Riccati Feedback Stabilization of the Stefan Problem

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

von **M. Sc. Björn Baran**

geb. am **08.12.1989** in Flensburg

genehmigt durch die Fakultät für Mathematik

der Otto-von-Guericke-Universität Magdeburg

Gutachter: **Prof. Dr. Peter Benner**

Prof. Dr. Tobias Breiten

eingereicht am: **24.04.2023**

Verteidigung am: **07.09.2023**

PUBLICATIONS

Most of the material presented in this thesis is either published or submitted for publication. This material was developed during the author's time as a Ph.D. student at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg in the period from April 1, 2016 to April 30, 2023. The author is the main author of the following publications produced during this period.

[Bar24] B. Baran. Linear Quadratic Regulator Computation for a Stefan Problem using M.-M.E.S.S. and FEniCS, February 2024. doi:10.5281/zenodo.10684136.

[BBHS18] B. Baran, P. Benner, J. Heiland, and J. Saak. Optimal control of a Stefan problem fully coupled with incompressible Navier–Stokes equations and mesh movement. *Analele Stiintifice ale Universitatii Ovidius Constanta: Seria Matematica*, XXVI(2):11–40, 2018. doi:10.2478/auom-2018-0016.

[BBS22] B. Baran, P. Benner, and J. Saak. Riccati feedback control of a two-dimensional two-phase Stefan problem. e-print arXiv:2209.05476v2, arXiv, September 2022. math.NA. URL: https://arxiv.org/abs/2209.05476.

[BBSS24] B. Baran, P. Benner, J. Saak, and T. Stillfjord. Numerical methods for closed-loop systems with non-autonomous data. e-print arXiv:2402.13656, arXiv, 2024. math.NA. URL: https://arxiv.org/abs/2402.13656.

[BBHS17] B. Baran, P. Benner, J. Heiland, and J. Saak. Towards Riccati-feedback control of complex flows with moving interfaces. *Proc. Appl. Math. Mech.*, 17(1):769–770, March 2017. doi:10.1002/pamm.201710352.

[BH16] B. Baran and J. Heiland. Adjoint-based optimal boundary control of a Stefan problem system fully coupled with Navier–Stokes equations. *Proc. Appl. Math. Mech.*, 16(1):779–780, October 2016. doi:10.1002/pamm.201610378.

[SBB19] J. Saak, B. Baran, and P. Benner. Riccati-feedback control of a two-dimensional two-phase Stefan problem. In T. Meurer and F. Woittennek, editors, *Tagungsband GMA-FA 1.30 'Modellbildung, Identifikation und Simulation in der Automatisierungstechnik' und GMA-FA 1.40 'Systemtheorie und Regelungstechnik', Workshops in Anif, Salzburg, 23.-27.09.2019*, pages 478–496, 2019.

These publications have found their way into this thesis and form large parts of it, with the exception of [BBHS17], although the suggestions in Section 5.2 are inspired by the ideas in [BBHS17]. Specifically, [BBHS18] represents the preliminary work for this thesis. What stands out from the listed publications are the introduction in Chapter 1, the conclusions and outlook in Chapter 5 and large parts of the numerical results in Section 4.4.

The mathematical background in Chapter 2 combines basics from all publications and additional literature. The methods and theoretical results in Chapter 3 and most of the numerical experiments and results in Chapter 4 have been submitted for publication and are available as preprints [BBSS24, BBS22].

ABSTRACT

The modeling, simulation, control, and stabilization of phase change problems is an active area of research. One of the most commonly used models for this phenomenon is the Stefan problem. In particular, the position and trajectory of the moving interface play an important role in the material quality or the energy and time efficiency in many real solidification processes. In particular, the feedback stabilization of the two-dimensional two-phase Stefan problem has the potential to automate such solidification processes.

The goal of this thesis is to conceptually and numerically investigate the derivation and computation of a feedback stabilization for the Stefan problem, and then to apply this feedback stabilization in a closed-loop simulation. However, the moving interface poses several challenges that are addressed in this thesis. To derive a feedback stabilization, the linear-quadratic regulator approach is chosen, which requires a linearization of the non-linear Stefan problem. This linear version of the Stefan problem can then be discretized in space to assemble matrices that are the coefficients for a differential Riccati equation. The main challenge here is that the moving interface causes the matrices to be time-dependent and thus the differential Riccati equation to be non-autonomous. To solve this Riccati equation numerically, a major contribution of this thesis are non-autonomous backward differentiation formulas.

Another challenge arising from the moving interface during the closed-loop simulation of the Stefan problem are numerical issues that cause a blow-up behavior of time-stepping schemes such as the implicit Euler method or the trapezoidal rule. A further major contribution of this thesis is the presentation of a time-adaptive strategy that is combined with the fractional-step theta scheme for the closed-loop simulation of the Stefan problem.

Finally, the different aspects of the derivation and computation of a feedback stabilization are investigated numerically to assess the applicability and performance of the proposed methods. In addition, the numerical codes and data used for the experiments are available to make the results of this thesis reproducible and reusable.

ZUSAMMENFASSUNG

Die Modellierung, Simulation, Steuerung und Stabilisierung von Phasenwechselproblemen ist ein aktives Forschungsgebiet. Eines der am häufigsten verwendeten Modelle für dieses Phänomen ist das Stefan-Problem. Insbesondere die Position und die Trajektorie der sich bewegenden Phasengrenze spielen eine wichtige Rolle für die Materialqualität oder die Energie- und Zeiteffizienz in vielen realen Erstarrungsprozessen. Insbesondere die Feedback-Stabilisierung des zweidimensionalen zweiphasigen Stefan-Problems hat das Potential, solche Erstarrungsprozesse zu automatisieren.

Ziel dieser Dissertation ist es, die Herleitung und Berechnung einer Feedback-Stabilisierung für das Stefan-Problem konzeptionell und numerisch zu untersuchen und anschließend diese Feedback-Stabilisierung in einer Closed-Loop-Simulation anzuwenden. Die bewegte Phasengrenze stellt jedoch mehrere Herausforderungen dar, die in dieser Dissertation behandelt werden. Um eine Feedback-Stabilisierung abzuleiten, wird der Ansatz des linearen-quadratischen Reglers gewählt, der eine Linearisierung des nichtlinearen Stefan-Problems erfordert. Diese lineare Version des Stefan-Problems kann dann im Raum diskretisiert werden, um Matrizen aufzustellen, die die Koeffizienten für eine Riccati-Differentialgleichung darstellen. Die größte Herausforderung besteht darin, dass die Matrizen aufgrund der sich bewegenden Phasengrenze zeitabhängig sind und die Riccati-Differentialgleichung somit nicht autonom ist. Um diese Riccati-Gleichung numerisch zu lösen, sind nicht-autonome BDF-Verfahren (Backward Differentiation Formulas) ein wichtiger Beitrag dieser Dissertation.

Eine weitere Herausforderung, die sich aus der beweglichen Phasengrenze während der Simulation des Stefan-Problems ergibt, sind numerische Probleme, die ein Blow-up-Verhalten von Zeitschrittverfahren wie der impliziten Euler-Methode oder der Trapezregel verursachen. Ein weiterer wichtiger Beitrag dieser Dissertation ist die Einführung einer zeitadaptiven Strategie, die mit dem Theta-Verfahren (fractional-step theta scheme) für die Simulation mit Stabilisierung des Stefan-Problems kombiniert wird.

Schließlich werden die verschiedenen Aspekte der Herleitung und Berechnung einer Feedback-Stabilisierung numerisch untersucht, um die Anwendbarkeit und Leistungsfähigkeit der vorgeschlagenen Methoden zu überprüfen. Darüber hinaus sind die für die Experimente verwendeten numerischen Codes und Daten verfügbar, um die Ergebnisse

dieser Dissertation reproduzierbar und wiederverwendbar zu machen.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1 INTRODUCTION

Contents

1.1. Motivation

A fundamental phenomenon that plays an important role both in nature and in many industrial processes is the transition between two aggregate states of different materials. In particular, the solid and liquid states of aggregation are elementary manifestations of matter. Each of them differs discontinuously in the mobility of its atoms and molecules and in the strength of the interactions between them. It is the discontinuity of this transition process which poses a mathematical challenge so that the modeling, simulation, control, and stabilization of these problems has been an active research field since two centuries. The present work contributes to this field by proposing, as well as conceptually and numerically investigating, a feedback stabilization for a model of the problem.

Two of the first mathematical models of the transition between the solid and liquid states of aggregation were published first in 1831 by G. Lamé and B. P. Clapeyron [LC31] and then later by J. Stefan [Ste89, Ste90, Ste91]. Today, the model that was published by J. Stefan is often called Stefan problem. It describes the process of solidification and melting of a pure material on a certain domain Ω as a non-linear partial differential equation (PDE). The domain contains the solid phase, that is the region Ω_s where the material is solid, and the liquid phase, the region Ω_l where the material is liquid. Between those two is an interface Γ_{int}, which is moving depending on the solidification or melting of the material.

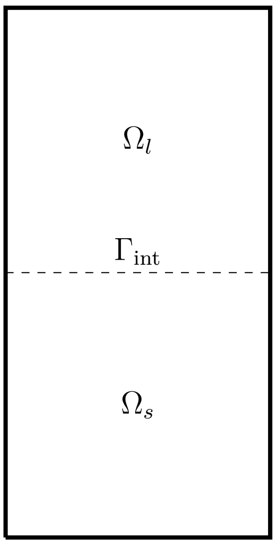

Figure 1.1.: One instance of the domain $\Omega(t) \subset \mathbb{R}^2$ of the Stefan problem.

For an example of the domain, see Figure 1.1. The unknowns of the Stefan problem are the position of the interface and its velocity Υ_{int} as well as the temperature Θ of the material. A simple version of the system of equations can be written in the following way:

$$\dot{\Theta} - \alpha \Delta \Theta = 0, \qquad \text{on } \Omega, \tag{1.1a}$$

$$\Upsilon_{\text{int}} = \left(\frac{1}{\ell} \left[k_s \partial_{\boldsymbol{n}_{\text{int}}} \Theta \big|_{\Omega_s} - k_l \partial_{-\boldsymbol{n}_{\text{int}}} \Theta \big|_{\Omega_l} \right] \right) \cdot \boldsymbol{n}_{\text{int}}, \qquad \text{on } \Gamma_{\text{int}}, \tag{1.1b}$$

$$\alpha = \begin{cases} k_s, & \text{on } \Omega_s, \\ k_l, & \text{on } \Omega_l. \end{cases} \tag{1.1c}$$

Here, the constants ℓ, k_s, and k_l are material parameters and $\boldsymbol{n}_{\text{int}}$ is the interface normal pointing from solid to liquid. The system of equations (1.1) is widely used since it allows to model the discontinuities of the Stefan problem with the parameter α and the jump term of the temperature gradient in the right hand side of Equation (1.1b). But at the same time, the discontinuities, the structure of Equation (1.1b) as an algebraic condition, and the time-dependent position of Γ_{int} pose significant challenges for the numerical simulation and optimal control of the system in a feasible and reliable manner. Therefore, new or improved algorithms and techniques to address these challenges are a valuable contribution of this thesis.

Of special importance in many applications of the Stefan problem is the position of the moving interface and for stabilization problems it can be the objective to steer the interface position. However, existing methods simplify the model to consider only one spatial dimension, one phase, or use an open-loop control for the two-dimensional two-phase version of the problem. Thus, the goal of this thesis is to conceptually derive a feedback stabilization algorithm for the two-dimensional two-phase Stefan problem. Furthermore, the goal is to compute this feedback stabilization and apply it in numerical experiments. These goals are pursued in order to go one step further on the path to having applicable methods for real-world applications.

To emphasize the importance of feedback stabilizations of the Stefan problem and related problems in a larger scope, it is important to mention that they can make it possible to automate many processes to an extend that they can be stabilized with a higher accuracy than humans could steer them manually. The same processes can also be improved with respect to their energy and time efficiency. During a solidification process, the quality of the resulting material can strongly depend on the shape of the interface [Van13]. Therefore, it is advantageous to use feedback stabilization to steer the interface position, e.g. along a flat trajectory.

To achieve this, either existing methods for the Stefan problem can be used as a starting point, or methods for similar problems can be adapted to the feedback stabilization of the two-dimensional two-phase Stefan problem. But, new challenging subproblems arise from the combination and implementation of the methods chosen in this thesis. The linear-quadratic regulator (LQR) approach is chosen for this purpose. It requires a linearization and discretization of the Stefan problem with special attention to the boundary condition at the interface. The LQR approach leads to large differential Riccati equations (DREs) with time-dependent coefficients due to the moving interface. For these specific non-autonomous DREs, only the theoretical concept of numerical solvers is available. However, existing implementations in the literature are limited to certain types of time-dependent coefficients and are not suitable for the Stefan problem. Another challenge arises in the simulation of the closed-loop system in which the computational feedback stabilization is applied. In this context, numerical problems may arise, e.g. in the form of blow-ups of the simulation.

1.2. State of the Art

Stefan Problem Since the original publications by J. Stefan [Ste89, Ste90, Ste91] as well as G. Lamé and B. P. Clapeyron [LC31], the Stefan problem is actively and intensively studied from the modeling to the theoretical analysis to the numerical simulation. An extensive historical survey can be found in the book by L. I. Rubenšteĭn [Rub71, Introduction: §1]. It is illustrated here that early works on the Stefan problem usually consider the one-dimensional or one-phase cases. This evolved to the consideration of higher dimensional and two-phase cases only decades after the original publication. The fact that there are several books dedicated to the Stefan problem emphasizes the importance of this topic. A selection of these books are [Rub71, Vis96, NCM11, Gup18, KK20b], which also treat the theoretical analysis of the Stefan problem.

The modeling of the Stefan problem entails the specific challenge to represent the interface and its velocity in a way that allows efficient numerical simulations, which are at the same time sufficiently accurate and can resolve the discontinuities. Many publications evolve around this challenge, like the comparison [BSVU06].

One possibility is to treat the interface implicitly as a mushy region of material. Then, no explicit representation or tracking of the inner boundary is necessary and the jump in the temperature gradient can be replaced with a steep but still continuous transition region. For instance, the enthalpy formulation of the Stefan problem, e.g. [Whi82a, Whi82b], has no sharp interface representation. As a consequence, the implementation of a numerical simulation of this model is relatively simple [Vab14, p. 219] but the exact interface position is unknown. This makes it more challenging to control the interface position.

In contrast, a sharp interface representation allows evaluating the interface position or velocity as explicit variables and comparing them to a desired position or velocity given a control setting. One way to treat the moving inner boundary explicitly is to represent it as the zero level set of a time-dependent, implicit function, e.g. [NPV91a, NPV91b] or [ZGT06]. A common method to approximate the temperature and the zero-level-set function is the extended finite element method (X-FEM). Here, the finite element method (FEM) functions are modified in a narrow band around the interface to deal with the discontinuity of the temperature gradient. For a detailed description of X-FEM see, e.g. [Ber10, BH11]. Another numerical method that is available in the literature to approximate the temperature and the zero-level-set function is presented in [CMOS97] where a finite difference scheme is used.

An alternative method for a sharp and explicit representation of the interface is to let the spatial discretization adaptively track the position of the interface. This approach on an adaptive mesh is also known as mesh movement or arbitrary Lagrangian-Eulerian (ALE) method. These methods are applied to different types of problems [DGH82] and have originally been developed for finite differences. However, they are also used together with FEM [HLZ81]. For the Stefan problem, the advantage of a moving mesh is that it can be aligned with the interface throughout the movement. Like this, discontinuities along the interface can be resolved. In [HZ07, Zie08] a moving mesh is combined with finite differences for the two-phase Stefan problem. Here, the interface is explicitly represented as a graph over one boundary of the two-dimensional domain. The same approach can be combined with the FEM as well [Bar16, BBHS18]. A similar formulation of the two-dimensional two-phase Stefan problem can be found in [BPS10, BPS13], [BMR01], and [BBHS18], where FEM and a moving mesh are used as well.

A sufficiently accurate and feasible method for the numerical simulation is a crucial prerequisite to develop open-loop control approaches for the Stefan problem. Several of the resources that are mentioned in this paragraph approach this challenge as well.

Open-loop Control of the Stefan Problem Several well established open-loop control strategies are applied to the Stefan problem in order to not just simulate the system but also steer it to a desired state. A general overview over these well established open-loop control strategies can be found in books about optimal control and partial differential equation (PDE)-constrained optimization, like [Trö10, LEG$^+$12].

In order to derive an open-loop control for the Stefan problem, requirements are:

- a control variable as an input to the system,
- a cost functional that penalizes the deviation of the system from a desired state,
- optimality conditions,
- an algorithm to numerically compute the control.

While there are many different possible choices for a control and cost functional, also depending on the systems that are coupled with the Stefan problem, the desired state is usually a particular interface position or trajectory. The optimality conditions that are present in the literature are either of first or second order nature. However, sufficient optimality conditions of second order have been derived theoretically only for a more general free boundary problem [ANS14, ANS15]. In contrast, several variants of fist-order optimality systems exist that are also implemented numerically. A common method is to use a quadratic tracking-type cost functional, which measures the control costs and the interface deviation. This is then combined with the formal Lagrange approach to derive an adjoint system and the gradient of the cost functional. With this, a gradient method can be used to approximate the optimal control. Some examples of this approach are contained in [ANS14, ANS15] and [Zie08], where a moving mesh is combined with finite differences. This is used to simulate the Stefan problem coupled with the NSE as well as Lorentz forces and a graph representation of the interface. The gradient method, which computes the optimal control is combined with a quadratic line minimization algorithm for the step size control. A similar approach is used in [Ber10, BH11] as well as [Mar12]. Here, the Stefan problem is not coupled to any other system and the interface is represented with a zero level set function. For the simulation, X-FEM is used instead of mesh movement and again, a gradient method to approximate the control. A combination of the previous approaches is developped in [BBHS18]. Here, the adjoint-based approach for the Stefan problem coupled with NSE and mesh movement is used in combination with FEM. The mesh movement is treated and coupled as an unknown to the system and incorporated in the adjoint system together with a graph representation of the interface. The gradient method is combined with a quadratic line minimization algorithm for the step size control. The open-loop approach in [BBHS18] is also used as a starting point to compute reference trajectories for closed-loop feedback stabilizations of the Stefan problem.

Closed-loop Control of the Stefan Problem Closed-loop, i.e. feedback stabilization, for the Stefan problem is a field that has been discussed in the literature only recently. Similar to the earlier work on the numerical simulation and open-loop control, it is a natural approach to first significantly simplify the problem and then to develop new methods for it before extending them to more challenging instances. In case of the Stefan problem, there are two common approaches to handle this non-linear problem in a formulation, which is more simple and thus more feasible for numerical computations.

The first approach is to consider the one-phase Stefan problem. An early derivation of a feedforward control is in [DPRM03]. Further, available in the literature are enthalpy-based [PBT12, PBT14], geometry-based [MC14], and backstepping-based [KDK19] methods, as well as an approach where the simulation model is built around the feedback stabilization's state definition [KDK16]. A direct extension of the methods for the one-phase Stefan problem to the two-phase case is not feasible since the incorporation of the coupling between the two phases is not straight forward.

The second approach to make the Stefan problem more simple is the consideration of the one-dimensional Stefan problem. First theoretical derivations of a closed-loop stabilization are in [RWW04, PBT10, KK19]. Lately, also an energy-shaping approach [KK20a], a backstepping method [EWW20], a flatness-based state feedback design [EWFR$^+$22], and model predictive control [EBRW21] have been developed for the one-dimensional problem. To the best of the authors' knowledge, non of these methods has been extended to the two-dimensional case and the only existing feedback stabilization approaches for the two-phase two-dimensional Stefan problem are [BBSS24, BBS22], which apply the LQR approach.

LQR and DRE Solver The method used in this work to derive a feedback stabilization for the Stefan problem is to apply the LQR approach. A key element of the LQR approach is to assemble and numerically solve a DRE.

Even though the LQR approach has not been applied to phase-change problems so far, it is well studied for related types of problems, e.g. convection-diffusion equations [Wei16]. It is important to note that the optimal feedback stabilization for a linear control system in state-space formulation with a quadratic cost functional is given by LQR, see e.g. [Loc01]. Although the convection-diffusion equations in [Wei16] and the Stefan problem are non-linear systems, they can be stabilized with this method under certain conditions. These conditions include that the system is stabilizable and the deviation from the desired trajectory is sufficiently small [Son98, Section 8.5].

A general introduction of the derivation of the LQR problem and the resulting DRE is available in [Rei72, BG91]. More particular, an extensive study on DREs and their numerous applications can be found in [AKFIJ03] or for generalized DREs in [KM90a]. A specific property of the Stefan problem is that the Stefan condition (1.1b) is an algebraic equation. Thus, the corresponding DREs have differential-algebraic structure as well.

For a general introduction to DREs, which result from differential-algebraic equations (DAEs), see [KM90b].

With this, the prerequisites are provided to derive the DRE theoretically and assemble it numerically. This numerical realization can be done, e.g. with the help of a linearization and a spatial discretization as it is demonstrated in [BBS22] using the FEM software FEniCS [LWH12]. Here, the moving mesh method is used and, thus, the spatial discretization is time-dependent. This results in a large-scale matrix-valued DRE with time-dependent coefficients, that is, a non-autonomous DRE. The next step is to approximate the numerical solution of the non-autonomous DRE with a sufficiently accurate and efficient method.

A crucial feature of efficient and feasible numerical methods to solve large-scale matrix-valued DREs is to approximate the solution by a low-rank factorization. It is proven theoretically for the autonomous DRE in [Sti18b] that this is a valid and sufficiently accurate approach. For the non-autonomous DRE a similar result is observed in numerical experiments only [BBSS24].

The numerical solution of large-scale matrix-valued DREs is well studied for the autonomous case. For this, well known methods, which use the low-rank structure of the numerical solution, are splitting schemes [Sti15a, Sti15b, Sti18a, OPW19, MOPP18], Rosenbrock and Peer methods [Men12, LMS15, Lan17, BL18] as well as the backward differentiation formulas (BDF) [BM04, Men12, LMS15, BM18]. Krylov subspace methods [BBH21, KM20, KS20, GHJK18], exponential integrators [LZL20], and an all-at-once space-time approach [BDS21] (computing a low-rank tensor format solution) for DREs have been developed recently as well. In [Men12, BM18] Rosenbrock and BDF methods and in [LMS15, Lan17, BL18] also Peer methods are studied for a non-autonomous DRE where the mass matrix is constant. In extension of this, splitting schemes and BDF methods are developed in [BBSS24] for non-autonomous DREs with a time-varying mass matrix. In contrast to the BDF methods, the splitting schemes require that the coefficients can be decomposed into a time-dependent scalar function times a constant matrix. Non-autonomous DREs resulting from the Stefan problem in combination with mesh movement exceed this case. The single matrix entries change very differently, possibly for all matrices and, thus, do not permit the decomposition that is required for the splitting schemes.

In summary, the non-autonomous DRE, which results from the Stefan problem, can be solved numerically with a suitable method like the non-autonomous BDF method. Then, the numerical solution can be used to compute a feedback gain matrix and, finally, a feedback stabilization during a closed-loop forward simulation.

Adaptive Time-Stepping for Related Problems The numerical simulation of closed-loop problems with free boundaries and moving interfaces, like the Stefan problem, is a very broad and active field of research, see e.g. [KDK19, LF20, JWN20, CRD20, KMC$^+$20], and there are numerous numerical methods available. Especially during the closed-loop simulation of the Stefan problem with quickly varying inputs, common methods like the implicit Euler method and the trapezoidal rule run the risk to break down as described in [FW18] or [BBSS24]. This very specific problem is treated rarely in the literature. A possible method to overcome this is the fractional-step-theta algorithm [BGP87] combined with adaptive time-stepping. It has successfully been applied to Navier-Stokes equations [MR14, MR15] as well as for fluid structure interaction (FSI) problems [FW18, Wic11, RW15] and the Stefan problem [BBSS24].

For the time-adaptivity, a common approach to determine the time step size are classical error estimates, e.g. heuristics, based on models of the actual error [GLS88, Tur99, JR10], like the one that is used in [FW18]. An alternative approach is residual-based, such as the dual-weighted residual, which gives the best results in [FW18]. However, the dual weighted residual has high computational costs, which are not feasible for the two-phase two-dimensional Stefan problem. Since here the purpose of the time-adaptivity is to prevent numerical issues that occur when the input is quickly varying, a specialized approach is developed in [BBSS24]. This approach for the time-adaptivity is to adapt an error-based heuristic to monitor the change in the input instead of the error estimate.

Additionally, for specific problems that have moving inner boundaries as well, there are other specialized time-adaptive strategies available. One example is an adaptive time-stepping applied for a two-phase flow with a proportional–integral–derivative (PID) control in [AG17]. Another example is [SUS15], where time and spatial discretizations are used that are adaptive with respect to Courant-Friedrichs-Lewy and non-self-intersection conditions for a dendritic Stefan problem.

1.3. Outline

The outline of this thesis is in line with the objective of the derivation and application of feedback stabilizations for the two-phase two-dimensional Stefan problem. This means that for the core algorithm of this thesis, which is proposed in Algorithm 3.1, the outline follows the path starting from the existing methods that are used as a starting point, then to the adapted and newly developed methods leading to the demonstration of the effectiveness of the proposed algorithm.

Chapter 2: Mathematical Background for the Feedback Stabilization of the Stefan Problem Although the field of feedback stabilization for the Stefan problem has only recently been discussed, there are well-studied methods for related problems. This chapter reviews various numerical methods from the existing literature for deriving, computing, and applying feedback stabilization to the Stefan problem and the subproblems that arise in this context. Some of these subproblems are already discussed in the literature, such as numerical simulation and open-loop control of the Stefan problem

(Sections 2.1 and 2.2). Other subproblems, such as LQR and DRE solvers, have been well studied, e.g. for convection-diffusion problems (Sections 2.3 and 2.4). Therefore, the approaches to these subproblems can be adapted accordingly.

In this chapter the methods for the feedback stabilization of the Stefan problem are collected, which are then used, adapted or extended in Chapter 3.

Chapter 3: Methods for the Feedback Stabilization of the Stefan Problem The central chapter of this thesis is dedicated to the main objective, namely to derive, compute and apply a feedback stabilization for the two-phase two-dimensional Stefan problem. One of the key contributions of this thesis is the core algorithm for this objective. This algorithm is formulated in Section 3.1. Each step of this algorithm has its own subproblems and the need for new methods. For the Stefan problem, the methods of Chapter 2 are adapted or extended.

The main feature of the Stefan problem that requires special treatment is the moving interface. This requires special attention to the associated boundary conditions in the spatial discretization (Section 3.2). It also causes the matrices in the resulting DRE to be time-dependent when using the LQR approach for the Stefan problem (Section 3.1). To solve these DREs numerically, non-autonomous BDF methods are presented in Section 3.3. Finally, special numerical problems can arise from the simulation of the Stefan problem with applied feedback stabilization. These can be treated using adaptive time stepping combined with a fractional step theta scheme (Section 3.4). Here, an error-based indicator function and two input-based indicator functions are described. These are specifically tailored to the closed-loop simulation of the Stefan problem.

Chapter 4: Numerical Results This chapter examines the numerical behavior and performance of the core results of this thesis. Each proposed method for the related subproblems has specific numerical properties. They are studied and compared with existing methods in several numerical experiments to demonstrate and verify the effectiveness of the implementations. All of the results are made reproducible by providing the codes and the data used for the experiments that are presented in this chapter (Section 4.1).

The solution of the non-autonomous DRE is the main computational effort for the computation of feedback stabilization for the Stefan problem. Therefore, the accuracy and efficiency of the non-autonomous BDF methods are investigated in Section 4.2. Once the feedback gain matrix is computed, the next crucial step is its reliable and efficient application in a forward simulation of the non-linear closed-loop Stefan problem. For this purpose, the behavior of the time-adaptive fractional-step theta algorithm is described in detail in Section 4.3. Lastly, with these two components settled, the LQR problem design is discussed in terms of different choices for problem parameters such as cost functional weight, inputs, outputs, and desired trajectories (Section 4.4).

Chapter 5: Conclusion & Outlook Finally, in the last chapter, Section 5.1 summarizes the main contributions of this thesis and points out how these contributions go beyond the existing literature. In Section 5.2, the limitations of the presented results and aspects that go beyond the scope of this thesis are highlighted. These limitations and aspects point in the direction of further research. A particularly promising extension of the work presented in this thesis is the inclusion of flow in the liquid phase, i.e. the coupling of the Stefan problem with the Stokes or Navier-Stokes equations. The general framework for this coupling is already described in this section, together with the relevant equations and the block structure of the resulting matrices.

CHAPTER 2

MATHEMATICAL BACKGROUND FOR THE FEEDBACK STABILIZATION OF THE STEFAN PROBLEM

Contents

This chapter gathers the methods that are then used, adapted, or extended in Chapter 3 for the feedback stabilization of the Stefan problem.

Even though the field of feedback stabilization for the Stefan problem has been discussed only recently, there are well studied methods for related problems. In the process of deriving, computing, and applying a feedback stabilization to the Stefan problem, several subproblems arise. Some of these subproblems, like the numerical simulation and the open-loop control of the Stefan problem, are already discussed in the literature (Sections 2.1 and 2.2). Other subproblems, such as LQR and DRE solvers, are well studied, e.g. for convection-diffusion problems (Sections 2.3 and 2.4), but not yet for the Stefan problem. Thus, the methods that are used to approach these subproblems can be adapted or extended accordingly.

2.1. Modeling and Simulation of the Stefan Problem

This section explains the equations that describe the Stefan problem, its boundary conditions, and initial values (Section 2.1.1). For the purpose of steering the interface position with open-loop controls and closed-loop stabilizations, an important distinction is the choice of a sharp interface representation and mesh movement (Section 2.1.3). This choice influences also the discretization in space and time (Section 2.1.2), in particular the spatial discretization and the FEM formulation. At the end of this section, the general framework for time-stepping schemes is presented (Section 2.1.4), with which the numerical simulation can be implemented.

2.1.1. Domain and Equations

The Stefan problem is described by equations characterizing the temperature and interface movement as in [BBHS18, BBS22]. In order to apply open-loop and closed-loop inputs to this problem a sharp interface representation is chosen, which makes the interface position available explicitly.

At each time $t \in [0, t_{\text{end}}]$, the Stefan problem is modeled on the two-dimensional domain $\Omega(t) \subset \mathbb{R}^2$. For this, one instance is illustrated in Figure 2.1. The domain $\Omega(t)$ is split into the two regions corresponding to the two phases. These are the region $\Omega_s(t)$ where the material is in its solid phase and accordingly, the region $\Omega_l(t)$ related to the liquid phase. The two phases are separated by the interface $\Gamma_{\text{int}}(t)$. This inner phase-boundary can move such that its position is time-dependent. Thus, also the two phases $\Omega_s(t)$ and $\Omega_l(t)$ are time-dependent and, as a consequence, so is the whole domain $\Omega(t)$ and its boundary regions. The boundary of $\Omega(t)$ is separated into $\Gamma_u(t)$, $\Gamma_{\text{cool}}(t)$ and $\Gamma_{\text{N}}(t)$ as depicted in Figure 2.1. Note that the outer shape of $\Omega(t)$ is constant for the realization chosen in this manuscript. Thus, the time-dependence of $\Omega(t)$ is not absolutely necessary even though its sub-domains are time-dependent. However, the time-dependence is kept in the notation in order to not restrict the presented methods to this case.

Compared to the definition of the Stefan problem from [BBHS18], a more compact form is used in this work, i.e. omitting the coupling with the Navier-Stokes equations. The interface graph formulation is only used for the open-loop control approach in Section 2.2 and is specified there. While the Navier-Stokes equations alone add additional algebraic constraints, making the DAE harder to classify, both the Navier-Stokes equations and the interface graph formulation add more non-linearities to the problem. In order to develop the general numerical strategy, the first step is to study the feedback

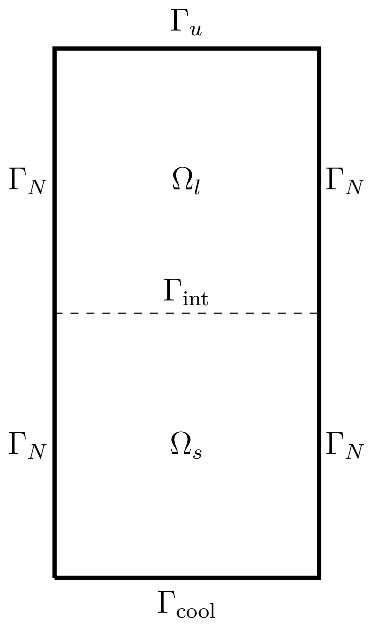

Figure 2.1.: One instance of the domain $\Omega(t) \subset \mathbb{R}^2$ of the Stefan problem.

stabilization problem for this simplified setting without these couplings, before delving into the additional technical challenges of the full problem formulation.

The temperature is denoted as $\Theta(t)$ and modeled with the partial differential Equation (2.1), which is a more detailed version of Equation (1.1):

$$\dot{\Theta} - \Upsilon \cdot \nabla\Theta - \alpha\Delta\Theta = 0, \qquad \text{on } (0, t_{\text{end}}] \times \Omega, \tag{2.1a}$$
$$\partial_{\boldsymbol{n}}\Theta = u, \qquad \text{on } (0, t_{\text{end}}] \times \Gamma_u, \tag{2.1b}$$
$$\Theta = \Theta_{\text{cool}}, \qquad \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{cool}}, \tag{2.1c}$$
$$\Theta = \Theta_{\text{melt}}, \qquad \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{int}}, \tag{2.1d}$$
$$\partial_{\boldsymbol{n}}\Theta = 0, \qquad \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{N}}, \tag{2.1e}$$
$$\Theta(0) = \Theta_0, \qquad \text{on } \Omega. \tag{2.1f}$$

In Equation (2.1b), the input $u(t)$ is applied as a Neumann condition on the control boundary $\Gamma_u(t)$. The Equations (2.1c) and (2.1d) describe the Dirichlet conditions on the cooling boundary $\Gamma_{\text{cool}}(t)$ and the interface $\Gamma_{\text{int}}(t)$ with the constants Θ_{cool} and Θ_{melt}, respectively. Equation (2.1f) presents the initial condition with the initial temperature distribution Θ_0. The heat conductivities in the solid phase k_s and in the liquid phase k_l are collected in $\alpha(\Theta(t))$:

$$\alpha = \begin{cases} k_s, & \text{on } \Omega_s, \\ k_l, & \text{on } \Omega_l. \end{cases} \tag{2.2}$$

In Equation (2.1a), the temperature is coupled with the extended interface movement $\Upsilon(\nabla\Theta(t))$. For each $t \in (0, t_{\text{end}}]$, $\Upsilon(\nabla\Theta(t))$ is modeled by a system of algebraic equations. This means that these equations do not contain time derivatives:

$$\Delta\Upsilon = 0, \qquad \text{on } \Omega, \tag{2.3a}$$
$$\Upsilon = \left(\frac{1}{\ell}[k(\nabla\Theta(t))]_l^s\right) \cdot \boldsymbol{n}_{\text{int}}, \qquad \text{on } \Gamma_{\text{int}}, \tag{2.3b}$$
$$\Upsilon = 0, \qquad \text{on } \Gamma_{\text{cool}} \cup \Gamma_u, \tag{2.3c}$$
$$\Upsilon \cdot \boldsymbol{n} = 0, \qquad \text{on } \Gamma_{\text{N}}, \tag{2.3d}$$
$$\Upsilon(0) = 0, \qquad \text{on } \Omega, \tag{2.3e}$$

On the interface, $\Upsilon\big|_{\Gamma_{\text{int}}}(\nabla\Theta(t)) = \Upsilon_{\text{int}}(\nabla\Theta(t))$ is the interface movement in normal direction, where $\boldsymbol{n}_{\text{int}}(t)$ is the unit normal vector pointing from $\Omega_s(t)$ to $\Omega_l(t)$. $\Upsilon_{\text{int}}(\nabla\Theta(t))$ is coupled to $\Theta(t)$ through the Stefan condition (2.3b). Here, ℓ is the latent heat constant and

$$[k(\nabla\Theta)]_l^s = k_s\partial_{\boldsymbol{n}_{\text{int}}}\Theta\big|_{\Omega_s} - k_l\partial_{-\boldsymbol{n}_{\text{int}}}\Theta\big|_{\Omega_l} \tag{2.4}$$

is the jump of the temperature gradient along $\Gamma_{\text{int}}(t)$. Equations (2.3c) and (2.3d) ensure that the outer boundaries of $\Omega(t)$ do not move such that the outer shape of the domain does not change.

The extended interface movement $\Upsilon(\nabla\Theta(t))$ is used in Section 2.1.2 for the mesh movement. The Stefan problem, which is described by the system of Equations (2.1) and (2.3), is a non-linear system of DAEs on a time-varying domain.

The non-linearities arise due to the coupling of $\Theta(t)$ and $\Upsilon_{\text{int}}(\nabla\Theta(t))$ in Equation (2.3b) and the two temperature-dependent phases in the definition of $\alpha(\Theta(t))$ in Equation (2.2). Hence, the dependence of $\Upsilon_{\text{int}}(\nabla\Theta(t))$ and $\alpha(\Theta(t))$ on $\Theta(t)$ leads to the non-linearity of the terms $\Upsilon(\nabla\Theta(t)) \cdot \nabla\Theta(t)$ and $\alpha(\Theta(t))\Delta\Theta(t)$ in Equation (2.1a).

The Stefan problem formulation presented in this section is discretized in the next section such that the Equations (2.1) and (2.3) can be solved numerically. Furthermore, the non-linearities are treated and the Stefan problem is formulated in a more general representation to make the LQR approach applicable.

2.1.2. Discretization and Linearization

In order to apply the LQR approach in Section 3.1, the Stefan problem needs to be formulated in a standard state-space format, which is linear and semi-discretized in space. Thus, this section describes how to transform the coupled DAE system of Equations (2.1) and (2.3) into the linear state-space formulation

$$\begin{aligned} \mathscr{M}\dot{x}^h &= \mathscr{A}x^h + \hat{\mathscr{B}}u^h, \\ y^h &= \mathscr{C}x^h. \end{aligned} \tag{2.5}$$

To generate the square matrices $\mathscr{A}(t), \mathscr{M}(t) \in \mathbb{R}^{n\times n}$, the input matrix $\hat{\mathscr{B}}(t) \in \mathbb{R}^{n\times m}$, and output matrix $\mathscr{C}(t) \in \mathbb{R}^{p\times n}$, the Stefan problem is linearized and spatially discretized (indicated by the index $(\cdot)^h$). Further, special care is taken on how the boundary conditions (2.1d) and (2.3b) are treated in the definition of the matrices for Equation (2.5) since they are of particular importance for the feedback stabilization problem (see Section 2.1.3).

For the spatial discretization, FEM is used on a mesh of triangular cells $\mathscr{Q}^h(t) = \{Q(t)\}$ that changes over time, driven by the movement of the interface (see Section 2.1.3).

In order to derive a linearization of the Stefan problem, a reference trajectory is used. This trajectory can be generated by applying an open-loop control approach to the non-linear problem (see Section 2.2). This (desired) reference trajectory contains the semi-discrete reference solutions $\tilde{\Theta}^h(t)$, $\tilde{\Upsilon}^h(\nabla\tilde{\Theta}^h(t))$, $\Gamma_{\text{int,ref}}(t)$, and the heat conductivity $\tilde{\alpha}(\tilde{\Theta}^h(t))$.

Using these reference states and the spatial discretization, a linearized, semi-discrete version of Equation (2.1a) is derived by using the semi-discrete states $(\Theta^h(t), \Upsilon^h(\nabla\Theta^h(t)))$. In particular, $\Theta^h(t)$ is replaced by $\tilde{\Theta}^h(t)$ in the convection term and $\alpha(\Theta^h(t))$ is replaced by the reference heat conductivity $\tilde{\alpha}(\tilde{\Theta}^h(t))$:

$$\dot{\Theta}^h - \Upsilon^h \cdot \nabla\tilde{\Theta}^h - \tilde{\alpha}\Delta\Theta^h = 0, \qquad \text{on } (0, t_{\text{end}}] \times \Omega. \tag{2.6}$$

To formulate the Stefan problem into the standard state-space format of Equation (2.5), i.e. to derive the matrices for this format, the semi-discrete variational formulations of the Stefan problem are posed. The underlying equations are Equation (2.6), together with the boundary and initial conditions from Equations (2.1) and (2.3):

$$\begin{aligned}
0 &= \int_\Omega \dot{\Theta}^h \cdot \mathsf{v}^h \mathrm{d}x - \int_\Omega \Upsilon^h \cdot \nabla\tilde{\Theta}^h \cdot \mathsf{v}^h \mathrm{d}x + \int_\Omega \alpha\nabla\Theta^h \cdot \nabla \mathsf{v}^h \mathrm{d}x - \int_{\Gamma_u} k_l u^h \cdot \mathsf{v}^h \mathrm{d}s, \\
0 &= -\int_\Omega \nabla\Upsilon^h \cdot \nabla\hat{\mathsf{v}}^h \mathrm{d}x + \int_{\Gamma_{\text{int}}} \Upsilon^h \cdot \hat{\mathsf{v}}^h \mathrm{d}s - \int_{\Gamma_{\text{int}}} \frac{1}{\ell}[k(\nabla\Theta^h)]_l^s \cdot \boldsymbol{n}_{\text{int}} \cdot \hat{\mathsf{v}}^h \mathrm{d}s.
\end{aligned}$$

The semi-discrete test functions are denoted with $\mathsf{v}^h(t)$ and $\hat{\mathsf{v}}^h(t)$ and these variational formulations are reformulated into a matrix-based form:

$$\begin{aligned}
M_\Theta \dot{\Theta}^h &= A_{\Theta\Theta}\Theta^h + A_{\Theta\Upsilon}\Upsilon^h + B_\Theta u^h, && \text{on } (0, t_{\text{end}}] \times \Omega, \\
0 &= A_{\Upsilon\Theta}\Theta^h + A_{\Upsilon\Upsilon}\Upsilon^h, && \text{on } (0, t_{\text{end}}] \times \Omega.
\end{aligned}$$

Here, the coefficient matrices are defined via the inner products

$$\begin{aligned}
\langle M_\Theta \Theta^h, \mathsf{v}^h \rangle &= \int\limits_\Omega \Theta^h \cdot \mathsf{v}^h \,\mathrm{d}x, \\
\langle A_{\Theta\Theta} \Theta^h, \mathsf{v}^h \rangle &= -\int\limits_\Omega \alpha \nabla \Theta^h \cdot \nabla \mathsf{v}^h \,\mathrm{d}x, \\
\langle A_{\Theta\Upsilon} \Upsilon^h, \mathsf{v}^h \rangle &= \int\limits_\Omega \Upsilon^h \cdot \nabla \tilde{\Theta}^h \cdot \mathsf{v}^h \,\mathrm{d}x, \\
\langle A_{\Upsilon\Upsilon} \Upsilon^h, \hat{\mathsf{v}}^h \rangle &= \int\limits_\Omega \nabla \Upsilon^h \cdot \nabla \hat{\mathsf{v}}^h \mathrm{d}x - \int\limits_{\Gamma_{\text{int}}} \Upsilon^h \cdot \hat{\mathsf{v}}^h \mathrm{d}s \\
\langle A_{\Upsilon\Theta} \Theta^h, \hat{\mathsf{v}}^h \rangle &= \int\limits_{\Gamma_{\text{int}}} \frac{1}{\ell} [k(\nabla \Theta^h)]_l^s \cdot \boldsymbol{n}_{\text{int}} \cdot \hat{\mathsf{v}}^h \mathrm{d}s, \\
\langle B_\Theta u^h, \mathsf{v}^h \rangle &= \int\limits_{\Gamma_u} k_l u^h \cdot \mathsf{v}^h \mathrm{d}s.
\end{aligned} \tag{2.7}$$

With these definitions, the semi-discrete linearized Stefan problem can be formulated in the format of Equation (2.5) as:

$$\begin{aligned}
\begin{bmatrix} M_\Theta & 0 \\ 0 & 0 \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \Theta^h \\ \Upsilon^h \end{bmatrix} &= \begin{bmatrix} A_{\Theta\Theta} & A_{\Theta\Upsilon} \\ A_{\Upsilon\Theta} & A_{\Upsilon\Upsilon} \end{bmatrix} \begin{bmatrix} \Theta^h \\ \Upsilon^h \end{bmatrix} + \begin{bmatrix} B_\Theta \\ 0 \end{bmatrix} \mathbf{u}^h, \\
\mathbf{y}^h &= \begin{bmatrix} C_\Theta & 0 \end{bmatrix} \begin{bmatrix} \Theta^h \\ \Upsilon^h \end{bmatrix}.
\end{aligned} \tag{2.8}$$

There are several plausible choices for the output matrix $C_\Theta(t)$, some of which are introduced in Section 4.4.3. With the zero-blocks in Equation (2.8), the DAE structure is clearly visible. Here it is important to note that $A_{\Upsilon\Upsilon}(t)$ is always non-singular since it represents the Poisson operator with a Dirichlet boundary part at each time instance t. Further, the mass matrix with respect to the temperature $M_\Theta(t)$ is symmetric positive definite and, in particular, always non-singular as well. In particular, Equation (2.8) is a DAE in semi-explicit form of differential index 1 (see e.g. [KM06]), for which the implicit index-reduction techniques from [FRM08] can be applied.

In other words, the Schur complement is used to remove the algebraic conditions. This results in an equivalent formulation of the Stefan problem on the hidden manifold [KM06], i.e. as an ordinary differential equation (ODE), with the coefficient matrices

$$\begin{aligned}\mathscr{M}(t) &= M_{\Theta}(t),\\ \mathscr{A}(t) &= A_{\Theta\Theta}(t) - A_{\Theta\Upsilon}(t)A_{\Upsilon\Upsilon}^{-1}(t)A_{\Upsilon\Theta}(t),\\ \hat{\mathscr{B}}(t) &= B_{\Theta}(t),\\ \mathscr{C}(t) &= C_{\Theta}(t).\end{aligned} \tag{2.9}$$

With the matrices from Equation (2.9), the Stefan problem can be transformed into the formulation of Equation (2.5) and the LQR approach can be applied for the computation of a feedback stabilization. In order to have a computationally efficient method, the sparse structure of the matrices is preserved and the generally dense Schur complement for $\mathscr{A}(t)$ is never computed explicitly. Instead, $\mathscr{A}(t)$ is applied implicitly and the matrices from Equation (2.8) are used. For details, see [FRM08], or the numerical implementation in [SKB].

Furthermore, to simulate the Stefan problem forward in time, a time discretization is used with the reference time steps

$$\begin{aligned}0 = t_0 < t_1 < \ldots < t_{n_t-1} < t_{n_t} = t_{\text{end}},\\ \mathscr{T}_{\text{fwd}}^{\text{ref}} = \{t_0, t_1, \ldots, t_{n_t-1}, t_{n_t}\}.\end{aligned} \tag{2.10}$$

The related numerical methods are described in Sections 2.1.4 and 3.4. The LQR approach requires to solve DREs backwards in time. For this, the same time-steps $\hat{t}_k = t_{n_t-k}$ are used in reversed order:

$$\begin{aligned}t_{\text{end}} = \hat{t}_0 > \hat{t}_1 > \ldots > \hat{t}_{n_t-1} > \hat{t}_{n_t} = t_0,\\ \mathscr{T}_{\text{bwd}} = \{\hat{t}_0, \hat{t}_1, \ldots, \hat{t}_{n_t-1}, \hat{t}_{n_t}\}.\end{aligned} \tag{2.11}$$

The same time-steps are used because the matrices from Equation (2.9), which form the coefficients of the differential Riccati equations, are assembled during the forward simulation. When a time-adaptive method is applied in the forward simulation, additional time-steps can be added to $\mathscr{T}_{\text{fwd}}^{\text{ref}}$.

2.1.3. Interface Representation and Mesh Movement

Having in mind the stabilization problem, which aims to steer the interface position to a desired trajectory, the interface $\Gamma_{\text{int}}(t)$ itself is represented explicitly and sharply through facets that are aligned with $\Gamma_{\text{int}}(t)$ [BBHS18]. In particular, in the initial partition, the facets of the mesh are aligned with $\Gamma_{\text{int}}(t)$. To have these facets aligned with the interface in every time step, the vertices on the interface are moved with $\Upsilon_{\text{int}}(\nabla\Theta(t))$ in normal direction. In order to prevent the mesh from degrading, i.e. avoid extreme cell deformations, or mesh tangling, $\Upsilon_{\text{int}}(\nabla\Theta(t))$ is smoothly extended to $\Upsilon(\nabla\Theta(t))$ on the whole domain $\Omega(t)$ with Equation (2.3). Figure 2.2 displays a possible mesh for two different interface positions.

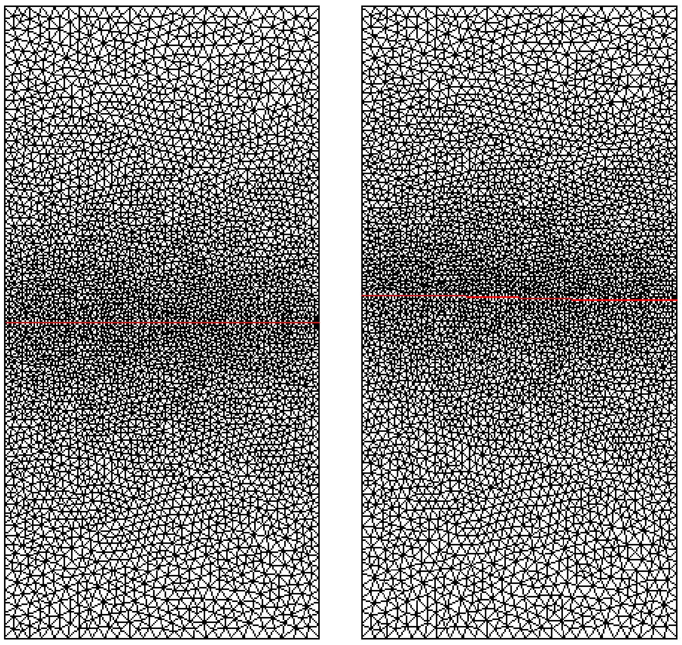

Figure 2.2.: Two instances of the mesh with the interface marked as a red line.

To couple Equations (2.1) and (2.3) correctly, it is important to note that the interface is a non-material surface. This means that the movement $\Upsilon_{\text{int}}(\nabla\Theta(t))$ of the interface and the mesh movement $\Upsilon(\nabla\Theta(t))$ are not related to the movement of any physical material points. As pointed out in [BPS13], the non-material movement $\Upsilon(\nabla\Theta(t))$ needs to be separated from the material movement in $\Theta(t)$ with an advection term $-\Upsilon(\nabla\Theta(t)) \cdot \nabla\Theta(t)$ in Equation (2.1a).

With the spatial discretization, which is described above, and the mesh movement, the semi-discrete interface is represented explicitly as the facets of the mesh that are aligned with it. However, this way, the interface position is not available as a state of the Stefan problem. Thus, an alternative explicit and sharp representation is used as well.

In the context of this work, it is assumed that the interface can be represented as a one-dimensional graph $g(t, x_1)$, i.e. a function over the boundary Γ_{cool} at the bottom of the domain:

$$\Gamma_{\text{int}}(t) = \left\{ \begin{bmatrix} x_1 \\ g(t, x_1) \end{bmatrix} : x_1 \in \Gamma_{\text{cool}} \right\}, \qquad \text{with } g\colon [0, t_{\text{end}}] \times \Gamma_{\text{cool}} \to \mathbb{R},$$

where the derivatives of $g(t, x_1)$ are abbreviated with

$$g_{x_1} := \frac{dg}{dx_1}, \qquad g_t := \dot{g}.$$

To map from Γ_{cool} to the interface $\Gamma_{\text{int}}(t)$, the function $\Phi\colon [0, t_{\text{end}}] \times \Gamma_{\text{cool}} \to [0, t_{\text{end}}] \times \Gamma_{\text{int}}$ is used, which is defined as

$$\Phi(t, x_1) := \left(t, \begin{bmatrix} x_1 \\ g(t, x_1) \end{bmatrix} \right).$$

The unit normal vector $\boldsymbol{n}_{\text{int}}(t)$ along the interface $\Gamma_{\text{int}}(t)$ points from the solid to the liquid phase. It can be expressed as (see [Zie08, Sec. 2.1])

$$\boldsymbol{n}_{\text{int}}(t, x_1) = \frac{1}{\sqrt{1 + g_{x_1}(t, x_1)^2}} \begin{bmatrix} -g_{x_1}(t, x_1) \\ 1 \end{bmatrix}. \tag{2.12}$$

With Equation (2.12), the velocity $\Upsilon_{\text{int}}(\nabla\Theta(t))$ of the interface $\Gamma_{\text{int}}(t)$ in normal direction $\boldsymbol{n}_{\text{int}}(t)$ at every point x_1 on the interface can be expressed as

$$\begin{aligned} \Upsilon_{\text{int}}(\nabla\Theta(t), x_1) &= \partial_t \begin{bmatrix} x_1 \\ g(t, x_1) \end{bmatrix} \cdot \boldsymbol{n}_{\text{int}}(t, x_1) = \begin{bmatrix} 0 \\ g_t(t, x_1) \end{bmatrix} \cdot \boldsymbol{n}_{\text{int}}(t, x_1) \\ &= \frac{g_t(t, x_1)}{\sqrt{1 + g_{x_1}(t, x_1)^2}} = g_t(t, x_1) \boldsymbol{n}_{\text{int}}(t, x_1) \cdot \boldsymbol{e_2}, \end{aligned} \tag{2.13}$$

where $\boldsymbol{e_2} = [0, 1]^\mathsf{T}$ is the unit vector in vertical direction. Using Equation (2.13), the Stefan condition (2.3b) can be reformulated to

$$\sqrt{1 + g_{x_1}^2} \cdot [k(\nabla\Theta)]_l^s \circ \Phi = \ell \cdot g_t, \qquad \text{on } \Gamma_{\text{cool}}. \tag{2.14}$$

The interface representation with the graph $g(t, x_1)$ is used as an output to track the interface position and deviation from the desired trajectory in the open-loop control approach in Section 2.2. For this approach, the additional non-linearities introduced by the graph representation are of no concern. For the closed-loop stabilization approach in Section 2.3, however, the Stefan problem is first linearized, and the non-linearities introduced by the graph representation pose additional challenges. Therefore, this representation is not used for the closed-loop stabilization approach.

Nevertheless, the numerical simulation of the Stefan problem is performed with the non-linear model, where different time-stepping schemes can be applied.

2.1.4. Time-stepping Schemes

The time-stepping schemes that can be used to simulate the Stefan problem in different variations, i.e. unstabilized, open-loop, or closed-loop stabilized, are first presented for an abstract system before the Stefan problem is plugged into these schemes. The abstract, stabilized, semi-discrete, dynamical system reads

$$\dot{\zeta}^h(t) = f^h(t; \zeta^h(t), u^h(t)) \tag{2.15}$$

on a finite time horizon $t \in \mathscr{I} = [t_0, t_{\text{end}}]$ and a domain $\Omega \subset \mathbb{R}^d$ of dimension $d > 0$. The index $(\cdot)^h$ indicates that the system (2.15), the state $\zeta^h(t)$, and the input $u^h(t)$ are discretized in space (see Section 2.1.2). The function $f^h\colon [t_0, t_{\text{end}}] \times \mathbb{R}^n \times \mathbb{R}^p \mapsto \mathbb{R}^n$ represents the Stefan problem, that is, Equations (2.1) and (2.3), in semi-discretized form.

For the time discretization, n_t sub-intervals $[t_{k-1}, t_k]$ are used to partition the time-interval $[t_0, t_{\text{end}}]$ with the time grid $\mathscr{T}_{\text{fwd}}^{\text{ref}}$ as in Equation (2.10):

$$\begin{aligned} \mathscr{T}_{\text{fwd}}^{\text{ref}} &= \{t_0, \dots t_{n_t} = t_{\text{end}}\}, \\ \tau_k &= t_k - t_{k-1}. \end{aligned}$$

With these time steps, the fully discrete approximations (ζ_k, u_k) are defined by

$$\begin{aligned} \zeta_k &\approx \zeta^h(t_k), \\ u_k &\approx u^h(t_k). \end{aligned}$$

For the computation of these discrete approximations, several time-stepping schemes are formulated in the same framework by denoting the parameter $\Sigma \in [0, 1]$. With this, the system (2.15) in discrete form can be approximated by

$$\frac{\zeta_k - \zeta_{k-1}}{\tau_k} = \Sigma f^h(\zeta_k, u_k) + (1 - \Sigma) f^h(\zeta_{k-1}, u_{k-1}). \tag{2.16}$$

In case the function f^h is non-linear, Equation (2.16) can be solved with, e.g. a Newton method, otherwise with a direct linear solver. As a result, several time-stepping schemes can be obtained by choosing different parameters Σ. For $\Sigma = 0$, Equation (2.16) is the explicit Euler scheme and for $\Sigma = 1$, the implicit Euler scheme. With $\Sigma = 0.5$, the scheme corresponds to a trapezoidal rule[1]. These are some straight forward choices for Σ. Nevertheless, the described framework can be used to formulate more time-stepping schemes as well, where Σ can vary in between time steps. This is used in Section 3.4 for time-adaptive fractional-step-theta schemes. For now, the implicit Euler scheme ($\Sigma = 1$) is used for the simulations that are performed during the computation of the open-loop control, which is described in the next section.

[1]In different PDE contexts, like in [FW18], the scheme with $\Sigma = 0.5$ is also called Crank-Nicolson

2.2. Open-loop Control of the Stefan Problem

For a non-linear PDE system, like the Stefan problem, an open-loop control problem can be defined. The aim is to use a control to steer the system to a desired state. In this case, the desired state is a specific interface position and the control is applied in Equation (2.1b).

Similar to the previous section, the open-loop control problem is described in this section for an abstract PDE system. This approach closely follows [BBHS18] which in turn uses results from [Zie08, Ber10, Bar16]. Then the Stefan problem is plugged into this more general framework and a cost functional is defined. With this, an adjoint system is derived via a Lagrange functional. More details on the Lagrange formalism for the optimal control of PDEs can be found in [Trö10]. This adjoint system is then solved numerically together with Equations (2.1), (2.3) and (2.14), which describes the Stefan problem and the interface graph. In a projected gradient algorithm combined with quadratic line minimization, a control is computed that moves the interface to a desired position.

For the abstract control problem, the state ζ is an element of the state-space $\mathscr{V}_x$ and the control u is an element of the set of admissible controls $\mathscr{V}_u^{\mathrm{ad}}$, which is a subset of the control space $\mathscr{V}_u$. The abstract state and control spaces are defined for the Stefan problem later in this section. With these definitions, the optimal control problem is stated as

$$\begin{aligned} &\min_{\zeta\in\mathscr{V}_x, u\in\mathscr{V}_u} \tilde{J}(\zeta, u) \\ \text{subject to}& \\ &f(\zeta, u) = 0, \\ &u \in \mathscr{V}_u^{\mathrm{ad}} \subset \mathscr{V}_u. \end{aligned} \tag{2.17}$$

It is assumed that for each $u \in \mathscr{V}_u^{\mathrm{ad}}$ there exists a unique state $\zeta \in \mathscr{V}_x$ that satisfies the constraint $f(\zeta, u) = 0$.

In the present Stefan problem, the state is defined as the tuple $\zeta = [\varTheta, \Upsilon, g]$ and the control constraint $u \in \mathscr{V}_u^{\mathrm{ad}}$ sets restrictions on the control. The set of admissible controls $\mathscr{V}_u^{\mathrm{ad}}$ is usually a convex subset of $\mathscr{V}_u$. In the case $\mathscr{V}_u^{\mathrm{ad}} = \mathscr{V}_u$, the problem is unrestricted. The state equation $f(\zeta, u) = 0$ connects the state and the control and represents the PDE constraints of the Stefan problem defined in Equations (2.1), (2.3) and (2.14).

The objective of the cost functional is to measure the deviation of the interface trajectory from a desired interface trajectory and the control costs. The cost functional is defined as

$$\begin{aligned}\tilde{J}(\zeta,u) :=& \frac{\lambda_{\text{end}}}{2} \int\limits_{\Gamma_{\text{cool}}} \left(g(t_{\text{end}},x_1) - g_d(t_{\text{end}},x_1)\right)^2 \,\mathrm{d}x_1 \\ &+ \frac{\lambda_{\text{all}}}{2} \int\limits_0^{t_{\text{end}}} \int\limits_{\Gamma_{\text{cool}}} \left(g(t,x_1) - g_d(t,x_1)\right)^2 \,\mathrm{d}x_1\mathrm{d}t + \frac{\lambda_u}{2} \int\limits_0^{t_{\text{end}}} \int\limits_{\Gamma_u} (u(t,x))^2 \,\mathrm{d}x\mathrm{d}t.\end{aligned} \tag{2.18}$$

In detail, the deviation of the interface trajectory is measured in the cost functional as the difference between the interface graph $g(t,x_1)$ and the interface graph $g_d(t,x_1)$ corresponding to the desired interface trajectory. The scalars $\lambda_{\text{end}}, \lambda_{\text{all}}, \lambda_u \geq 0$ are weight factors for the cost functional $\tilde{J}(\zeta,u)$: The first term aims to steer the interface position to the desired position at the terminal time t_{end}, the second term penalizes the interface deviation over the entire time horizon $(0,t_{\text{end}}]$, while the third term models control costs and has a regularizing effect [Trö10, p. 3].

Due to the assumption that a unique state ζ exists for every control $u \in \mathscr{V}_u^{\text{ad}}$, the cost functional can be simplified to $J(u) := \tilde{J}(\zeta(u),u)$. As a consequence the optimal control problem (2.17) can be simplified as well to:

$$\min_{u \in \mathscr{V}_u^{\text{ad}}} J(u). \tag{2.19}$$

Following [Bar16], to solve this minimization problem, first-order necessary optimality conditions are derived formally by applying the Lagrange formalism, i.e. a Lagrange multiplier, a Lagrange functional, and an adjoint system are derived. For this, a Lagrange multiplier is defined as the tuple of adjoint states $\gamma = [\omega, \varphi, \omega_{\text{int}}, \psi_{\text{mesh}}, \psi_{\text{int}}, \psi, \psi_{\text{cool}}]$ and the Lagrange functional as

$$\begin{aligned}
\mathscr{L}(\zeta,u,\gamma) &:= J(\zeta,u) - f(\zeta,u)\cdot\gamma \\
&= \frac{\lambda_{\text{end}}}{2}\int\limits_{\Gamma_{\text{cool}}} (g(t_{\text{end}},x_1) - g_d(t_{\text{end}},x_1))^2 \,\mathrm{d}x_1 \\
&+ \frac{\lambda_{\text{all}}}{2}\int\limits_0^{t_{\text{end}}}\int\limits_{\Gamma_{\text{cool}}} (g(t,x_1)-g_d(t,x_1))^2 \,\mathrm{d}x_1\mathrm{d}t + \frac{\lambda_u}{2}\int\limits_0^{t_{\text{end}}}\int\limits_{\Gamma_u} (u(t))^2 \,\mathrm{d}x\mathrm{d}t \\
&- \int\limits_0^{t_{\text{end}}}\int\limits_{\Omega} (\dot{\Theta} - \Upsilon\cdot\nabla\Theta - \alpha\Delta\Theta)\cdot\omega \,\mathrm{d}x\mathrm{d}t \qquad (2.20)\\
&- \int\limits_0^{t_{\text{end}}}\int\limits_{\Gamma_u} (\partial_{\boldsymbol{n}}\Theta - u)\cdot\varphi \,\mathrm{d}x\mathrm{d}t - \int\limits_0^{t_{\text{end}}}\int\limits_{\Gamma_{\text{int}}} (\Theta - \Theta_{\text{melt}})\cdot\omega_{\text{int}} \,\mathrm{d}x\mathrm{d}t \\
&- \int\limits_0^{t_{\text{end}}}\int\limits_{\Omega} (\Delta\Upsilon)\cdot\psi_{\text{mesh}} \,\mathrm{d}x\mathrm{d}t - \int\limits_0^{t_{\text{end}}}\int\limits_{\Gamma_{\text{int}}} \left(\Upsilon - \left(\frac{1}{\ell}[k(\nabla\Theta)]_l^s\right)\cdot\boldsymbol{n}_{\text{int}}\right)\cdot\psi_{\text{int}} \,\mathrm{d}x\mathrm{d}t \\
&- \int\limits_0^{t_{\text{end}}}\int\limits_{\Gamma_{\text{cool}}} \left(\sqrt{1+g_{x_1}^2}\cdot[k(\nabla\Theta)]_l^s\circ\Phi - \ell\cdot g_t\right)\cdot\psi \,\mathrm{d}x\mathrm{d}t.
\end{aligned}$$

The Lagrange multiplier γ is also called adjoint state. As described formally in [Bar16], the derivatives of $\mathscr{L}(\zeta,u,\gamma)$ with respect to the states $\zeta = [\Theta,\Upsilon,g]$ can be used to derive the equations of the adjoint system.

The adjoint system is now formulated first as an abstract adjoint equation and then, as before, the specific equations for the Stefan problem are plugged into this framework. The adjoint equation

$$f^*(\zeta,u,\gamma) = 0 \qquad (2.21)$$

is defined through the requirement that the first variation of the Lagrange functional vanishes in all admissible directions $\delta\zeta \in \mathscr{V}_x$, i.e.

$$f^*(\zeta,u,\gamma) = 0 \qquad \Leftrightarrow \qquad \mathscr{L}_\zeta(\zeta,u,\gamma)\delta\zeta = 0.$$

All terms from Equations (2.1), (2.3) and (2.14), which do not appear in $f(\zeta,u)$ and thereby in the Lagrange functional (2.20), are treated explicitly as conditions to the directions of variation $\delta\zeta$ (see Equation (2.22)). For the Stefan problem, Equation (2.21) has the form

$$D_{[\Theta,\Upsilon,g]}\mathscr{L}[\delta\Theta,\delta\Upsilon,\delta g] = 0.$$

The above variation of the Lagrange functional is carried out for Θ, Υ, and g, resulting in adjoint equations for the adjoint states $\gamma = [\omega, \varphi, \omega_{\text{int}}, \psi_{\text{mesh}}, \psi_{\text{int}}, \psi, \psi_{\text{cool}}]$. Since this is already done in [Bar16], it is shown here only for the temperature Θ, because the jump across the interface has to be treated specially and additional jump terms arise in the respective adjoint equations.

In detail, the explicit conditions to the direction of variation $\delta\Theta$ are

$$\begin{aligned} \delta\Theta &= 0, && \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{cool}}, \\ \partial_{\boldsymbol{n}}\delta\Theta &= 0, && \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{N}}, \\ \delta\Theta(0) &= 0, && \text{on } \Omega, \end{aligned} \tag{2.22}$$

and the variation of the Lagrange functional with respect to Θ reads

$$\begin{aligned} 0 &= D_\Theta \mathscr{L} \delta\Theta \\ &= D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_\Omega (\dot{\Theta} - \Upsilon \cdot \nabla\Theta - \alpha\Delta\Theta) \cdot \omega \, \mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\ &+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_u} (\partial_{\boldsymbol{n}}\Theta - u) \cdot \varphi \, \mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\ &+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} (\Theta - \Theta_{\text{melt}}) \cdot \omega_{\text{int}} \, \mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\ &+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} \Big(\Upsilon - \Big(\frac{1}{\ell}[k(\nabla\Theta)]_l^s\Big) \cdot \boldsymbol{n}_{\text{int}}\Big) \cdot \psi_{\text{int}} \, \mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\ &+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{cool}}} \Big(\sqrt{1 + g_{x_1}^2} \cdot [k(\nabla\Theta)]_l^s \circ \Phi - \ell \cdot g_t\Big) \cdot \psi \, \mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta. \end{aligned}$$

Next, integration by parts as well as the conditions (2.22) are applied to the variation of the first integral in the equation above with respect to the temperature. This leads to

$$\begin{aligned} &D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_\Omega (\dot{\Theta} - \Upsilon \cdot \nabla\Theta - \alpha\Delta\Theta) \cdot \omega \, \mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\ &= - \int_0^{t_{\text{end}}} \int_\Omega \delta\dot{\Theta} \cdot \omega \, \mathrm{d}x\mathrm{d}t + \int_0^{t_{\text{end}}} \int_\Omega \Upsilon \cdot \nabla\delta\Theta \cdot \omega \, \mathrm{d}x\mathrm{d}t + \int_0^{t_{\text{end}}} \int_\Omega \alpha\Delta\delta\Theta \cdot \omega \, \mathrm{d}x\mathrm{d}t \end{aligned}$$

$$
\begin{aligned}
&= -\int_0^{t_\text{end}}\int_\Omega \delta\dot{\Theta}\cdot\omega \,\mathrm{d}x\mathrm{d}t + \int_0^{t_\text{end}}\int_{\Omega_s} \Upsilon\cdot\nabla\delta\Theta\cdot\omega\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\Omega_l} \Upsilon\cdot\nabla\delta\Theta\cdot\omega\,\mathrm{d}x\mathrm{d}t + \int_0^{t_\text{end}}\int_{\Omega_s} k_s\Delta\delta\Theta\cdot\omega\,\mathrm{d}x\mathrm{d}t + \int_0^{t_\text{end}}\int_{\Omega_l} k_l\Delta\delta\Theta\cdot\omega\,\mathrm{d}x\mathrm{d}t \\
&= -\int_\Omega \omega(t_\text{end})\delta\Theta(t_\text{end})\,\mathrm{d}x + \int_\Omega \omega(0)\underbrace{\delta\Theta(0)}_{\overset{(2.22)}{=}\ 0}\,\mathrm{d}x + \int_0^{t_\text{end}}\int_\Omega \dot{\omega}\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\partial\Omega_s} \Upsilon\cdot(\omega\cdot\boldsymbol{n})\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t - \int_0^{t_\text{end}}\int_{\Omega_s} \Upsilon\cdot\nabla\omega\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\partial\Omega_l} \Upsilon\cdot(\omega\cdot\boldsymbol{n})\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t - \int_0^{t_\text{end}}\int_{\Omega_l} \Upsilon\cdot\nabla\omega\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\partial\Omega_s} k_s\omega\partial_{\boldsymbol{n}}\delta\Theta\,\mathrm{d}x\mathrm{d}t - \int_0^{t_\text{end}}\int_{\Omega_s} k_s\nabla\omega\cdot\nabla\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\partial\Omega_l} k_l\omega\partial_{\boldsymbol{n}}\delta\Theta\,\mathrm{d}x\mathrm{d}t - \int_0^{t_\text{end}}\int_{\Omega_l} k_l\nabla\omega\cdot\nabla\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&= -\int_\Omega \omega(t_\text{end})\delta\Theta(t_\text{end})\,\mathrm{d}x + \int_0^{t_\text{end}}\int_\Omega \dot{\omega}\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\Gamma_\text{int}} \Upsilon\cdot(\omega\cdot\underbrace{\boldsymbol{n}}_{=\ \boldsymbol{n}_\text{int}})\cdot\delta\Theta\Big|_{\Omega_s}\,\mathrm{d}x\mathrm{d}t + \int_0^{t_\text{end}}\int_{\Gamma_\text{int}} \Upsilon\cdot(\omega\cdot\underbrace{\boldsymbol{n}}_{=\ -\boldsymbol{n}_\text{int}})\cdot\delta\Theta\Big|_{\Omega_l}\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\partial\Omega_s\setminus\Gamma_\text{int}} \omega\cdot(\underbrace{\Upsilon\cdot\boldsymbol{n}}_{=\ 0})\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t + \int_0^{t_\text{end}}\int_{\partial\Omega_l\setminus\Gamma_\text{int}} \omega\cdot(\underbrace{\Upsilon\cdot\boldsymbol{n}}_{=\ 0})\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&\quad - \int_0^{t_\text{end}}\int_\Omega \Upsilon\cdot\nabla\omega\cdot\delta\Theta\,\mathrm{d}x\mathrm{d}t \\
&\quad + \int_0^{t_\text{end}}\int_{\Gamma_\text{int}} \omega k_s(\partial_{\boldsymbol{n}_\text{int}}\delta\Theta)_s - \omega k_l(\partial_{\boldsymbol{n}_\text{int}}\delta\Theta)_l\,\mathrm{d}x\mathrm{d}t
\end{aligned}
$$

$$
\begin{aligned}
&+\int_0^{t_{\mathrm{end}}}\int_{\Gamma_u} k_s\omega\partial_{\boldsymbol{n}}\delta\Theta\ \mathrm{d}x\mathrm{d}t+\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{N}}\cup\Gamma_{\mathrm{cool}}} \alpha\omega\underbrace{\partial_{\boldsymbol{n}}\delta\Theta}_{\overset{(2.22)}{=}0}\ \mathrm{d}x\mathrm{d}t\\
&-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{int}}} k_s(\partial_{\boldsymbol{n}}\omega)_s\delta\Theta\ \mathrm{d}x\mathrm{d}t-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{int}}} k_l(\partial_{\boldsymbol{n}}\omega)_l\delta\Theta\ \mathrm{d}x\mathrm{d}t\\
&-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{N}}\cup\Gamma_u} \alpha\partial_{\boldsymbol{n}}\omega\delta\Theta\ \mathrm{d}x\mathrm{d}t-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{cool}}} k_s\partial_{\boldsymbol{n}}\omega\underbrace{\delta\Theta}_{\overset{(2.22)}{=}0}\ \mathrm{d}x\mathrm{d}t\\
&+\int_0^{t_{\mathrm{end}}}\int_{\Omega} \alpha\Delta\omega\cdot\delta\Theta\ \mathrm{d}x\mathrm{d}t\\
=&-\int_{\Omega}\omega(t_{\mathrm{end}})\delta\Theta(t_{\mathrm{end}})\ \mathrm{d}x+\int_0^{t_{\mathrm{end}}}\int_{\Omega}(\dot{\omega}-\Upsilon\cdot\nabla\omega+\alpha\Delta\omega)\cdot\delta\Theta\ \mathrm{d}x\mathrm{d}t\\
&+\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{int}}}\omega[k_s(\partial_{\boldsymbol{n}_{\mathrm{int}}}\delta\Theta)_s-k_l(\partial_{\boldsymbol{n}_{\mathrm{int}}}\delta\Theta)_l]\ \mathrm{d}x\mathrm{d}t\\
&-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{int}}}[k_s(\partial_{\boldsymbol{n}_{\mathrm{int}}}\omega)_s-k_l(\partial_{\boldsymbol{n}_{\mathrm{int}}}\omega)_l]\delta\Theta\ \mathrm{d}x\mathrm{d}t+\int_0^{t_{\mathrm{end}}}\int_{\Gamma_u}k_s\omega\partial_{\boldsymbol{n}}\delta\Theta\ \mathrm{d}x\mathrm{d}t\\
&-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{N}}\cup\Gamma_u}\alpha\partial_{\boldsymbol{n}}\omega\delta\Theta\ \mathrm{d}x\mathrm{d}t.
\end{aligned}
$$

Further, inserting this into the variation of the Lagrange functional with respect to Θ, gives

$$
\begin{aligned}
0&=D_{\Theta}\mathscr{L}\delta\Theta\\
&=-\int_{\Omega}\omega(t_{\mathrm{end}})\delta\Theta(t_{\mathrm{end}})\ \mathrm{d}x+\int_0^{t_{\mathrm{end}}}\int_{\Omega}(\dot{\omega}-\Upsilon\cdot\nabla\omega+\alpha\Delta\omega)\cdot\delta\Theta\ \mathrm{d}x\mathrm{d}t\\
&\quad+\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{int}}}\omega[k(\nabla\delta\Theta)]_l^s\ \mathrm{d}x\mathrm{d}t-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{int}}}[k(\nabla\omega)]_l^s\delta\Theta\ \mathrm{d}x\mathrm{d}t\\
&\quad+\int_0^{t_{\mathrm{end}}}\int_{\Gamma_u}k_s\omega\partial_{\boldsymbol{n}}\delta\Theta\ \mathrm{d}x\mathrm{d}t-\int_0^{t_{\mathrm{end}}}\int_{\Gamma_{\mathrm{N}}\cup\Gamma_u}\alpha\partial_{\boldsymbol{n}}\omega\delta\Theta\ \mathrm{d}x\mathrm{d}t
\end{aligned}
$$

$$
\begin{aligned}
&+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_u} (\partial_{\boldsymbol{n}}\Theta - u) \cdot \varphi \,\mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\
&+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} (\Theta - \Theta_{\text{melt}}) \cdot \omega_{\text{int}} \,\mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\
&+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} \Big(\Upsilon - \Big(\frac{1}{\ell}[k(\nabla\Theta(t))]_l^s\Big) \cdot \boldsymbol{n}_{\text{int}}\Big) \cdot \psi_{\text{int}} \,\mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\
&+ D_\Theta\Bigg(- \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{cool}}} \Big(\sqrt{1+g_{x_1}^2} \cdot [k(\nabla\Theta)]_l^s \circ \Phi - \ell \cdot g_t\Big) \cdot \psi \,\mathrm{d}x\mathrm{d}t \Bigg) \cdot \delta\Theta \\
= &- \int_\Omega \omega(t_{\text{end}})\delta\Theta(t_{\text{end}}) \,\mathrm{d}x + \int_0^{t_{\text{end}}} \int_\Omega (\dot{\omega} - \Upsilon \cdot \nabla\omega + \alpha\Delta\omega) \cdot \delta\Theta \,\mathrm{d}x\mathrm{d}t \\
&+ \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} \omega[k(\nabla\delta\Theta)]_l^s \,\mathrm{d}x\mathrm{d}t - \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} [k(\nabla\omega)]_l^s \delta\Theta \,\mathrm{d}x\mathrm{d}t \\
&+ \int_0^{t_{\text{end}}} \int_{\Gamma_u} k_s\omega\partial_{\boldsymbol{n}}\delta\Theta \,\mathrm{d}x\mathrm{d}t - \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{N}}\cup\Gamma_u} \alpha\partial_{\boldsymbol{n}}\omega\delta\Theta \,\mathrm{d}x\mathrm{d}t \\
&- \int_0^{t_{\text{end}}} \int_{\Gamma_u} \partial_{\boldsymbol{n}}\delta\Theta \cdot \varphi \,\mathrm{d}x\mathrm{d}t - \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} \delta\Theta \cdot \omega_{\text{int}} \,\mathrm{d}x\mathrm{d}t \\
&+ \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{int}}} \Big(\frac{1}{\ell}[k(\nabla\delta\Theta)]_l^s\Big) \cdot \boldsymbol{n}_{\text{int}} \cdot \psi_{\text{int}} \,\mathrm{d}x\mathrm{d}t - \int_0^{t_{\text{end}}} \int_{\Gamma_{\text{cool}}} \sqrt{1+g_{x_1}^2} \cdot [k(\nabla\delta\Theta)]_l^s \circ \Phi \cdot \psi \,\mathrm{d}x\mathrm{d}t.
\end{aligned}
$$

By using proper variation of $\delta\Theta$, certain terms can be eliminated from the above equation. Thereby, terms which are integrated over the same domain and have the same multiplier on the right can be consolidated into one equation so that the following adjoint equations arise

$$
\begin{aligned}
\dot{\omega} - \Upsilon \cdot \nabla \omega + \alpha \Delta \omega &= 0, && \text{on } [0, t_{\text{end}}) \times \Omega, && \text{(2.23a)}\\
\omega_{\text{int}} + [k(\nabla \omega)]_l^s &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{int}}, && \text{(2.23b)}\\
\omega &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{int}}, && \text{(2.23c)}\\
\partial_{\boldsymbol{n}} \omega &= 0, && \text{on } [0, t_{\text{end}}) \times (\Gamma_{\text{N}} \cup \Gamma_u), && \text{(2.23d)}\\
k_s \omega - \varphi &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_u, && \text{(2.23e)}\\
\psi &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{cool}}, && \text{(2.23f)}\\
\psi_{\text{int}} &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{int}}, && \text{(2.23g)}\\
\omega(t_{\text{end}}) &= 0, && \text{on } \Omega. && \text{(2.23h)}
\end{aligned}
$$

The latter equations form the adjoint system for the adjoint state ω, which can be interpreted as the adjoint temperature variable. The first equation (2.23a) is similar to the heat equation, while the second equation (2.23b) is analogue to the Stefan condition. The other equations can be understood as boundary conditions and the initial condition at time $t = t_{\text{end}}$.

The variation of the Lagrange functional with respect to Υ and g are performed analogously. A detailed derivation of those can be found in [Bar16].

Similar to the state equation in (2.17), the adjoint Equation (2.21) for the present optimal control problem is a PDE system, which is called the adjoint system. This formal approach leads to

$$
\begin{aligned}
\dot{\omega} - \Upsilon \cdot \nabla \omega + \alpha \Delta \omega &= 0, && \text{on } [0, t_{\text{end}}) \times \Omega,\\
\omega_{\text{int}} + [k(\nabla \omega)]_l^s &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{int}},\\
\omega &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{int}},\\
\partial_{\boldsymbol{n}} \omega &= 0, && \text{on } [0, t_{\text{end}}) \times (\Gamma_{\text{N}} \cup \Gamma_u),\\
k_s \omega &= \varphi, && \text{on } [0, t_{\text{end}}) \times \Gamma_u,\\
&\ell \cdot \dot{\psi} && && \text{(2.24)}\\
&+(1 + g_{x_1}^2) \cdot [k(\partial_{x_2}^2 \Theta)]_l^s \circ \Phi \cdot \psi \\
&-\partial_{x_1}(2 g_{x_1} \cdot [k(\partial_{x_2} \Theta)]_l^s \circ \Phi \cdot \psi) = \lambda_{\text{all}}(g - g_d), && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{cool}},\\
\Delta \psi &= \omega \nabla \Theta, && \text{on } [0, t_{\text{end}}) \times \Omega,\\
\psi &= 0, && \text{on } [0, t_{\text{end}}) \times \partial\Omega,\\
\partial_{\boldsymbol{n}} \psi &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{N}},\\
\psi_{\text{int}} &= 0, && \text{on } [0, t_{\text{end}}) \times \Gamma_{\text{int}},\\
\omega(t_{\text{end}}) &= 0, && \text{on } \Omega,\\
\psi(t_{\text{end}}) + \frac{\lambda_{\text{end}}}{\ell}(g(t_{\text{end}}) - g_d(t_{\text{end}})) &= 0, && \text{on } \Gamma_{\text{cool}}.
\end{aligned}
$$

In this PDE system, the states $\mathit{\Theta}$, Υ, and g are given and the functions ω, ψ and φ are unknowns. Initial values for the adjoint states ω and ψ are given at t_{end}. Thus, contrary to the Equations (2.1), (2.3) and (2.14) that govern the Stefan problem, the equations in (2.24) must be solved backward in time.

The optimal control problem can be solved with a gradient method [Trö10]. Therefore, in addition to evaluations of the Equations (2.1), (2.3), (2.14) and (2.24), also the gradient of the cost functional ∇J with respect to the control u is required. As shown in [Bar16], ∇J can be expressed in terms of the Lagrange functional (2.20):

$$J_u(u)\delta u = \mathscr{L}_u(\zeta, u, \gamma)\delta u = \int_0^{t_{\text{end}}} \int_{\Gamma_u} (\lambda_u u + \varphi))\delta u \;\mathrm{d}x\mathrm{d}t.$$

With this, the *gradient condition* can be formulated:

$$\langle \mathscr{L}_u(\zeta, u, \gamma), \tilde{u} - u\rangle = \int_0^{t_{\text{end}}} \int_{\Gamma_u} (\lambda_u u + \varphi))(\tilde{u} - u) \;\mathrm{d}x\mathrm{d}t \geq 0, \qquad \text{for all } \tilde{u} \in \mathscr{V}_u^{\text{ad}}. \tag{2.25}$$

To bound the control, box constraints are employed for the control $\mathscr{V}_u^{\text{ad}} = \{u \in \mathscr{V}_u : \underline{u} \leq u(t) \leq \overline{u}, t \in [0, t_{\text{end}}]\}$ with lower and upper bounds $\underline{u} < \overline{u}$. The unrestricted case $\mathscr{V}_u^{\text{ad}} = \mathscr{V}_u$ can be expressed with $\underline{u} = -\infty$, $\overline{u} = \infty$. In this case, (2.25) simplifies to the *gradient equation*

$$0 = \int_{\Gamma_u} \lambda_u u + \varphi \;\mathrm{d}x, \qquad t \in (0, t_{\text{end}}]. \tag{2.26}$$

As a consequence, the required gradient of the cost functional can be expressed as

$$\nabla J = \int_{\Gamma_u} \lambda_u u + \varphi \;\mathrm{d}x, \tag{2.27}$$

and can be used in a gradient method, which is described here for the optimal control problem

$$\min_{u \in \mathscr{V}_u^{\text{ad}}} J(u).$$

Given a control $u^{k-1} \in \mathscr{V}_u^{\text{ad}}$, the projected gradient method [Trö10], described in Algorithm 2.1, uses the negative gradient $-\nabla J(u^{k-1})$ as the descent direction (Line 6).

To proceed, a step size s^k is computed in Line 5 with Algorithm 2.2. To ensure that the computed control is admissible, the projection $\mathbf{P}_{[\underline{u},\,\overline{u}]} \colon \mathscr{V}_u \to \mathscr{V}_u^{\text{ad}}$,

$$\mathbf{P}_{[\underline{u},\,\overline{u}]}(u) := \max\{\underline{u}, \min\{u, \overline{u}\}\},$$

is applied pointwise in time (Line 7).

Algorithm 2.1: Projected gradient method

```
    Input: initial control $u^0$
    Output: control $u^{k_{end}}$
1 $k = 1$
2 while not converged do
3     Solve Stefan problem (Equations (2.1), (2.3) and (2.14))
4     Solve adjoint system (2.24)
5     Compute step size $s^k$
6     $d^k = \int_{\Gamma_u} \lambda_u u + \varphi \, \mathrm{d}x$
7     $u^k = \mathbf{P}_{[\underline{u},\, \overline{u}]}(u^{k-1} - s^k \cdot d^k)$
8     $k = k + 1$
9 end
```

The possible stopping criteria[2], which are evaluated in Line 2 of Algorithm 2.1, depend on the norm of the step $\left\|s^k \cdot d^k\right\|_2 < \delta_{\text{abs}}$ and the relative change of the cost functional

$$\frac{|J(u^{k-1}) - J(u^k)|}{|J(u^{k-1})|} < \delta_{\text{rel}}, \tag{2.28}$$

with certain tolerances $\delta_{\text{abs}}, \delta_{\text{rel}} > 0$. Besides that, a maximum iteration number $k_{\max}$ is used.

It is worth noting that the choice of the step size s^k is of great importance for the performance of the projected gradient method. The algorithm to compute the step size is a modification of the method used in [Zie08]. Three sampling points are evaluated to approximate $q(s) \approx J(\mathbf{P}_{[\underline{u},\, \overline{u}]}(u^{k-1} - s \cdot d^k))$ with a quadratic polynomial $q \in \mathbb{P}_2$. The local minimum of q is used as the next sampling point to refine the approximation.

In every iteration of Algorithm 2.2, the cost functional $J(\mathbf{P}_{[\underline{u},\, \overline{u}]}(u^{k-1} - s_j \cdot d^k))$ must be evaluated at least once. These evaluations require the solution of the Stefan problem, i.e. Equations (2.1), (2.3) and (2.14), and are computationally expensive. To avoid excessive computational costs in Algorithm 2.2, a maximum iteration number $i_{\max}$ is added in Line 4. If $i > i_{\max}$, the sampling point s_j with the smallest cost value $k_j, j = 0, 1, 2$ is returned to Algorithm 2.1. Otherwise, with tolerances $\epsilon_{\text{abs}}, \epsilon_{\text{rel}} > 0$, the algorithm stops if the newly computed minimum s of the polynomial q is close to an already existing sampling point

$$|s - s_j| < \epsilon_{\text{abs}}, \qquad \text{for any } j = 0, 1, 2,$$

or if the relative change of the value of J at the new sampling point s is small

$$\frac{|J(\mathbf{P}_{[\underline{u},\, \overline{u}]}(u^{k-1} - s \cdot d^k)) - k_j|}{|k_1|} < \epsilon_{\text{rel}}.$$

[2]Details can be found in the source code ([Bar24, gradient_method.py: line 100 – 114)]

Algorithm 2.2: Quadratic line minimization

```
Input: The step direction $d^k$
Output: step size $s$
1  $i = 1$
2  Choose $s_0 = 0 < s_1 < s_2,\ \epsilon_{\text{grow}}$
3  $k_j = J(\mathbf{P}_{[\underline{u},\ \overline{u}]}(u^{k-1} - s_j \cdot d^k)), \quad j = 0, 1, 2$      // Needs 2 evaluations of
       Equations (2.1), (2.3) and (2.14) on $(0, t_{\text{end}}]$
4  while not converged do
5      $q \in \mathbb{P}_2 : q(s_j) = k_j, \quad j = 0, 1, 2$
6      $s = \operatorname*{argmin}_{\tilde{s} \in [s_0, s_2]} q(\tilde{s})$
7      if $|s - s_2| < \epsilon_{grow}$ then
8          $s_0 = s_1,\ k_0 = k_1,\ s_1 = s_2,\ k_1 = k_2$
9          $s_2 = 2 \cdot s_2$                    // Alternative $s_2 = s_2 + s_1 - s_0$
10         $k_2 = J(\mathbf{P}_{[\underline{u},\ \overline{u}]}(u^{k-1} - s_2 \cdot d^k))$          // Needs 1 evaluation of
               Equations (2.1), (2.3) and (2.14) on $(0, t_{\text{end}}]$
11     else if $s > s_1$ then
12         $s_0 = s_1,\ k_0 = k_1,\ s_1 = s$
13         $k_1 = J(\mathbf{P}_{[\underline{u},\ \overline{u}]}(u^{k-1} - s_1 \cdot d^k))$          // Needs 1 evaluation of
               Equations (2.1), (2.3) and (2.14) on $(0, t_{\text{end}}]$
14         $i = i + 1$
15     else
16         $s_2 = s_1,\ k_2 = k_1,\ s_1 = s$
17         $k_1 = J(\mathbf{P}_{[\underline{u},\ \overline{u}]}(u^{k-1} - s_1 \cdot d^k))$          // Needs 1 evaluation of
               Equations (2.1), (2.3) and (2.14) on $(0, t_{\text{end}}]$
18         $i = i + 1$
19     end
20 end
```

The main computational cost for solving the open-loop control problem with these algorithms lies in the evaluation of Equations (2.1), (2.3) and (2.14), which occurs several times, especially in the iteration steps of Algorithm 2.2. The convergence of Algorithm 2.1 strongly depends on the problem settings and the initial guess for the control. Also, different choices of the weights in the cost functional influence the convergence behavior as well as the choice of the desired interface position. A more detailed discussion of Algorithms 2.1 and 2.2 can be found in [Bar16, BBHS18].

In the case of a real-world application scenario, the control is computed in advance in an offline phase, and then the control is applied to the system. In case of any disturbances, perturbations, approximation errors, or model inaccuracies the system can still deviate from the desired trajectory and the open-loop control cannot react to this deviation. However, it can be used as a reference solution for a closed-loop stabilization problem, which can overcome the just described shortcomings of the open-loop control.

2.3. LQR for Non-linear Problems

In this section, the Riccati-feedback approach with LQR for general non-linear problems is rendered. The goal of this approach is to stabilize the pair of reference state and input $(\zeta^h_{\text{ref}}, u^h_{\text{ref}})$. This approach is then further specified for the Stefan problem in Section 3.1. The main distinction for the Stefan problem is its non-autonomous character. In this section, the general concept for the autonomous case is introduced, before the details related to the Stefan problem are presented in Section 3.1.

Similar to Section 2.1.4, an abstract, stabilized, semi-discrete, dynamical system is considered, which reads

$$\dot{\zeta}^h_{\Delta}(t) = f^h(t; \zeta^h_{\Delta}(t), u^h_{\mathscr{K}}(t)) \tag{2.29}$$

on a finite time horizon $t \in \mathscr{I} = [t_0, t_{\text{end}}]$ and a domain $\Omega \subset \mathbb{R}^d$ of dimension $d > 0$. The index $(\cdot)^h$ indicates that the system (2.15), the state ${\zeta_{\Delta}}^h(t)$, and the stabilization $u^h_{\mathscr{K}}(t)$ are discretized in space (see Section 2.1.2). Additionally, the index $(\cdot)_{\Delta}$ indicates that the difference state and input $(\zeta^h_{\Delta}, u^h_{\mathscr{K}})$ are considered

$$\begin{aligned} \zeta^h_{\Delta}(t) &= \zeta^h(t) - \zeta^h_{\text{ref}}(t), \\ u^h_{\mathscr{K}}(t) &= u^h(t) - u^h_{\text{ref}}(t), \end{aligned} \tag{2.30}$$

where $(\zeta^h_{\text{ref}}, u^h_{\text{ref}})$ are the reference state and input that are to be stabilized. The stabilization $u^h_{\mathscr{K}}(t)$ has the index $(\cdot)_{\mathscr{K}}$ to emphasize that it is derived with the feedback gain matrix $\mathscr{K}$.

For this, the LQR approach is used (e.g. [Son98]) to obtain a feedback-based stabilization of Equation (2.29). This approach is used because it is well studied for types of problems that are related to the Stefan problem, e.g. convection-diffusion equations [Wei16], and it demonstrates promising performance for these.

LQR is a method to numerically compute an optimal feedback stabilization for a linear system in state-space formulation, see, e.g. [Loc01]. Moreover, non-linear systems can be stabilized with this method if the deviation from the reference trajectory is sufficiently small [Son98, Section 8.5]. Therefore, the abstract non-linear Equation (2.29) needs to be linearized and formulated in terms of Equation (2.5).

To formulate the abstract stabilization problem, a quadratic cost functional is defined. It tracks the deviation of the system from the desired trajectory, which is indicated by the output y^h, and penalizes the input cost with a weight factor $0 < \lambda \in \mathbb{R}$. Additionally, the difference state at t_{end} is measured with S positive semidefinite. The cost functional is thus defined as

$$J(\zeta_\Delta^h, y^h, u_{\mathscr{K}}^h) = \frac{1}{2}\int_0^{t_{\text{end}}} \left\|y^h\right\|^2 + \lambda \left\|u_{\mathscr{K}}^h\right\|^2 \, \mathrm{d}t + \zeta_\Delta^h(t_{\text{end}})^\mathsf{T} S \zeta_\Delta^h(t_{\text{end}}). \tag{2.31}$$

This cost functional is minimized subject to the linear time-varying system. With this, the LQR problem reads

$$\begin{aligned} &\min_{u_{\mathscr{K}}^h} J(\zeta_\Delta^h, y^h, u_{\mathscr{K}}^h) \\ \text{subject to} \\ &\mathscr{M}\dot{\zeta}_\Delta^h = \mathscr{A}\zeta_\Delta^h + \hat{\mathscr{B}} u_{\mathscr{K}}^h, \\ &y^h = \mathscr{C}\zeta_\Delta^h. \end{aligned} \tag{2.32}$$

The unique solution to the LQR problem (2.32) (see, e.g. [Meh91, Loc01, Son98, BLP08]) is

$$u_{\mathscr{K}}^h = -\mathscr{K}\zeta_\Delta^h, \tag{2.33}$$

where the feedback gain matrix

$$\mathscr{K} = \frac{1}{\lambda}\hat{\mathscr{B}}^\mathsf{T}\mathbf{X}\mathscr{M} = \frac{1}{\sqrt{\lambda}}\mathscr{B}^\mathsf{T}\mathbf{X}\mathscr{M} \tag{2.34}$$

requires the solution $\mathbf{X}(t) \in \mathbb{R}^{n\times n}$ of a DRE. For the abstract autonomous problem, this is the large-scale matrix-valued autonomous generalized DRE

$$\begin{aligned} -\mathscr{M}^\mathsf{T}\dot{\mathbf{X}}\mathscr{M} &= \mathscr{C}^\mathsf{T}\mathscr{C} + \mathscr{A}^\mathsf{T}\mathbf{X}\mathscr{M} + \mathscr{M}^\mathsf{T}\mathbf{X}\mathscr{A} - \mathscr{M}^\mathsf{T}\mathbf{X}\mathscr{B}\mathscr{B}^\mathsf{T}\mathbf{X}\mathscr{M}, \\ \mathscr{M}^\mathsf{T}\mathbf{X}(t_{\text{end}})\mathscr{M} &= S. \end{aligned} \tag{2.35}$$

The coefficients of the DRE are the matrices $\mathscr{A}, \mathscr{M} \in \mathbb{R}^{n\times n}$, $\mathscr{B} \in \mathbb{R}^{n\times m}$, $\mathscr{C} \in \mathbb{R}^{p\times n}$, where the input matrix $\mathscr{B} = \frac{1}{\sqrt{\lambda}}\hat{\mathscr{B}}$ is scaled with the weight factor from the cost functional (2.31).

To numerically compute the feedback gain matrix $\mathscr{K}_k = \mathscr{K}(t_k)$ and the feedback stabilization $u_k = u_{\mathscr{K}}^h(t_k)$ for $t_k \in \mathscr{T}_{\text{bwd}}$, the solution of the DRE (2.35) is required. Efficient low-rank methods can be used for the computation of the numerical solution of the DRE, $\mathbf{X}_k = \mathbf{X}(t_k)$, since it is assumed that $\mathbf{X}_k$ has a low (numerical) rank as shown by [Sti18b].

2.4. Differential Riccati Equation Solver

In this section, methods to numerically compute the solution $\mathbf{X}(t) \in \mathbb{R}^{n\times n}$ of the DRE (2.35) are presented. For this autonomous DRE, i.e. when all matrix coefficients are constant in time, many efficient low-rank methods exist, as outlined in Section 1.2. These methods are mostly tailored for the autonomous DRE and not yet suited to handle time-dependent coefficients. In this section, the existing methods are described and then adapted to the non-autonomous DRE in Section 3.3.

Analogously to the adjoint system for the open-loop control problem, the DRE is numerically solved backwards in time, starting at t_{end}, with the equidistant time discretization $\mathscr{T}_{\text{bwd}}$ from Section 2.1.2. For reasons of better readability and for consistency with previously published material, the methods which are presented here, are formulated forward in time and a change of variables $t \to t_{\text{end}} + t_0 - t$ is performed. The DRE to be solved is

$$\begin{aligned} \mathscr{M}^{\mathsf{T}}\dot{\mathbf{X}}\mathscr{M} &= \mathscr{C}^{\mathsf{T}}\mathscr{C} + \mathscr{A}^{\mathsf{T}}\mathbf{X}\mathscr{M} + \mathscr{M}^{\mathsf{T}}\mathbf{X}\mathscr{A} - \mathscr{M}^{\mathsf{T}}\mathbf{X}\mathscr{B}\mathscr{B}^{\mathsf{T}}\mathbf{X}\mathscr{M}, \\ \mathscr{M}^{\mathsf{T}}\mathbf{X}(t_0)\mathscr{M} &= \mathbf{X}_0 = S. \end{aligned} \tag{2.36}$$

The DRE solvers in this section then approximate $\mathbf{X}_k \approx \mathbf{X}(t_k)$, which can be computed efficiently with a low-rank factorization (see Section 2.4.2), otherwise the full solutions can be computed with an ODE approach.

2.4.1. Vectorization to an ODE

A straight forward method to solve DREs is to vectorize them with

$$\operatorname{vec}(A) = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix},$$

where the matrix $A = [a_1, \dots, a_n] \in \mathbb{R}^{n\times n}$ has the columns $a_1, \dots, a_n \in \mathbb{R}^n$. Consequently, $\operatorname{vec}(A)$ is a vector of size n^2. This leads to a transformation of the DRE into an equivalent vector-valued non-linear ODE:

$$\begin{aligned} \operatorname{vec}(\dot{\mathbf{X}}) &= \operatorname{vec}(\mathscr{C}^{\mathsf{T}}\mathscr{C}\mathscr{M}^{-1} + \mathscr{M}^{-\mathsf{T}}\mathscr{A}^{\mathsf{T}}\mathbf{X} + \mathscr{M}^{\mathsf{T}}\mathbf{X}\mathscr{A}\mathscr{M}^{-1} - \mathbf{X}\mathscr{B}\mathscr{B}^{\mathsf{T}}\mathbf{X}), \\ \operatorname{vec}(\mathbf{X}(t_0)) &= \operatorname{vec}(\mathscr{M}^{-\mathsf{T}}\mathbf{X}_0\mathscr{M}^{-1}). \end{aligned} \tag{2.37}$$

With the above formulation, existing numerical solvers for ODEs can then be applied. However, this approach does not exploit the low-rank structure of the solution and is unfeasible for large-scale problems. Still, for sufficiently small examples, the vectorization method can be used and, e.g. the MATLAB® `ode*`-functions can be applied to generate a very accurate reference DRE solution for error comparisons.

2.4.2. Low-rank Methods

As indicated in [Sti18b], with $m, p \ll n$, the solution of the DRE has a numerical rank that is small. Even if the matrices $\mathscr{A}, \mathscr{M} \in \mathbb{R}^{n\times n}$ are sparse, at each time t, the solution $\mathbf{X}(t) \in \mathbb{R}^{n\times n}$ is a dense matrix. However, it is symmetric, often has a low numerical rank, and can therefore be approximated by

$$\mathbf{X}(t) \approx L(t)D(t)L(t)^{\mathsf{T}}, \tag{2.38}$$

where the low-rank factors $L(t) \in \mathbb{R}^{n\times s(t)}$ and $D(t) \in \mathbb{R}^{s(t)\times s(t)}$ have the rank $s(t) \ll n$ [LMS15]. With this low-rank factorization, just the low-rank factors $L(t)$ and $D(t)$ need to be stored. Consequently, the memory requirement for the storage of the solution reduces to $\mathcal{O}(s(t)n + s(t)^2)$ instead of $\mathcal{O}(n^2)$, per time step. This factorization technique is used by the methods introduced in this section for reasons of runtime and memory efficiency.

2.4.2.1. Backward Differentiation Formulas

The BDF methods [CH52] are chosen here because existing implementations can be adapted to the non-autonomous case relatively straight forward since the original method is designed to handle non-autonomous data. However, the existing BDF methods for matrix-valued DREs like Algorithm 3.2 in [LMS15] are tuned for constant data.

For solving stiff problems, the BDF methods are a very popular class of linear multistep methods, which use approximations of previous time steps. On the one hand, higher orders of convergence can be obtained. On the other hand, additional initial values are required in order to start the methods of order $\wp \geq 2$. These additional initial approximations can be computed with any different one-step method of sufficient order of convergence or with the specific start-up algorithm introduced in Section 3.3.

Following [Die92, BM04, Men12, Lan17], for a general matrix equation

$$\dot{\mathbf{X}}(t) = f(t, \mathbf{X}), \qquad \mathbf{X}(t_0) = \mathbf{X}_0, \tag{2.39}$$

the matrix-valued BDF method of order $\wp$ can be expressed as

$$\mathbf{X}_k = -\sum_{j=1}^{\wp} \alpha_j \mathbf{X}_{k-j} + \tau_k \beta f(t_k, \mathbf{X}_k) \tag{2.40}$$

with the coefficients β and α_j from Table 2.1 and the time step size τ_k.

To apply this general BDF scheme to the DRE (2.36), the DRE can be reformulated to

$$\begin{aligned} \dot{\mathbf{X}} &= \mathscr{M}^{-\mathsf{T}}\mathscr{C}^{\mathsf{T}}\mathscr{C}\mathscr{M}^{-1} + \mathscr{M}^{-\mathsf{T}}\mathscr{A}^{\mathsf{T}}\mathbf{X} + \mathbf{X}\mathscr{A}\mathscr{M}^{-1} - \mathbf{X}\mathscr{B}\mathscr{B}^{\mathsf{T}}\mathbf{X} =: f(t, \mathbf{X}), \\ \mathbf{X}(t_0) &= \mathscr{M}^{-\mathsf{T}}\mathbf{X}_0\mathscr{M}^{-1}, \end{aligned} \tag{2.41}$$

$\wp$	β	α_1	α_2	α_3	α_4
1	1	-1			
2	$\frac{2}{3}$	$-\frac{4}{3}$	$\frac{1}{3}$		
3	$\frac{6}{11}$	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$	
4	$\frac{12}{25}$	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$

Table 2.1.: Coefficients of the BDF method of order $\wp = 1 \dots 4$ [AP98, Table 5.2].

with the assumption that $\mathscr{M}^{-1}$ exists. Now, the above equation is in the format of Equation (2.39) and can be inserted into Equation (2.40). This yields the BDF scheme for a DRE:

$$\begin{aligned}\mathscr{M}^{\mathsf{T}}\mathbf{X}_k\mathscr{M} = &-\sum_{j=1}^{\wp}\alpha_j\mathscr{M}^{\mathsf{T}}\mathbf{X}_{k-j}\mathscr{M} + \tau_k\beta\mathscr{C}^{\mathsf{T}}\mathscr{C} \\ &+ \tau_k\beta\mathscr{A}^{\mathsf{T}}\mathbf{X}_k\mathscr{M} + \tau_k\beta\mathscr{M}^{\mathsf{T}}\mathbf{X}_k\mathscr{A} - \tau_k\beta\mathscr{M}^{\mathsf{T}}\mathbf{X}_k\mathscr{B}\mathscr{B}^{\mathsf{T}}\mathbf{X}_k\mathscr{M}.\end{aligned} \tag{2.42}$$

Alternatively, if $\mathscr{M}^{-1}$ does not exist, Equations (2.39) and (2.40) can be multiplied with $\mathscr{M}^{\mathsf{T}}$ from the left and $\mathscr{M}$ from the right to get Equation (2.42). With the low-rank approximation from Equation (2.38) and the definition of the matrices

$$\begin{aligned}\mathscr{A}_k &= \tau\beta\mathscr{A} - \frac{1}{2}\mathscr{M}, \\ \mathscr{B}_k &= \sqrt{\tau\beta}\mathscr{B}, \\ \mathscr{C}_k^{\mathsf{T}} &= \left[\mathscr{C}^{\mathsf{T}}, \mathscr{M}^{\mathsf{T}}L_{k-1}, \dots, \mathscr{M}^{\mathsf{T}}L_{k-\wp}\right], \\ \mathscr{S}_k &= \begin{bmatrix}\tau\beta I_p & & & \\ & -\alpha_1 D_{k-1} & & \\ & & \ddots & \\ & & & -\alpha_\wp D_{k-\wp}\end{bmatrix},\end{aligned} \tag{2.43}$$

Equation (2.42) can be reformulated as

$$0 = \mathscr{A}_k^{\mathsf{T}}\mathbf{X}_k\mathscr{M} + \mathscr{M}^{\mathsf{T}}\mathbf{X}_k\mathscr{A}_k + \mathscr{C}_k^{\mathsf{T}}\mathscr{S}_k\mathscr{C}_k - \mathscr{M}^{\mathsf{T}}\mathbf{X}_k\mathscr{A}_k\mathscr{B}_k\mathscr{B}_k^{\mathsf{T}}\mathbf{X}_k\mathscr{M}, \tag{2.44}$$

which is an algebraic Riccati equation (ARE) with $\mathbf{X}_k \approx L_kD_kL_k^{\mathsf{T}}$. Consequently, an ARE is solved in each step of the BDF method. The whole procedure is summarized in Algorithm 2.3, see also [LMS15, Algorithm 3.2].

The inputs of Algorithm 2.3 are the matrices $\mathscr{A}$, $\mathscr{M}$, $\mathscr{B}$, $\mathscr{C}$, the time grid $\mathscr{T}$, and the order $\wp$ of the BDF method. Further, the low-rank factors $L_0, \dots, L_{\wp-1}, D_0, \dots, D_{\wp-1}$ of

Algorithm 2.3: Autonomous low-rank factor BDF method of order $\wp$

```
    Input: $\mathscr{A}, \mathscr{M}, \mathscr{B}, \mathscr{C}, \lambda, \mathscr{T}, \wp, L_0, \dots, L_{\wp-1}, D_0, \dots, D_{\wp-1}$
    Output: $\mathscr{K}_k, k = 0, \dots, n_t - 1$
1 $\mathscr{K}_{n_t-k} = \frac{1}{\sqrt{\lambda}}\mathscr{B}^\mathsf{T} L_k D_k L_k^\mathsf{T} \mathscr{M}, \quad k = 1, \dots, \wp - 1$
2 for $k = \wp, \dots, n_t$ do
3     $\mathscr{A}_k = \tau\beta\mathscr{A} - \frac{1}{2}\mathscr{M}$
4     $\mathscr{B}_k = \sqrt{\tau\beta}\mathscr{B}$
5     $\mathscr{C}_k^\mathsf{T} = \left[\mathscr{C}^\mathsf{T}, \mathscr{M}^\mathsf{T} L_{k-1}, \dots, \mathscr{M}^\mathsf{T} L_{k-\wp}\right]$
6     $\mathscr{S}_k = \begin{bmatrix} \tau\beta I_p & & & \\ & -\alpha_1 D_{k-1} & & \\ & & \ddots & \\ & & & -\alpha_\wp D_{k-\wp} \end{bmatrix}$
7     Solve ARE (2.44) for $L_k$ and $D_k$
8     $\mathscr{K}_{n_t-k} = \frac{1}{\sqrt{\lambda}}\mathscr{B}^\mathsf{T} L_k D_k L_k^\mathsf{T} \mathscr{M}$
9 end
```

the initial values $\mathbf{X}_0, \dots, \mathbf{X}_{\wp-1}$ are required as inputs with sufficient accuracy to obtain the desired order of convergence $\wp$. Instead of storing the approximations for each time step, Algorithm 2.3 can compute the feedback gain matrices $\mathscr{K}_k \approx \mathscr{K}(t_k), k = 1, \dots, n_t$, directly, which further decreases the storage requirements to $\mathscr{O}(mn)$ per time step. Note that the amount of memory on disk after the computation is finished is monitored here, but more memory may be needed during the computation. However, the Newton alternating-direction implicit (Newton-ADI) method is used to solve the ARE (2.44), which can accumulate the feedback gain matrices directly, avoiding the need to assemble the low-rank solution factors. This is a significant performance advantage in terms of the memory requirements during the computations, especially for large-scale DREs. Other low-rank solvers for AREs are not able to accumulate the feedback gain matrices directly, such as projection-based methods like EKSM and RKSM. Another low-rank ARE solver that can directly accumulate the feedback gain matrices is RADI. However, this method is not yet implemented in combination with the BDF method and is therefore not used for this thesis.

The BDF method for autonomous DREs (Algorithm 2.3) can be extended to the non-autonomous case, which is required for the non-autonomous DRE that results from the LQR problem related to the Stefan problem, see Section 3.3.

2.4.2.2. Autonomous Splitting Schemes

Following [Sti15b], for the general idea of splitting schemes, an abstract equation

$$\dot{\mathbf{X}} = (F+G)\mathbf{X}, \qquad \mathbf{X}(t_0) = \mathbf{X}_0, \tag{2.45}$$

is considered with the abstract operators F and G. The exact analytic solution to the abstract Equation (2.45) is denoted as $\mathrm{e}^{t(F+G)}\mathbf{X}_0$. For the splitting schemes, Equation (2.45) is split into the two subproblems

$$\dot{\mathbf{X}} = F\mathbf{X} \qquad \text{and} \qquad \dot{\mathbf{X}} = G\mathbf{X}, \tag{2.46}$$

which are approximated numerically instead of the full problem. In most cases, F and G can be chosen so that the subproblems are cheaper or easier to solve numerically than Equation (2.45). The solutions of such subproblems can then be combined in different ways to approximations of different orders of convergence for the full problem.

One example of this approach is the exponential Lie splitting method. For a given solution $\mathbf{X}_k = \mathbf{X}(t_k)$ at time t_k and a step size τ_k this method can be expressed as

$$\begin{aligned} \dot{\mathbf{X}}_F &= F\mathbf{X}_F, \qquad & \mathbf{X}_F(0) &= \mathbf{X}_k, \\ \dot{\mathbf{X}}_G &= G\mathbf{X}_G, \qquad & \mathbf{X}_G(0) &= \mathbf{X}_F(\tau_k). \end{aligned} \tag{2.47}$$

Then, the approximate solution to the full problem is $\mathbf{X}_G(\tau_k) \approx \mathbf{X}(t_{k+1})$. Alternatively, the method described by Equation (2.47) can be denoted with the time-stepping operator

$$S_{\tau_k} = \mathrm{e}^{\tau_k F}\mathrm{e}^{\tau_k G}.$$

The Lie splitting is a first order method. Higher order methods can be constructed by combining the solutions of the subproblems from Equation (2.46) in different ways. For instance, the second-order Strang splitting can be expressed as

$$S_{\tau_k} = \mathrm{e}^{\frac{\tau_k}{2}F}\mathrm{e}^{\tau_k G}\mathrm{e}^{\frac{\tau_k}{2}F}.$$

This can be generalized to exponential splitting schemes of higher order $\wp$ by

$$S_{\tau_k} = \prod_{j=1}^{\wp} \mathrm{e}^{\alpha_j \tau_k F}\mathrm{e}^{\beta_j \tau_k G}$$

with appropriate coefficients α_j and β_j [MSS99, MQ02, HV03, HWL06].

The splitting schemes just described are particularly interesting for DREs because they can be split into the affine and quadratic parts of the equation. The two arising subproblems for a DRE in the format of Equation (2.46) are

$$\mathcal{M}^{\mathsf{T}}\dot{\mathbf{X}}_F\mathcal{M} = \mathcal{C}^{\mathsf{T}}\mathcal{C} + \mathcal{A}^{\mathsf{T}}\mathbf{X}_F\mathcal{M} + \mathcal{M}^{\mathsf{T}}\mathbf{X}_F\mathcal{A}, \tag{2.48}$$

$$\mathcal{M}^{\mathsf{T}}\dot{\mathbf{X}}_G\mathcal{M} = -\mathcal{M}^{\mathsf{T}}\mathbf{X}_G\mathcal{B}\mathcal{B}^{\mathsf{T}}\mathbf{X}_G\mathcal{M}, \tag{2.49}$$

which have closed-form solutions under the assumption that $\mathscr{M}^{-1}$ exists. Note that the subproblem (2.49) can be simplified to

$$\dot{\mathbf{X}}_G = -\mathbf{X}_G \mathscr{B}\mathscr{B}^\mathsf{T} \mathbf{X}_G. \tag{2.50}$$

The solution to the two subproblems are given by [Sti15a]

$$\mathbf{X}_F(t) = \mathrm{e}^{t\mathscr{M}^{-\mathsf{T}}\mathscr{A}^\mathsf{T}} \mathbf{X}_0 \mathrm{e}^{t\mathscr{A}\mathscr{M}^{-1}} + \int_0^t \mathrm{e}^{s\mathscr{M}^{-\mathsf{T}}\mathscr{A}^\mathsf{T}} \mathscr{M}^{-\mathsf{T}} \mathscr{C}^\mathsf{T} \mathscr{C} \mathscr{M}^{-1} \mathrm{e}^{s\mathscr{A}\mathscr{M}^{-1}} \mathrm{d}s, \tag{2.51}$$

$$\mathbf{X}_G(t) = (I + t\mathbf{X}_0 \mathscr{B}\mathscr{B}^\mathsf{T})^{-1} \mathbf{X}_0. \tag{2.52}$$

In addition, these two solutions can be expressed in a computationally more efficient low-rank representation. To do this, a time step size τ_k and the approximation at time t_k, represented as $\mathbf{X}_k = L_k D_k L_k^\mathsf{T}$, are given together with $\mathscr{C}^\mathsf{T}\mathscr{C} = \tilde{\mathscr{C}}^\mathsf{T} \mathcal{S} \tilde{\mathscr{C}}$. Using a simplified version of the Woodbury matrix inversion formula as in [Sti18a], the solution to Equation (2.52) can be expressed as

$$\begin{aligned}\mathbf{X}_G(t_{k+1}) = (I + \tau_k L_k D_k L_k^\mathsf{T} \mathscr{B}\mathscr{B}^\mathsf{T})^{-1} L_k D_k L_k^\mathsf{T} &= L_k (I + \tau_k D_k L_k^\mathsf{T} \mathscr{B}\mathscr{B}^\mathsf{T} L_k)^{-1} D_k L_k^\mathsf{T} \\ &= L_k D_G L_k^\mathsf{T},\end{aligned} \tag{2.53}$$

with $D_G = (I + \tau_k D_k L_k^\mathsf{T} \mathscr{B}\mathscr{B}^\mathsf{T} L_k)^{-1} D_k$. Similarly, the solution to the affine subproblem (2.51) can be expressed in the same low-rank format by

$$\begin{aligned}\mathbf{X}_F(t_{k+1}) &= \mathrm{e}^{\tau_k\mathscr{M}^{-\mathsf{T}}\mathscr{A}^\mathsf{T}} L_k D_k L_k^\mathsf{T} \mathrm{e}^{\tau_k\mathscr{A}\mathscr{M}^{-1}} + \int_0^{\tau_k} \mathrm{e}^{s\mathscr{M}^{-\mathsf{T}}\mathscr{A}^\mathsf{T}} \mathscr{M}^{-\mathsf{T}} \tilde{\mathscr{C}}^\mathsf{T} \mathcal{S} \tilde{\mathscr{C}} \mathscr{M}^{-1} \mathrm{e}^{s\mathscr{A}\mathscr{M}^{-1}} \mathrm{d}s, \\ &= \tilde{L}_F(t_{k+1}) D_k \tilde{L}_F^\mathsf{T}(t_{k+1}) + \int_0^{\tau_k} L_{\mathscr{C}}^\mathsf{T}(s) \mathcal{S} L_{\mathscr{C}}(s) \mathrm{d}s,\end{aligned} \tag{2.54}$$

with $\tilde{L}_F(t_{k+1}) = \mathrm{e}^{\tau_k\mathscr{M}^{-\mathsf{T}}\mathscr{A}^\mathsf{T}} L_k$ and $L_{\mathscr{C}}(s) = \tilde{\mathscr{C}} \mathscr{M}^{-1} \mathrm{e}^{s\mathscr{A}\mathscr{M}^{-1}}$. Both terms in Equation (2.54) are of low numerical rank, so their sum has a low rank. The integral in Equation (2.54) can be efficiently approximated with quadrature formulas of sufficient accuracy, e.g. of order $\mathscr{q} = \wp + 1$, such that the quadrature error does not impair the order of convergence $\wp$ of the splitting scheme:

$$\begin{aligned}\mathbf{X}_F(t_{k+1}) &= \tilde{L}_F(t_{k+1}) D_k \tilde{L}_F^\mathsf{T}(t_{k+1}) + \int_0^{\tau_k} L_{\mathscr{C}}^\mathsf{T}(s) \mathcal{S} L_{\mathscr{C}}(s) \mathrm{d}s \\ &\approx \tilde{L}_F(t_{k+1}) D_k \tilde{L}_F^\mathsf{T}(t_{k+1}) + \sum_{j=1}^{n_{\mathscr{q}}} w_j L_{\mathscr{C}}^\mathsf{T}(c_j) \mathcal{S} L_{\mathscr{C}}(c_j) \\ &= L_F(t_{k+1}) D_F(t_{k+1}) L_F^\mathsf{T}(t_{k+1}).\end{aligned} \tag{2.55}$$

Here, the low-rank factors $L_F(t_{k+1})$ and $D_F(t_{k+1})$ are the block matrices

$$\begin{aligned} L_F(t_{k+1}) &= \left[\tilde{L}_F(t_{k+1}), L_{\mathcal{C}}^{\mathsf{T}}(c_1), \ldots, L_{\mathcal{C}}^{\mathsf{T}}(c_{n_q})\right], \\ D_F(t_{k+1}) &= \begin{bmatrix} D_k & & & \\ & w_1 \mathcal{S} & & \\ & & \ddots & \\ & & & w_{n_q}\mathcal{S} \end{bmatrix}. \end{aligned} \tag{2.56}$$

Here, $\{c_j\}_{j=1}^{n_q}$ and $\{w_j\}_{j=1}^{n_q}$ are the nodes and weights of the quadrature formula. The integral in Equation (2.55) does not depend on $\mathbf{X}_k$ and is the same for every time step. If the time step size is fixed, it can be approximated once in an offline phase. Even for time steps that are not equidistant, e.g. in case of an adaptive time discretization, the integral can be split into

$$\int_0^{\tau_k} L_{\mathcal{C}}^{\mathsf{T}}(s)\mathcal{S}L_{\mathcal{C}}(s)\mathrm{d}s = \int_0^{t_1} L_{\mathcal{C}}^{\mathsf{T}}(s)\mathcal{S}L_{\mathcal{C}}(s)\mathrm{d}s + \int_{t_1}^{\tau_k} L_{\mathcal{C}}^{\mathsf{T}}(s)\mathcal{S}L_{\mathcal{C}}(s)\mathrm{d}s \tag{2.57}$$

to avoid recomputing the same integral several times.

The splitting scheme is summarized in Algorithm 2.4. For clarity, only the Lie splitting case is considered, where the non-linear subproblem is solved over a full time step, then the affine subproblem is solved over a full time step. Other splitting schemes use different combinations of these subproblems, see, e.g. [Sti18a], and it is straightforward to adapt Algorithm 2.4 to these cases. The output and the basic inputs of Algorithm 2.4 are the same as in Algorithm 2.3. The only difference is in the parameters $\{c_j\}_{j=1}^{n_q}$ and $\{w_j\}_{j=1}^{n_q}$, which denote the quadrature nodes and weights of a quadrature formula.

2.4.2.3. Autonomous Splitting vs. BDF

Splitting schemes have several advantages compared to the BDF methods for autonomous DREs [BBSS24]. On the one hand, the accuracy is more reliable as the behavior of the error is more uniform along the time interval. The BDF methods have larger transient errors at the end of the time interval and can get more accurate at the beginning of the time interval. On the other hand, the splitting schemes have the ability to pre-compute parts of the solution in an offline phase and reuse it in every iteration step, which can make them computationally more efficient than the BDF methods. Further, they do not require additional initial solutions for higher order schemes as the BDF methods do. These features make them very promising as solvers for the non-autonomous DREs as well since they can be extended to the non-autonomous case to a certain extent. The details are presented in the next section of this work.

Algorithm 2.4: Autonomous (Lie) splitting scheme

```
Input: $\mathscr{A}, \mathscr{M}, \mathscr{B}, \mathscr{C}, \lambda, \mathscr{T}, L_0, D_0, \{c_j\}_{j=1}^{n_q}, \{w_j\}_{j=1}^{n_q}$
Output: $\mathscr{K}_k, k = 0, \dots, n_t - 1$
1 for $k = 0, \dots, n_t - 1$ do
2     Set $\hat{L} = L_k$ and compute $\hat{D} = (I + \tau_k D_k L_k^\mathsf{T} \mathscr{B}\mathscr{B}^\mathsf{T} L_k)^{-1} D_k$ as in
       Equation (2.53)
3     Compute $\tilde{L} = \mathrm{e}^{\tau_k \mathscr{M}^{-\mathsf{T}} \mathscr{A}^\mathsf{T}} \hat{L}$ as in Equation (2.54) and set $\tilde{D} = \hat{D}$
4     Approximate the integral term in Equation (2.55) by quadrature
       $\sum_{j=1}^{n_q} \tilde{L}_j \tilde{D}_j \tilde{L}_j^\mathsf{T}$, where $\tilde{L}_j = \tilde{\mathscr{C}} \mathscr{M}^{-1} \mathrm{e}^{c_j \mathscr{A} \mathscr{M}^{-1}}$, $\tilde{D}_j = w_j \mathcal{S}$
5     Column compress $L_k D_k L_k^\mathsf{T}$ with $L_k = \begin{bmatrix} \tilde{L} & \tilde{L}_1^\mathsf{T} & \cdots & \tilde{L}_{n_q}^\mathsf{T} \end{bmatrix}$ and
       $D_k = \operatorname{blkdiag}(\tilde{D}, \tilde{D}_1, \dots, \tilde{D}_{n_q})$
6     $\mathscr{K}_{n_t-k-1} = \frac{1}{\sqrt{\lambda}} \mathscr{B}^\mathsf{T} L_k D_k L_k^\mathsf{T} \mathscr{M}$
7 end
```

2.4.2.4. Non-autonomous Splitting Schemes

Following closely [BBSS24], the splitting schemes described in Section 2.4.2.2 can be partially extended also to the non-autonomous DRE (3.6), which is the relevant matrix equation for the feedback stabilization of the Stefan problem, as explained in Section 3.1.

The subproblems, which are analogues to Equations (2.48) and (2.50) are

$$\mathscr{M}^\mathsf{T} \dot{\mathbf{X}}_F \mathscr{M} = \mathscr{C}^\mathsf{T} \mathscr{C} + (\mathscr{A} + \dot{\mathscr{M}})^\mathsf{T} \mathbf{X}_F \mathscr{M} + \mathscr{M}^\mathsf{T} \mathbf{X}_F (\mathscr{A} + \dot{\mathscr{M}}), \tag{2.58}$$

$$\dot{\mathbf{X}}_G = -\mathbf{X}_G \mathscr{B}\mathscr{B}^\mathsf{T} \mathbf{X}_G. \tag{2.59}$$

The solution to Equation (2.59) is given by

$$\mathbf{X}_G(t) = \left(I + \mathbf{X}_G(s) \int_s^t \mathscr{B}(\tau)\mathscr{B}(\tau)^\mathsf{T} \mathrm{d}\tau \right)^{-1} \mathbf{X}_G(s) \tag{2.60}$$

for $s \leq t$. This is easily seen by multiplying by the inverted term from the left and then differentiating. The only difference when solving this subproblem in the non-autonomous case is that the integral $\int_s^t \mathscr{B}(\tau)\mathscr{B}(\tau)^\mathsf{T} \mathrm{d}\tau$ has to be evaluated instead of just $\mathscr{B}\mathscr{B}^\mathsf{T}$.

Additionally, for the affine subproblem (2.58), the two-parameter semigroup

$$\mathscr{T}(r, t) = \exp\left(\int_r^t \left((\mathscr{A}(\tau) + \dot{\mathscr{M}}(\tau)) \mathscr{M}(\tau)^{-1} \right)^\mathsf{T} \mathrm{d}\tau \right)$$

is introduced. The integrand is denoted by $\mathbb{Q}(\tau) = \left((\mathscr{A}(\tau) + \dot{\mathscr{M}}(\tau)) \mathscr{M}(\tau)^{-1} \right)^\mathsf{T}$. An important condition for deriving a solution for the affine subproblem is

$$\mathbb{Q}(t)\mathbb{Q}(s) = \mathbb{Q}(s)\mathbb{Q}(t) \tag{2.61}$$

for any t and s. Under this condition, also the integrals commute:

$$\int_r^t \mathbb{Q}(\tau)\mathrm{d}\tau \int_r^s \mathbb{Q}(\tau)\mathrm{d}\tau = \int_r^s \mathbb{Q}(\tau)\mathrm{d}\tau \int_r^t \mathbb{Q}(\tau)\mathrm{d}\tau.$$

As a consequence, $\mathbb{Q}(\tau)$ commutes with $\mathcal{T}(r,t)$ and

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{T}(r,t) = \mathbb{Q}(\tau)\mathcal{T}(r,t).$$

Then, the solution to Equation (2.58) satisfies

$$\mathbf{X}_F(t) = \mathcal{T}(r,t)\mathbf{X}_F(r)\mathcal{T}(r,t)^{\mathsf{T}} + \int_r^t \mathcal{T}(s,t)\mathcal{M}(s)^{-\mathsf{T}}\mathcal{C}(s)^{\mathsf{T}}\mathcal{C}(s)\mathcal{M}(s)^{-1}\mathcal{T}(s,t)^{\mathsf{T}}\mathrm{d}s \tag{2.62}$$

for $t_0 \leq r \leq t$. This follows quickly by differentiating each of the two terms separately using the above identity and noting that the first term satisfies Equation (2.58) without the $\mathcal{C}(t)^{\mathsf{T}}\mathcal{C}(t)$ term, and the second satisfies Equation (2.58) but with $X_F(r) = 0$.

The required condition (2.61) holds if, for example, $\mathcal{A}(t) = \sigma(t)\bar{\mathcal{A}}$ and $\mathcal{M}(t) = \mu(t)\bar{\mathcal{M}}$ with constant matrices $\bar{\mathcal{A}}$ and $\bar{\mathcal{M}}$ and scalar functions $\sigma(t)$ and $\mu(t)$. This is a common application, e.g. when considering heat flow with a variable thermal conductivity. In case a different approach can be used to solve Equation (2.58) efficiently, the condition (2.61) is not required. However, condition (2.61) does not hold in general, especially not for the representation of the Stefan problem which is chosen in this thesis. More details can be found in Section 4.2, where the non-autonomous splitting schemes are compared to the non-autonomous BDF method from Section 3.3.

Analogous to Equation (2.53), the Sherman-Morrison-Woodbury matrix inversion lemma can be utilized to express the solution X_G of Equation (2.60) in low-rank form. With $\mathbf{X}_G(s) = L_G D_G L_G^{\mathsf{T}}$, the solution can be expressed as

$$\mathbf{X}_G(t) = L_G\Big(I + D_G L_G^{\mathsf{T}} \int_s^t \mathcal{B}(\tau)\mathcal{B}(\tau)^{\mathsf{T}}\mathrm{d}\tau L_G\Big)^{-1} D_G L_G^{\mathsf{T}}. \tag{2.63}$$

Since the integral does not depend on $\mathbf{X}_G(t)$ and the integrand has a low-rank factorization for each τ, it can easily and efficiently be approximated by applying a quadrature formula followed by a column compression step. This can potentially even be done in an offline phase.

For the affine problem, the constant term in Equation (2.62) is in low-rank form if the initial condition $\mathbf{X}_F(r)$ is, and so is the integrand in the integral term. Therefore, a quadrature rule can be applied to approximate the integral and column compress the resulting sum to acquire an approximation to $\mathbf{X}_F(t)$. For each of the terms of this sum, $\mathcal{T}(c,t)L$ needs to be evaluated for different values of c and L. For this purpose, the fact that it is the solution of the system

$$\mathcal{M}(s)^{\mathsf{T}}\dot{Y}(s) = \big(\mathcal{A}(s) + \dot{\mathcal{M}}(s)\big)^{\mathsf{T}}Y(s), \quad Y(r) = L, \tag{2.64}$$

Algorithm 2.5: Non-autonomous (Lie) splitting scheme

```
    Input: $\mathscr{A}(t), \mathscr{M}(t), \dot{\mathscr{M}}(t), \mathscr{B}(t), \mathscr{C}(t), \lambda, \mathscr{T}, L_0, D_0, \{c_j\}_{j=1}^{n_q}, \{w_j\}_{j=1}^{n_q}$
    Output: $\mathscr{K}_k, k = 0, \dots, n_t - 1$
1   Invert the direction of time in all matrix functions, e.g. $\mathscr{A}(t) \to \mathscr{A}(t_{\text{end}} + t_0 - t)$
2   for $k = 0, \dots, n_t - 1$ do
3       Set $\hat{L} = L_k$ and compute $\hat{D} = \left(I + D_k L_k^\mathsf{T} \int_{t_k}^{t_{k+1}} \mathscr{B}(\tau)\mathscr{B}(\tau)^\mathsf{T} \mathrm{d}\tau L_k\right)^{-1} D_k$ as in
         Equation (2.63)
4       Compute $\tilde{L} = \mathscr{T}(t_k, t_{k+1})\hat{L}$ by solving Equation (2.64) and set $\tilde{D} = \hat{D}$
5       Approximate the integral term in Equation (2.62) by quadrature
         $\sum_{j=1}^{n_q} \tilde{L}_j \tilde{D}_j \tilde{L}_j^\mathsf{T}$, where $\tilde{L}_j = \mathscr{T}(t_k + c_j, t_{k+1})\mathscr{M}(t_k + c_j)^{-\mathsf{T}}\mathscr{C}(t_k + c_j)^\mathsf{T}$,
         $\tilde{D}_j = \tau_k w_j I$
6       Column compress $L_{k+1}D_{k+1}L_{k+1}^\mathsf{T}$ with $L_{k+1} = \begin{bmatrix} \tilde{L} & \tilde{L}_1 & \cdots & \tilde{L}_s \end{bmatrix}$ and
         $D_{k+1} = \operatorname{blkdiag}(\tilde{D}, \tilde{D}_1, \dots, \tilde{D}_s)$
7       $\mathscr{K}_{n_t-k-1} = \frac{1}{\sqrt{\lambda}}\mathscr{B}(t_{k+1})^\mathsf{T} L_{k+1} D_{k+1} L_{k+1}^\mathsf{T} \mathscr{M}(t_{k+1})$
8   end
```

at the time $s = t$ is used.

In the autonomous case, a single integral can be approximated once and then reused in every splitting step, but in the non-autonomous case an integral term needs to be approximated in every step. Additionally, in the autonomous case, the semigroup property $T(r,t)L = T(r,s)T(s,t)L$ can be used to avoid recomputing the same values in the quadrature formula, see Equation (2.57). This is no longer possible in the non-autonomous case, because the semigroup will be applied to different L at different quadrature nodes. As a consequence, most of the benefits arising from the splitting that make these methods very competitive in the autonomous case are lost in the non-autonomous case.

The procedure is summarized in Algorithm 2.5. For clarity, only the Lie splitting case is considered, where first the non-linear subproblem is solved over a full time step, then the affine subproblem over a full time step. Other splitting schemes will use different combinations of these subproblems, see, e.g. [Sti18a], but it is straight forward to adapt Algorithm 2.5 to those cases. The output of Algorithm 2.5 is the same as for the autonomous version in Algorithm 2.4, and so are the basic inputs. The only difference is that the matrices are time-dependent and the time derivative of the mass matrix $\dot{\mathscr{M}}(t)$ is present.

The main drawback of the non-autonomous splitting schemes in the context of the Stefan problem are that the arising matrices for the DRE are not in the format to fulfill condition (2.61). Thus, the non-autonomous splitting schemes in their current state are not applicable for the Stefan problem.

Instead, the autonomous BDF method, which is described in this chapter, is extended for non-autonomous DREs in the next chapter. Furthermore, the LQR approach is adapted together with other methods from this chapter to derive, compute and apply a feedback stabilization for the Stefan problem.

CHAPTER 3

METHODS FOR THE FEEDBACK STABILIZATION OF THE STEFAN PROBLEM

Contents

The central chapter of this thesis is devoted to the main objective, namely the derivation, computation and application of a feedback stabilization for the two-phase two-dimensional Stefan problem. The foundation for this is laid in Chapter 2, where existing methods for similar stabilization problems and their subproblems are described. These methods are now adapted or extended to be suitable for the Stefan problem. The main feature of the Stefan problem that requires special treatment is the moving interface, which requires careful attention to the associated boundary conditions in the spatial discretization. Furthermore, it causes the matrices in the resulting DRE to be time-dependent. Finally, the simulation of the Stefan problem with the applied feedback stabilization can reveal special numerical issues that can be treated with adaptive time-stepping.

The main procedure for stabilizing the Stefan problem using LQR is summarized in Algorithm 3.1.

3.1. LQR for the Stefan Problem

In this section, the Riccati-feedback approach for the Stefan problem is formulated. After introducing the general principles of LQR in Section 2.3, the focus is on the non-autonomous character of this problem, which is induced by the moving interface and the consequently changing sub-domains. This results in time-dependent coefficients in Equation (2.5).

To formulate the LQR problem, the same quadratic cost functional $J(\zeta_\Delta^h, y^h, u_\mathcal{K}^h)$ is used as in Equation (2.31). The deviation of the system from the desired trajectory is tracked via the output, and the input costs are penalized with a weight factor $\lambda > 0$. Additionally, the difference state from Equation (2.30) at t_end is measured with S positive semidefinite. The cost functional is thus defined as

$$J(\zeta_\Delta^h, y^h, u_\mathcal{K}^h) = \frac{1}{2}\int_0^{t_\text{end}} \left\|y^h\right\|^2 + \lambda \left\|u_\mathcal{K}^h\right\|^2 \,\mathrm{d}t + \zeta_\Delta^h(t_\text{end})^\mathsf{T} S \zeta_\Delta^h(t_\text{end}). \tag{3.1}$$

This cost functional is minimized subject to a linear time-varying system similar to system (2.5). The matrices from Equation (2.9) that are used here for the Stefan problem are time-dependent due to the moving mesh resulting from the moving interface. This is the main difference to Section 2.3, and the corresponding LQR problem reads

$$\begin{aligned} &\min_{u_\mathcal{K}^h} J(\zeta_\Delta^h, y^h, u_\mathcal{K}^h) \\ &\text{subject to} \\ &\qquad \mathcal{M}(t)\dot{x}_\Delta^h(t) = \mathcal{A}(t)x_\Delta^h(t) + \hat{\mathcal{B}}(t)u_\mathcal{K}^h(t), \\ &\qquad y^h(t) = \mathcal{C}(t)x_\Delta^h(t). \end{aligned} \tag{3.2}$$

The unique solution to the LQR problem (3.2) is (see Theorem 3.1)

$$u_\mathcal{K}^h(t) = -\mathcal{K}(t)x_\Delta^h(t), \tag{3.3}$$

where the feedback gain matrix

$$\mathcal{K}(t) = \frac{1}{\lambda}\hat{\mathcal{B}}^\mathsf{T}(t)\mathbf{X}(t)\mathcal{M}(t) = \frac{1}{\sqrt{\lambda}}\mathcal{B}^\mathsf{T}(t)\mathbf{X}(t)\mathcal{M}(t) \tag{3.4}$$

requires the solution $\mathbf{X}(t) \in \mathbb{R}^{n\times n}$ of a DRE. For the Stefan problem, this is the large-scale matrix-valued non-autonomous generalized DRE

$$\begin{aligned} -\frac{\mathrm{d}}{\mathrm{d}t}(\mathcal{M}^\mathsf{T}\mathbf{X}\mathcal{M}) &= \mathcal{C}^\mathsf{T}\mathcal{C} + \mathcal{A}^\mathsf{T}\mathbf{X}\mathcal{M} + \mathcal{M}^\mathsf{T}\mathbf{X}\mathcal{A} - \mathcal{M}^\mathsf{T}\mathbf{X}\mathcal{B}\mathcal{B}^\mathsf{T}\mathbf{X}\mathcal{M}, \\ \mathcal{M}(t_\text{end})^\mathsf{T}\mathbf{X}(t_\text{end})\mathcal{M}(t_\text{end}) &= S. \end{aligned} \tag{3.5}$$

All coefficients of Equation (3.5) can be time-dependent (but the t-dependency is omitted for better readability). The coefficients of the DRE, at each time instance t, are the matrices $\mathcal{A}(t), \mathcal{M}(t) \in \mathbb{R}^{n\times n}$, $\mathcal{B}(t) \in \mathbb{R}^{n\times m}$, and $\mathcal{C}(t) \in \mathbb{R}^{p\times n}$ from Equation (2.9), where the input matrix $\mathcal{B}(t) = \frac{1}{\sqrt{\lambda}}\hat{\mathcal{B}}(t)$ is scaled with the weight factor from the cost functional (3.1). In order to solve the DRE (3.5), the time-derivative requires special treatment due to the time-dependent mass matrix $\mathcal{M}(t)$. The chain rule can be used to rearrange the left side in Equation (3.5):

$$-\frac{\mathrm{d}}{\mathrm{d}t}(\mathcal{M}^\mathsf{T}\mathbf{X}\mathcal{M}) = -\dot{\mathcal{M}}^\mathsf{T}\mathbf{X}\mathcal{M} - \mathcal{M}^\mathsf{T}\dot{\mathbf{X}}\mathcal{M} - \mathcal{M}^\mathsf{T}\mathbf{X}\dot{\mathcal{M}}.$$

The two terms containing $\dot{\mathcal{M}}(t)$ can be subtracted from Equation (3.5) to obtain a DRE with the time-derivative of $\mathcal{M}(t)$ moved to the right-hand side:

$$\begin{aligned}-\mathcal{M}^\mathsf{T}\dot{\mathbf{X}}\mathcal{M} &= \mathcal{C}^\mathsf{T}\mathcal{C} + (\dot{\mathcal{M}}+\mathcal{A})^\mathsf{T}\mathbf{X}\mathcal{M} + \mathcal{M}^\mathsf{T}\mathbf{X}(\dot{\mathcal{M}}+\mathcal{A}) - \mathcal{M}^\mathsf{T}\mathbf{X}\mathcal{B}\mathcal{B}^\mathsf{T}\mathbf{X}\mathcal{M},\\ \mathcal{M}(t_\text{end})^\mathsf{T}\mathbf{X}(t_\text{end})\mathcal{M}(t_\text{end}) &= S.\end{aligned} \tag{3.6}$$

Assumption 1:
The matrix pencil $\alpha\mathcal{M}(t) - \beta(\dot{\mathcal{M}}(t) + \mathcal{A}(t))$ is regular. ◊

According to [KM90a], that the matrix pencil is regular results from the underlying DAE being solvable. Note that Assumption 1 is observed to hold in the numerical experiments of Chapter 4.

Theorem 3.1:
If Assumption 1 holds, the unique solution to the LQR problem (3.2) is given by the stabilization function $u^h_{\mathcal{K}}$ defined by Equations (3.3) to (3.6). ◊

Proof. One result from [KM90a] is that the LQR problem (3.2) can be reduced to a standard control problem under certain conditions. These are [KM90a, condition (3.4) and (3.5)].

The condition [KM90a, condition (3.5)] simply requires that the singular value decomposition of the mass matrix at the final time t_end can be used such that

$$U(t_\text{end})\mathcal{M}(t) = \begin{bmatrix}\mathcal{M}_1(t)\\ 0\end{bmatrix} \qquad \text{for all } t \in [0, t_\text{end}],$$

with $U(t_\text{end})$ being the left singular vectors of $\mathcal{M}(t_\text{end})$. With the block structure of the matrices in Equation (2.8), condition [KM90a, condition (3.5)] is fulfilled even without the singular value decomposition and the resulting transformation that is performed in [KM90a].

In short, the condition [KM90a, condition (3.4)] requires that $\mathcal{M}(t)$ be continuous and not full rank[1]. In addition, the matrix pencil $\alpha\mathcal{M}(t) - \beta(\dot{\mathcal{M}}(t) + \mathcal{A}(t))$ must be

[1]If $\mathcal{M}(t)$ has full rank, the transformation into a standard control problem is not necessary. However, $\mathcal{M}(t)$ does not have full rank due to the algebraic Stefan condition.

regular and the index $\operatorname{ind}_\infty(\mathscr{M}(t), \dot{\mathscr{M}}(t) + \mathscr{A}(t)) \leq 1$. Together with Assumption 1, also [KM90a, condition (3.4)] is fulfilled and Equation (3.12) is already formulated in the form of [KM90a, Equation (3.12)–(3.16)]. Thus, the stabilization function $u^h_{\mathscr{K}}$ defined by Equations (3.3) to (3.6) is equivalent to the unique solution in [KM90a, Equation (2.19)]. Consequently, $u^h_{\mathscr{K}}$ is the unique solution to the LQR problem (3.2). □

The choice of several problem parameters is crucial in the design of the LQR problem just described. These parameters strongly influence the performance of the resulting feedback stabilization.

First, a very important choice in the design of the LQR problem is the weight parameter λ in the cost functional (3.1). A smaller λ causes the cost functional to be more dominated by the output deviation and the input cost term has less influence. Thus, a smaller λ results in more active feedback stabilization, which prevents the interface from deviating. However, it should also be noted that the λ is acting as a regularization parameter for the LQR problem. A smaller λ results in a more dominant quadratic term for the DRE (3.6). This can affect the accuracy of the solution and can make solving the DRE more computationally expensive.

Especially important for the purpose of deriving an effective feedback stabilization for the Stefan problem is the choice of the inputs and outputs to the system, i.e. the matrices $\mathscr{B}(t)$ and $\mathscr{C}(t)$. The inputs are typically chosen to act on boundary regions, as this is a more realistic setting for real-world applications. By selecting different parts of the boundary as the input boundary $\Gamma_u(t)$, different combinations of Neumann boundary conditions are available for the inputs, see Equation (2.1b) and Section 4.4.2.

When choosing the output $y(t)$, it is particularly important to keep in mind the goal of feedback stabilization. Since this goal is to stabilize the interface position, the output should track the deviation of the interface from the desired trajectory. However, the interface graph of Equation (2.14) is used only for the open-loop control problem because it introduces additional non-linearities. Applying any of the common methods for linearizing Equation (2.14) would result in equations that no longer contain enough information to sufficiently represent the interface position for the purpose of using it in the cost functional (3.1). Therefore, the interface graph is not included in the system (2.5), which represents the Stefan problem in the LQR problem. Thus, the interface position is not explicitly available as a state of the system and cannot be directly tracked in the cost functional (3.1).

Instead, on the one hand, the interface velocity can be monitored to indicate an interface deviation. As soon as the interface deviates from the desired trajectory also the interface velocity changes and indicates in which direction the interface moves. However, this alone is not sufficient to stabilize the interface position. It is possible that the interface moves with the same velocity at a different position than the desired trajectory. This would not be observable by the interface movement alone.

On the other hand, temperature measurements on the boundary can indicate an inter-

Algorithm 3.1: LQR for the Stefan problem

```
1 Solve the open-loop control problem (2.17) to get the reference trajectory
    and $\mathscr{A}(t)$, $\mathscr{M}(t)$, $\dot{\mathscr{M}}(t)$, $\mathscr{B}(t)$, $\mathscr{C}(t)$ of the LQR problem (3.2)
2 Solve DRE (3.6) with Algorithm 3.2 to get $\mathscr{K}_k, k = 1, \dots, n_t$
3 Apply $\mathscr{K}_k$, with Equation (3.3), in a forward simulation of the Stefan problem
    (see Section 3.4)
```

face deviation. Here, the measurements are taken at the boundary, which is a realistic setting for real-world applications. Especially, point measurements at the desired interface position on the boundary indicate if the interface is on the desired trajectory. This is because the temperature has to be equal to the melt temperature Θ_{melt} exactly at this point, see Equation (2.1d). In addition, temperature measurements at boundary regions near the desired interface position are options considered for the output. With these temperature measurements, the interface deviation can be sufficiently reconstructed that the resulting feedback stabilization is able to bring the interface position back to the desired trajectory even without the interface graph representation. More details and a comparison of these different choices are discussed in Section 4.4.

Besides the challenges arising from different choices for the problem parameters, the main computational effort is to solve the non-autonomous DRE (3.6) and compute the feedback gain matrix $\mathscr{K}(t)$ with the non-autonomous BDF method (Algorithm 3.2). Especially the presence of $\dot{\mathscr{M}}(t)$ imposes additional difficulties, which are addressed in Section 3.3.

A major focus of this thesis is to compute and apply a Riccati-feedback stabilization of the Stefan problem. The overall procedure for this is summarized in Algorithm 3.1 in three steps. The method that is chosen for the forward simulations of the Stefan problem in Lines 1 and 3 is a fractional-step-theta scheme (see Section 3.4) and for solving the DRE in Line 2 the non-autonomous BDF methods, which are described in Algorithm 3.2. The time discretization $\mathscr{T}_{\text{fwd}}^{\text{ref}}$, defined in Equation (2.10), is used in all three steps. In Line 2, it is used backwards in time ($\mathscr{T}_{\text{bwd}}$, Equation (2.11)) and, for the second forward solve in Line 3, additional time steps can be added adaptively. The time-adaptivity is used to prevent numerical issues that can occur with feedback stabilizations that have very large variation, as demonstrated in Section 4.3. A detailed description of this time-adaptive fractional-step-theta scheme with a relative input-based indicator function is presented in Section 3.4.

The references for the codes of the methods above are provided in Section 4.1 and the behavior of Algorithms 3.1 and 3.2 is showcased by means of several numerical experiments in Chapter 4.

The next section is dedicated to the special treatment of the boundary conditions related to the interface, which is essential for the successful computation of a feedback stabilization for the Stefan problem.

3.2. Boundary Conditions Related to the Interface

The treatment of most of the boundary conditions from Equations (2.1) and (2.3) can be straightforwardly done with common FEM techniques. However, the Stefan condition (2.3b) and the melt condition (2.1d) are of special importance for the present feedback stabilization problem. This section describes the special properties and treatment of these conditions defined on the interface.

In contrast, for the open-loop control problem, all boundary conditions can be treated with the common FEM approach to remove the corresponding degrees of freedom (DOFs) from the FEM matrices. This has no effect on the adjoint system, which is solved separately as a PDE system with its own adjoint boundary conditions. The analogue of the adjoint system for the LQR problem is the non-autonomous DRE (3.6). The coefficients of this DRE are the FEM matrices where the interface boundary conditions play a crucial role.

In this thesis, this is described in detail for the temperature $\Theta^h_\Delta(t) = \Theta^h(t) - \tilde{\Theta}^h(t)$ (in difference state and semi-discretized in space as in Section 2.3) and the melt condition (2.1d), which is a Dirichlet condition on the interface. For this, the DOFs corresponding to the interface are denoted as $\Theta^h_{\Delta,\Gamma}(t) = \Theta^h_\Delta(t)\big|_{\Gamma_{\text{int}}(t)}$, and it is assumed that

$$\Theta^h_\Delta = \begin{bmatrix} \bar{\Theta}^h_\Delta \\ \Theta^h_{\Delta,\Gamma} \end{bmatrix} \tag{3.7}$$

can be split accordingly, with $\bar{\Theta}^h_\Delta(t)$ being the remaining DOFs. With this, the coefficient matrices from Equation (2.7) and $C_\Theta(t)$ can be split analogously:

$$\begin{aligned} M_\Theta &= \begin{bmatrix} M_{\bar{\Theta}} & 0 \\ 0 & M_{\Theta_\Gamma} \end{bmatrix}, \; A_{\Theta\Theta} = \begin{bmatrix} A_{\bar{\Theta}\bar{\Theta}} & A_{\bar{\Theta}\Theta_\Gamma} \\ A_{\Theta_\Gamma\bar{\Theta}} & A_{\Theta_\Gamma\Theta_\Gamma} \end{bmatrix}, \; A_{\Theta\Upsilon} = \begin{bmatrix} A_{\bar{\Theta}\Upsilon} \\ A_{\Theta_\Gamma\Upsilon} \end{bmatrix}, \\ A_{\Upsilon\Theta} &= \begin{bmatrix} A_{\Upsilon\bar{\Theta}} & A_{\Upsilon\Theta_\Gamma} \end{bmatrix}, \; B_\Theta = \begin{bmatrix} B_{\bar{\Theta}} \\ B_{\Theta_\Gamma} \end{bmatrix}, \; C_\Theta = \begin{bmatrix} C_{\bar{\Theta}} & C_{\Theta_\Gamma} \end{bmatrix}. \end{aligned} \tag{3.8}$$

Note that $A_{\Upsilon\Upsilon}(t)$ remains the same. Then, the common FEM approach would be to remove the DOFs collected in $\Theta^h_{\Delta,\Gamma}$ from Equation (2.8):

$$\begin{bmatrix} M_{\bar{\Theta}} & 0 \\ 0 & 0 \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \bar{\Theta}^h_{\Delta} \\ \Upsilon^h \end{bmatrix} = \begin{bmatrix} A_{\bar{\Theta}\bar{\Theta}} & A_{\bar{\Theta}\Upsilon} \\ A_{\Upsilon\bar{\Theta}} & A_{\Upsilon\Upsilon} \end{bmatrix} \begin{bmatrix} \bar{\Theta}^h_{\Delta} \\ \Upsilon^h \end{bmatrix} + \begin{bmatrix} B_{\bar{\Theta}} \\ 0 \end{bmatrix} \mathbf{u}^h,$$

$$\mathbf{y}^h = \begin{bmatrix} C_{\bar{\Theta}} & 0 \end{bmatrix} \begin{bmatrix} \bar{\Theta}^h_{\Delta} \\ \Upsilon^h \end{bmatrix}.$$

This demonstrates that the matrices modeling the mesh movement Υ^h are also affected. In particular, the Stefan condition (2.3b) is not represented anymore as all relevant DOFs are removed. In fact, the matrix $A_{\Upsilon\bar{\Theta}}$ is an all-zero matrix (see Equation (2.7)). With this, crucial coupling information of the temperature and the mesh movement is neglected, the only source term in Equation (2.3) is removed, and the interface movement is not mapped into the coefficient matrices anymore. Further, the feedback gain matrix was observed to be numerically zero in experiments that were conducted with these matrices.

Consequently, a different treatment of the Dirichlet condition (2.1d) is required in this context where the DOFs corresponding to the interface are still present.

In order to treat the boundary conditions (2.1d) and (2.3b) appropriately for the present LQR problem, the semi-discrete version of the Dirichlet condition in Equation (2.1d) in terms of the difference state $\Theta^h_{\Delta}(t)$ can be reformulated to

$$\Theta_{\Delta,\mathrm{melt}} = \Theta^h_{\Delta}, \qquad \text{on } (0, t_{\mathrm{end}}] \times \Gamma_{\mathrm{int}}. \tag{3.9}$$

Moreover, both terms in Equation (3.9) are constant, i.e. $\Theta_{\Delta,\mathrm{melt}} = \Theta_{\mathrm{melt}} - \tilde{\Theta}_{\mathrm{melt}} = 0$, since the melt temperature Θ_{melt} and the reference melt temperature $\tilde{\Theta}_{\mathrm{melt}}$ are equal. Consequently $\Theta^h_{\Delta}(t)\big|_{\Gamma_{\mathrm{int}}(t)} = 0$ and therefore constant. Thus, also the time derivative equals zero, $\dot{\Theta}^h_{\Delta}(t)\Big|_{\Gamma_{\mathrm{int}}(t)} = 0$. As a result, by adding this time derivative to Equation (3.9), the Dirichlet condition in Equation (2.1d) can be replaced by the modified condition

$$\dot{\Theta}^h_{\Delta} = \Theta_{\Delta,\mathrm{melt}} - \Theta^h_{\Delta}, \qquad \text{on } (0, t_{\mathrm{end}}] \times \Gamma_{\mathrm{int}}. \tag{3.10}$$

To incorporate Equation (3.10) into the matrices in Equation (3.8), the same notation from Equation (3.7) is used:

$$\begin{aligned}
&M_\Theta = \begin{bmatrix} M_{\bar\Theta} & 0 \\ 0 & M_{\Theta_\Gamma} \end{bmatrix}, \quad \tilde A_{\Theta\Theta} = \begin{bmatrix} A_{\bar\Theta\bar\Theta} & A_{\bar\Theta\Theta_\Gamma} \\ 0 & -I \end{bmatrix}, \quad \tilde A_{\Theta\Upsilon} = \begin{bmatrix} A_{\bar\Theta\Upsilon} \\ 0 \end{bmatrix}, \\
&A_{\Upsilon\Theta} = \begin{bmatrix} A_{\Upsilon\bar\Theta} & A_{\Upsilon\Theta_\Gamma} \end{bmatrix}, \quad B_\Theta = \begin{bmatrix} B_{\bar\Theta} \\ B_{\Theta_\Gamma} \end{bmatrix}, \quad C_\Theta = \begin{bmatrix} C_{\bar\Theta} & C_{\Theta_\Gamma} \end{bmatrix}.
\end{aligned} \tag{3.11}$$

Like this, the conditions in Equations (2.1d) and (2.3b) are incorporated explicitly into the definition of $\tilde A_{\Theta\Theta}(t)$ and $\tilde A_{\Theta\Upsilon}(t)$. Regarding the remaining boundary conditions, the Neumann condition in Equation (2.1e) does not need any extra attention, since it is incorporated automatically. The Dirichlet boundary conditions in Equations (2.1c), (2.3c) and (2.3d) can be handled by the common approach to remove the rows and columns corresponding to the DOFs at the related boundary regions from the matrices as described above.

Finally, Equation (2.8) with the modified block matrices from Equation (3.11), and thus Equation (3.10) incorporated, reads

$$\begin{aligned}
\begin{bmatrix} M_\Theta & 0 \\ 0 & 0 \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \Theta^h_\Delta \\ \Upsilon^h_\Delta \end{bmatrix} &= \begin{bmatrix} \tilde A_{\Theta\Theta} & \tilde A_{\Theta\Upsilon} \\ A_{\Upsilon\Theta} & A_{\Upsilon\Upsilon} \end{bmatrix} \begin{bmatrix} \Theta^h_\Delta \\ \Upsilon^h_\Delta \end{bmatrix} + \begin{bmatrix} B_\Theta \\ 0 \end{bmatrix} \mathbf{u}^h, \\
\mathbf{y}^h &= \begin{bmatrix} C_\Theta & 0 \end{bmatrix} \begin{bmatrix} \Theta^h_\Delta \\ \Upsilon^h_\Delta \end{bmatrix}.
\end{aligned} \tag{3.12}$$

Notably, this modification preserves the DAE structure and index of Equation (2.8). Thus, the same index reduction technique that is described in Equation (2.9) can be applied with the modified matrices:

$$\begin{aligned}
\mathcal{M}(t) &= M_\Theta(t), \\
\mathcal{A}(t) &= \tilde A_{\Theta\Theta}(t) - \tilde A_{\Theta\Upsilon}(t) A^{-1}_{\Upsilon\Upsilon}(t) A_{\Upsilon\Theta}(t), \\
\hat{\mathcal{B}}(t) &= B_\Theta(t), \\
\mathcal{C}(t) &= C_\Theta(t).
\end{aligned} \tag{3.13}$$

In summary, it is essential to have the DOFs at the interface $\Gamma_{\text{int}}(t)$ represented in the spatial discretization for two reasons. First, the coupling of the temperature and the extended interface movement in Equation (2.3) would otherwise be neglected. Second, the goal of the feedback stabilization problem is to control the interface position.

Therefore, the DOFs at the interface are needed for the output that goes into the cost function (3.1).

The matrices in Equation (3.13) are the coefficients of the non-autonomous DRE (3.6). The numerical solution of this DRE is the main computational step to compute a feedback stabilization of the Stefan problem, which is one of the primary focuses of this thesis. So far, no method exists in the literature to solve this DRE numerically. Therefore, a major contribution of this thesis is to fill this gap with the non-autonomous BDF method described in the next section.

3.3. Non-autonomous BDF for the DRE

The solution of the DRE (3.6) is required to numerically compute the feedback gain matrix $\mathscr{K}_k = \mathscr{K}(t_k)$ and the feedback stabilization $u_k = u^h_{\mathscr{K}}(t_k)$ for $t_k \in \mathscr{T}_{\text{bwd}}$. For the matrices $\mathscr{B}(t)$ and $\mathscr{C}(t)$ in this DRE, only a small number of inputs and outputs are used, i.e. $m, p \ll n$. Consequently, it is assumed that $\mathbf{X}_k$ has a low (numerical) rank, motivated by [Sti18b]. This allows to use efficient low-rank methods for computing the numerical solution of the DRE, $\mathbf{X}_k = \mathbf{X}(t_k)$. As introduced in Section 2.4.2, $\mathbf{X}_k$ is approximated to high accuracy by the decomposition

$$\mathbf{X}_k \approx L_k D_k L_k^{\mathsf{T}},$$

where the matrices $L_k \in \mathbb{R}^{n \times s_k}$ and $D_k \in \mathbb{R}^{s_k \times s_k}$ have rank $s_k \ll n$. A promising method to solve the non-autonomous DRE (3.6) is the low-rank non-autonomous BDF method. This method is a key contribution of this thesis and described in Algorithm 3.2.

An input to Algorithm 3.2 are the time steps $\mathscr{T}_{\text{bwd}}$ in reverse order (see Equation (2.11)). The DRE is solved backwards in time because it replaces the adjoint equations of an open-loop control problem. Additional inputs to Algorithm 3.2 are the weight factor from the cost functional λ and the order of the BDF method $\wp \in \{1, 2, 3, 4\}$. Further, the low-rank factors $L_0, \dots, L_{\wp-1}, D_0, \dots, D_{\wp-1}$ of the initial values $\mathbf{X}_0, \dots, \mathbf{X}_{\wp-1}$ are required as inputs with sufficient accuracy to obtain the desired order of convergence $\wp$.

These initial values could be computed with a different DRE solver. But instead, the initial solution snapshots are computed with sufficiently small time steps of lower order BDF in the wind-up procedure given in Algorithm 3.3. This way, the method becomes entirely self-contained. No additional time steps are needed for $\wp = 1$, where the available initial solution $\mathbf{X}_0$ is sufficient, and for $\wp = 2$, where the additional initial solution $\mathbf{X}_1$ can be computed with one BDF 1 step [HV03, Section 3.5]. But, extra time steps are generated for $\wp \in \{3, 4\}$ with Algorithm 3.3.

This algorithm has the inputs $\mathscr{T}$, $\wp$, and $n_\wp$. With these, the smallest additional time step is of length $\tilde{\tau} = \frac{t_1 - t_0}{2^{n_\wp}}$. This is used, to compute small BDF steps with step size $\tilde{\tau}$ and increasing order from 1 up to $\wp - 1$. Then, BDF steps are computed with

Algorithm 3.2: Non-autonomous low-rank factor BDF method of order $\wp$

```
Input: $\mathscr{A}(t), \mathscr{M}(t), \dot{\mathscr{M}}(t), \mathscr{B}(t), \mathscr{C}(t), \lambda, \mathscr{T}_{\text{bwd}}, \wp, L_0, \dots, L_{\wp-1}, D_0, \dots, D_{\wp-1}$
Output: $\mathscr{K}_k, k = 1, \dots, n_t$
1 for $k = \wp, \dots, n_t$ do
2     $\mathscr{A}_k = \tau_k \beta(\dot{\mathscr{M}}(\hat{t}_k) + \mathscr{A}(\hat{t}_k)) - \frac{1}{2}\mathscr{M}(\hat{t}_k)$
3     $\mathscr{M}_k = \mathscr{M}(\hat{t}_k)$
4     $\mathscr{B}_k = \sqrt{\tau\beta}\mathscr{B}(\hat{t}_k)$
5     $\mathscr{C}_k^\mathsf{T} = \left[\mathscr{C}(\hat{t}_k)^\mathsf{T}, \mathscr{M}_k^\mathsf{T} L_{k-1}, \dots, \mathscr{M}_k^\mathsf{T} L_{k-\wp}\right]$
6     $\mathscr{S}_k = \begin{bmatrix} \tau\beta I_p & & & \\ & -\alpha_1 D_{k-1} & & \\ & & \ddots & \\ & & & -\alpha_\wp D_{k-\wp} \end{bmatrix}$
7     Solve ARE (3.14) for $L_k$ and $D_k$
8     $\mathscr{K}_{n_t-k} = \frac{1}{\sqrt{\lambda}}\mathscr{B}(\hat{t}_k)^\mathsf{T} L_k D_k L_k^\mathsf{T} \mathscr{M}_k$
9 end
```

time step lengths that are doubled $n_\wp$ times until the time steps coincide with the first $t_0, \dots, t_{\wp-1}$. Usually, $n_\wp = 10$ is chosen. Independently of n_t, this results in 10 extra time steps for BDF 3 and 22 for BDF 4. For implementation details of Algorithms 3.2 and 3.3, see [SKB, mess_bdf_dre.m].

In each step of the BDF method, an ARE is solved, which is similar to the ARE (2.44) related to the autonomous DRE (2.36). Here, the ARE related to the non-autonomous DRE (3.6),

$$\begin{aligned} 0 = {} & \mathscr{C}_k^\mathsf{T}\mathscr{S}_k\mathscr{C}_k + (\dot{\mathscr{M}}_k + \hat{\mathscr{A}}_k)^\mathsf{T}\mathbf{X}_k\mathscr{M}_k + \mathscr{M}_k^\mathsf{T}\mathbf{X}_k(\dot{\mathscr{M}}_k + \hat{\mathscr{A}}_k) \\ & - \mathscr{M}_k^\mathsf{T}\mathbf{X}_k\mathscr{B}_k\mathscr{B}_k^\mathsf{T}\mathbf{X}_k\mathscr{M}_k, \end{aligned} \tag{3.14}$$

is solved for the low-rank solution $\mathbf{X}_k \approx L_k D_k L_k^\mathsf{T}$. The difference between the AREs (2.44) and (3.14) is the presence of the time derivative of the mass matrix $\dot{\mathscr{M}}_k$

Algorithm 3.3: BDF start-up time-steps

```
    Input: $\mathscr{T}, \wp, n_\wp$
    Output: $\mathscr{T}$
1 if $\wp > 2$ then
2     $\tau = t_1 - t_0$
3     $\tilde{\tau} = \frac{\tau}{2^{n_\wp}}$
4     $\tilde{t}_0 = t_0$
5     $\tilde{t}_1 = t_0 + \tilde{\tau}$
6     if $\wp = 3$ then
7         for $k = 2, \ldots, n_\wp + 2$ do
8             $\tilde{t}_k = t_0 + 2^{k-1}\tilde{\tau}$
9         end
10        $\mathscr{T} = \{\tilde{t}_0, \ldots, \tilde{t}_{n_\wp+2}, t_2, \ldots, t_{n_t}\}$
11    else if $\wp = 4$ then
12        for $k = 1, \ldots, n_\wp + 2$ do
13            $\tilde{t}_{2k} = t_0 + 2^k\tilde{\tau}$
14            $\tilde{t}_{2k+1} = t_0 + 3 \cdot 2^{k-1}\tilde{\tau}$
15        end
16        $\mathscr{T} = \{\tilde{t}_0, \ldots, \tilde{t}_{2n_\wp+5}, t_4, \ldots, t_{n_t}\}$
17    end
18 end
```

and the definition of the matrices

$$\begin{aligned}
\hat{\mathscr{A}}_k &= \tau\beta\mathscr{A}(\hat{t}_k) - \frac{1}{2}\mathscr{M}(\hat{t}_k), & \mathscr{M}_k &= \mathscr{M}(\hat{t}_k),\\
\mathscr{A}_k &= \dot{\mathscr{M}}_k + \hat{\mathscr{A}}_k, & \dot{\mathscr{M}}_k &= \tau_k\beta\dot{\mathscr{M}}(\hat{t}_k),\\
\mathscr{C}_k^\mathsf{T} &= \left[\mathscr{C}^\mathsf{T}, \mathscr{M}^\mathsf{T}(\hat{t}_k)L_{k-1}, \ldots, \mathscr{M}^\mathsf{T}(\hat{t}_k)L_{k-\wp}\right], & \mathscr{B}_k &= \sqrt{\tau\beta}\mathscr{B}(\hat{t}_k),\\
\mathscr{S}_k &= \begin{bmatrix} \tau\beta I_p & & & \\ & -\alpha_1 D_{k-1} & & \\ & & \ddots & \\ & & & -\alpha_\wp D_{k-\wp} \end{bmatrix},
\end{aligned} \tag{3.15}$$

which are all time-dependent. The coefficients α_j and β are the same as for the autonomous BDF method (see Table 2.1). Note that the formulation of the matrix $\mathscr{A}_k = \tau_k\beta(\dot{\mathscr{M}}(\hat{t}_k) + \mathscr{A}(\hat{t}_k)) - \frac{1}{2}\mathscr{M}(\hat{t}_k) = \dot{\mathscr{M}}_k + \hat{\mathscr{A}}_k$ in Equation (3.15) is equivalent to the definition in Line 2 of Algorithm 3.2.

Algorithm 3.2 is embedded in the open source software package M-M.E.S.S. 2.1 [SKB], where it benefits from efficient solvers for the ARE (2.44) such as the Newton-ADI, which is used in this thesis. It can accumulate the feedback gain matrices directly, avoiding the need to assemble the low-rank solution factors. This is a significant performance advantage in terms of the memory requirements during the computations, especially for large-scale DREs.

With the non-autonomous BDF method that is presented here, the non-autonomous DRE (3.6) can be solved efficiently and the feedback gain matrices $\mathscr{K}_k$ can be computed. This can then be applied to compute an optimal feedback stabilization (see Equation (3.3)) and stabilize the Stefan problem in a forward simulation. However, to handle possible numerical issues during the simulation of this closed-loop system, a time adaptive strategy is applied, which is described in the next section.

3.4. Adaptive Time-stepping for the Simulation of the Stefan Problem

Once feedback gain matrices $\mathscr{K}_k$ are computed with the non-autonomous BDF method from the previous section, it can be applied to the solution of the non-linear closed-loop system for the Stefan problem (2.1) and (2.3), as denoted in Equation (3.3) to get a closed-loop input $u^h_{\mathscr{K}}(t, \zeta^h_\Delta(t))$. In that case, the numerical simulation of this closed-loop system can be challenging if $u^h_{\mathscr{K}}(t, \zeta^h_\Delta(t))$ has very large variation. This is demonstrated in Section 4.3, where numerical issues can cause blow-ups. Similar numerical issues occur in [FW18] for FSI problems with, e.g. the implicit Euler method and the trapezoidal rule. There, the numerical issues can be overcome with time-adaptive fractional-step-theta schemes. Since these FSI problems model moving interfaces and use time-dependent meshes, similar to the Stefan problem, this section follows [FW18] and the references therein to overcome the potential blow-up behavior by using time-adaptive fractional-step-theta schemes for the closed-loop simulation of the Stefan problem.

For this purpose, the system is considered to be spatially semi-discretized with finite elements (see Sections 2.1.2 and 3.2). The specific spatial discretization is a problem-related choice, which is specified in Chapter 4. So the closed-loop system for the Stefan problem (2.1) and (2.3) in semi-discretized form can be formulated as

$$\dot{\zeta}^h_\Delta(t) = f^h(t; \zeta^h_\Delta(t), u^h_{\mathscr{K}}(t, \zeta^h_\Delta(t))) \tag{3.16}$$

as in Section 2.1.4, Equation (2.15).

For the time discretization, n_t sub-intervals $[t_{k-1}, t_k]$ are used to partition the time-interval $[t_0, t_{\text{end}}]$ with the time grid $\mathscr{T}_{\text{fwd}}$:

$$\begin{aligned}\mathscr{T}_{\text{fwd}} &= \{t_0, \dots t_{n_t} = t_{\text{end}}\},\\ \tau_k &= t_k - t_{k-1}.\end{aligned}$$

In contrast to the time discretizations $\mathscr{T}_{\text{fwd}}^{\text{ref}}$ for Line 1 in Algorithm 3.1 and $\mathscr{T}_{\text{bwd}}$ for the DREs (see Equations (2.10) and (2.11)), the time steps in $\mathscr{T}_{\text{fwd}}$ are not necessarily equidistant to allow time-adaptivity, but $\mathscr{T}_{\text{fwd}}^{\text{ref}}, \mathscr{T}_{\text{bwd}} \subset \mathscr{T}_{\text{fwd}}$. With these time steps, discrete approximations (ζ_k, u_k) are defined by

$$\begin{aligned} \zeta_k &\approx \zeta_{\Delta}^h(t_k), \\ u_k &\approx u_{\mathscr{K}}^h(t_k, \zeta_k), \end{aligned}$$

and the discrete approximation $(\zeta_{k,\text{ref}}, u_{k,\text{ref}})$ of the reference trajectory $(\zeta_{\text{ref}}, u_{\text{ref}})$ on the corresponding time grid $\mathscr{T}_{\text{fwd}}^{\text{ref}}$.

For the computation of these discrete approximations, the closed-loop system (3.16) in discrete form can be approximated by the time-stepping scheme

$$\frac{\zeta_k - \zeta_{k-1}}{\tau_k} = \Sigma f^h(\zeta_k, u_k) + (1 - \Sigma) f^h(\zeta_{k-1}, u_{k-1}), \tag{3.17}$$

with $\Sigma \in [0, 1]$ as in Equation (2.16).

In addition to Section 2.1.4, the described framework can be used to formulate more time-stepping schemes as well, where Σ can vary in between time steps. This is used to formulate the fractional-step-theta scheme in terms of the scheme (3.17).

3.4.1. Fractional-step-theta Scheme

Time-adaptivity is combined with the fractional-step-theta scheme to have a more reliable and robust method regarding the numerical issues that are mentioned above. The fractional-step-theta scheme is known to be of second-order accuracy and to be A-stable [BGP87]. Another advantage is that it has little numerical dissipation, which is beneficial for many problems. To formulate this method in terms of Equation (3.17), the two parameters

$$\begin{aligned} \Theta &= 2 - \sqrt{2} \quad \text{and} \\ \beta &= \frac{\Theta - 1}{\Theta - 2} = 1 - \sqrt{\frac{1}{2}} \end{aligned}$$

are defined, following [Fai17]. In each time step, three sub-steps are performed, in which Equation (3.17) is solved with the parameters

$$\begin{aligned} (\Theta_{k,0}, \Theta_{k,1}, \Theta_{k,2}) &= (\Theta, 1 - \Theta, \Theta), \\ (\tau_{k,0}, \tau_{k,1}, \tau_{k,2}) &= (\beta\tau_k, (1 - 2\beta)\tau_k, \beta\tau_k). \end{aligned}$$

For each sub-step, the initial condition is the solution of the previous sub-step.

This fractional-step-theta scheme can be combined with a time-adaptive strategy as described next.

Algorithm 3.4: Time-adaptive step-size control

```
    Input: $t_k, t_{k+1,\mathrm{ref}}, \zeta_k, u_k, \tau_k, \mathsf{TOL}, \gamma, r, \underline{\delta}, \overline{\delta}$
    Output: $\tau_{k+1}$, RetryStep
 1  Compute $\mathcal{I}_k \in \left\{\mathcal{I}_k^e, \mathcal{I}_k^{\Delta u}, \mathcal{I}_k^{\dot{u}}\right\}$        // choose one variant
 2  $\delta = \left(\gamma \frac{\mathsf{TOL}}{\mathcal{I}_k}\right)^r$
 3  RetryStep $= \delta < \underline{\delta}$
 4  if $\underline{\delta} < \delta < \overline{\delta}$ then
 5      $\tau_{k+1} = \tau_k$
 6  else
 7      $\tau_{k+1} = \delta\tau_k$
 8  end
 9  if $t_k + \tau_{k+1} > t_{k+1,ref}$ then // ensure to meet the reference time steps
10      $\tau_{k+1} = t_{k+1,\mathrm{ref}} - t_k$
11  end
```

3.4.2. Time-adaptive Strategy

The numerical behavior of the methods for solving the closed-loop system significantly depends on the discretization in space and time. Namely, an equidistant time-discretization with too large time steps (n_t too small) can lead to several numerical issues. For instance, for non-linear FSI problems, numerical blow-ups with the implicit Euler scheme and the trapezoidal rule have been observed in [FW18] even though these two methods are A-stable. Related to this, similar issues can appear when solving the closed-loop system (3.17). This can potentially be caused by small weight parameters λ in the cost functional (3.1), which can cause the feedback stabilization $u^h_{\mathcal{K}}(t_k, \zeta_k)$ to have a very large variation. Such numerical blow-ups can be prevented by fine time-discretizations. However, since $u^h_{\mathcal{K}}(t_k, \zeta_k)$ is not known a priori, also the required time step size is unknown. Especially, choosing a fine, equidistant time-discretization with possibly smaller time steps than necessary can become very computationally costly.

To overcome this difficulty, the time steps are refined and coarsened adaptively in the time-intervals that might suffer from numerical blow-ups. In order to determine whether the time steps are supposed to be larger, smaller, or stay the same, an indicator is used, which is denoted as $\mathcal{I}_k = \mathcal{I}(t_k, \zeta_k, u_k)$.

This indicator is used in Algorithm 3.4, the time-adaptive step size control, which is similar to [FW18, Algorithm 3]. From the indicator $\mathcal{I}_k$, the threshold δ is computed with the two parameters $0 < \gamma \leq 1$ and $r > 0$. The bounds, $\underline{\delta} \leq 1 \leq \overline{\delta}$, for this threshold set the interval in which the current solution is accepted and the step size stays the same. If δ is not in this interval, the algorithm makes one of two choices. Either $\overline{\delta} < \delta$ and the current solution is considered reasonable (RetryStep = False). As a result, the step size for the next time step is increased, since it is assumed that the time step was solved with more accuracy than necessary. Or $\delta < \underline{\delta}$ and the solution is recalculated (RetryStep = True), with the smaller step size $\delta\tau_k$. Further, Algorithm 3.4 returns the possibly updated step size τ_{k+1}.

Additionally, it is ensured that the time steps from $\mathscr{T}_{\text{fwd}}^{\text{ref}}$ are respected in Lines 9 and 10. For these time steps, the feedback gain matrix $\mathcal{K}_k$ is computed, as described in Section 3.1. Otherwise, linear interpolation is used to compute $\mathcal{K}_k$ between time steps from $\mathscr{T}_{\text{fwd}}^{\text{ref}}$.

A crucial part of this strategy are the heuristic indicators $\mathcal{I}_k$ computed in Line 1. Three different variants of these indicators are defined here, which indicate whether the time step size should be changed adaptively. One is an error-based indicator, which comes with significantly higher computational costs, while the other two monitor the feedback stabilization and have no extra computational costs.

Error-based Indicator This indicator is based on a heuristic error estimation. First, the solution $(\tilde{\zeta}_k, \tilde{u}_k)$ is computed with one step of the fractional-step-theta scheme with step size τ_k. Additionally, (ζ_k, u_k) is computed with three steps of step size $\frac{\tau_k}{3}$. Then, the indicator

$$\mathcal{I}_k^e = \left\| \chi(\tilde{\zeta}_k) - \chi(\zeta_k) \right\|_2$$

estimates the error between $\tilde{\zeta}_k$ and ζ_k. The function $\chi\colon \mathbb{R}^n \mapsto \mathbb{R}^{n_e}$, with $n_e \leq n$, selects certain entries from the solution vector. It can be the identity as well ($n_e = n$).

This indicator is supposed to refine the time steps especially in time-intervals that might suffer from numerical issues. Since these issues strongly depend on the time-discretization, $\mathcal{I}_k^e$ is expected to be large in these time-intervals. However, the computation of $\mathcal{I}_k^e$ requires extra computational effort. A total of four time steps are computed instead of one, quadrupling the cost per time step. The approximate solution with the finer discretization, (ζ_k, u_k), is used to continue the time-stepping.

The two input-based indicators, that are described next, come without the extra computational effort.

Absolute Input-based Indicator It is assumed that the reference time-discretization $\mathscr{T}_{\text{fwd}}^{\text{ref}}$ is chosen such that the reference trajectory snapshots $(\zeta_{k,\text{ref}}, u_{k,\text{ref}})$ are computed with sufficient accuracy. Thus, no further refinement is required as long as the feedback stabilization u_k is inactive. This is the case as long as the state ζ_k does not deviate from $\zeta_{k,\text{ref}}$. Therefore, no numerical issues occur during the solution of the closed-loop system as long as the feedback stabilization is inactive. On the other hand, refinement of the time steps might be necessary, especially, when u_k changes quickly.

The indicator

$$\mathcal{I}_k^{\Delta u} = \|u_k - u_{k-1}\|_2$$

monitors the absolute change of the feedback stabilization. This indicator is computationally cheap to evaluate and tailored to the solution of closed-loop systems, i.e. the feedback stabilization, which is the focus of this manuscript.

Scaled Input-based Indicator Since $\mathcal{J}_k^{\Delta u}$ monitors an absolute value, this indicator might be sensitive when the time step size τ_k is large and less sensitive with a small time step size. Consequently, it can be strongly problem dependent. An alternative is to monitor the scaled change of the feedback stabilization:

$$\mathcal{J}_k^{\dot{u}} = \left\| \frac{u_k - u_{k-1}}{\tau_k} \right\|_2 .$$

The indicator $\mathcal{J}_k^{\dot{u}}$ is an approximation to the norm of the gradient of the feedback stabilization. Accordingly, it indicates to refine the time steps if u_k has very large variation as well. At the same time, it is more robust with respect to the time step size.

Another possible approach is to combine the two indicators $\mathcal{J}_k^{\Delta u}$ and $\mathcal{J}_k^{\dot{u}}$ to monitor the absolute and scaled change of the input simultaneously. However, this approach is not considered in this thesis because $\mathcal{J}_k^{\Delta u}$ and $\mathcal{J}_k^{\dot{u}}$ behave very similarly (see Section 4.3). The difference is mainly one of scaling, and thus it is not expected that a combination of both indicators will provide any benefit.

The main goal of this thesis is to derive and compute a feedback stabilization for the Stefan problem, and then to apply this feedback stabilization in a closed-loop simulation. The methods to achieve this goal are presented in this chapter and summarized in the core Algorithm 3.1. In particular, the moving interface and the resulting non-autonomous nature of the Stefan problem are addressed. With the results of this chapter, feedback stabilization can be computed and applied in a closed-loop simulation of the Stefan problem. These computations are investigated in several numerical experiments in the next chapter.

CHAPTER 4 NUMERICAL RESULTS

Contents

This chapter examines the core results of this thesis, presented in Chapter 3, regarding their numerical behavior and performance. All results are made reproducible by providing the codes and used data for the experiments that are presented in this chapter. The main computational effort for computing a feedback stabilization for the Stefan problem is to solve the non-autonomous DRE. Hence, the performance of the non-autonomous BDF methods is investigated in terms of accuracy and efficiency. Once the feedback gain matrix is computed, the next crucial step is to apply it in a forward simulation of the non-linear closed-loop Stefan problem in a reliable and efficient manner. For this, the behavior of the time-adaptive fractional-step-theta algorithm is laid out in detail. Finally, with these two components settled, the design of the LQR problem is discussed in terms of different choices for the problem parameters like the weight factor in the cost functional, inputs, outputs, and desired trajectories.

4.1. Code Availability

All codes and data to reproduce the presented results are available at [Bar24]. The non-autonomous BDF method is incorporated in the software package M-M.E.S.S. 2.1 [SKB]. Other used software packages are

- FEniCS 2018.1.0,
- NumPy 1.20.3,
- SciPy 1.3.3,
- MATLAB R2017b.

The computations are run on a computer with 2 Intel Xeon Silver 4110 (Skylake) CPUs with 8 cores each and 192GB DDR4 RAM on CentOS Linux release 7.9.2009.

4.2. Non-autonomous BDF for the DRE

In this section, the performance of the non-autonomous BDF method from Section 3.3 is examined in detail in terms of accuracy and computational efficiency. This is of special importance as solving the non-autonomous DRE is the step with the highest computational effort for computing a feedback stabilization for the Stefan problem. The first examples in this section relate to the Stefan problem, chosen so that a numerical error is computable and the non-autonomous BDF can be compared with existing methods. In detail, a small-scale finite-difference approximation of the Laplacian and a steel profile are used as models for these experiments. Both are non-autonomous and fulfill condition (2.61) such that the the non-autonomous splitting schemes from Section 2.4.2.4 are applicable for comparison. Additionally, the first two examples allow to compute the numerical error since they are available with sufficiently small matrix sizes. Then, the performance of the non-autonomous BDF method is examined for the large-scale fully non-autonomous DRE arising from the Stefan problem, which is the focus of this work.

4.2.1. Small-scale Academic Example

This sufficiently small example can be used to analyze the convergence behavior of the DRE solvers. In particular, the numerical errors and the order of convergence of the BDF methods and the splitting schemes are compared for this example. More specifically, it is shown how the BDF methods (Algorithm 3.2) of order 3 and 4 depend on the parameters for the BDF startup time steps of Algorithm 3.3. The runtimes of these large-scale methods are not compared because they would have little meaning in this small-scale example.

To generate this small academic example, a finite-difference approximation of the Laplacian on the unit square $[0,1]^2$ is used, and the matrices $\hat{\mathscr{A}} \in \mathbb{R}^{n\times n}, \hat{\mathscr{B}} \in \mathbb{R}^{n\times 3}, \hat{\mathscr{C}} \in \mathbb{R}^{1\times n}$ are assembled. Here, the three columns in $\hat{\mathscr{B}}$ correspond to distributed inputs on the squares $[0.25, 0.375]^2$, $[0.5, 0.625]^2$, and $[0.75, 0.875]^2$ respectively, while the output vector $\hat{\mathscr{C}} = \frac{1}{n}[1,\ldots,1]$ measures the average of all states. Then, all the matrices are multiplied with time-dependent scalar functions to get the time-dependent matrices

$$\begin{aligned}
\mathscr{A}(t) &= \left(1 + \frac{1}{2}\sin(2\pi t)\right)\hat{\mathscr{A}},\\
\mathscr{M}(t) &= \left(2 + \frac{1}{2}\sin(2\pi t)\right)\mathscr{I}_n,\\
\mathscr{B}(t) &= (3 + \cos(t))\hat{\mathscr{B}},\\
\mathscr{C}(t) &= (1 - \min(t,1))\hat{\mathscr{C}}.
\end{aligned}$$

This example is closely related to the Stefan problem because it models the Laplacian operator, which is one of the terms in Equation (2.1a). Another important property of this example is that it satisfies condition (2.61). Thus, the splitting methods are applicable to the non-autonomous DRE (3.6) arising from this model. Additionally, the size of the matrices is set to $n = 25$ and the time interval is set to $[t_0, t_{\mathrm{end}}] = [0, 0.1]$. As explained in Section 2.4.1, a reference approximation $\mathbf{X}_{\mathrm{ref}}(t)$ is computed using `ode15s` from MATLAB after vectorization. With this in mind, the quality of the approximate solutions is evaluated by comparison of the relative error pointwise in time:

$$e_{\mathrm{DRE}}(t) = \frac{\|\mathbf{X}(t) - \mathbf{X}_{\mathrm{ref}}(t)\|_2}{\|\mathbf{X}_{\mathrm{ref}}(t)\|_2}.$$

Following the theory presented in [AP98, Section 5.2.3, Example 5.9], a BDF method with $\wp \leq 6$ stages should converge with order $\wp$ as well. As described in Section 3.3, the initialization must be sufficiently accurate to achieve these convergence orders. For $\wp = 3$ (BDF 3) and $\wp = 4$ (BDF 4), the required initial values are computed with the extra time steps from Algorithm 3.3. This example shows how different choices of the $n_\wp$ parameter used as input to Algorithm 3.3 can affect the order of convergence. However, for $\wp = 1$ (BDF 1) and $\wp = 2$ (BDF 2) no extra time steps are required.

Correspondingly, Figure 4.1 shows the estimated order of convergence (EOC) for BDF 3 and 4 (see Equation (A.1)). In detail, with no extra time steps or $n_\wp$ too small both methods converge with order 2 only. At the same time, for $n_\wp \geq 8$ both methods achieve the convergence order $\wp$ in this example. Moreover, larger values of $n_\wp$ have no (especially also no negative) further effect on the convergence. As a result, $n_\wp = 10$ is used for all further computations. Note that the choice of $n_\wp$ depends on the problem.

As a result, BDF 1, 2, 3, and 4 can fulfill their theoretical convergence orders as well as the splitting methods of order $\wp = 1$ (split 1) and $\wp = 2$ (split 2), as can be seen

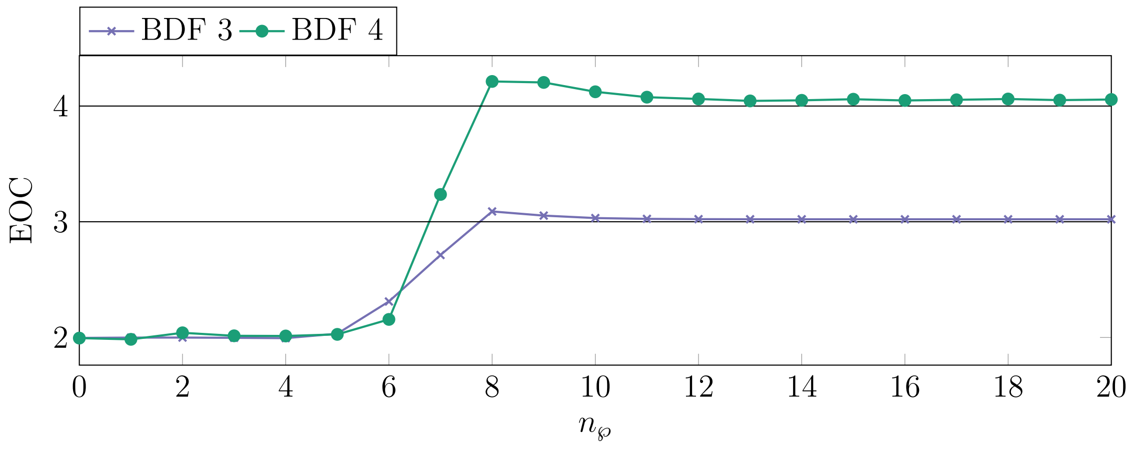

Figure 4.1.: Convergence orders of BDF 3 and 4 with different $n_\wp$, Small-scale Academic Example.

in Figure 4.2. This figure displays the error of the DRE solvers for different time step sizes. Another clear observation is that the splitting methods yield significantly smaller errors than the BDF methods of the same order.

This is confirmed by Figure 4.3, which shows the error of the different DRE solvers over the whole time interval for a fixed time step size. The BDF methods start with larger transient errors at t_{end}. Then, as the BDF methods solve backward in time, the errors decrease, resulting in BDF 4 being more accurate than the splitting methods in the first half of the time interval. A possible explanation for this is that the BDF schemes preserve fixed points, while the splitting schemes do not (see Equation (A.2)). At the same time, it is important to note that the errors of the splitting methods are varying less over the time-interval and stay on roughly the same level. Particularly for order $\wp = 1$ and $\wp = 2$, the splitting schemes outperform the BDF methods of the same order. In essence, the errors in Figure 4.3 are a typical behavior for these DRE solvers which is already known for the autonomous variants of BDF and splitting methods, see [Sti18a, Lan17].

For the next example, the non-autonomous BDF method presented in this thesis and the splitting methods benefit from the low-rank factorization of the solution, since the spatial discretizations can result in large-scale DREs.

4.2.2. Partially Non-autonomous Steel Profile

The semi-discretized heat transfer model for optimal cooling of steel profiles [HKS] is an autonomous model. Hence, the example is modified to be non-autonomous and the performance of the BDF and splitting methods is studied for different matrix sizes and weight parameters in Equation (3.1). This example is chosen for two reasons.

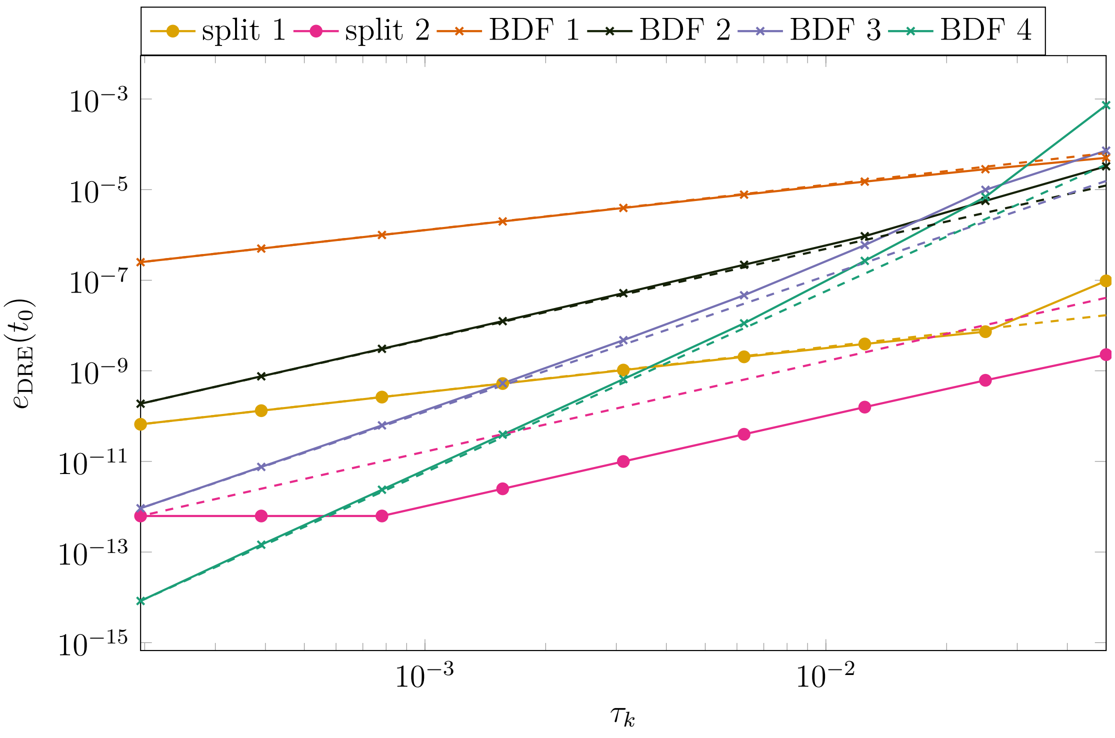

Figure 4.2.: Convergence of splitting and BDF and the theoretical convergence orders $\wp$ (dashed), Small-scale Academic Example.

First, the matrices $\mathcal{M}$, $\mathcal{S}$, $\mathcal{M}_\gamma$, $\mathcal{B}$ and $\mathcal{C}$ are available for different spatial discretizations $n \in \{109, 371, 1\,357, 5\,177,\ 20\,209, 79\,841\}$ and the model can easily be modified to be non-autonomous. Thus, it satisfies the condition (2.61), which is necessary for the splitting methods. The second reason is that this model is very close to the Stefan problem, which also models heat transfer.

In detail, the model for the steel profile example has 7 inputs and 6 outputs and the given matrices form a system equivalent to Equation (2.5):

$$\begin{aligned}\mathcal{M}\dot{x}^h &= (\frac{\kappa}{c\cdot\rho}\mathcal{S} + \frac{\gamma}{c\cdot\rho}\mathcal{M}_\gamma)x^h + \frac{\gamma}{c\cdot\rho}\mathcal{B}u^h,\\ y^h &= \mathcal{C}x^h.\end{aligned}$$

Here, the material parameters are

- $\kappa = 26.4\frac{kg\,m}{s^3\,{}^\circ C}$ - thermal conductivity,
- $c = 7\,620\frac{m^2}{s^2\,{}^\circ C}$ - heat capacity,
- $\gamma = 7.0164\frac{kg}{s^3\,{}^\circ C}$ - coefficient of thermal conductivity at the input boundary regions,

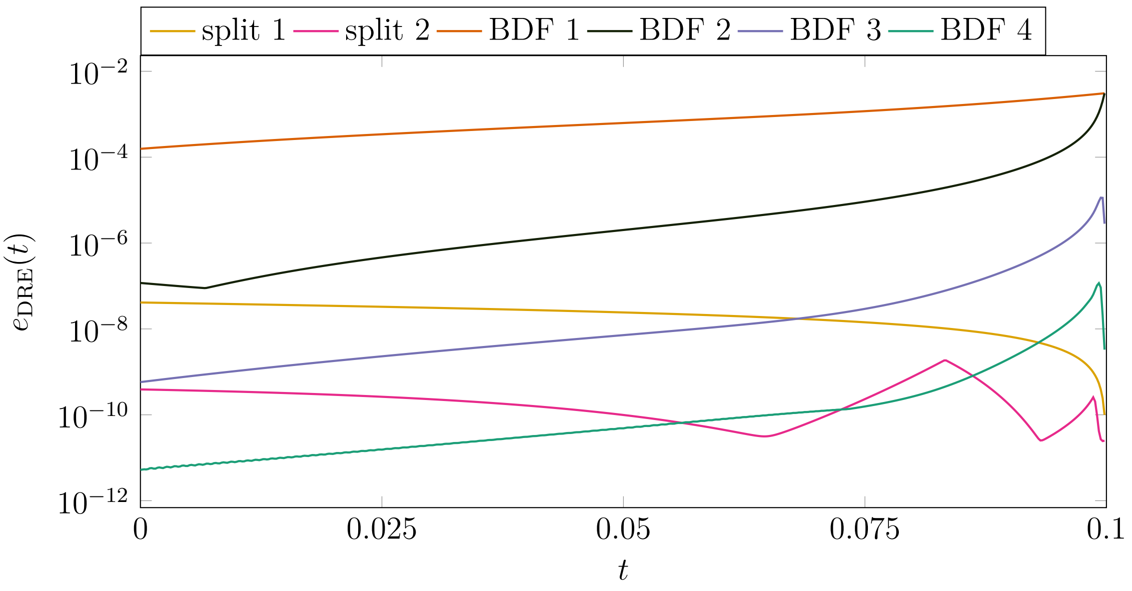

Figure 4.3.: Errors of splitting and BDF for $n_t = 512$ and $n_\wp = 10$, Small-scale Academic Example.

n	n_t	m	p	λ
109	128	7	6	1

Table 4.1.: Partially Non-autonomous Steel Profile default parameters (smallest example).

- $\rho = 654 \frac{kg}{m^3}$ - density.

Following [PS05], the real time-interval $[0, 45s]$ is used, which corresponds to model-time $[t_0, t_{\text{end}}] = [0, 4\,500]$. Furthermore, a non-linear, smooth and scalar function is added to the thermal conductivity to make this system partially non-autonomous:

$$\kappa(t) = 26.4 + 0.1 \cdot \left(2 + \cos\left(\frac{2 \cdot \pi \cdot t}{t_{\text{end}}}\right)\right) \frac{kg\, m}{s^3\, {}^\circ C}.$$

With this modification the example is similar to, e.g. [Lan17]. The difference lies in the scalar function $\kappa(t)$, where specifically a non-linear function is chosen here. Also, due to the constant mass matrix, Algorithm 3.2 is equivalent to the BDF method therein. In addition, the default matrix sizes, number of time steps, and weight parameter for this example are given in Table 4.1. Those are varied one by one to examine the behavior of the BDF and splitting methods.

Starting with different matrix dimensions n, Figure 4.4 displays the corresponding runtimes. The first thing to recognize is that the runtime of the BDF methods grows

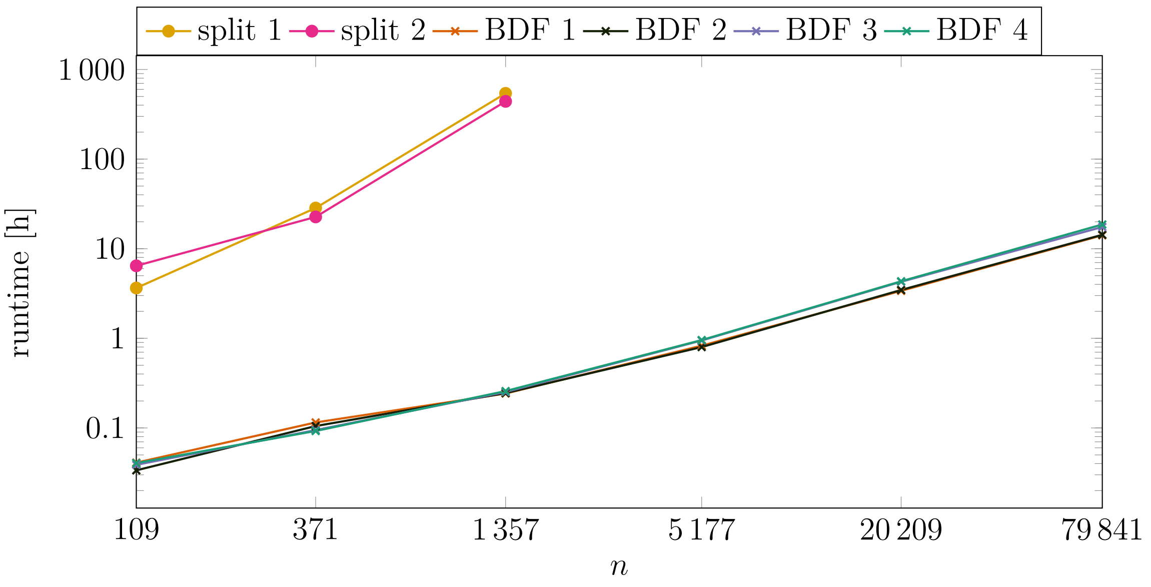

Figure 4.4.: Runtime of splitting and BDF for different n (matrix dimension), Partially Non-autonomous Steel Profile.

linearly with respect to the dimension and there is no significant difference between the orders $\wp \in \{1, \ldots, 4\}$. In contrast, the splitting methods require considerably more runtime. To illustrate, BDF methods take about 2.3 minutes for the smallest dimension $n = 109$, while splitting methods take 3.6 and 6.4 hours. In other words, the BDF methods are faster than split 1 by a factor of 94 and faster than split 2 by a factor of 167. For $n = 371$ it is 6.1 minutes versus 28.5 and 22.6 hours for BDF and splitting, respectively, which is a factor of 280 and 222 faster. Forthwith, for dimensions $n > 1\,357$, the computations are stopped since the splitting methods did not finish in reasonable time. Accordingly, the remaining tests are continued with $n = 109$ to have reasonable runtimes for the splitting methods even though, like this, the problem is not large-scale. However, this allows the computation of a reference solution and the comparison of numerical errors, which is not feasible for large-scale matrices.

Next, the runtime is compared in Figure 4.5 for $n_t \in \{2^2, \ldots, 2^{13}\}$, which grows with more time steps for all methods, as expected. Notably, the time needed to execute one time step is getting slightly smaller since the underlying subproblems converge faster. A possible explanation is that in Line 4 of Algorithm 3.2 the quadratic term of the ARE is scaled with $\sqrt{\tau}$. Moreover, the additional overhead to compute the initial values for BDF 3 and 4 with Algorithm 3.3 is visible for $n_t \in \{2^2, \ldots, 2^6\}$.

While Figure 4.5 shows only the runtime of the methods, Figure 4.6 allows to evaluate the efficiency of the methods by illustrating the runtime in relation to the relative error. Similar to the small scale example, a reference solution is computed through vectorizing

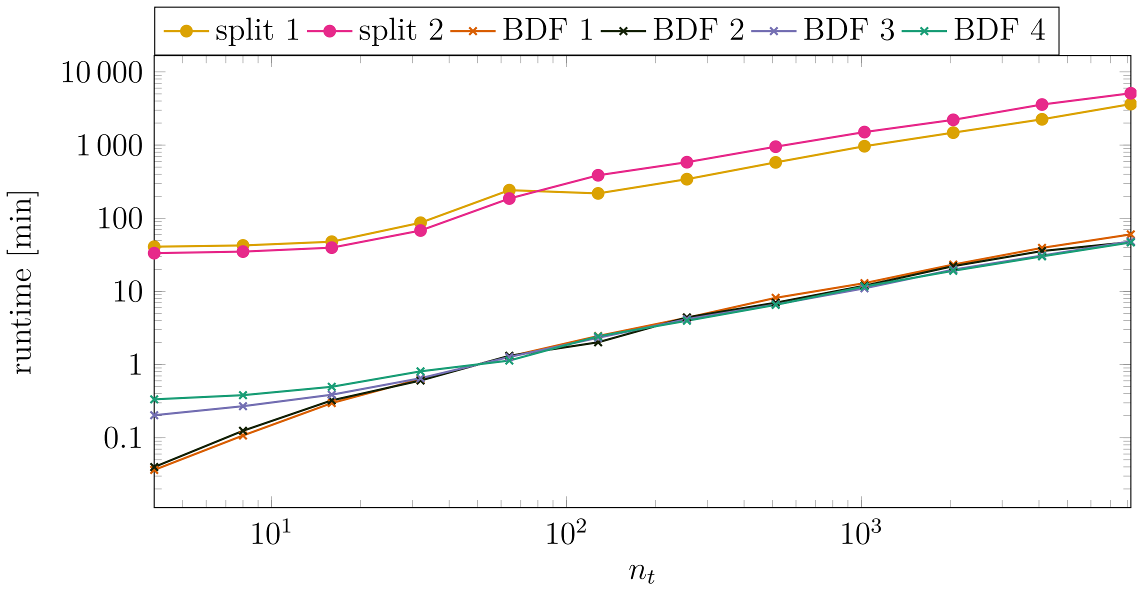

Figure 4.5.: Runtime of splitting and BDF for different n_t (no. of time steps), Partially Non-autonomous Steel Profile.

the DRE. In this example, only $\mathscr{A}(t)$ is time-dependent in the sense that it is a scalar function times a constant matrix. Therefore, it is sufficient to use the MATLAB function `ode45` and no stiff ODE solver is required. For better memory efficiency of the methods, the DRE solution is no longer stored, but only the feedback gain matrices $\mathscr{K}_k$. As a consequence, the errors are computed by comparing $\mathscr{K}_k$ at t_0:

$$e_{\mathscr{K}}(t_0) = \frac{\|\mathscr{K}_0 - \mathscr{K}_{0,\text{ref}}\|_2}{\|\mathscr{K}_{0,\text{ref}}\|_2}.$$

Figure 4.6 shows that the splitting methods achieve better accuracy than the BDF methods of the same order. However, the main computational benefit of the splitting schemes is lost in the non-autonomous case. While for an autonomous DRE only one integral term has to be approximated for the whole time interval, for a non-autonomous DRE such an integral approximation is needed in each time step. Therefore, the computational cost is increased by a factor roughly as large as the number of time steps. This is not the case for the BDF schemes, which therefore require significantly less computation time in a direct comparison. This effect can be mitigated by choosing larger tolerances for the splitting subproblem approximations. However, as seen in Figure 4.6 where this was tried for the 2nd-order splitting scheme, this runs the risk of destroying the overall convergence behavior. Also note that the BDF methods converge faster for higher orders, while requiring almost the same runtime for the same n_t. As an overall result of Figure 4.6, BDF 4 is the most efficient method for an accuracy below 10^{-5}.

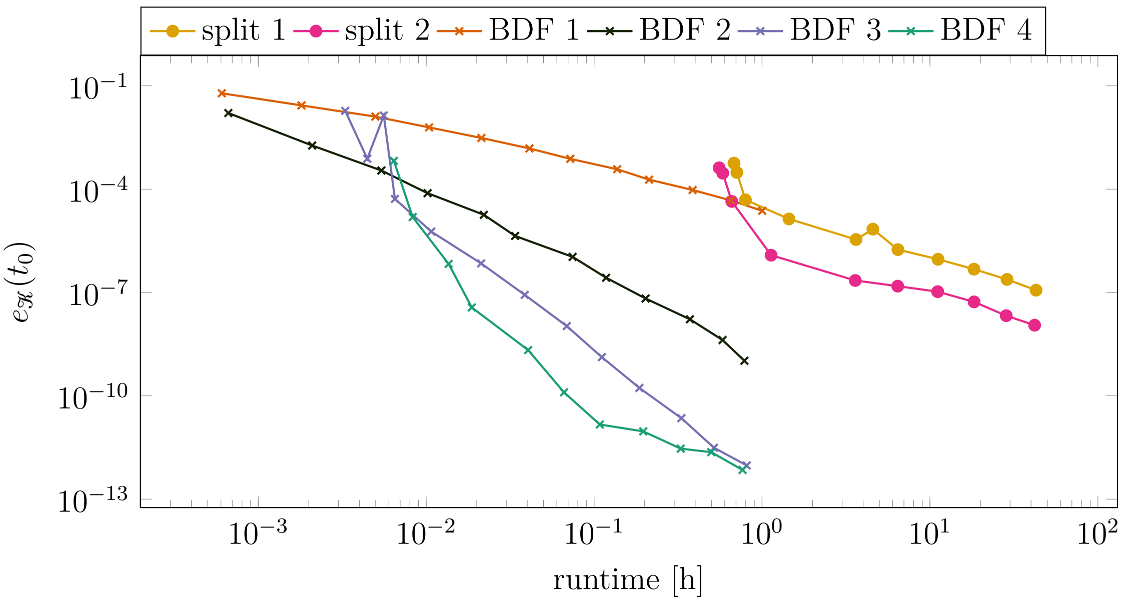

Figure 4.6.: Runtime vs. relative error (compared with ode45) of splitting and BDF, $n_t = 16\,384$, Partially Non-autonomous Steel Profile.

Further, the influence of the important parameters m, p, and λ on the runtime is studied. This is important because choosing larger m and p and smaller λ can significantly improve the performance of the resulting feedback stabilization. On the other hand, it increases the computational cost and a smaller λ reduces the regularity of the LQR problem. Therefore, a good trade-off between stabilization performance and computational cost must be chosen.

Regarding m, this model has 7 inputs and thus 7 columns in $\mathscr{B}(t)$ by default. On the one hand, to test the methods with $m \in \{1, \dots, 7\}$, $\mathscr{B}(t)$ is truncated after m columns. On the other hand, for $m \in \{8, \dots, 10\}$, additional columns are generated that are filled with random values. As a result, Figure 4.7 shows that the BDF methods need more time with those random columns, which are not physically motivated in the model. Besides this, the runtimes of the BDF and splitting methods are independent of m.

In contrast, the methods runtimes are affected by different numbers of rows in $\mathscr{C}(t)$. Figure 4.8 displays the runtimes for $p \in \{1, \dots, 10\}$. Again, the first 6 rows in $\mathscr{C}(t)$, which are defined in the model, are used for $p \leq 6$. This time, for $p \in \{7, \dots, 10\}$, additional physical measurements are added, which are constructed similar to the first 6 outputs. The resulting runtimes grow with more rows in $\mathscr{C}(t)$, but the BDF methods are affected more strongly in this respect. An explanation for this behavior can be found in the underlying iterative ARE solver (Newton-ADI). It generates solutions that have p times the number of iterations many columns. As a result, the ARE solution has significantly more columns for larger p. Hence, the column compression that is

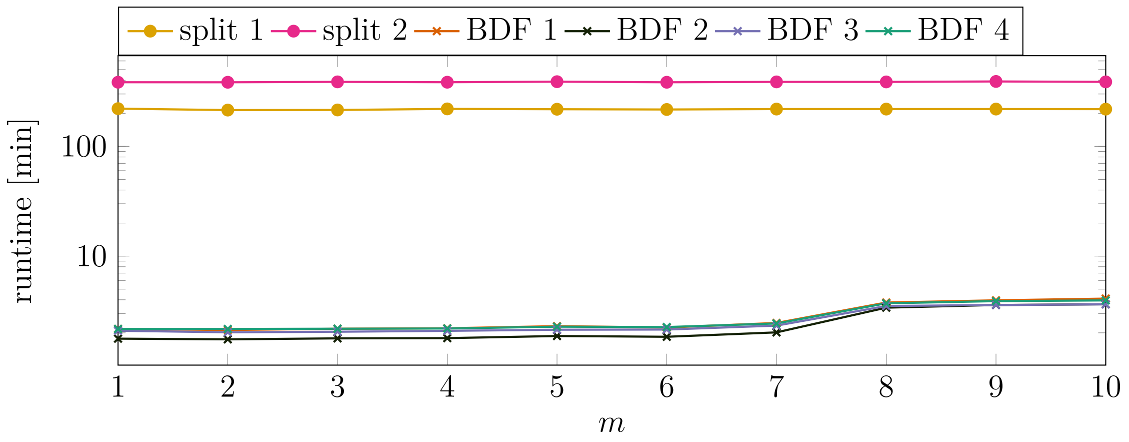

Figure 4.7.: Runtime of splitting and BDF for different m (columns in $\mathscr{B}(t)$), Partially Non-autonomous Steel Profile.

performed on these solutions is significantly more expensive.

Lastly, the weight λ in the cost functional (3.1) is an important parameter, as illustrated in Sections 4.3 and 4.4.1. For instance, the performance of the BDF methods is strongly affected by λ, which influences the balance between the quadratic and the constant term in the ARE (3.14). In detail, a smaller λ makes the quadratic part more dominant because the input matrix $\mathscr{B}(t) = \frac{1}{\sqrt{\lambda}}\hat{\mathscr{B}}(t)$ is scaled with λ. Thus, the inner ARE solver (Newton-ADI) in the BDF methods require more iterations to converge. This results in larger runtimes of the BDF methods for smaller λ as illustrated in Figure 4.9. In contrast, the runtimes of the splitting methods are not affected by λ since it only affects Equation (2.60), which is solved directly. However, note that the temporal discretization errors for both the BDF and splitting schemes depend on the problem being solved and thus on λ. To acquire an error of a specific size, differently sized time steps for different λ might be needed. Thus, the actual computational effort required for a certain accuracy varies for both classes of methods. A thorough investigation of these error structures is out of the scope of this work and could be the subject of future research.

A further observation is that the solutions computed by the splitting methods have a smaller numerical rank of $s(t_0) = 68$ compared to $s(t_0) = 84$ for the BDF methods (for more details see Figure A.1). This results in smaller memory requirements for the low-rank solution factors as well as lower computational effort to compute the feedback gain matrices $\mathscr{K}_k$. Ultimately, $\mathscr{K}_k$ is applied to generate a feedback stabilization and the actual DRE solution is not needed for this once $\mathscr{K}_k$ is computed. Thus, the on-disk memory requirement can be reduced by storing $\mathscr{K}_k \in \mathbb{R}^{n\times m}$ only, which is of fixed size and usually much smaller than the low-rank solution factors.

Overall, the splitting methods, in their current state, are not competitive for the

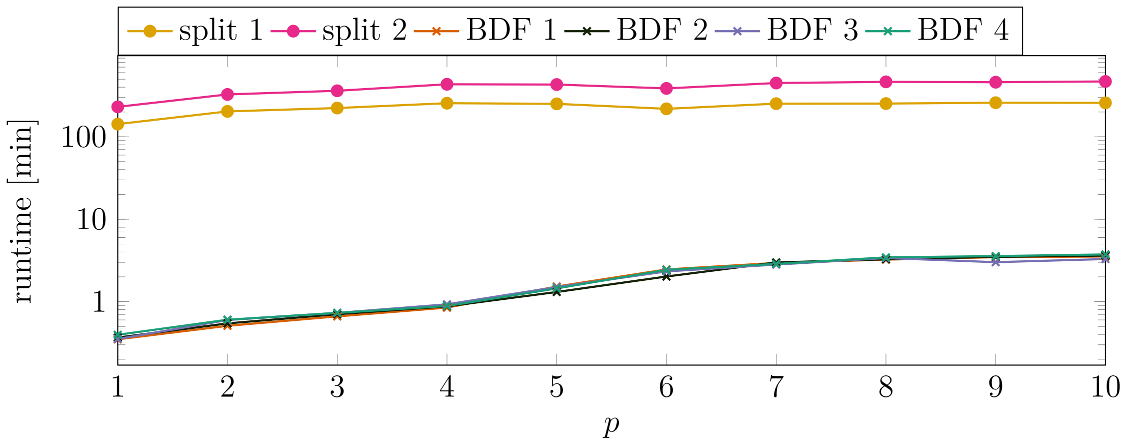

Figure 4.8.: Runtime of splitting and BDF for different p (rows in $\mathscr{C}(t)$), Partially Non-autonomous Steel Profile.

non-autonomous case. Other advantages of the splitting methods carry over to the non-autonomous case, like better predictability of the accuracy due to a more uniform error distribution and less sensitivity to several problem parameters. Besides the shorter runtimes, the BDF methods can handle more general time-dependencies in the coefficient matrices. They are not restricted to condition (2.61) or the case that, e.g. $\mathscr{A}(t) = \alpha(t)\bar{\mathscr{A}}$ and $\mathscr{M}(t) = \mu(t)\bar{\mathscr{M}}$ with constant matrices $\bar{\mathscr{A}}$ and $\bar{\mathscr{M}}$. In particular, this condition does not hold for the matrices arising from the Stefan problem in the present representation. Thus, the BDF methods are used exclusively for the Stefan problem, which is considered in the next example.

4.2.3. Two-dimensional Two-phase Stefan Problem

The Stefan problem, which is the focus of this work, provides large-scale DREs with time-dependent matrices. This time-dependency goes beyond condition (2.61). Both the splitting methods and the vectorization to an ODE are not applicable here. Hence, only the non-autonomous BDF method is used to compute the feedback gain matrices $\mathscr{K}_k$ for different combinations of problem parameters. Similar to the previous example, the influence of these problem parameters is investigated here, namely:

- n_t - number of time steps,
- m - number of inputs,
- p - number of outputs,
- λ - weight parameter.

In addition to the previous example, this non-linear problem is a large-scale DAE.

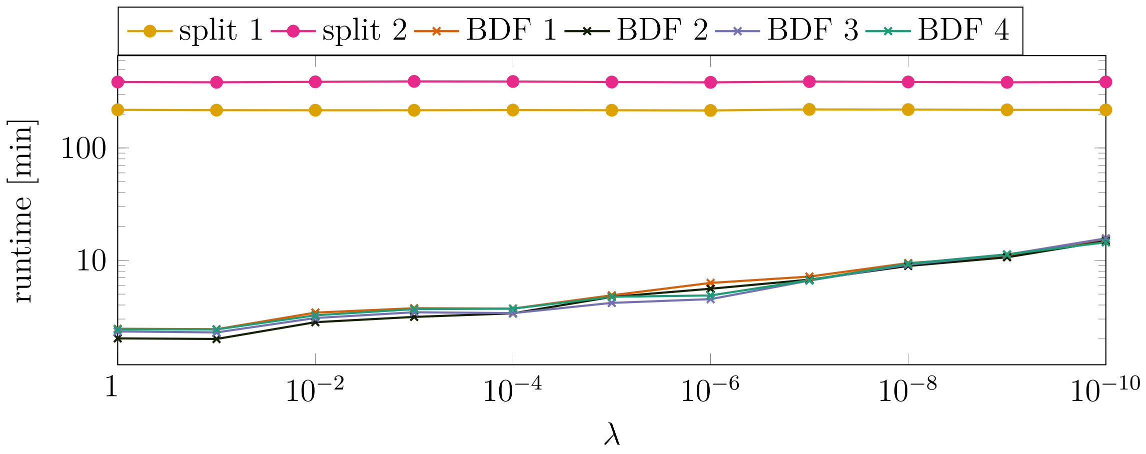

Figure 4.9.: Runtime of splitting and BDF for different λ (weight in cost functional), Partially Non-autonomous Steel Profile.

t_{end}	k_s	k_l	Θ_{cool}	Θ_{melt}	ℓ	Θ_0	τ_k
1	6	10	-1	0	10	$\Theta_{\text{cool}} \cdot (1 - 2 \cdot x_2)$	$\in [10^{-4}, 2.5 \cdot 10^{-3}]$

Table 4.2.: Two-dimensional Two-phase Stefan Problem model parameters

The particular domain $\Omega(t)$, chosen for this example, is a rectangle $[0, 0.5] \times [0, 1]$ with the initial interface position at height 0.5 as a horizontal line (see Figure 4.10).

For the discretization, a mesh of triangles with 3 899 vertices and 401 time-steps ($\mathscr{T}_{\text{fwd}}^{\text{ref}}$) is chosen (max. cell width: 0.0268, min. cell width: 0.0061). Using standard P1 elements in FEniCS, this results in 3 899 DOFs for $\Theta(t)$ and 7 798 DOFs for $\Upsilon(\nabla\Theta(t))$. After removing the DOFs corresponding to Dirichlet boundary conditions (see Section 3.2), the size of the full matrices from Equation (3.12) is $n = 11\,429$. The reduced matrices from Equation (3.13) have size $\tilde{n} = 3\,873$ but they are never computed explicitly. Instead, the full size, sparse matrices from Equation (3.12) are used for the computational operations. The model parameters are listed in Table 4.2.

The default parameters are depicted in Table 4.3. To examine their influence on the BDF methods, these are individually changed in the following experiments.

First, the number of time steps is varied with $n_t \in \{101, 401, 1\,001, 2\,001, 4\,001, 8\,001\}$. As a result, the runtime of the BDF methods grows linearly with n_t as can be seen in Figure 4.11. Another key point is that the runtimes for $\wp = 1, \ldots, 4$ are similar. A possible explanation for this is, that even though more columns are added to the constant term of the ARE with higher orders, these columns don't increase the numerical rank significantly. After adding these columns, a column compression is performed. For $n_t = 8\,001$, BDF 3 and 4 are even slightly faster than BDF 1 and 2. This observation results from the fact that the underlying ARE solver (Newton-ADI) requires fewer iterations to

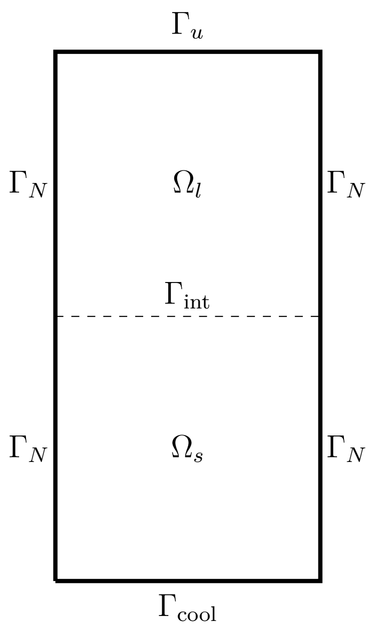

Figure 4.10.: One instance of the domain $\Omega(t) \subset \mathbb{R}^2$ of the Stefan problem.

n_t	m	p	λ	$n_\wp$
401	1	2	10^{-4}	10

Table 4.3.: Two-dimensional Two-phase Stefan Problem default parameters.

converge in this case.

As in the steel profile example (see Section 4.2.2), the number of inputs m, the number of outputs p, and the weight λ are varied in the cost functional (3.1). The results are consistent with these from the previous example and the resulting figures can be found in Appendix A.

In contrast to the previous example, only the low-rank BDF method is evaluated since the splitting methods are not applicable and the transformation to an ODE is not feasible. Thus, no reference solution is computed for the DRE and no exact errors can be compared. Instead, approximate errors are computed by comparing to the BDF 4 solution with the finest time-discretization ($n_t = 8\,001$) in Figure 4.12. Similar to the results in Figure 4.6, the BDF methods are more efficient with higher orders. However, the difference is smaller especially between BDF 3 and 4, which might not fully meet their order of convergence. Possible explanations for this are that the highly problem dependent parameter $n_\wp$ is chosen too small in this case or that additional approximation errors, e.g. from the implicit index-reduction techniques, accumulate and influence the

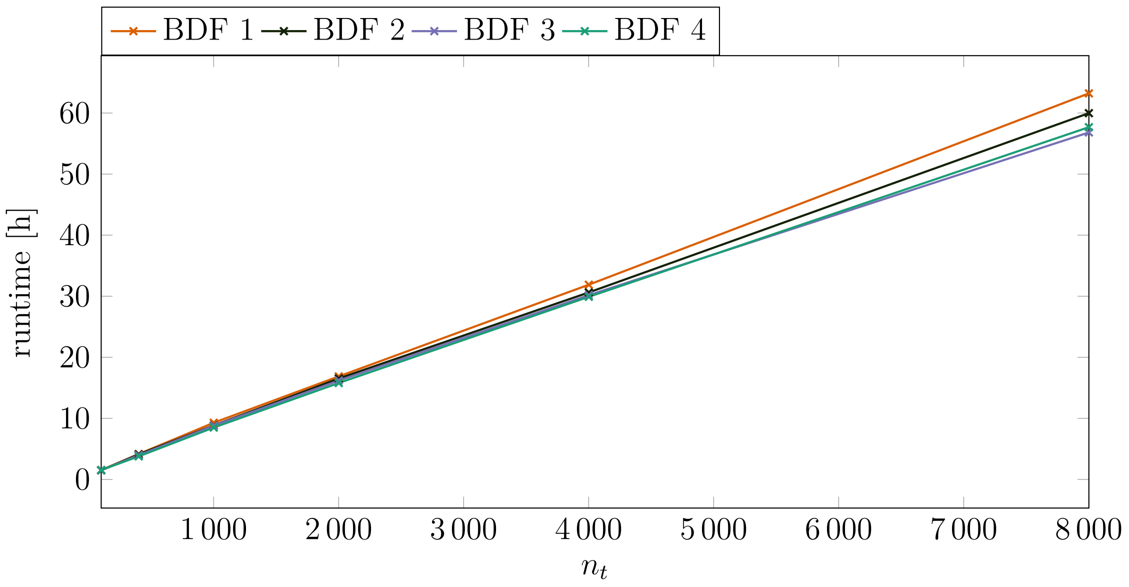

Figure 4.11.: Runtime of BDF for different n_t (no. of time steps), Two-dimensional Two-phase Stefan Problem.

convergence behaviour.

In summary, BDF 4 is the most accurate and computationally efficient method. Although it is of a higher order than BDF 1-3, it requires less runtime. Therefore, BDF 4 is the method of choice for numerically solving the non-autonomous DREs in the remainder of this thesis.

The main computational task for computing feedback stabilization for the Stefan problem is the numerical solution of the non-autonomous DREs. This section has shown that this task can be performed in an accurate and computationally efficient manner using the non-autonomous BDF methods, which is a major contribution of this thesis. The result that is used to compute feedback stabilizations from the numerical solution of the DREs is not the solution itself, but the feedback gain matrices computed with the solution (see Equation (3.4)). These feedback gain matrices can then be used in the closed-loop simulation of the Stefan problem to compute the actual feedback stabilization. This important step is performed for several examples in the next section using adaptive time stepping schemes.

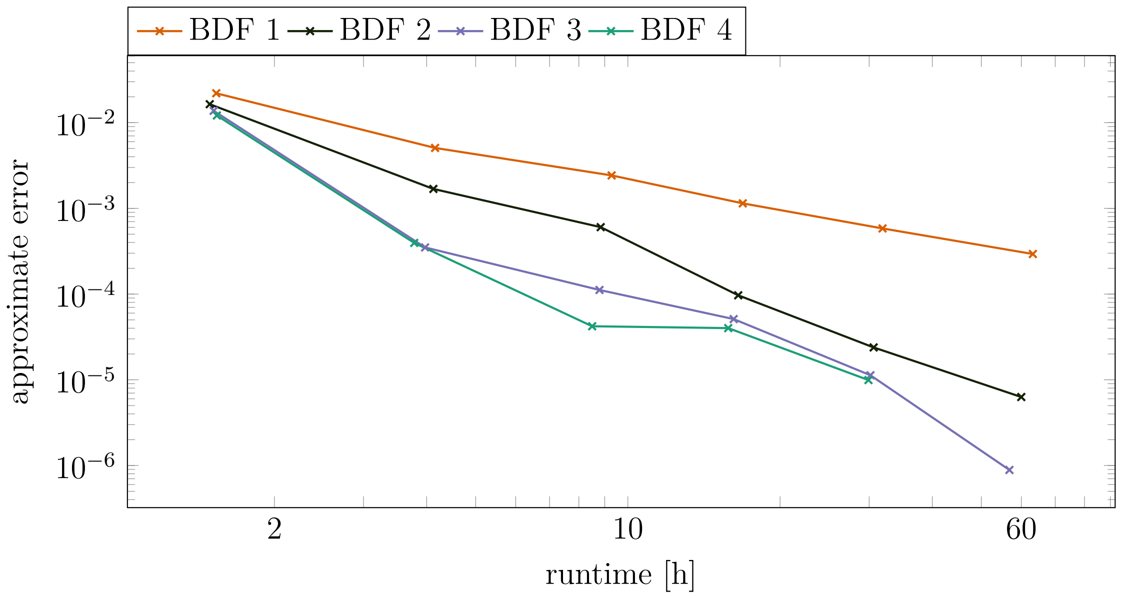

Figure 4.12.: BDF runtime vs. approximate error (compared with BDF 4, $n_t = 8\,001$), Two-dimensional Two-phase Stefan Problem.

4.3. Adaptive Time-stepping for the Simulation of the Stefan Problem

One of the main goals of this thesis is to compute and apply feedback stabilizations for the Stefan problem. This is done in a closed-loop simulation where the feedback gain matrices computed in the previous section are used to compute feedback stabilizations. In this section, it is shown that in these closed-loop simulations, numerical problems can lead to blow-up behavior when conventional time-stepping methods are used. This challenge is overcome by the adaptive time stepping methods of Section 3.4, which are specifically tailored to this numerical challenge and represent a key contribution of this thesis.

For this example, a perturbation to Equation (2.1) is introduced by augmenting the Dirichlet boundary condition (2.1c) at the bottom of the domain (see Figure 4.13) with the function $\varphi(t)$:

$$\Theta = \Theta_{\text{cool}} + \varphi, \qquad \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{cool}}.$$

Since the objective of the LQR problem is to stabilize the interface position from the discrete reference approximation $(x_{k,\text{ref}}, u_{k,\text{ref}})$, a single input $(m = 1)$ at the top of the domain is used as depicted in Equation (2.1b). Note that with the present formulation of the Stefan problem in Equation (2.1), the interface position cannot be measured explicitly. Thus, the interface position does not enter the cost functional (3.1). With

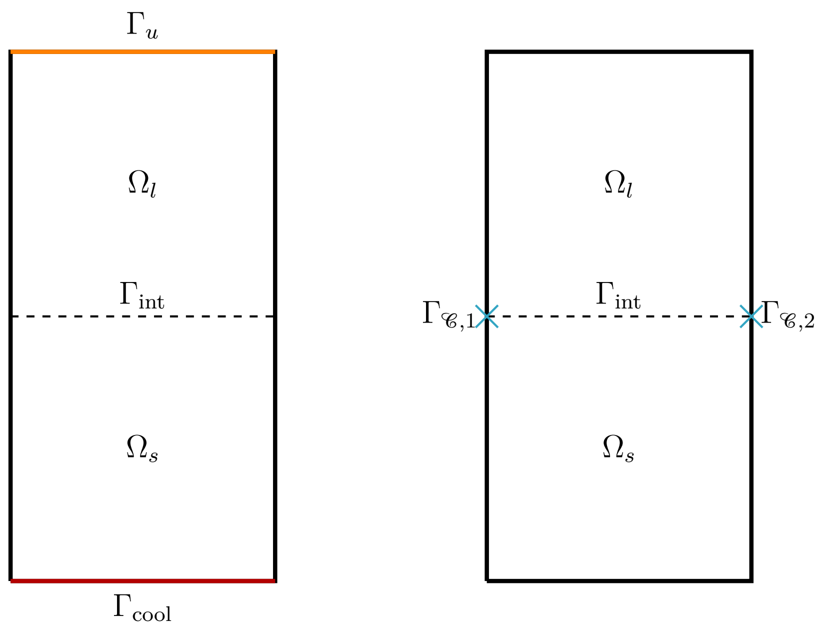

Figure 4.13.: Example of one input, one perturbation (left), and two point measurements (right)

this in mind, two temperature measurements at the walls Γ_{N} of the domain ($p = 2$) are used as output. These two outputs are intended to indicate the deviation of the interface from the reference position.

As a result, several feedback stabilizations are obtained by applying the feedback gain matrices computed in Section 4.2 with different weight factors λ. For these weight factors, the time-stepping methods implicit Euler (IE), trapezoidal rule (TR) and the fractional-step-theta scheme (FT) from Section 3.4.1 are tested to numerically solve Equation (3.16). A special feature of FT is that it is combined with the different time-adaptivity strategies of Section 3.4.2, which are the error-based indicator (FT-err), the absolute input-based indicator (FT-u), and the scaled input-based indicator (FT-dt-u). This comparison shows, which of the presented time-stepping methods is best suited for the closed-loop simulation of the Stefan problem.

Figure 4.14 displays the relative interface position

$$\begin{aligned}\Gamma_{\text{int},\Delta}(t, x^*) &= \Gamma_{\text{int,ref}}(t, x^*) - \Gamma_{\text{int}}(t, x^*),\\ x^* &= \underset{x_1 \in [0,0.5]}{\operatorname{argmax}} \left|\Gamma_{\text{int,ref}}(t, x_1) - \Gamma_{\text{int}}(t, x_1)\right|,\end{aligned}$$

at the point x^* on the interface which has the largest deviation from the reference trajectory. With a weight factor $\lambda = 10^{-4}$, the computed feedback stabilization successfully steers the interface back to the desired trajectory. In addition, with smaller weight factors, this can be achieved even quicker. Decreasing λ means that the cost functional (3.1)

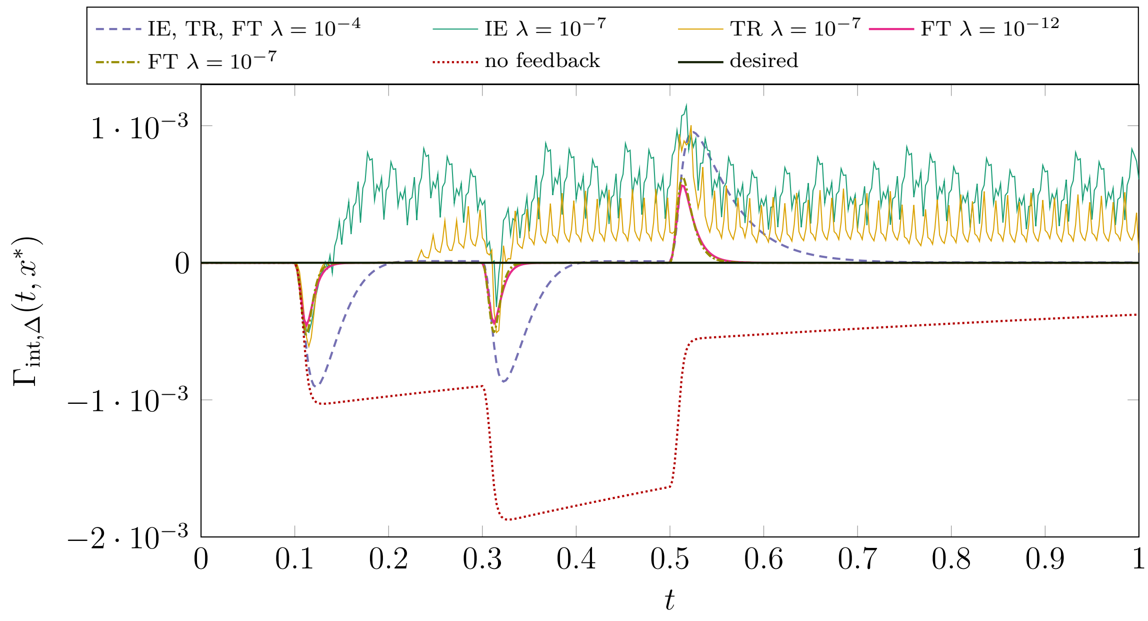

Figure 4.14.: Perturbed interface position relative to the reference trajectory with different feedback stabilizations.

is more dominated by the output deviation and the input cost term has less impact. Thus, a smaller λ leads to a more active feedback stabilization, which prevents the interface from deviating. In Figure 4.14 this is demonstrated with $\lambda \in \{10^{-4}, 10^{-7}, 10^{-12}\}$. Note that IE, TR, and FT give similar results for $\lambda = 10^{-4}$ and are not plotted separately because the difference is small enough not to be visible.

For smaller λ, the interface can be steered back even faster but IE and TR show a numerical blow-up behavior similar to the experiments in [FW18]. This is shown in Figure 4.14 for $\lambda = 10^{-7}$. In the experiments with the closed-loop simulation of the Stefan problem, these blow-ups occur while the feedback stabilization has very large variation and the time step size is too large to properly resolve this.

To overcome this issue, the time-adaptive FT is used, which uses smaller time steps to prevent these blow-ups. In Figure 4.14, this is demonstrated with $\lambda = 10^{-12}$ and FT-dt-u (the other time-adaptive strategies are discussed further below). Here, the computed feedback stabilization is able to steer the interface position back to the reference trajectory shortly after the perturbation.

The time-adaptive FT-dt-u comes with extra computational cost compared to IE and TR, which require both 15.6 minutes to numerically solve the closed-loop system with $n_t = 401$ time steps. On the contrary, for the same closed-loop system ($\lambda = 10^{-4}$), FT-dt-u computes 5 761 time steps and requires 546.3 minutes. This is partly due to the long time that is needed to steer the interface back to the desired trajectory. Hence, the input is active in this time period and the time-adaptivity is very expensive. In contrast, with $\lambda = 10^{-12}$, the interface is back to the reference position in shorter time and the feedback stabilization becomes inactive earlier as well as the time-adaptivity. Thus, FT-dt-u computes 2 405 time steps and requires 232.3 minutes. However, IE and TR fail with $n_t = 401$ and $\lambda \leq 10^{-7}$ and are therefore not suitable methods for closed-loop simulation of the Stefan problem, which justifies the additional computational effort of the time-adaptive strategies.

The time-adaptive strategy that produces the most reliable and computationally efficient behavior is FT-dt-u, which will now be discussed in detail with the results shown in Figure 4.15. The top part of Figure 4.15 shows the feedback stabilizations for $\lambda = 10^{-12}$ computed with FT-dt-u, FT-u, and FT-err. Further, the figure displays the perturbation $\varphi(t)$ as a dotted line. For clarity, it is important to note that the labeling of the y-axis is relative to the original boundary value Θ_{cool}. The center part of Figure 4.15 shows the corresponding perturbed interface trajectories, and the bottom part shows the different time step sizes.

An important parameter for the adaptive step size computation in Algorithm 3.4 is the tolerance TOL. With smaller tolerance, Algorithm 3.4 computes smaller time step sizes when the indicator grows. On the one hand, a larger tolerance can save some time steps by increasing the time step size earlier when indicator values are falling. This also reduces the computational effort. On the other hand, Algorithm 3.4 can fail to prevent a numerical blow-up if TOL is chosen too large and the time step sizes grow too early. This effect is presented in Figure 4.15 for FT-u with $\mathsf{TOL} = 10^{-2}$ and FT-err with $\mathsf{TOL} = 10^{-8}$. Both increase the time step size when the feedback stabilization is nearly inactive. However, in this case the time step is increased too early, causing blow-up behavior. This blow-up then leads to an increase of the indicator, and consequently a decrease of the time step size such that the feedback stabilizations become inactive again. The result is that the feedback stabilization is quickly bouncing back and forth between large and small values. To sum this up, for both, FT-u and FT-err, the choice of TOL is strongly problem dependent.

In contrast to this, the indicator FT-dt-u monitors the relative change of the feedback stabilization. Thus, as long as the feedback stabilization is active, the algorithm sets τ_k to the minimum value ($\tau_k = 0.0001$). However, when the input is inactive the algorithm sets τ_k back to the maximum value ($\tau_k = 0.0025$) as can be seen in Figure 4.15 (bottom, dashed line). This is expected to be the general behavior of FT-dt-u if TOL is chosen sufficiently small.

To emphasize a meaningful comparison, for FT-u, the indicator monitors the absolute change of the feedback stabilization. This means, with a smaller tolerance, FT-u behaves the same as FT-dt-u ($\mathsf{TOL} < 10^{-5}$). In particular, it depends on the magnitude of the feedback stabilization values, and the right choice of TOL is therefore strongly problem dependent. The numerical experiments performed indicate that FT-dt-u is more reliable with respect to the magnitude of the feedback stabilization values and thus the problem dependent choice of TOL. Also, a smaller TOL can be chosen to ensure reliable perfor-

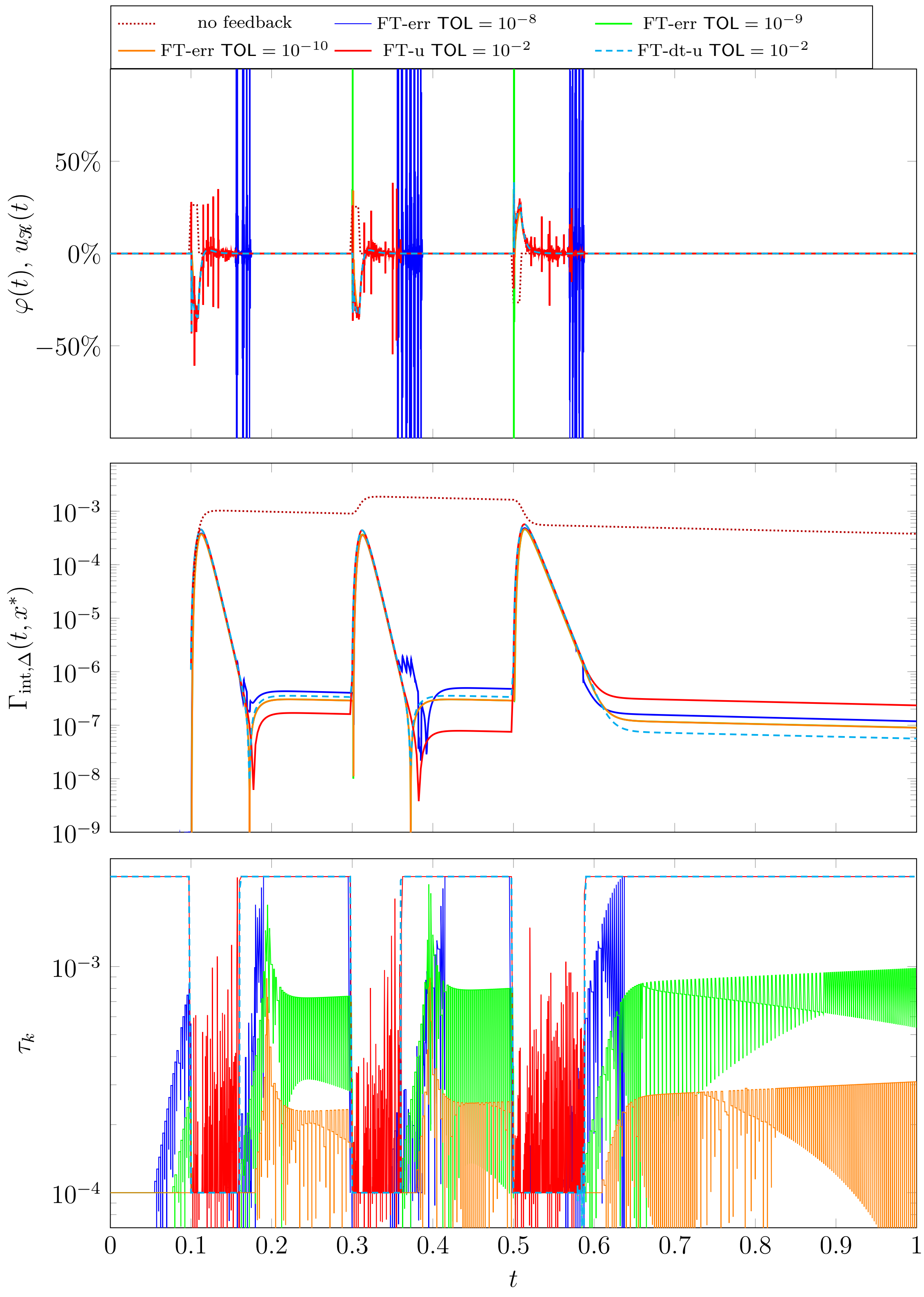

Figure 4.15.: Perturbation and different time-adaptive feedback stabilizations (clipped, max. value: 8.4, min. value: -39.3) (top), Perturbed interface position (center), and time step sizes (bottom) for $\lambda = 10^{-12}$.

mance and get the same n_t, since it jumps back and forth from the maximum to the minimum τ_k.

In contrast to the FT-u and FT-dt-u indicators, the FT-err indicator monitors the difference between the computed solutions for two different time step sizes. This results in a larger computational effort for a single time step compared to the alternative indicators. For instance, τ_k is reduced if the feedback stabilization has a very large variation and consequently the solution changes more. However, FT-err cannot distinguish between high activity of the reference and the perturbed solution. In addition, it enlarges τ_k gradually. Altogether, it requires significantly smaller TOL, larger n_t, and, consequently, increases the runtime to prevent a blow-up. In order to illustrate this, the computed time steps for three different choices of TOL are shown in Figure 4.15 (bottom). It can be seen that FT-err causes the time steps to jump back and forth between larger and smaller values very quickly. Some of this quick jumping is caused by restricting Algorithm 3.4 to match the reference time steps $\mathscr{T}_{\mathrm{fwd}}^{\mathrm{ref}}$. This can result in time step sizes smaller than the minimum τ_k if the adaptively computed time steps would otherwise skip the next time step from $\mathscr{T}_{\mathrm{fwd}}^{\mathrm{ref}}$.

The effects and differences between the time-adaptive strategies, which are described above also result in different runtimes and total number of time steps n_t, which are shown in Figure 4.16. Notably, IE and TR require 2.3 seconds per time step and it would take 390 minutes to solve the closed-loop system with 10 000 equidistant time steps. This number of time steps corresponds to the step size that FT-dt-u uses while the input is active. In comparison, FT-u requires 48% less runtime and 82% fewer time steps, and FT-dt-u requires 40% less runtime and 76% fewer time steps while using the same small time step size locally. Further, they simulate the closed-loop system more reliably in presence of strongly varying inputs due to their adaptivity, which is tuned to this problem. In detail, the runtime per time step is 6.9 seconds for FT-u and 5.8 seconds for FT-dt-u on average. FT-u takes longer on average because it discards time steps more often than FT-dt-u due to its attempt to take larger time steps too early. A discarded time step results in a re-computation of this time step with a smaller step size, which makes this time step more expensive. Finally, with around 21 seconds per time step and a larger number of time steps, FT-err is significantly more expensive than FT-dt-u.

As a consequence, FT-dt-u is chosen for the computations in the remaining sections, which discuss the influence of the problem parameters in the LQR-design for the Stefan problem.

4.4. LQR for the Stefan Problem

The last two sections showed the importance of the methods for the computation and numerical application of the feedback stabilization for the Stefan problem. In addition,

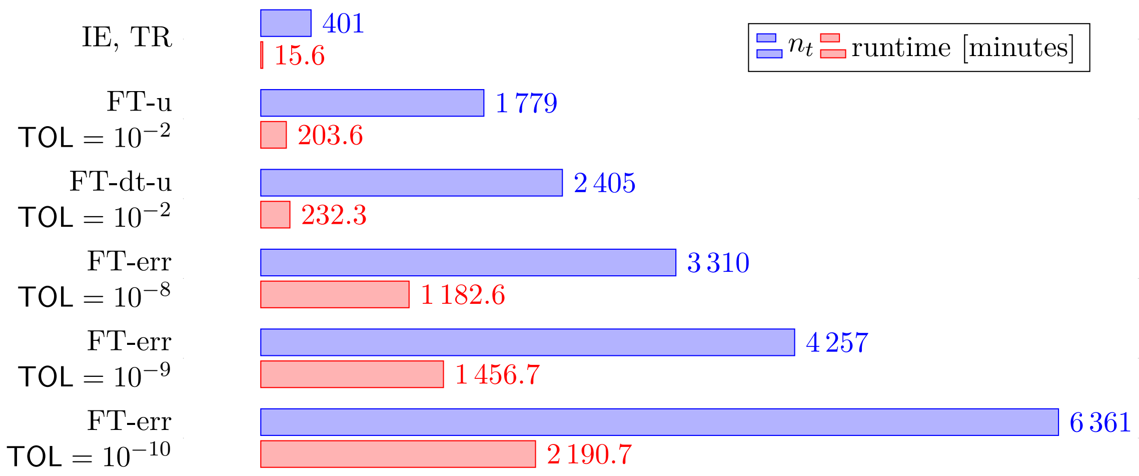

Figure 4.16.: Number of time steps n_t and runtimes of IE, TR, and FT with different time-adaptive strategies, $\lambda = 10^{-12}$.

the design of the LQR problem, i.e. the choice of the different problem parameters, is crucial for the performance of the feedback stabilization. In this section, several choices for the weights, inputs, and outputs are examined and compared. Further, the performance of the derived feedback stabilization is demonstrated for different scenarios of perturbations and various desired trajectories. The feedback gain matrices are computed with BDF 4 and the closed-loop simulations are performed with FT-dt-u in this entire section.

4.4.1. Choice of the Weight Parameter

As shown in Sections 4.2 and 4.3, the weight parameter λ in the cost functional (3.1) is a very important choice in the design of the LQR problem. It has a strong influence on the computational cost and the performance of the feedback stabilization. Decreasing λ results in the cost functional (3.1) being more dominated by the output deviation while the input cost term has less impact. Thus, a smaller λ leads to a more active feedback stabilization, which prevents the interface from deviating. However, note that λ acts as a regularization parameter to the stabilization problem. For the DRE, a smaller λ results in a more dominant quadratic term. Thus, solving the DRE becomes computationally more expensive for the BDF methods since the underlying ARE solver (Newton-ADI) requires more iterations to converge.

In Figure 4.17, the influence of the weight parameter on the behavior of the feedback stabilization is displayed for $\lambda \in \{10^{-3}, 10^{-4}, 10^{-6}, 10^{-12}\}$. Here, the top part shows the perturbation as well as the computed feedback stabilizations (relative to the cooling temperature Θ_{cool}). Further, the bottom part shows the resulting interface trajectories

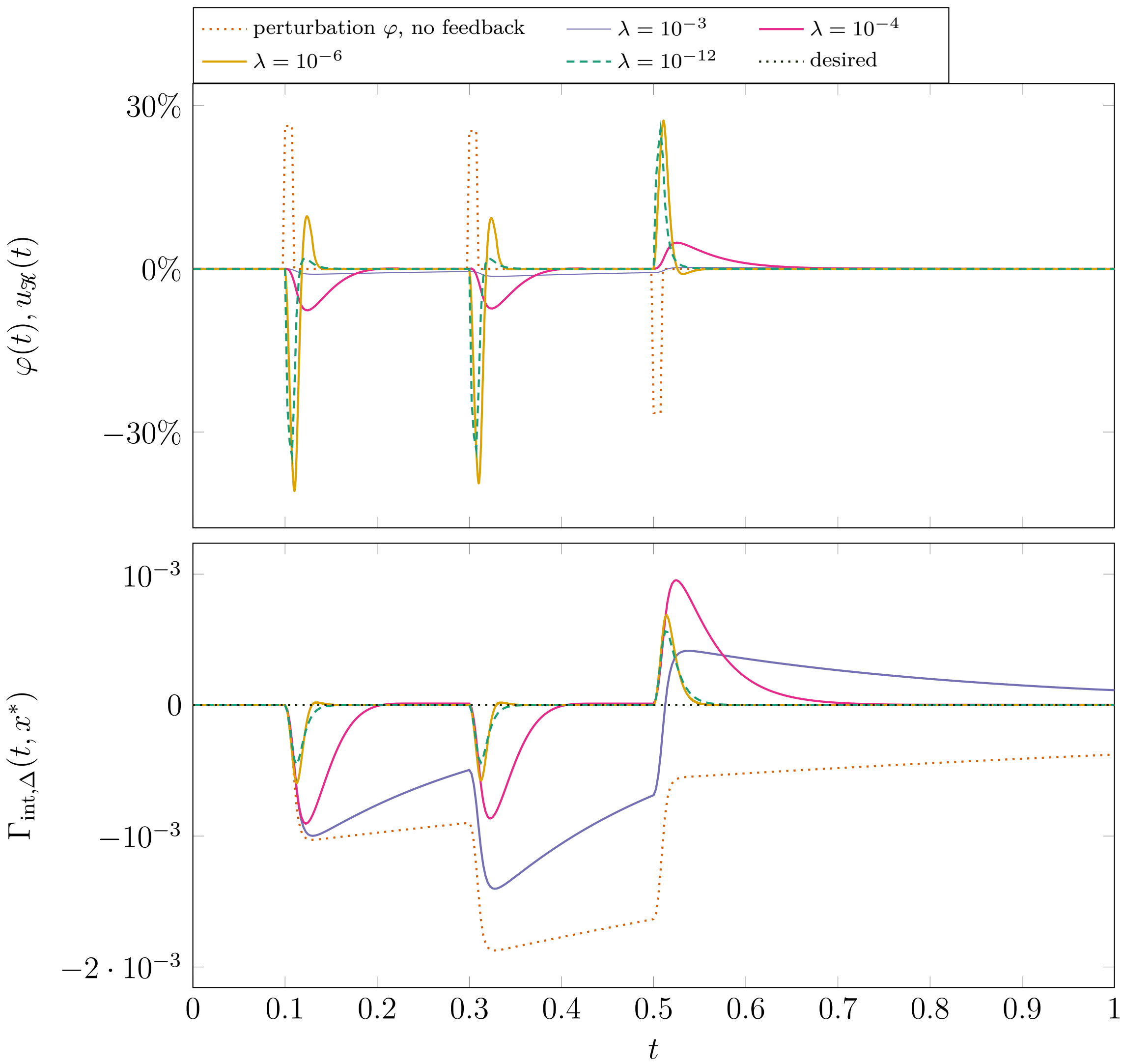

Figure 4.17.: Perturbations and feedback stabilizations (top), and perturbed and stabilized relative interface positions for different weights (bottom)

over time, again at the point on the interface with largest deviation. With a weight factor $\lambda = 10^{-3}$ and larger, the computed feedback stabilization shows very low activity. As a result, with $\lambda = 10^{-3}$, the feedback stabilization is not able to steer the interface back towards the reference trajectory, whereas the feedback stabilization is successful with $\lambda = 10^{-4}$. The interface is steered back to the desired trajectory before the next perturbation occurs. Moreover, as expected from the theory, with an even smaller weight, i.e. $\lambda = 10^{-6}$, the feedback stabilization is even more active and able to stabilize the interface even quicker. This does extend further for even smaller weights, as demonstrated with $\lambda = 10^{-12}$ in Figure 4.17.

At first glance, the input seems to be more active for $\lambda = 10^{-6}$ than for $\lambda = 10^{-12}$. However, it can be observed that for $\lambda = 10^{-12}$ the magnitude of the input increases slightly faster than for $\lambda = 10^{-6}$, if one pays attention to the moments when the interface is perturbed. This results in the interface deviating less while the perturbation is active and returning faster to the desired trajectory. For weight parameters smaller than $\lambda = 10^{-12}$, the BDF methods fail to converge in this example, probably due to the loss of the regularization effect of λ.

In conclusion, the weight parameter λ is a very important choice in the design of the LQR problem as it can be used to improve the performance of the feedback stabilization by steering the interface back to the desired trajectory quicker. In addition, the behavior of the feedback stabilization also strongly depends on the choice of inputs and outputs, as is demonstrated in the next sections.

4.4.2. Choice of the Controls

A crucial parameter for the design of the LQR problem for the feedback stabilization of the Stefan problem is the choice of the inputs that are applied to Equation (2.1). Since in a real world scenario it is more realistic to have inputs at the boundary than inside the domain or inputs acting on Equation (2.3), only boundary conditions modeled by Equation (2.1b) are used.

Figure 4.18 shows the domain $\Omega(t)$ and one example of a distribution of input boundary areas can be seen on the left side (orange lines). Here, the top boundary is split into four equidistant segments and there is one additional input boundary segment at each side wall slightly above the desired interface position. Within this context, other combinations of these are possible with only the segments at the top, only the segments at the walls, or with a different number of segments at the top boundary.

The same desired trajectory as in the previous section is used to compare the performance of feedback stabilization with different input choices. Two different perturbations $\varphi_1(t), \varphi_2(t) \in [-\Theta_{\text{cool}}, \Theta_{\text{cool}}]$ are generated, acting on the left part $(\Gamma_{\text{cool},1}(t))$ and right part $(\Gamma_{\text{cool},2}(t))$ of the Dirichlet boundary at the bottom of the domain $\Omega(t)$ (see Figure 4.18, left). For this, three random values are generated for each side, which are applied at the times $0.1, 0.4325$, and 0.765 (see Figure 4.19, top). In detail, two pertur-

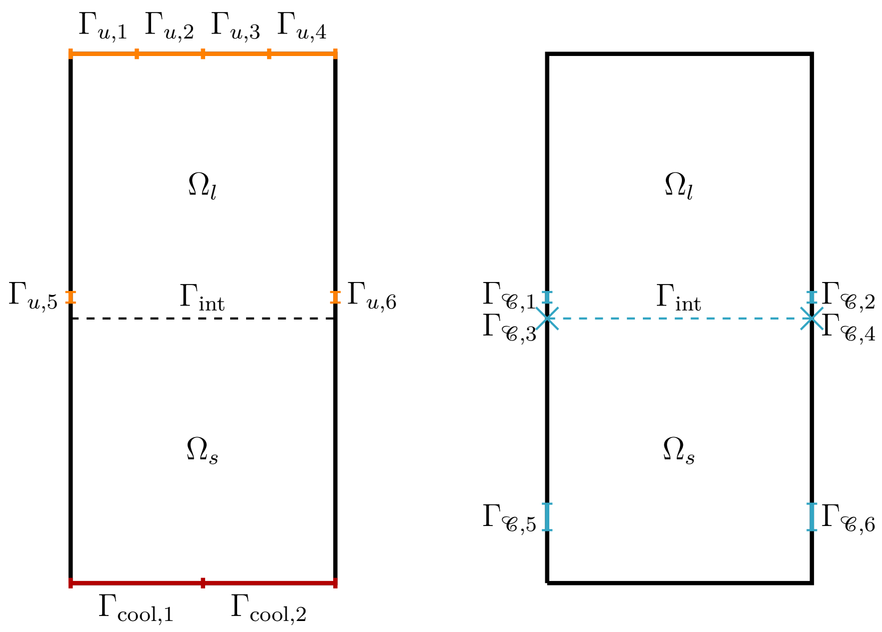

Figure 4.18.: Example of input, perturbation (left), and output areas (right)

bation functions are applied to

$$\begin{aligned} \Theta &= \Theta_{\text{cool}} + \varphi_1, && \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{cool},1}, \\ \Theta &= \Theta_{\text{cool}} + \varphi_2, && \text{on } (0, t_{\text{end}}] \times \Gamma_{\text{cool},2}. \end{aligned}$$

The effect of these perturbations is not only to move the interface away from the desired position, but also to add a certain amount of curvature to the deviated interface.

First, 1 to 10 inputs distributed equidistantly over the top boundary are compared to stabilize this interface position. Consequently, the input matrices are $\mathscr{B}_m(t) \in \mathbb{R}^{m\times n}$, for $m = 1, \ldots, 10$. The same two outputs are used at $\Gamma_{\mathscr{C},3}(t)$, $\Gamma_{\mathscr{C},4}(t)$ as in the previous experiments ($\mathscr{C}(t) \in \mathbb{R}^{2\times n}$). Additionally, the weight factor is set to $\lambda = 10^{-9}$ in the cost functional[1]. An example of the interface positions $\Gamma_{\text{int},\Delta}(t)$ relative to the desired interface trajectory is in Figure 4.19 at the time points $t \in [0.465, 0.56, 0.75, 1]$. Here, the unstabilized interface position and the feedback-stabilized interface positions with one and two inputs are displayed. This example shows that, as expected, one input has no effect on the curvature of the interface, while two inputs are sufficient. In detail, with the first two perturbations, the interface is pushed down and takes on a distinct curvature. Just after the second perturbation ($t = 0.465$), the feedback-stabilized interfaces are

[1]For different numbers of inputs m, the considered stabilization problems are not directly comparable even with the same λ. E.g., $\|\mathscr{B}(t)\|_2$ is different for different m. To compensate this at least partially, the input matrices are normalized $\mathscr{B}_m(t) = \frac{\|\mathscr{B}_2(t)\|_2}{\|\tilde{\mathscr{B}}_m(t)\|_2}\tilde{\mathscr{B}}_m(t)$ to all have the same norm as $\mathscr{B}_2(t)$.

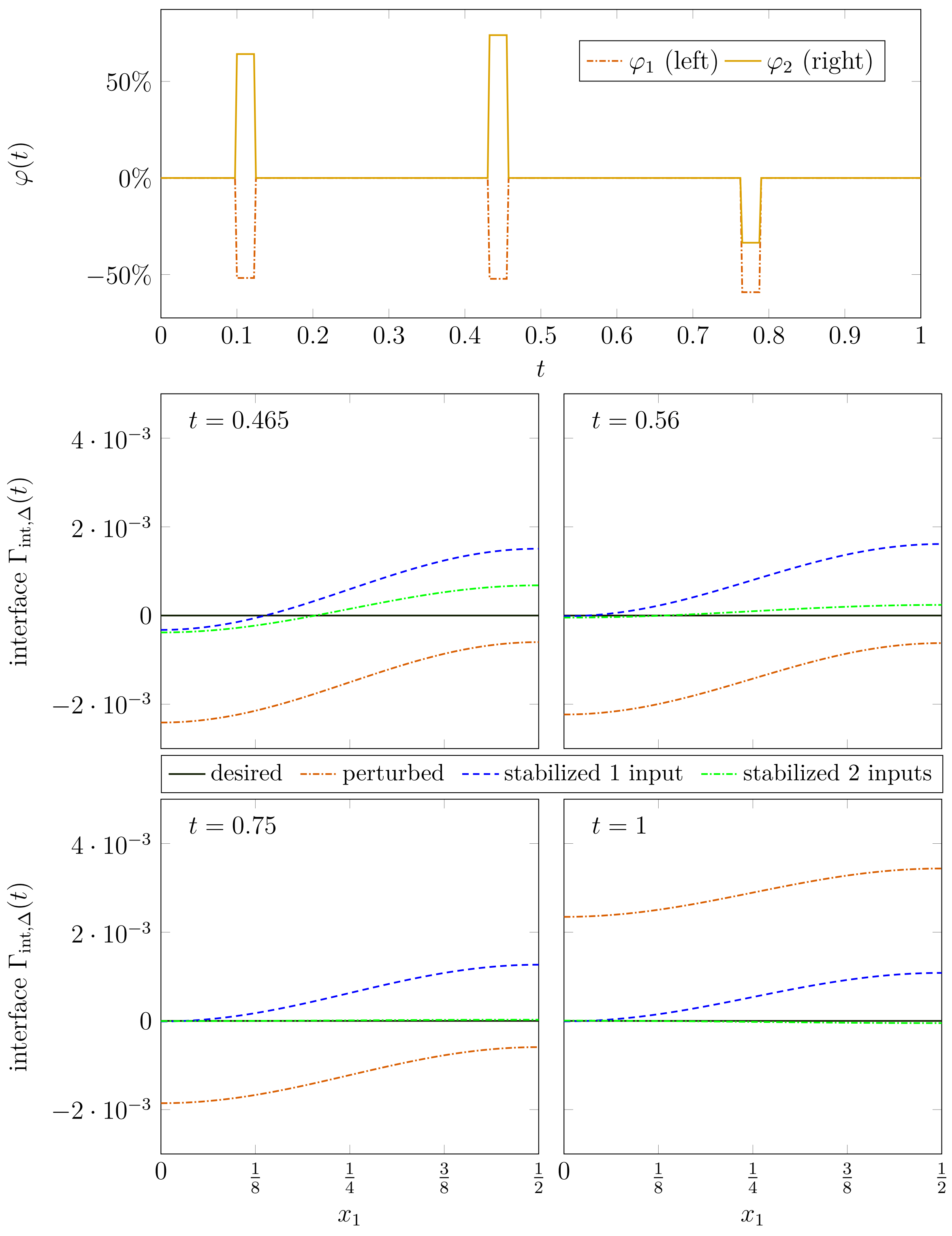

Figure 4.19.: Two different perturbations (top), and perturbed and stabilized relative interface positions (bottom)

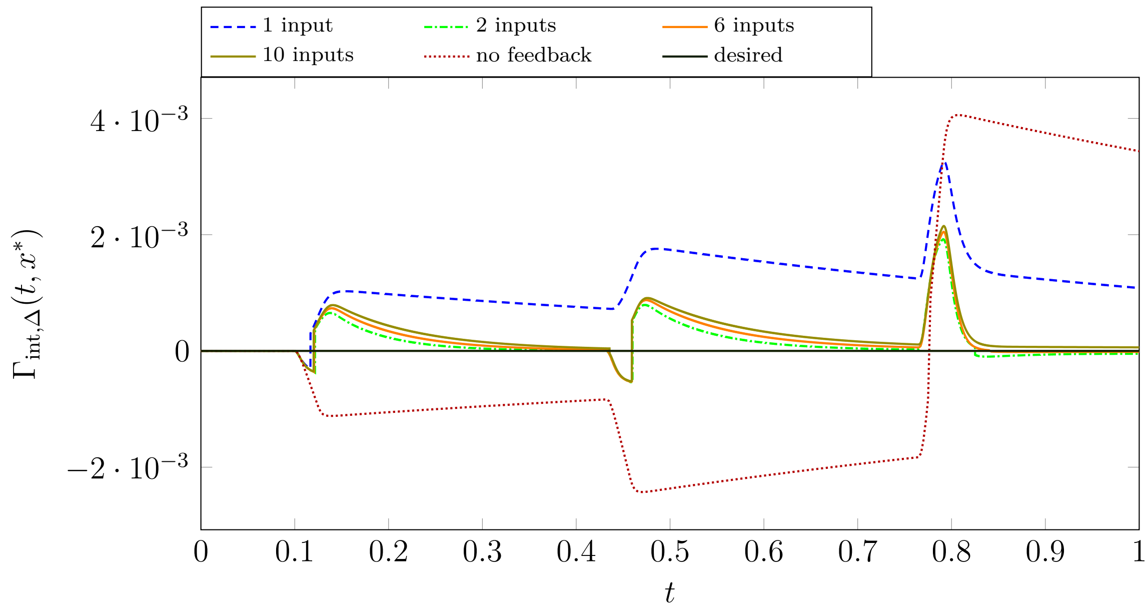

Figure 4.20.: Perturbed interface position relative to the reference trajectory with different number of inputs at the top of the domain.

basically back to the desired interface, but still have an undesired curvature. Already before the third perturbation, at $t = 0.56$, it is clearly visible that with two inputs the interface is almost flat again, like the desired interface, while the stabilization with only one input is not able to influence the curvature. A similar behavior is observed after the first perturbation. Right before the third perturbation ($t = 0.75$), the stabilization with two inputs steered the interface back to the desired interface position while the curvature did not change with one input. In contrast to the first two perturbations, the third perturbation moves the interface upwards above the desired position. At the end of the time interval ($t = 1$), the feedback stabilized interface with two inputs is again flat and back to the desired position. With one input and no feedback stabilization, the interface is curved or still has a certain distance to the desired position at the end of the time interval.

This experiment shows that the feedback stabilization with two inputs can move the perturbed interface back to the desired trajectory and additionally stabilize the curvature of the interface. With the chosen input and output setting, the position of the interface is corrected as fast as in the previous experiments. The curvature correction requires some more time but is still performed during the considered time interval. While one input is not sufficient to influence the curvature, the feedback stabilization behaves basically the same for $m = 2, \ldots, 10$. This is shown exemplarily for $m \in \{1, 2, 6, 10\}$ in Figure 4.20.The feedback stabilizations that are not shown have a similar behavior. Here, the interface trajectories are plotted over time, again at the point on the interface

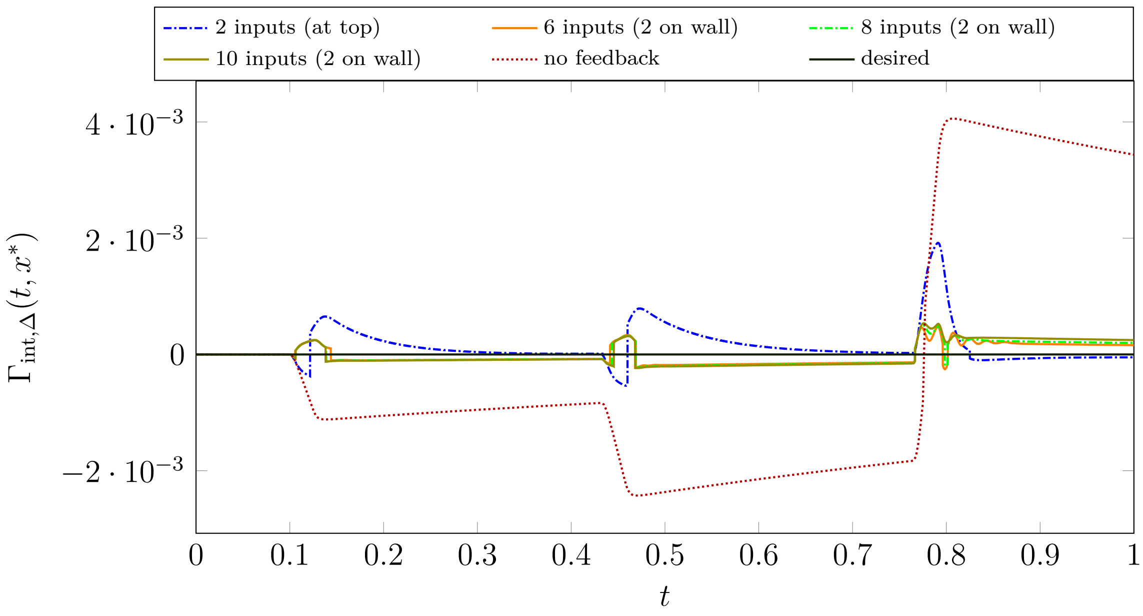

Figure 4.21.: Perturbed interface position relative to the reference trajectory with different number of inputs at the wall and the top of the domain.

with largest deviation. After the first two perturbations, it can be seen by the jumps in the curves that this point is changing. Again, while it is clear that one input is not sufficient, the interface trajectories for $m = 2, \ldots, 10$ inputs are not significantly different.

In contrast to that, the feedback stabilization can stabilize the interface position significantly faster if two additional inputs are added at the side walls in the liquid phase of the domain (see Figure 4.18, $\Gamma_{u,5}$ and $\Gamma_{u,6}$). These two additional inputs are closer to the interface and therefore have a greater effect on its position. This is shown in Figure 4.21 where the interface stabilization with two inputs at the top of the domain from before is compared to the stabilizations with six, eight, and ten inputs including the two inputs at the walls. Again the interface trajectories are very similar for $m = 6, \ldots, 10$, thus additional inputs at the top do have have a significant influence. Interestingly, for $m \leq 5$, the feedback stabilization does not succeed to steer the interface back to the desired trajectory.

In summary, the number of inputs that are well suited for the stabilization task depends on the LQR problem. In the example from Section 4.4.1 with just one perturbation at the bottom of the domain, a single input was sufficient. However, with a perturbation that introduces curvature to the interface, additional inputs are necessary to stabilize the interface position.

In addition to that, a very important parameter for the feedback stabilization problem is the choice of the outputs, which is discussed in the next section.

4.4.3. Choice of the Outputs

The outputs of the stabilization problem are an essential parameter as they determine the measured quantities in the cost functional (3.1). A useful example to demonstrate the importance of this choice is to consider only a single output at $\Gamma_{\text{int}}(t)$ (see Figure 4.18). It monitors the difference between the interface movement and the desired movement, i.e.

$$\langle \mathscr{C}_{\text{int}}(t), \mathsf{v}^h(t)\rangle = \int\limits_{\Gamma_{\text{int}}(t)} \left(\frac{1}{\ell}[k_s(\nabla\Theta_\Delta(t,s))_s - k_l(\nabla\Theta_\Delta(t,s))_l]\right)\cdot \boldsymbol{n}_{\text{int}}(t)\cdot \mathsf{v}^h(t)\mathrm{d}s$$

and is defined via the jump term of the Stefan condition (Equation (2.3b)). Only while the perturbation is actively driving the interface away from the desired trajectory would it generate a significant output, and thus an active feedback response. However, it cannot detect a difference in the position of the interface. Consequently, it would not steer the interface back but would keep it on a “parallel trajectory“. However, the output modeled by $\mathscr{C}_{\text{int}}(t)$ can be used in combination with additional measurements.

In contrast to just one output, $\Gamma_{\mathscr{C},3}(t)$ and $\Gamma_{\mathscr{C},4}(t)$ (see Figure 4.18) represent point measurements of the temperature at the location of the desired interface on the walls. These point measurements do not only indicate if the interface is deviated from the desired trajectory but also measure if the interface is above or below. Specifically, they measure whether the temperature at these points on the walls is above or below the melt temperature. This indicates the direction in which the interface is deviating. Additional outputs, which provide even more information, are $\Gamma_{\mathscr{C},1}(t)$, $\Gamma_{\mathscr{C},2}(t)$, $\Gamma_{\mathscr{C},5}(t)$, and $\Gamma_{\mathscr{C},6}(t)$. They measure averaged temperatures on the corresponding interval on the side walls.

The influence on the LQR problem of these different outputs is compared for a single perturbation and, consequently, a single input $u_{\mathscr{K}}(t)$. This input has the same value on $\Gamma_{u,1}(t) \cup \ldots \cup \Gamma_{u,4}(t)$, i.e. $\mathscr{B}(t) \in \mathbb{R}^{n\times 1}$. The top part of Figure 4.22 shows the perturbation and feedback stabilizations for two different LQR designs resulting from two different combinations of weights and outputs:

$$u_1: \qquad \lambda = 10^{-12}, \qquad \text{2 outputs: } \Gamma_{\mathscr{C},3}, \Gamma_{\mathscr{C},4},$$
$$u_2: \qquad \lambda = 1.6\cdot 10^{-2}, \qquad \text{7 outputs: } \Gamma_{\mathscr{C},1}, \ldots, \Gamma_{\mathscr{C},6}, \Gamma_{\text{int}}.$$

More detailed, in the first LQR design, the stabilization $u_1(t)$ is based on two outputs that measure the temperature at the desired interface position on the boundary, such that $\mathscr{C}(t) \in \mathbb{R}^{2\times n}$, while $u_2(t)$, in the second setting, uses seven outputs, i.e. $\mathscr{C}(t) \in \mathbb{R}^{7\times n}$. The resulting interface positions are displayed in the bottom part of Figure 4.22. As described in the previous section, the interface positions are relative to the desired interface position at the point $x^*(t)$ with the largest deviation on the interface.

The feedback stabilization $u_1(t)$ is most active shortly after the perturbation starts and it steers the interface back to the desired position in much reduced time compared

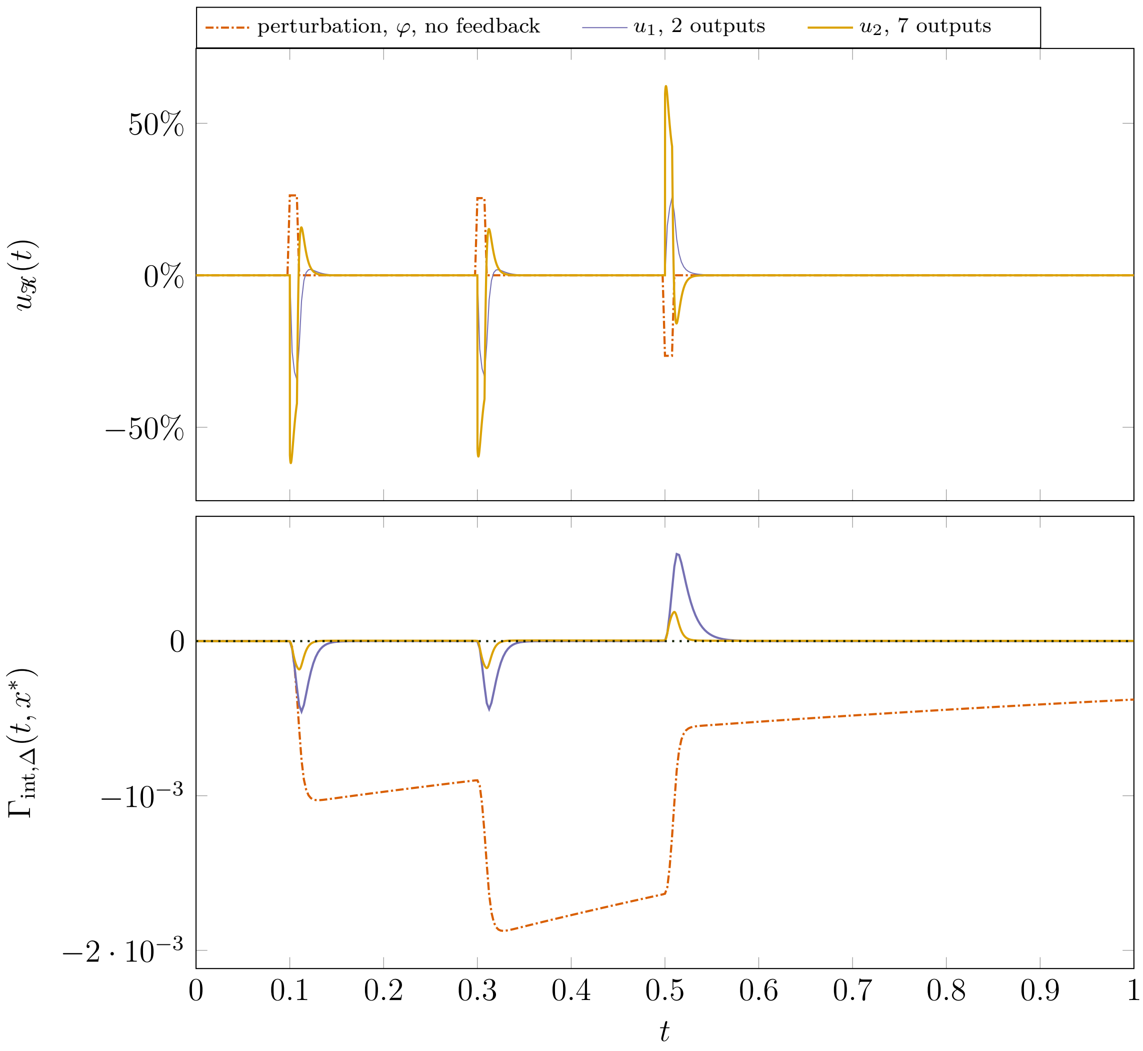

Figure 4.22.: Perturbation and feedback (top), relative interface position (bottom) for different number of outputs

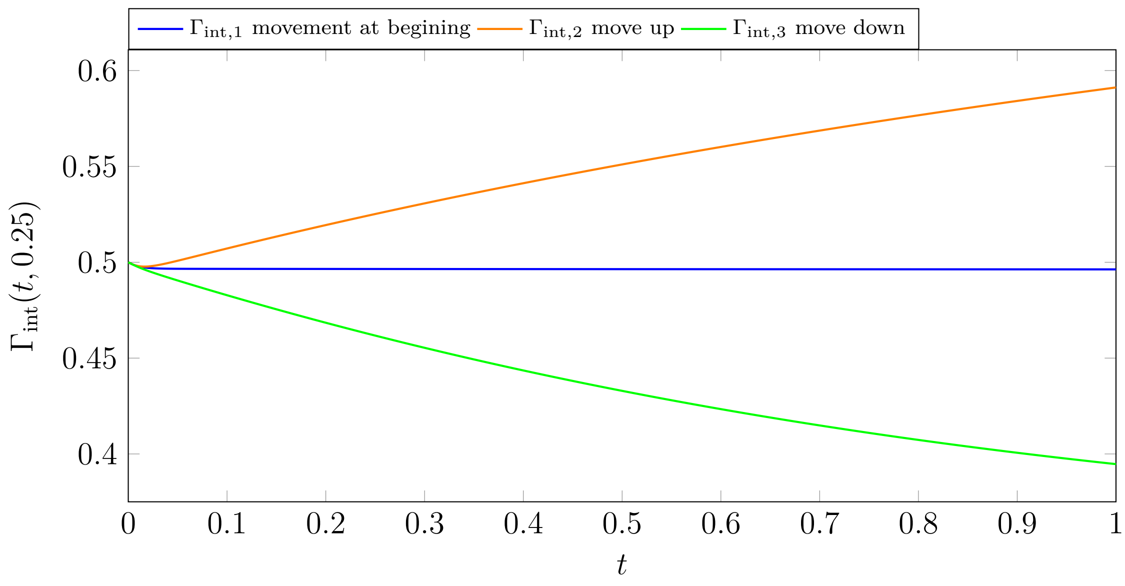

Figure 4.23.: Different desired interface trajectories

to no feedback stabilization, as expected from theory. Unlike a single output at $\Gamma_{\text{int}}(t)$, the two point measurements enable the stabilization to steer the interface back to the desired trajectory.

However, with additional outputs the stabilization can be accelerated even further. The weight for the LQR setting of $u_1(t)$ is not directly comparable to the setting for $u_2(t)$, since this is based on different outputs. The outputs at $\Gamma_{\mathscr{C},5}(t)$ and $\Gamma_{\mathscr{C},6}(t)$ allow to detect the temperature perturbation earlier and the output that monitors the movement of $\Gamma_{\text{int}}(t)$ can observe the deviation of the interface earlier. Thus, $u_2(t)$ is most active immediately after the perturbation starts and moves the interface back much faster.

4.4.4. Different Desired Interface Trajectories

A major contribution of this thesis is to handle a moving interface and thus non-autonomous data by using a moving mesh in the spatial discretization. In this section, the feedback stabilization is tested for three different desired interface trajectories $\Gamma_{\text{int},1}(t)$, $\Gamma_{\text{int},2}(t)$, and $\Gamma_{\text{int},3}(t)$ that are displayed in Figure 4.23. The desired interface trajectory $\Gamma_{\text{int},1}(t)$, which is used in the examples from the previous sections, is a flat horizontal line. At the beginning of the time interval, it moves down slightly from its initial height of 0.5 and then stays there. To clearly demonstrate that the proposed methods are capable to handle significantly stronger time-dependence in the data, two additional desired interface trajectories are tested in this section. The trajectory $\Gamma_{\text{int},2}(t)$ is a flat horizontal line moving from its initial height at 0.5 upward by 0.1. Contrary to this, $\Gamma_{\text{int},3}(t)$ moves downward by 0.1, starting from the same initial position.

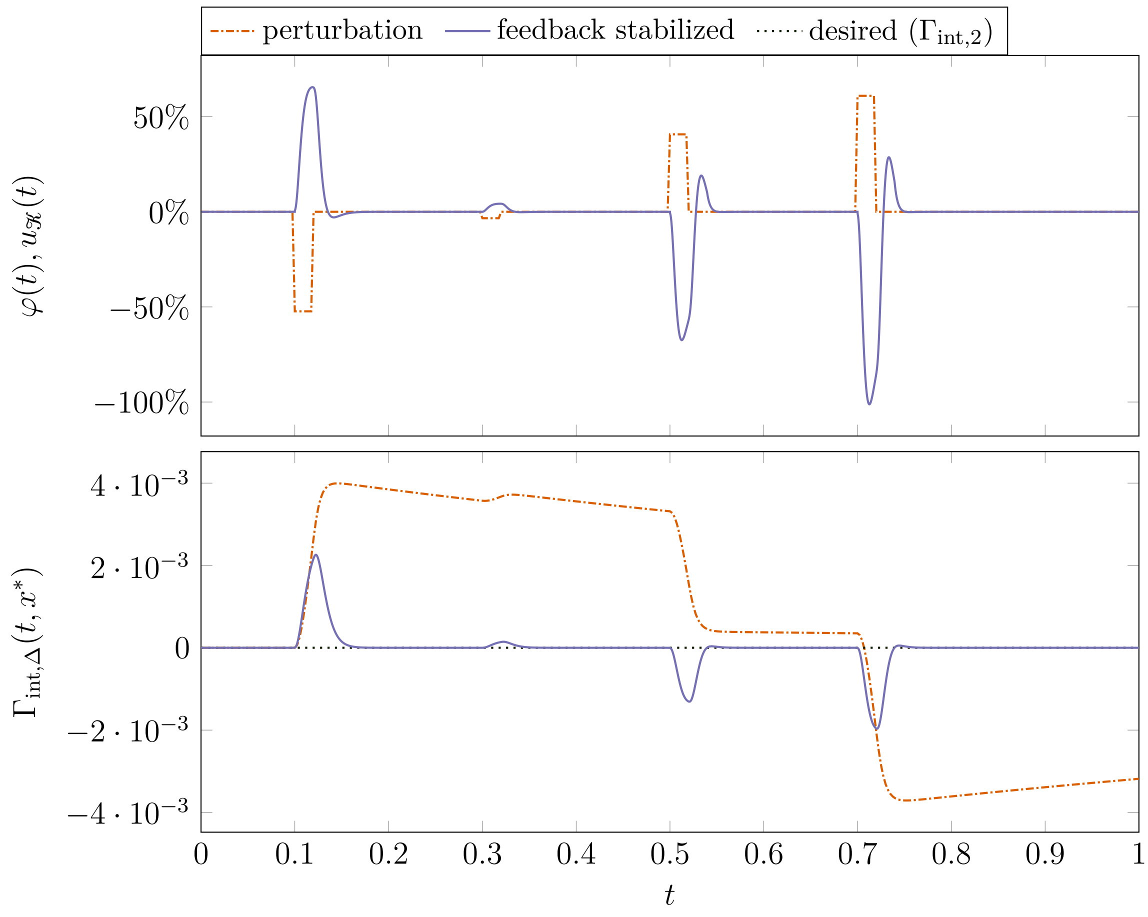

Figure 4.24.: Perturbation and feedback (top) and relative interface position (bottom) for an interface moving upwards

For the investigated LQR design, one input at the top of the domain, two outputs at $\Gamma_{\mathscr{C},3}(t)$ and $\Gamma_{\mathscr{C},4}(t)$, as well as the weight $\lambda = 10^{-6}$ are chosen. This time, four randomly generated perturbations are applied at the times $0.1, 0.3, 0.5$, and 0.7.

These perturbations can be seen in the top part of Figure 4.24 together with the feedback stabilization that is applied to the upward moving desired interface trajectory $\Gamma_{\text{int},2}(t)$. Despite the stronger time-dependence of the data, the BDF 4 method and FT-dt-u successfully work together to compute a feedback gain, apply it in the non-linear-closed loop simulation, resulting in a feedback stabilization that behaves similarly to the previous experiment. It stops the interface from deviating and drives it back to the desired position, as expected from the theory. This is displayed in the bottom part of Figure 4.24, which shows the interface position $\Gamma_{\text{int},\Delta}(t, x^*)$ relative to the desired interface position.

Similar to this, Figure 4.25 displays another set of perturbations and a feedback sta-

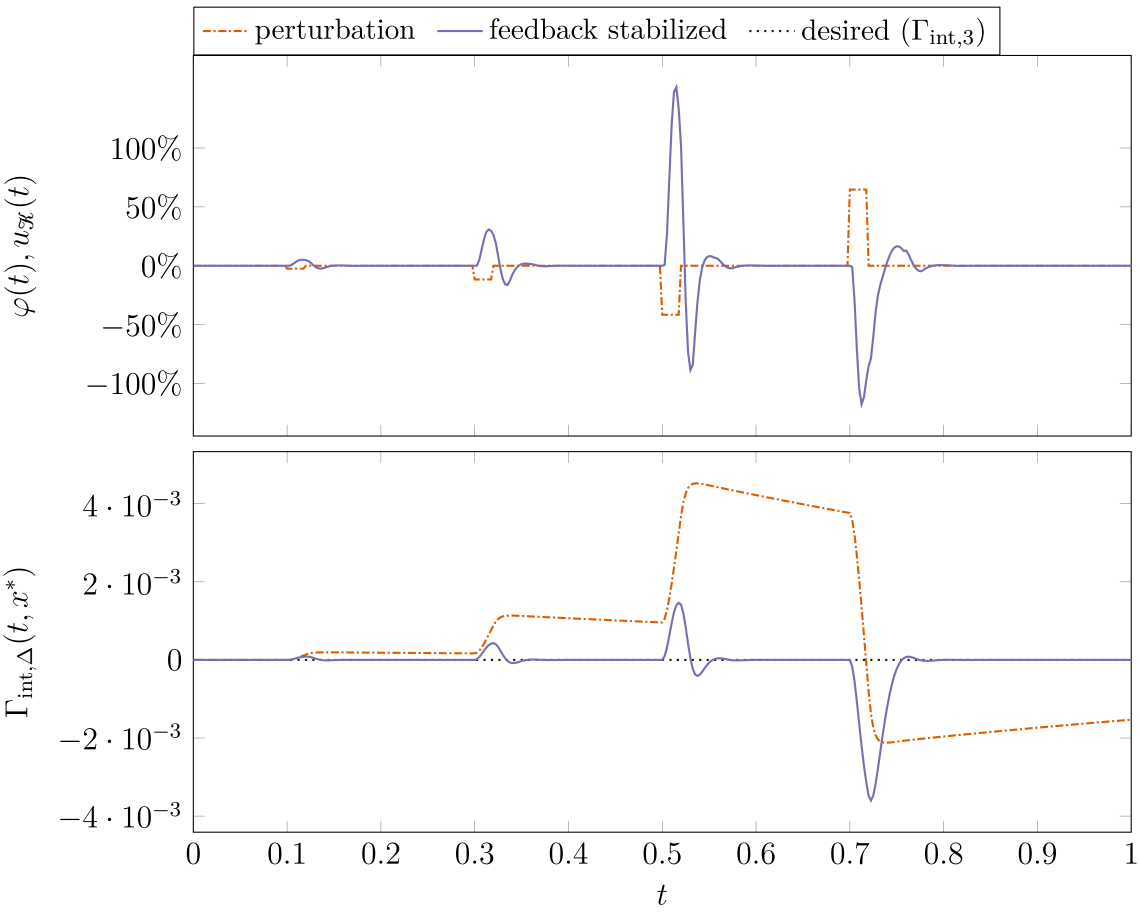

Figure 4.25.: Perturbation and feedback (top) and relative interface position (bottom) for an interface moving downwards

bilization for the downward moving desired interface trajectory $\Gamma_{\text{int},3}(t)$. Again, the feedback stabilization shows the expected behavior and successfully stabilizes the interface trajectory after each deviation, which is caused by the perturbations.

The experiments demonstrate the performance of the proposed method with strongly time-varying coefficients in the DRE. This demonstrates that BDF 4 and FT-dt-u can also cope with the stronger time dependence without difficulties. Note that the feedback stabilizations resulting from BDF 1 to 3 behave the same.

The numerical experiments in this chapter show that applying the LQR approach to the two-dimensional two-phase Stefan problem is a viable method to derive feedback stabilizations. The numerical experiments also confirm that the methods proposed in this thesis are a reliable and computationally efficient choice to compute and apply feedback stabilizations for the Stefan problem. Furthermore, the effects of the most important problem parameters on the LQR design are demonstrated in detail.

CHAPTER 5

CONCLUSION & OUTLOOK

Contents

5.1. Conclusion

This thesis investigates both conceptually and numerically the derivation and computation of a feedback stabilization for the two-dimensional two-phase Stefan problem. In addition, this thesis applies the feedback stabilization in a closed-loop simulation. Therefore, it fills a significant gap in the existing literature between closed-loop stabilizations for the one-dimensional case, e.g. [EWFR$^+$22], or the one-phase Stefan problem, e.g. [KDK19], and the open-loop control with two spatial dimensions and two phases, e.g. [BBHS18]. The simplifications of the model of the Stefan problem for the existing approaches have been made in order to make the approach numerically feasible or theoretically approachable. In detail, this reflects the challenges that are posed by this non-linear, discontinuous, differential-algebraic phase-change problem involving a moving interface.

A major contribution of the work consists in the adaptation, extension, and combination of well-established approaches and methods for related problems. In this way, a sharp interface representation with mesh movement and a FEM discretization, see e.g. [BPS13], is combined with the LQR approach, which has shown promising performance for related problems, see e.g. [Wei16].

Despite some specific details in the spatial discretization, a linearization of the non-linear Stefan problem is presented, which is essential for the LQR approach. By applying LQR to the Stefan problem, additional challenges arise, like time-dependent matrices in the resulting DRE, which go beyond e.g. [Lan17]. For this non-autonomous large-scale DRE, only the theoretical concept of the numerical solvers was available in the literature. As a result, this work provides a non-autonomous implementation of the BDF schemes that allows the numerical solution of these non-autonomous large-scale DREs. The presented numerical comparisons demonstrate that these non-autonomous BDF methods are computationally efficient and competitive, despite the fact that in the autonomous case other methods like splitting schemes [Sti18a] or Krylov subspace methods [BBH21] outperform BDF methods.

Another contribution of this thesis is an adaptive time-stepping strategy for the closed-loop simulation of the Stefan problem. This strategy resolves numerical issues that arise in the application of the computed feedback stabilization and that have not been addressed in the existing literature. It is a novel adaptation of similar strategies [FW18] specifically to the closed-loop simulation of the Stefan problem. This allows the feedback stabilization to be applied reliably with reasonable computational effort during closed-loop simulations of the Stefan problem.

The main contribution of this work is a core algorithm for the computation and application of a closed-loop stabilization to steer the interface position to a desired trajectory. Detailed numerical tests are performed to demonstrate the usability of the proposed methods for this important stabilization problem. The contributions of this work are important for a number of reasons. One is that in many solidification and crystallization processes, stabilizing the interface position is critical to material quality. On the other hand, closed-loop stabilization is key to making real-world applications more time and energy efficient. By focusing on the Stefan problem with two spatial dimensions and two phases, this work can be used as a cornerstone on the path to the application to real-world applications. In addition, all of the numerical implementations, data, and results are made available for easy use in future research.

5.2. Outlook

The focus of this thesis is on demonstrating the conceptual and numerical applicability of the proposed methods. For this purpose, several assumptions are made to ensure the existence and uniqueness of the computed solutions. However, a theoretical investigation and analytical proof of these assumptions is beyond the scope of this work and is a matter for future research.

Furthermore, the performance of the presented methods, especially of the feedback stabilization, depends strongly on the problem and the choice of the problem parameters. For instance, parameters such as the number of outputs or the weight in the cost functional are chosen manually and require extensive testing to obtain satisfactory results. Research directions that consider the optimal placement of the sensors that collect the measurement information are an interesting question that is likely to significantly improve the performance of the stabilization and are therefore worth considering.

Although the proposed adaptive time-stepping was able to solve all the test scenarios considered, certain drawbacks occurred. In particular, several tolerance values for the time-stepping are chosen manually. Moreover, the proposed methods have a significantly higher computational cost compared to standard methods such as Implicit Euler, which, in turn, are not able to reliably simulate the closed-loop system. A more thorough theoretical understanding of the numerical issues, which require the more sophisticated time-stepping, could significantly improve the performance and reliability.

The numerical implementation uses existing software packages such as M-M.E.S.S. and FEniCS wherever appropriate. This allows the implementation to benefit from easy-to-use user interfaces and built-in performance optimizations. Although the code for the implementation is available, some parts of it are not included in existing software packages. Thus, there is potential for numerical performance optimization and usability improvements. The additional implementation effort is significant and may be tackled in future work.

A promising direction for future research is to extend the class of problems considered. One way to do this is to allow other types of moving boundaries, such as a free outer boundary that changes the shape of the domain. Another important milestone on the way to a real-world application of the proposed methods is the incorporation and validation of the results from the numerical simulations by applying them to an actual physical experiment. A strong interdisciplinary collaboration is required for all developments in this direction, working at the intersection of physics and applied mathematics.

Furthermore, convection in the liquid phase can be included by coupling the Stefan problem with the Stokes or Navier-Stokes equations. To lay this out exemplary one can follow [BBHS18]. Here, the velocity $v(t)$ and the pressure $p(t)$ in the liquid phase are described by the incompressible Navier–Stokes equations for Newtonian fluids [Glo91]:

$$
\begin{aligned}
\dot{v} + ((v - \Upsilon) \cdot \nabla)v - \eta\Delta v + \nabla p &= 0, && \text{on } (0, t_{\text{end}}] \times \Omega_l, && (5.1a)\\
\nabla \cdot v &= 0, && \text{on } (0, t_{\text{end}}] \times \Omega_l, && (5.1b)\\
p \cdot \boldsymbol{n} - \eta\partial_{\boldsymbol{n}} v &= u \cdot \boldsymbol{n}, && \text{on } (0, t_{\text{end}}] \times \Gamma_u, && (5.1c)\\
p \cdot \boldsymbol{n} - \eta\partial_{\boldsymbol{n}} v &= 0, && \text{on } (0, t_{\text{end}}] \times (\Gamma_{\text{out}} \cap \partial\Omega_l). && (5.1d)
\end{aligned}
$$

The constant η is the kinematic viscosity. In addition to the momentum and mass balance Equations (5.1a) and (5.1b), Equation (5.1c) can define an inflow boundary condition on $\Gamma_u(t)$ which might be used instead of or in combination with Equation (2.1b) for the input. Other input boundary conditions are also possible, such as a Dirichlet condition at the top of the domain representing a driven cavity. Further, the mesh movement $\Upsilon(t)$ is coupled to $v(t)$ in Equation (5.1a). To couple the temperature with the Navier–Stokes equations, Equation (2.1a) is reformulated to:

$$
\dot{\Theta} + (v - \Upsilon) \cdot \nabla\Theta - \alpha\Delta\Theta = 0, \qquad \text{on } (0, t_{\text{end}}] \times \Omega. \tag{5.2}
$$

Besides the additional non-linearities that are introduced in Equations (5.1a) and (5.2), another algebraic condition is added to the system by Equation (5.1b). While the algebraic Stefan condition Equation (2.3b) can be treated with index reduction techniques as described in Equation (3.13), these techniques do not transform the system to an ODE here. A possible approach for the Navier–Stokes equations is to use a projection technique for index-2 DAE systems [GSW13]. However, this projection technique is not suitable for the algebraic structure arising from the Stefan condition (see Equation (3.12)). This can be seen by looking at the block structure resulting from the linearization and discretization in space of the Stefan problem coupled with the Navier–Stokes equations:

$$\begin{bmatrix} M_\Theta & 0 & 0 & 0 \\ 0 & M_v & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \Theta^h \\ v^h \\ \Upsilon^h \\ \mathfrak{p}^h \end{bmatrix} = \begin{bmatrix} A_{\Theta\Theta} & A_{\Theta v} & A_{\Theta\Upsilon} & A_{\Theta\mathfrak{p}} \\ A_{v\Theta} & A_{vv} & A_{v\Upsilon} & A_{v\mathfrak{p}} \\ A_{\Upsilon\Theta} & A_{\Upsilon v} & A_{\Upsilon\Upsilon} & A_{\Upsilon\mathfrak{p}} \\ A_{\mathfrak{p}\Theta} & A_{\mathfrak{p}v} & A_{\mathfrak{p}\Upsilon} & 0 \end{bmatrix} \begin{bmatrix} \Theta^h \\ v^h \\ \Upsilon^h \\ \mathfrak{p}^h \end{bmatrix} + \begin{bmatrix} B_\Theta \\ B_v \\ 0 \\ 0 \end{bmatrix} \mathbf{u}^h,$$

$$\mathbf{y}^h = \begin{bmatrix} C_\Theta & C_v & 0 & 0 \end{bmatrix} \begin{bmatrix} \Theta^h \\ v^h \\ \Upsilon^h \\ \mathfrak{p}^h \end{bmatrix}.$$

Note that all these matrices are time-dependent. The index reduction technique from Equation (3.13) applied here leads to the matrices

$$A = \begin{bmatrix} A_{\Theta\Theta} & A_{\Theta v} \\ A_{v\Theta} & A_{vv} \end{bmatrix}, \qquad J = \begin{bmatrix} A_{\Theta\mathfrak{p}} \\ A_{v\mathfrak{p}} \end{bmatrix} - \begin{bmatrix} A_{\Theta\Upsilon} \\ A_{v\Upsilon} \end{bmatrix} A_{\Upsilon\Upsilon}^{-1} A_{\Upsilon\mathfrak{p}},$$

$$M = \begin{bmatrix} M_\Theta & 0 \\ 0 & M_v \end{bmatrix}, \qquad G = \begin{bmatrix} A_{\mathfrak{p}\Theta} & A_{\mathfrak{p}v} \end{bmatrix} - A_{\mathfrak{p}\Upsilon} A_{\Upsilon\Upsilon}^{-1} \begin{bmatrix} A_{\Upsilon\Theta} & A_{\Upsilon v} \end{bmatrix}.$$

Necessary assumptions for these transformations and the projection techniques from [GSW13] to be applicable are that $A_{\Upsilon\Upsilon}$ and $GM^{-1}J$ are non-singular, $A_{\mathfrak{p}\Upsilon} A_{\Upsilon\Upsilon}^{-1} A_{\Upsilon\mathfrak{p}} = 0$, and the matrices J and G have full rank. These assumptions are usually fulfilled for the Stefan problem, because $A_{\Upsilon\Upsilon}$ is always non-singular, since it represents the Poisson operator with a Dirichlet boundary part (see Equation (2.7)). Furthermore, the pressure $\mathfrak{p}$ is not directly coupled to the mesh movement Υ, which means that $A_{\mathfrak{p}\Upsilon}, A_{\Upsilon\mathfrak{p}} = 0$. It follows that the rank of J and G is not changed by the transformation, and since M is the mass matrix for temperature and velocity, it is non-singular by construction. As a result, the above assumptions are fulfilled for the case of the Stefan problem. The resulting block structure of the system reads:

$$
\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \Theta^h \\ v^h \\ p^h \end{bmatrix} = \begin{bmatrix} A & J \\ G & 0 \end{bmatrix} \begin{bmatrix} \Theta^h \\ v^h \\ p^h \end{bmatrix} + \begin{bmatrix} B_\Theta \\ B_v \\ 0 \end{bmatrix} \mathbf{u}^h,
$$

$$
\mathbf{y}^h = \begin{bmatrix} C_\Theta \; C_v & 0 \end{bmatrix} \begin{bmatrix} \Theta^h \\ v^h \\ p^h \end{bmatrix}. \tag{5.3}
$$

For this system, the projection technique for index-2 DAE systems from [GSW13] can be applied to bring it into the state-space formulation for Equation (3.2). To apply the methods from Chapter 3, the problem parameters such as inputs, outputs, and the weight factor in the cost functional must be chosen in a manner appropriate for this setting.

APPENDIX A

APPENDIX

Estimated Order of Convergence A BDF method with $\wp \leq 6$ stages is expected to converge with order $\wp$ as well. This can be validated numerically by computing the estimated order of convergence. For this, the relative error pointwise in time is compared for two different time discretizations:

$$\mathrm{EOC}(n_\wp) = \frac{\log\left(\frac{e_{\mathrm{DRE}}(t_0,\tau_{k-1},n_\wp)}{e_{\mathrm{DRE}}(t_0,\tau_k,n_\wp)}\right)}{\log\left(\frac{\tau_{k-1}}{\tau_k}\right)} \tag{A.1}$$

Here, τ_{k-1} and τ_k are the two smallest time step sizes used in the experiments (see Figure 4.2).

Fixed Point Property of BDF Methods Let's assume that the solution to the general matrix equation (2.39) is constant:

$$\mathbf{X}(t) = C.$$

Then, the general matrix equation (2.39) simplifies to

$$f(t,\mathbf{X}) = \dot{\mathbf{X}}(t) = 0.$$

Additionally, the coefficients of BDF 1-4 sum up to $\sum_{j=1}^{\wp} \alpha_j = -1$ (see Table 2.1). Thus, the error for BDF 1-4 (Equation (2.40)) is zero:

$$C = \mathbf{X}_k = -\sum_{j=1}^{\wp} \alpha_j \underbrace{\mathbf{X}_{k-j}}_{=\,C} + \tau_k \beta \underbrace{f(t_k,\mathbf{X}_k)}_{=\,0} = -\sum_{j=1}^{\wp} \alpha_j C = C. \tag{A.2}$$

Consequently, the matrix-valued BDF 1-4 preserve fixed points.

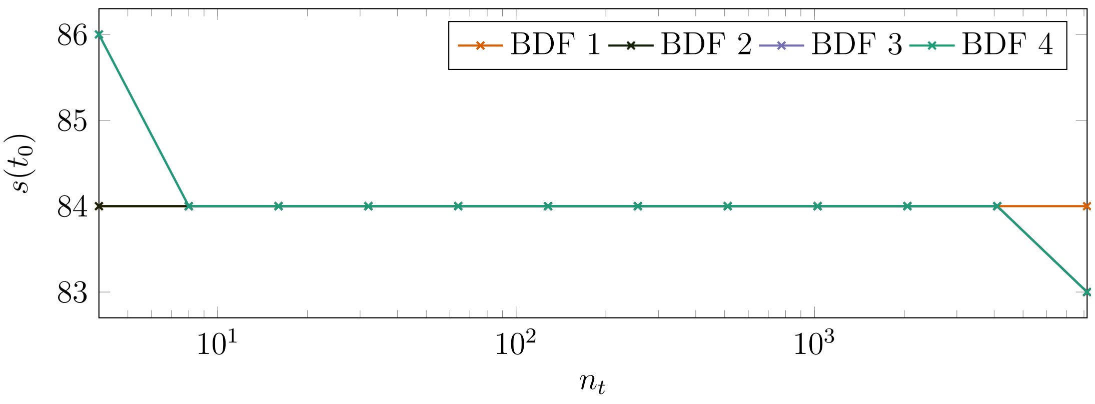

Figure A.1.: $s(t_0)$ (numerical rank of the solution) for BDF for different n_t (no. of time steps), Partially Non-autonomous Steel Profile.

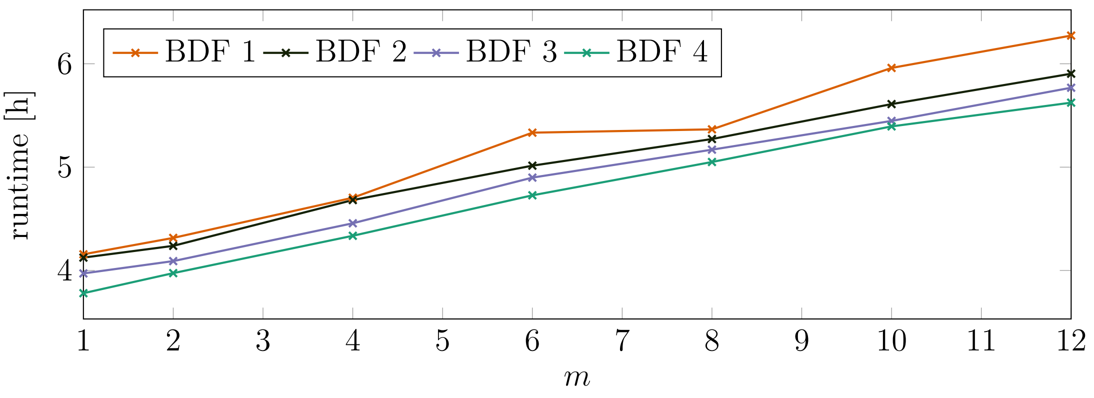

Figure A.2.: Runtime of BDF for different m (columns in $\mathscr{B}(t)$), Two-dimensional Two-phase Stefan Problem.

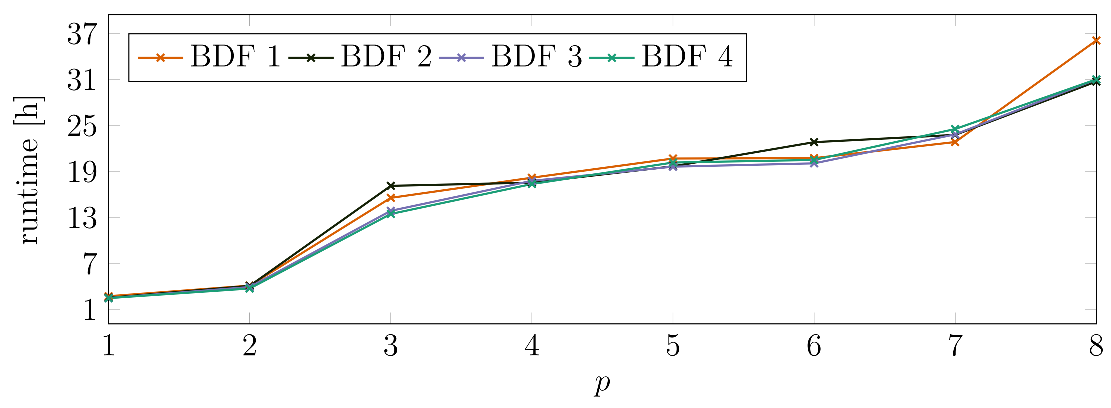

Figure A.3.: Runtime of BDF for different p (rows in $\mathscr{C}(t)$), Two-dimensional Two-phase Stefan Problem.

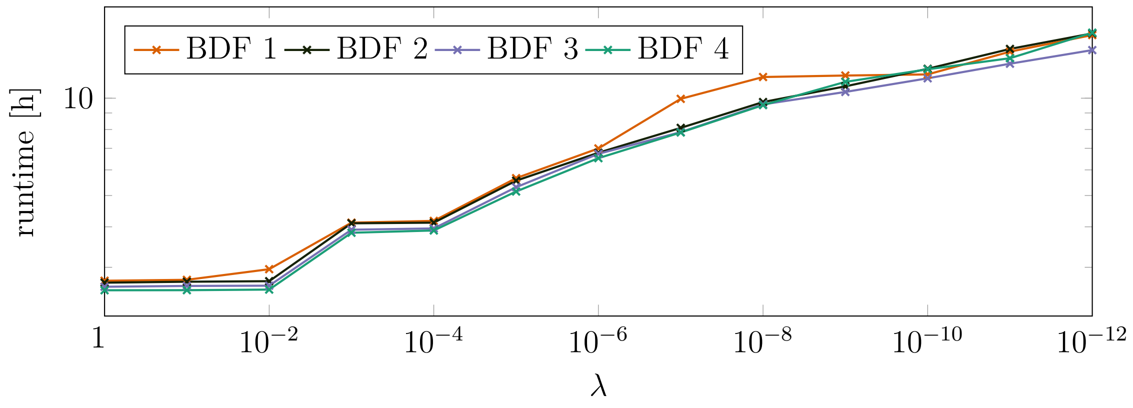

Figure A.4.: Runtime of BDF for different λ (weight in cost functional), Two-dimensional Two-phase Stefan Problem.

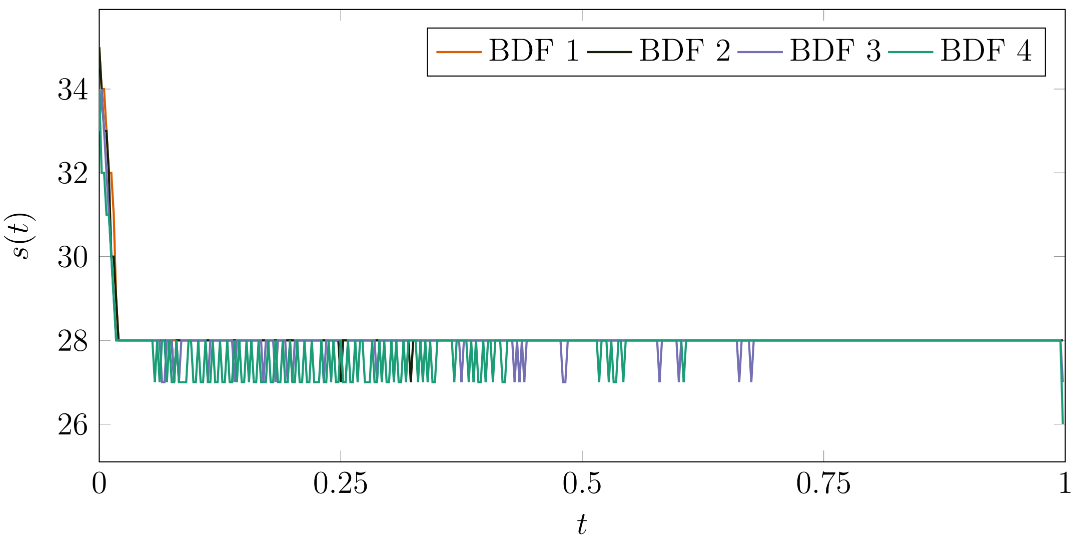

Figure A.5.: Numerical rank of the solution $s(t)$ for BDF 1 to 4, Two-dimensional Two-phase Stefan Problem.

BIBLIOGRAPHY

[AG17] D. Akakpo and E. Gildin. A control perspective to adaptive time-stepping in reservoir simulation. In *Society of Petroleum Engineers - SPE Reservoir Simulation Conference 2017*, pages 213–232, 2017. doi:10.2118/182601-ms. 8

[AKFIJ03] H. Abou-Kandil, G. Freiling, V. Ionescu, and G. Jank. *Matrix Riccati Equations in Control and Systems Theory*. Systems & Control: Foundations & Applications. Birkhäuser, Basel, Switzerland, 2003. doi:10.1007/978-3-0348-8081-7. 6

[ANS14] H. Antil, R. H. Nochetto, and P. Sodré. Optimal control of a free boundary problem: Analysis with second-order sufficient conditions. *SIAM J. Control Optim.*, 52(5):2771–2799, 2014. doi:10.1137/120893306. 5

[ANS15] H. Antil, R. H. Nochetto, and P. Sodré. Optimal control of a free boundary problem with surface tension effects: A priori error analysis. *SIAM J. Numer. Anal.*, 53(5):2279–2306, 2015. doi:10.1137/140958360. 5

[AP98] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, 1998. doi:10.1137/1.9781611971392. xiii, 36, 63

[Bar16] B. Baran. Optimal control of a Stefan problem with gradient-based methods in FEniCS. Master's thesis, Otto-von-Guericke-Universität, Magdeburg, Germany, 2016. URL: http://nbn-resolving.de/urn:nbn:de:gbv:ma9:1-7871. 4, 21, 22, 23, 24, 28, 29, 31

[Bar24] B. Baran. Linear Quadratic Regulator Computation for a Stefan Problem using M.-M.E.S.S. and FEniCS, February 2024. doi:10.5281/zenodo.10684136. iii, 30, 62

[BBH21] M. Behr, P. Benner, and J. Heiland. Invariant Galerkin trial spaces and Davison-Maki methods for the numerical solution of differential Riccati equations. *Appl. Math. Comput.*, 410:126401, 2021. doi:10.1016/j.amc.2021.126401. 7, 96

[BBHS17] B. Baran, P. Benner, J. Heiland, and J. Saak. Towards Riccati-feedback control of complex flows with moving interfaces. *Proc. Appl. Math. Mech.*, 17(1):769–770, March 2017. doi:10.1002/pamm.201710352. iii, iv, 117

[BBHS18] B. Baran, P. Benner, J. Heiland, and J. Saak. Optimal control of a Stefan problem fully coupled with incompressible Navier–Stokes equations and mesh movement. *Analele Stiintifice ale Universitatii Ovidius Constanta: Seria Matematica*, XXVI(2):11–40, 2018. doi:10.2478/auom-2018-0016. iii, iv, 4, 5, 12, 17, 21, 31, 95, 97, 117

[BBS22] B. Baran, P. Benner, and J. Saak. Riccati feedback control of a two-dimensional two-phase Stefan problem. e-print arXiv:2209.05476v2, arXiv, September 2022. math.NA. URL: https://arxiv.org/abs/2209.05476. iii, iv, 6, 7, 12

[BBSS24] B. Baran, P. Benner, J. Saak, and T. Stillfjord. Numerical methods for closed-loop systems with non-autonomous data. e-print arXiv:2402.13656, arXiv, February 2024. math.NA. URL: https://arxiv.org/abs/2402.13656. iii, iv, 6, 7, 8, 40, 41, 117

[BDS21] T. Breiten, S. Dolgov, and M. Stoll. Solving differential Riccati equations: A nonlinear space-time method using tensor trains. *Numer. Algebra, Control. Optim.*, 11(3):407–429, 2021. doi:10.3934/naco.2020034. 7

[Ber10] M. Bernauer. *Motion Planning for the Two-Phase Stefan Problem in Level Set Formulation*. PhD thesis, Technische Universität Chemnitz, Chemnitz, Germany, 2010. URL: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-63654. 4, 5, 21

[BG91] R. R. Bitmead and M. Gevers. *Riccati Difference and Differential Equations: Convergence, Monotonicity and Stability*, pages 263–291. Springer-Verlag, Berlin, Heidelberg, 1991. doi:10.1007/978-3-642-58223-3_10. 6

[BGP87] M. O. Bristeau, R. Glowinski, and J. Periaux. Numerical methods for the Navier-Stokes equations. applications to the simulation of compressible and incompressible viscous flows. *Comput. Phys. Rep.*, 6(1):73 – 187, 1987. doi:10.1016/0167-7977(87)90011-6. 8, 57

[BH11] M. K. Bernauer and R. Herzog. Optimal control of the classical two-phase Stefan problem in level set formulation. *SIAM J. Sci. Comput.*, 33(1):342–363, 2011. doi:10.1137/100783327. 4, 5

[BH16] B. Baran and J. Heiland. Adjoint-based optimal boundary control of a Stefan problem system fully coupled with Navier–Stokes equations. *Proc. Appl. Math. Mech.*, 16(1):779–780, October 2016. doi:10.1002/pamm.201610378. iv

[BL18] P. Benner and N. Lang. Peer methods for the solution of large-scale differential matrix eqautions. e-print arXiv:1804.08524, arXiv, July 2018. math.NA. URL: https://arxiv.org/pdf/1804.08524.pdf. 7

[BLP08] P. Benner, J.-R. Li, and T. Penzl. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. *Numer. Lin. Alg. Appl.*, 15(9):755–777, 2008. doi:10.1002/nla.622. 33

[BM04] P. Benner and H. Mena. BDF methods for large-scale differential Riccati equations. In B. De Moor, B. Motmans, J. Willems, P. Van Dooren, and V. Blondel, editors, *Proc. 16th Intl. Symp. Mathematical Theory of Network and Systems, MTNS 2004*, 2004. 7, 35

[BM18] P. Benner and H. Mena. Numerical solution of the infinite-dimensional LQR-problem and the associated differential Riccati equations. *J. Numer. Math.*, 26(1):1–20, 2018. published online May 2016. doi:10.1515/jnma-2016-1039. 7

[BMR01] G. Beckett, J. A. Mackenzie, and M. L. Robertson. A moving mesh finite element method for the solution of two-dimensional Stefan problems. *J. Comput. Phys.*, 168(2):500–518, 2001. doi:10.1006/jcph.2001.6721. 4

[BPS10] E. Bänsch, J. Paul, and A. Schmidt. An ALE FEM for solid-liquid phase transitions with free melt surface. Technical report, Zentrum für Technomathematik, University of Bremen, 2010. URL: http://www.math.uni-bremen.de/zetem/cms/media.php/262/report1007.pdf. 4

[BPS13] E. Bänsch, J. Paul, and A. Schmidt. An ALE finite element method for a coupled Stefan problem and Navier–Stokes equations with free capillary surface. *Internat. J. Numer. Methods Fluids*, 71(10):1282–1296, 2013. doi:10.1002/fld.3711. 4, 18, 95

[BSVU06] J. H. Brusche, A. Segal, C. Vuik, and H. P. Urbach. A comparison of enthalpy and temperature methods for melting problems on composite domains. In *Numerical mathematics and advanced applications*, pages 585–592. Springer-Verlag, Berlin, 2006. doi:10.1007/978-3-540-34288-5_55. 4

[CH52] C. F. Curtiss and J. O. Hirschfelder. Integration of stiff equations*. *Proceedings of the National Academy of Sciences*, 38(3):235–243, 1952. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.38.3.235, doi:10.1073/pnas.38.3.235. 35

[CMOS97] S. Chen, B. Merriman, S. Osher, and P. Smereka. A simple level set method for solving Stefan problems. *J. Comput. Phys.*, 135(1):8–29, 1997. doi:10.1006/jcph.1997.5721. 4

[CRD20] I. Cvok, I. Ratkovic, and J. Deur. Optimization of control parameters of vehicle air-conditioning system for maximum efficiency. *SAE Technical Papers*, 2020-April(April), 2020. doi:10.4271/2020-01-1242. 8

[DGH82] J. Donea, S. Giuliani, and J. P. Halleux. An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid-structure interactions. *Comput. Methods Appl. Mech. Engrg.*, 33(1):689 – 723, 1982. doi:10.1016/0045-7825(82)90128-1. 4

[Die92] L. Dieci. Numerical integration of the differential Riccati equation and some related issues. *SIAM J. Numer. Anal.*, 29(3):781–815, 1992. doi:10.1137/0729049. 35

[DPRM03] W. B. Dunbar, N. Petit, P. Rouchon, and P. Martin. Motion planning for a nonlinear Stefan problem. *ESAIM: Control, Optimisation and Calculus of Variations*, 9:275–296, 2003. doi:10.1051/cocv:2003013. 6

[EBRW21] S. Ecklebe, T. Buchwald, P. Rüdiger, and J. Winkler. Model predictive control of the vertical gradient freeze crystal growth process. *IFAC-PapersOnLine*, 54(6):218–225, 2021. 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021. doi:10.1016/j.ifacol.2021.08.548. 6

[EWFR$^+$22] S. Ecklebe, F. Woittennek, C. Frank-Rotsch, N. Dropka, and J. Winkler. Toward model-based control of the vertical gradient freeze crystal growth process. *IEEE Trans. Control Syst. Technol.*, 30(1):384–391, 2022. doi:10.1109/TCST.2021.3058006. 6, 95

[EWW20] S. Ecklebe, F. Woittennek, and J. Winkler. Control of the vertical gradient freeze crystal growth process via backstepping. *IFAC-PapersOnLine*, 53(2):7758–7764, 2020. 21st IFAC World Congress. doi:10.1016/j.ifacol.2020.12.1537. 6

[Fai17] L. Failer. *Optimal Control of Time-Dependent Nonlinear Fluid-Structure Interaction.* Dissertation, Technische Universität München, München,

2017. URL: https://mediatum.ub.tum.de/doc/1355578/1355578.pdf. 57

[FRM08] F. Freitas, J. Rommes, and N. Martins. Gramian-based reduction method applied to large sparse power system descriptor models. *IEEE Trans. Power Syst.*, 23(3):1258–1270, August 2008. doi:10.1109/TPWRS.2008.926693. 16, 17

[FW18] L. Failer and T. Wick. Adaptive time-step control for nonlinear fluid-structure interaction. *J. Comput. Phys.*, 366:448–477, 2018. doi:10.1016/j.jcp.2018.04.021. 8, 20, 56, 58, 77, 96

[GHJK18] Y. Güldoğan, M. Hached, K. Jbilou, and M. Kurulay. Low rank approximate solutions to large-scale differential matrix Riccati equations. *Appl. Math. (Warsaw)*, 45(2):233–254, 2018. doi:10.4064/am2355-1-2018. 7

[Glo91] 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 (Seattle, WA, 1990)*, volume 28 of *Lectures in Appl. Math.*, pages 219–301. Amer. Math. Soc., Providence, RI, 1991. doi:10.1016/S1570-8659(03)09003-3. 97

[GLS88] K. Gustafsson, M. Lundh, and G. Söderlind. A PI stepsize control for the numerical solution of ordinary differential equations. *BIT*, 28(2):270–287, 1988. doi:10.1007/BF01934091. 8

[GSW13] S. Gugercin, T. Stykel, and S. Wyatt. Model reduction of descriptor systems by interpolatory projection methods. *SIAM J. Sci. Comput.*, 35(5):B1010–B1033, 2013. doi:10.1137/130906635. 98, 99

[Gup18] S. C. Gupta. *The Classical Stefan Problem (Second Edition)*. Elsevier, Amsterdam, 2018. doi:10.1016/B978-0-444-63581-5.09985-1. 3

[HKS] C. Himpe, M. Köhler, and J. Saak. Oberwolfach Rail Reimplementation in FEniCS. URL: https://gitlab.mpi-magdeburg.mpg.de/models/fenicsrail. 64

[HLZ81] T. J. R. Hughes, W. K. Liu, and T. K. Zimmermann. Lagrangian–Eulerian finite element formulation for incompressible viscous flows. *Comput. Methods Appl. Mech. Engrg.*, 29(3):329 – 349, 1981. doi:10.1016/0045-7825(81)90049-9. 4

[HV03] W. Hundsdorfer and J. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, volume 33 of *Springer Series in*

Computational Mathematics. Springer-Verlag, Berlin, 2003. doi:10.1007/978-3-662-09017-6. 38, 53

[HWL06] E. Hairer, G. Wanner, and C. Lubich. *Geometric Numerical Integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, second edition, 2006. doi:10.1007/978-3-662-05018-7. 38

[HZ07] M. Hinze and S. Ziegenbalg. Optimal control of the free boundary in a two-phase Stefan problem with flow driven by convection. *ZAMM Z. Angew. Math. Mech.*, 87(6):430–448, 2007. doi:10.1002/zamm.200610326. 4

[JR10] V. John and J. Rang. Adaptive time step control for the incompressible Navier–Stokes equations. *Comp. Meth. Appl. Mech. Eng.*, 199(9):514 – 524, 2010. doi:10.1016/j.cma.2009.10.005. 8

[JWN20] Q. Jin, J. T. Wen, and S. Narayanan. Moving boundary model for dynamic control of multi-evaporator cooling systems facing variable heat loads [modèle à frontières mobiles pour la régulation dynamique des systèmes de refroidissement à évaporateurs multiples confrontés à des charges thermiques variables]. *Int. J. Refrig.*, 120:481–492, 2020. doi:10.1016/j.ijrefrig.2020.09.014. 8

[KDK16] S. Koga, M. Diagne, and M. Krstic. Output feedback control of the one-phase Stefan problem. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, page 526–531. IEEE Press, 2016. doi:10.1109/CDC.2016.7798322. 6

[KDK19] S. Koga, M. Diagne, and M. Krstic. Control and state estimation of the one-phase Stefan problem via backstepping design. *IEEE Trans. Autom. Control*, 64(2):510–525, 2019. doi:10.1109/TAC.2018.2836018. 6, 8, 95

[KK19] S. Koga and M. Krstic. Control of two-phase Stefan problem via single boundary heat input. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2914–2919, 2019. doi:10.1109/CDC.2018.8619638. 6

[KK20a] S. Koga and M. Krstic. Single-boundary control of the two-phase Stefan system. *Systems Control Lett.*, 135:104573, 2020. doi:10.1016/j.sysconle.2019.104573. 6

[KK20b] S. Koga and M. Krstic. *Two-Phase Stefan Problem*, pages 139–157. Springer International Publishing, Cham, 2020. doi:10.1007/978-3-030-58490-0_5. 3

[KM90a] P. Kunkel and V. Mehrmann. Numerical solution of differential algebraic Riccati equations. *Linear Algebra Appl.*, 137/138:39–66, 1990. doi:10.1016/0024-3795(90)90126-W. 6, 47, 48

[KM90b] P. Kunkel and V. Mehrmann. Numerical solution of Riccati differential algebraic equations. In M. A. Kaashoek, J. H. Schuppen, and A. S. M. Ran, editors, *Proceedings of the International Symposium on the Mathematical Theory of Networks and Systems, Amsterdam, Netherlands, June 1989*, volume III, pages 479–487, Basel, 1990. Birkhäuser. 7

[KM06] P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations: Analysis and Numerical Solution.* Textbooks in Mathematics. EMS Publishing House, Zürich, Switzerland, 2006. 16

[KM20] A. Koskela and H. Mena. Analysis of Krylov subspace approximation to large-scale differential Riccati equations. *Electron. Trans. Numer. Anal.*, 52:431–454, 2020. doi:10.1553/etna_vol52s431. 7

[KMC$^+$20] S. Koga, M. Makihata, R. Chen, M. Krstic, and A.P. Pisano. Energy storage in paraffin: A PDE backstepping experiment. *IEEE Trans. Control Syst. Technol.*, pages 1490–1502, 2020. doi:10.1109/TCST.2020.3014295. 8

[KS20] G. Kirsten and V. Simoncini. Order reduction methods for solving large-scale differential matrix Riccati equations. *SIAM J. Sci. Comput.*, 42(4):A2182–A2205, 2020. doi:10.1137/19M1264217. 7

[Lan17] N. Lang. *Numerical Methods for Large-Scale Linear Time-Varying Control Systems and related Differential Matrix Equations.* Dissertation, Technische Universität Chemnitz, Germany, June 2017. Logos-Verlag, Berlin, ISBN 978-3-8325-4700-4. URL: https://www.logos-verlag.de/cgi-bin/buch/isbn/4700. 7, 35, 64, 66, 95

[LC31] G. Lamé and B. P. Clapeyron. Mémoire sur la solidification par refroidissement d'un globe liquide. In *Annales Chimie Physique*, volume 47, pages 250–256, 1831. 1, 3

[LEG$^+$12] G. Leugering, S. Engell, A. Griewank, M. Hinze, R. Rannacher, V. Schulz, M. Ulbrich, and S. Ulbrich. *Constrained Optimization and Optimal Control for Partial Differential Equations.* Birkhäuser, Basel, Switzerland, 2012. doi:10.1007/978-3-0348-0133-1. 5

[LF20] L. Lu and Y. Feng. Observability and stabilization of 1-D wave equations with moving boundary feedback. *Acta Appl. Math.*, 170(1):731–753, 2020. doi:10.1007/s10440-020-00356-4. 8

[LMS15] N. Lang, H. Mena, and J. Saak. On the benefits of the LDL^T factorization for large-scale differential matrix equation solvers. *Linear Algebra Appl.*, 480:44–71, 2015. doi:10.1016/j.laa.2015.04.006. 7, 35, 36

[Loc01] A. Locatelli. *Optimal Control: An Introduction*. Birkhäuser, Basel, Switzerland, 2001. doi:10.1007/978-3-0348-8328-3. 6, 32, 33

[LWH12] A. Logg, G. N. Wells, and J. Hake. *DOLFIN: a C++/Python Finite Element Library*, chapter 10. Springer-Verlag, Berlin, 2012. doi:10.1145/1731022.1731030. 7

[LZL20] D. Li, X. Zhang, and R. Liu. Exponential integrators for large-scale stiff Riccati differential equations. *J. Comput. Appl. Math.*, 389:113360, 2020. doi:10.1016/j.cam.2020.113360. 7

[Mar12] J. Marburger. Adjoint-based optimal control of time-dependent free boundary problems. e-print arXiv:1212.3789, arXiv, 2012. URL: http://adsabs.harvard.edu/abs/2012arXiv1212.3789M. 5

[MC14] A. Maidi and J.-P. Corriou. Boundary geometric control of a linear Stefan problem. *Journal of Process Control*, 24(6):939–946, 2014. Energy Efficient Buildings Special Issue. doi:10.1016/j.jprocont.2014.04.010. 6

[Meh91] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution.* Number 163 in Lecture Notes in Control and Information Sciences. Springer-Verlag, Berlin, July 1991. doi:10.1007/BFb0039443. 33

[Men12] H. Mena. *Numerical Solution of Differential Riccati Equations Arising in Optimal Control Problems for Parabolic Partial Differential Equations*. Unidad de Publicaciones de la Facultad de Ciencias, Quito-Ecuador, first edition, May 2012. Available as ISBN: 978-9978-383-09-4. 7, 35

[MOPP18] H. Mena, A. Ostermann, L.-M. Pfurtscheller, and C. Piazzola. Numerical low-rank approximation of matrix differential equations. *J. Comput. Appl. Math.*, 340:602–614, 2018. doi:10.1016/j.cam.2018.01.035. 7

[MQ02] R. I. McLachlan and G. R. W. Quispel. *Splitting methods*, volume 11 of *Acta Numerica*, page 341–434. Cambridge University Press, 2002. doi:10.1017/CBO9780511550140.005. 38

[MR14] D. Meidner and T. Richter. Goal-oriented error estimation for the fractional step theta scheme. *Comput. Methods Appl. Math.*, 14(2):203 – 230, 2014. doi:10.1515/cmam-2014-0002. 8

[MR15] D. Meidner and T. Richter. A posteriori error estimation for the fractional step theta discretization of the incompressible Navier–Stokes equations. *Comp. Meth. Appl. Mech. Eng.*, 288:45 – 59, 2015. doi:10.1016/j.cma.2014.11.031. 8

[MSS99] A. Murua and J. M. Sanz-Serna. Order conditions for numerical integrators obtained by composing simpler integrators. *Philos. Trans. Roy. Soc. A*, 357(1754):1079–1100, 1999. doi:10.1098/rsta.1999.0365. 38

[NCM11] M. Niezgodka, A. Crowley, and A. M. Meirmanov. *The Stefan Problem*. De Gruyter, 2011. doi:10.1515/9783110846720.245. 3

[NPV91a] R. H. Nochetto, M. Paolini, and C. Verdi. An adaptive finite element method for two-phase Stefan problems in two space dimensions. I: Stability and error estimates. *Math. Comput.*, 57(195):73–108, 1991. doi:10.2307/2938664. 4

[NPV91b] R. H. Nochetto, M. Paolini, and C. Verdi. An adaptive finite element method for two-phase Stefan problems in two space dimensions. II: Implementation and numerical experiments. *SIAM Journal on Scientific and Statistical Computing*, 12(5):1207–1244, 1991. doi:10.1137/0912065. 4

[OPW19] A. Ostermann, C. Piazzola, and H. Walach. Convergence of a low-rank Lie–Trotter splitting for stiff matrix differential equations. *SIAM J. Numer. Anal.*, 57(4):1947–1966, 2019. doi:10.1137/18M1177901. 7

[PBT10] B. Petrus, J. Bentsman, and B. G. Thomas. Feedback control of the two-phase Stefan problem, with an application to the continuous casting of steel. In *49th IEEE Conference on Decision and Control (CDC)*, pages 1731–1736, 2010. doi:10.1109/CDC.2010.5717456. 6

[PBT12] B. Petrus, J. Bentsman, and B. G. Thomas. Enthalpy-based feedback control algorithms for the Stefan problem. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 7037–7042, 2012. doi:10.1109/CDC.2012.6426035. 6

[PBT14] B. Petrus, J. Bentsman, and B. G. Thomas. Application of enthalpy-based feedback control methodology to the two-sided Stefan problem. In *2014 American Control Conference*, pages 1015–1020, 2014. doi:10.1109/ACC.2014.6859062. 6

[PS05] Benner. P. and J. Saak. Linear-quadratic regulator design for optimal cooling of steel profiles. Technical Report SFB393/05-05, Sonderforschungsbereich 393 *Parallele Numerische Simulation für Physik und Kontinuumsmechanik*, TU Chemnitz, D-09107 Chemnitz (Germany), 2005. URL: `http://nbn-resolving.de/urn:nbn:de:swb:ch1-200601597`. 66

[Rei72] W. T. Reid. *Riccati Differential Equations*, volume 86 of *Mathematics in Science and Engineering*. Academic Press, New York, 1972. `doi:10.1115/1.3426914`. 6

[Rub71] L. I. Rubenšteĭn. *The Stefan problem*, volume 27 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, R.I., 1971. Translated from the Russian by A. D. Solomon. `doi:10.1090/mmono/027`. 3

[RW15] T. Richter and T. Wick. On time discretizations of fluid-structure interactions. In *Multiple Shooting and Time Domain Decomposition Methods*, pages 377–400, Cham, 2015. Springer International Publishing. `doi:10.1007/978-3-319-23321-5_15`. 8

[RWW04] J. Rudolph, J. Winkler, and F. Woittennek. Flatness based trajectory planning for two heat conduction problems occurring in crystal growth technology. volume 1, 2004. 6

[SBB19] J. Saak, B. Baran, and P. Benner. Riccati-feedback control of a two-dimensional two-phase Stefan problem. In T. Meurer and F. Woittennek, editors, *Tagungsband GMA-FA 1.30 'Modellbildung, Identifikation und Simulation in der Automatisierungstechnik' und GMA-FA 1.40 'Systemtheorie und Regelungstechnik', Workshops in Anif, Salzburg, 23.-27.09.2019*, pages 478–496, 2019. iv

[SKB] J. Saak, M. Köhler, and P. Benner. M-M.E.S.S. – the Matrix Equations Sparse Solvers library. See also: `https://www.mpi-magdeburg.mpg.de/projects/mess`. `doi:10.5281/zenodo.632897`. 17, 54, 56, 62

[Son98] E. D. Sontag. Mathematical Control Theory, 1998. `doi:10.1007/978-1-4612-0577-7`. 6, 32, 33

[Ste89] J. Stefan. Über einige Probleme der Theorie der Wärmeleitung. *Sitzungber., Wien, Akad. Mat. Natur*, 98:473–484, 1889. 1, 3

[Ste90] J. Stefan. Über die Theorie der Eisbildung. *Monatshefte für Mathematik*, 1(1):1–6, 1890. 1, 3

[Ste91] J. Stefan. Über die Theorie der Eisbildung, insbesondere über die Eisbildung im Polarmeere. *Annalen der Physik und Chemie*, 42:269–286, 1891. 1, 3

[Sti15a] T. Stillfjord. Low-rank second-order splitting of large-scale differential Riccati equations. *IEEE Trans. Autom. Control*, 60(10):2791–2796, 2015. doi:10.1109/TAC.2015.2398889. 7, 39

[Sti15b] T. Stillfjord. *Splitting schemes for nonlinear parabolic problems.* PhD thesis, Lund University, 2015. URL: https://lup.lub.lu.se/search/ws/files/3905802/5277358.pdf. 7, 38

[Sti18a] T. Stillfjord. Adaptive high-order splitting schemes for large-scale differential Riccati equations. *Numer. Algorithms*, 78:1129–1151, 2018. doi:10.1007/s11075-017-0416-8. 7, 39, 40, 43, 64, 96

[Sti18b] T. Stillfjord. Singular value decay of operator-valued differential Lyapunov and Riccati equations. *SIAM J. Control Optim.*, 56:3598–3618, 2018. doi:10.1137/18M1178815. 7, 33, 35, 53

[SUS15] T. Song, K. Upreti, and G. Subbarayan. A sharp interface isogeometric solution to the Stefan problem. *Comp. Meth. Appl. Mech. Eng.*, 284:556–582, 2015. doi:10.1016/j.cma.2014.10.013. 8

[Trö10] F. Tröltzsch. *Optimal control of partial differential equations. Theory, methods and applications.*, volume 112 of *Graduate Studies in Mathematics*. American Mathematical Society (AMS), Providence, RI, 2010. doi:10.1090/gsm/112. 5, 21, 22, 29

[Tur99] S. Turek. *Efficient solvers for incompressible flow problems*, volume 6 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 1999. doi:10.1007/978-3-642-58393-3. 8

[Vab14] P. N. Vabishchevich, editor. *Computational Technologies. Advanced Topics.* De Gruyter, Berlin, München, Boston, 2014. doi:10.1515/9783110359961. 4

[Van13] J. Vanhellemont. The v/g criterion for defect-free silicon single crystal growth from a melt revisited: Implications for large diameter crystals. *J. Cryst. Growth*, 381:134–138, 2013. doi:10.1016/j.jcrysgro.2013.06.039. 3

[Vis96] A. Visintin. *Models of Phase Transitions.* Birkhäuser, Basel, Switzerland, 1996. doi:10.1007/978-1-4612-4078-5. 3

[Wei16] H. K. Weichelt. *Numerical Aspects of Flow Stabilization by Riccati Feedback.* Dissertation, Otto-von-Guericke-Universität, Magdeburg, Germany, January 2016. doi:10.25673/4493. 6, 32, 95

[Whi82a] R. E. White. An Enthalpty Formulation of the Stefan Problem. *SIAM Journal on Numerical Analysis*, 19(6):1129–1157, 1982. doi:10.1137/0719082. 4

[Whi82b] R. E. White. A numerical solution of the enthalpy formulation of the Stefan problem. *SIAM Journal on Numerical Analysis*, 19(6):1158–1172, 1982. doi:10.1137/0719083. 4

[Wic11] T. Wick. *Adaptive Finite Element Simulation of Fluid-Structure Interaction with Application to Heart-Valve Dynamics.* Dissertation, Heidelberg University, Heidelberg, Germany, December 2011. doi:10.11588/heidok.00012992. 8

[ZGT06] N. Zabaras, B. Ganapathysubramanian, and L. Tan. Modelling dendritic solidification with melt convection using the extended finite element method. *J. Comput. Phys.*, 218(1):200 – 227, 2006. doi:10.1016/j.jcp.2006.02.002. 4

[Zie08] S. Ziegenbalg. *Kontrolle freier Ränder bei der Erstarrung von Kristallschmelzen.* PhD thesis, Technische Universität Dresden, Dresden, Germany, 2008. In German. URL: http://nbn-resolving.de/urn:nbn:de:bsz:14-ds-1212521184972-55836. 4, 5, 19, 21, 30

STATEMENT OF SCIENTIFIC COOPERATIONS

This work is based on articles and reports (published and unpublished) that have been obtained in cooperation with various coauthors. To guarantee a fair assessment of this thesis, this statement clarifies the contributions that each individual coauthor has made. The following people contributed to the content of this work:

- Peter Benner: academic supervisor, coauthor of all publications
- Jan Heiland: supervisor of the authors' Master thesis and coauthor of the resulting publications [BBHS17, BBHS18]
- Jens Saak: academic mentor, coauthor of all publications
- Tony Stillfjord: coauthor of [BBSS24], developed non-autonomous splitting schemes

EHRENERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert oder verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadenersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, 24.04.2023

Björn Baran