parameters of the solver components, the choice of the discretisation and the hierarchy have at first been illustrated and verified based on analytical test examples with the heat equation and and Stokes equations. In a second step, the analysis has focused on the Stokes and the Navier–Stokes equations, demonstrated the level-independent convergence behaviour, the robustness regarding changes to the regularisation parameters as well as the superlinear convergence of the Newton iteration in the unconstrained and constrained cases. Finally, a benchmark problem from CFD has been used to demonstrate the influence of the space-time discretisation on the accuracy and efficiency of the solver.

Chapter 5 has carried out the first part of the analysis by applying the space-time multigrid solver to linear analytic test problems for the heat equation and the Stokes equations. It has been shown that the solver converges level-independently for linear and higher order space-time discretisations. However, the Crank–Nicolson scheme necessitates the correct prolongation/restriction operators. Local problems in space have been solved inexactly without influencing the global convergence. Finally, it has been demonstrated that the choice of the multigrid cycle and the level hierarchy of the space-time meshes is crucial for the overall computational costs of the algorithm; an improper choice leads to a loss of the linear complexity in terms of CPU time.

Chapter 6 has analysed the space-time discretisation. With the help of analytical test examples based on the heat equation and the Stokes equations, the influence of the space-time mesh to the error in the solution has been illustrated. In particular, the space discretisation has to be coupled to the time discretisation in order to reduce the global error. This can be done either by choosing an appropriate refinement (higher refinement level in space or more timesteps in time) or by using a higher order approach (higher order finite elements or timestepping scheme). All in all, it has been verified that the space-time mesh should be chosen such that the space error and the time error are equilibrated.

Finally, the chapter has focused on the modified Crank–Nicolson scheme. It has been shown that this scheme provides second order accuracy. Furthermore, the convergence properties of the nonlinear and linear solver are similar, independent of whether the modified or the 'traditional' Crank–Nicolson method is used. Thus, the solver is stable against this choice.

Chapter 7 has focused on an analysis of the space-time multigrid and Newton solver based on a linear and a nonlinear 'Driven–Cavity' example from CFD. It has been shown that both solvers are robust with respect to the choice of the regularisation parameters and that anisotropies in the space-time grid influence the efficiency of the linear solver. Up to a certain extent, it has been possible to compensate for a loss in the efficiency by a proper choice of the coarsening strategy.

A comparison of different strategies for choosing the nonlinear solver has revealed that both the standard Newton method and the inexact Newton method show a quadratic convergence behaviour. The inexact Newton has turned out to be the preferable choice in practice due to the fact that the ratio between convergence speed and CPU time is nearly optimal without any parameter adjustment. Finally, applying the semismooth Newton variant as nonlinear solver in the case of constrained control, the solver has still converged superlinearly.

Chapter 8 has concentrated on the Navier–Stokes equations and defined a benchmark problem based on a stationary Re=20 and a nonstationary Re=100 'Flow–Around–Cylinder' benchmark known from CFD. The influence of the regularisation parameters in the KKT system on the solution has been analysed. Furthermore, the advantage of higher order methods has been shown. These have allowed to reduce the number of time intervals and the number of unknowns in space without loss in the accuracy. Finally, the edge-oriented stabilisation has been applied, and it has been demonstrated that this stabilisation technique works well in the optimal control context.

9.2. Possible future extensions

The proposed solver strategy has been introduced based on a set of distributed control model problems. The heat equation, the Stokes equations and the Navier–Stokes equations have been used to demonstrate that methods used for the discretisation and simulation of the underlying PDE can be adapted to an optimal control setting. This renders the method flexible in terms of the choice of the space and time discretisation and the design of underlying preconditioning techniques, and a lot of modifications are possible here. Furthermore, the model problems which have been used for the demonstration of the method are only basic examples which can be extended in numerous ways, with respect to the underlying equation as well as with respect to the optimal control problem.

The following section gives an overview of possible extensions to the discretisation, introduces enhancements to the solver components and sketches the application of the solver methodology to more general PDEs and optimal control problems. The section (and the whole work) closes with an introduction about advanced algorithmic issues for really large size problems. These include the aspects 'memory management' and 'parallelism' and are of high importance for industrial applications.

9.2.1. Advanced discretisation and solver components

The proposed solver concept decouples the space discretisation from the time discretisation and does not apply, e. g., special space-time finite elements. This leads to a high flexibility in the choice of both discretisations and allows to incorporate also most recent technologies actually developed for the underlying equations.

Higher order space discretisations Chapter 6 has demonstrated that the space and the time discretisation are coupled w. r. t. accuracy. For the considered non-analytical model problems in Chapter 8 however, the space discretisation has had a stronger influence to the overall solution than the time discretisation. Unfortunately, a finer mesh in space leads to a tremendous increase of the number of unknowns since this happens in every time interval. For laminar, smooth problems from CFD, higher order finite elements are therefore preferable since they provide much higher accuracy for the same number of unknowns.

For the Stokes and the Navier–Stokes equations, the highest order finite element pair considered in this work has been the Q_2/P_1^{disc} element pair. Of course, one can even think of using a $Q_n/P_{n-1}^{\text{disc}}$ approach to achieve even higher accuracy. A possible recent alternative developed in [109] is for example the $\tilde{Q}_2/P_1^{\text{disc}}$ finite element pair which is based on integral mean values on the element edges (in 2D) or faces (in 3D). As shown in this article, the $\tilde{Q}_2$ element can even be extended to a $\tilde{Q}_n$ element. In comparison to Q_n, the main advantage of $\tilde{Q}_n$ is the reduced coupling of the degrees of freedom, which takes only place via the edges/faces and not via the vertices (as it is the case for Q_n). Therefore, the matrix stencils are smaller, which leads to higher computational efficiency. The reduced coupling is also a big advantage in parallel computing (see also below), e. g., in the context of the currently developed software package FEAST [14, 106, 150]. It reduces the amount of communication on the boundary of subdomains.

Higher order time discretisations The time discretisation in Chapter 2 has been carried out with a one-step θ-scheme. Chapter 8 has demonstrated that the Crank–Nicolson scheme has been highly advantageous in comparison to the implicit Euler scheme in terms

of accuracy. This way, the overall number of time intervals — and in connection, the total amount of memory as well as the computational time — has been reduced significantly.

If the solution is smooth in time (which is the case for most laminar flow problems), a higher order timestepping technique is therefore preferable. The demonstrated discretisation strategy in Chapter 2 is modular enough be used also with other timestepping techniques. A possible choice for a future extension is for example the Fractional step θ-scheme [142], which improves the stability of the time stepping. Another alternative which has recently been developed [96] is the cGP(m) scheme which uses a finite element approach also in time, thus allowing a time discretisation with arbitrary order.

However, there are open questions involved in this context. The discretisation scheme is designed in such a way that the *First-Optimise-Then-Discretise* approach commutes with the *First-Discretise-Then-Optimise* approach, and a discretisation with a different timestepping technique should respect this design aim as well. Depending on the choice of the timestepping scheme — and in correspondence, the choice of the time integration in the discrete functional — a larger coupling of the unknowns in time can be expected, leading to more 'off-diagonal blocks'. In the case of the cGP(m) scheme, there will even be subblocks in the space-time matrix corresponding to the different time intervals (since each time interval contains a set of fully coupled solutions in time). The realisation is therefore much more complicated. In contrast, the special one-step θ-scheme applied in Chapter 2 resulted in a three-band block matrix.

Finally, the interpretation of the pressure and the right-hand side in time should be considered, similar to the Crank–Nicolson approach. This influences the choice of prolongation/restriction operators in time. It is noted that, having the prolongation at hand, the restriction in time can be created with the help of the concept of discrete abstract functions, cf. Section 3.4 on page 53ff.

Advanced solver components The solver method in Chapter 3 has been defined in a basic way and allows a couple of extensions to improve efficiency and robustness. Basically, it is possible to modify the outer Newton solver, the space-time multigrid solver and/or the local multigrid solver in space — or some of them in combination, as it has been done for the inexact Newton algorithm. One can think of the following extensions, some of them adaptions of well known strategies from the literature:

Newton line search The Newton line search strategy is a globalisation strategy which aims at improving the robustness of the nonlinear solver. For this purpose, an automatically chosen damping parameter damps the Newton steps. For an overview about this method, see Appendix B.

Cascaded linear space-time preconditioners In the proposed solver strategy, the Newton solver has been preconditioned by a space-time multigrid solver. A common extension is to embed the space-time multigrid solver as preconditioner in, e. g., a BiCGStab or GMRES algorithm. This is a similar strategy as it has been used in Section 3.5.3 on page 69f for cascaded smoothers and can help to improve the robustness.

Space-time pressure Schur complement techniques The definitions of the space-time smoothers follow the strategy to loop forward and backward in time, solving a local linear system in each timestep. Similar to the PSCSmoother in space (Section 3.6 on page 70) one can alternatively think of a pressure Schur complement solver in space and time. This would have to loop about all cells in space and time,

setting up a local linear system which covers the current, previous and next timestep in each spatial cell.

Patch-based pressure Schur complement techniques For higher robustness of the spatial multigrid, the PSCSmoother component can be modified to set up local systems in space which cover a patch of cells. This has been introduced in [132] in the context of the simulation of the Boussinesq equations. Finally, also the PSCSmoother can be embedded as a preconditioner in a BiCGStab or GMRES smoother to improve robustness.

9.2.2. Further model problems and applications

The solver methodology in this work has been introduced using a set of basic model problems of distributed control type. However, a couple of further extensions to other equations, discretisation strategies and optimisation problems are possible which have a structure similar to the considered model problems. Thus, the solver methodology can be adapted to in a straightforward way. Some of these are sketched in the following.

Boundary control In boundary control, the control forces act on the boundary of the domain. This is often easier to realise in practice: While the distributed control of the heat equation in a 3D domain, e. g., has to be applied by microwaves in a rather complication way, boundary control can be applied by simple heating devices. A brief introduction about this type of control is given in Appendix A.

Parametrised control spaces This type of control is rather common in practice. It models the situation that the control acting in the domain or on the boundary is given by only finitely many parameters, independent on the refinement of the underlying equation. Appendix A gives a short overview about the application of the proposed solver strategy in this case.

Quasi-Newtonian flow problems For Quasi-Newtonian flow, the diffusion operator in the Navier–Stokes equations has to be modified in a nonlinear fashion. A typical formulation reads in extension to the Navier–Stokes model in Section 2.1 on page 22f,

$$\begin{aligned} y_t - \operatorname{div}\sigma + y\nabla y + \nabla p &= u, \\ -\operatorname{div} y &= 0, \end{aligned}$$

with $\sigma = \sigma(y,p) = 2\nu(y,p)D(y)$ for the deformation tensor $D(y) = \frac{1}{2}(\nabla y + (\nabla y)^{\mathsf{T}})$ and a nonconstant viscosity $\nu(y,p)$. Additional initial/boundary conditions are omitted for simplicity. The choice of the viscosity influences the properties of the fluid. A possible choice is for example $\nu(y,p) = \nu_0 \zeta^{\frac{r}{2}-1}$ with constants $\nu_0 > 0$, $r > 1$ and $\zeta := D_\Pi(y) = \frac{1}{2}\mathrm{tr}(D^2(y))$ the second invariant of the deformation tensor, which represents the Power law model. For an overview about possible choices, see [124].

This equation is rather similar to the standard Navier–Stokes equations but involves an additional nonlinearity in the diffusion part. In [124], Ouazzi derived the corresponding Fréchet derivative. If this equation is used as a constraint in an optimal control problem similar to Section 2.1, the adjoint of this operator has to be derived in order to formulate the corresponding continuous KKT system. Moreover, the complete discretisation strategy

can be applied, so the KKT system can be solved with the proposed hierarchical approach in the same way as for the Navier–Stokes model problem.

Non-isothermal flow problems Non-isothermal flow problems couple an additional temperature equation to the standard Navier–Stokes equations. A typical example is the Boussinesq approximation,

$$\begin{aligned} y_t - \nu\Delta y + y\nabla y + \nabla p + \mathrm{Gr}\ \Theta g &= 0, \\ -\operatorname{div} y &= 0, \\ \Theta_t - \frac{1}{\mathrm{Pr}}\Delta\Theta + y\nabla\Theta &= u, \end{aligned}$$

with $g = (0,1)^\mathsf{T}$ in 2D or $g = (0,0,1)^\mathsf{T}$ in 3D the direction of gravity, Gr the Grashof number, Pr the Prandl number and $\Theta : \mathcal{Q} \to \mathbb{R}$ a temperature variable. $\mathcal{Q} = (0,T) \times \Omega$ denotes again the usual space-time cylinder. In this form, the equation can be incorporated as a constraint in a distributed control problem similar to Section 2.1 on page 22ff. The variable $u : \mathcal{Q} \to \mathbb{R}$ in this example defines a control acting on the temperature equation. In this formulation, the flow can be influenced by modifying the temperature, e. g., by magnetic fields, heaters or microwaves.

This example has been the guiding equation for the optimal control of semiconductor melts in [11]. Baerwolff et al. apply the same discretisation scheme as used in Chapter 2 of this work. Thus, the complete solver structure described in Chapter 3 can be applied, and the problem can be solved in the same hierarchical way. Non-isothermal flow can also be coupled to Quasi-Newtonian flow; for an overview, see for example [46]. The control can be applied in the context of distributed control as well as boundary control; examples can be found, e. g., in [1].

Prolongation and restriction for the third equation can be carried out in the same way as for the primal Navier–Stokes equations. However, the third equation (and its corresponding dual) induces a structural change in the local systems solved by the spatial multigrid in each time interval. As a consequence, the PSCSmoother algorithm has to be changed, which is the main difference to the Navier–Stokes based model problem.

Concluding remarks In addition to the above extensions, one can also think about applying the method, e. g., to fluid-structure interaction problems or compressible flows. Compressible flows are considered for example in [41, 42] where Collis et al. derive adjoint systems involving the optimal control of the kinetic energy in a subdomain at the final time or the heat transfer or shear stress over a part of the boundary, respectively. Nonstationary fluid-structure interaction problems have been considered, e. g., in [120] in the context of the optimal control of the position of a rigid body in a flow. In both cases, the underlying KKT equations contain a primal equation forward in time and a dual equation backward in time, thus, the discretisation and solver approach can be applied. However, due to the change in the equations, the PSCSmoother component has to be changed appropriately.

Another aspect, which has not been in the close focus of this work, is the extension to domains $\Omega \subset \mathbb{R}^3$. All considered numerical examples are analysed in two spatial dimensions. Since the discretisation strategy is formulated independent of the dimension, the whole discretisation and solver strategy can be applied to 3D in the same way as in 2D. However, this quickly leads back to algorithmic problems which are typical for the context of optimal control with PDE constraints: The size of the underlying problem.

9.2.3. The problem size

Although the proposed solver methodology resolves the problem of efficiency degeneration for higher mesh levels, another major problem, the problem size itself, remains. Due to the ellipticity of the underlying KKT equation, the time acts as an additional dimension that quickly increases the problem size. The test examples in the chapters about numerical tests quickly grew larger than 12GB memory already on a medium spatial level due to a high number of timesteps. For an optimisation in 3D, the situation would even be more dramatic, as the time introduces a fourth dimension. For smooth problems, using a higher order finite element discretisation with a higher order timestepping scheme can be a preferred way to reduce the total number of unknowns which have to be saved. However, for real industrial applications, this is usually not enough.

Parallelism and memory management A possible remedy can be seen in the combination of parallel computing with multigrid schemes. Different approaches can be successful here:

- For faster preconditioning in space (in case the spatial problems must be computed on a very fine mesh), domain decomposition can be used and a distributed multigrid scheme in space can be applied. One possible approach is the ScaRC concept [106] which has successfully been implemented in the FEAST project [15, 106, 147, 149, 150].

- For problems with many unknowns in time, the different timesteps can be distributed onto different processors such that each processor maintains only one or a limited number of timesteps in its memory. That way, the complete main memory of a cluster can be used to store all the space-time vectors that appear in the space-time solver.

- In combination with the previous point, it is possible to define distributed solvers. The FBJacSolver for example is trivially parallelisable, a processor holding a subset of timesteps can apply the underlying solution operator in space to every of its timesteps in any arbitrary order, the subsets are completely decoupled. A solver like FBGsSolver or FBSimSolver is harder to parallelise; one can thing of red-black parallelisation techniques, possibly in combination with additional block-decoupling strategies: A processor applies the block scheme just to the time intervals which are present in its main memory. On the one hand this usually leads to some loss in the convergence speed in practice due to the missing coupling of the time-blocks. On the other hand, the parallel processing of the blocks speeds up the overall computation. It is a question of its own how to distribute the solutions in time onto the different processors to reduce the overall CPU time.

- As an extension to the previous point, one can even think of a ScaRC preconditioner in time. Each processor contains a number of timesteps and applies a local time-multigrid scheme to these. The different time-blocks are coupled by the solutions at their time-endpoints.

If the required memory is too large for all timesteps to be kept in the main memory of the computer (independent of whether it is a single-processor machine or a cluster), one has to think about clever memory-management routines, and in particular, 'out-of-core' techniques: The main set of solutions must be saved to disk while the necessary subvectors have to be read into the main memory. Here, one can use multithreading: One core reads

solutions from the hard disk while the other assembles the nonlinear spatial systems and performs the preconditioning. Of course, this necessitates a fast filesystem to prevent too much loss of efficiency due to IO operations. One can also think of keeping the smaller space-time problems of the hierarchy in the memory while only swapping out the higher levels to hard disk.

Finally, due to the extreme high computational time necessary for large scale optimisation problems, it is essential to incorporate checkpointing and restart facilities. This prevents data loss in case the calculation breaks down due to hardware damage or similar issues. Note that the general structure of the algorithm naturally supports this: The complete nonlinear solution can be saved, e. g., after each nonlinear iteration and/or after each linear iteration. Restarting from such a saved solution allows to seamlessly continue the iteration. With an appropriate design of the algorithm, this aspect can be combined with the *out-of-core* techniques from above to save memory.

Model reduction and adaptivity In addition to globally higher order spaces, clever memory management and parallel processing, one should think about further techniques to reduce the number of unknowns. One possibility is the use of model reduction methods like POD ('Proper Orthogonal Decomposition') which have been proven to be useful in [85]. Another approach, which is typically known from simulation, is adaptivity: Apart from the p-adaptivity which increases the order of the trial space where needed, but which is less useful for nonsmooth problems, the following two mesh-based adaption methods are of interest:

- By the use of h- (and what is called here 'k'-) adaptivity, additional unknowns in space and time are introduced. But where h-adaptivity in space can be hard due to the complicated handling of hanging nodes in quadrilateral meshes, k-adaptivity in time should be simpler: Time is a single dimension, so k-adaptivity means imposing additional time intervals where required and enlarging/removing time intervals where not required.

- r-adaptivity can be used to 'deform' the underlying mesh such that unknowns are shifted to where they are needed without changing the topological structure. In space, this was done for example in [63–65], in time this would mean to shift the location of timesteps, leading to a non-equidistant time-mesh. This would lead to different weights in front of the mass matrices realising the timestepping scheme but would in particular not destroy the tridiagonal structure of the space-time matrices.

A dynamic change on the mesh (on the highest level, in space or in time) can be carried out, e. g., after a specific number of nonlinear iterations. Such a change usually leads to an interpolation of the current (fine grid) solution and thus, the nonlinear iteration has to be restarted. The most critical point in using such methods is the creation of the multigrid operators: All types of adaptivity directly influence the definition of the prolongation/restriction operators in the space-time multigrid scheme. But apart from these technical details, the whole method is expected to work well.

A

Further model problems

To demonstrate discretisation and solver algorithms, the descriptions in Chapters 2 and 3 concentrated on a set of model problems of distributed control type. However, the method is general enough to be applied to a couple of more general, well known and common example problems as well. Exemplary, this appendix gives an brief overview about the application of the discretisation and solver strategy to boundary control (Section A.1) and parametrised control spaces (Section A.2). These two model problems are highly relevant in practice: Boundary control allows to influence a solution by modifying the boundary conditions, which is usually easier than controlling volume forces. Furthermore, parametrised control spaces model the situation that a control is given only by finitely many control parameters.

A.1. Boundary control

A related, alternative control strategy is the boundary control approach. This is sketched here as a first example for the generalisation of the proposed discretisation and solution approach to more general optimisation problems.

Instead of applying a force in the full domain or in parts of it, this strategy controls a flow via the boundary or parts of it, e. g., by prescribing a special inflow. From the analysis point of view, the boundary control is a much harder problem than the distributed control. Especially the use of Dirichlet control necessitates the use of non-variational and hence more elaborate function spaces, leading to the theory of very weak solutions, cf. [16, 35]. A number of recent papers has been published about this topic, see for example [51, 78, 86, 92, 111]. However, ignoring the analysis for the moment, the corresponding KKT systems can formally be written down without too many difficulties.

The following paragraphs give an overview about this type of control strategy to highlight the differences to the distributed control on a special example and to illustrate possible realisations of this control strategy within the proposed solver methodology. However, as this work focuses on the distributed control as the central model problem, there is no further discussion or numerical analysis of the boundary control.

Boundary control of the Navier–Stokes equations As a model problem, the optimal Dirichlet boundary control of the Navier–Stokes equations is considered, given by

$$J(y,u) := \frac{1}{2}||y-z||_{Q}^{2} + \frac{\alpha}{2}||u||_{L^2((0,T)\times\Gamma_c)}^{2} \quad \longrightarrow \quad \min! \tag{A.1}$$

$$
\begin{aligned}
\text{s.t.} \qquad y_t - \nu\Delta y + y\nabla y + \nabla p &= 0 && \text{in } \mathcal{Q},\\
-\operatorname{div} y &= 0 && \text{in } \mathcal{Q},\\
y(0,\cdot) &= y^0 && \text{in } \Omega,\\
y &= g && \text{at } (0,T)\times\Gamma_d,\\
y &= u && \text{at } (0,T)\times\Gamma_c,\\
\nu\partial_\eta y - p\eta &= 0 && \text{at } (0,T)\times\Gamma_n,
\end{aligned}
$$

for $y : \mathcal{Q} \to \mathbb{R}^{\dim}$ and $p : \mathcal{Q} \to \mathbb{R}$. The parameter $\nu > 0$ defines the viscosity which is assumed to be constant, and $\eta : \partial\Omega \to \mathbb{R}^{\dim}$ stands for the outer unit normal vector of the domain $\Omega \subset \mathbb{R}^{\dim}$. In this example, the boundary $\Gamma := \partial\Omega$ of the domain is decomposed into $\Gamma =: \Gamma_d \cup \Gamma_n \cup \Gamma_c$ with Γ_n defining the outflow or Neumann boundary, Γ_d defining the fixed Dirichlet boundary and Γ_c denoting the Dirichlet control boundary.

Following [84, 137] and taking into account that $\lambda = 0$ on $(0,T)\times(\Gamma \setminus \Gamma_n)$, the corresponding KKT system reads

$$
\begin{aligned}
y_t - \nu\Delta y + y\nabla y + \nabla p &= 0 && \text{in } \mathcal{Q},\\
-\operatorname{div} y &= 0 && \text{in } \mathcal{Q},\\
y(0,\cdot) &= y^0 && \text{in } \Omega,\\
y &= g && \text{at } (0,T)\times\Gamma_d,\\
y &= u && \text{at } (0,T)\times\Gamma_c,\\
\nu\partial_\eta y - p\eta &= 0 && \text{at } (0,T)\times\Gamma_n,
\end{aligned}
$$

$$
\begin{aligned}
-\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi &= y - z && \text{in } \mathcal{Q},\\
-\operatorname{div}\lambda &= 0 && \text{in } \mathcal{Q},\\
\nu\partial_\eta\lambda - \xi\eta + (y\eta)\lambda &= 0 && \text{at } (0,T)\times\Gamma_n,\\
\lambda &= 0 && \text{at } (0,T)\times(\Gamma\setminus\Gamma_n),\\
\lambda(T) &= 0 && \text{in } \Omega,
\end{aligned}
$$

$$
u = -\frac{1}{\alpha}\left(\nu\partial_\eta\lambda - \xi\eta,\right) \qquad \text{at } (0,T)\times\Gamma_c.
$$

The control equation can again be eliminated which leads essentially to the following set of equations,

$$
\begin{aligned}
y_t - \nu\Delta y + y\nabla y + \nabla p &= 0 && \text{in } \mathcal{Q}, && \text{(A.2a)}\\
y &= g && \text{at } (0,T)\times\Gamma_d, && \text{(A.2b)}\\
y + \frac{1}{\alpha}\left(\nu\partial_\eta\lambda - \xi\eta\right) &= 0 && \text{at } (0,T)\times\Gamma_c, && \text{(A.2c)}
\end{aligned}
$$

$$
\begin{aligned}
-\lambda_t - \nu\Delta\lambda - y\nabla\lambda + (\nabla y)^{\mathsf{T}}\lambda + \nabla\xi &= y - z && \text{in } \mathcal{Q}, && \text{(A.2d)}\\
\lambda &= 0 && \text{at } (0,T)\times(\Gamma_c\cup\Gamma_d,) && \text{(A.2e)}
\end{aligned}
$$

complemented by initial/end time/natural boundary and incompressibility conditions from above.

Differences to distributed control Compared to the distributed control of the Navier–Stokes equations, this system shows some similarities and some fundamental differences that also influence the way the system must be treated by the discretisation and the solver:

a) Similar to the case of the distributed control, the system contains a dual velocity $\lambda : \mathcal{Q} \to \mathbb{R}^{\dim}$ and a dual pressure $\xi : \mathcal{Q} \to \mathbb{R}$. These can be discretised in the same way as the primal velocity/pressure y and p. Therefore, in every time interval, the basic systems still have a form of the problems discussed in Section 2.5 on page 31.

b) The boundary conditions (A.2b) and (A.2e) are not critical, since they describe standard Dirichlet boundary conditions for y and λ.

c) The boundary condition (A.2c) is the most critical one from the implementational point of view. Its implementation depends on the type of solver which is being used.

- For a standard solver that uses a one-level forward-backward solution strategy, the computed λ from the last backward sweep can be used as Dirichlet boundary conditions for the next forward sweep,

$$y_{\text{new}} := -\frac{1}{\alpha}\Big(\nu\partial_\eta\lambda - \xi\eta\Big), \qquad \text{at } (0,T)\times\Gamma_c. \tag{A.3}$$

- However, to apply the fully monolithic nonlinear solver approach developed in this work, (A.2c) has to be treated as it is, which leads to a nonlinear boundary condition due to the coupling to λ.

  To avoid the explicit evaluation of the control term on the right-hand side in (A.3) on the boundary, it is advisable to implement the Dirichlet boundary condition (A.2c) in a weak form using penalised boundary integrals (cf. [13, 101]). This introduces some numerical difficulties due to the penalty parameter, but the realisation in the proposed solver methodology framework is straightforward since all boundary operators can be realised as matrix-vector operations.

A.2. Parametrised control spaces — linear combinations of input fields

Consider the case of the optimal distributed control of the heat equation (2.1), the Stokes equations (2.2) or the Navier–Stokes equations (2.3), see page 22f. In this form, solving the KKT system aims at computing the *best possible* right-hand side for the equation: The dual variable gives the right-hand side of the primal equation and is discretised in the same way as the primal variable. Increasing the level therefore indirectly leads to a higher accuracy of the control.

However, in real-life problems, it is often not possible to realise a computed u as physical quantity in an experiment due to technical reasons: An example is a magnetic field influencing the flow of a liquid in crystal growth (cf. [71]). A generator for such a magnetic field is usually not able to generate arbitrary magnetic fields. It has only a limited number of design parameters that allow the user to influence the field. In such a situation, the best possible solution $u = -\frac{1}{\alpha}\lambda$ is of little help for the user who tries to find the optimal setting for his or her specific magnetic field generator.

It is possible to model such a situation with a modified distributed control problem only based on the available design parameters. This leads to a modified KKT system. Based on the Poisson equation, a small introduction is given in the following to demonstrate the applicability of the proposed solver approach.

Parametrised control for the Poisson equation Consider the optimal distributed control of the Poisson equation. The set of solutions of the Poisson equation is defined in the space $V \subset H^1(\Omega)$, and $U \in \mathbb{R}^M$ denotes a set of $M \in \mathbb{N}$ design parameters for the optimal control problem. Following Example 3.1 in [92], one possibility to formulate the optimal distributed control of the Poisson equation in this case is

$$J(y, \vec{u}) := \frac{1}{2}||y - z||_Q^2 + \frac{\alpha}{2}|\vec{u}|_{\mathbb{R}^M}^2 \quad \longrightarrow \quad \min!$$

$$\begin{aligned} \text{s.t.} \qquad -\Delta y &= B\vec{u} \qquad \text{in } \Omega, \\ y &= g \qquad \text{at } \Gamma, \end{aligned}$$

with $y : \Omega \to \mathbb{R}$, $g : \Gamma \to \mathbb{R}$ and the mapping B realising a linear combination of input fields, see also [50] and in particular Example 2.1 in [87]. To be more precise, with a set of M linear independent functions in space $\psi_1, ..., \psi_M \in L^2(\Omega)$, the operator $B : \mathbb{R}^M \to V^*$ can be defined as

$$B\vec{u} := \sum_{i=1}^{M} u_i \psi_i$$

and maps the design parameters $\vec{u} = (u_1, ..., u_M)$ to the space V^* of right-hand sides, which is defined as the dual space of V. Formally, the corresponding KKT system is essentially defined as

$$\left.\begin{aligned} -\Delta y &= B\vec{u} \qquad \text{in } \Omega \\ -\Delta \lambda &= y - z \qquad \text{in } \Omega \\ \vec{u} &= -\frac{1}{\alpha} B^* \lambda \end{aligned}\right\} \tag{A.4}$$

complemented by initial/end time/boundary conditions, with $B^* : V^* \to \mathbb{R}^M$ denoting the adjoint operator to B.

The operator B and its adjoint If the system is discretised with finite elements in space and if $\{\psi_1, ..., \psi_M\}$ coincide with the finite element basis, the mapping B is just the mapping of the degrees of freedom to the discrete finite element space. In the more general case that the support of each ψ_i is the full domain Ω, $i = 1, ..., M$, proper projection operators must be used:

The adjoint B^* of B can be calculated using the definition,

$$\langle \vec{u}, B^* w \rangle = (B\vec{u}, w)_\Omega$$

for all $u \in U$, $w \in V$, where $\langle \cdot, \cdot \rangle$ denotes the standard scalar product in $\mathbb{R}^M$. A simple computation reveals

$$\langle \vec{u}, B^* w \rangle = (B\vec{u}, w)_\Omega = \sum_{i=1}^{M} u_i (\psi_i, w)_\Omega = \vec{u}^\mathsf{T} \vec{f},$$

and thus, there is $B^* w = \vec{f}$ with $\vec{f} = (f_1, ..., f_M)$ defined by $f_i := (\psi_i, w)_\Omega$, $i = 1, ..., M$.

A.1 Remarks. a) In the special case that V is a finite element space, $w \in V$ has the representation

$$w = \sum_{j=1}^{N} w_j \phi_j$$

with the coefficient vector $\vec{w} = (w_1, ..., w_N)$ for a set $\phi_1, ..., \phi_N$ of $N \in \mathbb{N}$ finite element basis functions. The adjoint $\vec{f} = B^* w$ is given by

$$f_i = \sum_{j=1}^{N} w_j (\psi_i, \phi_j)_\Omega,$$

i. e., there is $B^* w = W^\mathsf{T} \vec{w}$ and $W = (W_{ij})$ being an L^2 projection matrix defined by $W_{ij} = (\phi_j, \psi_i)_\Omega$. Correspondingly, $w = B\vec{u}$ is given by $\vec{w} = W\vec{u}$.

b) The extension of a) to the heat equation is straightforward, since the operators B and B^* are purely spatial operators; they have to be applied in every timestep. An extension to the Stokes and Navier–Stokes equations can be done in a similar way by applying the corresponding B and B^* operators in every velocity component.

c) The elimination of $\vec{u}$ in (A.4) leads to the system

$$\begin{aligned} -\Delta y &= -\frac{1}{\alpha} B B^* \lambda \quad \text{in } \Omega \\ -\Delta \lambda &= y - z \qquad\quad \text{in } \Omega \end{aligned}$$

and for the heat equation and the Stokes/Navier–Stokes equations, the corresponding equations look similar. In this form, the proposed solver methodology can be applied directly. On the linear algebra level, the operation $BB^*\lambda$ is realised by multiplying the vector with the degrees of freedom of λ to the matrices W and W^T. However, it has to be noted that although the number of design variables is reduced, the final size of the system is not reduced, only a special type of projection is applied to the dual variable λ.

B

Globalisation and enhanced robustness

One of the main disadvantages of the Newton algorithm in Section 3.2 on page 51 is that it is only locally second order convergent with a limited convergence radius. It can frequently happen that an initial iterate w_0^σ leads to a non-convergent iteration. In such a situation, 'globalisation strategies' can be applied to enhance robustness. There are different possibilities known from the literature, and for reference purposes, two variants are documented below. Section B.1 describes the 'adaptive Newton' which uses a couple of nonlinear defect correction steps before switching to the actual Newton algorithm. Section B.2 gives an introduction into the 'Newton line search' method which realises a step size control for Newton steps. Both methods can be seen as simple extensions to the proposed algorithms aiming at global convergence.

B.1. The adaptive Newton algorithm

The nonlinear defect correction loop introduced in Section 3.2 on page 51 can be summarised in two steps,

$$\begin{aligned} &1.)\ \text{Solve:} && C(w_i^\sigma) g_i = d_i := \left(f^\sigma - G^\sigma(w_i^\sigma) w_i^\sigma \right) \\ &2.)\ \text{Update:} && w_{i+1}^\sigma := w_i^\sigma + g_i, \end{aligned}$$

for an initial guess $w_0^\sigma \in W^{\mathrm{NLMAX}}$ and $i \in \mathbb{N}_0$ denoting the current iteration. One possibility to maximise the convergence radius while still tying to obtain superlinear convergence is to combine the fixed point method with the Newton or inexact Newton method; this variant is called *adaptive Newton* here, see Algorithm B.1. With r_i^{OptNL}, the i-th nonlinear residual is denoted, n_{FP} refers to a maximum number of fixed point steps (e. g., $n_{\mathrm{FP}} = 10$) and $\varepsilon_{\mathrm{OptFP}}$ a relative limit for the residual where to switch from the fixed point method to the (inexact) Newton (e. g., $\varepsilon_{\mathrm{OptFP}} = 10^{-2}$).

The idea of the algorithm, which has also beed used in a similar form in [124] for numerical simulations, is to use a weaker (but more stable) fixed point method in the first iterations while switching to the stronger Newton algorithm as soon as the solution reaches to a region of (stable) convergence. In this context, the intention of the algorithm is not directly to provide global convergence of the nonlinear iteration — the backtracking line search algorithm to be introduced in Section B.2 is much better suited for this task, and the adaptive Newton can even be combined with it. However, from a heuristic point of view, one step of the fixed point method with the Newton preconditioner $C(w_i^\sigma) := F^\sigma(w_i^\sigma)$ can be seen as being a much harder task for the linear (space-time multigrid) solver than one step with the standard preconditioner $C(w_i^\sigma) := G^\sigma(w_i^\sigma)$ (which is only a part of the Newton preconditioner). Thus, this method tries to enhance the robustness by providing simpler linear subproblems to the linear solver in the first couple of steps.

Algorithm B.1 Adaptive Newton algorithm

Predefined constant: $n_{\mathrm{FP}} \in \mathbb{N}_0$: maximum number of fixed point steps
Predefined constant: $\varepsilon_{\mathrm{OptFP}} > 0$: limit for the residuals in the fixed point iteration

function AdaptiveNewton(w_0^σ,f^σ)
  $i \leftarrow 0$
  while (not converged) **do**
    if ($i < n_{\mathrm{FP}}$) and $||r_i^{\mathrm{OptNL}}||_{l_2} > \varepsilon_{\mathrm{OptFP}}||r_0^{\mathrm{OptNL}}||_{l_2}$ **then**
      Calculate w_{i+1}^σ by one step of the fixed point iteration (3.9)
      with the preconditioner $C(w_i^\sigma) := G^\sigma(w_i^\sigma)$.
    else
      Calculate w_{i+1}^σ by one step of the fixed point iteration (3.9)
      with the Newton preconditioner $C(w_i^\sigma) := F^\sigma(w_i^\sigma)$,
      eventually solved inexactly.
    end if
    $i \leftarrow i + 1$
  end while
  return w_i^σ
end function

B.2. Newton line-search

A common strategy to enhance the robustness of the Newton algorithm is to introduce an adaptively chosen damping parameter. The following introduction gives a brief overview about this method as an extension of the space-time Newton method considered in this work. More detailed, comprehensive discussions about this topic can be found, e. g., in [52, 53, 56, 104, 123, 126, 135, 160]).

In many cases, the Newton line-search strategy ensures global convergence of the Newton algorithm, independent of the initial iterate. The corresponding algorithm is a modification of (3.8) on page 51 with $C(w_i^\sigma) = F(w_i^\sigma)$ and can be formulated as follows, using the notation from Section 3.2,

$$w_{i+1}^\sigma := w_i^\sigma + \omega_i F(w_i^\sigma)^{-1}\big(f^\sigma - G^\sigma(w_i^\sigma)w_i^\sigma\big), \qquad i \in \mathbb{N}_0, \tag{B.1}$$

with a sequence of appropriately chosen damping parameters $0 < \omega_i \leq 1$ and an initial iterate $w_0^\sigma \in W^{\mathrm{NLMAX}}$. To be more precise, the algorithm can be formulated in three steps, starting from an initial iterate w_0^σ, see Algorithm B.2. The most important point of this algorithm is the calculation of the optimal damping parameter in (B.2b). The function GetOptimalDamping(...) that calculates this parameter can be rather complex, but a usual choice for this function is a *backtracking line search strategy* that starts with an initial guess $\omega_i = 1$ and reduces this until a desired step length is found.

The damping parameter ω_i is typically computed based on the nonlinear residual in $\mathbb{R}^n$, $n \in \mathbb{N}$ denoting the number of degrees of freedom on the space-time cylinder. It has to be remarked that the system which is solved by the damped Newton has the form

$$G^\sigma(w^\sigma)w^\sigma = f^\sigma,$$

for w^σ, $f^\sigma \in W^{\mathrm{NLMAX}}$, see (3.1) on page 48. Based on this form, the corresponding

Algorithm B.2 Damped Newton algorithm with line search

function NEWTONDAMPED(w_0^σ,f^σ)
  $i \leftarrow 0$
  while (w_i^σ not converged) **do**
    Solve $F(w_i^\sigma)g_i = d_i := \left(f^\sigma - G^\sigma(w_i^\sigma)w_i^\sigma\right)$ (B.2a)
    $\omega_i \leftarrow$ GETOPTIMALDAMPING(...) (B.2b)
    $w_{i+1}^\sigma \leftarrow w_i^\sigma + \omega_i g_i$ (B.2c)
    $i \leftarrow i + 1$
  end while
  return w_i^σ
end function

nonlinear residual in $\mathbb{R}^n$ is defined by means of a function $R : W^{\mathrm{NLMAX}} \to \mathbb{R}^n$,

$$\begin{aligned} d^\sigma &:= G^\sigma(w^\sigma)w^\sigma - f^\sigma, \\ R(w^\sigma) &:= \vec{d^\sigma}, \end{aligned}$$

with $\vec{d^\sigma} \in \mathbb{R}^n$ denoting the vector containing the degrees of freedom of d^σ. Furthermore, $||\cdot||$ denotes the standard Euclidean norm in $\mathbb{R}^n$. With this notation, Algorithm B.3 illustrates a common variant for such a backtracking strategy. It is motivated by the *Wolfe conditions* known from nonlinear optimisation, which read as follows, cf. [123, Chapter 3.1].

B.1 Definition. For a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, $s \in \mathbb{R}^n$ is assumed to be a descent direction in a point $x \in \mathbb{R}^n$, i. e., $\nabla f(x)s < 0$. Furthermore, $0 < \alpha_1 < \alpha_2 < 1$ are two given parameters. A step length parameter $0 \leq \omega \leq 1$ satisfies the *Wolfe conditions*, if the point $x + \omega s$ on the line between x and $x + s$ fulfils the following two conditions:

a) *sufficient decrease condition*

$$f(x + \omega s) - f(x) < \alpha_1\, \omega\, \nabla f(x)^{\mathsf{T}} s \tag{B.3a}$$

b) *curvature condition*

$$\nabla f(x + \omega s)^{\mathsf{T}} s \geq \alpha_2 \nabla f(x)^{\mathsf{T}} s. \tag{B.3b}$$

B.2 Remarks. a) Defining $\xi(\omega) := f(x + \omega s)$, there is $\xi(0) = f(x)$ and $\xi'(0) = \nabla f(x)s$, and ξ is continuously differentiable due to the differentiability of f. If s is a descent direction, which corresponds to $\xi'(0) < 0$, the Wolfe conditions read, written in terms of ξ,

$$\begin{aligned} \xi(\omega) &< \xi(0) + \alpha_1 \omega \xi'(0), \\ \xi'(\omega) &\geq \alpha_2 \xi'(0). \end{aligned}$$

Thus, (B.3a) can be interpreted to aim at finding some ω reducing f enough to guarantee a certain decrease, relative to a 'flattened' linear model. On the other hand, (B.3b) prevents ω from being reduced too much due to $\xi'(\omega) \to \xi'(0)$ for $\omega \to 0$.

b) In practice, α_1 is usually chosen small (e. g., $\alpha_1 := 10^{-4}$) and α_2 large, near at 1 (e. g., $\alpha_2 := 0.9$), cf. [123, Chapter 3.1]

Backtracking by a quadratic 1D model based on the *sufficient decrease* condition

Algorithm B.3 defines one example for the function GETOPTIMALDAMPING(...) and was proposed, e. g., in [104, Chapter 8]. Starting from $\omega = 1$, ω is reduced until a certain condition holds; the final ω then defines the new step length. The condition in line 3 as well as the choice of the formula in line 4 can be motivated by the sufficient decrease condition as sketched below, see also [52, 105]. It is possible to prove that similar to the standard Newton solver, this strategy leads to quadratic convergence near the optimum, cf. [104, Theorem 8.2.1] or [52, Theorem 6.3.4]. Furthermore, under some assumptions, global convergence is guaranteed, cf. [52, Section 6.3.1].

Algorithm B.3 Calculate optimal damping by a quadratic model of the cost functional in the *sufficient decrease* condition

Predefined constant: $0 < \alpha < 1$: a 'minimum descent' parameter, e. g., $\alpha = 10^{-4}$

1: **function** GETOPTIMALDAMPING_1(w_i,g_i)
2: $\omega \leftarrow 1$
3: **while** $||R(w_i + \omega g_i)|| \geq (1 - \alpha\omega)||R(w_i)||$ **do**
4: $\omega \leftarrow \omega^{\text{new}} := \frac{||R(w_i)||^2\omega^2}{||R(w_i+\omega g_i)||^2+(2\omega-1)||R(w_i)||^2}$
5: **end while**
6: **return** ω
7: **end function**

Newton algorithm for nonlinear equations

A general nonlinear problem in $\mathbb{R}^n$ can be expressed in the form

$$A(x) = 0, \tag{B.4}$$

with $x \in \mathbb{R}^n$ and $A : \mathbb{R}^n \to \mathbb{R}^n$ continuously differentiable. By $J : \mathbb{R}^n \to \mathbb{R}^{n\times n}$, the Jacobian of A is denoted, and it is assumed that (B.4) has a unique solution $x^* \in \mathbb{R}^n$.

In the following, $x_0 \in \mathbb{R}^n$ is an initial approximation to x^*, $i \in \mathbb{N}_0$ and x_i a current iterate. The linear model of A in x_i reads

$$M_i(x) = A(x_i) + J(x_i)(x - x_i). \tag{B.5}$$

The Jacobian $J(x_i)$ is assumed to be nonsingular in the following. As a consequence, each M_i has a unique root which is used to define x_{i+1} as a new approximate to x^*,

$$M_i(x) = 0 \quad \Rightarrow \quad x_{i+1} := x = x_i - J(x_i)^{-1}A(x_i).$$

This iteration defines the standard Newton algorithm for nonlinear equations.

Newton steps in the nonlinear optimisation context

A target function involving $A(x)$ can be defined by

$$r(x) := A(x)^{\mathsf{T}}A(x) = ||A(x)||^2, \tag{B.6}$$

with $||\cdot||$ denoting the standard Euclidean norm. By definition, there is $r(x) \geq 0$ for all $x \in \mathbb{R}^n$ and $r(x) = 0 \Leftrightarrow A(x) = 0 \Leftrightarrow x = x^*$. Thus, finding a solution to $A(x) = 0$ is equivalent to finding a global minimum of $r(\cdot)$ in $\mathbb{R}^n$.

The standard Newton algorithm in the nonlinear optimisation context reads as follows, cf. [105, Section 2.2]. Starting from $x_0 \in \mathbb{R}^n$, having computed $x_i \in \mathbb{R}^n$, $i \in \mathbb{N}_0$, the function $r(x)$ is approximated by a quadratic model in x_i,

$$\begin{aligned} r_i(x) &:= r(x_i) + \nabla r(x_i)^\mathsf{T}(x - x_i) + \frac{1}{2}(x - x_i)^\mathsf{T}\nabla^2 r(x_i)(x - x_i) \\ &= ||A(x_i)||^2 + 2\,A(x_i)^\mathsf{T}J(x_i)(x - x_i) + \frac{1}{2}(x - x_i)^\mathsf{T}\nabla^2 r(x_i)(x - x_i), \end{aligned}$$

and the next iterate is defined by

$$x_{i+1} := \operatorname*{argmin}_{x \in \mathbb{R}^n} r_i(x).$$

This approach is modified in the following, see [52, Section 6.5]. Using (B.5), an alternative approximation to $r(\cdot)$ in x_i is given by

$$\begin{aligned} m_i(x) &:= M_i(x)^\mathsf{T}M_i(x) = ||M_i(x)||^2 \\ &= ||A(x_i)||^2 + 2\,A(x_i)^\mathsf{T}J(x_i)(x - x_i) + ||J(x_i)(x - x_i)||^2, \end{aligned} \tag{B.7}$$

which matches $r_i(x)$ in the first two terms. Having computed x_i, the next iterate x_{i+1} is defined by minimising m_i,

$$x_{i+1} := \operatorname*{argmin}_{x \in \mathbb{R}^n} m_i(x). \tag{B.8}$$

Although this does not reflect the standard Newton method for nonlinear optimisation anymore, this formula still defines a descent algorithm. The iteration aims at minimising $r(\cdot)$ with special Newton descent steps. This is a well known fact from the literature and can be verified as follows.

The link between the two algorithms Taking a close look, the above two algorithms are closely related.

a) There is $m_i(x_i) = r(x_i)$ and $\nabla m_i(x_i) = 2\,A(x_i)^\mathsf{T}J(x_i) = \nabla r(x_i)$, thus $m_i(\cdot)$ is a second order approximation to $r(\cdot)$ in x_i.

b) Due to a) there is for all $s \in \mathbb{R}^n$:

$$\nabla m_i(x_i)s < 0 \quad \Leftrightarrow \quad \nabla r(x_i)s < 0,$$

i. e., in x_i, all descent directions of $m_i(\cdot)$ are descent directions of $r(\cdot)$ and vice versa. In particular, for $x_i \neq x^*$, the Newton direction

$$s_i := x_{i+1} - x_i = -J(x_i)^{-1}A(x_i)$$

is a descent direction of $m_i(\cdot)$ and $r(\cdot)$ in x_i due to $\nabla r(x_i)s_i = -||A(x_i)||^2 < 0$.

c) By definition, there is $m_i(x) \geq 0$ for all $x \in \mathbb{R}^n$ and

$$m_i(x) = 0 \quad \Leftrightarrow \quad M_i(x) = 0 \quad \Leftrightarrow \quad x_{i+1} = x = x_i - J(x_i)^{-1}A(x_i).$$

Thus, $m_i(\cdot)$ reaches its unique minimum in x_{i+1} which is calculated in the same way as in the case of nonlinear equations. As a consequence, this special descent algorithm for minimising the target function $r(x)$ is equivalent to the Newton algorithm for nonlinear equations if starting from the same x_0.

All in all, the Newton iteration for nonlinear equations can be interpreted in the nonlinear optimisation context as a special descent algorithm with Newton descent directions s_i. The damped Newton is therefore a descent algorithm with variable step size ω_i, and finding an optimal damping parameter corresponds to finding an optimal step size.

Sufficient decrease check The Wolfe conditions above specify two restrictions on the step size. Algorithm B.3 is defined based on the *sufficient decrease condition*, ignoring the *curvature condition*. For $f := r$, $x = x_i$, $s = s_i$ and $0 \leq \omega \leq 1$ in the sufficient decrease condition (B.3a), there is

$$\xi(\omega) = r(x_i + \omega s_i) = ||A(x_i + \omega s_i)||^2, \quad \xi(0) = ||A(x_i)||^2, \quad \xi'(0) = -2\,||A(x_i)||^2.$$

Thus, ω fulfils the condition if

$$\begin{aligned} & \xi(\omega) < \xi(0) + \alpha\omega\xi'(0) \\ \Leftrightarrow \quad & ||A(x_i + \omega s_i)||^2 < (1 - 2\alpha\omega)||A(x_i)||^2. \end{aligned} \tag{B.9}$$

Due to the fact that $(1-\alpha\omega)^2 = 1-2\alpha\omega+\mathcal{O}(\alpha^2)$, the last inequality can safely be replaced for small parameters α by

$$||A(x_i + \omega s_i)|| < (1 - \alpha\omega)||A(x_i)||. \tag{B.10}$$

With A corresponding to R, x_i corresponding w_i^σ and s_i corresponding to g_i, this defines the sufficient decrease check in line 3 of Algorithm B.3.

Calculating a new step size It is assumed that $0 < \omega \leq 1$ is a given step length that does not fulfil condition (B.9). The task is to find a new parameter $\omega^{\text{new}} < \omega$ with $\xi(\omega^{\text{new}}) < \xi(\omega)$. This can be obtained, e. g., by the 1D minimisation of a quadratic model of ξ, see [104, 105]. For $\lambda \in \mathbb{R}$, one defines

$$q(\lambda) := \xi(0) + \xi'(0)\lambda + c\lambda^2 \quad \text{with} \quad c := c(\omega) := \frac{\xi(\omega) - \xi(0) - \xi'(0)\omega}{\omega^2}.$$

By definition, there is $q(0) = \xi(0)$, $q'(0) = \xi'(0)$ and $q(\omega) = \xi(\omega)$. It can easily be proven that q has a minimum in $\lambda^* := -\frac{\xi'(0)}{2c}$ for $0 \leq \alpha < 1$ if (B.9) is violated, see [105, 123]. Furthermore, the inequality $0 < \lambda^* \leq \frac{\omega}{2(1-\alpha)}$ can be verified directly, thus for $0 \leq \alpha < \frac{1}{2}$, there is $\lambda^* < \omega$. λ^* defines the new step length. There is

$$\omega^{\text{new}} := \lambda^* = -\frac{\xi'(0)}{2c} = \frac{||A(x_i)||^2\omega^2}{||A(x_i + \omega s_i)||^2 + (2\omega - 1)||A(x_i)||^2} \tag{B.11}$$

and with A corresponding to R, x_i corresponding w_i^σ and s_i corresponding to g_i, this defines the formula in line 4 of Algorithm B.3.

B.3 Remarks. There are a couple of possible extensions of this algorithm. For example, from $i = 1$ on, a cubic interpolation for ξ can be used to calculate a more accurate step size (see for example [104, 105]). Another extension involves satisfying the curvature condition (see [123, Chapter 3]). A third commonly used modification is to bound the new step length in relation to the old one, i. e., ω^{new} is forced to be located in the interval

$$\omega^{\text{new}} \in [\beta_1\,\omega, \beta_2\,\omega]$$

in every step, for two constants $0 < \beta_1 < \beta_2 < 1$, e. g., $\beta_1 = 0.1$, $\beta_2 = 0.5$. This is an important safeguarding strategy to avoid too small/large steps (see [104, Chapter 8] or [105, Chapter 3.1]). Note however, in many practical examples, all these modifications do not significantly enhance the robustness, so they are ignored in simple implementations.

C

Alternative solution concepts

To the best of the author's knowledge, the described and analysed solution approach is the first realisation of a complete hierarchical discretisation and solver concept for the optimal distributed control of nonstationary fluid flow. However, the method shares similarities with other solution strategies from the literature, in particular the 'Nonlinear multigrid strategy' and the 'Integral equation approach'. In particular the last one was used by numerous authors (see for example [11, 34, 71, 84, 85, 152]), although barely in a hierarchical style. Both concepts have advantages and disadvantages in comparison to the solver strategy proposed in Chapters 2 and 3. This appendix gives a brief introduction into these methods including a short discussion about the differences.

C.1. The nonlinear multigrid strategy

The general nonlinear multigrid strategy has been proposed, e. g., in [29] or [75, Chapter 9]. The idea of this scheme is to create a hierarchy of nonlinear problems which is exploited by a multigrid scheme similar to the linear case. However, all defects are created based on the original nonlinear operator instead of its linearised counterpart.

In [20, 21], Borzì described an adaption of the Full-Approximation-Scheme ('FAS') from [29] as solver for nonstationary, nonlinear optimal control problems. The model problem considered in there formally reads in a simplified form as follows. The usual space-time cylinder is denoted by $\mathcal{Q} = (0,T) \times \Omega$, with $T > 0$, $\Omega \subset \mathbb{R}^2$. For a function $z : \mathcal{Q} \to \mathbb{R}^2$, find $u : \mathcal{Q} \to \mathbb{R}^2$ with

$$J(y,u) := \frac{1}{2}||y-z||_{\mathcal{Q}}^2 + \frac{\alpha}{2}||u||_{\mathcal{Q}}^2 \quad \longrightarrow \quad \min!$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \Delta y + N(y) &= u \qquad \text{in } \mathcal{Q}, \\ y(0,\cdot) &= y^0 \qquad \text{in } \Omega, \end{aligned}$$

for a function $y : \mathcal{Q} \to \mathbb{R}^2$, a parameter $\alpha > 0$ and an initial condition $y^0 : \Omega \to \mathbb{R}^2$. This form is similar to the heat equation but involves an additional nonlinearity $N : L^2(\mathcal{Q}) \to \mathbb{R}^2$. Similar to Section 2.2 on page 23ff, the problem is reformulated with the usual Lagrange multiplier approach as KKT system. The control u is eliminated which results in a system of the form

$$\begin{aligned} y_t - \Delta y + N(y) &= -\frac{1}{\alpha}\lambda \qquad \text{in } \mathcal{Q}, \\ -\lambda_t - \Delta \lambda + DN^*(y)\lambda &= y - z \qquad \text{in } \mathcal{Q}, \end{aligned}$$

complemented by initial/end time/boundary conditions. This is a nonlinear system of the form

$$A(y, \lambda) = f$$

for an appropriate operator A and right-hand side f.

Borzì applies the finite difference scheme for the discretisation in space and the implicit Euler scheme in time on a hierarchy of space-time meshes similar to Section 3.1 on page 48ff. This results in a hierarchy of problems

$$A_l(y_l, \lambda_l) = f_l, \qquad l = 1, ..., \mathrm{NLMAX},$$

or in short

$$A_l(x_l) = f_l, \qquad l = 1, ..., \mathrm{NLMAX},$$

with A_l, y_l, λ_l, f_l being discrete counterparts to A, y, λ and f and $x_l = (y_l, \lambda_l)$. The variable l defines a level identifier and $\mathrm{NLMAX} \in \mathbb{N}$ is the maximum level of the hierarchy. This hierarchy is the starting point for the FAS scheme, which is basically a modification of the standard multigrid scheme for linear equations, see in particular [19, 22, 75, 79, 158].

Idea of the FAS scheme The idea of this scheme roughly sketched in the following; a more detailed description can be found, e. g., in [19]. For a given approximate solution x_l on level $l \in \{2, ..., \mathrm{NLMAX}\}$, there is to find an update g_l on level l such that

$$x_l^* := x_l + g_l$$

solves the nonlinear equation, i. e.,

$$A_l(x_l^*) = f_l \tag{C.1}$$

This necessitates smoothing and a coarse grid correction. For the coarse grid correction, equation (C.1) is reformulated as follows,

$$\begin{aligned} & A_l(x_l^*) = f_l, \\ \Leftrightarrow \qquad & A_l(x_l^*) - A_l(x_l) = \underbrace{f_l - A_l(x_l)}_{=:d_l}. \end{aligned}$$

This equation is formulated on the coarse grid using appropriate restriction/coarse grid projection operators. On level $l-1$, there is a x_{l-1} which satisfies the coarse grid problem

$$\begin{aligned} & A_{l-1}(x_{l-1}) - A_{l-1}(Ix_l) = \underbrace{Rd_l}_{=:d_{l-1}} \\ \Leftrightarrow \qquad & A_{l-1}(x_{l-1}) \;=\; d_{l-1} + A_{l-1}(Ix_l) \end{aligned} \tag{C.2}$$

Here, $R = R^{l-1}$ denotes the restriction from level l to $l-1$ and $I = I^{l-1}$ a coarse grid projection operator (injection) from level l to level $l-1$. Since by definition

$$f_l = d_l + A_l(x_l),$$

the right-hand side of (C.2),

$$\tilde{f}_{l-1} := d_{l-1} + A_{l-1}(Ix_l),$$

is an approximation to f_{l-1}. Consequently, x_{l-1} is an approximation to a solution x_{l-1}^* on level $l-1$ with

$$A_{l-1}(x_{l-1}^*) = f_{l-1}.$$

This solution can be written in the form

$$x_{l-1}^{*} = I x_l + g_{l-1} \tag{C.3}$$

for an appropriate update g_{l-1} on level $l-1$.

With the solution x_{l-1} of the coarse grid problem (C.2) at hand, motivated by (C.3), an approximate update on level $l-1$ is now defined as

$$g_{l-1} := x_{l-1} - I x_l,$$

and a new nonlinear solution on level l follows using a prolongation $P = P^{l-1}$ from level $l-1$ to level l by

$$x_l^{\text{new}} := x_l + P g_{l-1}$$

The prolongation and the coarse grid projection are defined as injection, the restriction as weighted mean in space and time similar to Section 3.4.3 on page 56f. For the smoothing, a nonlinear modification of the Gauß–Seidel method is applied which incorporates in each timestep the solution of the previous and next timestep. This approach shares the same idea as the FBGsSmoother algorithm proposed in Section 3.5 on page 65ff but focuses on the scalar case and would have to be formulated in a different way for being applied to saddle point problems or a space-discretisation with finite elements.

Discussion Comparing this solution technique to the approach proposed in this work, some similarities can be observed. Both approaches concentrate on a nonlinear system involving the primal and dual variables, eliminating the control. Furthermore, similarities can be seen in the smoothing operators and in the way, prolongation/restriction operators are defined. However, the above modification of the FAS scheme does not apply a Newton iteration but directly solves the nonlinear system in a multilevel fashion. Thus, superlinear convergence cannot be expected, but the convergence is still fast and level-independent. In fact, Borzì proved and illustrated this for the 1D case in [21]. For a comprehensive overview about the FAS scheme including a numerical comparison to Newton-Multigrid schemes concerning efficiency and robustness, the interested reader is referred to [79].

At the end it must be said that a realisation of this method in the context of optimal control for the nonstationary Navier–Stokes equations is not yet existing. A realisation and comparison to the proposed discretisation and solver strategy regarding efficiency and robustness is beyond the scope of this work.

C.2. The integral equation method

The integral equation method was originally developed by Hackbusch [72–74] for general elliptic problems. In 1997, Tröltzsch and Goldberg [140] extended the approach to the optimal control of the heat equation before Büttner [34] proved convergence for a space-time multigrid method built upon this approach. In the following, a brief overview of this method is given without going too much into detail. This should highlight the differences, advantages and disadvantages in comparison to the method in this work, as the integral equation method is frequently used by other authors and can be seen as an alternative.

To briefly present the integral equation method as analysed, e. g., in [34, 72–74, 140], for simplicity the optimal distributed control of the heat equation is considered. The

associated KKT system (ignoring boundary conditions and constraints for convenience) formally reads

$$\begin{aligned} y_t - \Delta y &= u && \text{in } \mathcal{Q}, \\ -\lambda_t - \Delta\lambda &= y - z && \text{in } \mathcal{Q}, \\ u &= -\frac{1}{\alpha}\lambda && \text{in } \mathcal{Q} \end{aligned}$$

and has to be interpreted in the weak sense, with $y, \lambda, u : \mathcal{Q} \to \mathbb{R}$, all smooth enough. An operator $\mathcal{P}$ representing the left-hand side is defined by $\mathcal{P} : y \mapsto (\frac{\partial}{\partial t}y - \Delta y)$; the adjoint $\mathcal{P}^*$ of its Fréchet derivative is given by $\mathcal{P}^* : \lambda \mapsto (-\frac{\partial}{\partial t}\lambda - \Delta\lambda)$. The KKT system therefore reads in a short form

$$\mathcal{P}y = u, \tag{C.4a}$$

$$\mathcal{P}^*\lambda = y - z, \tag{C.4b}$$

$$u = -\frac{1}{\alpha}\lambda. \tag{C.4c}$$

Under suitable assumptions (cf. [34, 72, 140]), $\mathcal{P}$ and $\mathcal{P}^*$ have inverse operators $\mathcal{P}^{-1}$ and $\mathcal{P}^{-*}$, respectively, such that the last equation (C.4c) for the control u can be reformulated by (C.4a) and (C.4b) to the fixed point equation

$$\begin{aligned} u &= -\frac{1}{\alpha}\mathcal{P}^{-*}\left(\mathcal{P}^{-1}u - z\right) \\ &= \left(-\frac{1}{\alpha}\mathcal{P}^{-*}\mathcal{P}^{-1}\right)u - \left(-\frac{1}{\alpha}\mathcal{P}^{-*}\right)z \\ &=: \mathcal{K}u + f. \end{aligned} \tag{C.5}$$

This equation is an *integral equation of Fredholm type* (cf. [72, 73, 140]), i. e., the above equation can be written in the form

$$u(w) = \int_{\mathcal{Q}} k(w,s)u(s)ds + f(w)$$

for some *kernel function* $k : \mathcal{Q}^2 \to \mathbb{R}$ with $w, s \in \mathcal{Q}$; one can now easily recognise where the name of the method comes from. In a compact form, this equation reads

$$(\mathcal{I} - \mathcal{K})u = f. \tag{C.6}$$

Solution algorithms From this integral equation, a fixed point iteration can be derived directly. Given a starting point $u_0 : \mathcal{Q} \to \mathbb{R}$, either the iteration

$$u^n := \mathcal{K}u^{n-1} + f, \qquad n \in \mathbb{N},$$

can be applied, similar to [72, 140], or a defect correction scheme for u can be formulated,

$$u^n := u^{n-1} + C^{-1}(\underbrace{f - u^{n-1} + \mathcal{K}u^{n-1}}_{=:d^n}), \tag{C.7}$$

with an appropriate preconditioning operator C. Alternatively, on the discrete level, (preconditioned variants of) GMRES, CG (if $(\mathcal{I}-\mathcal{K})$ is positive definite), BiCGStab or others can be used, as suggested in [34].

The fixed point iteration can be used to formulate a two-grid algorithm in space and time. At first, a constant number of iterations with an iterative solver on the space-time fine mesh forms a smoother. Afterwards, one iteration with the above fixed point method follows where the preconditioner C is replaced by a restriction–solution–prolongation operator by means of a coarser space-time grid. On the coarse mesh, an iterative solver can again be used. Recursively applying this technique for a hierarchy of space-time meshes results in a space-time multigrid for the control u.

C.1 Remarks. a) The core component of the whole algorithm is the operator $\mathcal{K}$, defined as $\mathcal{K} = -\frac{1}{\alpha}\mathcal{P}^{-*}\mathcal{P}^{-1}$. That means, for each defect $d^n = f - (\mathcal{I} - \mathcal{K})u^{n-1}$, one primal equation forward in time and one dual equation backward in time have to be solved. The philosophy behind this algorithm is an operator splitting: Similar to the FBSimSmoother smoother, this approach decouples the solution of the primal equation from the dual equation, which more or less allows black box simulation solvers to be used for the two time sweeps.

b) For nonlinear equations, the method can be embedded as a linear solver into a Newton iteration for the control u; this was introduced in [34, Section 6.3].

Discussion Both methods, the integral equation method as well as the space-time multigrid approach described in this work, are different solution methods for the same discrete KKT system. Therefore, computed solutions are the same. But while the integral equation method can be interpreted as a Schur complement approach (working in the control space due to the elimination of primal/dual variables), the space-time multigrid approach described in this work focuses on the 'opposite' variables: primal and dual variables are treated in a monolithic way, the control u is eliminated.

An advantage of the integral equation method is the reduced memory consumption as proper orthogonal decomposition ('POD') and checkpointing techniques can be applied. On the other hand, a disadvantage can be seen in the lack of appropriate choices for the preconditioner in (C.7) which is a crucial point for stability: The construction of a good preconditioner C as an approximation to $(\mathcal{I} - \mathcal{K})^{-1}$ is far from being trivial. To best of the author's knowledge, the usual choice is the Richardson preconditioner $C := \omega\mathcal{I}$ (with a damping parameter $\omega > 0$) which is known to be weak in many situations.

As indicated above, both methods can be used in a multigrid framework, and in [34], level-independent convergence of such an approach has been proven for the optimal control of the heat equation. However, to the best of the author's knowledge, the integral equation method has not yet been realised in a hierarchical context for the optimal control of the nonstationary Navier–Stokes equations. A realisation and a fair comparison between both approaches concerning robustness, memory consumption and convergence speed is still missing and beyond the scope of this work.

D

Modified Crank–Nicolson discretisations

The time discretisation of the space-time problem in Chapter 2.6 uses a special interpretation for the pressure and the right-hand side. Formally, the scheme interprets the pressure at time interval $[t_{n-1}, t_n]$ as point value at $t = t_{n-1+\theta} = (1-\theta)t_{n-1} + \theta t_n$. For $\theta = 1/2$, $p_{n-\frac{1}{2}}$ in the timestepping scheme is therefore interpreted as the pressure $p(\frac{t_{n-1}+t_n}{2})$, i. e., in the midpoint of the time interval. This fact is well accepted in the community, but to the best of the author's knowledge, it has not found its way to the literature. An exact analytical derivation of bounds for the time error is beyond the scope of this work. However, the following appendix derives a modified Crank–Nicolson scheme in a formal way to motivate this interpretation. For the underlying theory, the interested reader is referred to [8, 96, 131].

Similar to Section 1.7 on page 17, $\Omega \subset \mathbb{R}^{\dim}$ for $\dim \in \mathbb{N}$ is a domain with a boundary that is smooth enough, $T > 0$, $[0,T]$ a time interval of interest, $\mathcal{Q} = (0,T) \times \Omega$ a space-time cylinder and $\Sigma = (0,T) \times \partial\Omega$. The underlying spaces are defined as $U := L^2(\mathcal{Q})$, $V := H^{1,1}(\mathcal{Q})^{\dim}$, $Z_0 := \{q \in L^2(\Omega) : \int_\Omega q = 0\}$ and $Z := L^2(0,T;Z_0)$. In the literature, the nonstationary Stokes equations with homogeneous initial and boundary conditions, which is considered here for simplicity, is usually formulated as

$$\begin{aligned} y_t - \Delta y + \nabla p &= f && \text{in } Q, \\ -\operatorname{div} y &= 0 && \text{in } Q, \\ y(0,\cdot) &= 0 && \text{in } \Omega, \\ y &= 0 && \text{at } \Sigma, \end{aligned}$$

with $f \in U$, $y \in V$ and $p \in Z$. This should be understood in the weak sense on the space-time cylinder, i. e., for $y \in V_0 := V \cap L^2(0,T;H_0^1(\Omega)^{\dim})$,

$$\begin{aligned} (y_t, v)_{\mathcal{Q}} + (\nabla y, \nabla v)_{\mathcal{Q}} - (p, \operatorname{div} v)_{\mathcal{Q}} &= (f,v)_{\mathcal{Q}} && \text{for all } v \in L^2(0,T;H_0^1(\Omega)^{\dim}), \\ (-\operatorname{div} y, q)_{\mathcal{Q}} &= 0 && \text{for all } q \in L^2(\mathcal{Q}), \\ y(0,\cdot) &= 0 && \text{in } \Omega. \end{aligned}$$

Following [96, 131], a special Galerkin–Petrov discretisation is applied to this problem which allows to derive different formulations of the Crank–Nicolson method. For that purpose, subspaces of piecewise constant step functions are defined as follows: An equidistant time discretisation of $[0,T]$ is chosen with $N \in \mathbb{N}$ time intervals based on $0 = t_0 < t_1 < ... < t_N = T$, $k := T/N$ and $t_n = nk$ for $n = 0, ..., N$. The time intervals are denoted by $I_0 := \{t_0\}$, $I_n = (t_{n-1}, t_n]$. Then, subspaces of V_0 and Z with piecewise

constant step functions in time are given by

$$V_k := \{w \in V_0 : w(t,\cdot) = w_n \in H^1(\Omega)^{\dim} \text{ for almost all } t \in I_n\} \subset V_0,$$
$$Z_k := \{q \in Z : z(t,\cdot) = z_n \in L^2(\Omega) \text{ for almost all } t \in I_n\} \subset Z.$$

Using these spaces as test spaces, there holds in particular

$$\begin{aligned} (y_t, v)_{\mathcal{Q}} + (\nabla y, \nabla v)_{\mathcal{Q}} - (p, \operatorname{div} v)_{\mathcal{Q}} &= (f, v)_{\mathcal{Q}} && \text{for all } v \in V_k, \\ (-\operatorname{div} y, q)_{\mathcal{Q}} &= 0 && \text{for all } q \in Z_k, \\ y(0,\cdot) &= 0 && \text{in } \Omega, \end{aligned}$$

or in detail

$$\begin{aligned} \sum_{i=1}^{N} \int_{I_n} (y_t, v)_\Omega \, dt + \sum_{i=1}^{N} \int_{I_n} (\nabla y, \nabla v)_\Omega \, dt & \\ - \sum_{i=1}^{N} \int_{I_n} (p, \operatorname{div} v)_\Omega \, dt &= \sum_{i=1}^{N} \int_{I_n} (f, v)_\Omega \, dt && \text{for all } v \in V_k, \\ \sum_{i=1}^{N} \int_{I_n} (-\operatorname{div} y, q)_\Omega \, dt &= 0 && \text{for all } q \in Z_k, \\ y(0,\cdot) &= 0 && \text{in } \Omega. \end{aligned}$$

Due to the definition of the test spaces, there is $v(t,\cdot) = v_n \in V$ and $q(t,\cdot) = q_n \in Z$ for almost all $t \in I_n$. The above sum therefore decomposes into N independent time integrals

$$\begin{aligned} \int_{I_n} (y_t, v_n)_\Omega \, dt + \int_{I_n} (\nabla y, \nabla v_n)_\Omega \, dt & \\ - \int_{I_n} (p, \operatorname{div} v_n)_\Omega \, dt &= \int_{I_n} (f, v_n)_\Omega, dt && \text{for all } v_n \in V, \\ \int_{I_n} (-\operatorname{div} y, q_n)_\Omega \, dt &= 0 && \text{for all } q_n \in Z, \end{aligned}$$

for $n = 1, ..., N$. These time integrals are now approximated by using cubature in time. First, the functions y, f and p are assumed to be more regular in time; in particular, $y \in C(0,T;V)$, $f \in C(0,T;V^*)$ and $p \in C(0,T;Z)$ are assumed. Applying the trapezoidal rule to the velocity integrals, the midpoint rule to the pressure integral and approximating the time derivative by a central difference scheme in time, for all $v_n \in V$ and $q_n \in Z$ the approximation

$$\begin{aligned} (y_n - y_{n-1}, v_n)_\Omega + k \left[\frac{1}{2} (\nabla y_n, \nabla v_n)_\Omega + \frac{1}{2} (\nabla y_{n-1}, \nabla v_n)_\Omega \right] & \\ - k (p_{n-\frac{1}{2}}, \operatorname{div} v_n)_\Omega &= k \left[\frac{1}{2} (f_n, v_n)_{V^*,V} + \frac{1}{2} (f_{n-1}, v_n)_\Omega \right] \\ (-\operatorname{div} y_n, q_n)_\Omega &= 0 \\ y_0 &= 0 \end{aligned}$$

is obtained, with $y_n \in V$ and $p_{n-\frac{1}{2}} \in Z$ corresponding to $y(t_n) \in V$ and $p(\frac{t_n + t_{n-1}}{2}) \in Z$, respectively. It is well accepted that this interpretation approximates the continuous formulation up to an error $\mathcal{O}(k^2)$, thus being of second order in time and exact if y and p

are piecewise linear functions in time (with values in V and Z, respectively). Division by k and resorting unknown variables to the left-hand side (and ignoring the divergence/initial condition for simplicity) results in

$$\begin{aligned}&(y_n/k, v_n)_\Omega + \frac{1}{2}(\nabla y_n, \nabla v_n)_\Omega - (p_{n-\frac{1}{2}}, \operatorname{div} v_n)_\Omega \\ &= (y_{n-1}/k, v_n)_\Omega - \frac{1}{2}(\nabla y_{n-1}, \nabla v_n)_\Omega + \left[\frac{1}{2}(f_n, v_n)_\Omega + \frac{1}{2}(f_{n-1}, v_n)_{V^*,V}\right].\end{aligned}$$

Formally, the solution sets

$$(y_0, ..., y_n) \in V^{N+1}, \quad (p_{\frac{1}{2}}, ..., p_{N-\frac{1}{2}}) \in V^N$$

are interpreted as being the weak solutions (w. r. t. space) of the 'traditional' Crank–Nicolson scheme

$$\frac{y_n}{k} - \frac{1}{2}\Delta y_n + \nabla p_{n-\frac{1}{2}} = \frac{y_{n-1}}{k} + \frac{1}{2}\Delta y_{n-1} + \left[\frac{1}{2}f_n + \frac{1}{2}f_{n-1}\right] \tag{D.1}$$

with $(y_0, ..., y_n)$ approximating $(y(t_0), ..., y(t_N))$ and $(p_{\frac{1}{2}}, ..., p_{N-\frac{1}{2}})$ approximating $(p(\frac{t_1-t_0}{2}), ..., p(\frac{t_N-t_{N-1}}{2}))$.

Without reducing the order of the time discretisation, the midpoint rule can be used for the right-hand side as well. This results in

$$\begin{aligned}(y_n - y_{n-1}, v_n)_\Omega + k\left[\frac{1}{2}(\nabla y_n, \nabla v_n)_\Omega + \frac{1}{2}(\nabla y_{n-1}, \nabla v_n)_\Omega\right] \\ - k(p_{n-\frac{1}{2}}, \operatorname{div} v_n)_\Omega = k(f_{n-\frac{1}{2}}, v_n)_\Omega\end{aligned}$$

or equivalently,

$$\begin{aligned}&(y_n/k, v_n)_\Omega + \frac{1}{2}(\nabla y_n, \nabla v_n)_\Omega - (p_{n-\frac{1}{2}}, \operatorname{div} v_n)_\Omega \\ &\qquad = (y_{n-1}/k, v_n)_\Omega - \frac{1}{2}(\nabla y_{n-1}, \nabla v_n)_\Omega + (f_{n-\frac{1}{2}}, v_n)_\Omega,\end{aligned}$$

which uses the notation $f_{n-\frac{1}{2}} := f(\frac{t_n+t_{n-1}}{2})$. Formally, the solution sets

$$(y_0, ..., y_N) \in V^{N+1}, \quad (p_{\frac{1}{2}}, ..., p_{N-\frac{1}{2}}) \in V^N$$

are interpreted as the weak solutions (w. r. t. space) of the modified Crank–Nicolson scheme

$$\frac{y_n}{k} - \frac{1}{2}\Delta y_n + \nabla p_{n-\frac{1}{2}} = \frac{y_{n-1}}{k} + \frac{1}{2}\Delta y_{n-1} + f_{n-\frac{1}{2}} \tag{D.2}$$

with $(y_0, ..., y_n)$ and $(p_{\frac{1}{2}}, ..., p_{N-\frac{1}{2}})$ again approximating $(y(t_0), ..., y(t_N))$ and $(p(\frac{t_1-t_0}{2}), ..., p(\frac{t_N-t_{N-1}}{2}))$, respectively.

For both schemes, the approximation of the pressure can be interpreted as being located in the midpoints of each time interval. The modified scheme which also evaluates the right-hand side in the midpoints of the time intervals is the key for the Crank–Nicolson discretisation of the KKT system in Section 2.6, and the numerical examples in Section 6.3 confirm the second order approximation properties.

Interpreting p and f in the midpoints in time is crucial at different points of the algorithm if moving on to an optimal control problem on the space-time cylinder. On

the one hand, this influences the way prolongation and restriction operators are set up; with the correct interpolation, a faster convergence can be expected. On the other hand, the evaluation of different physical quantities is affected: For the evaluation of drag/lift coefficients for example, the common strategy is to interpret the pressure as being located in the endpoints of the time interval, similar to the velocity. Instead, interpreting the pressure as being located in the midpoints of the time intervals, the evaluation of the drag/lift coefficients can be done with a higher accuracy.

E

Parametric and nonparametric finite elements

For the discretisation in space with finite elements, it is known (see for example [26, 33]) that the choice of the finite element spaces for the velocity and the pressure is crucial for the stability. The pair Q_1/Q_1 is unstable for example and needs special stabilisation techniques (cf. [60, 95]) to avoid spurious oscillations in the solution. As a remedy, finite element spaces of different order for velocity and pressure can be chosen. However, this often leads to the choice of a nonconforming or discontinuous finite element space in the velocity and/or the pressure.

This work uses two common finite element pairs, which are both partially nonconforming and discontinuous, namely $\tilde{Q}_1/Q_0$ and Q_2/P_1^{disc}. The first pair, $\tilde{Q}_1/Q_0$, is a common choice of a first order discretisation (cf. [142]). While the definition of the Q_0 space for the pressure is rather straightforward, the definition of the $\tilde{Q}_1$ space is not unique. In particular for the nonconforming, integral mean value based variant of this family, which is preferred in this work and which is known to be the most stable variant (cf. [142–145]), a clear definition is missing in the literature, at least to the best of the author's knowledge. While in [109] a first description of this topic was given, the following chapter gives an extended introduction.

The other element pair Q_2/P_1^{disc}, which has been used and analysed concerning its accuracy in a simulation, e. g., in [124], is chosen as an example for a higher order discretisation. The velocity space is continuous and conforming. However, the pressure space is discontinuous as it was for the $\tilde{Q}_1/Q_0$ pair. This pair is expected to show a higher accuracy for smooth solutions than $\tilde{Q}_1/Q_0$ pair — at least if the accuracy of the time discretisation is high enough. The exact definition of the P_1^{disc} space (linear and discontinuous) is also part of the following appendix.

E.1. General terms and definitions

For describing the approach of parametric and nonparametric elements, first, some general facts and definitions about finite element functions are recalled which can be found, e. g., in [37, 38]. It is noted that the notation used here slightly differs from that used in previous chapters concerning the indices. The dimension of the underlying space is denoted by $\dim \in \mathbb{N}$, e. g., $\dim = 2$ or 3. For an arbitrary geometric element $K \in \Omega_h$ of a mesh $\Omega_h \subset \mathbb{R}^{\dim}$, a finite element is given by two sets. On the one hand, a finite dimensional space of *shape functions* on K is needed,

$$\Pi(K) = \operatorname{span}\{m_1, ..., m_n\},$$

with all $m_i : \mathbb{R}^{\dim} \to \mathbb{R}$ being linear independent and usually polynomial or rational. Furthermore, a set $\tau := \{\tau_1, ..., \tau_n\}$ of n linear independent, linear and continuous node

functionals $\tau_k : C^s(K) \to \mathbb{R}$ is necessary, $s \geq 0$. The set has to be unisolvent, i. e., for all combinations of α_1, ..., $\alpha_n \in \mathbb{R}$ there exists a unique function $p \in \Pi(K)$ such that

$$\tau_k(p) = \alpha_k, \qquad k = 1, ..., n.$$

A finite element function $f_h : \mathbb{R}^{\dim} \to \mathbb{R}$ is piecewise defined on the element K by

$$f_{h|K}(x) = p(x)$$

for a function $p(x) = \sum_{i=1}^{n} a_i m_i(x) \in \Pi(K)$, a_1, ...,$a_n \in \mathbb{R}$. To calculate the coefficients a_i for a set of values α_1, ..., α_n, a linear system for the a_i is formed,

$$\tau_k(p) = \tau_k(\sum_{i=1}^{n} a_i m_i) = \sum_{i=1}^{n} \tau_k(m_i) \cdot a_i = \alpha_k, \qquad k = 1, ..., n.$$

Obviously, the coefficients a_i are independent of K because of the linearity of the τ_k.

E.1 Example. For a quadrilateral element K, a_1,...,a_4 refer to the corner points, e_1,...,e_4 to the edges, M_1,...,M_4 to the midpoints of the edges and M to the centre of the element. Furthermore, $f \in C(\mathbb{R}^{\dim})$ is an arbitrary function with $f_{|K} \in C^1(K)$ for all elements $K \in \Omega_h$. The node functionals of the Q_1 space are defined as

$$\tau_{Q_1}(f) := \{f(a_1), f(a_2), f(a_3), f(a_4)\}.$$

The node functionals of the $\tilde{Q}_1$ space are defined as

$$\tau_{\tilde{Q}_1}(f) := \left\{ \frac{1}{|e_1|} \int_{e_1} f \, ds, \frac{1}{|e_2|} \int_{e_2} f \, ds, \frac{1}{|e_3|} \int_{e_3} f \, ds, \frac{1}{|e_4|} \int_{e_4} f \, ds \right\},$$

and the node functionals of the P_1^{disc} space as

$$\tau_{P_1^{\text{disc}}}(f) := \{f(M), \partial_x f(M), \partial_y f(M)\}.$$

The element P_1^{disc} on the other hand is an extension to the Q_0 space whose node functional is defined by

$$\tau_{Q_0}(f) := \{f(M)\}.$$

Finally, the Q_2 space is defined by the node functionals corresponding to the vertices, edge midpoints M_1,...,M_4 and the element centre,

$$\begin{aligned} \tau_{Q_2}(f) := \{&f(a_1), f(a_2), f(a_3), f(a_4), \\ &f(M_1), f(M_2), f(M_3), f(M_4), \\ &f(M)\}. \end{aligned}$$

From global to local basis functions

A finite element function f_h is globally defined as a linear combination $f_h = \sum_{i=1}^{N} f_i \varphi_i(x)$, of $N \in \mathbb{N}$ global basis functions $\{\varphi_i\}$, with $\{f_i\}$ representing the global degrees of freedom. Every φ_i has a local representation on the element K. There are exactly n functions in $\{\varphi_i\}$ such that $\text{supp}(\varphi_i) \cap K = K$. Without loss of generality,

$$\text{supp}(\varphi_i) \cap K = K \qquad \text{for } i = 1, ..., n$$

can be assumed. Every φ_i is locally a linear combination of the shape functions,

$$\varphi_{i|K} = \sum_{j=1}^{n} a_{ij} m_j(x) =: p_i \in \Pi(K),$$

and therefore, for $x \in K$, there is

$$f_{h|K}(x) = \sum_{i=1}^{N} f_i \varphi_{i|K}(x) = \sum_{i=1}^{n} f_i \varphi_{i|K}(x) = \sum_{i=1}^{n} f_i p_i(x). \tag{E.1}$$

The p_i are called a 'local basis functions'. They are uniquely defined by the relationship

$$\tau_k(p_i) = \delta_{ki}, \qquad k, i = 1, ..., n.$$

with δ_{ij} denoting the Kronecker delta. The difference between parametric and nonparametric elements is in the definition of the shape functions.

E.2. Parametric elements

For *parametric* finite elements the shape functions are defined by means of a reference element and a proper mapping from the reference element $\hat{K} \subset \mathbb{R}^{\dim}$ to the real element K. With $\{\hat{m}_i : \mathbb{R}^{\dim} \to \mathbb{R}, i = 1, ..., n\}$, a set of $n \in \mathbb{N}$ linear independent (and usually polynomial) shape functions is defined on $\hat{K}$. A bijective mapping $\sigma = \sigma_K : \hat{K} \to K$ maps from the reference to the real element K. The shape functions on the real element are defined as

$$m_i(x) := \hat{m}_i \circ \sigma^{-1}(x) = \hat{m}_i(\hat{x}), \qquad i = 1, ..., n$$

with $\hat{x} = \sigma_K^{-1}(x)$. It is remarked that in general due to the mapping, these are rational.

Furthermore, the existence of a set $\hat{\tau} := \{\hat{\tau}_1, ..., \hat{\tau}_n\}$ of d linear independent, linear and continuous node functionals $\hat{\tau}_k : C^s(\hat{K}) \to \mathbb{R}$ is assumed, $s \geq 0$, which is unisolvent on $\hat{K}$. The corresponding node functionals for an arbitrary element K are defined by means of the mapping σ,

$$\tau_k(p) := \hat{\tau}_k(p \circ \sigma^{-1}), \qquad p \in C^s(\mathbb{R}^{\dim}),\ k = 1, ..., n.$$

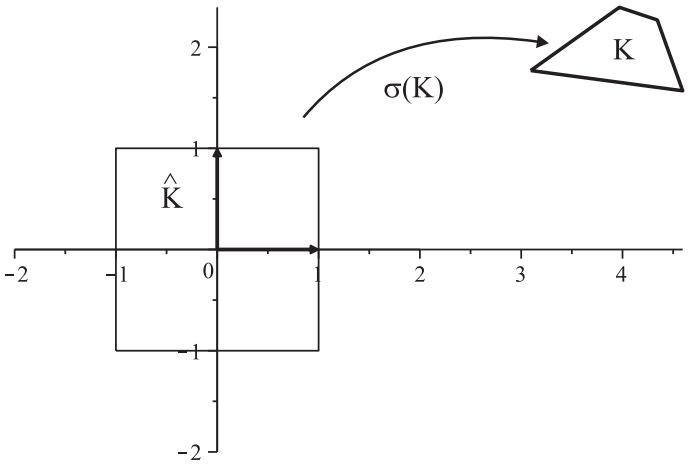

Figure E.1: Mapping σ from a reference to a real quadrilateral element.

E.2 Example. As an example, the reference quadrilateral in 2D is considered, $\hat{K} := [-1,1]^2$, see Figure E.1. For $z = (z_1, z_2)^\mathsf{T} \in \hat{K}$, the Q_1-element is given by the four shape functions

$$\hat{m}_0(z) = 1, \quad \hat{m}_1(z) = z_1, \quad \hat{m}_2(z) = z_2, \quad \hat{m}_3(z) = z_1 z_2.$$

For an arbitrary element $K \in \Omega_h$ of the mesh, there exists a bilinear mapping $\sigma : \mathbb{R}^2 \to \mathbb{R}^2$,

$$\sigma(z) = \sigma_K(z) = \begin{pmatrix} s_0 + s_1 z_1 + s_2 z_2 + s_3 z_1 z_2 \\ t_0 + t_1 z_1 + t_2 z_2 + t_3 z_1 z_2 \end{pmatrix}$$

which, on under mild conditions on the cell, bijectively maps the corners of $\hat{K}$ to the corners of K, s_j, $t_j \in \mathbb{R}$. The shape functions on the real element are then defined as

$$m_i(x) := \hat{m}_i(\hat{x}), \qquad i = 1, ..., 4.$$

with $\hat{x} = \sigma^{-1}(x)$, $x \in \mathbb{R}^2$.

Definition of local basis functions for parametric elements

As stated above, the set of local basis functions for an element K is uniquely defined by the relationship

$$\tau_k(p_i) = \delta_{ki}, \qquad k, i = 1, ..., n.$$

Because of the linearity, this leads to

$$\delta_{ki} \;=\; \tau_k(p_i) \;=\; \sum_{j=1}^{n} a_{ij} \tau_k(m_j) \;=\; \sum_{j=1}^{n} a_{ij} \hat{\tau}_k(\hat{m}_j) \;=\; \hat{\tau}_k(\hat{p}_i)$$

for the local basis functions $\hat{p}_i : \hat{K} \to \mathbb{R}$, $\hat{p}_i(\hat{x}) := \sum_{j=1}^{n} a_{ij} \hat{m}_j(\hat{x})$ on the reference element $\hat{K}$. These local basis functions are obviously independent of the mesh. The $\{a_{ij}\}$ can be computed once in advance and used for all elements K. For $\hat{x} \in \hat{K}$ and $x = \sigma(\hat{x})$ there holds

$$p_i(x) = p_i \circ \sigma(\hat{x}) = \hat{p}_i(\hat{x}).$$

As soon as the coefficients are known, (E.1) can be rewritten in terms of $\hat{x}$,

$$f_{h|K}(x) = \sum_{i=1}^{n} f_i p_i(x) = \sum_{i=1}^{n} f_i \hat{p}_i(\hat{x}).$$

For calculating derivatives and integrals (e. g., in the assembly of matrices and vectors), $\hat{x} = \sigma^{-1}(x)$ in combination with the chain rule has to be used. The evaluation of $\hat{p}_i(\hat{x})$ is particularly cheap if $\hat{x}$ is known in advance; e. g., during in the assembly of matrices and vectors, $\hat{x}$ represents coordinates of cubature points on the reference element which are often the same for all K.

All finite elements built upon this approach are called *parametric elements* as the mapping σ maps the *parameter space* $\hat{K}$ to the space of real world coordinates. Typical properties of parametric elements are, e. g., their clear and straightforward definition and the high efficiency in the assembly of matrices and vectors as the local basis functions can be computed in advance on a reference element.

E.3. Nonparametric elements

Nonparametric elements are harder to find in the literature (see for example [142–145]) and often, a clear definition is missing. The following paragraphs give a closer introduction about what nonparametric elements are and how they are defined.

Again, $\hat{K}$ denotes a reference element, $K \in \Omega_h$ an arbitrary element of a mesh Ω_h, $\{\hat{m}_i : \mathbb{R}^{\dim} \to \mathbb{R},\, i = 1, ..., n\}$ a set of shape functions on $\hat{K}$ and $\sigma : \hat{K} \to K$ a bijective mapping from the reference to the real element. For very high angles in the real element K (up to nonconvex elements), the mapping σ can degenerate (see for example [162]). As a consequence, the underlying finite element space loses the property of being unisolvent and therefore, linear systems cannot be computed anymore.

Another restriction is crucial in particular for CFD applications: The possible loss of approximation properties on perturbed meshes. For example, an efficient low order approximation of the velocity and pressure in CFD is the $\tilde{Q}_1$ finite element space for the velocity and the Q_0 space for the pressure. This element pair is based on rectangular elements and fulfils the discrete LBB condition. In [145], Turek et al. analysed the influence of the shape of the underlying quadrilaterals to the approximation properties of the discretisation. If the parametric variant of $\tilde{Q}_1$ was used, it was shown that the approximation order suffers if the elements do not fulfil a special 'uniformity condition'. For strongly perturbed meshes, the element pair loses its approximation properties, thus rendering it unsuitable for being used on general meshes obtained, e. g., by certain mesh adaption methods.

To cope with these problems, shape functions are directly defined on the real element instead of on the reference element. Finite elements defined by this approach are called *nonparametric elements* as they use real world coordinates for being defined. There are different methods available how to define local polynomials in the real world space, and in the following, a quadrilateral 2D mesh is used to illustrate how such elements can be defined. The approach is of course much more general, it can be adapted to 2D triangular meshes as well as to 3D tetrahedral or hexahedral meshes and others. Usually, each parametric element can be converted to a nonparametric element by using the corresponding steps.

E.3.1. Local coordinate systems

Nonparametric elements are in most cases defined by using a 'local' coordinate system for each element $K \in \Omega_h$ upon which shape functions are defined. For that purpose, a definition of local coordinate vectors is needed. Usually, this is carried out by connecting opposite midpoints (of edges in 2D or face midpoints in 3D) which leads to a set of vectors $\vec{v}_1, ..., \vec{v}_d$, $\vec{v}_i = \vec{v}_i(K)$ for the element K (see Figure E.2), $i = 1, ..., \dim$. The most common approach to define a local coordinate system based on these vectors is by a linear mapping as follows.

The linear mapping approach

In the linear mapping approach, a linear mapping from the real world coordinate system to an intermediate reference element is used. A pure affine linear mapping $\sigma : \hat{K} \to \mathbb{R}^{\dim}$ is defined by

$$\sigma(z) = Az + b, \quad z \in \hat{K}, \tag{E.2}$$

for $A \in \mathbb{R}^{\dim \times \dim}$ and $b \in \mathbb{R}^{\dim}$. From this definition follows $\sigma^{-1}(x) = A^{-1}(x - b)$, i. e., the inverse σ^{-1} if σ is linear and explicitly known. The mapping $\sigma : \hat{K} \to \mathbb{R}^{\dim}$ can be

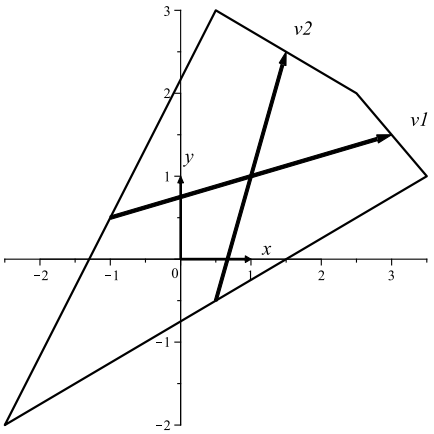

Figure E.2: Definition of a local coordinate system on an arbitrary element.

formulated using the vectors $\{\vec{v}_i\}$ and the midpoint M of the element K:

$$\sigma(z) = Az + b := \frac{1}{2} \begin{pmatrix} v_1^1 & v_2^1 & \dots \\ v_1^2 & v_2^2 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} z + M.$$

It is noted that, e. g., for $\dim = 2$, A is a 2×2 system, so A^{-1} can easily be computed. Furthermore, the mapping σ does not map $\hat{K}$ to K; instead, $\sigma(\hat{K})$ is a parallelogram where the midpoint of the reference element and its edge midpoints (or face midpoints in 3D, resp.) are mapped onto the element/edge(/face) midpoints of K (see Figure E.3). The image $\tilde{K} := \sigma^{-1}(K)$ is a deformed reference element (see Figure E.4).

The representations of the shape functions in the coordinates of the real element K are then defined as

$$m_i(x) := m_i(K, x) := \hat{m}_i \circ \sigma^{-1}(x).$$

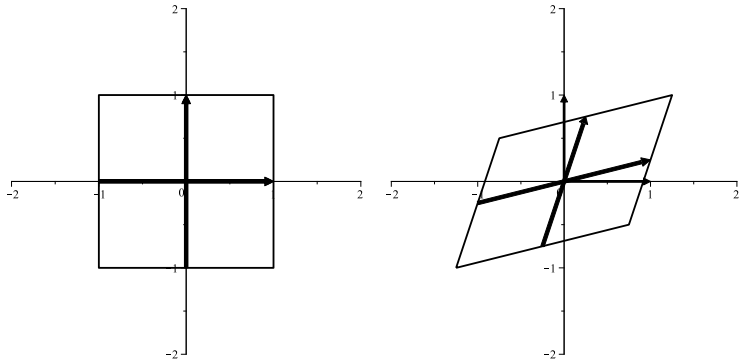

Figure E.3: Mapping of the quadrilateral reference element $\hat{K}$ (left). Right: $\sigma(\hat{K})$. Example for $v_1 = \begin{pmatrix} 1 \\ 1/4 \end{pmatrix}$ and $v_2 = \begin{pmatrix} 1/4 \\ 3/4 \end{pmatrix}$.

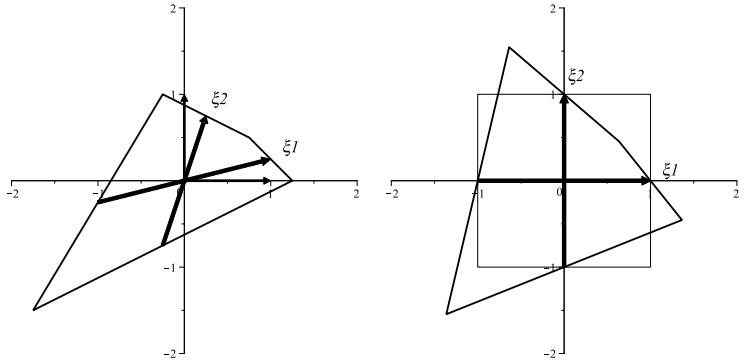

Figure E.4: Effect of the inverse mapping. Left: An arbitrary quadrilateral element in 2D with centre in $(0,0)$. Right: $\sigma^{-1}(K)$ and the reference element.

E.3.2. Definition of local basis functions for nonparametric elements

As stated above, the set of local basis functions for an element K is uniquely defined by the relationship

$$\tau_k(p_i) = \delta_{ki}, \qquad k,i = 1,...,n.$$

Because of the linearity, this leads to the linear system

$$\delta_{ki} \;=\; \tau_k(p_i) \;=\; \sum_{j=1}^{n} a_{ij}\tau_k(m_j).$$

In contrast to parametric finite elements, it is not possible to precompute the coefficients $\{a_{ij}\}$ in advance, they are usually different for every K. Therefore at this point, a small local matrix has to be inverted to compute these coefficients, which follows from

$$VA = I \qquad \Leftrightarrow \qquad A = V^{-1}$$

with I being the identity matrix and

$$V = \begin{pmatrix} \tau_1(m_1) & \tau_1(m_2) & ... \\ \tau_2(m_1) & \tau_2(m_2) & ... \\ ... & ... & \ddots \end{pmatrix}, \qquad A = \begin{pmatrix} a_{11} & a_{21} & ... \\ a_{12} & a_{22} & ... \\ ... & ... & \ddots \end{pmatrix}.$$

By calculating $A = V^{-1}$, the coefficients a_{ij} of the local basis functions are obtained, and using the f_i from above, the representation (E.1) can be written as

$$f_{h|K}(x) = \sum_{i=1}^{n} f_i p_i(x) = \sum_{i=1}^{n} f_i \sum_{j=1}^{n} a_{ij} m_j(x).$$

The functions $m_j(\cdot)$ in this sum can usually explicitly be calculated using their definition and the fact that σ^{-1} is linear.

E.3 Example. a) For the integral mean value based variant of the $\tilde{Q}_1$ in 2D, the shape functions in the new coordinate systems on the reference element are given by

$$\hat{m}_1(z) = 1,\ \hat{m}_2(z) = z_1,\ \hat{m}_3(z) = z_2,\ \hat{m}_4(z) = z_1^2 - z_2^2$$

for $z = (z_1, z_2)^\mathsf{T} \in \hat{K}$. Using the linear mapping approach and assuming $a_i =: (a_i^x, a_i^y)$ for the corner points of the element gives

$$\begin{aligned}\sigma(z) &= \frac{1}{4}\begin{pmatrix} a_2^x + a_3^x - a_4^x - a_1^x & a_3^x + a_4^x - a_1^x - a_2^x \\ a_2^y + a_3^y - a_4^y - a_1^y & a_3^y + a_4^y - a_1^y - a_2^y \end{pmatrix}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} a_1^x + a_2^x + a_3^x + a_4^x \\ a_1^y + a_2^y + a_3^y + a_4^y \end{pmatrix} \\ &= Bz + b,\end{aligned}$$

and for $x \in \mathbb{R}^2$, $z = z(x) = B^{-1}(x - b)$ can directly be evaluated. The shape functions on the real element follow as

$$\begin{aligned} m_1(x) &= 1, & m_2(x) &= z_1(x), \\ m_3(x) &= z_2(x), & m_4(x) &= z_1(x)^2 - z_2(x)^2. \end{aligned}$$

After computing the matrix $A = V^{-1}$ (where, e. g., the 2-point Gauss formula is applied for the boundary integrals on the element edges to compute the $\tau_k(m_j)$ in real-world coordinates for V), the values of the four basis functions in x on the real element are computable by $p_i(x) = \sum_{j=1}^4 a_{ij} m_j(x)$, $i = 1, ..., 4$.

b) In case of the midpoint-based variant $\tilde{Q}_1^{\mathrm{MP}}$ of $\tilde{Q}_1$ in combination with the linear-mapping approach, the cost intensive computation of the matrix $A = V^{-1}$ can be avoided. This variant of $\tilde{Q}_1$ uses the node functionals

$$\tau(f) := \tau_{\tilde{Q}_1^{\mathrm{MP}}}(f) := \{f(M_1), f(M_2), f(M_3), f(M_4)\}$$

with $\{M_i\}$ denoting again the midpoints of the edges e_1,...,e_4 of an element K. The shape functions are again

$$\hat{m}_1(z) = 1,\ \hat{m}_2(z) = z_1,\ \hat{m}_3(z) = z_2,\ \hat{m}_4(z) = z_1^2 - z_2^2.$$

A linear mapping $\sigma : \hat{K} \to K$, $\sigma(z) = Az + b$ as in (E.2) maps the edge midpoints of the reference element to $\{M_i\}$, i. e.,

$$\begin{aligned} \sigma(0, -1) &= M_1, & \sigma(1, 0) &= M_2, \\ \sigma(0, 1) &= M_3, & \sigma(-1, 0) &= M_4. \end{aligned}$$

Therefore, $\sigma^{-1}(M_i)$ and $\hat{m}_k(\sigma^{-1}(M_j))$ are known in advance, independent of the actual element. This leads to the basis functions

$$\tau_j(p_i) = p_i(M_j) = \sum_{k=1}^4 a_{ik} m_k(M_j) = \sum_{k=1}^4 a_{ik} \hat{m}_k(\sigma^{-1}(M_j)),$$

$$\Rightarrow \quad A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 \end{pmatrix}^{-1} = \frac{1}{4}\begin{pmatrix} 1 & 0 & -2 & -1 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & -1 \\ 1 & -2 & 0 & -1 \end{pmatrix},$$

$$\begin{aligned} p_1(x) &= \frac{1}{4}(m_1(x) - 2m_3(x) - m_4(x)) &&= \frac{1}{4}(1 - 2\hat{m}_3 - \hat{m}_4) \circ \sigma^{-1}(x), \\ p_2(x) &= \frac{1}{4}(m_1(x) + 2m_2(x) + m_4(x)) &&= \frac{1}{4}(1 + 2\hat{m}_2 + \hat{m}_4) \circ \sigma^{-1}(x), \\ p_3(x) &= \frac{1}{4}(m_1(x) + 2m_3(x) - m_4(x)) &&= \frac{1}{4}(1 + 2\hat{m}_3 - \hat{m}_4) \circ \sigma^{-1}(x), \\ p_4(x) &= \frac{1}{4}(m_1(x) - 2m_2(x) + m_4(x)) &&= \frac{1}{4}(1 - 2\hat{m}_2 + \hat{m}_4) \circ \sigma^{-1}(x). \end{aligned}$$

F

List of Symbols

General notations:

IE	Implicit Euler time discretisation scheme
CN	Crank–Nicolson time discretisation scheme
$\mathbb{R}$	Set of real numbers
$\mathbb{N}$	Natural numbers
$\mathbb{N}_0$	Natural numbers including zero
δ_{ij}	$= 1$ if $i = j$, $= 0$ otherwise; Kronecker delta
dim	$= 2$, or $= 3$, dimension of the underlying space
Ω	Domain, bounded open set in $\mathbb{R}^{\text{dim}}$, dim $= 2$ or $= 3$
Γ	$= \partial\Omega$, domain boundary
Γ_d	Part of the domain boundary corresponding to Dirichlet boundary conditions
Γ_n	Part of the domain boundary corresponding to do-nothing boundary conditions
η	Outer unit normal vector of Ω
T	Maximum time on a time interval $[0, T]$
$\mathcal{Q}$	$= (0, T) \times \Omega$, space-time cylinder
Σ	$= (0, T) \times \Gamma$, boundary of the space cylinder in time
$\mathcal{O}(\cdot)$	Landau symbol, asymptotic bound
$(\cdot, \cdot)$	Scalar product
$\|\| \cdot \|\|$	Norm
X^*	Dual space of a Hilbert space X
$(\cdot, \cdot)_{X^*,X}$	Dual pairing in a Hilbert space $\mathcal{X}$
$C(\Omega)$	Space of continuous functions on Ω
$L^2(\Omega)$	Space of square integrable functions on Ω
$(\cdot, \cdot)_\Omega$	$= (\cdot, \cdot)_{L^2(\Omega)}$, L^2 scalar product, $(v, w)_\Omega = \int_\Omega vw\, dx$
$\|\| \cdot \|\|_\Omega$	$= \|\| \cdot \|\|_{L^2(\Omega)}$, Norm associated with $(\cdot, \cdot)_\Omega$.

$H^1(\Omega)$	Space of square integrable functions with first weakly derivatives
$H_0^1(\Omega)$	Subspace of $H^1(\Omega)$ with weak zero boundary conditions
$L^2(0,T;X)$	Standard space of abstract functions $v : (0,T) \to X$, square-integrable in time
$H^1(0,T;X)$	$= \{y \in L^2(0,T;X) \,\vert\, y_t \in L^2(0,T;X)\}$.
$L^2(\mathcal{Q})$	Space of square integrable functions on Ω
$H^{1,1}(\mathcal{Q})$	$= L^2(0,T;H^1(\Omega)) \cap H^1(0,T;L^2(\Omega))$
$H_0^{1,1}(\mathcal{Q})$	$= H^1(0,T;L^2(\Omega)) \cap L^2(0,T;H_0^1(\Omega))$
$L^2(\mathcal{Q})^n$	$= L^2(0,T;(L^2(\Omega))^n)$
$H^{1,1}(\mathcal{Q})^n$	$= L^2(0,T;(H^1(\Omega))^n) \cap H^1(0,T;(L^2(\Omega))^n)$
$H_0^{1,1}(\mathcal{Q})^n$	$= H^1(0,T;(L^2(\Omega))^n) \cap L^2(0,T;(H_0^1(\Omega))^n)$
$(\cdot,\cdot)_{L^2(\mathcal{Q})}$	Scalar product in $L^2(\mathcal{Q})$
$\|\cdot\|_{L^2(\mathcal{Q})}$	Norm in $L^2(\mathcal{Q})$
$(\cdot,\cdot)_{(L^2(\mathcal{Q}))^n}$	Scalar product in $(L^2(\mathcal{Q}))^n$
$\|\cdot\|_{(L^2(\mathcal{Q}))^n}$	Norm in $(L^2(\mathcal{Q}))^n$
$(\cdot,\cdot)_{\mathcal{Q}}$	$= (u,v)_{L^2(\mathcal{Q})}$ or $= (u,v)_{(L^2(\mathcal{Q}))^n}$, depending on the context
$(\cdot,\cdot)_{\mathcal{Q}}$	Norm associated with $(\cdot,\cdot)_{\mathcal{Q}}$
∇y	$= (\partial_{x_1} y, \partial_{x_2} y, ...)^\mathsf{T}$, gradient of a scalar function; for functions on $\mathcal{Q}$: applied to the spatial component
Δy	$= \nabla \cdot (\nabla y) = \partial_{x_1}^2 y + \partial_{x_2}^2 y + ...$, Laplacian of a scalar function; for vector valued functions: applied to every component; for functions on $\mathcal{Q}$: applied to the spatial component
$\operatorname{div} y$	$= \partial_{x_1} y_1 + \partial_{x_2} y_2 + ...$, divergence of a vector function; for functions on $\mathcal{Q}$: applied to the spatial component
$(\cdot)^\mathsf{T}$	Transpose (of a matrix, a vector,...)
$\nabla^2(\cdot)$	Hessian of an operator
∂_t	Time derivative; also denoted with subscript t, e. g., $y_t = \partial_t y$
∂_{tt}	Second time derivative; also denoted with subscript tt, e. g., $y_{tt} = \partial_{tt} y$

Notations regarding continuous KKT systems:

V	$= H^{1,1}(\Omega)$ in case of the heat equation, $= H^{1,1}(\Omega)^{\dim}$ in case of the Stokes/Navier–Stokes equations
V_0	$= V \cap L^2(0,T;H_0^1(\Omega))$ in case of the heat equation, $= V \cap L^2(0,T;H_0^1(\Omega)^{\dim})$ in case of the Stokes/Navier–Stokes equations
Z_0	$= \{q \in L^2(\Omega) : \int_\Omega q = 0\}$
Z	$= L^2(0,T;Z_0)$

y^0	Initial condition
y	Primal velocity or temperature
λ	Dual velocity or temperature
p	Primal pressure
ξ	Dual pressure
ν	> 0, viscosity constant

$J(\cdot)$	Target functional, to be minimised
$\mathcal{H}$	Operator representing the continuous primal equation
$\mathcal{X}^*$	Adjoint of an operator $\mathcal{X}$, $(x, \mathcal{X}y) = (\mathcal{X}^*x, y)$
$D\mathcal{H}$	Fréchet derivative of $\mathcal{H}$
$D\mathcal{H}^*$	Operator representing the continuous dual equation
$L(\cdot)$	Lagrange functional associated with $J(\cdot)$

Notations regarding a time discretisation:

N	Number of time intervals
k	$= 1/N$, length of each time interval
$f = (f_1, f_2, ...)$	Space-time right-hand side vector

$\mathcal{A}$	Laplace or Stokes operator, $\mathcal{A}v = -\Delta v$ or $\mathcal{A}v = -\nu\Delta v$
$\mathcal{I}$	Identity operator, $\mathcal{I}v = v$
$\mathcal{G} = \nabla$	Gradient operator, $\mathcal{G}q = \nabla q$
$\mathcal{D} = -\operatorname{div}$	Divergence operator, $\mathcal{D}v = -\operatorname{div} v$

Notations regarding a time discretisation with the implicit Euler scheme:

x_n	n-th component of a vector x
x	Discrete primal solution, $= (y_0, y_1, ..., y_N)$ for the heat equation, $= (y_0, p_0, y_1, p_1, ..., y_N, p_N)$ for the Stokes/Navier–Stokes equations
ψ	Discrete dual solution, $= (\lambda_0, \lambda_1, ..., \lambda_N)$ for the heat equation, $= (\lambda_0, \xi_0, \lambda_1, \xi_1, ..., \lambda_N, \xi_N)$ for the Stokes/Navier–Stokes equations
y	$= (y_0, y_1, ..., y_N)$, discrete primal temperature/velocity
λ	$= (\lambda_0, \lambda_1, ..., \lambda_N)$, discrete dual temperature/velocity
p	$= (p_0, p_1, ..., p_N)$, discrete primal pressure
ξ	$= (\xi_0, \xi_1, ..., \xi_N)$, discrete dual pressure
$w = (w_1, w_2, ...)$	Space-time solution vector, $w_n = (y_n, \lambda_n)$ for the heat equation,

$w_n = (y_n, \lambda_n, p_n, \xi_n)$ for the Stokes/Navier–Stokes equations

$X_k = (V \times Z)^{N+1}$

$\mathcal{K}_n = (y_n \nabla(\cdot))$, convection operator
$\overline{\mathcal{K}}_n = ((\cdot)\nabla) y_n$, second convection operator
$\mathcal{C}_n = \mathcal{A} + \mathcal{K}(y_n)$
$= -\nu\Delta(\cdot) + (y_n \nabla(\cdot))$
$\mathcal{N}_n = \mathcal{A} + \mathcal{K}(y_n) + \overline{\mathcal{K}}(y_n)$
$= -\nu\Delta(\cdot) + (y_n \nabla(\cdot)) + ((\cdot)\nabla y_n)$

$\mathcal{H}^k = \mathcal{H}^k(x)$ Space-time matrix, primal equation; discrete counterpart to $\mathcal{H}$
$D\mathcal{H}^k = D\mathcal{H}^k(x)$ Space-time matrix; discrete counterpart to $D\mathcal{H}$
$D\mathcal{H}^{k,*} = D\mathcal{H}^{k,*}(x)$ Space-time matrix, dual equation; discrete counterpart to $D\mathcal{H}^*$
$\mathbf{G} = \mathbf{G}(w)$ Space-time matrix that combines primal and dual equation
$\mathbf{F} = \mathbf{F}(w)$ Fréchet derivative matrix of $\mathbf{G}$
$\mathbf{G}_n$ n-th diagonal matrix of $\mathbf{G}$
$\mathbf{F}_n$ n-th diagonal matrix of $\mathbf{F}$
$\breve{\mathbf{I}}_n$ First lower offdiagonal in $\mathbf{F}$ and $\mathbf{G}$, row n
$\hat{\mathbf{I}}_n$ First upper offdiagonal in $\mathbf{F}$ and $\mathbf{G}$, row n

Notations regarding a time discretisation with the Crank–Nicolson scheme:

$t_{n-1+\theta} = \theta t_n + (1-\theta) t_{n-1}$
$x_{n-1+\theta}$ n-th component of a space-vector x using a time discretisation with the one-step θ-scheme; interpreted as being located at $t_{n-1+\theta}$
x Discrete primal solution;
$= (y_0, y_1, ..., y_N)$ for the heat equation,
$= (y_0, p_{-1+\theta}, y_1, p_\theta, ..., y_N, p_{N-1+\theta})$ for the Stokes/Navier–Stokes equations
ψ Discrete dual solution,
$= (\lambda_{-1+\theta}, \lambda_\theta, ..., \lambda_{N-1+\theta})$ for the heat equation,
$= (\lambda_{-1+\theta}, \xi_0, \lambda_\theta, \xi_1, ..., \lambda_{N-1+\theta}, \xi_N)$ for the Stokes/Navier–Stokes equations
$y = (y_0, y_1, ..., y_N)$, discrete primal temperature/velocity
$\lambda = (\lambda_{-1+\theta}, \lambda_\theta, ..., \lambda_{N-1+\theta})$, discrete dual temperature/velocity
$p = (p_{-1+\theta}, p_\theta, ..., p_{N-1+\theta})$, discrete primal pressure
$\xi = (\xi_0, \xi_1, ..., \xi_N)$, discrete dual pressure
$w = (w_1, w_2, ...)$ Space-time solution vector,
$w_n = (y_n, \lambda_{n-1+\theta})$ for the heat equation,
$w_n = (y_n, \lambda_{n-1+\theta}, p_{n-1+\theta}, \xi_n)$ for the Stokes/Navier–Stokes equations

$$
\begin{aligned}
X_k &= (V \times Z)^{N+1} \\
\\
\mathcal{A}^\tau &= -\tau\Delta(\cdot) \\
\overline{\mathcal{K}}^*(y_n) &= (\nabla y)^\mathsf{T}(\cdot) \\
\mathcal{C}_n^\tau &= \tau(\mathcal{A} + \mathcal{K}(y_n)) \\
\mathcal{N}_n^\tau = \mathcal{N}^\tau(y_n) &= \tau(\mathcal{A} + \mathcal{K}(y_n) + \overline{\mathcal{K}}(y_n)) \\
\mathcal{N}_n^{\tau,*} = \mathcal{N}^{\tau,*}(y_n) &= \tau(\mathcal{A} - \mathcal{K}(y_n) + \overline{\mathcal{K}}^*(y_n)) \\
\mathcal{R}_n^\tau = \mathcal{R}^\tau(\lambda_{n-1+\theta}) &= \tau\big(- ((\cdot)\nabla)\lambda_{n-1+\theta} + (\nabla(\cdot))^\mathsf{T}\lambda_{n-1+\theta}\big)
\end{aligned}
$$

$\mathcal{H}^k = \mathcal{H}_\theta^k(x)$	Space-time matrix, primal equation; discrete counterpart to $\mathcal{H}$
$D\mathcal{H}^k = D\mathcal{H}_\theta^k(x)$	Space-time matrix; discrete counterpart to $D\mathcal{H}$
$D\mathcal{H}^{k,*} = D\mathcal{H}_\theta^{k,*}(x)$	Space-time matrix, dual equation; discrete counterpart to $D\mathcal{H}^*$
$\mathbf{G} = \mathbf{G}^\theta(w)$	Space-time matrix that combines primal and dual equation
$\mathbf{F} = \mathbf{F}^\theta(w)$	Fréchet derivative matrix of $\mathcal{G}$
$\mathbf{G}_n$	n-th diagonal matrix of $\mathbf{G}$
$\mathbf{F}_n$	n-th diagonal matrix of $\mathbf{F}$
$\breve{\mathbf{I}}_n$	First lower offdiagonal in $\mathbf{G}$, row n
$\hat{\mathbf{I}}_n$	First upper offdiagonal in $\mathbf{G}$, row n
$\breve{\mathbf{J}}_n$	First lower offdiagonal in $\mathbf{F}$, row n
$\hat{\mathbf{J}}_n$	First upper offdiagonal in $\mathbf{F}$, row n

Notations regarding a discretisation in space:

Ω_h	Triangulation of the domain Ω
V_h	Finite element space for a velocity/temperature
Z_h	Finite element space for a pressure
$\mathcal{V}_h$	$= V_h \times V_h$ for the heat equation, $= V_h \times V_h \times Z_h \times Z_h$ for the Stokes/Navier–Stokes equations
#dof	Number of degrees of freedom
$\mathcal{I}^h, \mathcal{A}^h, \mathcal{D}^h, ...$	Discrete counterparts of $\mathcal{I}$, $\mathcal{H}$, $\mathcal{D}$,... in space
σ	$= (h, k)$
w^σ	$= (w_0^h, w_1^h, ...) \in \mathcal{V}_h^{N+1}$, discrete solution vector
f^σ	$= (f_0^h, f_1^h, ...) \in (\mathcal{V}_h^*)^{N+1}$, discrete right-hand side vector
$G^\sigma(w^\sigma)$	Discrete counterpart to $\mathbf{G}(w)$, implicit Euler time discretisation
$G^{\theta,\sigma}(w^\sigma)$	Discrete counterpart to $\mathbf{G}(w)$, Crank–Nicolson time discretisation
G_n	n-th diagonal block of G^σ
F_n	n-th diagonal block of F^σ
$\breve{M}_n$	First lower offdiagonal in G^σ, row n

$\hat{M}_n$	First upper offdiagonal in G^σ, row n
$\mathcal{X}_h^* \cong \mathcal{X}_h$	Isomorphic association of the dual space $\mathcal{X}_h^*$ of a discrete space $\mathcal{X}_h$ via the dual pairing $(\cdot,\cdot)_{\mathcal{X}_h^*,\mathcal{X}_h}$

Notations regarding the construction of hierarchies:

L	Number of levels in a spatial hierarchy
Ω_l	Mesh at level l of a spatial hierarchy
V^l	Finite element space at level l of the spatial hierarchy
$V^{l,*}$	$= (V^l)^*$, corresponding dual space / space f right hand sides
M	Number of levels in a temporal hierarchy
T^m	Decomposition of $[0,T]$, time level m; implicit Euler time discretisation
$N = N_m$	Number of time intervals of T^m
T_θ^m	Decomposition of $[0,T]$, time level m; Crank–Nicolson time discretisation
$W^{l,m}$	$= (V^l)^{N_m+1}$
NLMAX	Number of levels in a space-time hierarchy
W^{NLMAX}	$= W^{L,M}$
$W^1, W^2, ..., W^{\mathrm{NLMAX}}$	Hierarchy created from W^{NLMAX} by coarsening
$(W^{l,m})^\times$	$= \left((V^l)^*\right)^{N_m+1}$, space of discrete right-hand sides
$(W^{l,m})^\times \cong (V^l)^{N_m+1}$	Isomorphic association via the dual pairing $(\cdot,\cdot)_{\mathcal{V}_h^*,\mathcal{V}_h}$ in space, applied to every component in time; $(W^{l,m})^\times = \left((V^l)^*\right)^{N_m+1} \cong (V^l)^{N_m+1}$
w^{lm}	Discrete solution vector at space level l and time level m
f^{lm}	Discrete right-hand vector at space level l and time level m
G^{lm}	Space-time system matrix at space level l and time level m
W^l	$= W^{l,l}$, simplified notation
w^l	Discrete solution vector corresponding to W^l, simplified notation
f^l	Discrete right-hand vector corresponding to W^l, simplified notation
G^l	Space-time system matrix corresponding to W^l, simplified notation
$(W^l)^\times$	$= (W^{l,l})^\times$, space of discrete right-hand sides, simplified notation

Notations regarding nonlinear solvers:

d_i	i-th nonlinear residual

g_i	i-th nonlinear correction
w_i	i-th nonlinear iterate
$C(w_i^\sigma)$	Preconditioner for the nonlinear defect, $= G^\sigma(w_i^\sigma)$ or $= F^\sigma(w_i^\sigma)$

Notations regarding linear solvers:

d^l	$= (d_0^l, ..., d_N^l)$, space-time defect vector at level l
g^l	$= (g_0^l, ..., g_N^l)$, space-time correction vector at level l
P^l	Space-time prolongation operator from level l to level $l+1$
R^l	Space-time restriction operator from level $l+1$ to level l
$S(\cdot)$	Smoothing operator
NSMpre	Number of presmoothing steps
NSMpost	Number of postsmoothing steps
ω	> 0, damping parameter
$\vec{v} \cong v$	Vector $\vec{v}$ containing the degrees of freedom corresponding to a finite element function v
$A \cong C$	A discrete matrix $A \in \mathbb{R}^{n \times n}$ corresponding to an operator C in the finite element space
A^{-1}	Inverse of a matrix A; in a multigrid setting: Coarse grid solver
x, b	$\in \mathbb{R}^n$, vectors
x^*	$\in \mathbb{R}^n$, solution of a linear system $Ax = b$

Notations regarding prolongation/restriction/coarse grid operators:

$P_{\text{space}} = P_{\text{space}}^m$	Prolongation operator in space from space level m to $m+1$
$R_{\text{space}} = R_{\text{space}}^m$	Restriction operator in space from space level $m+1$ to m
$I_{\text{space}} = I_{\text{space}}^m$	Interpolation operator in space from space level $m+1$ to m
$P_{\text{time}} = P_{\text{time}}^l$	Prolongation operator in time from time level l to $l+1$
$R_{\text{time}} = R_{\text{time}}^l$	Restriction operator in time from time level $l+1$ to l
$I_{\text{time}} = I_{\text{time}}^l$	Interpolation operator in time from time level $l+1$ to l

Ξ	$= \{\xi_0 < \xi_1 < ... < \xi_N \,\vert\, 0 \le \xi_n \le T,\, n = 0, ..., N\}$
X	A finite dimensional Hilbert space
$Y_0 = Y_0(\Xi, X)$	All functions $v : \Xi \to X$ with $v(t) = 0$ for $t \in \Xi$
$Y(\Xi, X)$	$= \{v : [0,T] \to X\} \,/\, Y_0$, space of discrete abstract functions from Ξ to X
$\bar{v}$	$= (v(\xi_0), ..., v(\xi_N))^\mathsf{T}$ for a function $v \in Y(\Xi, X)$
$(\cdot,\cdot)_\Xi$	Scalar product for elements in $Y(\Xi, X)$
$\|\|\cdot\|\|_\Xi$	Norm induced by $(\cdot,\cdot)_\Xi$
$\langle\cdot,\cdot\rangle$	Dual pairing in $Y(\Xi, X)$
$Y(\Xi, X)^*$	Dual space of $Y(\Xi, X)$

$Y(\Xi,X)^* \cong Y(\Xi,X)$	Identification of $Y(\Xi,X)^*$ and $Y(\Xi,X)$ via the dual pairing
$Y(T^l,V^m) \cong W^{l,m}$	Identification via the components in time; $v \in Y(T^l,V^m) \;\Leftrightarrow\; \bar{v} \in W^{l,m}$
$(W^{l,m})^\times \cong Y(T^l,V^m)^*$	Identification via the components in time; $(W^{l,m})^\times \cong (V^l)^{N_m+1} \cong Y(T^l,V^m) \cong Y(T^l,V^m)^*$

w_i^p	For $w=(w_0,...,w_N)$: 'Time-primal' part of w_i, all components of w_i located at the endpoints of time interval i (usually y and ξ)
w_i^d	'Time-dual' part of w_i, all components of w_i located in-between the endpoints of time interval i (usually λ and p)
w^p	$=(w_0^p,...,w_N^p)$ for $w=(w_0,...,w_N)$
w^d	$=(w_0^d,...,w_N^d)$ for $w=(w_0,...,w_N)$

f_i^p	For a right-hand side/defect vector $f=(f_0,...,f_N)$: 'Time-primal' part of f_i, all components of f_i located in-between the endpoints of time interval i (usually f_λ and f_p)
f_i^d	'Time-dual' part of f_i, all components of f_i located at the endpoints of time interval i (usually f_y and f_ξ)

Notations regarding smoothing operators in space:

K	$\in \Omega_h$, an element of a mesh
$I(K)$	Index set with all degrees of freedom corresponding to a finite element function on element K
$x_{I(K)}$	Vector containing the degrees of freedom of x corresponding to $I(K)$
$A_{I(K)}$	Matrix containing the rows $I(K)$ of a matrix A
$A_{I(K),I(K)}$	Matrix containing the rows and columns $I(K)$ of a matrix A
$\mathrm{diag}(A)$	The elements on the diagonal of a matrix A

Notations regarding stopping criteria and residuals:

$\varepsilon_{\mathrm{OptNL}}$	Relative stopping criterion for the nonlinear space-time solver
$\varepsilon_{\mathrm{OptMG}}$	Relative stopping criterion for the linear space-time solver
$\varepsilon_{\mathrm{CoarseMG}}$	Relative stopping criterion for the space-time coarse grid solver
$\varepsilon_{\mathrm{SpaceMG}}$	Relative stopping criterion for the linear solver in space
$\varepsilon_{\mathrm{SimNL}}$	Relative stopping criterion for the nonlinear solver in a time interval during a simulation
$\varepsilon_{\mathrm{SimMG}}$	Relative stopping criterion for the linear solver during the nonlinear iteration of a simulation
r_m^{OptNL}	m-th nonlinear residual
r_m^{OptMG}	m-th linear residual
r_m^{CoarseMG}	m-th linear residual in the space-time coarse grid solver

r_m^{SimNL}	m-th nonlinear residual in the simulation solver in a time interval during a simulation
r_m^{SimMG}	m-th linear residual in the simulation solver during the nonlinear iteration of a simulation

Bibliography

[1] F. Abergel and R. Temam. On some control problems in fluid mechanics. *Theoretical and Computational Fluid Dynamics*, 1:303–325, 1990.

[2] F. Abraham, M. Behr, and M. Heinkenschloss. The effect of stabilization in finite element methods for the optimal boundary control of the Oseen equations. *Finite Elem. Anal. Des.*, 41:229–251, 2004.

[3] G. V. Alekseyev and V. V. Malikin. Numerical analysis of optimal boundary control problems for the stationary Navier-Stokes equations. *Comput. Fluid Dyn. J.*, 3(1): 1–26, 1994.

[4] I. Altrogge, T. Preusser, T. Kröger, Ch. Büskens, P. L. Pereira, D. Schmidt, and H. Peitgen. Multi-scale optimization of the probe placement for radio-frequency ablation. Preprint SPP1253-06-01, SPP1253, 2007.

[5] H. Antil, R. H. W. Hoppe, and Ch. Linsenmann. Optimal design of stationary flow problems by path-following interior-point methods. Preprint SPP1253-18-03, SPP1253, 2007.

[6] Th. Apel. *Anisotropic finite elements: Local estimates and applications.* Advances in Numerical Mathematics. Teubner, 1999. ISBN 3-519-02744-5.

[7] Th. Apel, D. Sirch, and G. Winkler. Error estimates for control constrained optimal control problems: Discretization with anisotropic finite element meshes. Preprint SPP1253-02-06, SPP1253, 2008.

[8] A. K. Aziz and P. Monk. Continuous finite elements in space and time for the heat equation. *Math. Comput.*, 53(186):255–274, 1989.

[9] R. E. Bank and T. F. Dupond. An optimal order process for solving finite element equations. *Math. Comput.*, 36(153):35–51, 1981.

[10] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, Ch. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* SIAM, 1994.

[11] G. Bärwolff and M. Hinze. Optimization of semiconductor melts. *ZAMM*, 86:423–437, 2006.

[12] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming. Theory and Algorithms.* Wiley-Interscience, 2006. 3rd edition, ISBN 0471486000.

[13] Y. Bazilevs and Th. J. R. Hughes. Weak imposition of Dirichlet boundary conditions in fluids mechanics. *Comp. Fluids*, 36:12–26, 2007. doi: 10.1016/j.compfluid.2005.07.012.

[14] Ch. Becker. *Strategien und Methoden zur Ausnutzung der High-Performance-Ressourcen moderner Rechnerarchitekturen für Finite-Element-Simulationen und ihre Realisierung in FEAST (Finite Element Analysis & Solution Tools)*. PhD thesis, Universität Dortmund, Logos Verlag, Berlin, 2007. `http://www.logos-verlag.de/cgi-bin/buch?isbn=1637`, ISBN 978-3-8325-1637-6.

[15] Ch. Becker, S. H. M. Buijssen, S. Kilian, and S. Turek. High performance FEM simulation via FEAST and application to parallel CFD via FEATFLOW. In H. Rollnik and D. Wolf, editors, *NIC Symposium 2001*, Volume 9 of *NIC-Serie*, pages 493–502. Forschungszentrum Jülich, 2002. Forschungszentrum Jülich, 2001.

[16] M. Berggren. Approximations of very weak solutions to boundary-value problems. *SIAM J. Num. Anal.*, 42(2):860–877, 2004.

[17] J. T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, 2001. ISBN 0898714885.

[18] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders. Large-scale PDE-constrained optimization. In *Lecture Notes in Computational Science and Engineering*. Springer, Berlin, 2003.

[19] A. Borzi. Introduction to multilevel methods. Technical report, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, 1994. Lecture Notes in Mathematics.

[20] A. Borzi. Fast multigrid methods for parabolic optimal control problems, 2002. Proceedings of the 18th GAMM-Seminar.

[21] A. Borzi. Multigrid methods for parabolic distributed optimal control problems. *J. Comp. Appl. Math.*, 157:365–382, 2003.

[22] A. Borzi. *Multigrid Methods for Optimality Systems*. Habilitation thesis, University of Graz, 2003.

[23] A. Borzi and R. Griesse. Experiences with a space-time multigrid method for the optimal control of a chemical turbulence model. *Int. J. Numer. Meth. Fluids*, 47: 879–885, 2005.

[24] A. Borzi and V. Schulz. Multigrid methods for PDE optimization. *SIAM Review*, 51 (2):361–395, 2009.

[25] N. Botkin, K. Hoffmann, and V. Turova. Stable solutions of Hamilton–Jacobi equations. application to control of freezing processes. Preprint SPP1253-080, SPP1253, 2009.

[26] D. Braess. *Finite Elements*. Cambridge University Press, 2nd edition, 2001. ISBN 0-521-01195-7.

[27] Ch. Brandenburg, F. Lindemann, M. Ulbrich, and S. Ulbrich. A continuous adjoint approach to shape optimization for Navier Stokes flow. Preprint SPP1253-14-01, SPP1253, 2008.

[28] Ch. Brandenburg, F. Lindemann, M. Ulbrich, and S. Ulbrich. Advanced numerical methods for PDE constrained optimization with application to optimal design in Navier Stokes flow. In S. Engell, A. Griewank, M. Hinze, G. Leugering, R. Rannacher, and V. Schulz, editors, *Constrained Optimization and Optimal Control for Partial Differential Equation*. Birkhäuser, 2010.

[29] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comput.*, 31:333–390, 1977.

[30] A. L. Braslow. A history of suction-type laminar-flow control with emphasis on flight research. Monographs in Aerospace History 13, American Institute of Aeronautics and Astronautics, Washington, D.C., 1999.

[31] Brenner. A nonconforming multigrid method for the stationary Stokes equations. *Math. Comput.*, 55(192):411–137, 1990.

[32] S. C. Brenner. An optimal-order multigrid method for P_1 nonconforming finite elements. *Math. Comput.*, 52(185):1–15, 1989.

[33] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *RAIRO*, 8(R-2):129–151, 1974.

[34] G. Büttner. *Ein Mehrgitterverfahren zur optimalen Steuerung parabolischer Probleme*. PhD thesis, Fakultät II – Mathematik und Naturwissenschaften der Technischen Universität Berlin, 2004. `http://edocs.tu-berlin.de/diss/2004/buettner_guido.pdf`.

[35] E. Casas. Error estimates for the numerical approximation of Dirichlet boundary control for semilinear elliptic equations. *SIAM J. Control Opt.*, 45(5):1586–1611, 2006.

[36] Texas Advanced Computing Center. GotoBLAS library, 2010. `http://www.tacc.utexas.edu/tacc-projects`.

[37] Ph. G. Ciarlet. *The finite element method for elliptic problems*. Studies in mathematics and its applications, Vol. 4. North-Holland Publishing Company, Amsterdam, New-York, Oxford, 1978. ISBN 0444850287.

[38] Ph. G. Ciarlet and J. L. Lions. *Finite Element Methods (Part 1)*, Volume II of *Handbook of Numerical Analysis*. Elsevier, 1991.

[39] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley-Interscience, 1983.

[40] S. S. Collis and M. Heinkenschloss. Analysis of the Streamline Upwind/Petrov Galerkin method applied to the solution of optimal control problems. Technical Report TR02-01, Department of Computational and Applied Mathematics, Rice University, 2002. `http://www.caam.rice.edu/`.

[41] S. S. Collis, K. Ghayour, M. Heinkenschloss, M. Ulbrich, and S. Ulbrich. Numerical solution of optimal control problems governed by the compressible Navier–Stokes equations. In *Optimal Control of Complex Structures*, Volume 139 of *International Series of Numerical Mathematics*, pages 43–55. Birkhäuser, 2001.

[42] S. S. Collis, K. Ghayour, M. Heinkenschloss, M. Ulbrich, and S. Ulbrich. Optimal control of unsteady compressible viscous flows. *Int. J. Numer. Meth. Fluids*, 40: 1401–1429, 2002. doi:10.1002/d.420.

[43] R. M. Colombo, G. Guerra, M. Herty, and V. Sachers. Modeling and optimal control of networks of pipes and canals. Preprint SPP1253-19-02, SPP1253, 2008.

[44] S. Conti, H. Held, M. Pach, M. Rumpf, and R. Schultz. Shape optimization under uncertainty – a stochastic programming perspective. Preprint SPP1253-074, SPP1253, 2009.

[45] R. Cools, D. P. Laurie, and L. Pluym. Algorithm 764: Cubpack++. a C++ package for automatic two-dimensional cubature. *ACM Transactions on Mathematical Software*, 23(1):1–15, 1997.

[46] H. Damanik, J. Hron, A. Ouazzi, and S. Turek. Monolithic Newton-multigrid solution techniques for incompressible nonlinear flow models. Ergebnisberichte des Instituts für Angewandte Mathematik, Nr. 426, Fakultät für Mathematik, TU Dortmund, 2011.

[47] P. A. Davidson. *An Introduction to Magnetohydrodynamics*. Cambridge University Press, 2001.

[48] D. Davis. UMFPACK. Mathematic library, University of Florida, `http://www.cise.ufl.edu/research/sparse/umfpack/`, 2004.

[49] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, New York, 2nd edition, 1984.

[50] K. Deckelnick and M. Hinze. Variational discretization of parabolic control problems in the presence of pointwise state constraints. Preprint SPP1253-08-08, SPP1253, 2009.

[51] K. Deckelnick, A. Günther, and M. Hinze. Finite element approximation of Dirichlet boundary control for elliptic PDE's on two and three-dimensional curved domains. *SIAM J. Control Opt.*, 48(4):2789–2819, 2009.

[52] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for unconstrained Optimization and Nonlinear Equations*. Number 16 in Classics In Applied Mathematics. SIAM, 1996. ISBN 0898713641.

[53] P. Deuflhard. *Newton methods for Nonlinear Problems. Affine Invariances and Adaptive Algorithms*. Springer, Berlin, 2004. ISBN 3540210997.

[54] P. Dhar and R. Dhar. Optimal control for bio-heat equation due to induced microwave. *Appl. Math. Mech.*, 31(4):529–534, 2010.

[55] J. Douglas and T. F. Dupond. Interiour penalty procedures for elliptic and parabolic Galerkin methods. In *Computing methods in applied sciences (2nd International Symposium, Versailles, 1975)*, Volume 58 of *Lecture Note in Physics*. Springer, Berlin, 1976.

[56] S. C. Eisenstat and H. F. Walker. Globally convergent inexact Newton methods. *SIAM J. Optim.*, 4(2):393–422, 1994.

[57] E. Emmrich. *Gewöhnliche und Operator-Differentialgleichungen.* Vieweg, 2004. ISBN 3528032138.

[58] L. C. Evans. *Partial Differential Equations*, Volume 19 of *Graduate Studies in Mathematics.* American Mathematical Society, 1998. ISBN 0821807722.

[59] Th. Flaig. Crank-Nicolson schemes for optimal control problems with evolution equations. Preprint SPP1253-113, SPP1253, 2010.

[60] L. P. Franca and Th. J. R. Hughes. A new finite element formulation for computational fluid dynamics: VII. The Stokes-problem with various well-posed boundary conditions: Symmetric formulations that converge for all velocity/pressure spaces. *Comput. Meth. Appl. Mech. Eng.*, 65:85–96, 1987.

[61] R. Glowinski. Finite element methods for the numerical simulation of incompressible viscous flow. introduction to the control of the Navier–Stokes equations. In *Vortex Dynamics and Vortex Methods*, Volume 28 of *Lectures in Applied Mathematics*, pages 219–301. American Mathematical Society, 1991.

[62] M. S. Gockenbach. *Understanding and Implementing the Finite Element Method.* SIAM, 2006. ISBN 0898716144.

[63] M. Grajewski. *A new fast and accurate grid deformation method for r-adaptivity in the context of high performance computing.* Logos Verlag, Berlin, 2008. `http://www.logos-verlag.de/cgi-bin/buch?isbn=1903`, ISBN 978-3-8325-1903-2.

[64] M. Grajewski, M. Köster, and S. Turek. Mathematical and numerical analysis of a robust and efficient grid deformation method in the finite element context. *SIAM J. Sci. Comput.*, 31(2):1539–1557, 2008.

[65] M. Grajewski, M. Köster, and S. Turek. Numerical analysis and implementational aspects of a new multilevel grid deformation method. *Appl. Num. Math.*, 60(8): 767–781, 2010. doi:10.1016/j.apnum.2010.03.017.

[66] P. M. Gresho and R. L. Sani. Resume and remarks on the open boundary condition minisymposium. *Int. J. Numer. Meth. Fluids*, 18:983–1008, 1994.

[67] R. Griesse and K. Kunisch. Optimal control for a stationary MHD system in velocity-current formulation. *SIAM J. Control Opt.*, 45(5):1822–1845, 2006.

[68] FEAT Group. Featflow - high performance finite element analysis tool, 1989. `http://www.featflow.de`.

[69] M. D. Gunzburger. *Perspectives in Flow Control and Optimization.* SIAM, 2003. ISBN 089871527X.

[70] M. D. Gunzburger and S. Manservisi. Analysis and approximation of the velocity tracking problem for Navier–Stokes flows with distributed control. *SIAM J. Num. Anal.*, 37(5):1481–1512, 2000.

[71] M. D. Gunzburger, E. Ozugurlu, J. Turner, and H. Zhang. Controlling transport phenomena in the Czochralski crystal growth process. *J. Crystal Growth*, 234:47–62, 2002.

[72] W. Hackbusch. Fast solution of elliptic control problems. *J. Opt. Theory and Appl.*, 31(4):565–581, 1980.

[73] W. Hackbusch. Die schnelle Auflösung der Fredholmschen Integralgleichung zweiter Art. *Beiträge zur numerischen Mathematik*, 9, 1981.

[74] W. Hackbusch. Numerical solution of linear and nonlinear parabolic optimal control problems. *Lecture Notes in Control and Information Science*, 30, 1981.

[75] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer Series in Computational Mathematics. Springer, Berlin, 1985. ISBN 3-540-12761-5.

[76] W. Hackbusch. Multigrid methods for FEM and BEM applications. In E. Stein, R. de Borst, and Th. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics*, chapter 20. John Wiley & Sons Ltd., 2004.

[77] A. Hamdi and A. Griewank. Reduced quasi-Newton method for simultaneous design and optimization. Preprint SPP1253-11-02, SPP1253, 2008.

[78] M. Heinkenschloss. Formulation and analysis of a sequential quadratic programming method for the optimal Dirichlet control of Navier–Stokes flow. In W. W. Hager and P. M. Pardalos, editors, *Optimal Control – Theory, Algorithms and Applications*, pages 178–203. Kluwer, 1998.

[79] V. E. Henson. Multigrid methods for nonlinear problems: An overview. Preprint UCRL-JC-150259, Lawrence Livermore National Laboratory, 2002.

[80] R. Herzog and K. Kunisch. Algorithms for PDE-constrained optimization. In *GAMM-Mitteilungen*, Volume 33, pages 163–176. Wiley-Interscience, 2010.

[81] J. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier–Stokes equations. *Int. J. Num. Meth. Fluids*, 22(5):325–352, 1996.

[82] E. Hille and R. S. Phillips. *Functional Analysis and Semi-Groups*, Volume 31 of *Colloquium Publications*. American Mathematical Society, revised edition, 1996. ISBN 0821810316.

[83] M. Hintermüller and M. Hinze. A SQP-semi-smooth Newton-type algorithm applied to control of the instationary Navier–Stokes system subject to control constraints. *SIAM J. Optim.*, 16:1177–1200, 2006.

[84] M. Hinze. *Optimal and instantaneous control of the instationary Navier–Stokes equations*. Habilitation thesis, Institut für Numerische Mathematik, Technische Universität Dresden, 2000.

[85] M. Hinze and K. Kunisch. Three control methods for time-dependent fluid flow. *Flow Turbulence Combust.*, 68:273–298, 2000.

[86] M. Hinze and K. Kunisch. Second order methods for boundary control of the instationary Navier–Stokes system. *ZAMM*, 84(3):171–187, 2004.

[87] M. Hinze and F. Tröltzsch. Discrete concepts versus error analysis in PDE-constrained optimization. In *GAMM-Mitteilungen*, Volume 33, pages 148–162. Wiley-Interscience, 2010.

[88] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. Lecture notes of the autumn school modelling and simulation with partial differential equations, 2005. Hamburg, September 26-30.

[89] M. Hinze, M. Köster, and S. Turek. A hierarchical space-time solver for distributed control of the Stokes equation. Preprint SPP1253-16-01, SPP1253, 2008.

[90] M. Hinze, M. Köster, and S. Turek. A space-time multigrid solver for distributed control of the time-dependent Navier–Stokes system. Preprint SPP1253-16-02, SPP1253, 2008.

[91] M. Hinze, M. Köster, and S. Turek. A hierarchical space-time solver for optimal distributed control of fluid flow, 2009. Proceedings of the Conference on Modeling, Simulation and Optimization of Complex Processes, Heidelberg, July 21-25, 2008, accepted.

[92] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE Constraints*, Volume 23 of *Mathematical Modelling: Theory and Applications*. Springer, Berlin, 2009. ISBN 9781402088384.

[93] M. Hinze, M. Köster, and S. Turek. A space-time multigrid method for optimal flow control. In *Themenband I, Optimization with Partial Differential Equations (DFG SPP 1253)*, ISNM-Series. Birkhäuser, 2011. accepted, to appear.

[94] K. Hoffmann and N. Botkin. Optimal control in cryopreservation of cells and tissues. Preprint SPP1253-17-03, SPP1253, 2008.

[95] Th. J. R. Hughes, L. P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuska-Brezzi condition. A stable Petrov-Galerkin formulation of the Stokes problem accomodating equal-order interpolations. *Comput. Meth. Appl. Mech. Eng.*, 59(1):85–99, 1986.

[96] S. Hussain, F. Schieweck, and S. Turek. Higher order Galerkin time discretizations and fast multigrid solvers for the heat equation. *J. Num. Math.*, 19(1):41–61, 2011. DOI 10.1515/ JNUM.2011.003.

[97] V. John and P. Knobloch. On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations part I – a review. *Comput. Meth. Appl. Mech. Engrg.*, 196:2197–2215, 2007.

[98] V. John and P. Knobloch. On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations part II - analysis for P_1 and Q_1 finite elements. *Comput. Meth. Appl. Mech. Engrg.*, 197:1997–2014, 2008.

[99] V. John and G. Matthies. Higher order finite element discretizations in a benchmark problem for incompressible flows. Preprint, Otto-von-Guericke-Universität Magdeburg, Institut für Analysis und Numerik, 2001.

[100] V. John and E. Schmeyer. Finite element methods for time-dependent convection-diffusion-reaction equations with small diffusion. *Comput. Meth. Appl. Mech. Engrg.*, 198:475–494, 2008.

[101] M. Juntunen and R. Stenberg. Nitsche's method for general boundary conditions. *Math. Comput.*, 78(267):1353–1374, 2009.

[102] D. Kahaner, C. Moler, and S. G. Nash. *Numerical Methods and Software.* Prentice Hall, 1989. ISBN 0136272584.

[103] S. Kameswaran and L. T. Biegler. Advantages of nonlinear-programming-based methodologies for inequality path-constrained optimal control problems – a numerical study. *SIAM J. Sci. Comput.*, 30(2):957–981, 2008.

[104] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* SIAM, 1995.

[105] C. T. Kelley. *Iterative Methods for Optimization.* Frontiers in Applied Mathematics 18. SIAM, 1999. ISBN 0898714338.

[106] S. Kilian. *ScaRC als verallgemeinerter Mehrgitter- und Gebietszerlegungsansatz für parallele Rechnerplattformen.* Logos Verlag, Berlin, 2002. `http://www.logos-verlag.de/cgi-bin/buch?isbn=0092`, ISBN 978-3-8325-0092-8.

[107] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[108] M. Köster. Robuste Mehrgitter-Krylowraum-Techniken für FEM-Verfahren. Diplomarbeit, TU Dortmund, Fakultät für Mathematik, Lehrstuhl 3 für Angewandte Mathematik und Numerik, Diploma thesis, 2004.

[109] M. Köster, A. Ouazzi, F. Schieweck, S. Turek, and P. Zajac. New robust nonconforming finite elements of higher order. *Appl. Num. Math.*, 2011. submitted.

[110] C. Kratzenstein and Th. Slawig. Oneshot parameter identification – simultaneous model spin-up and parameter optimization in a box model of the north atlantic thermohaline circulation. Preprint SPP1253-082, SPP1253, 2009.

[111] K. Kunisch and B. Vexler. Constrained Dirichlet boundary control in L^2 for a class of evolution equations. *SIAM J. Control Opt.*, 46(5):1726–1753, 2007.

[112] J. L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations.* Springer, Berlin, 1971.

[113] H. Maurer and H. D. Mittelmann. Optimization techniques for solving elliptic control problems with control and state constraints. I: Boundary control. *J. Comp. Appl. Math.*, 16:29–55, 2000.

[114] H. Maurer and H. D. Mittelmann. Optimization techniques for solving elliptic control problems with control and state constraints. II: Distributed control. *J. Comp. Appl. Math.*, 18:141–160, 2001.

[115] D. Meidner and B. Vexler. A priori estimates for space-time finite element discretisations of parabolic optimal control problems. part I: Problems without control constraints. *SIAM J. Control Opt.*, 47(3):1150–1177, 2008.

[116] D. Meidner and B. Vexler. A priori estimates for space-time finite element discretisations of parabolic optimal control problems. part II: Problems with control constraints. *SIAM J. Control Opt.*, 47(3):1301–1329, 2008.

[117] D. Meidner, R. Rannacher, and B. Vexler. A priori error estimates for finite element discretizations of parabolic optimization problems with pointwise state constraints in time. Preprint SPP1253-098, SPP1253, 2010.

[118] A. Meister. *Numerik linearer Gleichungssysteme.* Vieweg, 1999. ISBN 3528031352.

[119] K. W. Morton and D. F. Mayers. *Numerical solution of partial differential equations.* Cambridge University Press, 2005. ISBN 0521607930.

[120] M. Moubachir and J. Zolesio. Optimal control of fluid-structure interaction systems: The case of a rigid solid. Report 4611, INRIA - Institut National de Recherche en Informatique et en Automatique, 2002.

[121] I. Neitzel, U. Prüfert, and Th. Slawig. A smooth regularization of the projection formula for constrained parabolic optimal control problems. Preprint TU 05-2009, Technische Universität Berlin, Fakultät II - Mathematik und Naturwissenschaften, 2009.

[122] NETLIB. LAPACK – Linear Algebra PACKage, 1992. `http://www.netlib.org/lapack/`.

[123] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, Berlin, 1999.

[124] A. Ouazzi. *Finite element simulation of nonlinear fluids with application to granular material and powder.* Phd thesis, TU Dortmund, Fakultät für Mathematik, Lehrstuhl 3 für Angewandte Mathematik und Numerik, 2005.

[125] E. Özkaya and N. R. Gauger. Single-step one-shot aerodynamic shape optimization. Preprint SPP1253-10-04, SPP1253, 2008.

[126] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker. Globalization techniques for Newton–Krylov methods and applications to the fully-coupled solution of the Navier–Stokes equations. *SIAM Review*, 48(4):700–721, 2006.

[127] M. Probst, M. Lülfesmann, M. Nicolai, M. Bücker, D. Behr, and Ch. Bischof. Sensitivities of optimal shapes of artificial grafts with respect to flow parameters. Preprint SPP1253-04-04, SPP1253, 2009.

[128] R. Rannacher, M. Schäfer, and S. Turek. Evaluation of a CFD benchmark for laminar flows. Preprints SFB 359, Nr. 98–23, Universität Heidelberg, 1998.

[129] Wolfram Research. Mathematica, 2011. `http://www.wolfram.com/mathematica/`.

[130] A. Schiela and A. Günther. Interior point methods in function space for state constraints – inexact Newton and adaptivity. Preprint SPP1253-08-06, SPP1253, 2009.

[131] F. Schieweck. A-stable discontinuous Galerkin–Petrov time discretization of higher order. *J. Num. Math.*, 18(1):25–57, 2010.

[132] R. Schmachtel. *Robuste lineare und nichtlineare Lösungsverfahren für die inkompressiblen Navier–Stokes-Gleichungen.* Phd thesis, Universität Dortmund, Fakultät für Mathematik, Lehrstuhl 3 für Angewandte Mathematik und Numerik, 2003.

[133] S. Schmidt and V. Schulz. Impulse response approximations of discrete shape Hessians with application in CFD. Preprint SPP1253-10-02, SPP1253, 2008.

[134] S. Schmidt, C. Ilic, N. R. Gauger, and V. Schulz. Shape gradients and their smoothness for practical aerodynamic design optimization. Preprint SPP1253-10-03, SPP1253, 2008.

[135] J. N. Shadid, Ray. Tuminaro, and H. F. Walker. An inexact Newton method for fully coupled solution of the Navier–Stokes equations with heat and mass transport. *J. Comput. Phys.*, 137(1):155–185, 1997.

[136] M. H. Shojaefard, A. R. Noorpoor, A. Avanesians, and M. Ghaffarpour. Numerical investigation of flow control by suction and injection on a subsonic airfoil. *Am. J. Appl. Sci.*, 2(10):1474–1480, 2005.

[137] Th. Slawig. PDE-constrained control using COMSOL multiphysics - control of the Navier–Stokes equations, 2006. Proceedings of the COMSOL Users Conference 2006, Frankfurt, Page 181–183.

[138] R. T. Trimbitas. Adaptive cubatures on triangle. *Result. Math.*, 53:453–462, 2009. doi: 10.1007/s00025-008-0357-6.

[139] F. Tröltzsch. *Optimal Control of Partial Differential Equations.* American Mathematical Society, 2010. ISBN 0821849042.

[140] F. Tröltzsch and H. Goldberg. On a SQP-multigrid technique for nonlinear parabolic boundary control problems. Preprint, Fakultät für Mathematik, Technische Universität Chemnitz, 1997. `http://www.mathematik.tu-chemnitz.de/preprint/1997/PREPRINT_11.html`.

[141] U. Trottenberg, Cornelis. W. Oosterlee, and A. Schüller. *Multigrid.* Academic Press, New York, 2001. ISBN 012701070X.

[142] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach.* Springer, Berlin, 1999. ISBN 3-540-65433-X.

[143] S. Turek and H. Oswald. A parallel multigrid algorithm for solving the incompressible Navier–Stokes equations with nonconforming finite elements in three dimensions. In *Proc. Parallel CFD 1996, Capri/Italy*, 1996.

[144] S. Turek and A. Ouazzi. Unified edge-oriented stabilization of nonconforming FEM for incompressible flow problems: Numerical investigations. *J. Num. Math.*, 15(4): 299–322, 2007.

[145] S. Turek and R. Rannacher. A simple nonconforming quadrilateral stokes element. *Num. Meth. Part. D. E.*, 8(2):97–111, 1992.

[146] S. Turek and M. Schäfer. Benchmark computations of laminar flow around cylinder. In E.H. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, Volume 52 of *Notes on Numerical Fluid Mechanics*, pages 547–566. Vieweg, 1996.

[147] S. Turek, Ch. Becker, and S. Kilian. Some concepts of the software package FEAST. In J. M. Palma, J. J. Dongarra, and V. Hernandes, editors, *Vector and Parallel Processing – VECPAR 1998*, Volume 1573 of *Lecture Notes in Computer Science*, pages 271–284. Springer, Berlin, 1999. doi:10.1007/10703040_22.

[148] S. Turek, Ch. Becker, and S. Kilian. Hardware-oriented numerics and concepts for PDE software. *Future Generat. Comput. Syst.*, 22(1–2):217–238, 2006. doi:10.1016/j.future.2003.09.007.

[149] S. Turek, D. Göddeke, Ch. Becker, S. H. M. Buijssen, and H. Wobker. UCHPC: unconventional high-performance computing for finite element simulations, 2008. Winner of the first PRACE award; `http://www.prace-project.eu/news/prace-award-presented-to-young-scientist-at-isc201908`.

[150] S. Turek, D. Göddeke, Ch. Becker, S. H. M. Buijssen, and H. Wobker. FEAST – realisation of hardware-oriented numerics for HPC simulations with finite elements. *Concurrency and Computation: Practice and Experience*, 6:2247–2265, 2010. Special Issue Proceedings of ISC 2008. doi:10.1002/cpe.1584.

[151] M. Ulbrich. Semismooth Newton methods for the operator equations in function spaces. *SIAM J. Optim.*, 3:805–841, 2003.

[152] M. Ulbrich. Constrained control of Navier–Stokes flow by semismooth Newton methods. *Syst. Contr. Lett.*, 48:297–311, 2003.

[153] B. van Bloemen Waanders, R. Bartlett, P. Boggs, K. Long, and A. Salinger. Large scale non-linear programming for PDE constrained optimization. Technical Report SAND2002-3198, Sandia National Laboratories, 2002. `http://endo.sandia.gov/DAKOTA/papers/pdeco_ldrd_finalreport.pdf`.

[154] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2): 631–644, 1992.

[155] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *J. Comput. Phys.*, 65:138–158, 1986.

[156] W. Vogt. Adaptive Verfahren zur numerischen Quadratur und Kubatur. Preprint M 1/06, Technische Universität Ilmenau, Institut für Mathematik, 2006.

[157] D. Wachsmuth. *Optimal control of the unsteady Navier–Stokes equations*. Phd thesis, Fakultät II - Mathematik und Naturwissenschaften, Technische Universität Berlin, 2006.

[158] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons Ltd., 1992. `http://www.mgnet.org/mgnet-books-wesseling.html`.

[159] H. Wobker and S. Turek. Numerical studies of Vanka-type smoothers in computational solid mechanics. *Adv. Appl. Math. Mech.*, 1(1):29–55, 2009.

[160] P. Wriggers. *Nonlinear Finite Element Methods*. Springer, Berlin, 2008. ISBN 9783540710004.

[161] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, pages 1–44, 1992.

[162] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*. McGraw Hill, fourth edition, 1994.