Model Predictive Control for Partial Differential Equations

Von der Universität Bayreuth
zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Abhandlung

vorgelegt von

Nils Altmüller

aus Bad Oldesloe

Bayreuth 2014

1. Gutachter:	Prof. Dr. Lars Grüne
2. Gutachter:	Prof. Dr. Kurt Chudej

Inhaltsverzeichnis

Deutsche Zusammenfassung

Das Thema dieser Dissertation ist die Modellprädiktive Regelung (Model Predictive Control (MPC)) von partiellen Differentialgleichungen (Partial Differential Equations (PDE)). Im Englischen werden weiterhin die Bezeichnungen „Receding Horizon Control“(RHC) sowie „Nonlinear Model Predictive Control“(NMPC) bei nichtlinearen Systemen verwendet. Die modellprädiktive Regelung ist eine Methode der Kontrolltheorie und wird zur Stabilisierung eines Gleichgewichts in Kontrollsystemen genutzt. Die Idee ist hierbei, die Lösung eines Optimalsteuerungsproblems auf einem unendlichen Zeithorizont durch eine iterative Abfolge von Optimalsteuerungsproblemen auf einem endlichen Zeithorizont zu approximieren. Das Vorgehen lässt sich in drei Schritten beschreiben:

- Der aktuelle Zustand des Systems wird gemessen.
- Ausgehend von einem Modell wird das Verhalten des Systems prädiziert und aus der Lösung eines Optimalsteuerungsproblem eine optimale Folge von Steuerwerten berechnet.
- Das erste Element dieser Folge wird im System implementiert und der Optimierungshorizont wird vorwärts auf den nächsten Zeitschritt verschoben.

Durch das Wiederholen dieser Schritte wird eine Steuerfolge auf dem unendlichen Zeithorizont erzeugt.
Die modellprädiktive Regelung wurde in den vergangenen Jahrzehnten theoretisch untersucht (siehe [80]) und mit großem Erfolg in industriellen Anwendungen genutzt (siehe [79]). Hierbei lag der Schwerpunkt in der Regel auf Systemen, die von gewöhnlichen Differentialgleichungen (ODE) erzeugt werden. Diese werden in der Systemtheorie auch als endlich dimensional bezeichnet. In den Naturwissenschaften und technischen Anwendungen treten jedoch häufig Prozesse auf, welche durch partielle Differentialgleichungen beschrieben werden. Dies sind beispielsweise Probleme der Wärmeausbreitung oder der Struktur- und Strömungsmechanik. Diese Systeme werden in der Systemtheorie auch als unendlich dimensionale- oder verteilt-parametrische Systeme bezeichnet. Da die Idee von MPC nicht von der Art des dynamischen Systems abhängt, ist es naheliegend das Konzept auch für partielle Differentialgleichungen zu nutzen. Dies wurde z.B. im Rahmen der Strömungskontrolle in [16] und [50] sowie in der Anwendung auf Gleichungen der Verfahrenstechnik in [30] umgesetzt. Es ist zu beachten, dass die theoretische Analyse des zugehörigen Optimalsteuerungsproblems deutlich schwieriger als im ODE Fall sein kann. Dies gilt ebenfalls für die numerischen Algorithmen zur Lösung des Optimierungsproblems.

Ein wichtiger Aspekt im Zusammenhang mit MPC ist die zugehörige Stabilitätsanalyse. Hier wurden in der Vergangenheit häufig künstliche Endbeschränkungen und -kosten eingeführt, um die Stabilität des Systems garantieren zu können. Da jedoch in der Anwendung oftmals auf derlei Hilfsmittel verzichtet wird (siehe [79]), betrachten wir in dieser Arbeit die so genannte unrestringierte modellprädiktive Regelung. Eine wichtige Rolle in der Stabilitätsanalyse dieser Methode spielt der Optimierungshorizont: Ist dieser zu kurz, kann es zur Instabilität oder zu einer schlechten Regelgüte kommen. Ist der Optimierungshorizont zu lang gewählt, ist eine Berechnung des Optimalsteuerungsproblems in sinnvoller Zeit unter Umständen nicht möglich. Die Horizontlänge ist daher in dieser Arbeit eine wichtige Größe.

Gliederung und eigener Beitrag

Diese Arbeit gliedert sich in sechs Kapitel:

- Der erste Abschnitt des ersten Kapitels stellt bekannte Grundlagen der mathematischen Kontrolltheorie vor. Die Darstellung der Ergebnisse folgt im Wesentlichen [45]. Zunächst werden zeitkontinuierliche und zeitdiskrete Kontrollsysteme eingeführt und der Unterschied zwischen einem offenen und einem geschlossenem Regelkreis erläutert. Weiterhin wird gezeigt, wie zeitkontinuierliche Systeme mittels Abtastung zeitdiskret interpretiert werden können. Anschließend wird der Begriff der asymptotischen Stabilität eines Gleichgewichtes mit Hilfe von Vergleichs- und Lyapunovfunktionen eingeführt. Im zweiten Abschnitt des Kapitels wird die Methode der modellprädiktiven Regelung vorgestellt. Hierfür werden die benötigten Hilfsmittel wie Zustandskosten, Kostenfunktional und optimale Wertefunktion definiert. Der Schwerpunkt des Abschnittes liegt auf dem Konzept der relaxierten Lyapunov Ungleichung sowie auf Abschätzungen des Grades der Suboptimalität des MPC Reglers. Die entsprechenden Sätze bilden die Grundlage der theoretischen Ergebnisse in Kapitel 3.

- Im ersten Abschnitt von Kapitel 2 wird zunächst das Optimalsteuerungsproblem als abstraktes Optimierungsproblem im Banachraum beschrieben. Dies erleichtert das Aufstellen der notwendigen Bedingungen, welche auch für die Algorithmen in Kapitel 4 relevant sind. In Abschnitt 2.2 werden dann die benötigten Resultate der Optimalsteuerung partieller Differentialgleichungen aus [91] präsentiert. Hierzu werden zunächst die zugehörigen Sobolev Räume definiert. Der Schwerpunkt liegt dann auf den Existenz- und Regularitätssätzen für die in dieser Arbeit betrachteten partiellen Differentialgleichungen. Anschließend wird in Abschnitt 2.3 auf die Diskretisierung der partiellen Differentialgleichung eingegangen. Für die Ortsdiskretisierung wird zunächst das Galerkin Verfahren vorgestellt. Danach erfolgt eine kurze Übersicht der Finite Elemente Methode (nach [42]) sowie der Modellreduktionstechnik Proper Orthogonal Decomposition (POD) (nach [96]). Abschließend wird mit dem semi-impliziten Euler Verfahren eine Möglichkeit der Zeitdiskretisierung präsentiert.

- Kapitel 3 stellt den theoretischen Beitrag dieser Arbeit dar. Hier werden konkrete Abschätzungen des minimal stabilisierenden Horizonts für verschiedene Klassen von partiellen Differentialgleichungen bewiesen. Weiterhin werden die theoretischen Resultate genutzt, um die Abhängigkeit des Horizonts von Parametern zu diskutieren. Anhand von numerischen Simulationen werden die Resultate erläutert. In Abschnitt 3.1 wird die semilineare Wärmeleitungsgleichung mit verteilter Steuerung bezüglich des Horizonts analysiert. Der Abschnitt verallgemeinert die bereits publizierten Resultate in [8]. Insbesondere erfolgt eine Untersuchung über die Auswirkung unterschiedlicher Kostenfunktionale. Abschnitt 3.2 beinhaltet einen kurzen Einschub über die Backstepping Methode, welche im Folgenden benötigt wird. Die Herleitung orientiert sich an [62]. Die lineare Wärmeleitungsgleichung mit Dirichlet Randsteuerung wird in Abschnitt 3.3 betrachtet. Der Schwerpunkt liegt auch hier auf der Parameterabhängigkeit des Horizonts sowie auf dem Einfluss der verschiedenen Kostenfunktionale. Weiterhin werden die Unterschiede zur verteilten Steuerung aufgezeigt und durch numerische Ergebnisse belegt. Spezialfälle dieser Resultate wurden bereits in [4] veröffentlicht. In Abschnitt 3.4 wird die lineare Wärmeleitungsgleichung mit Neumann Randsteuerung betrachtet. Hierbei liegt das Hauptaugenmerk darauf, dass das Stabilitätsverhalten, abhängig von der Parameterwahl, entweder dem der verteilten Steuerung oder dem der Dirichlet Randsteuerung ähnelt. Der eindimensionale Spezialfall wurde bereits in [5] publiziert. Zusätzlich wird hier eine Analyse der unterschiedlichen Kostenfunktionale vorgenommen. In Abschnitt 3.5 wird gezeigt, wie die entwickelte Methode angepasst werden kann, um einen zusätzliche Konvektionsterm zu berücksichtigen. Die Auswirkungen des Konvektionsterms auf den minimal stabilisierenden Horizont wird theoretisch betrachtet und numerisch verifiziert. Ein Vergleich der hier benutzten Methode mit einem in [75] entwickelten Verfahren findet sich in Abschnitt 3.6 anhand der räumlich diskretisierten, linearen Wärmeleitungsgleichung. Es werden insbesondere die Vor- und Nachteile beider Methoden beschrieben. In Abschnitt 3.7 wird gezeigt, dass die Analyse nicht nur auf parabolische, sondern auch auf hyperbolische partielle Differentialgleichungen anwendbar ist. Am Beispiel der randgesteuerten Wellengleichung wird die Bedeutung der endlichen Ausbreitungsgeschwindigkeit dieser Gleichung dargelegt. Ein Spezialfall dieser Resultate wurde bereits in Zusammenarbeit mit Karl Worthmann in [8] und [6] publiziert. Im Gegensatz hierzu werden die Beweistechniken jedoch komplett geändert um allgemeinere Resultate zu erhalten. Insbesondere kann auf eine explizite Lösungsdarstellung verzichtet werden.

- Kapitel 4 beschäftigt sich mit den Algorithmen, die im entwickelten Softwarepaket implementiert sind. In Abschnitt 4.1 werden einige aus der Literatur bekannten Algorithmen zur Optimalsteuerung von partiellen Differentialgleichungen vorgestellt. Insbesondere wird hierbei auf die Anwendbarkeit der Verfahren für den MPC Algorithmus eingegangen. Abschnitt 4.2 befasst sich mit der Möglichkeit, die modellprädiktive Regelung mit der Modellreduktionstechnik Proper Orthogonal Decomposition zu kombinieren. Es werden aus der Literatur bekannte Algorithmen präsentiert sowie neue Algorithmen entwickelt. Weiterhin werden die Vor- und Nachteile der verschiedenen Ansätze diskutiert. In den Arbeiten [77] und [38] wurden Algorithmen entwickelt, in denen die Horizontlänge während des MPC Verfahrens nicht länger konstant ist, sondern in jedem Schritt geändert werden kann. In Abschnitt 4.3 wird analysiert, inwiefern sich diese Algorithmen für die Anwendung auf partielle Differentialgleichungen eignen. Insbesondere wird eine neue Methode präsentiert, die zeigt, wie in diesem Zusammenhang Mehrgittermethoden sinnvoll eingesetzt werden können.

- Im ersten Abschnitt von Kapitel 5 werden zunächst Beispiele für semilineare partielle Differentialgleichungen motiviert und vorgestellt, welche in Kapitel 6 als Benchmark Probleme dienen. Der Fokus liegt hier auf der nicht monotonen Schlögl Gleichung sowie auf einer nichtlinearen partiellen Differentialgleichung aus der Verfahrenstechnik. Der Abschnitt 5.2 befasst sich mit der tatsächlichen Umsetzung der Algorithmen im C++ Softwarepaket. Hierbei liegt der Schwerpunkt auf der strukturellen Darstellung der objektorientierten Klassen. Außerdem werden Quellcodeauszüge angegeben, um dem Anwender die Benutzung des Programms aufzuzeigen. Ein vollständiges Beispiel findet sich in Appendix A. Weiterhin befindet sich das Softwarepaket auf der beigelegten CD-ROM.

- Das letzte Kapitel 6 beinhaltet die numerischen Resultate dieser Arbeit. In Abschnitt 6.1 wird für die Testbeispiele aus 5.1 gezeigt, dass eine Echtzeit Realisierung des MPC Algorithmus möglich ist. Danach werden in Abschnitt 6.2 die Optimierungsalgorithmen aus Abschnitt 4.1 numerisch miteinander verglichen und hinsichtlich ihrer Anwendbarkeit auf MPC analysiert. Weiterhin werden Parameterstudien bezüglich Diskretisierung, Optimierungshorizont, Regularisierungs- und Reaktionsparameter durchgeführt. Die Algorithmen aus Abschnitt 4.2, welche MPC mit POD kombinieren, werden in Abschnitt 6.3 verglichen. Abschließend werden in Abschnitt 6.4 der Nutzen und die Grenzen der Algorithmen mit variabler Horizontlänge in der Anwendung auf partielle Differentialgleichungen durch numerische Simulationen aufgezeigt.

Die Hauptbeiträge dieser Arbeit sind:

- Die theoretische Herleitung und Analyse konkreter Horizontabschätzungen für den MPC Algorithmus

- Das Entwickeln neuer Algorithmen und der numerische Vergleich mit aus der Literatur bekannten Methoden
- Die Implementierung der Algorithmen in einem C++ Softwarepaket

Danksagung

Ich möchte mich an dieser Stelle sehr herzlich bei meinem Doktorvater Prof. Dr. Lars Grüne für die hervorragende Betreuung dieser Arbeit bedanken. Dies gilt sowohl für viele hilfreiche Anregungen als auch für das Gewähren der Freiheit, dieses interessante Forschungsgebiet von verschiedenen Seiten beleuchten zu können.
Weiterhin möchte ich Herrn Prof. Dr. Hans Josef Pesch für die Unterstützung während des Studiums, die „Vermittlung“ an den Lehrstuhl für angewandte Mathematik und die „Wiederaufnahme“ an den Lehrstuhl für Ingenieurmathematik bedanken.
Für die gute Zusammenarbeit in der Lehre gilt mein Dank Herrn Prof. Dr. Kurt Chudej sowie Herrn Prof. Dr. Walter Olbricht.
Für die vielen interessanten (mathematischen) Diskussionen sowie das gute Arbeitsklima an den Lehrstühlen möchte ich mich bei Simon Bechmann, Julia Fischer, Sigrid Kinder, Marcus von Lossow, Florian Müller, Jürgen Pannek, Marleen Stieler, Stefan Wendl und Karl Worthmann bedanken.
Für die finanzielle Unterstützung, die interessanten Doktorandentreffen sowie die Ermöglichung des Forschungsaufenthalts an der K.U. Leuven gilt mein Dank dem internationalen Doktorandenkolleg „Identifikation, Optimierung und Steuerung für technische Anwendungen“ des Elitenetzwerks Bayern.
Der größte Dank gilt meiner Familie, ohne deren Unterstützung diese Arbeit nicht möglich gewesen wäre.

Summary

This thesis deals with Model Predictive Control (MPC) of Partial Differential Equations (PDE). Other designations are Nonlinear Model Predictive Control (NMPC) for nonlinear systems and Receding Horizon Control (RHC). MPC is a method in control theory which is used for the stabilization of equilibrium points in a control system. The idea is to approximate an optimal control problem on an infinite time horizon through a sequence of finite horizon optimal control problems. The procedure is the following:

- Measure the current state of the system.
- Predict the behaviour of the system based on the model and compute the optimal control sequence by solving the optimal control problem.
- Implement the first element of the control sequence to the system and shift the horizon forward in time.

Repeating these steps leads to a control sequence on the infinite time horizon.
Model Predictive Control was theoretically investigated during the last decades and it was successfully applied to industrial processes. The focus was mainly on systems which originate from Ordinary Differential Equations (ODE). These systems are also called finite dimensional systems. However, in many engineering applications partial differential equations are necessary to describe the physical phenomena. Examples are given by heat conduction, fluid flows and electromagnetic waves. These systems are known as infinite dimensional or distributed parameter systems. Since the idea of MPC is independent of the underlying system, it is quite natural to use this concept for partial differential equations, as well. The application of MPC to flow control problems, for instance, can be found in [16] and [50]. Further applications in the context of process engineering can be found in [30]. It should be mentioned that optimal control of partial differential equations can be a difficult task from the theoretical as well as from the numerical point of view.
An important aspect in the context of MPC is the corresponding stability analysis. In the MPC literature artificial terminal constraints or costs are often incorporated to guarantee stability of the system. However, in most industrial applications these tools are not used, cf. [79]. Thus, in this thesis we exclusively investigate the so called unconstrained MPC approach. The optimization horizon plays an important role in the stability analysis: A too short horizon can yield an unstable behaviour or a bad performance of the MPC controller. In contrast to that a large horizon increases the complexity of the resulting optimal control problem. Therefore, this

parameter plays a important role in our analysis.

Outline and Contribution

This thesis consists of six chapters:

- In the first section of Chapter 1 the basic concepts of the mathematical control theory are introduced. The representation essentially follows [45]. First, continuous-time and discrete-time control systems are defined and the difference between open-loop and closed-loop control is explained. Furthermore, we show that continuous-time control systems can be interpreted in a discrete way by using the idea of sampling. The concept of asymptotic stability of an equilibrium is introduced by using comparison functions. In the second part of this chapter we introduce the Model Predictive Control (MPC) algorithm and define the required terms like stage cost, cost functional and optimal value function. The focus of this section is the relaxed Lyapunov inequality and the suboptimality of the MPC controller. The corresponding theorems form the basis of the theoretical investigations in Chapter 3.
- In the first section of Chapter 2 we interpret the optimal control problem as an abstract optimization problem in a Banach space. This simplifies the development of the necessary conditions which are also required in Chapter 4. In Section 2.2 we present some results concerning the optimal control of partial differential equations, cf. [91]. The focus lies on the presentation of existence- and regularity theorems as well as on the definition of the required Sobolev spaces. Afterwards, in Section 2.3 the discretization of the PDE is considered. For the spatial discretization the Galerkin method is introduced. Its variants Finite Element Method (FEM), cf. [42], and the model reduction technique Proper Orthogonal Decomposition (POD), cf. [96], are discussed. Moreover, we present with the semi-implicit Euler method an example for the time discretization.
- Chapter 3 is the main theoretical contribution of this thesis. We present concrete estimates for the minimal stabilizing horizon for different classes of partial differential equations. Furthermore, the theoretical results are used to explain the dependence of the horizon on several parameters. In Section 3.1 we analyse the semilinear heat equation with distributed control with regard to the optimization horizon. This section generalizes our results in [8]. Especially, we examine the influence of different cost functionals. In Section 3.2 we give a short introduction to the backstepping method, which is required in the remaining chapter. The representation follows [62]. The linear heat equation with Dirichlet boundary control is investigated in Section 3.3. We analyse

the parameter dependence of the horizon and the influence of different stage cost. Furthermore, the results are compared with those from the distributed control. Special cases of these results are already published in [4]. In Section 3.4 the linear heat equation with Neumann boundary control is looked at. The main observation is that the stability behaviour is, depending on the choice of the parameters, either similar to the distributed control case or similar to the Dirichlet boundary control case. The special case of the one dimensional heat equation was already published in [5]. In addition, we investigate the influence of different cost functionals. In Section 3.5 we show how to adapt our method to incorporate an additional convection term. The influence of the convection is analysed from a theoretical and a numerical point of view. Afterwards, we compare the method presented in this thesis with an approach developed in [75]. By using the example of the spatially discretized heat equation we discuss the pros and cons of both methods. In Section 3.7 we demonstrate that the described analysis is not only applicable to parabolic, but also to hyperbolic partial differential equations. With the example of the boundary controlled wave equation we analyse the important role of the finite propagation speed in this context. Some special cases of these results are already published in a joint work with Karl Worthmann in [8] and [6]. However, the proofs in this section are completely different so that more general results can be obtained.

- In Chapter 4 we present the algorithms which are implemented in the software package. Section 4.1 is concerned with well known algorithms for solving PDE constrained optimal control problems. Especially, their applicability to MPC is discussed. The possibility to combine Model Predictive Control with Proper Orthogonal Decomposition is demonstrated in Section 4.2. We compare known algorithms as well as new approaches and analyse the advantages and disadvantages. In [77] and [38] the authors suggest to allow different optimization horizons in each MPC step. In Section 4.3 we discuss if these algorithms are suitable for the application on PDEs. Especially, we describe a new approach how multigrid methods can be successfully used in this context.

- In the first part of Chapter 5 we introduce two semilinear PDEs which are used as benchmark problems in Chapter 6. In detail we look at the nonmonoton Schlögl equation and at the nonlinear model of a catalytic rod. Section 5.2 is about the actual implementation of the algorithms in the C++ sourcecode. The focus is on the structure of the object-oriented classes. Furthermore, fragments of the sourcecode are displayed to explain the functionality of the program. A complete example is given in Appendix A. The sourcecode can be found on the enclosed CD-ROM.

- The last Chapter 6 contains the numerical results of this thesis. In Section 6.1 we demonstrate that a real-time realization of the MPC algorithm is possible for the introduced examples in Section 5.1. Afterwards, in Section 6.2 we compare the numerical performance of the optimization algorithms presented in Section 4.1 and the applicability to the MPC algorithm. We investigate the computing time considering several parameters like discretization, optimization horizon, regularization- and reaction parameter. The MPC-POD algorithms introduced in Section 4.2 are analysed in Section 6.3. Finally, we demonstrate the advantages and limitations of the adaptive horizon algorithms presented in Section 6.4.

The main contributions of this thesis are:

- The theoretical analysis of the minimal stabilizing horizon in the MPC algorithm for different classes of PDEs;
- The development of new algorithms and the numerical comparison with known approaches;
- The implementation of the algorithms in a C++ software package;

1 Mathematical Control Theory

In this Chapter we give a short introduction into *mathematical control theory.* Control theory is concerned with the analysis and design of *control systems.* In the first part we present an overview about control systems where we focus on sampled data systems, stability and Lyapunov functions. A detailed introduction into this topic can be found in [86]. For an overview with the emphasis on linear infinite dimensional systems we refer to [26]. In the second part we introduce the concept of *Model Predictive Control* (MPC) with the focus on suboptimality estimates. Our presentation is essentially based on [45].

1.1 Introduction to Control Theory

Control systems

In this section we introduce the concept of *control systems.* The goal is to *control* the *state* of a system such that a desired behaviour is obtained. First, we give a formal definition:

Definition 1.1
Let Y and U be reflexive Banach spaces. A ***control system*** *is a quadruple $\Sigma = (\mathbb{T}, Y, U, f)$ with state space Y, control space U and right hand side $f : \mathcal{D}_f \to Y^*$, where $\mathcal{D}_f$ is a subset of $Y \times U$. For a* ***time set*** *$\mathbb{T} = \mathbb{R}$ the system is called* ***continuous-time*** *control system and the transition map is given by the solution of*

$$\dot{y}_u(t) = f(y_u(t), u(t)), \quad y_u(0) = y_0, \quad t \in \mathbb{R}_0^+, \tag{1.1}$$

where $y_0 \in Y$ denotes the initial value. For $\mathbb{T} = \mathbb{N}_0$ the system is called ***discrete-time*** *control system and the transition map is given by the solution of*

$$y_u(n+1) = f(y_u(n), u(n)), \quad y_u(0) = y_0, \quad n \in N_0. \tag{1.2}$$

The space of control sequences $u : \mathbb{T} \to U$ is denoted by $\mathcal{U}$.

Definition 1.2
The function $y_u(t, y_0)$ of (1.1) *or* (1.2) *emanating from initial value $y_u(0) = y_0 \in Y$ is called* ***solution trajectory*** *of* (1.1) *or* (1.2) *for $t \in \mathbb{T}$.*

It should be mentioned that the choice of the spaces Y and U strongly depends on the investigated PDE and the kind of control. In Section 2.2 we introduce the appropriate function spaces and solution concepts for the PDEs considered in this thesis.

Remark 1.3
We want to mention that the control theory literature generally denotes the state by x and the state space by X. However, we use the notation y and Y while x denotes the independent variable in the PDE. Furthermore, we use the abbreviation $y(\cdot) = y_u(\cdot)$ when $u(\cdot)$ is apparent from the context.

It is important to note that state and control space are arbitrary reflexive Banach spaces. This allows us to deal with ordinary as well as partial differential equations. In control theory we distinguish between *open-loop* and *closed-loop* control systems.

Definition 1.4
Let a control system (1.1) *or* (1.2) *be given. A function $u : \mathbb{T} \to \mathbb{U}$ based on some initial condition y_0 is called an **open-loop** control law. A function $F : \mathbb{Y} \to \mathbb{U}$ is called **closed-loop** or **feedback** control law.*

A schematic representation of both control concepts is displayed in Figure 1.1. The control in the open-loop case only depends on the initial value and the model. The main advantage of this approach is that the computation of an appropriate control can usually be done *offline*.

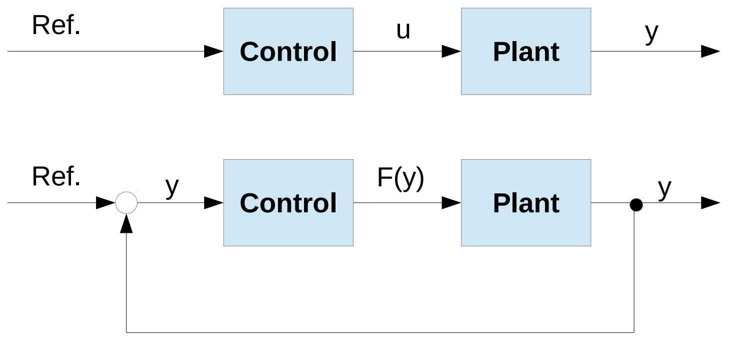


Figure 1.1: Schematic representation of an *open-loop* (above) and a *closed-loop* (below) control system.

However, in practical applications the exact model is generally unknown and perturbations or uncertainties can occur. Thus, the open-loop control can lead to an undesirable behaviour. In contrast to that the feedback control law is able to address these problems, because the control depends on the current state. In Figure 1.1 we observe the main difference between open-loop (above) and closed-loop (below) control: In the closed-loop case the control does not only use information about the reference signal, but also about the currently measured state. Thus, the feedback control is able to correct deviations from the desired behaviour. Since the current control is influenced by the current state the computation of the control sequence has to be done *online*. Depending on the computational burden to determine the feedback this can yield problems.

Obviously, the feedback law formally eliminates the control from the system and we end up with the *continuous-time dynamical system*

$$\dot{y}(t) = f(y(t), F(y(t))) =: g(y(t)), \quad t \in \mathbb{R} \tag{1.3}$$

with a function $g : Y \to Y$. In the discrete case we obtain the so called *discrete-time dynamical system*

$$y(i+1) = f(y(i), F(y(i))) =: g(y(i)), \quad i \in \mathbb{N}_0. \tag{1.4}$$

Next, we give an example for a continuous-time control system by the controlled heat equation.

Example 1.5 (Heat equation)
We look at the linear heat equation

$$\begin{aligned} y_t(x,t) &= \Delta y(x,t) + \mu y(x,t) + u(x,t) && \text{in} \quad \Omega \times (0,\infty) && (1.5a)\\ y(x,t) &= 0 && \text{on} \quad \partial\Omega \times (0,\infty) && (1.5b)\\ y(x,0) &= y_0(x) && \text{in} \quad \Omega && (1.5c) \end{aligned}$$

with domain $\Omega \subset \mathbb{R}^n$, reaction parameter $\mu \in \mathbb{R}$ and initial function $y_0(x)$. We use the notation $y_t := \frac{\partial y}{\partial t}$ for the time derivative and $\Delta := \sum_{i=1}^n \frac{\partial^2}{\partial x_i^2}$ for the Laplacian. The state $y(x,t)$ can be seen as the heat distribution inside the domain. In this example the control $u(x,t)$ acts as a heat source in the whole domain. The task is to choose the control u in such a way that the corresponding temperature distribution y is close to a prescribed temperature.
If we interpret the state as an abstract function we obtain that the state space is given by $y(t) := y(\cdot,t) \in H_0^1(\Omega) = Y$. For the control space we get $u(t) := u(\cdot,t) \in L^2(\Omega) = U$, cf. [91]. A precise definition of the corresponding function spaces as well as the regularity results for this equation are presented in Section 2.2.

In control systems from practical applications the state and control values are often restricted. For the previous example, for instance, it is natural to introduce control bounds, because the capacities for cooling and heating are generally restricted. Therefore, it is reasonable to introduce nonempty subsets $\mathbb{Y} \subset Y$ and $\mathbb{U} \subset U$ which contain the admissible values for state and control, respectively. A suitable choice of control constraints in Example 1.5 are so called *box constraints*

$$\mathbb{U} = \{u \in L^2(\Omega) : u_a \le u \le u_b\} \tag{1.6}$$

with the lower and upper bound $u_a, u_b \in L^2(\Omega)$, respectively. We want to point out that in this thesis state and control constraints do not play an important role.

Especially state constraints are a difficult task in the context of PDE constrained optimization, cf. [52]. In the following we will not pay attention to feasibility issues. Particularly, for discrete time control systems we assume that for each $y \in \mathbb{Y}$ there exists a feasible control $u \in \mathbb{U}$ such that $f(y,u) \in \mathbb{Y}$. This assumption is called *controlled forward invariance* of $\mathbb{Y}$, cf. [45].

Sampled data systems

Although most models in real applications and all systems considered in this thesis are given by continuous time systems, in the context of Model Predictive Control it is reasonable to interpret them as discrete time models. In order to convert a continuous time system (1.1) into a discrete time system (1.2) we use the concept of *sampled data systems.* Starting point is the continuous time system

$$\dot{y}(t) = f(y(t), v(t)) \tag{1.7}$$

with control function $v : \mathbb{R} \to V$, where V denotes the continuous time control space. Note that we change the notation of the control to distinguish between the continuous control v and the discrete control u. We assume that (1.7) has a unique solution $\varphi(t, t_0, y_0, v)$ for all control functions $v \in L^\infty(\mathbb{R}, V)$ and all initial values $y(t_0) = y_0 \in Y$. For the continuous time PDEs investigated in this thesis this requirement is met, see Section 2.2. Next we introduce an equidistant sampling grid $t_0 < t_1 < \cdots < t_N$ with $t_n = nT$, where $T > 0$ denotes the *sampling time.* The idea of *sampling* is to find a discrete time system (1.2) in which the values on the sampling grid coincide with those of the continuous time system, i.e.,

$$\varphi(t_n, t_0, y_0, v) = y_u(n, y_0), \quad n = 0, 1, \ldots, N. \tag{1.8}$$

For the control function $v \in L^\infty([t_0, t_N], V)$ we define the corresponding discrete time control sequence $u(\cdot) \in U^N$ with $U = L^\infty([0,T], V)$ by

$$u(n) := v|_{[t_n, t_{n+1}]}(\cdot + t_n), \quad n = 0, \ldots, N-1. \tag{1.9}$$

Since $u(n)$ is the restriction of v onto the interval $[t_n, t_{n+1}]$, equation (1.8) holds and the trajectories of the discrete and continuous time system coincide on the sampling grid, cf. [45]. The corresponding discrete time system is given by

$$y(n+1) = f(y(n), u(n)) := \varphi(T, 0, y(n), u(n)). \tag{1.10}$$

Remark 1.6
By using the method of sampling we can reproduce every continuous time solution at the sampling times. Motivated by this fact in the remaining chapter we restrict ourselves to discrete time systems. The possibility to rewrite a continuous time system in a discrete way is quite important in our theoretical investigations in Chapter 3.

In the practical implementation it is necessary to use finite dimensional subspaces $U \subset L^\infty([0,T],V)$. A popular choice is the implementation via *zero order hold* where the control $u(n)$ remains constant on each sampling interval $[t_n, t_{n+1}]$ and, thus, we obtain $u(n) \in V$. The algorithmic realization in Chapter 4 is based on this method.

Example 1.7 (Example 1.5 continued)
We apply the sampling method to the continuous time system (1.5)*. The unique solution is denoted by $\varphi(t,0,y_0,u)$. The corresponding sampled data system with sampling time $T > 0$ is recursively defined by*

$$y(n+1) := \varphi(T,0,y(n),u(n)). \tag{1.11}$$

With this definition we obtain $y(n) = y(\cdot, nT) \in H_0^1(\Omega) = Y$, where $y(x,t)$ denotes the solution of (1.5)*. The discrete time control sequence is defined according to* (1.9) *while the control space is given by $u(n) \in U = L^\infty([0,T], L^2(\Omega))$.*

Remark 1.8
We want to point out that an explicit solution of the continuous-time system (1.1) *is generally not available and it is necessary to discretize the system to obtain a numerical approximation of the solution, see Section 2.3.2. Thus, a discrete-time control system arises in a natural way. However, in our theoretical analysis we do not investigate the influence of approximation errors and, thus, we assume that the exact solution is known on the sampling grid. In order to be as close as possible to this assumption in our numerical Chapter 6 we use a tolerance for the ODE solver that is much higher than what is reasonable from a practical point of view.*

Stability of control systems

Our goal in this thesis is to find a feedback law that stabilizes the system at an *equilibrium point.* These points $y^* \in \mathbb{Y}$ are characterized by

$$f(y^*, u^*) = y^* \tag{1.12}$$

for at least one control value $u^* \in \mathbb{U}$. For Example 1.5 it is obvious that the uncontrolled equation ($u \equiv 0$) has the equilibrium $y^* \equiv 0$. Throughout this thesis we focus on stabilizing this equilibrium. In order to look at equilibrium points in more detail we introduce the concept of *stability.* A suitable tool to describe stability for nonlinear systems are so called *comparison functions*, cf. [85].

Definition 1.9
*We define the following classes of **comparison functions**:*

$$\begin{aligned}
\mathcal{K} &:= \{\alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+ \mid \alpha \text{ is continuous and strictly increasing with } \alpha(0) = 0\} \\
\mathcal{K}_\infty &:= \{\alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+ \mid \alpha \in \mathcal{K}, \alpha \text{ is unbounded}\} \\
\mathcal{L} &:= \{\delta : \mathbb{R}_0^+ \to \mathbb{R}_0^+ \mid \delta \text{ is continuous and strictly decreasing with} \lim_{t\to\infty} \delta(t) = 0\} \\
\mathcal{KL} &:= \{\beta : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+ \mid \beta \text{ is continuous}, \beta(\cdot, t) \in \mathcal{K}, \beta(r, \cdot) \in \mathcal{L}\}
\end{aligned}$$

In the next example we present an important representative of the class $\mathcal{KL}$.

Example 1.10
We look at an exponentially decaying function. For an ***overshoot bound*** $C \geq 1$ *and a* ***decay rate*** $\sigma \in (0, 1)$ *we define*

$$\beta(r, n) = C\sigma^n r. \tag{1.13}$$

This function will play a dominant role in our theoretical investigations of parabolic PDEs in Chapter 3.

Next, we use the comparison functions in order to define *asymptotic stability* for discrete-time dynamical system, cf. [45].

Definition 1.11
Let $y^* \in Y$ *be an equilibrium for the discrete-time dynamical system* (1.4)*, i.e.,* $g(y^*) = y^*$*. Then the equilibrium is said to be* ***asymptotically stable*** *if there exists a neighbourhood* $\mathcal{N}(y^*)$ *and a* $\mathcal{KL}$ *function* β *such that for each* $y_0 \in \mathcal{N}(y^*)$ *the state trajectory* $y(n; y_0), n \in \mathbb{N}_0$ *satisfies the inequality*

$$\|y(n; y_0) - y^*\| \leq \beta(\|y_0 - y^*\|, n), \quad n \in N_0. \tag{1.14}$$

Moreover, y^* *is called* ***globally asymptotically stable*** *if* $\mathcal{N}(y^*) = Y$.

In Example 1.5 the equilibrium $y^* \equiv 0$ is globally asymptotically stable for $\mu < \lambda_1$, where λ_1 denotes the smallest eigenvalue of the negative Laplacian, see Section 3.1 for details. For $\mu > \lambda_1$ this equilibrium is unstable.
Now, we introduce the concept of *Lyapunov functions* which is an important tool to prove stability of closed-loop control systems, cf. [45]. Lyapunov functions are often interpreted as the energy of the system. Thus, they characterize the equilibrium as the state where no energy in the system is present.

Definition 1.12
Let $y^* \in Y$ *be an equilibrium point for the dynamical system* (1.4) *and* $\mathbb{Y} \subset Y$ *a subset of the state space. A function* $V : \mathbb{Y} \to \mathbb{R}_0^+$ *is called* ***Lyapunov function*** *on* $\mathbb{Y}$ *if the following conditions are satisfied:*

- *There exist functions* $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ *such that*

$$\alpha_1(\|y - y^*\|) \leq V(y) \leq \alpha_2(\|y - y^*\|) \tag{1.15}$$

  holds for all $y \in \mathbb{Y}$.

- *There exists a function* $\alpha_V \in \mathcal{K}$ *such that*

$$V(g(y)) \leq V(y) - \alpha_V(\|y - y^*\|) \tag{1.16}$$

  holds for all $y \in \mathbb{Y}$ *with* $g(y) \in \mathbb{Y}$.

Moreover, if $\mathbb{Y} = Y$, then $V(\cdot)$ is called ***global Lyapunov function****.*

Lyapunov functions are well suited to analyse stability of dynamical systems, because (under suitable conditions) the existence of a Lyapunov function is a sufficient condition for stability, cf. [45]. Next, we transfer the concept of stability to discrete-time control systems.

Definition 1.13
Let $y^* \in Y$ be an equilibrium for the discrete-time control system (1.2), i.e., $f(y^*, u^*) = y^*$. Then the equilibrium is said to be ***asymptotically controllable*** *if there exists a neighbourhood $\mathcal{N}(y^*)$ and a $\mathcal{KL}$ function β such that for each $y_0 \in \mathcal{N}(y^*)$ there exist a control sequence $u \in U^\infty$ that guarantees*

$$\|y_u(n; y_0) - y^*\| \leq \beta(\|y_0 - y^*\|, n), \quad n \in N_0. \tag{1.17}$$

Moreover, y^* is called ***globally asymptotically controllable*** *if $\mathcal{N}(y^*) = Y$.*

Remark 1.14
We want to point out that there exists a wide range of different concepts concerning controllability. Especially for infinite dimensional systems we distinguish between ***approximately controllable****,* ***exactly controllable*** *and* ***null controllable****, cf. [25]. An introduction into controllability of linear infinite dimensional systems can be found in [26]. A detailed overview with the focus on parabolic and hyperbolic PDEs is given in [102]. Although these concepts are not in the focus of this thesis, we want to point out that the considered PDEs in the ensuing chapters possess a property that is called* ***null controllable in time*** *T. This means that for each initial function y_0 there exists a control u and a time $T > 0$ such that $y_u(T) = 0$, where the corresponding function spaces depend on the considered PDE. It can be shown that the linear heat equation with finite domain is null controllable for arbitrarily small values of T, cf. [102]. Under suitable conditions on the nonlinearity the same holds true for the semilinear heat equation, cf. [33]. In contrast to that the boundary controlled wave equation is null controllable in time $T > 0$, where T depends on the domain and, thus, cannot be arbitrarily small. This fact is due to the finite propagation speed of the wave and will play an important role in our investigations in Section 3.7. We want to mention that the analysis concerning controllability becomes much more complicated if state or control constraints are incorporated.*

1.2 Model Predictive Control

In this section we introduce the concept of Model Predictive Control (MPC). In recent years the idea to apply MPC to infinite dimensional systems becomes popular. The possibility to solve large scale optimal control problems in a reasonable time plays a role in this context. Since there is much literature concerning this topic we only want to give a short overview. An important application of MPC is given by the control of fluid flows. This subject is strongly related to the control of the

Navier-Stokes equation and can be found, e.g., in [16] and [37]. In [30], [28] and [29] the authors apply MPC to parabolic PDEs which originate from process engineering. A relatively new approach is given by the combination of predictive control with model reduction techniques. In this context the reduction technique Proper Orthogonal Decomposition plays an important role and the combination with MPC is investigated e.g. in [3], [54] and [37]. In Section 4.2 we will use this concept as well. A stability analysis of the MPC method for a general class of abstract PDEs is presented in [57], where the focus is on control Lyapunov functions. Finally, we want to mention that *instantaneous control* can be interpreted as an MPC variant with the shortest possible horizon. (Note that the use of the term *instantaneous control* is not unique, see the discussion in Section 3.7.2.) This method was successfully applied to flow control problems, cf. [50], and to the control of the Fokker-Planck equation, cf. [10].

In the following the aim is to *stabilize* the equilibrium y^* of the discrete-time control system with a feedback law $F : \mathbb{Y} \to \mathbb{U}$. *Stabilize* means in this context that we want to find a feedback, such that the corresponding closed-loop system is *asymptotically stable.* In order to measure the distance of the current state y to the desired equilibrium we introduce the continuous *stage costs* (also called *running costs*) $l : Y \times U \to \mathbb{R}_0^+$. It is often reasonable to penalize the control effort in the current step as well. In practical applications, for instance, one should have in mind that the insertion of high energy can be expensive. Furthermore, for theoretical and computational reasons it is useful to take the control costs into account. We chose stage costs which are nonnegative and uniquely identify the equilibrium y^*, i.e.,

$$l(y^*, u^*) = 0 \quad \text{and} \quad l(y,u) > 0 \quad \text{for all } y \in Y, u \in U \text{ with } y \neq y^*. \tag{1.18}$$

In this thesis we always assume $y^* = 0$ and $u^* = 0$. Therefore, in the remaining chapter we restrict ourselves to consider this case.
For Example 1.5 suitable stage costs are given by

$$l(y(n), u(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot, nT)\|^2_{L^2(\Omega)}, \tag{1.19}$$

i.e., we penalize the state as well as the control in the $L^2(\Omega)$-norm, cf. [91]. The constant $\lambda > 0$ denotes a so called regularization or Tikhonov parameter.

Definition 1.15
Let a discrete time control system $(\mathbb{T}, X, U, f)$ be given. Then we define the ***cost functional*** $J_N : Y \times U^N \to \mathbb{R}_0^+$ *by*

$$J_N(y_0, u(\cdot)) := \sum_{n=0}^{N-1} l(y_u(n, y_0), u(n)) \tag{1.20}$$

with ***optimization horizon*** $N \in \mathbb{N}_{\geq 2}$ *and* $y_0 \in Y$. *The corresponding* ***optimal value function*** $V_N : Y \to \mathbb{R}_0^+$ *is denoted by*

$$V_N(y_0) := \inf_{u(\cdot) \in \mathbb{U}^N} J_N(y_0, u(\cdot)). \tag{1.21}$$

A control sequence $u^*(\cdot) \in \mathbb{U}^N$ is called ***optimal control sequence*** *for the finite horizon problem, if*

$$V_N(y_0) = J_N(y_0, u^*(\cdot)) \tag{1.22}$$

holds. Analogously, we define the ***infinite horizon cost functional***

$$J_\infty(y_0, u(\cdot)) := \sum_{n=0}^{\infty} l(y_u(n, y_0), u(n)) \tag{1.23}$$

with corresponding optimal value function

$$V_\infty(y_0) := \inf_{u(\cdot)\in\mathbb{U}^\infty} J_\infty(y_0, u(\cdot)). \tag{1.24}$$

A control sequence $u^*(\cdot) \in \mathbb{U}^\infty$ is called ***optimal control sequence*** *for y_0, if*

$$V_\infty(y_0) = J_\infty(y_0, u^*(\cdot)) \tag{1.25}$$

holds. The corresponding trajectory $y_u^*(\cdot, y_0)$ is called ***optimal trajectory****.*

In the following we assume that $V_\infty(y_0) < \infty$. By using *Bellman's optimality principle*, cf. [86], one can derive an *optimal feedback law.*

Theorem 1.16
For an initial value $y_0 \in \mathbb{Y}$ we define the ***optimal feedback law***

$$u^*(n) := F(y_u(n, y_0)) := \arg\min_{v\in\mathbb{U}}\{V_\infty(f(y_u(n, y_0), v)) + l(y_u(n, y_0), v)\}. \tag{1.26}$$

Then $u^*(\cdot)$ minimizes the infinite horizon cost functional (1.23)*. The resulting feedback is also called* ***infinite horizon optimal control law****. Furthermore, it holds that*

$$J_\infty(y_0, F) = V_\infty(y_0). \tag{1.27}$$

The proof can be found e.g. in [45]. Note that we need to assume that the minimum in (1.26) actual exists. Although Theorem 1.16 provides a possibility to obtain a feedback law, for practical applications this method is not suitable, because it requires the knowledge of $V_\infty(y)$. Thus, we have to solve an infinite horizon optimal control problem in each step, which is even for finite dimensional systems a difficult task. The approach to solve this optimization problem for a high dimensional system via *Hamilton-Jacobi-Bellman* equations seems to be for the foreseeable future not appropriate, cf. [62]. However, there exist new approaches where this method is combined with a model reduction technique, cf. [65] and [60]. For the special case of a linear system without state and control constraints the solution of the infinite horizon optimization problem can be obtained by solving *algebraic Riccati equations*, cf. [15].

In order to overcome the problem of solving an optimal control problem on an infinite horizon we use the concept of *Model Predictive Control (MPC)*. The idea is to substitute the infinite horizon problem by an iterative sequence of finite horizon optimization problems. For this purpose we define the **Optimal Control Problem with horizon N (OCP_N)**

$$\min_{u(\cdot)\in\mathbb{U}^N} J_N(y_0,u(\cdot)) := \sum_{n=0}^{N-1} l(y_u(n,y_0),u(n)) \tag{1.28a}$$

subject to

$$y_u(0,y_0)=y_0, \quad y_u(n+1,y_0)=f(y_u(n,y_0),u(n)). \tag{1.28b}$$

It should be mentioned that in our notation the shortest reasonable horizon is given by $N=2$. (The motivation for this notation is that the optimal control problem with horizon N leads to a control sequence with N elements.)

In Figure 1.2 we display one step of the Basic MPC Algorithm 1.1. The previously applied controls and the trajectory in the past are coloured in black. In the first step we measure the current state $y(n)$ and set $y_0 := y(n)$. Afterwards, we solve the finite horizon optimal control problem (1.28) to obtain the open loop optimal control sequence $u^*(\cdot)$ (blue) and the corresponding open loop trajectory $y_u(\cdot)$ (red). In the last step we apply the *MPC feedback* $F_N(y(n)) := u^*(0)$ to the system and get the next state $y_{F_N}(n+1)$. This means that the first element of the open loop control sequence provides the feedback law in this step. By shifting the horizon forward in time and repeating this procedure we obtain the MPC feedback on an infinite time horizon, see Figure 1.2 (below). The resulting *MPC closed loop trajectory* is given by

$$y_{F_N}(n+1)=f(y_{F_N}(n),F_N(y_{F_N}(n))). \tag{1.29}$$

In order to rate the quality of the MPC feedback we look at the cost functional on the infinite horizon

$$V_\infty^{F_N}(y_0) := \sum_{n=0}^{\infty} l(y_{F_N}(n),F_N(y_{F_N}(n))). \tag{1.30}$$

Obviously, the cost produced by the MPC feedback is always higher than or equal to the cost obtained by the optimal feedback, i.e., we have $V_\infty^{F_N}(y_0) \geq V_\infty(y_0)$.

```
1 for $n = 0, 1, 2, \ldots$ do
2     Measure the state $y(n) \in Y$;
3     Set $y_0 := y(n)$, solve the optimal control problem (1.28) and denote the
      obtained optimal control sequence by $u^*(\cdot) \in \mathbb{U}^N(y_0)$;
4     Define the MPC-feedback $F_N(y(n)) := u^*(0) \in \mathbb{U}$ and use this control in
      the next sampling period;
5 end
```

Algorithm 1.1: Basic MPC Algorithm

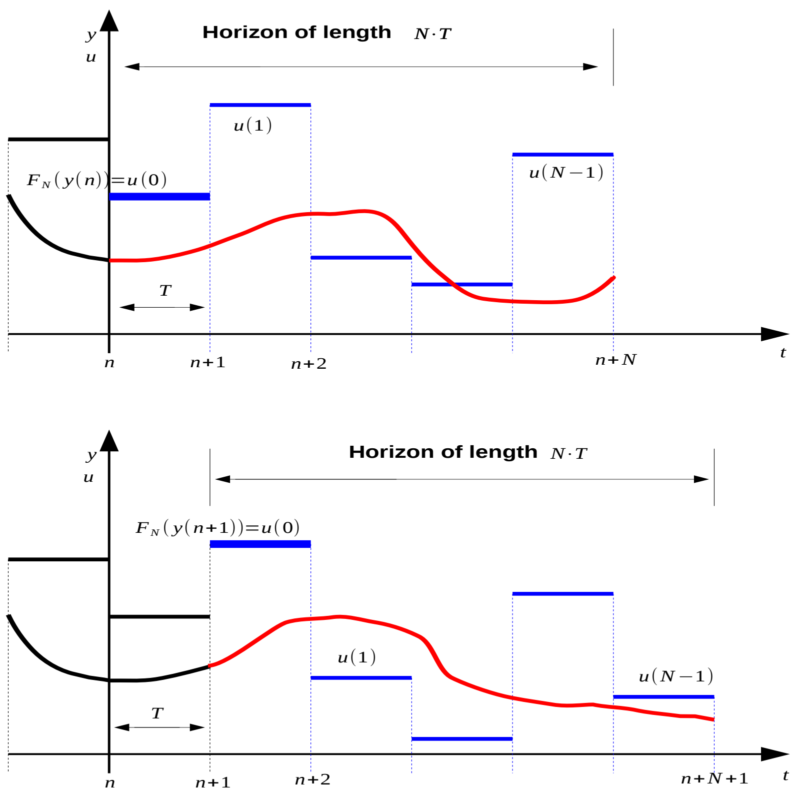


Figure 1.2: Illustration of the basic MPC Algorithm 1.1: The figures above display the predicted control (blue) and the corresponding state trajectory (red) in the current MPC step. The first element of the control sequence is implemented as a feedback, i.e., $F_N(y(n)) = u(0)$. Afterwards, the horizon is shifted and the procedure is repeated (below).

In the following we want to answer two important questions which arise in the context of MPC:

- Does the MPC feedback stabilize the closed loop system (1.29) for a given optimization horizon N?
- How good is the MPC feedback with horizon N compared to the optimal feedback (1.26)?

Remark 1.17
We want to point out that in the MPC literature stabilizing terminal constraints or terminal costs are often introduced in the optimization problem (1.28) *to prove stability of the MPC closed loop system, cf. [80]. In contrast to that we use the MPC scheme without artificial constraints, because these problems are much easier to handle from the numerical point of view. Especially, incorporating terminal*

constraints in the PDE constrained optimization problem is a difficult task and only very few literature about this topic is available. In [57] the authors present a method to construct control Lyapunov functions as terminal costs for a class of semilinear PDEs. However, for more complicated infinite dimensional systems the construction of these functions seems to be a delicate problem. Finally we want to mention that unconstrained MPC schemes are often used in industrial applications, cf. [79].

In order to answer the question concerning the quality of the MPC feedback we use the concept of the *relaxed Lyapunov inequality*, cf. [43]. The proof of the following theorem can be found in [45].

Theorem 1.18
Let $F_N : \mathbb{Y} \to U$ be a feedback law and $V_N : \mathbb{Y} \to \mathbb{R}_0^+$ a function which satisfies the ***relaxed Lyapunov inequality***

$$V_N(f(y, F_N(y))) \leq V_N(y) - \alpha l(y, F_N(y)) \tag{1.31}$$

for some $\alpha \in (0, 1]$ and all $y \in \mathbb{Y}$. Then the ***suboptimality estimate***

$$V_\infty^{F_N}(y) \leq \frac{1}{\alpha} V_\infty(y) \tag{1.32}$$

holds for all $y \in \mathbb{Y}$.
If, in addition, $\mathcal{K}_\infty$-functions $\underline{\eta}$, $\overline{\eta}$ exist satisfying

$$\underline{\eta}(\|y\|) \leq l^*(y) := \min_{u \in U} l(y, u) \quad \textit{and} \quad V_N(y) \leq \overline{\eta}(\|y\|) \tag{1.33}$$

for all $y \in \mathbb{Y}$, then the closed loop system (1.29) is asymptotically stable and, thus, in particular converges to the equilibrium $y^* = 0$ as $n \to \infty$.

The parameter α is called *suboptimality degree* because it measures the performance of the MPC feedback compared to the optimal feedback. An α value close to 1 indicates that the MPC feedback is nearly as good as the optimal feedback. For a small $\alpha > 0$ we can guarantee stability, however, with possibly bad performance. If $\alpha \leq 0$, no statement about the stability of the closed loop system is possible.
The next goal is to present a possibility to determine the value of α. For this purpose we introduce the *exponential controllability with respect to the stage costs l.*

Definition 1.19
The system (1.2) is called ***exponentially controllable*** *with respect to the stage costs l if there exist an overshoot bound $C \geq 1$ and a decay rate $\sigma \in (0, 1)$ such that for each $y \in Y$ there exists $u_y \in \mathcal{U}$ satisfying*

$$l(y_{u_y}(n; y), u_y(n)) \leq C\sigma^n \min_{u \in U} l(y, u) = C\sigma^n l^*(y) \tag{1.34}$$

for all $n \in \mathbb{N}_0$.

It is important to note that we do not require that $u_y \in \mathcal{U}$ is in any sense optimal. By using the exponential controllability we can formulate the following theorem which forms the basis of our analysis in Chapter 3. The proof can be found in [46].

Theorem 1.20 (Stability Theorem)
Let the controllability condition (1.34) *holds with overshoot constant $C \geq 1$ and decay rate $\sigma \in (0, 1)$. Furthermore, let the prediction horizon N be chosen such that the stability condition*

$$\alpha_N := 1 - \frac{(\gamma_N - 1)\prod_{i=2}^{N}(\gamma_i - 1)}{\prod_{i=2}^{N}\gamma_i - \prod_{i=2}^{N}(\gamma_i - 1)} > 0 \tag{1.35}$$

holds with $\gamma_i := C\sum_{n=0}^{i-1}\sigma^n$. Then, the relaxed Lyapunov Inequality (1.31) *holds with $\alpha = \alpha_N$ for each $y \in Y$ and, consequently, the suboptimality estimate*

$$V_\infty^{F_N}(y) \leq \frac{1}{\alpha_N}V_\infty(y), \qquad y \in Y. \tag{1.36}$$

If, in addition, $\mathcal{K}_\infty$-functions $\underline{\eta}$, $\overline{\eta}$ exist satisfying

$$\underline{\eta}(\|y\|) \leq l^*(y) \leq \overline{\eta}(\|y\|) \tag{1.37}$$

for all $y \in Y$, then the closed loop system (1.29) *is asymptotically stable and thus in particular converges to the equilibrium $y^* = 0$ as $n \to \infty$.*

Note that condition (1.37) is always satisfied for the quadratic stage costs considered in this thesis. For the stage cost (1.19) we have

$$l^*(y) = \min_{u \in U} l(y, u) = \frac{1}{2}\|y\|^2_{L^2(\Omega)} \tag{1.38}$$

and, thus, we can choose $\underline{\eta}(r) = \overline{\eta}(r) = \frac{1}{2}r^2$.

Remark 1.21
It should be mentioned that $\alpha_N > 0$ is a sufficient condition to guarantee stability of the MPC closed loop system. For $\alpha_N \leq 0$ no statement about the stability behaviour of a concrete system is possible. Theorem 1.20 is tight in the following sense: If $\alpha_N < 0$, then there exist a control system (1.2) *and stage cost l such that the exponential controllability condition* (1.34) *holds, but the MPC closed loop system* (1.29) *is not asymptotically stable. The proof is given in [45].*

The choice of an appropriate horizon is an important task in the construction of an MPC controller: On the one hand a too short horizon can yield an unstable behaviour or a poor performance. On the other hand a large horizon leads to an optimization problem which requires much more computing time.
In order to investigate α_N in more detail we visualize formula (1.35) by using the concept of *stability regions* introduced by [94]. The value of α_N only depends on

the exponential constants C, σ and on the optimization horizon N. For a given suboptimality degree $\bar{\alpha}$ we display the regions in the (C, σ)-plane where we have $\alpha_N \geq \bar{\alpha}$ for different optimization horizons N. Since we are especially interested in the *minimal stabilizing horizon*, we consider $\bar{\alpha} = 0$. In Figure 1.3 we observe that the required horizon N is small if the overshoot bound C is close to 1 or the decay rate σ close to 0. However, it can be seen that the influence of both constants is very different: For a fixed value of σ it is possible to obtain stability with the shortest reasonable horizon $N = 2$ by reducing the value of C. For a fixed value of C this is generally impossible, even for arbitrarily small values of the decay rate σ.
This observation plays an important role in our theoretical analysis in Chapter 3. If

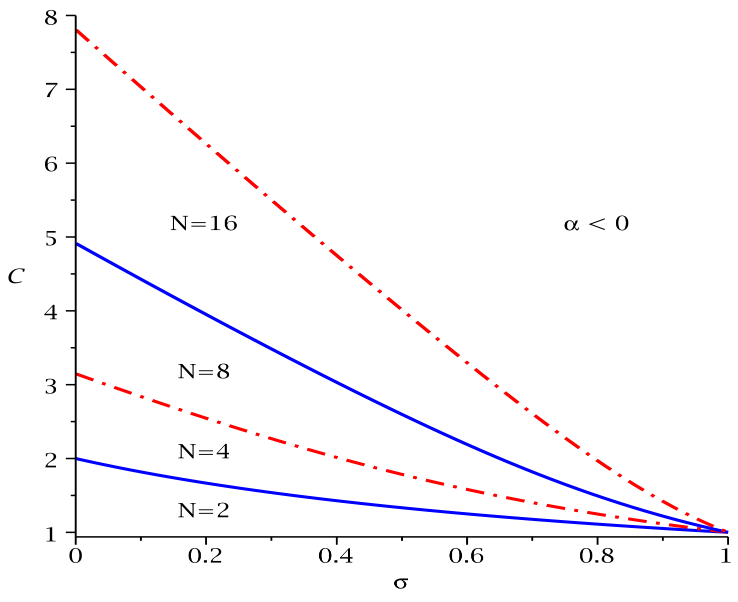


Figure 1.3: Stability areas for different optimization horizons N in dependence of C and σ

one can use the stage cost l as *design variable* this fact can be exploited to construct an MPC controller which only requires a short horizon. The idea is to reduce the value of the overshoot bound C by a subtle design of l, see Chapter 3 for details.

Remark 1.22
It should be noted that there are further approaches for the stabilization of PDEs beside MPC. Well known methods, which are originally developed for finite dimensional systems, are the concepts PID control and ***pole placement****, cf. [90]. The application of* ***Lyapunov based techniques*** *to nonlinear hyperbolic and parabolic PDEs can be found in [23]. A further approach to stabilize infinite dimensional systems is given by the* ***Gramian technique****. In [25] this method was successfully applied to rotating body-beam systems and to the Euler equation from fluid dynamics. In recent years the concept of* ***backstepping*** *has become popular. The main*

advantage is that the stabilizing feedback is often given in an explicit way. (We will use this property in Chapter 3.) However, the main advantage of the MPC approach is that state and control constraints can be incorporated in contrast to most other methods.

2 Optimal Control of PDEs

This chapter deals with the analysis of PDE constrained optimal control problems and with the discretization of the corresponding PDEs. In the first section we introduce an abstract optimization problem in a Banach space and recapitulate the first order necessary conditions. The presentation essentially follows [52]. In Section 2.2 we present some existence and regularity results for the optimal control problems investigated in this thesis. The main theorems concerning the semilinear heat equation can be found in [91]. Discretization schemes for PDEs are introduced in Section 2.3. For the spatial discretization we focus on the Galerkin method. In this context we discuss the Finite Element Method (FEM), cf. [42], and the model reduction technique Proper Orthogonal Decomposition (POD), cf. [96]. We close this chapter by presenting a simple example for the time discretization: the semi-implicit Euler method. It should be mentioned that there is a wide range of literature on the topics considered in this chapter and it is far out of the scope of this thesis to give an overview. We will also not introduce all required concepts from functional analysis.

2.1 Banach Space Optimization

In this section we look at an optimal control problem in the following general form

$$\min_{y\in Y, u\in U} J(y,u) \quad \text{subject to} \quad e(y,u)=0, \quad u\in\mathbb{U} \tag{2.1}$$

with the objective function $J : Y\times U\to\mathbb{R}$ and the state equation $e : Y\times U\to Z$. The Banach spaces U, Y and Z are assumed to be reflexive. In the context of our PDE constrained optimal control problem $J(y,u)$ denotes the cost functional (also called objective function), $e(y,u)=0$ represents the PDE and the admissible set of control values $\mathbb{U}$ is given by pointwise box constraints. As already said in Chapter 1 state constraints are not the focus of this thesis. In the context of PDE constrained optimization the theoretical analysis as well as the numerical implementation of state constraints are much more involved compared to the ODE case, cf. [52].
First, we require some definitions from optimal control theory:

Definition 2.1

1. *A state-control pair* $(\bar y,\bar u)\in Y\times\mathbb{U}$ *is called* ***optimal*** *for* (2.1) *if and only if* $e(\bar y,\bar u)=0$ *and*

$$J(\bar y,\bar u)\le J(y,u) \quad \textit{for all} \quad (y,u)\in Y\times\mathbb{U},\ e(y,u)=0. \tag{2.2}$$

2. *A functional $F : X \to \mathbb{R}$ is called* ***weakly lower semicontinuous*** *if and only if*
$$x_k \rightharpoonup x \quad \Rightarrow \quad \liminf_{k\to\infty} F(x_k) \geq F(x), \tag{2.3}$$
where we use the notation $x_k \rightharpoonup x$ for the weak convergence.

3. *The operator $F : U \subset X \to Y$ is called* ***Fréchet differentiable*** *(F-differentiable) at $x \in U$ if there exists a bounded linear operator $F' \in \mathcal{L}(X, Y)$ such that*
$$\lim_{\|h\|_X \to 0} \frac{1}{\|h\|_X} \|F(x+h) - F(x) - F'(x)h\|_Y = 0. \tag{2.4}$$

We use the notation $\langle \cdot, \cdot \rangle_{V^*,V}$ for the duality pairing in the Banach space V and $\langle \cdot, \cdot \rangle_V$ for the scalar product if V is a Hilbert space. The following assumptions are required to prove existence of an optimal pair $(\bar{y}, \bar{u})$.

Assumption 2.2

1. *$\mathbb{U} \subset U$ is convex, nonempty, bounded and closed.*
2. *The state equation $e(y, u) = 0$ has a bounded unique solution operator $u \in \mathbb{U} \mapsto y(u) \in Y$.*
3. *$(y, u) \in Y \times U \mapsto e(y, u) \in Z$ is continuous under weak convergence.*
4. *J is weakly lower semicontinuous in y and u.*

The proof of the following existence theorem can be found in [52]:

Theorem 2.3
Let Assumption 2.2 hold. Then problem (2.1) *has an optimal solution $(\bar{y}, \bar{u})$.*

Note that the first assumption is always satisfied in the case of box constraints. The second assumption is quite important for the theoretical analysis as well as for the numerical algorithms. By using the solution operator $y = y(u) =: S(u)$, where S is the so called *control to state operator*, we can eliminate the constraints $e(y, u) = 0$ from the optimization problem. This leads to the *reduced cost functional* $\hat{J}(u) := J(S(u), u)$ and the reduced optimization problem
$$\min_{u\in\mathbb{U}} \hat{J}(u) = \min_{u\in\mathbb{U}} J(S(u), u). \tag{2.5}$$

In the context of optimal control this procedure is very reasonable because there is the independent control input u and the state variable y, which depends on the control. The reduced control problem is the starting point for the optimization algorithms presented in Section 4.1. In Chapter 4 we also need the ability of computing the derivatives of the reduced cost functional. Therefore, we assume that J and e are continuously F-differentiable and there exists a unique solution operator $u \in U \mapsto y(u) \in Y$. If in addition $e_y(y(u), u) \in \mathcal{L}(Y, Z)$ is continuously invertible, the implicit function theorem ensures that $y(u)$ is continuously differentiable. Differentiating the state equation with respect to u yields $e_y(y(u),u)y'(u)+e_u(y(u),u) = 0$ and, thus,

$$y'(u) = -e_y(y(u),u)^{-1}e_u(y(u),u). \tag{2.6}$$

By using this equation and after some calculations, cf. [52], we obtain the well known representation of the first derivative of the reduced cost functional

$$\hat{J}'(u) = e_u(y(u),u)^* p(u) + J_u(y(u),u) \tag{2.7}$$

where the *adjoint state* $p = p(u) \in Z^*$ satisfies the *adjoint equation*

$$e_y(y(u),u)^* p = -J_y(y(u),u). \tag{2.8}$$

Thus, the derivative $\hat{J}'(u)$ can be computed by the following algorithm:

- Determine the adjoint state by solving the adjoint equation (2.8).
- Compute $\hat{J}'(u)$ via (2.7).

Note that in Chapter 4 we will also use the gradient $\nabla\hat{J}$, which is the Riesz representation of the derivative $\hat{J}'$. However, in the literature this notation is not unique.

In order to derive a representation of the second derivative it is reasonable to introduce the so called *Lagrange function* $L : Y \times U \times Z^* \to \mathbb{R}$

$$L(y,u,p) = J(y,u) + \langle p, e(y,u)\rangle_{Z^*,Z}. \tag{2.9}$$

The Lagrange function is also a useful tool to write the necessary optimality conditions in a compact form, see Theorem 2.4. Since we are interested in the second derivative of the reduced functional, we assume that J and e are twice continuously F-differentiable. The derivation of the following formula can be found in [52]

$$\begin{aligned}\hat{J}''(u) =& y'(u)^* L_{yy}(y(u),u,p(u))y'(u) + y'(u)^* L_{yu}(y(u),u,p(u)) \\ &+ L_{uy}(y(u),u,p(u))y'(u) + L_{uu}(y(u),u,p(u)).\end{aligned} \tag{2.10}$$

In Chapter 4.1 we will explain that it is generally not necessary to compute the whole operator $\hat{J}''(u)$, but only operator-vector-products $\hat{J}''(u)s$. By using formula (2.10) and the representation of $y'(u)$ (2.6), we obtain the following algorithm to calculate these products, cf. [52]:

- Compute the sensitivity

$$e_y(y(u),u)\delta_s y = -e_u(y(u),u)s$$

- Compute

$$\begin{aligned}h_1 &= L_{yy}(y(u),u,p(u))\delta_s y + L_{yu}(y(u),u,p(u))s \\ h_2 &= L_{uy}(y(u),u,p(u))\delta_s y + L_{uu}(y(u),u,p(u))s\end{aligned}$$

- Compute
$$e_y(y(u),u)^*h_3 = -e_u(y(u),u)^*h_1$$
and set $\hat{J}''(u)s = h_2 + h_3$.

Finally, we present the important first order necessary conditions for the abstract optimization problem (2.1). The proof can be found in [52].

Theorem 2.4
Let $(\bar{y},\bar{u})$ be an optimal solution of the problem (2.1) and let Assumption 2.2 hold. Furthermore we assume that J and e are continuously F-differentiable and that $e_y(y(u),u) \in \mathcal{L}(Y,Z)$ has a bounded inverse for all $u \in \mathbb{U}$. Then there exists an adjoint state (or Lagrange multiplier) $p \in Z^*$ such that the following optimality conditions hold

$$e(\bar{y},\bar{u}) = 0, \tag{2.11a}$$
$$e_y(\bar{y},\bar{u})^*p = -J_y(\bar{y},\bar{u}), \tag{2.11b}$$
$$\langle J_u(\bar{y},\bar{u}) + e_u(\bar{y},\bar{u})^*p, u-\bar{u}\rangle_{U^*,U} \geq 0 \quad \forall u \in \mathbb{U}. \tag{2.11c}$$

Using the Lagrange function (2.9) we can write (2.11) in the compact form

$$L_p(\bar{y},\bar{u},p) = e(\bar{y},\bar{u}) = 0, \tag{2.12a}$$
$$L_y(\bar{y},\bar{u},p) = 0, \tag{2.12b}$$
$$\langle L_u(\bar{y},\bar{u},p), u-\bar{u}\rangle_{U^*,U} \geq 0 \quad \forall u \in \mathbb{U}. \tag{2.12c}$$

We close this section with a short discussion of the variational inequality (2.11c). For this purpose we again address the reduced control problem. The following theorem presents an alternative representation of (2.11c) in the case of box constraints by introducing Lagrange multipliers. The proof can be found in [52].

Theorem 2.5
Let $U = L^2(\Omega), u_a, u_b \in L^2(\Omega), u_a \leq u_b$ and $\mathbb{U}$ be given by

$$\mathbb{U} = \{u \in L^2(\Omega) : u_a \leq u \leq u_b\}. \tag{2.13}$$

Then the following conditions are equivalent

- $\bar{u} \in \mathbb{U}$,
$$\langle J_u(y(\bar{u}),\bar{u}) + e_u(y(\bar{u}),\bar{u})^*p, u-\bar{u}\rangle_U = \langle \nabla\hat{J}(\bar{u}), u-\bar{u}\rangle_U \geq 0 \quad \forall u \in \mathbb{U}. \tag{2.14}$$

- *There are Lagrange multipliers $\mu_b, \mu_a \in U^* = L^2(\Omega)$ with*

$$\nabla J(\bar{u}) + \mu_b - \mu_a = 0 \tag{2.15a}$$
$$\mu_a \geq 0, \quad u_a - \bar{u} \leq 0, \quad \langle \mu_a, u_a - \bar{u}\rangle_U = 0 \tag{2.15b}$$
$$\mu_b \geq 0, \quad \bar{u} - u_b \leq 0, \quad \langle \mu_b, \bar{u} - u_b\rangle_U = 0. \tag{2.15c}$$

2.2 Optimal Control of PDEs

In this section we start with the definition of the required function spaces. A detailed introduction can be found in [32]. Especially, the precise definition of a Lipschitz domain and weak derivatives can be found there. Furthermore, we assume that the definition of the Lebesgue spaces $L^p(\Omega)$ with the corresponding L^p-norm

$$\begin{aligned} \|u\|_{L^p(\Omega)} &= \left(\int_\Omega |u(x)|^p \, dx \right)^{1/p}, \qquad p \in [1, \infty), \\ \|u\|_{L^\infty(\Omega)} &= \operatorname*{ess\,sup}_{x \in \Omega} \|u(x)\| \end{aligned}$$

is known. Functions which only differ on a set of measure zero belong to the same equivalence class. Next, we introduce the so called *Sobolev spaces*

Definition 2.6
Let $\Omega \in \mathbb{R}^n$ with boundary $\Gamma := \partial\Omega$ be open. For $k \in \mathbb{N}_0, p \in [1, \infty]$, we define the ***Sobolev space*** *$W^{k,p}(\Omega)$ by*

$$W^{k,p}(\Omega) := \{u \in L^p(\Omega) : u \, \textit{has weak derivatives} \, D^\alpha u \in L^p(\Omega) \, \textit{for all} \, |\alpha| \leq k\}$$

with the associated norm

$$\begin{aligned} \|u\|_{W^{k,p}(\Omega)} &:= \left(\sum_{|\alpha| \leq k} \|D^\alpha u\|^p_{L^p(\Omega)} \right)^{1/p}, \qquad p \in [1, \infty) \\ \|u\|_{W^{k,\infty}(\Omega)} &:= \sum_{|\alpha| \leq k} \|D^\alpha u\|_{L^\infty(\Omega)}. \end{aligned}$$

- *For the important case $p = 2$ we define $H^k(\Omega) := W^{k,2}(\Omega)$.*
- *The closure of $C_0^\infty(\Omega)$ in $W^{k,p}(\Omega)$ is denoted by $W_0^{k,p}(\Omega)$.*
- *The dual space of the Hilbert space $H_0^1(\Omega)$ is denoted by $H^{-1}(\Omega)$.*

In the context of parabolic PDEs it is reasonable to interpret the solution $y(x, t)$ as an *abstract function* (also called *vector-valued function*) $y(t) = y(\cdot, t) \in X$ with values in the Banach space X.

Definition 2.7
For $1 \leq p < \infty$ we denote by $L^p(a, b; X)$ the Banach space of ***abstract functions*** *$y : [a, b] \to X$ with the property*

$$\int_a^b \|y(t)\|_X^p \, dt < \infty. \tag{2.16}$$

The corresponding norm is given by

$$\|y\|_{L^p(a,b;X)} := \left(\int_a^b \|y(t)\|_X^p \, dt \right)^p. \tag{2.17}$$

Analogously, we define the Banach space $L^\infty(a,b;X)$ by

$$\|y\|_{L^\infty(a,b;X)} := \operatorname{ess}\sup_{t\in[a,b]} \|y(t)\|_X < \infty. \tag{2.18}$$

For a Hilbert space V we introduce the space

$$W(0,T;V) := \{y : y \in L^2(0,T;V), y_t \in L^2(0,T;V^*)\} \tag{2.19}$$

with the associated norm

$$\|y\|_{W(0,T;V)} := \left(\int_0^T (\|y(t)\|_V^2 + \|y_t(t)\|_{V^*}^2)\, dt\right). \tag{2.20}$$

For the important case $V = H^1(\Omega)$ we write

$$W(0,T) := \{y : y \in L^2(0,T;H^1), y_t \in L^2(0,T;H^{-1})\}.$$

Next, we present the *Friedrichs-Poincaré inequality* (also known an Friedrichs inequality). The proof can be found in [24].

Theorem 2.8
Let $\Omega \subset \mathbb{R}^n$ denote a bounded Lipschitz domain and let $\Gamma_0 \subset \Gamma$ be a measurable set such that $|\Gamma_0| > 0$. Then there exists a constant $c > 0$ such that

$$\int_\Omega |y|^2\, dx \le c \int_\Omega |\nabla y|^2\, dx \tag{2.21}$$

for all $y \in H^1(\Omega)$ that satisfy $y = 0$ on Γ_0.

This useful inequality will play an important role in our theoretical investigations. Therefore, in Chapter 3 we will use a version of this inequality where the best Friedrichs constant is explicitly given.

Now, we have the necessary tools to analyse the semilinear heat equation with a *distributed control* function $u(x,t)$ and *Neumann control* $v(x,t)$ on the boundary

$$y_t(x,t) - \Delta y(x,t) + f(y(x,t)) = u(x,t) \qquad \text{in } Q := \Omega \times (0,T) \tag{2.22a}$$
$$\partial_\nu y(x,t) = v(x,t) \qquad \text{on } \Sigma := \Gamma \times (0,T) \tag{2.22b}$$
$$y(0,x) = y_0(x) \qquad \text{in } \Omega \tag{2.22c}$$

In most practical applications only one of these control functions is present in the system.

Remark 2.9
It should be mentioned that most theorems presented in this section are (under suitable conditions) valid for more general PDEs :

- *The negative Laplacian can be replaced by an elliptic differential operator*
$$\mathcal{A}y(x) = -\sum_{i,j=1}^{n} D_i(a_{ij}(x)D_j y(x)), \quad x \in \Omega$$
satisfying $\sum_{i,j=1}^{n} a_{ij}(x)\xi_i\xi_j \geq \gamma|\xi|^2,\ \gamma > 0,\ \forall \xi \in \mathbb{R}^n$.
- *The nonlinear function f may depend on x and t, i.e., $f(x,t,y)$.*
- *A nonlinear function $b(x,t,y)$ can be incorporated on the boundary.*
- *A homogeneous Dirichlet condition can be imposed on the boundary.*

In general, one cannot expect to find *classical solutions* of (2.22) and, thus, we are interested in so called *weak solutions.* For this purpose we multiply equation (2.22a) with $\phi \in H^1(\Omega)$ and integrate over Ω. Using integration by parts we obtain

$$\begin{aligned}&\int_\Omega y_t(x,t)\phi(x)\,dx + \int_\Omega \nabla y(x,t)\nabla\phi(x)\,dx + \int_\Omega f(y(x,t))\phi(x)\,dx\\ &= \int_\Omega u(x,t)\phi(x)\,dx + \int_\Gamma v(x,t)\phi(x)\,dS. \end{aligned}\tag{2.23}$$

This representation also forms the basis of the Galerkin method presented in Section 2.3.1.

Definition 2.10
A function $y \in W(0,T)$ is called ***weak solution*** *of* (2.22) *if*

- (2.23) *holds for each $\phi \in H^1(\Omega)$ and a.e. time $0 \leq t \leq T$*
- *and the initial condition $y(0) = y_0$ is satisfied.*

The following assumptions are required to prove the existence of a weak solution:

Assumption 2.11
Let $\Omega \subset \mathbb{R}^n, n \geq 1$, be a bounded Lipschitz domain (for $n = 1$ a bounded interval). The function f is assumed to be locally Lipschitz, continuous and monotonically increasing.

Theorem 2.12
Suppose that Assumption 2.11 hold. Then the semilinear parabolic initial value problem (2.22) *has a* ***unique weak solution*** *$y \in W(0,T) \cap C(\bar{Q})$ for any triple $u \in L^r(Q), v \in L^s(\Sigma)$ and $y_0 \in C(\bar{\Omega})$ with $r > n/2 + 1$, and $s > n + 1$.*

The proof of an even more general result can be found in [81]. The uniqueness is quite important, because it guarantees the existence of a unique *control to state* mapping $S : L^r(Q) \times L^s(\Sigma) \to W(0,T) \cap C(\bar{Q}), (u,v) \mapsto y$.

After these results concerning the state equation we introduce the general cost functional, cf. [91],

$$\begin{aligned}\min J(y,u,v) := \int_\Omega \phi(y(x,T))\,dx + \iint_Q \varphi(y(x,t),u(x,t))\,dxdt \\ \iint_\Sigma \psi(y(x,t),v(x,t))\,dSdt.\end{aligned} \tag{2.24}$$

Assumption 2.13

- *$\Omega \subset \mathbb{R}^n$ is a bounded Lipschitz domain.*
- *The functions ϕ, φ and ψ are twice differentiable with respect to y, u and v.*
- *Let φ and ψ be convex with respect to u and v, respectively.*
- *We have $f_y(y) \geq 0$ and $y_0 \in C(\bar{\Omega})$.*
- *The sets of admissible control values are given by*

  $$\begin{aligned}\mathbb{U} &= \{u \in L^\infty(Q) : u_a(x,t) \leq u(x,t) \leq u_b(x,t) \quad \text{for a.e. } (x,t) \in Q\}\\ \mathbb{V} &= \{v \in L^\infty(\Sigma) : v_a(x,t) \leq v(x,t) \leq v_b(x,t) \quad \text{for a.e. } (x,t) \in \Sigma\}\end{aligned}$$

  with $u_a, u_b \in L^\infty(Q), u_a(x,t) \leq u_b(x,t)$ and $v_a, v_b \in L^\infty(\Sigma), v_a(x,t) \leq v_b(x,t)$.

The existence of an optimal control can be shown with these assumptions, cf. [91].

Theorem 2.14
Suppose that Assumption 2.13 holds. Then the optimal control problem with cost functional (2.24) and PDE (2.22) has at least one optimal pair $(\bar{u},\bar{v}) \in \mathbb{U} \times \mathbb{V}$ with associated state $\bar{y} = y(\bar{u},\bar{v})$.

We close the investigation of this optimal control problem by presenting the first order optimality conditions, cf. [91].

Theorem 2.15
Suppose that Assumption 2.13 is satisfied and let $(\bar{u},\bar{v})$ an optimal control pair for the optimal control problem with cost functional (2.24) and state equation (2.22). Let the adjoint state $p \in W(0,T) \cap L^\infty(Q)$ be the solution of the corresponding adjoint equation

$$-p_t(x,t) - \Delta p(x,t) + f_y(\bar{y}(x,t))p(x,t) = \varphi_y(\bar{y}(x,t),\bar{u}(x,t)) \qquad \text{in } Q \tag{2.25a}$$

$$\partial_\nu p(x,t) = \psi_y(\bar{y}(x,t),\bar{v}(x,t)) \qquad \text{on } \Sigma \tag{2.25b}$$

$$p(x,T) = \phi_y(\bar{y}(x,T)) \qquad \text{in } \Omega. \tag{2.25c}$$

Then the variational inequalities

$$\iint_Q (p(x,t) + \varphi_u(\bar{y}(x,t),\bar{u}(x,t)))(u(x,t) - \bar{u}(x,t))\,dxdt \geq 0 \quad \forall u \in \mathbb{U} \tag{2.26a}$$

$$\iint_\Sigma (p(x,t) + \psi_v(\bar{y}(x,t),\bar{v}(x,t)))(v(x,t) - \bar{v}(x,t))\,dxdt \geq 0 \quad \forall v \in \mathbb{V} \tag{2.26b}$$

are satisfied.

Obviously, we have exactly the same structure as in Theorem 2.4 where we investigated the abstract optimal control system: The necessary conditions consist of the state equation (2.22), the adjoint equation (2.25) and the variational inequalities (2.26).

Example 2.16
In this example we consider the semilinear heat equation with distributed control and a quadratic cost functional. To shorten the notation we neglect the function arguments. We look at the optimal control problem

$$\min J(y,u) = \frac{1}{2}\|y - y_d\|_{L^2(Q)} + \frac{\lambda}{2}\|u\|_{L^2(Q)} \tag{2.27}$$

subject to

$$\begin{aligned} y_t - \Delta y + f(y) &= u && \text{in } Q \\ \partial_\nu y &= 0 && \text{on } \Sigma \\ y(0) &= y_0 && \text{in } \Omega \end{aligned}$$

where $\lambda > 0$ denotes a regularization parameter and y_d the desired state. Then the adjoint equation is given by

$$\begin{aligned} -p_t - \Delta p + f_y(\bar{y})p &= \bar{y} - y_d && \text{in } Q \\ \partial_\nu p &= 0 && \text{on } \Sigma \\ p(T) &= 0 && \text{in } \Omega. \end{aligned}$$

For the variational inequality we obtain

$$\iint_Q (p + \lambda\bar{u})(u - \bar{u}) \geq 0 \quad \forall u \in \mathbb{U}.$$

By using the ***projection operator*** *$P_{[u_a,u_b]}(v) := \max\{u_a, \min\{u_b, v\}\}$, the variational inequality can be written in the following useful form, cf. [91],*

$$\bar{u}(x,t) = P_{[u_a,u_b]}\left\{-\frac{1}{\lambda}p(x,t)\right\}.$$

Remark 2.17
In Section 3.1 we analyse a larger class of semilinear PDEs which is not fully covered by the theorems in this section. This especially concerns the non monotonic Schlögl equation presented in Section 5.1. However, in this case the transformation $y(x,t) := e^{\mu t}v(x,t)$ (for a particular choice of $\mu \in \mathbb{R}$) leads to a monotonically increasing nonlinearity, which can be addressed with the methods presented in this section, cf. [20].

Remark 2.18
The treatment of Dirichlet boundary control is more complicated than the corresponding Neumann control because the standard variational formulation does not work in this case. However, the linear heat equation with Dirichlet boundary control considered in this thesis was already studied in the early seventies by [68] using the so called transposition method. The existence and regularity results concerning the boundary controlled linear wave equation considered in Section 3.7.2 can also be found in [68]. A nice approach to deal with general linear quadratic optimal control problems is given by the theory of strongly continuous semigroups, see e.g. [15] and [67].

2.3 Discretization of semilinear parabolic PDEs

In this section we present a method for the numerical solution of the semilinear heat equation (2.22). We choose an approach where we start with a discretization of the space variable. Afterwards, we solve the resulting semi-discretized system by a time-stepping scheme. In the literature this technique is known as **vertical method of lines**, cf. [42]. For the spatial discretization we use the so called **Galerkin approximation**. The introduced concepts can be found in most textbooks concerning the numerical approximation of PDEs, see e.g. [61] and [42] for a practical approach. The stability theory of finite elements and the associated theory of the approximation errors for a semilinear PDE can be found in [89]. An introduction to finite elements in the context of optimal control is given in [73]. For the time discretization we discuss the **semi-implicit Euler method**.

2.3.1 Galerkin Approximation

The Galerkin method for the semilinear heat equation (2.22) is based on the weak variational formulation (2.23). For $y \in W(0,T)$ the variational representation was given by

$$\begin{aligned}&\int_\Omega y_t(x,t)\phi(x)\,dx + \int_\Omega \nabla y(x,t)\nabla\phi(x)\,dx + \int_\Omega f(y(x,t))\phi(x)\,dx\\ &= \int_\Omega u(x,t)\phi(x)\,dx + \int_\Gamma v(x,t)\phi(x)\,dS. \qquad (2.28)\end{aligned}$$

for all *test functions* $\phi \in V := H^1(\Omega)$ and a.e. $t \in [0,T]$. The idea of the Galerkin approach is to approximate the infinite dimensional function space V by a finite dimensional subspace $V_h := \operatorname{span}\{\psi_1, \ldots, \psi_{N_G}\} \subset V$. The linear independent *ansatz functions* $\{\psi_j\}_{j=1}^{N_G}$ form a basis of V_h, where N_G denotes the dimension of the Galerkin approximation. In the next step we approximate the functions y, u and v with respect to the ansatz functions

$$y_h(x,t) = \sum_{j=1}^{N_G} y_j(t)\psi_j(x), \quad u_h(x,t) = \sum_{j=1}^{N_G} u_j(t)\psi_j(x), \tag{2.29a}$$

$$v_h(x,t) = \sum_{j=1}^{N_G} v_j(t)\tau_\Gamma(\psi_j)(x) \quad \text{and} \quad y_{0h}(x) = \sum_{j=1}^{N_G} y_{0j}\psi_j(x) \tag{2.29b}$$

with coefficients $y_j(t), u_j(t), v_j(t)$ and initial condition y_{0j}. The continuous trace operator is denoted by $\tau_\Gamma : H^1(\Omega) \to L^2(\Gamma)$. Moreover, we define the coefficient vectors

$$\begin{aligned} \boldsymbol{y_h}(t) &:= (y_1(t), \dots, y_{N_G}(t))^\top, & \boldsymbol{u_h}(t) &:= (u_1(t), \dots, u_{N_G}(t))^\top \\ \boldsymbol{v_h}(t) &:= (v_1(t), \dots, v_{N_G}(t))^\top, & \boldsymbol{y_0} &:= (y_{01}, \dots, y_{0N_G})^\top \end{aligned}$$

with $\boldsymbol{y_h}, \boldsymbol{u_h}, \boldsymbol{v_h} : [0,T] \to \mathbb{R}^{N_G}$ and $\boldsymbol{y_0} \in \mathbb{R}^{N_G}$.
By inserting (2.29) into the weak formulation (2.28) and letting the *test functions* ϕ range over $\psi_i(x)$ for $i = 1, \dots, N_G$ we get

$$\begin{aligned} &\sum_{j=1}^{N_G} \dot{y}_j(t) \int_\Omega \psi_j(x)\psi_i(x)\, dx + \sum_{j=1}^{N_G} y_j(t) \int_\Omega \nabla\psi_j(x)\nabla\psi_i(x)\, dx \\ &+ \int_\Omega f\left(\sum_{j=1}^{N_G} y_j(t)\psi_j(x)\right)\psi_i(x)\, dx \\ &= \sum_{j=1}^{N_G} u_j(t) \int_\Omega \psi_j(x)\psi_i(x)\, dx + \sum_{j=1}^{N_G} v_j(t) \int_\Gamma \tau_\Gamma(\psi_j)(x)\tau_\Gamma(\psi_i)(x)\, dS. \end{aligned} \tag{2.30}$$

The initial value is given by the projection of $y_0(x)$ onto the subspace V_h, i.e.,

$$y_{0j} = \int_\Omega y_0(x)\psi_j(x)\, dx, \quad \text{and} \quad \boldsymbol{y_0} =: \mathcal{S}(y_0(x)). \tag{2.31}$$

By using the standard notation for the *mass matrix* M and the *stiffness matrix* K, i.e.,

$$M_{ij} := \int_\Omega \psi_i(x)\psi_j(x)\, dx \quad \text{and} \quad K_{ij} := \int_\Omega \nabla\psi_j(x)\nabla\psi_i(x)\, dx, \quad i,j = 1, \dots N_G$$

we obtain the following N_G-dimensional ODE initial value problem

$$M\dot{\boldsymbol{y}}_{\boldsymbol{h}}(t) + K\boldsymbol{y_h}(t) + F(\boldsymbol{y_h}(t)) = M\boldsymbol{u_h}(t) + Q\boldsymbol{v_h}(t) \tag{2.32a}$$

$$\boldsymbol{y_h}(0) = \boldsymbol{y_0}. \tag{2.32b}$$

Here we used the abbreviation

$$Q_{ij} := \int_\Gamma \tau_\Gamma(\psi_i)(x)\tau_\Gamma(\psi_j)(x)\, dS \quad i,j = 1,\dots N_G \quad \text{and} \tag{2.33}$$

$$F(\boldsymbol{y_h}(t))_i := \int_\Omega f\left(\sum_{j=1}^{N_G} y_j(t)\psi_j(x)\right)\psi_i(x)\, dx \quad i = 1,\dots N_G. \tag{2.34}$$

Thus, the Galerkin semi discretization leads to a system of (nonlinear) ODEs which has to be solved by a time stepping scheme.

Remark 2.19
It is not necessary to take both the test functions and the ansatz functions from the same finite dimensional subspace V_h. The choice of different spaces for ansatz- and test functions generally leads to asymmetric mass- and stiffness matrices. In the literature this approach is known as ***Petrov-Galerkin method****, cf. [19].*

Note that the spatial discretization of the PDE also provides a natural discretization of the cost functional and it is reasonable to choose the same approximation order for both. For the important stage costs (1.19), for instance, we obtain

$$\begin{aligned}
l(y_h(n), u_h(n)) &= \frac{1}{2}\|y_h(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u_h(\cdot, nT)\|^2_{L^2(\Omega)} \\
&= \frac{1}{2}\left\langle \sum_{j=1}^{N_G} y_j(nT)\psi_j(\cdot), \sum_{i=1}^{N_G} y_i(nT)\psi_i(\cdot)\right\rangle_{L^2(\Omega)} + \frac{\lambda}{2}\left\langle \sum_{j=1}^{N_G} u_j(nT)\psi_j(\cdot), \sum_{i=1}^{N_G} u_i(nT)\psi_i(\cdot)\right\rangle_{L^2(\Omega)} \\
&= \frac{1}{2}\sum_{j=1}^{N_G}\sum_{i=1}^{N_G} y_j(nT)y_i(nT)\langle\psi_j(\cdot),\psi_i(\cdot)\rangle_{L^2(\Omega)} + \frac{\lambda}{2}\sum_{j=1}^{N_G}\sum_{i=1}^{N_G} u_j(nT)u_i(nT)\langle\psi_j(\cdot),\psi_i(\cdot)\rangle_{L^2(\Omega)} \\
&= \frac{1}{2}\boldsymbol{y}_h(nT)^\top M\boldsymbol{y}_h(nT) + \frac{\lambda}{2}\boldsymbol{u_h}(nT)^\top M\boldsymbol{u_h}(nT).
\end{aligned} \tag{2.35}$$

The stage cost for the boundary control case can be obtained in exactly the same way. The only difference is that in the second term the mass matrix M has to be replaced by the boundary mass matrix Q, i.e.,

$$l(y_h(n), v_h(n)) = \frac{1}{2}\boldsymbol{y}_h(nT)^\top M\boldsymbol{y}_h(nT) + \frac{\lambda}{2}\boldsymbol{v_h}(nT)^\top Q\boldsymbol{v_h}(nT). \tag{2.36}$$

It should be noted that we have not yet used the particular structure of the ansatz functions $\{\psi_j\}_{j=1}^{N_G} \subset V_h$. In the following we present two different methods for obtaining these functions.

Finite Element Method

First, we look at the *Finite Element Method (FEM)*. A detailed introduction can be found e.g. in [42], [19] and [61]. The idea is to choose the ansatz functions

$\{\psi_j\}_{j=1}^{N_G} \subset V_h$ in such a way that the resulting mass- and stiffness matrices have a sparse structure. This can be achieved by subdividing the domain $\bar{\Omega}$ into a finite number of subsets K_i (the so called *elements*), i.e., $\bar{\Omega} = \cup K_i$ and approximate the solution on these elements by (low dimensional) polynomials. We focus on the probably most famous approach in which the two dimensional domain is discretized by a triangulation $\mathcal{T}_h$ and the ansatz functions are piecewise linear on each element.

In the following we assume that $\mathcal{T}_h$ is a *regular triangulation* of Ω, cf. [42] for the precise definition, with maximum triangle diameter $h > 0$ and *nodes* (vertices of the triangle) $\{p_i\}_{i=1}^{N_{FEM}}$, where N_{FEM} denotes the number of nodes. Then the piecewise linear ansatz functions $\{\psi_j\}_{j=1}^{N_{FEM}}$ are defined by

$$\psi_i(p_j) = \delta_{ij}, \quad i, j = 1, \ldots, N_{FEM}. \tag{2.37}$$

These pyramid-shape functions are also denoted as *Courant elements* or *nodal basis*, cf. [19]. Note that the dimension of V_h coincides with the number of nodes. However, this changes if a Dirichlet boundary condition is imposed.
The main advantage of the FEM approach is that the mass matrix $M_{ij} = \int_\Omega \psi_i \psi_j \, dx$ only contains non-zero entries if the corresponding triangles have a common edge or node. The same holds for the stiffness matrix K. Thus, with this approach we obtain a large scale problem with a sparse structure.
After this specification of the subspace V_h we look at the approximation functions (2.29) at grid point p_i

$$y_h(p_i, t) = \sum_{j=1}^{N_{FEM}} y_j^{FEM}(t)\psi_j(p_i) = y_i^{FEM}(t), \quad i = 1, \ldots, N_{FEM}. \tag{2.38}$$

Thus, the approximation of the solution at point p_i coincides with the i-th coordinate function of the solution $\boldsymbol{y_h^{FEM}}(t)$. The same is true for the control $\boldsymbol{u_h^{FEM}}(t)$. Before we finally write down the resulting ODE system we want to investigate a useful property of the nodal basis concerning the nonlinear function (2.34). At the grid point p_i we have

$$f\left(\sum_{j=1}^{N_{FEM}} y_j^{FEM}(t)\psi_j(p_i)\right) = f(y_i^{FEM}(t)), \quad i = 1, \ldots, N_{FEM}. \tag{2.39}$$

The function $\sum_{j=1}^{N_{FEM}} f(y_j^{FEM}(t))\psi_j(x)$ coincides with $f\left(\sum_{j=1}^{N_{FEM}} y_j^{FEM}(t)\psi_j(x)\right)$ on the grid points. Furthermore, the difference between these functions is of exactly the same order as the FEM error. Thus, we can replace

$$F(\boldsymbol{y_h}(t))_i = \int_\Omega f\left(\sum_{j=1}^{N_{FEM}} y_j^{FEM}(t)\psi_j(x)\right)\psi_i(x)\, dx, \quad i = 1, \ldots N_{FEM} \tag{2.40}$$

by

$$\sum_{j=1}^{N_{FEM}} f\left(y_j^{FEM}(t)\right) \int_\Omega \psi_j(x)\psi_i(x)\,dx \quad i = 1, \dots N_{FEM}. \tag{2.41}$$

It should be mentioned that (2.41) is a considerable simplification because the ansatz functions do not longer occur in the nonlinear function. Analogously to (2.32), we end up with the following ODE system

$$M\dot{\boldsymbol{y}}_{\boldsymbol{h}}^{\boldsymbol{FEM}}(t) + K\boldsymbol{y}_{\boldsymbol{h}}^{\boldsymbol{FEM}}(t) + Mf(\boldsymbol{y}_{\boldsymbol{h}}^{\boldsymbol{FEM}}(t)) = M\boldsymbol{u}_{\boldsymbol{h}}^{\boldsymbol{FEM}}(t) + Q\boldsymbol{v}_{\boldsymbol{h}}^{\boldsymbol{FEM}}(t) \tag{2.42}$$

$$\boldsymbol{y}_{\boldsymbol{h}}^{\boldsymbol{FEM}}(0) = \boldsymbol{y_0} \tag{2.43}$$

where f is interpreted componentwise. The characteristic properties of this system are the high dimension and the sparsity structure of the matrices.
In our numerical implementation we use the *Matlab PDE toolbox* where the matrices M, K and Q can be directly obtained by the commands *assema* and *assemb*. For linear finite elements we have (under suitable conditions) the following well known a-priori error estimate, cf. [89],

$$\|y_h(x,t) - y(x,t)\| < ch^2, \quad \text{for fixed } t \in [0,T]. \tag{2.44}$$

Remark 2.20
It should be mentioned that there exist further important methods for the spatial discretization of PDEs, e.g. the ***Finite Difference Method*** *(FDM) and the* ***Finite Volume Method*** *(FVM). The finite difference method is often applied for one dimensional problems or in case of simple structured domains. We will use this approach for the implementation of the one dimensional examples in Chapter 6. In this case there is a strong relation between FEM and FDM, cf. [42].*

Proper Orthogonal Decomposition

In this section we introduce a second variant to obtain the finite dimensional subspace V_h in the Galerkin approximation: the model reduction technique *Proper Orthogonal Decomposition (POD)*. The method is also called *Karhunen-Loeve decomposition* and there exist much literature concerning this topic. A detailed introduction can be found in [96], where also the strong relation to the singular value decomposition is analysed.
We start this section by observing that the basis function in the FEM approach are not correlated to the physical properties of the underlying equation. On the one hand this fact is beneficial because the same ansatz functions can be used for very different classes of PDEs. On the other hand the resulting system of ODEs is quite large. The idea of POD is to choose the ansatz functions in such a way that they reflect the dynamics of the system. For this purpose we *assume* that we know the solution of the semilinear PDE (2.22) for given control functions $u(x,t)$ and $v(x,t)$ at discrete time instances $0 = t_1 < \cdots < t_n = T$. These so called *snapshots* can be

obtained, for instance, by measurement of a real physical process. In our algorithms presented in Chapter 4 we use high dimensional FEM models to get the solution on the time instances. The *snapshot ensemble* is denoted by $\mathcal{Y} := \{y_1^s(x), \dots, y_n^s(x)\}$ with $y_i^s(x) := y(x, t_i)$. If we assume that $\{\psi_i(x)\}_{i=1}^l$ denote an orthonormal basis of $\mathcal{Y}$, where $l = \dim \mathcal{Y}$, we can express each element of $\mathcal{Y}$ as

$$y_j^s(x) = \sum_{i=1}^{l} \langle y_j^s(x), \psi_i(x) \rangle_{L^2(\Omega)} \psi_i(x) \quad \text{for} \quad j = 1, \dots, n. \tag{2.45}$$

Depending on the number of snapshots and the dynamics of the system the dimension of $\mathcal{Y}$ can be quite large. The idea of POD is to extract the dominant basis functions and to consider a reduced basis of dimension $N_{POD} \ll l$. Of course, in this case we cannot expect to satisfy equation (2.45) as well. However, we will choose the reduced basis functions such that (2.45) is approximately satisfied in a least square sense, i.e.,

$$\min_{\{\psi_i(x)\}_{i=1}^{N_{POD}}} \frac{1}{n} \sum_{j=1}^{n} \left\| y_j^s(x) - \sum_{i=1}^{N_{POD}} \langle y_j^s(x), \psi_i(x) \rangle_{L^2(\Omega)} \psi_i(x) \right\|_{L^2(\Omega)}^2 \tag{2.46}$$
$$\text{subject to} \quad \langle \psi_i(x), \psi_j(x) \rangle_{L^2(\Omega)} = \delta_{ij} \text{ for } 1 \le i \le N_{POD}, 1 \le j \le i.$$

The solution $\{\psi_i(x)\}_{i=1}^{N_{POD}}$ of the minimization problem (2.46) is called *POD basis of rank N_{POD}* and can be obtained by solving an eigenvalue problem, cf. [96]. We follow the representation of [27]. First, we build up the so called *correlation matrix* $K \in \mathbb{R}^{n \times n}$, which is positive semi-definite with rank l, corresponding to the snapshots $\{y_i^s(x)\}_{i=1}^n$ by

$$K_{ij} = \frac{1}{n} \langle y_i^s(x), y_j^s(x) \rangle_{L^2(\Omega)}.$$

Afterwards, we compute the positive eigenvalues $\lambda_1 \ge \dots \ge \lambda_l > 0$ and the corresponding eigenvectors $v_1, \dots, v_l \in \mathbb{R}^n$. Then the POD basis of rank $N_{POD} \le l$ is given by

$$\psi_i(x) = \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{n} (v_i)_j y_j^s(x), \tag{2.47}$$

where $(v_i)_j$ denotes the j-th component of the eigenvector v_i.

Using the computed POD basis we can formulate the Galerkin approximation

$$y_h(x,t) = \sum_{j=1}^{N_{POD}} y_j^{ROM}(t) \psi_j(x), \quad u_h(x,t) = \sum_{j=1}^{N_{POD}} u_j^{ROM}(t) \psi_j(x), \tag{2.48a}$$

$$v_h(x,t) = \sum_{j=1}^{N_{POD}} v_j^{ROM}(t) \tau_\Gamma(\psi_j)(x) \quad \text{and} \quad y_{0h}(x) = \sum_{j=1}^{N_{POD}} y_{0j}^{ROM} \psi_j(x). \tag{2.48b}$$

Note that in contrast to the FEM case an interpretation of the coefficient vectors

$$\begin{aligned}\boldsymbol{y_h^{ROM}}(t) &:= (y_1^{ROM}(t), \dots, y_{N_{POD}}^{ROM}(t))^\top, & \boldsymbol{u_h^{ROM}}(t) &:= (u_1(t), \dots, u_{N_{POD}}^{ROM}(t))^\top \\ \boldsymbol{v_h^{ROM}}(t) &:= (v_1^{ROM}(t), \dots, v_{N_{POD}}^{ROM}(t))^\top, & \boldsymbol{y_0^{ROM}} &:= (y_{01}^{ROM}, \dots, y_{0N_{POD}}^{ROM})^\top\end{aligned}$$

with respect to spatial coordinates is not possible. In order to write the formulas (2.48) in a compact way we define the linear operator $\mathcal{P} : \mathbb{R}^{N_{POD}} \to L^2(\Omega)$ by

$$w(x) = \mathcal{P}(\boldsymbol{w^{ROM}}) := \sum_{j=1}^{N_{POD}} w_j^{ROM} \psi_j(x)$$

with $\boldsymbol{w^{ROM}} \in \mathbb{R}^{N_{POD}}$. Furthermore, the initial value is given by the projection of the initial function $y_0(x)$ onto the reduced basis, i.e.,

$$y_{0j}^{ROM} = \int_\Omega y_0(x)\psi_j(x)\, dx, \quad \text{and} \quad \boldsymbol{y_0^{ROM}} =: \mathcal{S}(y_0(x)). \tag{2.49}$$

By using the general Galerkin ODE system (2.32) and the concepts of the spatially discrete cost function (2.35) we can formulate the **reduced POD optimal control problem**

$$M\boldsymbol{\dot{y}_h^{ROM}}(t) + K\boldsymbol{y_h^{ROM}}(t) + F(\boldsymbol{y_h^{ROM}}(t)) = M\boldsymbol{u_h^{ROM}}(t) + Q\boldsymbol{v_h^{ROM}}(t) \tag{2.50a}$$

$$\boldsymbol{y_h^{ROM}}(0) = \boldsymbol{y_0^{ROM}} \tag{2.50b}$$

with the corresponding quadratic cost functional

$$\begin{aligned} J(\boldsymbol{y_h^{ROM}}(t), \boldsymbol{u_h^{ROM}}(t), \boldsymbol{v_h^{ROM}}(t)) = \int_0^T &\frac{1}{2}\boldsymbol{y}_h(t)^\top M \boldsymbol{y}_h(t) + \frac{\lambda}{2}\boldsymbol{u_h}(t)^\top M \boldsymbol{u_h}(t) \\ &+ \frac{\hat{\lambda}}{2}\boldsymbol{v_h}(t)^\top Q \boldsymbol{v_h}(t)\, dt \end{aligned} \tag{2.50c}$$

where $\lambda, \hat{\lambda} > 0$ denote the regularization parameters. It is important to note that in contrast to the finite element case we cannot get rid of the ansatz functions in the nonlinearity f, because the approximation approach (2.41) does not work, see also Remark 2.21. Furthermore, we want to mention that we use the notation y^{ROM} (Reduced Order Model) instead of y^{POD}. On the one hand this indicates that also other model reduction techniques can be used. On the other hand we will need the notation y^{POD} later for the closed loop trajectory of the *full model* that uses the control obtained from the reduced POD optimization problem.
The most important property of the ODE system (2.50a) is that the dimension is quite small (in our simulations we have $N_{POD} \leq 5$) and the corresponding matrices are dense while the finite element method provides high dimensional matrices with a sparse structure.

In the context of POD some questions arise:

- How do we obtain the control for the creation of the snapshots?
- Where should the snapshots be localized?
- What information should be incorporated into the snapshot ensemble?
- How large is the model error?

The answers for these questions are not easy and there is some literature concerning these topics. Some ideas how to find the control for the snapshot generation are shortly discussed in Section 4.2. Further information can be found in [37] and [27]. The location and the number of snapshots are often determined in a more or less heuristic way. However, some works deal with the optimal snapshots location, cf. [64]. In [27] it was shown that it is reasonable to incorporate not only the snapshots of the state equation, but also the snapshots of the adjoint equation and derivative information, cf. [63], into the ensemble. The question concerning the error between the control from the POD model and the exact one demonstrates a major problem of the POD method. Although there exist *a-posteriori* error estimates for some classes of PDEs, cf. [92], only very few *a-priori* estimates are available for this method.

Remark 2.21
It should be mentioned that the POD approach is not a universal remedy for solving optimal control problems. One major drawback is the lack of appropriate a-priori error estimates. Furthermore, one has to take into account that the resulting POD matrices are dense. In [20], for instance, the authors demonstrate that the required computing time for a POD model of dimension 15 *with fully populated matrices was higher than the* 300*-dimensional FE based model with sparse matrices. Moreover, we want to recall that in the nonlinear function F of* (2.50a) *the ansatz functions are still present. Thus, the evaluation requires operations in the high dimensional space, which is especially disadvantageous for optimization algorithms. However, current research on the Discrete Empirical Interpolation Method (DEIM) indicates that this difficulty can be overcome by building new basis functions upon the nonlinear term, cf. [22].*

2.3.2 Discretization in Time

In this section we present a time-stepping scheme to solve the semi-discretized PDE (2.32). Much has been written about ODE solvers and in particular ODE integrators for discretized PDEs. We want to restrict ourselves to the first order *Euler method*, which is the most simple example of an ODE solver. However, arbitrary time stepping schemes can be incorporated into our C++ implementation. Again, we look at the ODE system (2.32)

$$M\dot{\boldsymbol{y}}_{\boldsymbol{h}}(t) + K\boldsymbol{y}_{\boldsymbol{h}}(t) + F(\boldsymbol{y}_{\boldsymbol{h}}(t)) = M\boldsymbol{u}_{\boldsymbol{h}}(t) + Q\boldsymbol{v}_{\boldsymbol{h}}(t) \tag{2.51}$$

$$\boldsymbol{y}_{\boldsymbol{h}}(0) = \boldsymbol{y}_{\boldsymbol{0}}. \tag{2.52}$$

The goal is to find an approximation of $\boldsymbol{y_h}(t)$ for given control functions $\boldsymbol{u_h}(t)$ and $\boldsymbol{v_h}(t)$. In order to shorten the notation we define $P(t) := M\boldsymbol{u_h}(t) + Q\boldsymbol{v_h}(t)$. Furthermore, we introduce a time grid $0 = t_0 < t_1 < \ldots$ with $t_i := i\tau$, where τ denotes the stepsize of the integration scheme.
It is well known, cf. [42], that *explicit* methods are generally not qualified for semi-discretized PDEs due to the stiffness of the resulting matrices. Thus, we start our investigations with the *implicit Euler method*

$$(M + \tau K)\,\boldsymbol{y_h}(t_{i+1}) + \tau F(\boldsymbol{y_h}(t_{i+1})) = M\boldsymbol{y_h}(t_i) + \tau P(t_i). \tag{2.53}$$

In order to obtain the solution $\boldsymbol{y_h}(t_{i+1})$ we have to solve a nonlinear equation system in each time step. However, for high dimensional systems or strongly nonlinear problems this is a difficult task. This leads to the idea to combine the advantages of the explicit- and the implicit Euler method. The *semi-implicit* Euler method treats the nonlinear function F in an explicit way, i.e., we obtain

$$(M + \tau K)\,\boldsymbol{y_h}(t_{i+1}) = M\boldsymbol{y_h}(t_i) + \tau F(\boldsymbol{y_h}(t_i)) + \tau P(t_i). \tag{2.54}$$

Although we preserve the beneficial stability behaviour of the implicit method (under suitable conditions), we only have to solve a linear system of equations. This idea is a special case of more general *operator splitting methods*, cf. [56]. The stability and convergence analysis of the semi-implicit Euler method can be found in [87].
Similar to the spatial discretization the time stepping scheme also provides a discrete approximation of the time integral in the cost functional (2.24). Note that in our investigations the stepsize τ is much smaller than the sampling time T, see Remark 1.8.

Remark 2.22
By using the semi-implicit Euler method the effort for solving a nonlinear ODE system is essentially the same as for solving a linear ODE system. In Chapter 4 we will see that the advantage of the SQP method is that only linear PDEs have to be solved. In contrast to that the (reduced) Newton method requires the solution of nonlinear PDEs. If the resulting ODE system is solved by this semi-implicit scheme the advantage of the SQP method is negligible, see also Section 4.1.2.

Remark 2.23
As already mentioned, most explicit ODE solvers are not suited for large scale stiff systems. However, there exist stabilized explicit methods which are developed for the application on high dimensional semi-discretized parabolic PDEs. In Chapter 6 we will successfully use a representative of the ROCK (Orthogonal-Runge-Kutta-Chebyshev) family, cf. [56]. In this context we implement the method of mass lumping to apply the explicit method, cf. [42].

3 Minimal Stabilizing Horizons

In this chapter we come back to the MPC method introduced in Section 1.2. As already mentioned, the crucial parameter in the analysis of the MPC algorithm is given by the length of the optimization horizon. Especially, the smallest horizon, for which the MPC feedback stabilizes the system, is of interest from the theoretical and practical point of view. In the following this horizon is called *minimal stabilizing horizon*. The aim of this chapter is to estimate this particular horizon for different partial differential equations. This chapter contains the theoretical contribution of this thesis.
Theorem 1.20 provides a possibility for estimating the minimal stabilizing horizon that guarantees a desired suboptimality estimate by using the exponential controllability condition (1.34). In this chapter we aim at estimating the overshoot constant C and the decay rate σ for different classes of parabolic PDEs. The results will be used for explaining the qualitative behaviour of the horizon with regard to several parameters. Furthermore, we can employ the method to give design guidelines for the stage cost l in order to shorten the horizon. Note that some exponential estimates for the presented equations are known in the literature, e.g. [26]. However, the exponential constants are often not specified and the control cost is not incorporated.
Our road map is inspired by Theorem 1.20 and is given by the following scheme:

1. Find (not necessarily optimal) controls u such that the exponential controllability condition (1.34) is fulfilled.

2. Calculate C and σ for this particular control.

3. Determine the minimal horizon $\bar{N}$ that guarantees the desired suboptimality degree by using (1.35)

Note that the effort for finding an appropriate control u in step 1 is quite different for various types of equations. Thus, the generality of the presented. results strongly depends on the considered PDE and the control structure. In Section 3.1 we derive the exponential constants for a class of n-dimensional semilinear heat equations with distributed control (3.1.1) and compare the results with a numerical simulation (3.1.2). Afterwards, in Section 3.2 we introduce with the well known *backstepping approach*, cf. [62], a possibility to obtain a stabilizing feedback for the boundary controlled one dimensional heat equation. On the basis of these results we derive the exponential constants for the Dirichlet controlled heat equation in Section 3.3 and for Neumann control in Section 3.4. The influence of an additional convection term in the heat equations is analysed in Section 3.5. In Section 3.6 we compare our method to determine a stabilizing horizon with an approach presented in [75]. In order to demonstrate that our road map is not only applicable to parabolic systems but also to hyperbolic PDEs in Section 3.7 we look at the example of the linear wave equation.

3.1 Distributed Control

In this section we apply the steps of our road map to a semilinear heat equation with distributed control. Using a feedback control law we derive the exponential constants for two different stage costs. Afterwards, we use these results in order to explain the numerical results from the simulation.

3.1.1 Derivation of the Exponential Constants

We consider the semilinear heat equation

$$\begin{aligned} y_t(x,t) &= \Delta y(x,t) + f(y(x,t))y(x,t) + u(x,t) && \text{in} \quad \Omega \times (0,\infty) && (3.1a)\\ y(x,t) &= 0 && \text{on} \quad \partial\Omega \times (0,\infty) && (3.1b)\\ y(x,0) &= y_0(x) && \text{in} \quad \Omega && (3.1c)\end{aligned}$$

with homogeneous Dirichlet boundary conditions and distributed control inside the domain. We assume that $f \in C^1(\mathbb{R},\mathbb{R})$ and $f(y) \leq M, \forall y \in \mathbb{R}$ with a constant $M \in \mathbb{R}$. In Theorem 3.6 we additionally require $(f(y)y)' \leq M, \forall y \in \mathbb{R}$.

Remark 3.1
An important sufficient condition for the global existence of solutions of semilinear PDEs with nonlinearity $g(y)$ is given by, cf. [21],

$$yg(y) \leq C|y|^2 \quad \text{for} \quad |y| \leq K$$

with constants $K, C < \infty$. In the case $g(y) = f(y)y$ this condition is always satisfied with $C = M$.

Remark 3.2
In [8] we consider more general nonlinearities by replacing $f(y)y$ by a continuously differentiable function $g(y)$ satisfying $g(0) = 0$. However, in this case we only obtain local results and $g'(0)$ plays the role of M.
As an example we look at the nonlinearity of Schlögl type $g(y) = \mu(y - y^3)$. We obtain $g'(0) = \mu$ and $f(y) = \mu(1 - y^2) \leq \mu = M$.

It is well known that for the uncontrolled equation ($u \equiv 0$) the origin $y(x) \equiv 0$ can be an unstable equilibrium for $M \geq \lambda_1$ where $\lambda_1 = \lambda_1(\Omega)$ denotes the smallest eigenvalue of $(-\Delta)$ in $H_0^1(\Omega)$. Our goal consists of stabilizing this unstable equilibrium. For the choice of the stage cost l there are various possibilities. A heuristic analysis of the influence of different stage costs on the MPC algorithm can be found in [7]. A common choice, cf. [57], of l is given by

$$l(y(n), u(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot, nT)\|^2_{L^2(\Omega)}, \tag{3.2}$$

i.e., we penalize the state as well as the control in the $L^2(\Omega)$-norm. The constant $\lambda > 0$ denotes a so called regularization or Tikhonov parameter. In order to fulfill the first step of our road map we have to find a control u such that the exponential controllability condition (1.34) is satisfied. A key issue in our method is that we do not need that this control is optimal in any sense. It is well known, cf. [26], that the state feedback

$$u(x, t) := -Ky(x, t) \tag{3.3}$$

stabilizes the linear heat equation (i.e. $f \equiv \mathit{const}$) for a sufficient large constant $K \in \mathbb{R}$.

Since we are interested in tight estimates for the controllability constants we require the following version of the Friedrichs inequality (2.21), where the best *Friedrichs constant* is explicitly given:

Theorem 3.3

Let Ω be a Lipschitz domain. For $u \in H^1_0(\Omega)$ the following inequality holds

$$\int_\Omega |u(x)|^2 \, dx \leq C \int_\Omega |\nabla u(x)|^2 \, dx. \tag{3.4}$$

The optimal Friedrichs constant is given by $C = \frac{1}{\lambda_1}$ where λ_1 denotes the least eigenvalue of the Dirichlet eigenproblem

$$\begin{aligned} -\Delta u &= \mu u \quad && \text{in} \quad \Omega \\ u &= 0 && \text{on} \quad \partial\Omega. \end{aligned}$$

The proof of a more general result can be found in [13].

Theorem 3.4

The semilinear heat equation (3.1) with control (3.3), $K > M - \lambda_1$, and stage cost (3.2) satisfies the exponential controllability condition (1.34). The corresponding constants are given by $\sigma = e^{-2T(\lambda_1 - M + K)} \in (0, 1)$ and $C = (1 + \lambda K^2) \in \mathbb{R}$.

Proof. The particular control (3.3) reduces the equation (3.1) to

$$y_t(x, t) = \Delta y(x, t) + (f(y(x, t)) - K)y(x, t). \tag{3.5}$$

Note that we get a more regular solution than in the general case, because there is no longer a control function in the equation. In order to obtain an estimation for σ we use classical Lyapunov methods presented in e.g. [26]. For this purpose we define

$$V(t) := \frac{1}{2}\|y(\cdot, t)\|^2_{L^2(\Omega)} = \frac{1}{2}\int_\Omega y(x, t)^2 \, dx \tag{3.6}$$

and differentiate $V(t)$ with respect to time

$$\begin{aligned}\dot V(t) &= \int_\Omega y(x,t)y_t(x,t)\,dx = \int_\Omega y(x,t)(\Delta y(x,t) + (f(y(x,t)) - K)y(x,t))\,dx\\ &\le \int_{\partial\Omega} y(x,t)\partial_\nu y(x,t)\,dS - \int_\Omega \nabla y(x,t)\cdot\nabla y(x,t)\,dx + (M-K)\int_\Omega y(x,t)^2\,dx\\ &\le (-\lambda_1 + M - K)\int_\Omega y(x,t)^2\,dx = 2(-\lambda_1 + M - K)V(t)\end{aligned}$$

where we used the tight Friedrichs inequality (3.4) in the last estimate. This yields

$$V(t) \le e^{-2t(\lambda_1 - M + K)}V(0) \tag{3.7}$$

and thus

$$l^*(y(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} \le \frac{1}{2}e^{-2nT(\lambda_1-M+K)}\|y(\cdot,0)\|^2_{L^2(\Omega)} = \sigma^n l^*(y_0)$$

with $\sigma := e^{-2T(\lambda_1-M+K)}$.
The overshoot constant C is determined by including the control effort

$$\begin{aligned}l(y(n),u(n)) &= \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot,nT)\|^2_{L^2(\Omega)}\\ &= \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}K^2\|y(\cdot,nT)\|^2_{L^2(\Omega)}\\ &= Cl^*(y(n)) \quad \text{with} \quad C := (1+\lambda K^2).\end{aligned}$$

By combining the previous results we obtain the desired inequality (1.34)

$$l(y(n),u(n)) \le Cl^*(y(n)) \le C\sigma^n l^*(y_0). \tag{3.8}$$

This completes the proof. □

The analysis in [7] shows that the minimal stabilizing horizon strongly depends on the choice of the stage cost. An alternative choice of l is to penalize the gradient of the state y instead of the state itself. Since we consider homogeneous Dirichlet boundary conditions this stage cost also identifies the equilibrium $y \equiv 0$ and we obtain

$$l(y(n),u(n)) = \frac{1}{2}\|\nabla y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot,nT)\|^2_{L^2(\Omega)}. \tag{3.9}$$

In order to prove a similar result to Theorem 3.4 for stage cost (3.9) we need the following extension of the Friedrichs inequality:

Lemma 3.5
For $u \in H^2(\Omega)\cap H^1_0(\Omega)$ the following inequality holds

$$\int_\Omega |\nabla u(x)|^2\,dx \le C\int_\Omega (\Delta u(x))^2\,dx$$

where C is the same tight Friedrichs constant as in Theorem 3.3, *Proof.* By using the Cauchy-Schwarz inequality and the classical Friedrichs inequality (3.4) we obtain

$$\begin{aligned}\int_\Omega |\nabla u(x)|^2\,dx &= \int_{\partial\Omega} u(x)\partial_\nu u(x)\,dS - \int_\Omega u(x)\Delta u(x)\,dx\\ &\le \left(\int_\Omega u(x)^2\,dx\right)^{\frac12}\left(\int_\Omega (\Delta u(x))^2\,dx\right)^{\frac12}\\ &\le \sqrt{C}\left(\int_\Omega |\nabla u(x)|^2\,dx\right)^{\frac12}\left(\int_\Omega (\Delta u(x))^2\,dx\right)^{\frac12}\end{aligned}$$

and thus

$$\left(\int_\Omega |\nabla u(x)|^2\,dx\right)^{\frac12} \le \sqrt{C}\left(\int_\Omega (\Delta u(x))^2\,dx\right)^{\frac12}$$

with the same optimal constant C. □

Theorem 3.6
The semilinear heat equation (3.1) with control (3.3), $K > M - \lambda_1$, and stage cost (3.9) satisfies the exponential controllability condition (1.34). The corresponding constants are given by $\sigma = e^{-2T(\lambda_1 - M + K)} \in (0,1)$ and $C = \left(1 + \frac{\lambda K^2}{\lambda_1}\right) \in \mathbb{R}$.

Proof. Since we do not change the control we obtain the same reduced PDE (3.5) as before. In particular, we have the same regularity results. Similar to (3.6) we define

$$V(t) := \frac12 \|\nabla y(\cdot,t)\|^2_{L^2(\Omega)} = \frac12 \int_\Omega \nabla y(x,t)\cdot\nabla y(x,t)\,dx. \tag{3.10}$$

Differentiating with respect to time yields

$$\begin{aligned}\dot V(t) &= \int_\Omega \nabla y(x,t)\cdot\nabla y_t(x,t)\,dx = \int_{\partial\Omega} y_t(x,t)\partial_\nu y(x,t)\,dS - \int_\Omega \Delta y(x,t) y_t(x,t)\,dx\\ &\le -\int_\Omega (\Delta y(x,t))^2\,dx + (M-K)\int_\Omega \nabla y(x,t)\nabla y(x,t)\,dx\\ &\le (-\lambda_1 + M - K)\int_\Omega \nabla y(x,t)\cdot\nabla y(x,t) = 2(-\lambda_1 + M - K)V(t)\end{aligned}$$

and thus

$$l^*(y(n)) = \frac12 \|\nabla y(\cdot,nT)\|^2_{L^2(\Omega)} \le \frac12 e^{-2nT(\lambda_1 - M + K)}\|\nabla y(\cdot,0)\|^2_{L^2(\Omega)} = \sigma^n l^*(y_0)$$

with $\sigma = e^{-2T(\lambda_1 - M + K)}$. In order to estimate the overshoot bound C we consider

$$\begin{aligned}l(y(n),u(n)) &= \frac12\|\nabla y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot,nT)\|^2_{L^2(\Omega)}\\ &= \frac12\|\nabla y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}K^2\|y(\cdot,nT)\|^2_{L^2(\Omega)}\\ &\le C l^*(y(n)) \quad\text{with}\quad C := \left(1 + \frac{\lambda K^2}{\lambda_1}\right).\end{aligned} \tag{3.11}$$

Together with the previous result we obtain the desired inequality. □

3.1.2 Numerical Results

In this section we apply the theoretical results presented before to a simple example. This section is divided into two parts: In the first one we follow our road map and compute concrete values of the minimal stabilizing horizon for our test example. Afterwards, we compare these *quantitative* results with the minimal stabilizing horizon we observed in a numerical simulation of the closed loop system. In the second part we use our theoretical results for a *qualitative* analysis in order to describe the dependence of the stabilizing horizon on different parameters and stage costs.
Our test example in this section is given by the linear one dimensional heat equation

$$y_t(x,t) = y_{xx}(x,t) + \mu y(x,t) + u(x,t) \quad \text{in} \quad (0,L)\times(0,\infty) \tag{3.12a}$$
$$y(0,t) = y(L,t) = 0 \quad \text{in} \quad (0,\infty) \tag{3.12b}$$
$$y(x,0) = y_0(x) \quad \text{in} \quad (0,L) \tag{3.12c}$$

with the corresponding stage cost

$$l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(0,L)} + \frac{\lambda}{2}\|u(\cdot,nT)\|^2_{L^2(0,L)} \tag{3.13}$$

and

$$l(y(n),u(n)) = \frac{1}{2}\|y_x(\cdot,nT)\|^2_{L^2(0,L)} + \frac{\lambda}{2}\|u(\cdot,nT)\|^2_{L^2(0,L)} \tag{3.14}$$

respectively. The parameters in this example are given by the reaction value μ, the interval length L, the regularization parameter λ and the sampling time T.

Quantitative analysis

In Section 3.1.1 we derived the exponential constants C and σ for a prescribed control u. Thus, two steps of our road map are already done. In the last step we use formula (1.35) in order to determine the smallest horizon that guarantees stability. The procedure is as follows: We insert the K-dependent constants $\sigma(K)$ and $C(K)$ in (1.35) and obtain $\alpha_N(K) = \alpha_N(C(K),\sigma(K),N)$. If there exists $K \in \mathbb{R}$ such that $\alpha_N(K) > 0$ Theorem 1.20 guarantees stability with horizon N. Therefore, the minimal stabilizing horizon is the smallest horizon $\bar{N}$ with $\max_K \alpha_{\bar{N}}(K) > 0$. Note that the optimization over K is an easy one dimensional problem. In the implementation for the example we use *Maple* to solve this problem.
Furthermore, we determine the smallest horizon where we observe stability in the numerical simulation of the MPC closed loop system. The arising control problems are solved with the optimization algorithms presented in Chapter 4.
First, we look at the 1D heat equation (3.12) with stage cost (3.13), domain $\Omega = (0,1)$ and sampling time $T = 0.01$. In Table 3.1 we display the minimal stabilizing horizon N_T computed by the results of Section 3.1.1 and the horizon observed in the numerical simulation N_P. Obviously, the values of N_T are always equal or greater than those of N_P. Thus, the theoretical results provide indeed upper bounds for the

	$\lambda = 0.001$		$\lambda = 0.005$		$\lambda = 0.01$	
μ	N_T	N_P	N_T	N_P	N_T	N_P
10	2	2	2	2	2	2
11	2	2	4	2	9	3
12	2	2	8	3	18	4
13	2	2	12	3	27	4
14	3	2	16	4	34	5
15	3	2	20	4	42	6

Table 3.1: Minimal stabilizing horizon computed from theory N_T and observed in the numerical simulation N_P

stabilizing horizons. However, it can be seen that the computed horizons are quite conservative for this example. This observation is true regardless of the parameter setting and the type of stage cost (but much less for small values of λ and μ). This behaviour is caused by different reasons: First, we have to note that the condition $\alpha_N > 0$ is only sufficient to guarantee stability. For $\alpha_N \leq 0$ no statement about stability of a concrete example is possible (see also Remark 1.21). Furthermore, Theorem 1.20 only requires the values of the exponential constants C and σ but no information about the underlying dynamical system. It seems to be reasonable that tighter estimates can be obtained by taking the special structure of the PDE and the stage cost into account (see Section 3.6). Finally, it is important to note that the chosen control is not the optimal one and this fact can also produce conservatism.
It should be mentioned that the described problems arise from the power of the used method: It can be applied to very general systems and the knowledge of an optimal control is not required. Since the conservatism of the results is also present for boundary control and different stage cost, in the remaining chapter we will focus on a *qualitative analysis.*

Remark 3.7
It should be mentioned that the derived exponential constants C and σ can be used to gain improved stability results. In [9] we reduce the conservatism of the estimates by replacing the controllability condition (1.35) *by a boundedness condition on the finite horizon optimal value functions. For details we refer to [99].*

Qualitative analysis

In the quantitative analysis we have seen that the theoretically computed horizons are in general quite conservative for a concrete example. Nevertheless, in the following we will show that the presented methods are very well suited for a *qualitative analysis* concerning the stability behaviour depending on parameters and different stage costs.
In the first step we want to analyse the impact of the different stage costs (3.2)

and (3.9) on the minimal stabilizing horizon. Obviously, we obtain the same decay rate $\sigma = e^{-2T(\lambda_1 - M + K)}$ for both costs (see Theorem 3.4 and 3.6). Thus, we can only explain the different behaviour through the overshoot constant C. The eigenvalue λ_1 influences not only the decay rate but also the overshoot bound in (3.9). It is important to note that λ_1 solely depends on the domain Ω. Thus, in order to investigate the difference between the stage costs in detail it is reasonable to look at a problem with different domains. In view of our test example this means that we consider (3.12) with varying interval length L. For the numerical simulation we choose the destabilizing reaction value $\mu \equiv 15$, the regularization parameter $\lambda = 0.01$ and the sampling time $T = 0.01$. The smallest eigenvalue for this problem is given by $\lambda_1 = (\pi/L)^2$, cf. [88]. Thus, the origin is an unstable equilibrium for $L \geq 1$ ($\lambda_1 = (\pi/L)^2 < 15 = \mu$).
The overshoot constants are given by $C = (1 + \lambda K^2)$ for stage cost (3.2) and by $C = (1 + \frac{\lambda K^2}{\lambda_1})$ for (3.9). Since a smaller constant C leads to a shorter horizon we expect to observe shorter horizons for (3.9) than for (3.2) if $\lambda_1(L) < 1$ and vice versa for $\lambda_1(L) > 1$. In order to demonstrate that this behaviour in fact occurs, we consider the minimal stabilizing horizon of (3.12) with varying interval length L. The results of the numerical simulation are presented in Table 3.2. The first two

L	$\lambda_1(L)$	$N_{\|y\|}$	$N_{\|\nabla y\|}$
1	π^2	6	2
2	$\pi^2/4$	8	5
3	$\pi^2/9$	8	7
π	1	8	8
4	$\pi^2/16$	8	10
5	$\pi^2/25$	8	12

Table 3.2: Minimal stabilizing optimization horizons for the reaction-diffusion equation (3.12) with $T = 0.01$ and $\lambda = 0.01$ determined by numerical MPC closed loop simulations

rows show the varying interval length L with the corresponding eigenvalue λ_1. The minimal stabilizing horizon observed in the numerical example is denoted by $N_{\|y\|}$ for stage cost (3.2) and by $N_{\|\nabla y\|}$ for stage cost (3.9). Obviously, the values $N_{\|\nabla y\|}$ are smaller than $N_{\|y\|}$ up to $L = \pi$. This behaviour changes to the opposite for $L > \pi$. The observation corresponds exactly to the theoretical results presented in Theorem 3.4 and Theorem 3.6: For $L < \pi$ there is $\lambda_1 > 1$ and the value C for the stage cost (3.9) is smaller than the corresponding C for (3.2). The smaller value of C leads to a shorter horizon (see Figure 1.3). Note that the chosen initial function $y_0(x) = \sin(\frac{\pi}{L}x)$ is the corresponding eigenfunction to the least eigenvalue and therefore the Friedrichs inequality in the theorems is tight. This explains why the turning point $\lambda_1 = 1$ is tight. For different initial functions it is possible to obtain shorter horizons for (3.9) even if $L > \pi$.

Furthermore, we see that $N_{\|y\|}$ grows for small L but then it remains constant. In contrast to this $N_{\|\nabla y\|}$ increases monotonically. We can use the previous theorems to explain this behaviour as well. For stage cost (3.2) the eigenvalue λ_1 (and therefore L) influences only the decay rate $\sigma = e^{-2T(\lambda_1 - M + K)}$ but not the overshoot bound $C = (1 + \lambda K^2)$. In order to ensure that $\sigma \in (0,1)$ the gain parameter K has to compensate $M - \lambda_1$. Thus, for small λ_1 we need a large parameter K and this leads to large C and a long horizon N. In the second row we see that there is a big decay (compared to M) from $\lambda_1 = \pi^2$ to $\lambda_1 = \pi^2/4$ for the interval length $L = 1$ and $L = 2$. For larger L the decay in λ_1 is comparatively small, the K and therefore the horizon N does not change considerably. The same argument holds true for the influence of λ_1 to σ in the case of stage cost (3.9). However, in contrast to (3.2) the overshoot bound $C = (1 + \frac{\lambda K^2}{\lambda_1})$ also depends on the eigenvalue λ_1. Since λ_1 converges to zero for growing interval length L we obtain a large constant C and thus we expect a large horizon N.

3.2 Introduction to Backstepping

In this section we introduce the backstepping approach to construct an exponential stabilizing feedback for the one dimensional heat equation with either Dirichlet or Neumann boundary control. One advantage of this method is that the resulting feedback is given as an explicit formula. We consider

$$y_t(x,t) = y_{xx}(x,t) + \mu y(x,t) \qquad \text{in} \quad (0,1) \times (0,\infty) \tag{3.15a}$$

$$y(0,t) = 0 \qquad \text{in} \quad (0,\infty) \tag{3.15b}$$

with either Dirichlet control

$$y(1,t) = u(t) \qquad \text{in} \quad (0,\infty) \tag{3.16a}$$

or Neumann control

$$y_x(1,t) = u(t) \qquad \text{in} \quad (0,\infty) \tag{3.16b}$$

on the right boundary ($x = 1$). On the left boundary ($x = 0$) we impose a homogeneous Dirichlet condition. In the derivation of the feedback we follow [62].
The idea behind the method of backstepping is to transform (3.15) into a so called *target system* which is exponentially stable. The coordinate transformation is done with a Volterra kernel

$$w(x,t) = y(x,t) - \int_0^x k(x,z)y(z,t)\, dz. \tag{3.17}$$

A suitable target system is given by

$$w_t(x,t) = w_{xx}(x,t) + (\mu - K)w(x,t) \tag{3.18a}$$

$$w(0,t) = 0 \tag{3.18b}$$

with $w(1,t) = 0$ for (3.16a) and $w_x(1,t) = 0$ for (3.16b). From (3.17) it follows directly that (3.18b) is fulfilled for arbitrary $k(x,z)$. We differentiate (3.17) with respect to x and t to derive the Volterra kernel $k(x,z)$. For this purpose we introduce the following notation

$$\begin{aligned} k_x(x,x) &= \frac{\partial}{\partial x} k(x,z)|_{z=x}, \\ k_z(x,x) &= \frac{\partial}{\partial z} k(x,z)|_{z=x}, \\ \frac{d}{dx} k(x,x) &= k_x(x,x) + k_z(x,x) \end{aligned}$$

By using the Leibniz differentiation rule we obtain

$$w_x(x,t) = y_x(x,t) - k(x,x)y(x,t) - \int_0^x k_x(x,z)y(z,t)\, dz \tag{3.19}$$

and thus

$$\begin{aligned} w_{xx}(x,t) =& y_{xx}(x,t) - y(x,t)\frac{d}{dx}k(x,x) - k(x,x)y_x(x,t) - k_x(x,x)y(x,t) \\ &- \int_0^x k_{xx}(x,z)y(x,t)\, dz. \end{aligned} \tag{3.20}$$

Note that we have not used the PDE (3.15) yet, but only the Volterra transformation (3.17). Differentiating (3.17) with respect to t yields

$$\begin{aligned} w_t(x,t) =& y_t(x,t) - \int_0^x k(x,z)y_t(z,t)\, dz \\ =& y_{xx}(x,t) + \mu y(x,t) - \int_0^x k(x,z)(y_{zz}(z,t) + \mu y(z,t))\, dz \\ =& y_{xx}(x,t) + \mu y(x,t) - k(x,x)y_x(x,t) + k(x,0)y_x(0,t) + k_x(x,x)y(x,t) \\ &- k_x(x,0)y(0,t) - \int_0^x k_{zz}(x,z)u(z,t)\, dz - \int_0^x \mu k(x,z)y(z,t)\, dz \end{aligned} \tag{3.21}$$

where we used partial integration in the last equality. By combining (3.20), (3.21) and (3.17) to the target system (3.18) we obtain

$$\begin{aligned} w_t(x,t) - w_{xx}(x,t) - (\mu - K)w(x,t) =& \left(K + 2\frac{d}{dx}k(x,x)\right) y(x,t) \\ +k(x,0)y_x(0,t) + \int_0^x \left(k_{xx}(x,z) - k_{zz}(x,z) - Kk(x,z)\right) y(z,t)\, dz &\overset{!}{=} 0 \end{aligned}$$

In order to keep the right hand side to be zero we get the following PDE for the kernel $k(x,z)$

$$k_{xx}(x,z) - k_{zz}(x,z) = Kk(x,z) \tag{3.22a}$$

$$k(x,0) = 0 \tag{3.22b}$$

$$k(x,x) = -\frac{K}{2}x. \tag{3.22c}$$

This PDE is of wave equation type (also known as Klein-Gordon-Equation) with unusual boundary conditions. It can be shown that the PDE is well posed and the solution is given by

$$k(x,z) = -Kz\frac{I_1(\sqrt{K(x^2-z^2)})}{\sqrt{K(x^2-z^2)}} \tag{3.23}$$

where I_1 denotes the first modified Bessel function. Furthermore, the inverse transformation to (3.17) can be obtained in the same way. All together we get

$$\begin{aligned} w(x,t) &= y(x,t) + \int_0^x k_1(x,z)y(z,t)\,dz \\ &= y(x,t) + \int_0^x Kz\frac{I_1\left(\sqrt{K(x^2-z^2)}\right)}{\sqrt{K(x^2-z^2)}}y(z,t)\,dz \qquad (3.24) \\ y(x,t) &= w(x,t) + \int_0^x k_2(x,z)w(z,t)\,dz \\ &= w(x,t) - \int_0^x Kz\frac{J_1\left(\sqrt{K(x^2-z^2)}\right)}{\sqrt{K(x^2-z^2)}}w(z,t)\,dz \qquad (3.25) \end{aligned}$$

where J_1 denotes the classical Bessel function. To fulfill the homogeneous condition on the right we have to choose the feedback control as

$$u(t) = y(1,t) = \int_0^1 k(1,z)y(z,t)\,dz = -\int_0^1 Kz\frac{I_1\left(\sqrt{K(1-z^2)}\right)}{\sqrt{K(1-z^2)}}y(z,t)\,dz \tag{3.26}$$

for the Dirichlet controlled case and as

$$\begin{aligned} u(t) = y_x(1,t) &= k(1,1)y(1,t) + \int_0^1 k_x(1,z)y(z,t)\,dz \\ &= -\frac{K}{2}y(1,t) - \int_0^1 Kz\frac{I_2\left(\sqrt{K(1-z^2)}\right)}{1-z^2}y(z,t)\,dz \qquad (3.27) \end{aligned}$$

for the Neumann controlled PDE. This section can be summarized in the following theorem

Theorem 3.8
Define the linear and bounded operator $K : H^i(0,1) \to H^i(0,1), (i = 0,1,2)$ by

$$w(x,t) = (Ky)(x,t) = y(x,t) + \int_0^x k_1(x,z)y(z,t)\,dz$$

where $k_1(x,z)$ is given by (3.24). Then
1. K has a linear bounded inverse $K^{-1}: H^i(0,1) \to H^i(0,1), (i = 0,1,2)$, and
2. K converts the system (3.15) into (3.18).

Proof. For the proof of the first assertion and the transformation properties cf. [69]. The proof of the explicit formulas for the kernel $k_1(x,z)$ and $k_2(x,z)$ can be found in [84]. □

3.3 Dirichlet Boundary Control

3.3.1 Exponential Constants for Dirichlet Boundary Control

In this section we consider a boundary control problem for the one dimensional linear heat equation

$$\begin{aligned}
y_t(x,t) &= y_{xx}(x,t) + \mu y(x,t) && \text{in} \quad (0,1)\times(0,\infty) && (3.28a)\\
y(0,t) &= 0 && \text{in} \quad (0,\infty) && (3.28b)\\
y(1,t) &= u(t) && \text{in} \quad (0,\infty) && (3.28c)\\
y(x,0) &= y_0(x) && \text{in} \quad (0,1). && (3.28d)
\end{aligned}$$

There is a homogenous Dirichlet boundary condition on the left ($x = 0$) and Dirichlet boundary control on the right ($x = 1$). The general procedure is the same as in Section 3.1.1. In order to obtain a control that satisfies the controllability condition we use the technique of backstepping presented in the previous section. Note that the target system (3.18) has the same structure as the reduced PDE (3.5) we used in Theorem 3.4. Once again K denotes the gain parameter.
The following lemma is well known from the analysis of integral equations, cf. [31]. Since we are interested in an explicit representation of the boundedness constant we recapitulate the proof.

Lemma 3.9
Let $u(x) \in L^2(0,1)$ and $u(x) = v(x) + \int_0^x k_i(x,z)v(z)\,dz, (i = 1,2)$. Then the estimate

$$\|u(\cdot)\|^2_{L^2(0,1)} \le (1+L_i)^2 \|v(\cdot)\|^2_{L^2(0,1)} \tag{3.29}$$

holds with the constant $L_i := \left(\int_0^1 \int_0^x k_i(x,z)^2\,dz\,dx\right)^{1/2}$.

Proof. By using the Cauchy-Schwarz inequality we get

$$\begin{aligned}
\|u(\cdot)\|_{L^2(0,1)} &= \left\| v(\cdot) + \int_0^x k_i(x,z)v(z)\,dz \right\|_{L^2(0,1)} \qquad (3.30)\\
&\le \left(\int_0^1 v(x)^2\,dx\right)^{\frac{1}{2}} + \left(\int_0^1 \left(\int_0^x k_i(\cdot,z)v(z)\,dz\right)^2 dx\right)^{\frac{1}{2}}
\end{aligned}$$

$$\leq \left(\int_0^1 v(x)^2\,dx\right)^{\frac{1}{2}} + \left(\int_0^1 \left(\int_0^x k_i(x,z)^2\,dz\right)\left(\int_0^x v(z)^2\,dz\right)\,dx\right)^{\frac{1}{2}}$$
$$\leq \left(1 + \left(\int_0^1\int_0^x k_i(x,z)^2\,dz\,dx\right)^{\frac{1}{2}}\right)\|v(\cdot)\|_{L^2(0,1)} = (1+L_i)\|v(\cdot)\|_{L^2(0,1)}$$

with $L_i := \left(\int_0^1\int_0^x k_i(x,z)^2\,dz\,dx\right)^{1/2}$, which is finite since the operator is bounded in $L^2(0,1)$ (see Theorem 3.8). □

Similar to the case of distributed control we look at two different stage costs. Suitable costs for boundary control are e.g.

$$l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|u(nT)|^2. \tag{3.31}$$

This cost functional corresponds to (3.2).

Theorem 3.10
The Dirichlet boundary controlled heat equation (3.28) with control (3.26), $K > \mu - \lambda_1$, and stage cost (3.31) satisfies the exponential controllability condition (1.34). The corresponding constants are given by $\sigma = e^{-2T(\lambda_1-\mu+K)} \in (0,1)$ and $C = (1 + \lambda K^2\eta(K))\xi(K) \in \mathbb{R}$ with $\eta(K) := \int_0^1 \left(x\frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\right)^2 dx$ and $\xi(K) := (1+L_1(K))^2(1+L_2(K))^2$.

Proof. Since the target system (3.18) is a special case of the reduced PDE (3.5) we can use the information about the decay rate. By combining Theorem 3.4 and Lemma 3.9 we obtain

$$\begin{aligned}\|y(\cdot,t)\|^2_{L^2(\Omega)} &\leq (1+L_2)^2\|w(\cdot,t)\|^2_{L^2(\Omega)} \leq (1+L_2)^2 e^{-2t(\lambda_1-\mu+K)}\|w(\cdot,0)\|^2_{L^2(\Omega)}\\ &\leq (1+L_1)^2(1+L_2)^2 e^{-2t(\lambda_1-\mu+K)}\|y(\cdot,0)\|^2_{L^2(\Omega)}\end{aligned}$$

and thus

$$l^*(y(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} \leq \frac{1}{2}\xi(K)\sigma^n\|y(\cdot,0)\|^2_{L^2(\Omega)} = \xi(K)\sigma^n l^*(y_0) \tag{3.32}$$

with $\xi(K) := (1+L_1(K))^2(1+L_2(K))^2$ and $\sigma := e^{-2T(\lambda_1-\mu+K)}$. Moreover, we get

$$\begin{aligned}l(y(n),u(n)) &= \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|u(nT)|^2\\ &= \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\left|\int_0^1 Ky(x,nT)x\frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\,dx\right|^2\\ &\leq \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2} + \frac{\lambda}{2}K^2\|y(\cdot,nT)\|^2_{L^2}\int_0^1\left(x\frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\right)^2 dx\end{aligned}$$

$$= \tilde{C} l^*(y(n)) \tag{3.33}$$

with $\tilde{C} := (1 + \lambda K^2 \eta(K))$ and $\eta(K) := \int_0^1 \left(x \frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}} \right)^2 dx.$
By combining (3.32) and (3.33) we obtain

$$l(y(n), u(n)) \leq \tilde{C} l^*(y(n)) \leq \tilde{C} \xi(K) \sigma^n l^*(y_0) = C \sigma^n l^*(y_0) \tag{3.34}$$

with $C := \tilde{C} \xi(K)$. □

From Section 3.3.1 we know that an alternative choice for the stage cost is given by penalizing the gradient of the state. Thus, we consider

$$l(y(n), u(n)) = \frac{1}{2} \|y_x(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2} |u(nT)|^2. \tag{3.35}$$

Since there is still a homogenous Dirichlet condition on the left ($x = 0$) this cost also identifies the equilibrium $y \equiv 0$, i.e., $l(y, 0) = 0 \Leftrightarrow y = 0$. Now, we can prove a similar result to Theorem 3.10:

Theorem 3.11
The Dirichlet boundary controlled heat equation (3.28) with control (3.26), $K > \mu - \lambda_1$, and stage cost (3.35) satisfies the exponential controllability condition (1.34). The corresponding constants are given by $\sigma = e^{-2T(\lambda_1 - \mu + K)} \in (0,1)$ and $C = (1 + \frac{4\lambda K^2 \eta(K)}{\pi^2}) \xi(K) \in \mathbb{R}$ with $\eta(K) := \int_0^1 \left(x \frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}} \right)^2 dx$ and $\xi(K) := (1 + M_1(K))^2 (1 + M_2(K))^2$.

Proof. By using the Leibniz formula we differentiate the Volterra transformation (3.17) with respect to x and get

$$w_x(x,t) = y_x(x,t) - k(x,x) y(x,t) - \int_0^x k_x(x,z) y(z,t)\, dz. \tag{3.36}$$

Similar arguments as in Lemma 3.9 yield

$$\begin{aligned}
\|w_x(\cdot,t)\|_{L^2(\Omega)} \leq & \|y_x(\cdot,t)\|_{L^2(\Omega)} + \|k(\cdot,\cdot) y(\cdot,t)\|_{L^2(\Omega)} + \left\| \int_0^x k_x(\cdot,z) y(z,t)\, dz \right\|_{L^2(\Omega)} \\
\leq & \|y_x(\cdot,t)\|_{L^2(\Omega)} + \|K\frac{x}{2}\|_{L^2(\Omega)} \|y(\cdot,t)\|_{L^2(\Omega)} \\
& + \left\| \int_0^x k_x(\cdot,z)\, dz \right\|_{L^2(\Omega)} \|y(\cdot,t)\|_{L^2(\Omega)} \\
\leq & \left(1 + \left(\frac{K}{\sqrt{12}} + \left\| \int_0^x k_x(\cdot,z)\, dz \right\|_{L^2(\Omega)} \right) \frac{2}{\pi} \right) \|y_x(\cdot,t)\|_{L^2(\Omega)}
\end{aligned} \tag{3.37}$$

and thus

$$\|w_x(\cdot,t)\|^2_{L^2(\Omega)} \leq (1+M_1(K))^2 \|y_x(\cdot,t)\|^2_{L^2(\Omega)} \tag{3.38}$$

with $M_1(K) = (\frac{K}{\sqrt{12}} + (\int_0^1 \int_0^x k_x(x,z)^2\,dz\,dx)^{\frac{1}{2}})\frac{2}{\pi}$. From the inverse transformation we obtain $M_2(K)$ in the same way. In order to determine the decay rate we can use the information from Theorem 3.6. Together with the previous result we get

$$\begin{aligned}\|y_x(\cdot,t)\|^2_{L^2(\Omega)} &\leq (1+M_2)^2 \|w_x(\cdot,t)\|^2_{L^2(\Omega)} \leq (1+M_2)^2 e^{-2t(\lambda_1-\mu+K)} \|w_x(\cdot,0)\|^2_{L^2(\Omega)} \\ &\leq (1+M_1)^2(1+M_2)^2 e^{-2t(\lambda_1-\mu+K)} \|y_x(\cdot,0)\|^2_{L^2(\Omega)}\end{aligned}$$

and thus

$$l^*(y(n)) = \frac{1}{2}\|y_x(\cdot,nT)\|^2_{L^2(\Omega)} \leq \frac{1}{2}\xi(K)\sigma^n \|y_x(\cdot,0)\|^2_{L^2(\Omega)} = \xi(K)\sigma^n l^*(y_0) \tag{3.39}$$

with $\xi(K) := (1+M_1(K))^2(1+M_2(K))^2$ and $\sigma := e^{-2T(\lambda_1-\mu+K)}$. Moreover, we get

$$\begin{aligned}l(y(n),u(n)) &= \frac{1}{2}\|y_x(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\left|u(nT)\right|^2 \\ &= \frac{1}{2}\|y_x(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\left|\int_0^1 Ky(x,nT)x\frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\,dx\right|^2 \\ &\leq \frac{1}{2}\|y_x(\cdot,nT)\|^2_{L^2} + \frac{\lambda}{2}K^2\|y(\cdot,nT)\|^2_{L^2}\int_0^1\left(x\frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\right)^2 dx \\ &\leq \tilde{C}l^*(y(n))\end{aligned} \tag{3.40}$$

with $\tilde{C} := (1+\frac{4\lambda K^2\eta(K)}{\pi^2})$ and $\eta(K) := \displaystyle\int_0^1 \left(x\frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\right)^2 dx.$
By combining (3.39) and (3.40) we obtain

$$l(y(n),u(n)) \leq \tilde{C}l^*(y(n)) \leq \tilde{C}\xi(K)\sigma^n l^*(y_0) = C\sigma^n l^*(y_0) \tag{3.41}$$

with $C := \tilde{C}\xi(K)$. □

3.3.2 Numerical Results

Now, we want to show how the previous results can be used to explain the numerical observations. As a first step we compare the results for boundary control with the corresponding results for the distributed case. Afterwards, we analyse the difference between the stage costs (3.31) and (3.35).
In order to compare the Dirichlet result to that from the distributed case we look at the main difference between the constants in Theorem 3.4 and Theorem 3.10. Since we choose the target system (3.18) which is similar to the reduced system (3.5), we obtain the same decay rate σ for both systems. The difference is due to

the occurrence of the K dependent functions $\eta(K)$ and $\xi(K)$. In order to explain the influence of these functions on the optimization horizon we compare the heat equation with distributed control (3.12) with the boundary controlled heat equation (3.28) in terms of the regularization parameter λ and the reaction parameter μ.

In Table 3.3 the results of the numerical simulation are presented. The minimal

	$\lambda = 0.001$		$\lambda = 0.01$		$\lambda = 0.05$	
μ	N_D	N_B	N_D	N_B	N_D	N_B
11	2	2	3	2	7	2
12	2	3	4	3	11	3
13	2	3	4	3	14	4
14	2	4	5	4	15	5
15	2	5	6	5	16	6
16	2	6	6	6	16	7
17	2	7	7	7	16	8
18	2	7	7	7	16	8
19	2	8	8	8	16	9

Table 3.3: Minimal stabilizing horizon for distributed N_D and boundary N_B control with varying λ and μ determined by numerical simulations of the MPC closed loop

stabilizing horizon observed in the MPC closed loop is denoted by N_D for the distributed controlled PDE (3.12) and by N_B for the Dirichlet boundary controlled PDE (3.28). First, one observes that the shortest possible horizon ($N = 2$) is sufficient to guarantee stability for the distributed controlled case if $\lambda = 0.001$ and $\mu \leq 19$. This leads to the question whether for each μ it is possible to find a λ that is sufficient for stability with $N = 2$. To answer this question we again look at the overshoot constant $C = (1 + \lambda K^2)$ for the distributed case. Obviously, it is possible for each fixed μ (and therefore K) to bring C arbitrarily close to 1 by reducing the parameter λ. Thus, we obtain stability with an optimization horizon $N = 2$ (see Figure 1.3). In contrast to this, for $\lambda = 0.001$ we do not get stability with this horizon for the boundary controlled equation for arbitrary reaction parameter μ. (In fact we do not obtain stability with $N = 2$ for arbitrarily small regularization parameter λ.) In order to explain why this is in general not possible we have to take a closer look at the overshoot bound $C = (1 + \lambda K^2 \eta(K))\xi(K)$. For arbitrarily small values of λ the constant converges to $C = \xi(K)$. Since $\xi(K)$ is an increasing function with $\xi(K) > 1$ for $K > 0$ we cannot expect stability with $N = 2$ for general values of μ, not even for arbitrarily small λ. Of course, it is possible to reach stability with this horizon for particularly small values of μ, e.g. if $\lambda = 0.001$ and $\mu = 11$. This is due to the fact that $N = 2$ is also possible if $C < 2$ and σ small enough (see Figure 1.3). In Chapter 6 we will see that for the boundary controlled PDE this *overshooting* is natural and actually observable, see Figure 6.5 (right).

Furthermore, we observe that the horizon for the distributed controlled PDE is very sensitive with respect to the regularization parameter λ: for increasing values of λ the minimal stabilizing horizon grows dramatically. This can be explained by the influence of λ on the overshoot bound: an increasing value of λ results directly in an increasing C and therefore in a larger horizon. This leads to the question why this occurs for the boundary controlled PDE only in a moderate way. In order to give an answer to this question we have to take a closer look at the function $\eta(K)$. $\eta(K)$ is an increasing function with $\eta(0) = 0$ and $\eta(22.81) \approx 1$. Since $\mu < 20$ and therefore $K < 20$ we obtain $\eta(K) < 1$, i.e., the function damps the influence of λ on the overshoot constant.
In the second step we consider the influence of the different stage costs (3.31) and (3.35) on the stabilizing horizon. In Section 3.1.2 we observed (for $L = 1$) that penalizing the gradient of the state (stage cost (3.9)) leads to a significant shorter horizon than penalizing the state itself (stage cost (3.2)). We want to answer the question whether we obtain the same behaviour for the boundary controlled problem. First, we observe that we have the same decay rate σ for both stage costs. Furthermore, both overshoot constants are influenced by the same function $\eta(K)$. Therefore, the above arguments concerning the sensitivity on the regularization parameter λ are also true for the cost (3.35). The differences between the stage costs are due to the function $\xi(K)$. Since both have the same structure $\xi(K) := (1 + f_1(K))^2(1 + f_2(K))^2$ we have to take a closer look at the functions $L_1(K), L_2(K)$ and $M_1(K), M_2(K)$ respectively. A numerical evaluation shows that $M_1(K) \gg L_1(K)$ and $M_2(K) \gg L_2(K)$. Therefore, we expect a converse behaviour as for the distributed control problem: for the boundary controlled PDE (3.28) the stage cost (3.31) leads to shorter stabilizing horizons than stage cost (3.35). This corresponds to the observation in the numerical example. For stage cost (3.31), $\lambda = 0.001$ and $\mu = 11$ we need $N = 2$ to get stability (Table 3.3). In contrast to this the minimal stabilizing horizon for stage cost (3.35) is at least $N = 29$, even for this comparatively small reaction parameter.

3.4 Neumann Boundary Control

In this section we look at the linear heat equation with Neumann boundary control. In contrast to the previous section we distinguish between small and large values of the reaction parameter μ. In Subsection 3.4.1 we will derive the exponential constants for the n-dimensional heat equation for sufficiently small values of μ. Afterwards, we consider the case for arbitrary values of μ. In order to construct a control we use the method of backstepping presented in Section 3.2. This technique only works for the one dimensional case. In Subsection 3.4.3 we will demonstrate that the different properties of the controls are reflected in the numerical simulation. The exponential estimates in this section differ from the previous sections: the overshoot constant depends on a generic constant, i.e., the formulas cannot be used to determine explicit horizons, but they allow us to give qualitative explanations.

3.4.1 Exponential Constants for Neumann Boundary Control I

In this section we consider the n-dimensional heat equation with Neumann boundary control

$$
\begin{aligned}
y_t(x,t) &= \Delta y(x,t) + \mu y(x,t) && \text{in} \quad \Omega \times (0,\infty) && \text{(3.42a)}\\
\partial_\nu y(x,t) &= u(x,t) && \text{on} \quad \Gamma_N \times (0,\infty) && \text{(3.42b)}\\
y(x,t) &= 0 && \text{on} \quad \Gamma_D \times (0,\infty) && \text{(3.42c)}\\
y(x,0) &= y_0(x) && \text{in} \quad \Omega && \text{(3.42d)}
\end{aligned}
$$

with $\partial\Omega = \Gamma_N \cup \Gamma_D$, $\Gamma_N \cap \Gamma_D = \emptyset$ and $|\Gamma_N| \neq \emptyset$. $|\Gamma_D| \neq \emptyset$ ensures that the origin is the only equilibrium. The stage cost for this problem are given by

$$
l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot,nT)\|_{L^2(\Gamma_N)} \tag{3.43}
$$

It is well known that the feedback

$$
\partial_\nu y(x,t) = -Ky(x,t) \quad \text{on} \quad \Gamma_N \times (0,\infty) \tag{3.44}
$$

can stabilize the problem for small values of the reaction parameter μ, i.e., $\mu < C_D$, see Theorem 3.12. In order to prove a similar result to Theorem 3.4 we need the following generalization of the Friedrichs inequality

Theorem 3.12
For $u \in H^1(\Omega)$ the following inequality holds

$$
\int_\Omega |\nabla u|^2 \, dx + s\int_{\partial\Omega} |u|^2 \, dS \geq C_F(s)\int_\Omega |u|^2 \, dx \tag{3.45}
$$

where the optimal Friedrichs constant $C_F(s)$ is given by the least eigenvalue of

$$
\begin{aligned}
-\Delta u &= \mu u \quad && \text{in} \quad \Omega\\
\partial_\nu u + su &= 0 && \text{on} \quad \partial\Omega.
\end{aligned}
$$

Moreover, $C_F(s)$ is increasing and

$$
\lim_{s\to\infty} C_F(s) = C_D \tag{3.46}
$$

where C_D is the least eigenvalue of the Dirichlet eigenproblem

$$
\begin{aligned}
-\Delta u &= \mu u \quad && \text{in} \quad \Omega\\
u &= 0 && \text{on} \quad \partial\Omega.
\end{aligned}
$$

Proof. A generalization of the first result is proven in [12]. A proof for the second result can be found in [13]. □

Moreover, we need

Corollary 3.13
For $u \in H^2(\Omega)$ with $\partial_\nu u(x) = -Ku(x), x \in \partial\Omega, K \in \mathbb{R}^+$, the following inequality holds

$$C_F(K)\left[\int_\Omega |\nabla u(x)|^2\,dx + K\int_{\partial\Omega} |u(x)|^2\,dS\right] \le \int_\Omega (\Delta u(x))^2\,dx$$

with the optimal Friedrichs constant given in Theorem 3.12.

Proof. By using integration by parts and the Cauchy-Schwarz inequality we obtain

$$\begin{aligned}
&\int_\Omega |\nabla u(x)|^2\,dx + K\int_{\partial\Omega}|u(x)|^2\,dS\\
&= \int_{\partial\Omega} u(x)\partial_\nu u(x)\,dS - \int_\Omega u(x)\Delta u(x)\,dx + K\int_{\partial\Omega}|u(x)|^2\,dS\\
&\le \left[\int_\Omega u(x)^2\,dx\right]^{\frac{1}{2}}\left[\int_\Omega (\Delta u(x))^2\,dx\right]^{\frac{1}{2}}\\
&\le \frac{1}{\sqrt{C_F(K)}}\left[\int_\Omega |\nabla u(x)|^2\,dx + K\int_{\partial\Omega}|u(x)|^2\,dS\right]^{\frac{1}{2}}\left[\int_\Omega (\Delta u(x))^2\,dx\right]^{\frac{1}{2}}
\end{aligned}$$

and thus

$$\sqrt{C_F(K)}\left[\int_\Omega |\nabla u(x)|^2\,dx + K\int_{\partial\Omega}|u(x)|^2\,dS\right]^{\frac{1}{2}} \le \left[\int_\Omega (\Delta u(x))^2\,dx\right]^{\frac{1}{2}}.$$

□

Theorem 3.14
Let $\mu < C_D$. Then the Neumann controlled heat equation (3.42) with control (3.44), where K is chosen such that $C_F(K) > \mu$, and stage cost (3.43) satisfies the exponential controllability condition (1.34). The corresponding constants are given by $\sigma = e^{-2T(C_F(K)-\mu)} \in (0,1)$ and $C = (1+\lambda K^2 M) \in \mathbb{R}$ with $M \in \mathbb{R}$.

Proof. In order to use the Lyapunov methods from the previous theorems we define

$$V(t) := \frac{1}{2}\|y(\cdot,t)\|^2_{L^2(\Omega)} = \frac{1}{2}\int_\Omega y(x,t)^2\,dx \tag{3.47}$$

and differentiate $V(t)$ with respect to time

$$\begin{aligned}
\dot V(t) &= \int_\Omega y(x,t)y_t(x,t)\,dx = \int_\Omega y(x,t)(\Delta y(x,t) + \mu y(x,t))\,dx\\
&= \int_{\Gamma_N} y(x,t)\partial_\nu y(x,t)\,dS - \int_\Omega \nabla y(x,t)\cdot\nabla y(x,t)\,dx + \mu\int_\Omega y(x,t)^2\,dx\\
&= -K\int_{\Gamma_N} y(x,t)^2\,dS - \int_\Omega \nabla y(x,t)\cdot\nabla y(x,t)\,dx + \mu\int_\Omega y(x,t)^2\,dx\\
&\le (-C_F(K)+\mu)\int_\Omega y(x,t)^2\,dx = -2(C_F(K)-\mu)V(t)
\end{aligned}$$

and thus

$$V(t) \leq e^{-2t(C_F(K)-\mu)}V(0). \tag{3.48}$$

Because of Theorem 3.12 and $\mu < C_D$ there exists $K \in \mathbb{R}$ such that $C_F(K) > \mu$. This yields

$$l^*(y(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} \leq \frac{1}{2}e^{-2nT(C_F(K)-\mu)}\|y(\cdot,0)\|^2_{L^2(\Omega)} = \sigma^n l^*(y_0)$$

with $\sigma = e^{-2T(C_F(K)-\mu)} \in (0,1)$. In order to prove the exponential decay of the derivative we define similarly to (3.10)

$$V(t) := \frac{1}{2}\|\nabla y(\cdot,t)\|^2_{L^2(\Omega)} = \frac{1}{2}\int_\Omega \nabla y(x,t)\cdot\nabla y(x,t)\;dx$$

and differentiate with respect to time

$$\begin{aligned}
\dot{V}(t) =& \int_\Omega \nabla y(x,t)\cdot\nabla y_t(x,t)\;dx\\
=& -\int_\Omega \Delta y(x,t) y_t(x,t)\;dx = -\int_\Omega (\Delta y(x,t))^2\;dx - \mu\int_\Omega \Delta y(x,t)y(x,t)\;dx\\
\leq& -C_F(K)\left[\int_\Omega |\nabla y(x,t)|^2\;dx + K\int_{\Gamma_N} y(x,t)^2\;dS\right]\\
&-\mu\int_{\Gamma_N} y(x,t)\partial_\nu y(x,t)\;dS + \mu\int_\Omega |\nabla y(x,t)|^2\;dx\\
\leq& -(C_F(K)-\mu)\int_\Omega |\nabla y(x,t)|^2\;dx - \underbrace{(C_F(K)-\mu)}_{\geq 0}K\int_{\Gamma_N} y(x,t)^2\;dS\\
\leq& -2(C_F(K)-\mu)V(t).
\end{aligned}$$

Thus, we get

$$\frac{1}{2}\|\nabla y(\cdot,t)\|^2_{L^2(\Omega)} \leq \frac{1}{2}e^{-2t(C_F(K)-\mu)}\|\nabla y(\cdot,0)\|^2_{L^2(\Omega)}$$

By combining the previous results with the trace theorem, cf. [32], we can prove the boundedness of the control effort

$$\int_{\partial\Omega} y(x,t)^2\;dS \leq C\|y(\cdot,t)\|^2_{H^1(\Omega)} \leq C\|y_0(\cdot)\|^2_{H^1(\Omega)}.$$

Therefore, there exists a constant $M \in \mathbb{R}$ such that $\|y(\cdot,t)\|^2_{L^2(\Gamma_N)} \leq M\|y(\cdot,t)\|^2_{L^2(\Omega)}$ and we determine the overshoot constant by

$$\begin{aligned}
l(y(n),u(n)) &= \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot,nT)\|^2_{L^2(\Gamma_N)}\\
&= \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}K^2\|y(\cdot,nT)\|^2_{L^2(\Gamma_N)}\\
&\leq Cl^*(y(n)) \quad \text{with} \quad C := (1+\lambda K^2 M).
\end{aligned}$$

□

3.4.2 Exponential Constants for Neumann Boundary Control II

In this subsection we consider the Neumann controlled PDE (3.42) with an arbitrarily large reaction parameter μ. Since the backstepping approach only works in one dimension we investigate

$$\begin{aligned} y_t(x,t) &= y_{xx}(x,t) + \mu y(x,t) && \text{in} \quad (0,1)\times(0,\infty) && \text{(3.49a)}\\ y(0,t) &= 0 && \text{in} \quad (0,\infty) && \text{(3.49b)}\\ y_x(1,t) &= u(t) && \text{in} \quad (0,\infty) && \text{(3.49c)}\\ y(x,0) &= y_0(x) && \text{in} \quad (0,1). && \text{(3.49d)} \end{aligned}$$

As stage cost we use the one dimensional version of (3.43), i.e.,

$$l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|u(nT)|^2. \tag{3.50}$$

Theorem 3.15
The Neumann controlled heat equation (3.49) with control (3.27), where K is chosen such that $K > \mu - \lambda_1$, and stage cost (3.50) satisfies the exponential controllability condition (1.34). The corresponding constants are given by $\sigma = e^{-2T(\lambda_1-\mu+K)} \in (0,1)$ and $C = \left(1 + \lambda K^2\left(\frac{\sqrt{M}}{2} + \tilde{\eta}(K)\right)^2\right)\xi(K) \in \mathbb{R}$ with

$$\tilde{\eta}(K) := \left(\int_0^1 \left(x\frac{I_2(\sqrt{K(1-x^2)})}{(1-x^2)}\right)^2 dx\right)^{\frac{1}{2}}$$

and $\xi(K) := (1+L_1(K))^2(1+L_2(K))^2$.

Proof. First, we note that the Volterra kernel for the Neumann control is exactly the same as in the Dirichlet case (only the feedback control and the eigenvalue change). With the same calculation as in Theorem 3.10 we get

$$l^*(y(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} \le \frac{1}{2}\xi(K)\sigma^n\|y(\cdot,0)\|^2_{L^2(\Omega)} = \xi(K)\sigma^n l^*(y_0) \tag{3.51}$$

with $\xi(K) := (1+L_1(K))^2(1+L_2(K))^2$ and $\sigma := e^{-2T(\lambda_1-\mu+K)}$. With similar arguments as in Theorem 3.10 and Lemma 3.9 we obtain

$$\begin{aligned} |u(t)| &\le \frac{K}{2}\,|y(1,t)| + \left|\int_0^1 Kz\frac{I_2\left(\sqrt{K(1-z^2)}\right)}{1-z^2}y(z,t)\,dz\right| \\ &\le \frac{K\sqrt{M}}{2}\left(\int_0^1 y(x,t)^2\,dx\right)^{\frac{1}{2}} \\ &\quad + K\left(\int_0^1 y(x,t)^2\,dx\right)^{\frac{1}{2}}\left(\int_0^1\left(z\frac{I_2\left(\sqrt{K(1-z^2)}\right)}{1-z^2}\right)^2 dz\right)^{\frac{1}{2}} \end{aligned}$$

$$\leq \left(\frac{\sqrt{M}}{2} + \tilde{\eta}(K)\right) K \left(\int_0^1 y(x,t)^2\, dx\right)^{\frac{1}{2}}$$

This yields

$$l(y(n), u(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|u(nT)|^2 \leq \tilde{C} l^*(y(n)) \tag{3.52}$$

with $\tilde{C} = \left(1 + \lambda K^2 \left(\frac{\sqrt{M}}{2} + \tilde{\eta}(K)\right)^2\right)$. By combining (3.51) and (3.52) we obtain

$$l(y(n), u(n)) \leq \tilde{C} l^*(y(n)) \leq \tilde{C}\xi(K)\sigma^n l^*(y_0) = C\sigma^n l^*(y_0) \tag{3.53}$$

with $C := \tilde{C}\xi(K)$. □

It should be mentioned that the smallest eigenvalue in the previous theorem is given by $\lambda_1 = \pi^2/4$ and, thus, is different to that in the Dirichlet case where we have $\lambda_1 = \pi^2$.

3.4.3 Numerical Results

In this section we investigate the Neumann controlled heat equation in the context of our theoretical results from the previous section. The case of Neumann control is remarkable because we obtained different exponential constants for different magnitudes of the reaction value μ. This is the opposite behaviour to the distributed and the Dirichlet boundary control. The focus of this section is on the investigation whether the different controls from Section 3.4.1 and 3.4.2 result in a different behaviour of the minimal stabilizing horizon.
For this purpose we consider the Neumann controlled equation in one dimension (3.49). From the spectral properties we know that the uncontrolled equation is unstable for $\mu \geq \pi^2/4$. The smallest eigenvalue of the corresponding Dirichlet problem is given by $C_D = \pi^2$. Thus, for $\pi^2/4 \leq \mu < \pi^2$ we can stabilize (3.49) with control (3.44). Moreover, we can use the results from Theorem 3.14. For larger values of the reaction parameter ($\mu \geq \pi^2$) we have to use the weaker results from Theorem 3.15. We focus again on the analysis of the overshoot bound. It is visible that the overshoot constant in Theorem 3.14 ($C = (1 + \lambda K^2 M)$) has a very similar structure to that in the distributed controlled case in Theorem 3.4 ($C = (1 + \lambda K^2)$). In contrast to this the overshoot constant for large values of μ ($C = (1 + \lambda K^2 (\frac{\sqrt{M}}{2} + \tilde{\eta}(K))^2)\xi(K)$) resembles that of the Dirichlet controlled case ($C = (1 + \lambda K^2 \eta(K))\xi(K)$). Therefore, from our theory we expect a Dirichlet like behaviour for $\mu \geq \pi^2$ and a distributed control like behaviour for $\pi^2/4 \leq \mu < \pi^2$. This is exactly what we observe in the numerical simulation.
First, we recapitulate the major differences between Dirichlet and distributed control from Subsection 3.3.2. One observation was that it is always possible to stabilize the heat equation with distributed control with the smallest horizon by reducing the

value of the regularization parameter λ. Obviously, the arguments from Subsection 3.3.2 are also true for the Neumann controlled equation with $\mu < \pi^2$: We can bring $C = (1 + \lambda K^2 M)$ arbitrarily close to 1 by reducing the value of λ. Thus, it is possible to stabilize the problem with a horizon $N = 2$ for sufficiently small λ. Since the control (3.44) does not stabilize the problem for a reaction parameter $\mu \geq \pi^2$ we have to choose the control (3.27) and we can use the results from Theorem 3.15. With the same arguments as for the Dirichlet controlled system we have $C \to \xi(K)$ for $\lambda \to 0$. Thus, for $\mu \geq \pi^2$ we cannot guarantee stability with a horizon $N = 2$ even for arbitrarily small λ.
An interesting question is whether the bound π^2 is actually observable in the numerical simulation. To answer the question we have to keep the influence of the decay rate σ small. One possibility to do this is to reduce the sampling time T. For $T \to 0$ we obtain $\sigma \to 1$ and, thus, we can observe the pure influence of C. In Figure 3.1 the maximum value of μ is displayed where stability is obtained with $N = 2$ depending on the sampling time T (blue line). Furthermore, we see the bound π^2 derived from the theory (dashed black line). It is observable that the curve tends to the theoretical bound for small values of T. For even smaller values of T, the numerical errors become predominant.

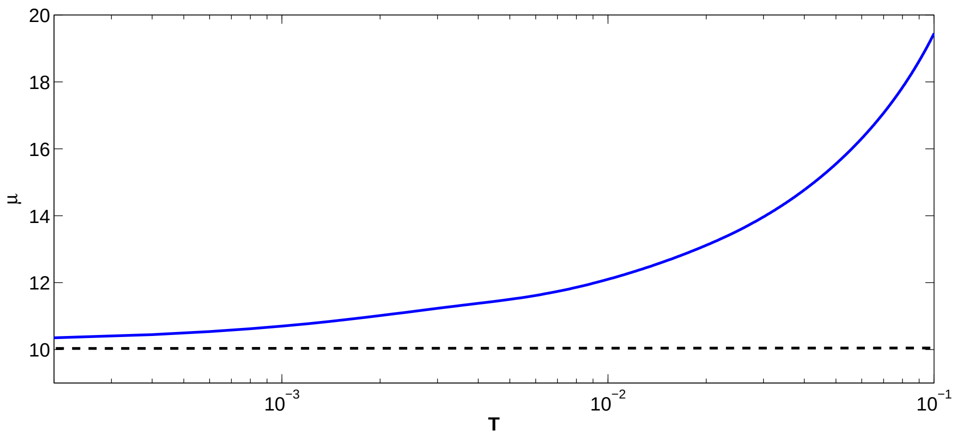


Figure 3.1: Maximum value of μ where stability with $N = 2$ is observed in the numerical simulation, in dependence of the sampling time T.

Furthermore, we have seen in Section 3.3.2 that in the Dirichlet case the stage cost, where the gradient of the state is penalized, leads to a much longer horizon than the stage cost, where the state is penalized. In the case of distributed control we observe the contrary behaviour for $L = 1$ (see Table 3.2). In the next example we investigate this problem for the Neumann controlled case. We consider the one dimensional heat equation (3.49) with stage cost

$$l(y(n), u(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|u(nT)|^2 \tag{3.54}$$

and

$$l(y(n), u(n)) = \frac{1}{2}\|y_x(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|u(nT)|^2 \tag{3.55}$$

Following our analysis we expect that stage cost (3.55) yields shorter stabilizing horizons than (3.54) for $\mu < \pi^2$ ('distributed control like behaviour'). This property changes to the opposite for $\mu \geq \pi^2$ ('Dirichlet control like behaviour'). In the numerical simulation we choose the parameters $T = \lambda = 0.01$. The results are presented in Table 3.4. We denote the minimal stabilizing horizon for stage cost (3.54) and (3.55) with $N_{\|y\|}$ and $N_{\|\nabla y\|}$, respectively. Obviously, the required horizon for the cost (3.55) is much smaller than for (3.54) in the case of a reaction parameter up to $\mu = 9$. This behaviour changes drastically for values above $\mu = 10$. Thus, the numerical results perfectly match our theoretical findings. Once again we observe that our theoretical bound $\mu = \pi^2$ is actually tight.

μ	$N_{\|y\|}$	$N_{\|\nabla y\|}$
5	3	2
6	4	2
7	4	2
8	5	2
9	6	2
10	6	16
11	7	33

Table 3.4: Comparison of the minimal stabilizing horizon for stage costs (3.54) $N_{\|y\|}$ and (3.55) $N_{\|\nabla y\|}$ depending on the reaction parameter μ. The parameters for the numerical MPC simulation are given by $T = \lambda = 0.01$.

3.5 Convection

In this section we examine the heat equation with an additional convection term and a diffusion parameter $\varepsilon > 0$. Although after slightly changes the presented methods are also applicable to distributed and Neumann boundary control we restrict ourselves to the case of Dirichlet control. Thus, we look at the Dirichlet boundary controlled PDE (3.28) with additional convection term

$$\tilde{y}_t(x,t) = \varepsilon\tilde{y}_{xx}(x,t) + b\tilde{y}_x(x,t) + \mu\tilde{y}(x,t) \quad \text{in} \quad (0,1)\times(0,\infty) \tag{3.56a}$$

$$\tilde{y}(0,t) = 0 \quad \text{in} \quad (0,\infty) \tag{3.56b}$$

$$\tilde{y}(1,t) = \tilde{u}(t) \quad \text{in} \quad (0,\infty) \tag{3.56c}$$

$$\tilde{y}(x,0) = \tilde{y}_0(x) \quad \text{in} \quad (0,1) \tag{3.56d}$$

with $b \in \mathbb{R}$. It is well known from the theory of convection dominated diffusion equations that it is possible to transform (3.56) into a PDE without convection, cf.

[61]. For the one dimensional equation (3.56) the transformation is given by

$$y(x,t) := \tilde{y}(x,t)e^{\frac{b}{2\varepsilon}x}. \tag{3.57}$$

To prove that this transformation actually eliminates the convection we differentiate $y(x,t)$ with respect to t and x

$$\begin{aligned} y_t(x,t) &= \tilde{y}_t(x,t)e^{\frac{b}{2\varepsilon}x} \\ y_{xx}(x,t) &= \tilde{y}_{xx}(x,t)e^{\frac{b}{2\varepsilon}x} + \frac{b}{\varepsilon}\tilde{y}_x(x,t)e^{\frac{b}{2\varepsilon}x} + \frac{b^2}{4\varepsilon^2}\tilde{y}(x,t)e^{\frac{b}{2\varepsilon}x} \end{aligned}$$

and get

$$\begin{aligned} &y_t(x,t) - \varepsilon y_{xx}(x,t) - \left(\mu - \frac{b^2}{4\varepsilon^2}\right) y(x,t) \\ &= \tilde{y}_t(x,t)e^{\frac{b}{2\varepsilon}x} - \varepsilon\tilde{y}_{xx}(x,t)e^{\frac{b}{2\varepsilon}x} - b\tilde{y}_x(x,t)e^{\frac{b}{2\varepsilon}x} - \frac{b^2}{4\varepsilon^2}\tilde{y}(x,t)e^{\frac{b}{2\varepsilon}x} - \left(\mu - \frac{b^2}{4\varepsilon^2}\right)\tilde{y}(x,t)e^{\frac{b}{2\varepsilon}x} \\ &= 0 \end{aligned}$$

Therefore, we obtain the heat equation without convection but with an altered reaction term

$$\begin{aligned} y_t(x,t) &= \varepsilon y_{xx}(x,t) + \left(\mu - \frac{b^2}{4\varepsilon^2}\right) y(x,t) && \text{in} \quad (0,1)\times(0,\infty) && (3.58a) \\ y(0,t) &= 0 && \text{in} \quad (0,\infty) && (3.58b) \\ y(1,t) &= u(t) = \tilde{u}(t)e^{\frac{b}{2\varepsilon}} && \text{in} \quad (0,\infty) && (3.58c) \\ y(x,0) &= \tilde{y}_0(x)e^{\frac{b}{2\varepsilon}x} && \text{in} \quad (0,1). && (3.58d) \end{aligned}$$

Remark 3.16
Note that this method to eliminate the convection term is not restricted to the one dimensional equation. Furthermore, the idea can be generalized to spatially varying functions $\varepsilon(x)$ and $b(x)$, cf. [62]. However, in these cases the transformation becomes much more complicated.
In the FEM literature the transformation trick is also known as Lagrange-Galerkin method, cf. [61]. The name Lagrange originates from the Lagrangian description in fluid dynamics where the observer moves with the fluid.

Now, we are in the setting of Section 3.3 and can apply the known results to (3.58). Analogous to (3.18) the desired target system for the PDE (3.58) is given by

$$\begin{aligned} w_t(x,t) &= \varepsilon w_{xx}(x,t) + \left(\mu - \frac{b^2}{4\varepsilon^2} - K\right) w(x,t) && \text{in} \quad (0,1)\times(0,\infty) && (3.59a) \\ w(0,t) &= 0 && \text{in} \quad (0,\infty) && (3.59b) \\ w(1,t) &= 0 && \text{in} \quad (0,\infty) && (3.59c) \end{aligned}$$

with the control feedback

$$\tilde{u}(t) = -\int_0^1 e^{-\frac{b}{2\varepsilon}(1-x)} K\tilde{y}(x,t)x \frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\, dx. \tag{3.60}$$

By applying the results from Section 3.3 to system (3.59) we directly obtain $\sigma = e^{-2T(\pi^2-\mu+K+\frac{b^2}{4\varepsilon^2})}$. In order to get an estimate for the overshoot constant C we have to investigate the feedback law (3.60) in more detail

$$\begin{aligned}
\tilde{u}(nT)^2 &= \left(\int_0^1 e^{-\frac{b}{2\varepsilon}(1-x)} K\tilde{y}(x,nT)x \frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\, dx\right)^2 \\
&\le \int_0^1 \left(e^{-\frac{b}{2\varepsilon}(1-x)} K\tilde{y}(x,nT)\right)^2 dx \cdot \int_0^1 \left(x\frac{I_1(\sqrt{K(1-x^2)})}{\sqrt{K(1-x^2)}}\right)^2 dx \\
&\le K^2 \int_0^1 e^{-\frac{b}{\varepsilon}(1-x)} \tilde{y}(x,nT)^2\, dx \cdot \eta(K) \\
&\le K^2 \int_0^1 \tilde{y}(x,nT)^2\, dx \cdot \eta(K) \cdot \begin{cases} 1 & : \quad b \ge 0 \\ e^{\frac{|b|}{\varepsilon}} & : \quad b < 0 \end{cases}
\end{aligned}$$

For stage cost (3.31) we obtain

$$l(y(n),u(n)) = \frac{1}{2}\|\tilde{y}(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|\tilde{u}(nT)|^2 \le \tilde{C}l^*(y(n) \tag{3.61}$$

where $\tilde{C}$ is given by

$$\tilde{C} := 1 + \lambda K^2 \eta(K) \cdot \begin{cases} 1 & : \quad b \ge 0 \\ e^{\frac{|b|}{\varepsilon}} & : \quad b < 0 \end{cases}$$

With the same arguments as in Theorem 3.10 we get $C = \tilde{C}\xi(K)$.

Numerical results

In the next step, we want to analyse the reaction-convection-diffusion equation (3.56) with regard to the convection parameter b. From the decay rate σ we directly observe that $\frac{b^2}{4\varepsilon^2}$ counteracts the destabilizing reaction term μ. As a result we observe that for a convection value $b \neq 0$ the equation is stable for larger values of μ, independently of the sign of b. Since the convection term has a beneficial effect on stability we expect shorter stabilizing horizons for larger $|b|$. Furthermore, we see that in contrast to σ the overshoot constant C depends on the sign of b. This asymmetry is due to the fact that the control only acts on the left boundary. Obviously, for values $b < 0$ we obtain a large overshoot constant and thus, we expect longer stabilizing horizons than for values $b \ge 0$.
In the numerical simulation we look at the reaction-convection-diffusion equation

(3.56) with varying convection value b and regularization parameter λ. The sampling time is given by $T = 0.01$ and the reaction value by $\mu = 15$. The minimal stabilizing horizon observed in the MPC closed loop is displayed in Table 3.5. Obviously, the horizon shrinks for large values of $|b|$. This corresponds to our theoretical considerations for σ. For $b \geq 0$ we observe a monotonically decay for the horizon. This behaviour is reasonable because for $b \geq 0$ the overshoot constant does not depend on b. Thus, the decay can again explained by the behaviour of σ. For small negative values of b the decay rate cannot compensate the higher value of C and we observe a larger horizon. Since the influence of C grows with an increasing regularization parameter this effect is stronger for large values of λ.

b	-5	-4	-3	-2	-1	0	1	2	3	4
$\lambda = 0.01$	2	2	5	6	6	5	4	3	3	2
$\lambda = 0.1$	2	5	11	11	9	8	6	5	3	2

Table 3.5: Minimal stabilizing horizon of the reaction-convection-diffusion equation (3.56) depending on the convection parameter b. The parameters used in the numerical MPC simulation are given by $\mu = 15$ and $T = 0.01$.

Finally, we want to remark that the findings from Section 3.3 concerning stability with the shortest possible horizon is also true for the PDE (3.56): Since we have $C \to \xi(K) > 1$ for $\lambda \to 0$ we cannot expect to observe stability for $N = 2$, even for arbitrarily small λ. This is indeed visible in the numerical simulation.

3.6 Method of Nevistic/Primbs

As we have seen in the previous sections the predicted minimal stabilizing horizons are very conservative. In this section we show the possibility to improve the results concerning suboptimality and stabilizing horizons by taking advantage of special structures of the control problem. In the following we briefly describe the method of Nevistic and Primbs presented in [75]. The method is developed for finite dimensional linear-quadratic control problems without state or control constraints and it uses Riccati difference equations (RDE). However, after an appropriate discretization this technique is also applicable for linear PDEs.

We look at the following finite dimensional linear system

$$y(n+1) = Ay(n) + Bu(n), \quad y(0) = y_0 \tag{3.62}$$

where $y(n) \in \mathbb{R}^n$ and $u(n) \in \mathbb{R}^m$. The quadratic cost functional is given by

$$V_N(y_0) = \inf_{u(\cdot)} \sum_{k=0}^{N-1} y^\top(k)Qy(k) + u^\top(k)Ru(k). \tag{3.63}$$

Note that we have a different notation to [75]. To be consistent in this thesis we changed the limit of the sum from N to N-1. The matrices $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are positive definite, A is invertible and $[A, B]$ is a stabilizable pair.
The presented method is based on the knowledge of the optimal value function. In the linear quadratic control problem without constraints this information can be obtained by the Riccati Difference Equation (RDE) which is given by

$$P_{j+1} = A^\top \left[P_j - P_jB(B^\top P_jB + R)^{-1}B^\top P_j\right] A + Q \tag{3.64}$$

with $P_0 = Q$. We define $\underline{\lambda_N}$ and $\overline{\lambda_N}$ as the smallest and the largest eigenvalue of P_N. Furthermore, let $\beta_N = \min\{\beta |\ \beta P_N \geq P_{N+1}\}$. Then the following stability theorem holds:

Theorem 3.17
Let N be such that

$$\underline{\lambda_0} - (\beta_{N-1} - 1)\overline{\lambda_N} > 0,$$

then the receding horizon policy is stabilizing and V_N is a Lyapunov function for the closed loop system with

$$V_N(y(k + 1)) \leq \gamma_N V_N(y(k))$$

where $\gamma_N = (1 - \frac{\underline{\lambda_0}-(\beta_{N-1}-1)\overline{\lambda_N}}{\overline{\lambda_N}})$. Furthermore, a bound for the infinite-horizon performance is given by

$$V_\infty^{\mu_N}(y_0) \leq \left(1 + \left(\frac{\beta_{N-1} - 1}{\beta_{N-1}}\right) \frac{\gamma_N}{1 - \gamma_N}\right) V_N(y_0) \tag{3.65}$$

The proof can be found in [75]. Note that the inverse of $\left(1 + \left(\frac{\beta_{N-1}-1}{\beta_{N-1}}\right) \frac{\gamma_N}{1-\gamma_N}\right)$ plays the role of the suboptimality degree α_N in our method.
In the next step we apply the presented method to the discretized one dimensional heat equation with distributed control

$$y_t(x, t) = y_{xx}(x, t) + \mu y(x, t) + u(x, t) \qquad \text{in} \quad (0, 1) \times (0, \infty) \tag{3.66a}$$
$$y(0, t) = y(1, t) = 0 \qquad \text{in} \quad (0, \infty) \tag{3.66b}$$
$$y(x, 0) = y_0(x) \qquad \text{in} \quad (0, 1). \tag{3.66c}$$

and quadratic stage cost

$$l(y(n), u(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot, nT)\|^2_{L^2(\Omega)}. \tag{3.67}$$

Since the previous results are formulated for finite dimensional discrete time systems we have to rewrite (3.66) in an appropriate way. In the first step we discretized the spatial variable by central finite differences with M inner grid points and obtain

$$y_h'(t) = \tilde{A}y_h(t) + \tilde{B}u_h(t) \tag{3.68}$$

with

$$\tilde{A} := \frac{1}{h_x^2} \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix} + \mu I_M \quad \text{and} \quad \tilde{B} := I_M$$

where I_M denotes the unit matrix and $h_x = 1/(M+1)$ the spatial discretization. Equation (3.68) is a linear system of ODEs with constant coefficients and the solution exists and is unique. The solution can be formally written as

$$y_h(t) = e^{t\tilde{A}} y_{h0} + \int_0^t e^{(t-\tau)\tilde{A}} \tilde{B} u_h(\tau)\, d\tau \tag{3.69}$$

where $e^{\cdot}$ denotes the matrix exponential. It is remarkable that the solution formula also holds for infinite dimensional systems if $\tilde{A}$ is interpreted as the infinitesimal generator of $e^{t\tilde{A}}$ in the sense of semigroup theory, cf. [78].
By introducing the sampling time T and taking into account that the control is constant in each sampling interval we obtain

$$y(n+1) = e^{T\tilde{A}} y(n) + \tilde{A}^{-1} \left(e^{T\tilde{A}} - I_N \right) \tilde{B} u(n)$$

with $y(n) := y_h(nT)$. By defining $A := e^{T\tilde{A}}$ and $B := \tilde{A}^{-1} \left(e^{T\tilde{A}} - I_N \right) \tilde{B}$ we get exactly the required form (3.62).
Furthermore we have to approximate the L^2-norm in the stage cost (3.67). By using the trapezoidal rule we obtain the following matrices for Q and R

$$Q := \frac{h_x}{2} \begin{pmatrix} \frac{1}{2} & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \frac{1}{2} \end{pmatrix} \quad \text{and} \quad R := \lambda Q.$$

In order to compare the results of [75] with our method we implemented the procedure in Matlab. Since the main focus of this section is not on an efficient code but only on the results we implement the matrices straightforward by using the matrix exponential *expm*. The required values in (3.65) are the maximal eigenvalue $\overline{\lambda_N}$ of P_N and the value β_N that can be computed as the maximum eigenvalue of $P_{N+1}P_N^{-1}$, cf. [75]. In order to ensure that the Riccati matrices P_N are correctly computed we also determine the optimal control sequence by

$$u_N(k) = -\left(B^\top P_{N-(k+1)} B + R \right)^{-1} B^\top P_{N-(k+1)} A y(k)$$

and compare the results with those one obtained by the optimal control algorithms presented in Section 4.1.

To get the values for the method presented in Section 3.1 we proceed as already described: Insert the K depending values $\sigma(K)$ and $C(K)$ from Theorem 3.4 into the α_N formula (1.35) and determine that value of K which produces the largest α_N.
We consider the linear heat equation (3.66) with parameter $T = 0.01$, $\lambda = 0.01$, $M = 100$ and $\mu = 12$.

N	α_N N/P	α_N
7	-0.0291	-0.1483
8	0.0935	-0.1398
9	0.2080	-0.1256
10	0.3140	-0.1061
11	0.4108	-0.0816
12	0.4979	-0.052
13	0.5751	-0.019
14	0.6426	0.017
15	0.7010	0.057

Table 3.6: Suboptimality degree α_N determined by the method of Nevistic/Primbs [75] and by the method presented in Section 3.1 in dependence of the horizon N

In Table 3.6 the α_N values for both methods are displayed. Obviously, the values derived from the method presented in Section 3.1 are smaller than those we determined by the procedure of Nevistic/Primbs. As a result, the α value becomes positive for a smaller horizon N and, thus, we can guarantee stability for a smaller horizon. In the example the predicted minimal stabilizing horizon for the first method is given by $N = 8$ and, thus, much more closer to the true horizon $N = 4$ than the predicted horizon $N = 14$ from the second method. This outcome is also observable for all considered parameters.
This result is not surprising because the method of this section uses the special structure of the linear quadratic problem to obtain information about the optimal value function. The drawback is that this approach will not work for different control structures or if control or state constraints are incorporated. The power of the technique presented in Section 3.1 is that it is also applicable for nonlinear and infinite dimensional systems. Furthermore, no knowledge of the optimal control is required. However, the price of this generality is given by the conservatism of the results.

Remark 3.18
We want to mention that the results concerning the minimal stabilizing horizon can even be improved if we take the exact optimal solution into account. For the one dimensional linear heat equation with distributed control and quadratic cost functional

the following upper bound can be proven

$$\bar{N} = \left\lceil \frac{\lambda}{T}(\mu - \pi^2) \right\rceil + 1. \tag{3.70}$$

This estimate on the minimal stabilizing horizon yields significantly better results than both other methods.

3.7 Linear Wave Equation

In this section we show that the presented method is not only applicable for parabolic PDEs but also for hyperbolic equations. Using the example of the linear, boundary controlled wave equation we demonstrate the applicability of the road map and explain the difference to the parabolic heat equation. The finite propagation speed of hyperbolic systems plays an important role.

3.7.1 Exponential Constants for the Wave Equation

We look at the one dimensional linear wave equation

$$y_{tt}(x,t) - c^2 y_{xx}(x,t) = 0 \qquad \text{in } (0,L) \times (0,\infty) \tag{3.71a}$$

$$y(0,t) = 0 \qquad \text{in } (0,\infty) \tag{3.71b}$$

$$y_x(L,t) = v(t) \qquad \text{in } (0,\infty) \tag{3.71c}$$

with Neumann boundary control on the right side of the domain $\Omega = (0,L)$. On the left side we impose a homogeneous Dirichlet condition. The propagation speed is given by c. In contrast to the heat equation the wave equation is of second order in time and, thus, we need an initial condition for the state $y(x,0) = y_0(x) \in H^1(\Omega)$ as well as for $y_t(x,0) = y_1(x) \in L^2(\Omega)$. The PDE (3.71) is known to be finite time controllable for $T \geq \frac{2L}{c}$, cf. [47].
The equilibrium is given by $(y, y_t) = (0,0)$. Again, we define the discrete state as $y(n) = y(x, nT)$ and the control as $u(n)(\cdot) = v(\cdot)|_{[nT,(n+1)T)}$. The choice of an appropriate cost functional is quite important. First, we note that the functional

$$l(y(n), u(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\int_{nT}^{(n+1)T} v(t)^2 \, dt.$$

from the parabolic case is not applicable since it does not identify the equilibrium. A suitable choice with regard to the equilibrium is given by

$$l(y(n), u(n)) = \frac{1}{2}\|y(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{1}{2}\|y_t(\cdot, nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\int_{nT}^{(n+1)T} v(t)^2 \, dt. \tag{3.72}$$

Since the energy of the system plays an important role for the wave equation (for the uncontrolled equation the energy is conserved) it seems to be an appropriate task to minimize the energy of the system. The total energy consists of the potential energy y_x and the kinetic energy y_t

$$E(t) := \frac{1}{2}\int_0^L y_x(x,t)^2 + \frac{1}{c^2} y_t(x,t)^2 \, dx \tag{3.73}$$

Therefore, the corresponding stage costs are given by

$$l(y(n),u(n)) = \frac{1}{2}\|y_x(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{1}{2}\|y_t(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}\int_{nT}^{(n+1)T} v(t)^2 \, dt. \tag{3.74}$$

Note that because of the Poincaré inequality the stage costs (3.72) and (3.74) are equivalent. We will see that we have to alter the energy (3.74) to a *weighted energy* to get suitable estimates. First, we give the motivation for this concept, see also [8] and [6]. A well known feedback to stabilize the wave equation (3.71) is given by

$$y_x(1,t) = -\frac{K}{c} y_t(1,t) \tag{3.75}$$

with $K \in (0,1]$. In order to explain the behaviour of this feedback we consider the

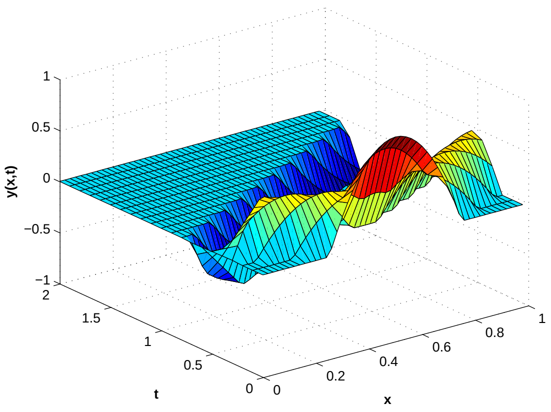


Figure 3.2: Solution trajectory of the wave equation (3.71) with control (3.75)

following example:

Example 3.19

We consider the wave equation (3.71) *with control* (3.75) *and initial conditions*

$$y_0(x) = \begin{cases} -16x^2 + 16x - 3 & : \quad 0.25 \le x \le 0.75 \\ 0 & : \quad \textit{else} \end{cases} \tag{3.76}$$

and $y_1(x) = 0$. *The parameters are given by* $K = L = c = 1$. *In Figure 3.2 the solution trajectory of* (3.71) *with control* (3.75) *is displayed. It is obvious that for*

$T \geq 2$ the origin is obtained. Figure 3.3 (left) displays the energy of this system. It can be seen that the energy is monotonically, but not strictly monotonically decreasing. This is due to the fact that the propagation speed of the wave is finite and the control acts only on the right boundary. Therefore, the control can only reduce the energy of the system if the wave reaches the right side. In view of the constants for the exponential controllability this leads to a large overshoot constant C. This means the MPC algorithm requires a long prediction horizon to observe a decay in the stage cost. Actually we can only prove stability for a prediction horizon of at least $T \geq 2$.
This problem leads to the introduction of a so called weighted energy. In order to motivate the choice of the weighted energy we analyse the exact solution of the equation. Note, that we do not use this representation in the proof of the exponential controllability, in contrast to [8]. The solution of (3.71) *with control* (3.75)*, which can be found by using the method of reflection, cf. [88], is given by*

$$\begin{aligned} y(x,t) &= \frac{1}{2}(y_0(x+ct) + y_0(x-ct)) + \frac{1}{2c}\int_{x-ct}^{x+ct} y_1(s)\,ds \qquad \textit{for} \quad x > ct \\ y(x,t) &= \frac{1}{2}(y_0(ct+x) - y_0(ct-x)) + \frac{1}{2c}\int_{ct-x}^{ct+x} y_1(s)\,ds \qquad \textit{for} \quad x < ct \end{aligned} \tag{3.77}$$

with the partial derivatives

$$\begin{aligned} y_t(x,t) &= \frac{c}{2}(y_0'(x+ct) - y_0'(x-ct)) + \frac{1}{2}(y_1(x+ct) + y_1(x-ct)) \qquad \textit{for} \quad x > ct \\ y_t(x,t) &= \frac{c}{2}(y_0'(ct+x) - y_0'(ct-x)) + \frac{1}{2}(y_1(ct+x) - y_1(ct-x)) \qquad \textit{for} \quad x < ct \end{aligned}$$

and

$$\begin{aligned} y_x(x,t) &= \frac{1}{2}(y_0'(x+ct) + y_0'(x-ct)) + \frac{1}{2c}(y_1(x+ct) - y_1(x-ct)) \qquad \textit{for} \quad x > ct \\ y_x(x,t) &= \frac{1}{2}(y_0'(ct+x) + y_0'(ct-x)) + \frac{1}{2c}(y_1(ct+x) + y_1(ct-x)) \qquad \textit{for} \quad x < ct. \end{aligned}$$

The simplicity of the solution is due to the fact that we only take into account $K = 1$ and, thus, no reflections occur on the right side.
The energy of the system is given by

$$\begin{aligned} E(t) :=& \frac{1}{2}\int_0^1 y_x(x,t)^2 + \frac{1}{c^2}y_t(x,t)^2\,dx \\ =& \frac{1}{4}\int_0^1 (y_x(x,t) + \frac{1}{c}y_t(x,t))^2 + (y_x(x,t) - \frac{1}{c}y_t(x,t))^2\,dx \\ =& \frac{1}{4}\int_0^{ct} (y_0'(ct+x) + \frac{1}{c}y_1'(ct+x))^2 + (y_0'(ct-x) + \frac{1}{c}y_1'(ct-x))^2\,dx \\ &+ \frac{1}{4}\int_{ct}^1 (y_0'(x+ct) + \frac{1}{c}y_1'(x+ct))^2 + (y_0'(x-ct) - \frac{1}{c}y_1'(x-ct))^2\,dx. \end{aligned}$$

Obviously, the energy can be split into two parts: $(y_x(x,t) + \frac{1}{c}y_t(x,t))^2$ *represents the movement of the energy to the left and* $(y_x(x,t) - \frac{1}{c}y_t(x,t))^2$ *to the right side. Furthermore, it can be seen that the decrease of the energy is given by*

$$\int_{1-ct}^{1} (y_0'(x) - \frac{1}{c}y_1(x))^2 \, dx.$$

This explains, why we do not observe a strict decay for the energy: If $y_0'(x)$ *and*

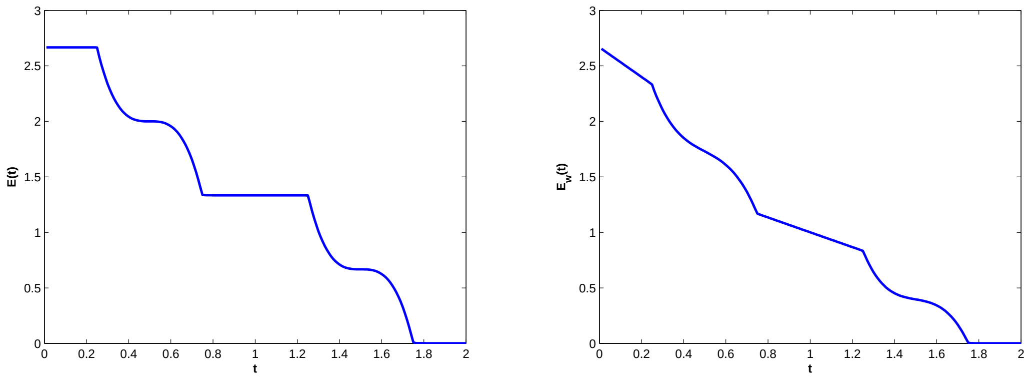


Figure 3.3: Classical energy $E(t)$ (left) and weighted energy $E_w(t)$ (right) for the Example 3.19

$y_1(x)$ *vanish for* $x \in [1-ct, 1]$ *or the terms cancel out each other, the energy stays constant. In order to overcome this fact we introduce a so called weighted energy which takes the movement of the wave into account*

$$E_w(t) := \frac{1}{4}\int_0^1 \omega_1(x)\left(y_x(x,t) + \frac{1}{c}y_t(x,t)\right)^2 + \omega_2(x)\left(y_x(x,t) - \frac{1}{c}y_t(x,t)\right)^2 dx \tag{3.78}$$

where $\omega_i : [0,1] \to \mathbb{R}, i = 1,2,$ *are continuous functions. A natural choice for the weight functions is given by* $\omega_1 = 2 + x$ *and* $\omega_2 = 2 - x$*, see Figure 3.4. The arrows indicate the direction of the propagation, i.e. the energy far away from the actuator is stronger penalized than the energy close to the right side. In Figure 3.3 (right) it can be seen that the weighted energy* (3.78) *with control* (3.75) *actually is a strictly monotonically decreasing function.*

After this motivation we rewrite the weighted energy (3.78) in an appropriate way and define the functional

$$V(t) := \frac{1}{2}\int_0^L y_x(x,t)^2 + \frac{1}{c^2}y_t(x,t)^2 + \frac{\delta}{c^2}xy_x(x,t)y_t(x,t)\,dx \tag{3.79}$$

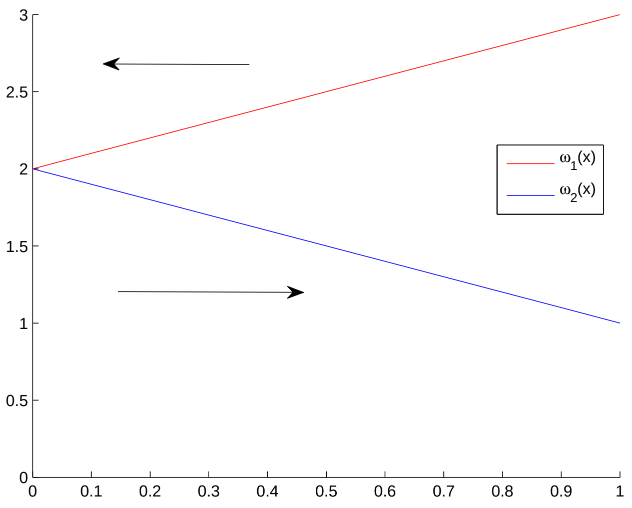


Figure 3.4: Weightfunctions ω_1, ω_2 in the weighted energy (3.78)

with $\delta \in (0,2)$ (for $\delta = 1$ we obtain (3.78)). This leads us to the following stage cost

$$l(y(n), u(n)) := V(nT) + \frac{\lambda}{2} \int_0^T u(n)(\cdot)^2 \, dt. \tag{3.80}$$

In [8] we derived the exponential constants C and σ for the case $K = \delta = 1$. Since for these parameters the solution can be easily found (see Example 3.19), the proof was based on the exact representation. In the following theorem we generalize the results from [8] by allowing $\delta \in (0,2)$ and $K \in (0,1]$. Thus, we allow reflections on the right boundary (this happens for $K \in (0,1)$). In this case it is much more complicated to give a representation of the solution. Therefore, the following proof is based on Lyapunov methods presented in [62] and avoids the knowledge of the exact solution.

Theorem 3.20
The boundary controlled wave equation (3.71) *with control* (3.75)*,* $K \in (0,1]$*, and stage cost* (3.80)*,* $0 < \delta \leq \frac{4Kc}{L(K^2+1)}$*, satisfies the exponential controllability condition (1.34). The corresponding constants are given by* $\sigma = e^{-\frac{\delta}{\delta \frac{L}{c}+2}T}$ *and* $C = \left(1 + \frac{\lambda K}{(2-\delta\frac{L}{c})c}\right)$.

Proof. In order to estimate σ we differentiate (3.79) with respect to time and get

$$\dot{V}(t) = \int_0^L y_x(x,t) y_{xt}(x,t) + \frac{1}{c^2} y_t(x,t) y_{tt}(x,t)$$
$$+ \frac{\delta}{2c^2} x \left(y_{xt}(x,t) y_t(x,t) + y_x(x,t) y_{tt}(x,t) \right) \, dx$$

$$\begin{aligned}
&= y_x(x,t)y_t(x,t)\big|_0^L - \int_0^L y_{xx}(x,t)y_t(x,t)\,dx + \int_0^L y_t(x,t)y_{xx}(x,t)\,dx \\
&\quad + \frac{\delta}{4c^2}\int_0^L x\frac{d}{dx}\left(c^2 y_x(x,t)^2 + y_t(x,t)^2\right)\,dx \\
&= -\frac{K}{c}y_t(x,t)^2 + \frac{\delta}{4}\left[x\left(y_x(x,t)^2 + \frac{1}{c^2}y_t(x,t)^2\right)\right]_0^L - \frac{\delta}{4}\int_0^L y_x(x,t)^2 + \frac{1}{c^2}y_t(x,t)^2\,dx \\
&\le -\frac{K}{c}y_t(x,t)^2 + \frac{\delta L}{4}\left(\frac{K^2}{c^2}y_t(1,t)^2 + \frac{1}{c^2}y_t(1,t)^2\right) \\
&\quad - \frac{\delta}{2(\delta\frac{L}{c}+2)}\int_0^L y_x(x,t)^2 + \frac{1}{c^2}y_t(x,t)^2 + \frac{\delta}{c^2}xy_x(x,t)y_t(x,t)\,dx \\
&\le -\frac{\delta}{\delta\frac{L}{c}+2}V(t).
\end{aligned}$$

This yields

$$l^*(y(n) \le \sigma^n l^*(y(0)) \quad \text{with} \quad \sigma = e^{-\frac{\delta}{\delta\frac{L}{c}+2}T}.$$

With the same arguments (and $\delta = 0$) we obtain

$$E(t_0) = E(t_1) + \frac{K}{c}\int_{t_0}^{t_1} y_t(1,\tau)^2\,d\tau$$

with $0 \le t_0 \le t_1$. This means that the decay of the energy is exactly what we take out of the system. Furthermore, we get

$$\begin{aligned}
l(y(n),u(n)) &= V(nT) + \frac{\lambda}{2}\int_0^T u(n)(\cdot)^2\,dt = V(nT) + \frac{\lambda K^2}{2c^2}\int_{nT}^{(n+1)T} y_t(1,t)^2\,dt \\
&\le V(nT) + \frac{\lambda K}{2c}(E(nT) - E((n+1)T)) \le V(nT) + \frac{\lambda K}{2c}E(nT) \\
&\le \left(1 + \frac{\lambda K}{(2-\delta\frac{L}{c})c}\right) l^*(y(n)).
\end{aligned}$$

With $C = \left(1 + \frac{\lambda K}{(2-\delta\frac{L}{c})c}\right)$ we obtain the desired inequality. □

3.7.2 Numerical results

In this section we give some numerical results in order to illustrate the findings from Theorem 3.20. Since energy arguments play an important role in our reasoning, we choose the energy conserving Newmark scheme for the time discretization, cf. [55]. Further remarks concerning the discretization of the wave equation in the context of optimization can be found in [101] and [36]. The spatial discretization is given by $\delta x = 0.001$.
First, we show that the decay rate, derived in Theorem 3.20, holds for Example 3.19.

With the parameter $c = L = \delta = 1$ and $T = 0.01$ we get $\sigma = e^{-T/3}$. In Figure 3.5 the σ- values from the numerical simulation are displayed for the classical energy (blue 'o') and the weighted energy (red 'x'). Obviously, the values of the weighted energy are bounded from above by $\sigma = e^{-0.01/3}$ (solid black line) while the classical energy achieves values of one and is therefore useless for our method.

For the next example we look at the numerical observations from [55] and [66]. The

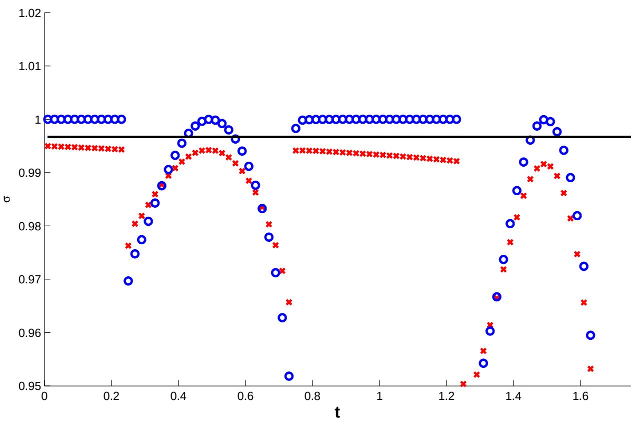


Figure 3.5: σ- values from the numerical simulation for the classical energy ('o') and the weighted energy ('x'). The solid line (black) displays the theoretical bound of σ derived in Theorem 3.20.

results indicate that the MPC algorithm performs very well, even for the shortest possible horizon $N = 2$. This special kind of MPC, where the prediction horizons do not overlap, is also called instantaneous control. Note that the use of the term instantaneous control is not unique. In [53] the authors used this notation for $N = 2$, but the corresponding optimal control problem is only approximately solved by exactly one gradient step. This technique, in turn, sometimes is also called *one step gradient method*, cf. [66].

In the following we investigate the stability behaviour of instantaneous control. From the α-formula (1.35) for $N = 2$ we obtain the sufficient stability condition $\alpha_2 = 1 - (C(1 + \sigma) - 1)^2 > 0$. For Example 3.19 and the constants $\delta = K = 1$ we get the condition $T > -3 \log\left(\frac{1-\lambda}{1+\lambda}\right)$. In the simulation we choose $\lambda = 0.001$ and $T = 0.01 > -3 \log\left(\frac{1-0.001}{1+0.001}\right) \approx 0.006$.

We compare the closed loop solution of the instantaneous control with the optimal control open loop solution on the time interval $[0, 2]$. The solution trajectory for the instantaneous control (red 'x') and the optimal control (solid blue line) at different time snapshots are depicted in Figure 3.6. Obviously, both trajectories are quite close together. This observation is pretty surprising, since the optimal control takes information of the whole interval $[0, 2]$ (in this example 200 time steps), while the instantaneous control technique only uses the information of the next time step in

each iteration. Whereas at $t = 2$ differences between the solutions are visible, at $t = 4$ the equilibrium is also for the instantaneous control almost attained (as in [55] the optimal control is continued by zero input on $(2, 4]$).
It is remarkable that the computation time for the optimal control in the whole time interval is more than 21000 times longer than for one single step of the instantaneous control. Thus, the overall time for the instantaneous control algorithm on the interval $[0; 2]$ is more than 105 times smaller than the optimal open loop control. Since the controls from both methods are quite close together and the computation of the instantaneous control is in real time possible (even for this fine spatial discretization), this technique seems to be very attractive for the wave equation.
The numerical results in [66] show that the success of the presented method is not due to the introduced weighted energy. However, only for this concept our analysis is applicable and we can prove that a stabilizing feedback is obtained. A generalization of the weighted energy to the n-dimensional wave equation seems rather complicated, because of the domain dependence of the weight functions. Only for simple domains an appropriate weight function can be found, e.g., for a two dimensional rectangular domain. In contrast to this, the instantaneous control algorithm with the classical energy can be easily generalized to the multi dimensional case. An example for a successful application of this method on a L shaped domain can be found in [55].

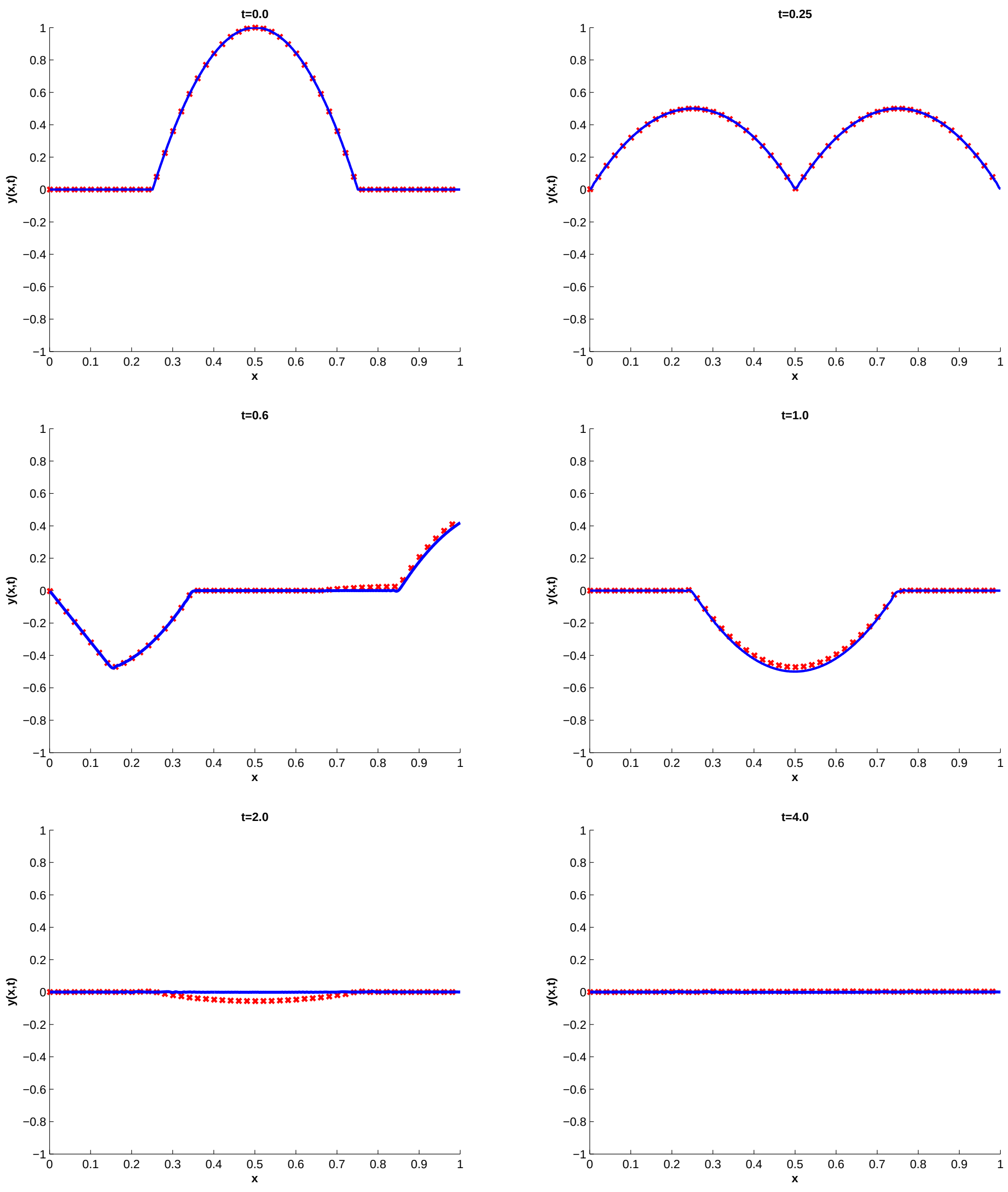


Figure 3.6: Comparison of optimal control (solid blue line) and instantaneous control (red 'x') at different time snapshots.

4 Algorithms

In this chapter we present algorithms for solving MPC problems with PDEs. Furthermore, we investigate methods for solving the arising subproblems. The basic idea of MPC was presented in Section 1.2 and the most simple MPC Algorithm 1.1 forms the basis for the following considerations.
The essential effort of MPC is to solve an optimal control problem in each time step. Since this is, especially in the case of PDEs, a time consuming task, it is quite important to find efficient algorithms to solve such problems. Recently a lot of literature concerning this topic has been published. A detailed introduction to the field of optimization algorithms for infinite dimensional systems can be found in [52] and [18]. In Section 4.1 we summarize well known algorithms in PDE constrained optimization.
One possibility to significantly reduce the computing time for the optimal control problem is the use of model reduction techniques. Here, we focus on the reduction method Proper Orthogonal Decomposition (POD) presented in Section 2.3.1. In Section 4.2 we present known algorithms, which combine model predictive control with model order reduction and compare those with our new approach.
In the ensuing Section 4.3 we introduce the idea of *adaptive horizon MPC* from [38] and [77] and show how to implement these algorithms in the context of infinite dimensional control systems. Moreover, we investigate the possibility to combine adaptive horizon MPC with multigrid methods.

4.1 Algorithms in PDE Optimization

In this section we present algorithmic approaches for solving PDE constrained optimization problems. The presentation is based on [52], [49] and [18]. First, we recapitulate the abstract optimization problem (2.1)

$$\min_{(y,u)\in Y\times U} J(y,u) \quad \text{subject to} \quad e(y,u)=0, \quad u\in\mathbb{U} \tag{4.1}$$

with $J : Y\times U \to \mathbb{R}, e : Y\times U \to Z$. In our context $J(y,u)$ denotes the cost functional, $e(y,u)=0$ represents the PDE and the admissible set of control values $\mathbb{U}$ is given by pointwise box constraints.
The algorithms presented in this section are based on the so called *Black-Box* approach. This means we distinguish between the independent control variable u and the dependent state variable $y(u)$. We already investigated this concept in Section 2.1 where we introduced the unique solution operator $y=S(u)$ to eliminate the

constraint $e(y,u)=0$. The *reduced cost functional* is given by $\hat{J}(u) := J(S(u),u)$. Thus, the state y is eliminated from the optimization problem. However, the control constraints are still present.
In contrast to this, the *All-at-Once* technique considers the control and state variables as independent optimization variables, which are coupled through the PDE $e(y,u)=0$. For details we refer to [40].
Furthermore, we distinguish between first order gradient-type methods and higher order (Quasi-) Newton methods. The following algorithms are formulated in a function space setting. The presentation is based on [52] and [18]. A detailed overview of algorithms in finite dimensional optimization can be found in [76] and [35, 34].

4.1.1 First Order Methods

The methods under observation in this thesis are so called *descent algorithms*. The idea is to find at the current iterate $u_k \in U$ a descent direction $d_k \in U$ satisfying

$$\langle \nabla \hat{J}(u_k), d_k \rangle_U < 0 \tag{4.2}$$

and a step length σ_k satisfying

$$\hat{J}(u_k + \sigma_k d_k) < \hat{J}(u_k). \tag{4.3}$$

Therefore, the general descent method is given by

```
   Input: initial point $u_0 \in U$
1  for $n = 0, 1, \dots$ do
2      if $\nabla \hat{J}(u_n) = 0$ then
3          Stop
4      end
5      Choose a descent direction $d_n \in U$: $\langle \nabla \hat{J}(u_n), d_n \rangle_U < 0$;
6      Choose a stepsize $\sigma_n > 0$ such that $\hat{J}(u_n + \sigma_n d_n) < \hat{J}(u_n)$;
7      Set $u_{n+1} := u_n + \sigma_n d_n$;
8      Set $n = n + 1$;
9  end
```

Algorithm 4.1: General descent method

Since we have no further conditions on the stepsize, for this general algorithm it is not possible to prove convergence, cf. [52]. (It may occur that the stepsize decreases too fast). Moreover, for a practical implementation the stopping criterion (2) should be replaced by an appropriate one, e.g. $\|\nabla \hat{J}(u_k)\|_U < \varepsilon$ with prescribed $\varepsilon > 0$. The choice of the steplength σ_k plays an important role in the algorithm. A widely used criterion for an appropriate step size is given by the *Armijo Rule*

$$\hat{J}(u_k + \sigma_k d_k) \leq \hat{J}(u_k) + \delta \sigma_k \langle \nabla \hat{J}(u_k), d_k \rangle_U \tag{4.4}$$

with a constant $\delta \in (0,1)$. This condition guarantees a sufficient large decrease and can be generalized to the problem with box constraints, cf. [52]. In order to ensure that the cost functional does not increase too fast beyond the minimum in addition the following inequality

$$\langle \nabla \hat{J}(u_k + \sigma_k d_k), d_k \rangle_U > \gamma \langle \nabla \hat{J}(u_k), d_k \rangle_U \text{ with } \gamma \in (0,1) \tag{4.5}$$

should be satisfied. Both inequalities together are known as *Wolfe-Powell condition.* A frequently used method to find an appropriate step length is the bisection technique, i.e., we choose the maximum $\sigma_k \in \{1, 1/2, 1/4, \dots\}$ that satisfies condition (4.4) and (4.5). A detailed overview of line search algorithms and their geometrical meaning can be found in [76].
An alternative method to obtain a suitable step size is the *inexact line search*, cf. [49] in the context of PDE constrained optimization. The idea is to minimize $\varphi(t) := \hat{J}(u_k + td_k)$ for fixed d_k. Solving this problem by Newton's method is not useful because the Hessian is not computed in first order methods. The Taylor expansion combined with the secant method yields

$$\varphi(t) \approx \varphi(0) + \varphi'(0)t + \frac{\varphi'(\sigma) - \varphi'(0)}{2\sigma} t^2 \tag{4.6}$$

and minimization of the right hand side leads to

$$t = -\frac{\sigma \varphi'(0)}{\varphi'(\sigma) - \varphi'(0)} = -\frac{\sigma \langle \nabla \hat{J}(u_k), d_k \rangle_U}{\langle \nabla \hat{J}(u_k + \sigma d_k), d_k \rangle_U - \langle \nabla \hat{J}(u_k), d_k \rangle_U}. \tag{4.7}$$

Starting with an initial value $\sigma_0 > 0$ and setting $\sigma_{i+1} = t_i$ yields an iterative procedure. Although this inexact line search may lead to no descent, it yields quite good results in the practical implementation. The results in Chapter 6 indicate that only very few iterations of this line search are necessary to obtain a significant reduction in the number of steps in the descent method, compared to the bisection method.

Next, we focus on the determination of an appropriate descent direction. The *gradient methods* belong to the most simple optimization algorithms. They are in general of first order and comparatively easy to implement. However, for complicated problems, where no second order information is available, they are still of interest.
The crucial step is the efficient computation of the gradient. There are essentially three different methods in order to obtain the first order information. One possibility is given by numerical differentiation. In this case the gradient is built up by finite difference methods. This approach is used e.g. in the MPC software package *yane*. For details concerning numerical differentiation we refer to [76] and in the context of PDE constrained optimization to [41]. Moreover, it is possible to use *Automatic Differentiation.* This technique decomposes the code for the evaluation of the cost functional in elementary arithmetic operations, where exact differentiation rules can be applied iteratively. In this thesis, we focus on a third approach, the *adjoint based* approach to determine the gradient. An overview of these methods

and a comparative study in the context of PDE constrained optimization can be found in [41].
The adjoint method is based on the analysis in Chapter 2 and uses the representation of the gradient in formula (2.7). Thus, the algorithm to compute the gradient at some point $u \in U$ of the reduced cost functional is given by:

```
  Input: $u \in U$
  Output: $\nabla \hat{J}(u)$
1 Solve the state equation $e(y(u), u) = 0$;
2 Solve the adjoint equation $e_y(y(u), u)^* p = -J_y(y(u), u)$;
3 Compute the reduced gradient $\nabla \hat{J}(u) = e_u(y(u), u)^* p + J_u(y(u), u)$
```

Algorithm 4.2: Evaluation of the reduced gradient $\nabla \hat{J}(u)$

Obviously, the computation of the gradient needs the solution of the (potentially nonlinear) state equation and the linear adjoint equation. This algorithm enables us to compute first order optimization algorithms.

Remark 4.1
*Note, that Algorithm 4.2 is only exact in the function space setting. However, in the real implementation the state and adjoint equation cannot be computed exactly and have to be discretized. The way presented in this section is in the literature known as **optimize then discretize (OD)** approach, i.e., we build up the optimization algorithms and the gradient in the function space and afterwards we discretize the arising subproblems for the state equation $e(y, u)$, the adjoint equation $e_y(y, u)^*$ and the cost functional $J(y, u)$ e.g. by the finite element method presented in Section 2.3.1. The opposite approach is given by **discretize then optimize (DO)**. In this case the cost functional and the state equation are discretized and a finite dimensional nonlinear optimization problem is obtained. With the classical finite dimensional optimality system the adjoint of the discrete system and the discrete gradient can be determined. For general discretizations of the adjoint equation in the (OD) case we cannot expect that the discrete gradient coincides with that one from the (DO) case. However, an inconsistent gradient could lead to problems in the optimization algorithm. Thus, it is reasonable to use a discretization for the adjoint equation that matches the discretization of the state equation and the cost functional. A detailed overview of the construction of appropriate discretizations of the adjoint equation can be found in [71]. The important case, in which we discretize the time variable by a semi implicit Euler scheme (see Section 2.3.1), can be found e.g. in [39].*

Projected Gradient Methods

The first algorithm we present is the *gradient method*, also known as steepest descent method. The idea is to go in each step of the descent Algorithm 4.1 towards the negative gradient, i.e., we choose $d_n := -\nabla \hat{J}(u_n)$. The convergence proof of the gradient method with a step size satisfying the Armijo condition can be found in

[76] for finite dimensional problems and in [52] for the Banach space setting.

```
  Input: initial guess $u_0 \in U$, max steps $n_{\max}$, tolerance $tol$
1 Set $n := 0$;
2 Evaluate $d_0 := -\nabla \hat{J}(u_0)$;
3 while ($n < n_{max}$ && $\|d_n\|_U > tol$) do
4     Evaluate $d_n := -\nabla \hat{J}(u_n)$ with Algorithm 4.2;
5     Calculate a step length $\sigma_n$, satisfying Armijo's rule (4.4);
6     Set $u_{n+1} := u_n + \sigma_n d_n$;
7     Set $n = n + 1$
8 end
```

Algorithm 4.3: Gradient algorithm

The gradient method is known to quickly reduce the objective function in the first steps, but then it slows down. Since the initial guess for the optimal control problem in the MPC algorithm is in general quite good (except for the first step), this algorithm seems to be inappropriate for MPC. As we will see in Chapter 6 this is often actually true. However, for some problems the gradient method is a serious alternative.

One advantage of this method is that it can easily be generalized to the case with box control constraints. In this case we use the projection $P_{[u_a,u_b]}(v) := \max\{u_a, \min\{u_b, v\}\}$ to obtain admissible controls, where u_a and u_b denote the lower and upper bound. Furthermore, the Armijo rule and the Wolfe-Powell condition have to be adapted to the constrained case

$$\hat{J}(P_{[u_a,u_b]}(u^k + \sigma_k d^k)) \leq \hat{J}(u^k) + \delta\sigma_k \langle \nabla \hat{J}(u^k), \tilde{d}^k \rangle_U \tag{4.8a}$$

$$\langle \nabla \hat{J}(P_{[u_a,u_b]}(u^k + \sigma_k d^k)), d^k \rangle_U > \gamma \langle \nabla \hat{J}(u^k), \tilde{d}^k \rangle_U \tag{4.8b}$$

with $0 < \delta < \gamma < \frac{1}{2}$ and $\tilde{d}_k$ is as follows

$$\tilde{d}_k = \begin{cases} d_k & : \quad u_k \in int(U_{ad}) \\ 0 & : \quad u_k \in \partial U_{ad} \text{ and } d_k \text{ points outwards } \mathbb{U} \end{cases}$$

The resulting method is known as *projected gradient algorithm* and is given by

```
   Input: initial guess $u_0 \in \mathbb{U}$, max steps $n_{\max}$, tolerance $tol$
1  Set $n := 0$;
2  Evaluate $d_0 := -\nabla \hat{J}(u_0)$;
3  while ($n < n_{max}$ && $\|d_n\|_U > tol$) do
4      Evaluate $d_n := -\nabla \hat{J}(u_n)$ with Algorithm 4.2;
5      Calculate a step length $\sigma_n$, satisfying projected Armijo rule (4.8a);
6      Set $u_{n+1} := P_{[u_a,u_b]}(u_n + \sigma_n d_n)$;
7      Set $n = n + 1$;
8  end
```

Algorithm 4.4: Projected gradient algorithm

Nonlinear Conjugate Gradient Methods

The next algorithm under consideration is the *Nonlinear Conjugate Gradient (NCG)* algorithm. The conjugate gradient (CG) algorithm is a well known iterative method for solving systems of linear equations with a symmetric and positive definite coefficient matrix. Especially for large scale systems CG has attractive convergence properties (see also the application to Newton's method in 4.1.2). The idea of the CG method originates from the minimization of a convex, quadratic cost function. The NCG algorithm generalizes the CG approach to general nonlinear functions. A detailed overview of this method for finite dimensional optimal control problems can be found in [76]. A presentation of the NCG algorithm in the case of PDE constrained optimization can be found in [49], where it was used for the optimal control of the Navier-Stokes equation. The NCG algorithm in the Banach space setting, cf. [18], is given by

```
   Input: initial guess $u_0 \in U$, max steps $n_{\max}$, tolerance $tol$
1  Set $n := 0$;
2  Evaluate $d_0 := r_0 := -\nabla \hat{J}(u_0)$;
3  while ($n < n_{max}$ && $\|d_n\| > tol$) do
4      Calculate a step length $\sigma_n$ satisfying Armijo's rule (4.4);
5      Set $u_{n+1} := u_n + \sigma_n d_n$ ; $r_{n+1} := -\nabla \hat{J}(u_{n+1})$;
6      Determine step length $\beta_{n+1}$ ;
7      Set $d_{n+1} := r_{n+1} + \beta_{n+1} d_n$ and $n := n + 1$;
8      if $(r_n, d_n) \leq 0$ then
9          Set $d_n := r_n$;
10     end
11 end
```

Algorithm 4.5: Nonlinear Conjugate Gradient algorithm

There are different options for the choice of β in step (6). Quite popular formulas (especially for the finite dimensional optimization) are given by the Fletcher-Reeves and the Polak-Ribiere methods

$$\beta_{k+1}^{FR} := \frac{\langle r_{k+1}, r_{k+1}\rangle_U}{\langle r_k, r_k\rangle_U}, \quad \beta_{k+1}^{PR} := \frac{\langle r_{k+1}, r_{k+1} - r_k\rangle_U}{\langle r_k, r_k\rangle_U}, \quad \beta_{k+1}^{PR+} := \max\{\beta_{k+1}^{PR}, 0\}$$

More elaborate formulations, cf. [18], are given by Dai-Yuan

$$\beta_{k+1}^{DY} = \frac{\langle r_{k+1}, r_{k+1}\rangle_U}{\langle d_k, r_{k+1} - r_k\rangle_U}$$

and by Hager-Zhang

$$\beta_{k+1}^{HZ} = \frac{\langle \sigma_k, r_{k+1}\rangle_U}{\langle d_k, y_k\rangle_U} \quad \text{with} \quad \sigma_k = y_k - \frac{\langle y_k, y_k\rangle_U}{\langle y_k, d_k\rangle_U}$$

and $y_k = r_{k+1} - r_k$. The results in [97] indicate that β_{k+1}^{DY} and β_{k+1}^{HZ} are favourable in the context of PDE constrained optimization. It is important to note that the β formulas are based on the control space inner product. In the implementation this has to be approximated in an appropriate way. Using the finite dimensional Euclidean inner product would lead to scaling problems and, thus, to bad convergence properties. Even in the infinite dimensional setting there is a significant difference in the convergence speed between choosing the L^2 or H^1 inner product, cf. [97].

It is well known, cf. [76], that the NCG method generally yields better results than the steepest descent method. The nonlinear CG method was successfully used in applications like the optimal control of quantum systems [97] and the control of laser processes [95]. The numerical results in Chapter 6 indicate that the NCG method also yields good results for the MPC algorithm.

4.1.2 Higher Order Methods

In this section we present higher order optimization algorithms. We start with the optimization problem, where no additional control constraints are present. Later on we will see how this problem can be handled. The most popular higher order algorithm for finite and infinite dimensional optimization problems is given by Newton's method. The basic algorithm reads

```
  **Input**: initial guess $u_0 \in \mathcal{N}(u^*)$, max steps $n_{\max}$, tolerance $tol$
1 **while** $(n < n_{max} \quad \&\& \quad \|\nabla \hat{J}(u_n)\| > tol)$ **do**
2     Solve $\nabla^2 \hat{J}(u_n)\delta u_n = -\nabla \hat{J}(u_n)$ ;
3     Update $u_{n+1} = u_n + \delta u_n$ ;
4     Set $n = n + 1$ ;
5 **end**
```

Algorithm 4.6: Newton algorithm

The condition $u_0 \in \mathcal{N}(u^*)$ means that the initial guess has to be close to the optimal value u^*, because the algorithm is only *locally* convergent. Algorithm 4.6 requires second derivatives of the reduced cost functional and the solution of a linear operator equation in each optimization step. After discretization of the control space the latter results in an often high dimensional linear system of equations. Especially in the case of distributed control, where the number of control variables is rather high, this problem has to be solved with iterative methods. A further problem is that building up the Hessian $\nabla^2 \hat{J}(u)$ in an explicit way is a time consuming task. Moreover, since the dimension of the Hessian is the square of the number of control variables, this can also lead to storage problems. In order to demonstrate this we consider the example of a two dimensional distributed control problem. Let the spatial discretization in each dimension be given by $h_x = \frac{1}{100}$ and an optimization horizon of $N = 20$ is chosen. If we store the variables in the C++ format *double*, we will obtain a memory requirement for the Hessian of $(100^2 \cdot 20)^2 \cdot 8$ byte ≈ 298 Gbyte. This is in general not practicable in view of the internal memory. For a finer discretization, a higher dimensional problem or a longer horizon it is in general infeasible to build up the Hessian explicitly. To overcome the described problems in the following section we consider the application of *Krylov type* methods to the Newton algorithm.

Newton-CG Method

Next, we explain a possibility to make Algorithm 4.6 applicable for PDE constrained optimization. The idea is to solve the linear system of equations (2) by an iterative Krylov method like CG or MINRES. The main advantage is that these algorithms do not require the explicitly formed Hessian, but only the evaluation of matrix-vector products $\nabla^2 \hat{J}(u)\delta u$. The evaluation of the reduced Hessian times a vector can be done by the methods presented in Section 2.2. The following Algorithm 4.7 is based on the representation of the Hessian (2.10).

```
  Input: $u \in U$, $s \in U$
  Output: $\nabla^2 \hat{J}(u)s$
1 Solve the state equation $e(y(u), u) = 0$ ;
2 Solve the adjoint equation $e_y(y(u), u)^* p = -J_y(y(u), u)$ ;
3 Solve the linearized state equation $e_y(y(u), u)\delta_s y = -e_u(y(u), u)s$ ;
4 Compute $h_1 = L_{yy}(y(u), u, p(u))\delta_s y + L_{yu}(y(u), u, p(u))s$;
5 Compute $h_2 = L_{uy}(y(u), u, p(u))\delta_s y + L_{uu}(y(u), u, p(u))s$;
6 Compute the adjoint equation $e_y(y(u), u)^* h_3 = -e_u(y(u), u)^* h_1$ ;
7 Compute $\nabla^2 \hat{J}(u)s = h_2 + h_3$
```

Algorithm 4.7: Evaluation of the reduced Hessian times a vector $\nabla^2 \hat{J}(u)s$

Note that the steps (1) and (2) are typically already done for the evaluation of the gradient. Therefore, the additional effort essentially derives from the solution of a linearized state equation (3) and a linear adjoint equation (6). The Lagrange function is again given by $L(y, u, p) = J(y, u) + \langle p, e(y, u)\rangle_{Z^*,Z}$. All cost functionals considered in this thesis have the structure $J(y, u) = J_1(y) + J_2(u)$, i.e., the control and the state variable are decomposed. In this case we obtain $L_{yu} = L_{uy} = 0$. Since J_1 and J_2 are usually quadratic functionals and $e(y, u)$ represents a semilinear PDE, where the partial derivative of the *Nemitzki operator* can be easily computed analytically, the evaluation of L_{yy} and L_{uu} is not a big problem.

Example 4.2
In this example we show how to obtain the partial derivatives of the Lagrangian in the case of an abstract semilinear PDE and a quadratic cost functional. We consider $e(y, u) = Ay + f(y) + Bu = 0$ *and* $J(y, u) = \frac{1}{2}\|Qy - y_d\|_Y^2 + \frac{\lambda}{2}\|u\|_U^2$ *where* A, B, Q *are appropriate operators. With the Lagrangian* $L(y, u, p) = J(y, u) + \langle p, e(y, u)\rangle_{Z^*,Z}$ *we obtain*

$$\begin{aligned}
L(y, u, p) &= \frac{1}{2}\|Qy - y_d\|_Y^2 + \frac{\lambda}{2}\|u\|_U^2 + \langle p, Ay + f(y) + Bu\rangle_{Z^*,Z},\\
L_y(y, u, p) &= Q^*(Qy - y_d) + A^*p + f'(y)p,\\
L_u(y, u, p) &= \lambda u + B^*p,\\
L_{yy}(y, u, p) &= Q^*Q + f''(y)p,\\
L_{uu}(y, u, p) &= \lambda I_U,\\
L_{yu}(y, u, p) &= L_{uy}(y, u, p) = 0.
\end{aligned}$$

Algorithm 4.8 is the so called Newton-CG method. The idea is to solve the linear equation using the conjugate gradient method and to avoid an explicit representation of the Hessian.

```
   Input: initial guess $u_0 \in \mathcal{N}(u^*)$, max steps $n_{\max}$, tolerance $tol$
1  while ($n < n_{max}$ && $\|\nabla \hat{J}(u_n)\| > tol$) do
2      Compute $b_n = -\nabla \hat{J}(u_n)$ with Algorithm 4.2 ;
3      Choose $\delta u_0^n$ and set $k = 0$;
4      Compute $r_0^n = b_n - \nabla^2 \hat{J}(u_n)\delta u_0^n$ with Algorithm 4.7 and set $d_0^n = r_0^n$;
5      while ($k < k_{max}$ and $\|r_k\|_U < tol_{CG}$) do
6          Compute $z = \nabla^2 \hat{J}(u_n) d_k^n$ with Algorithm 4.7 ;
7          Evaluate $\alpha_k^n = \frac{\langle r_k^n, r_k^n\rangle_U}{\langle d_k^n, z\rangle_U}$ ;
8          Set $\delta u_{k+1}^n = \delta u_k^n + \alpha_k^n d_k^n$ and $r_{k+1}^n = r_k^n - \alpha_k^n z$ ;
9          Evaluate $\beta_k^n = \frac{\langle r_{k+1}^n, r_{k+1}^n\rangle_U}{\langle r_k^n, r_k^n\rangle_U}$ and set $d_{k+1}^n = r_{k+1}^n + \beta_k^n d_k^n$ ;
10         Set $k = k + 1$ ;
11     end
12     Update $u_{n+1} = u_n + \delta u_k^n$ ;
13     Set $n = n + 1$ ;
14 end
```

Algorithm 4.8: Newton-CG algorithm

The Newton-CG method consists of an outer Newton loop for the optimization and the inner CG loop, in which the arising linear system is solved. The main computational effort within the CG loop is the evaluation of the Hessian times vector product (6). For this task, we have to solve one linearized state equation and one adjoint equation in each CG step. The effort to obtain the gradient in (2) is the same as in the gradient type methods, i.e., the evaluation of the nonlinear state and the linear adjoint PDE.
It seems to be appropriate to solve the linear system only with high accuracy when the current Newton iterate is close to the optimum. To save iterations in the CG loop we choose the tolerance tol_{CG} depending on the outer loop. In [49] the authors suggest the following stopping criterion for the inner loop

$$\frac{\|r_k^n\|_U}{\|\nabla \hat{J}(u_0)\|_U} \leq \min\left\{ \left(\frac{\|\nabla \hat{J}(u_n)\|_U}{\|\nabla \hat{J}(u_0)\|_U} \right)^q, p\frac{\|\nabla \hat{J}(u_n)\|_U}{\|\nabla \hat{J}(u_0)\|_U} \right\} \tag{4.9}$$

with constants $q \in (1, 2]$ and $p > 0$. Note that p is only relevant in the first steps, but afterwards the influence of q becomes dominant. The value of q determines the convergence order of the outer loop. For $q = 2$ we obtain the desired quadratic convergence behaviour. The case $q = 1$ yields a linear and $q = 1.5$ a superlinear order. It is important to note that we have a trade off behaviour: If we choose $q = 2$ we have the desired quadratic convergence for the Newton method, but a lot of CG steps have to be done in the inner loop. In contrast to that, for $q = 1.5$ we only obtain a superlinear convergence order, but we require less CG steps. Detailed numerical results concerning the number of CG steps in dependence on q can be found in [49], where the Newton-CG method was applied to an optimization problem in semiconductor design.
Since the Newton update is automatically scaled in the function space, cf. [18], a steplength of $\sigma_k = 1$ is reasonable and yields good numerical results, see Chapter 6. However, for some problems with nonconvex cost functional it is useful to shorten the stepsize.
In the literature, the Newton-CG method is also known as *inexact Newton method*. A convergence proof for finite dimensional optimization problems can be found in [76]. The application to a PDE constrained optimization problem, namely the optimal control of the Navier-Stokes equations, can be found in [51]. The convergence proof for infinite dimensional systems can also be found in this reference.

Remark 4.3
We want to point out that the main advantage of Algorithm 4.8 is to avoid the explicit formulation of the Hessian. Even in the case, where a direct method like Gaussian elimination is favourable to solve the linear equation, it is often useful to choose Algorithm 4.8. Building up the full Hessian with Algorithm 4.7 and taking the symmetry into account requires dim U_h PDE solutions. Contrary to that we need two PDEs solved in each CG step. Thus, we should prefer Algorithm 4.8 if

the condition $\frac{dim\, U_h}{2} \geq n_{cg}$ is satisfied, cf. [14]. The numerical results in Chapter 6 indicate that even for large problems only few CG steps are necessary, see also [51]. Therefore, even for boundary control problems, where the number of control variables is comparatively small, Algorithm 4.8 yields good results.

Remark 4.4
It is known that the CG method requires a symmetric and positive definite matrix. Although the Hessian of the reduced cost functional is indeed symmetric and positive definite, it is not guaranteed that these properties are preserved for the discretized Hessian. In [18] the authors suggest to use further Krylov type methods like symmetric LQ (SYMMLQ), GMRES or BiCG to overcome this problem.

BFGS Algorithm

In this section we focus on so called *Quasi-Newton methods*. These methods require, just like gradient methods, only first order information and avoid the necessity to compute Hessian operations. However, they can produce superlinear convergence and outperform the first order methods presented in Section 4.1.1. The most prominent candidate of this type is the BFGS algorithm (named after Broyden, Fletcher, Goldfarb and Shanno).
A detailed derivation of this method for finite dimensional optimization problems can be found in [76]. The idea is to find an approximation B for the true Hessian. We start with an initial guess, e.g. $B_0 = I$ and update the approximation B_n in each step by

$$B_{n+1} = B_n - \frac{B_n s_n s_n^\top B_n}{s_n^\top B_n s_n} + \frac{y_n y_n^\top}{y_n^\top s_n} \tag{4.10}$$

with $s_n = u_{n+1} - u_n$ and $y_n = \nabla \hat{J}(u_{n+1}) - \nabla \hat{J}(u_n)$. Because of scaling problems, it is not useful to apply (4.10) directly to the discretized problem. In order to generalize (4.10) we have to explain how the dyadic products are defined in the function space setting. The rank-one operator $w \otimes z \in \mathcal{L}(U), w, z \in U$ is defined by

$$(w \otimes z)(v) := \langle z, v \rangle_U w.$$

With this notation we can formulate the following function space BFGS algorithm.

```
Input: initial guess $u_0 \in \mathcal{N}(u^*)$, max steps $n_{\max}$, $B_0 \in \mathcal{L}(U)$ symmetric
1 while ($n < n_{max}$ && $\|\nabla \hat{J}(u_n)\| > tol$) do
2     Solve $B_n \delta u_n = -\nabla \hat{J}(u_n)$ ;
3     Update $u_{n+1} = u_n + \delta u_n$ ;
4     Compute $\nabla \hat{J}(u_{n+1})$ by Algorithm 4.2 ;
5     Set $s_n = u_{n+1} - u_n$, $y_n = \nabla \hat{J}(u_{n+1}) - \nabla \hat{J}(u_n)$ ;
6     Update $B_{n+1} = B_n + \frac{y_n \otimes y_n}{\langle y_n, s_n \rangle_U} - \frac{B_n s_n \otimes B_n s_n}{\langle B_n s_n, s_n \rangle_U}$ ;
7     Set $n = n + 1$ ;
8 end
```

Algorithm 4.9: BFGS algorithm

As already mentioned, the appropriate approximation of the scalar products is important. The convergence proof of this algorithm in the function space setting can be found in [51]. Note that the BFGS algorithm only requires first order gradient information and, thus, one solution of the nonlinear state equation and one of the linear adjoint equation in each step. The additional effort in contrast to the NCG algorithm is given by the necessity to solve the linear system in step (2) and the computation of the dyadic products in step (6). Moreover, just like in the Newton Algorithm 4.6, the storage of the full matrix B_n causes problems for large optimization problems. However, especially for boundary control problems, Algorithm 4.9 yields reasonable results, see Chapter 6.
One possibility to overcome the drawback of having to solve the linear system is to use updates that approximate the inverse of the Hessian, i.e. $H = B^{-1}$, instead of the Hessian itself. By applying the Sherman-Morrison-Woodbury formula, cf. [76], to (4.10) we obtain

$$H_{k+1} = H_k + \frac{s_k^\top y_k + y_k^\top H_k y_k}{(s_k^\top y_k)^2}(s_k s_k^\top) - \frac{H_k y_k s_k^\top + s_k y_k^\top}{s_k^\top y_k} \tag{4.11}$$

for the finite dimensional problem. This formula can also be generalized to the function space setting. By replacing B_{k+1} by formula (4.11) in step (6) and inserting $\delta u_k = -H_k \nabla \hat{J}(u_k)$ in step (2), we avoid the solution of the linear system.
To overcome the storage problem, as well, we consider a matrix free version of the BFGS algorithm presented in [18]. The idea is based on the observation that the algorithm described before only requires the action of H_k on the vector $\nabla \hat{J}(u_k)$, but not the matrix H_k itself. By using the definition of the dyadic products and the recursion relation for H_k, we obtain a recursive formula for $z_k = H_k y_k$, which reads

$$z_k = H_0 y_k + \sum_{j=1}^{k-1} \left[c_j \langle s_j, y_k \rangle_U r_j - \langle z_j, y_k \rangle_U s_j \right] \tag{4.12}$$

where $c_j = \langle s_j, y_j \rangle_U^{-1}, d_j = 1 + c_j \langle y_j, z_j \rangle_U$, and $r_j = d_j s_j - z_j$. The search direction is given by

$$p_{k+1} = -H_0 g_{k+1} - \sum_{j=1}^{k} \left[c_j \langle s_j, g_{k+1} \rangle_U r_j - \langle z_j, g_{k+1} \rangle_U s_j \right]. \tag{4.13}$$

The main advantage of the matrix-free BFGS Algorithm 4.10 is given by the fact that only the vectors $\{s_j, y_j, z_j\}$ are stored, but not the full inverse of the Hessian H_k. Thus, this method allows us to handle quite large optimization problems without getting into trouble with memory requirements. However, it should be mentioned that the computational effort in each optimization step is higher compared to the BFGS algorithm presented before.

```
   Input: $u_0 \in N(u^*), H_0 = I, g_0 = \nabla \hat{J}(u_0), p_0 = -g_0$
 1 Compute $u_1 = u_0 + \sigma_0 p_0$ with $\sigma_0$ satisfying Armijo;
 2 Compute $g_1 = \nabla \hat{J}(u_1), y_0 = g_1 - g_0, s_0 = \alpha_0 p_0$, and $p_1 = -H_0 g_1$ ;
 3 while ($n < n_{max}$ && $\|g_{n+1}\| > tol$) do
 4     Update $u_{n+1} = u_n + \sigma_n p_n$ with $\sigma_n$ satisfying Armijo ;
 5     Compute $g_{n+1} = \nabla \hat{J}(u_{n+1})$, $y_n = g_{n+1} - g_n$, $s_n = \sigma_n p_n$ ;
 6     Compute $z_n$ with (4.12);
 7     Compute and save $c_n = \langle s_n, y_n \rangle_U^{-1}$, $d_n = 1 + c_n \langle y_n, z_n \rangle_U$, and
       $r_n = d_n s_n - z_n$ ;
 8     Compute new search direction $p_{n+1}$ with (4.13) ;
 9     Set $n = n + 1$ ;
10 end
```

Algorithm 4.10: Matrixfree BFGS algorithm

Remark 4.5
The presented algorithms in this section have in common that they are only locally convergent, i.e., an appropriate initial guess has to be available. In the context of MPC it is often possible to obtain a good start solution from the optimal control sequence of the previous MPC step. For the first MPC step or the case of instantaneous control ($N = 2$), where no previous information exists, it is reasonable to compute few steps with the gradient method and switch to higher order algorithms if we are close to the optimum.
General globalization strategies for Quasi- and inexact Newton methods can be found in [76] for finite dimensional optimization and in [52] for PDE constrained optimization.

Primal- Dual Active Set Method

In this section we present an approach which incorporates control constraints into the optimal control algorithm. The method is known as *primal-dual active set strategy* and can be found in e.g. [58], [49] and [39]. We consider box constraints $\mathbb{U} = \{u \in L^2(Q) : u_a(x,t) \leq u(x,t) \leq u_b(x,t)\}$ with bounds $u_a, u_b \in L^\infty(Q)$. In order to keep the notation simple we will neglect the arguments x and t when it is useful. Furthermore, we look at the case of distributed control, where the control acts on the whole domain Q. The adaption to boundary control or distributed control on partial domains is straightforward.
First, we recapitulate the first order optimality condition (2.15) for the control constrained case

$$\nabla J(u) + \mu_b - \mu_a = 0 \tag{4.14a}$$

$$\mu_a \geq 0, \quad u_a - u \leq 0, \quad \langle \mu_a, u_a - u \rangle_U = 0 \tag{4.14b}$$

$$\mu_b \geq 0, \quad u - u_b \leq 0, \quad \langle \mu_b, u - u_b \rangle_U = 0, \tag{4.14c}$$

where μ_a and μ_b are the corresponding Lagrange multiplier for the inequality constraints. We define the so called *active sets*, i.e. those subsets of Q where the bounds u_a or u_b are attained, by

$$\mathcal{A}^+ := \{(x,t) \in Q : \; u(x,t) = u_b(x,t)\} \tag{4.15a}$$
$$\mathcal{A}^- := \{(x,t) \in Q : \; u(x,t) = u_a(x,t)\}. \tag{4.15b}$$

By introducing the multiplier $\mu := \mu_b - \mu_a$ we obtain the following important characterization (cf. [39] for the proof)

$$\mathcal{A}^+ = \{(x,t) \in Q : \; \mu + c(u(x,t) - u_b(x,t)) > 0\} \tag{4.16a}$$
$$\mathcal{A}^- = \{(x,t) \in Q : \; \mu + c(u(x,t) - u_a(x,t)) < 0\} \tag{4.16b}$$

with $c \in \mathbb{R}^+$. The idea of the primal-dual active set method is to determine the active sets for the **primal variable** u and then compute the **dual variable** from the optimality condition (4.14a) via

$$\mu = -\nabla J(u). \tag{4.17}$$

The primal-dual active set algorithm reads

```
   Input: Initial guess $u_0$, $\mu_0$
1  Set $k = 1$ ;
2  while ($k < k_{max}$ and stopping criterion is violated) do
3      Compute the active sets $\mathcal{A}_k^+$, $\mathcal{A}_k^-$ from $(\mu_{k-1}, u_{k-1})$ with (4.16) ;
4      Solve $\min_{u_k \in U} \hat{J}(u_k)$ s.t. $u_k = u_b$ on $\mathcal{A}_k^+$ and $u_k = u_a$ on $\mathcal{A}_k^-$ ;
5      Compute the multiplier $\mu_k$ from the current iterate by (4.17) ;
6      Set $k = k + 1$ ;
7  end
```

Algorithm 4.11: Primal-dual active set strategy

In step (3) we determine the active sets via formula (4.16) using the information of u and μ from the previous step. Subsequently, in step (4) we solve the optimization problem, where the values on the active sets are fixed by the corresponding bounds. Since the control values on the active sets $\mathcal{A}_k := \mathcal{A}_k^+ \cup \mathcal{A}_k^-$ are fixed they can be eliminated from the optimization problem. Therefore, we obtain an unconstrained optimization problem on the *inactive set* $\mathcal{I}_k = Q \setminus \mathcal{A}_k$ that can be solved, e.g., by the Newton Algorithm 4.8. In the last step (5) the multiplier μ is updated by the optimality condition (4.17). A suitable stopping criterion for this algorithm is the agreement of the active sets in two consecutive iterates. The numerical results in [39] indicate that this criterion is often satisfied within less than five steps.
Note, that Algorithm 4.11 consists of an outer iteration, where the active sets are determined, and an inner iteration, where the unconstrained optimal control problem is solved. An alternative approach is given if these iterations interchange, i.e.

the control problem forms the outer iteration and the active sets are updated in the inner iteration, cf. [58] and [39]. In the case where the optimization problems are solved by SQP methods Algorithm 4.11 is also known as *PD-SQP* and the opposite as *SQP-PD* method. The proof of local superlinear convergence for these two algorithms can be found in [58].
The last algorithm in this section is a variant of the primal-dual active set strategy, where the interpretation of these methods as *semismooth Newton* algorithm becomes more obvious. The starting point is again the optimality system (4.14) where the complementarity conditions (4.14b) can be rewritten in terms of min and max operations as

$$\nabla \hat{J}(u) + \mu = 0 \tag{4.18a}$$
$$\max\{0, \mu + c(u - u_b)\} + \min\{0, \mu + c(u - u_a)\} - \mu = 0. \tag{4.18b}$$

The idea to apply Newton's method directly to system (4.18) fails, since (4.18b) is not differentiable in a classical sense due to the presence of the min and max operators. However, (4.18b) is so called Newton differentiable (cf. [52] for the detailed definition) in the case $c = \lambda$, where λ denotes the regularization parameter. Therefore, it is possible to apply a so called semismooth Newton method, cf. [59], to system (4.18). Note the two different types of nonlinearities in (4.18): On the one hand we have the possibly highly nonlinear (due to the PDE solution operator $S(u)$), but typically smooth functional $\hat{J}(u) = J(S(u), u)$. On the other hand there are the simple, but not differentiable max and min operations.
By using the notation of the active sets the semismooth Newton method yields, cf. [49], the linear operator equation

$$\begin{pmatrix} \nabla^2 \hat{J}(u_k) & I \\ c\chi_{\mathcal{A}_k} & -\chi_{\mathcal{I}_k} \end{pmatrix} \begin{pmatrix} \delta u \\ \delta \mu \end{pmatrix} = - \begin{pmatrix} \nabla \hat{J}(u_k) + \mu_k \\ c\chi_{\mathcal{A}_k^+}(u_k - u_b) + c\chi_{\mathcal{A}_k^-}(u_k - u_a) - \chi_{\mathcal{I}_k}\mu_k \end{pmatrix} \tag{4.19}$$

In order to apply an iterative method to the (discretized) linear system, e.g. the CG algorithm, it is necessary to rewrite (4.19) in an equivalent symmetric form. This can be done by using the definition of the active sets and by taking into account that $\delta\mu = 0$ on the inactive set $\mathcal{I}_k$. We obtain

$$\begin{pmatrix} \nabla^2 \hat{J}(u_k) & \chi_{\mathcal{A}_k} \\ \chi_{\mathcal{A}_k} & 0 \end{pmatrix} \begin{pmatrix} \delta u \\ \delta \mu|_{\mathcal{A}_k} \end{pmatrix} = - \begin{pmatrix} \nabla \hat{J}(u_k) + \mu_k \\ \chi_{\mathcal{A}_k^+}(u_k - u_b) + \chi_{\mathcal{A}_k^-}(u_k - u_a) \end{pmatrix} \tag{4.20}$$

As already mentioned before it is not reasonable to build up the Hessian $\nabla^2 \hat{J}(u_k)$ explicitly, but only evaluate the matrix vector products $\nabla^2 J(u_k)d$ by Algorithm 4.7 in each CG step. Furthermore, it is possible to replace the Hessian $\nabla^2 \hat{J}(u_k)$ by a quasi- Newton approximation like the BFGS method presented in this section.

```
  Input: Initial guess $u_0$, $\mu_0$
1 Set $k = 0$ ;
2 while ($k < k_{max}$ and stopping criterion is violated) do
3     Evaluate the reduced gradient $\nabla J(u_k)$ with Algorithm 4.2 ;
4     Compute the active sets $\mathcal{A}_k^+$ and $\mathcal{A}_k^-$ with (4.16) ;
5     Solve the Newton system (4.20) iteratively via Krylov methods;
6     Update $u_{k+1} := u_k + \delta u, \quad \mu_{k+1}|_{\mathcal{A}_k} := \mu_k|_{\mathcal{A}_k} + \delta\mu_k|_{\mathcal{A}_k}, \quad \mu_{k+1}|_{\mathcal{I}_k} = 0$ ;
7     Set $k = k + 1$ ;
8 end
```

Algorithm 4.12: Semismooth Newton method

A suitable stopping criterion for Algorithm 4.12 is given by

$$\left\| \begin{pmatrix} \nabla \hat{J}(u_k) + \mu_k \\ \chi_{\mathcal{A}_k^+}(u_k - u_b) + \chi_{\mathcal{A}_k^-}(u_k - u_a) \end{pmatrix} \right\| < \varepsilon.$$

Note that in Algorithm 4.12 the active sets are updated in each Newton iteration (step (4)). In contrast to that in the PD-Newton Algorithm 4.11 the active set update follows the exact solution of the unconstrained problem. Thus, in the case where the Newton method yields the exact solution within one step in the inner iteration, both algorithms coincide. A numerical study, in which the three presented algorithm are compared can be found in [58]. Furthermore, the proof of superlinear convergence for the semismooth Newton Algorithm 4.12 can also be found in this reference. In view of the application to the discretized problem the semismooth Newton algorithm enjoys a property called *mesh independence*. This means that the number of Newton steps is independent of the mesh size (for the proof we refer to [93]). In our numerical simulations Algorithm 4.12 yields quite good results.

Further methods

In the previous section we presented some popular algorithms for PDE constrained optimization. However, there is a wide range of further algorithms and it is by far out of the scope of this thesis to give an overview of this topic. Since some of these methods are quite popular we want to mention them at least.

In contrast to the black-box approach the all-at-once method treats the control and state variables as independent optimization variables. The idea of **sequential quadratic programming (SQP)** is to solve the optimization problem (4.1) by a sequence of quadratic programming (QP) problems, i.e., problems with quadratic cost functional and linear constraints. In the case without control constraints the problem reads

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{uu} & e_y^* \\ \mathcal{L}_{uy} & \mathcal{L}_{yy} & e_u^* \\ e_y & e_u & 0 \end{pmatrix} \begin{pmatrix} y_{k+1} - y_k \\ u_{k+1} - u_k \\ p_{k+1} \end{pmatrix} = - \begin{pmatrix} J_y(y_k, u_k) \\ J_u(y_k, u_k) \\ e(y_k, u_k) \end{pmatrix}. \tag{4.21}$$

First, we note that the linear system (4.21) is quite large, even in the case that the control space is small (e.g. boundary control). This is by reason that (4.21) contains the (discretized) state, control and adjoint variables. In contrast to the reduced objective approach we obtain a large linear system with a sparse structure. Therefore, it is important to apply *preconditioned* iterative solver. In this context it is remarkable that a special choice of the preconditioner leads to a system that can be interpreted as a SQP method for the reduced system, cf. [40]. The main advantage of the SQP algorithm is that only linearized equations have to be solved. This becomes an important point if the effort for solving the nonlinear equation $e(y, u) = 0$ is much higher than for the linearized equation $e_y(y, u) = 0$. However, if it is possible to treat the nonlinearity in an explicit way during the time stepping routine (as we do this in the semi-implicit Euler method) this advantage is neglectable.

A different approach is given by the application of **multigrid methods** to the optimal control problem. The method originates from the numerical solution of PDEs where multigrid techniques are used for a long time, cf. [42]. In the context of PDE optimization the idea is to apply multigrid algorithms to the fully discretized first order optimality system. A detailed introduction to this topic can be found in [18], where especially semilinear PDEs are investigated. The application to reaction diffusion processes can be found in [17].

In the special case where linear quadratic optimal control problems without additional constraints are considered, it is possible to use **Riccati equations** to find the optimal solution. A detailed overview of how this method can be applied to MPC and how it can be generalized to semilinear PDEs by linearization techniques can be found in [48].

An alternative approach to deal with control constraints are **interior point methods**. The application to PDE constrained optimization can be found, e.g., in [98].

4.2 Proper Orthogonal Decomposition in Model Predictive Control

In Section 4.1 we presented efficient algorithms for solving the optimal control problem in each MPC step. However, for a fine discretization or a multidimensional model from a real application, online optimization is generally not possible. When we consider MPC as a low dimensional approximation in time of the infinite horizon problem, it seems to be desirable to reduce the spatial dimension as well. One approach to do this is the use of the model reduction technique Proper Orthogonal Decomposition (POD) presented in Section 2.3.1. In Figure 4.1 we see different possibilities to treat infinite dimensional system on an infinite time horizon. If we in-

clude the algorithms from Section 4.1 in the MPC step we consider high (but finite) dimensional problems on a short time horizon. (Although most of the optimization algorithms are formulated in an infinite dimensional way, we have to discretize the problem in order to solve the underlying PDEs.) The last step is the reduction from the high dimensional FEM model to a low dimension POD model.
Note that the model order reduction technique is also of interest for the infinite horizon problem. In [65] the authors reduced the infinite dimensional problem via POD and obtained a feedback by solving the Hamilton-Jacobi-Bellman equation arising from the reduced model. This method was successfully applied to the Burgers equation.
Another possibility to get a reduced order model is to decompose the differential operator into a (finite dimensional) unstable subsystem and an (infinite dimensional) stable subsystem. The idea is to control only the often low dimensional unstable system. This method is also known as *modal decomposition* and can be found in e.g. [23]. The combination of the modal decomposition with MPC for semilinear parabolic systems can be found in [28], [29] and [30].

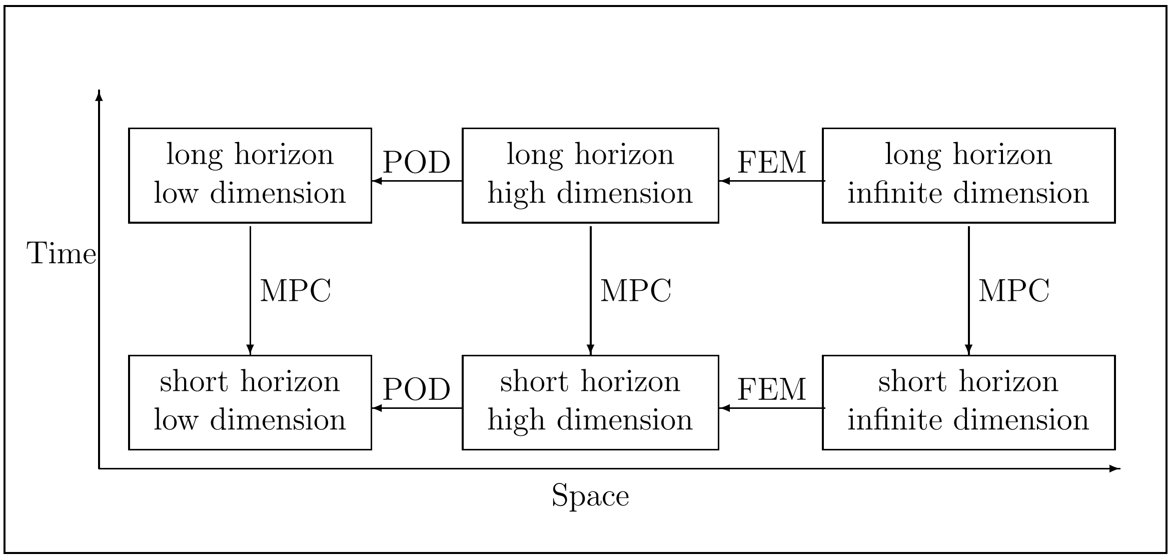

long horizon low dimension
POD
long horizon high dimension
FEM
long horizon infinite dimension
Time
MPC
MPC
MPC
short horizon low dimension
POD
short horizon high dimension
FEM
short horizon infinite dimension
Space

Figure 4.1: Relation between different approaches

The combination of MPC and POD was investigated some years ago for applications in flow control, e.g. [16], and chemical engineering [54]. In order to present algorithms where we combine MPC with POD we repeat the main steps of the POD method:

- Solve the state equation to obtain snapshots $y_j = y(t_j)$
- Build up the correlation matrix $K_{ij} = \frac{1}{n}\langle y_i, y_j\rangle_{L^2}$

- Compute eigenvalues and eigenvectors of K
- Determine the ansatz functions for the Galerkin approximation and build up the reduced model

An important question is how to obtain the control sequence for the simulations of the snapshots. In practical applications it is often possible to give an initial guess by physical considerations, see also the discussion in [37]. Furthermore, there exist methods where the snapshots are generated by random controls. An easy adaption strategy is given by the following idea: Start with an initial control, e.g. $u = 0$, and compute the arising snapshots. The optimal control of the corresponding reduced system is used to update the snapshot ensemble. In case the change of the control in two consecutive steps is small enough the algorithm terminates. A more elaborate variant of this algorithm can be found in [27]. One advantage of the MPC method is that a suitable initial guess is already known from the previous step. Therefore, it is in general not necessary to use the adaption strategy. The numerical results in Chapter 6 and in [3] indicate that no adaption strategy is required even for the first step, where no initial guess is available.

There are several possibilities to combine MPC with POD. One obvious method is given in Algorithm 4.13. The idea is to compute the snapshots on a long time horizon (1) and build up the reduced model (2). These steps can be done offline before the MPC algorithm comes into play. The optimal control problem in each MPC step (5) is solely solved for the reduced problem (2.50). By using (2.48) the optimal control sequence of the reduced model provides an approximation of the high dimensional control problem $u = \mathcal{P}(u^{ROM})$. The first element of this sequence is implemented to the full system (6) and yields the state for the next iteration (7). The essential property and the advantage of this algorithm is that we build up the reduced system only once before the online computation begins.

```
1 Create snapshots by solving the state equation on a long horizon ;
2 Build up the reduced order model $f_{ROM}$ ;
3 for $i = 1, 2, \dots$ do
4     Project the current state to the reduced state $y^{ROM}(i) = \mathcal{S}y(i)$;
5     Solve the reduced order optimal control problem with horizon $N$;
6     Compute the first element of the full control sequence by the reduced
      model $F(y(i)) := u(0) = \mathcal{P}(u^{ROM}(0))$;
7     Compute new state $y(i+1) = f(y(i), F(y(i)))$ and set $i := i + 1$;
8 end
```

Algorithm 4.13: MPC-POD-Algorithm 1

Obviously, the algorithm only yields reasonable results if the system does not change so much and, thus, the reduced model remains a good approximation of the full model during the process. In [100] this algorithm was combined with linearization

techniques.
In order to overcome the drawback from the previous algorithm we look at Algorithm 4.14. In contrast to Algorithm 4.13 the reduced model is computed within each MPC step (3). However, the snapshots are only computed for the optimization horizon N to obtain an appropriate model for the current optimization problem.

```
1 for $i = 1, 2, \dots$ do
2     Create snapshots by solving the state equation on the horizon $N$;
3     Build up the reduced order model $f_{ROM}$ ;
4     Project the current state to the reduced state $y^{ROM}(i) = \mathcal{S}y(i)$;
5     Solve the reduced order optimal control problem with horizon $N$;
6     Compute the first element of the full optimal control sequence by the
      reduced model $F(y(i)) := u(0) = \mathcal{P}(u^{ROM}(0))$;
7     Compute new state $y(i+1) = f(y(i), F(y(i)))$ and set $i := i + 1$;
8 end
```

Algorithm 4.14: MPC-POD-Algorithm 2

The drawback of this method is that all computations for obtaining the reduced model, including the simulation for the snapshots, have to be done online.

It seems to be desirable to update the reduced model only in the case when a suitable approximation of the full system fails. Otherwise, the model should remain the same during the MPC steps. The idea to estimate the quality of the reduced model can be found in Figure 4.2.

$$\begin{tikzcd}
\boxed{u^{ROM}(i)} \arrow[r, "\mathcal{P}"] \arrow[d, "f_{ROM}"] & \boxed{u(i)} \arrow[d, "f"] \\
\boxed{y^{ROM}(i+1)} \arrow[r, "\mathcal{P}"] & \begin{array}{|c|}\hline y_1(i+1) \\ \hline y_2(i+1) \\ \hline \end{array}
\end{tikzcd}$$

Figure 4.2: Model error estimator

The previous algorithms coincide in the following two steps: Compute the high dimensional control from the reduced control sequence and apply the feedback to the full model to obtain the next state. Thus, the steps to obtain the state

$y_1 = f(y(i), F(y(i)))$ are those we have to do anyway. Furthermore, the state of the reduced model y^{ROM} at the next time step is a byproduct of the optimization problem. The only additional step is to compute $y_2 = \mathcal{P}(y^{ROM})$ via formula (2.48), which can be done quite fast. If we consider the relative difference between y_1 and y_2 in an appropriate norm we get a measure of the viability of the current model. Note that this procedure only yields a statement about the quality of the model but not about the quality of the control. This idea leads to Algorithm 4.15. While the relative model error e is below a given bound ε the reduced model does not change. If this criterion fails the reduced model is updated and the current step is computed again.

```
1  Create snapshots by solving the state equation ;
2  Build up the reduced order model $f_{ROM}$ ;
3  $e = 0$, $k = 0$;
4  while $e \leq \varepsilon$ do
5      Project the current state to the reduced state $y^{ROM}(i) = \mathcal{S}y(i)$;
6      Solve the reduced order optimal control problem with horizon $N$;
7      Compute the first element of the full optimal control sequence by the
       reduced model $F(y(i)) := u(0) = \mathcal{P}(u^{ROM}(0))$;
8      Compute new state $y_1 = y(i+1) = f(y(i), F(y(i)))$ ;
9      Compute the full state from the reduced state $y_2 = \mathcal{P}(y^{ROM}(i+1))$;
10     Evaluate the model error $e = \|y_1 - y_2\|_{L^2} / \|y_1\|_{L^2}$;
11     if $e > \varepsilon$ && $k > 0$ then
12         GOTO 1
13     else
14         Set $i := i+1, k := k+1$;
15     end
16 end
```

Algorithm 4.15: MPC-POD-Algorithm 3

In order to demonstrate the algorithm we look at the example of the catalytic rod presented in Chapter 6. In Figure 4.3 the error of the model for the three presented algorithms is displayed. In this example we consider 50 MPC steps and the prescribed error is given by $\varepsilon = 0.01$. This means that we tolerate a maximum relative error of 1% in each step. The blue line displays the model error of Algorithm 4.13 in which the reduced model is built up only one time. Obviously, the error is rapidly monotonically increasing (at the final time $t = 0.5$ a relative error of $\approx 150\%$ is reached). In contrast to that the error of Algorithm 4.14 (black line), where the reduced model is built up in each time step, is below $\approx 0.1\%$. The red line shows the error for Algorithm 4.15. It is evident that the value is always below the prescribed bound of 1%. Moreover, it can be seen that the reduced model is built up at the beginning and it has to be recomputed three times in order to stay below the bound.

As we will see in Chapter 6 the computing time for the additional three updates of Algorithm 4.15 compared to Algorithm 4.13 is rather small. In contrast to that, Algorithm 4.14 needs more than twice the computation time due to the 49 additional updates. The numerical results indicate that Algorithm 4.14 yields a good tradeoff between computing time and model accuracy.

Finally, we want to remark that the variable k in Algorithm 4.15 ensures that no

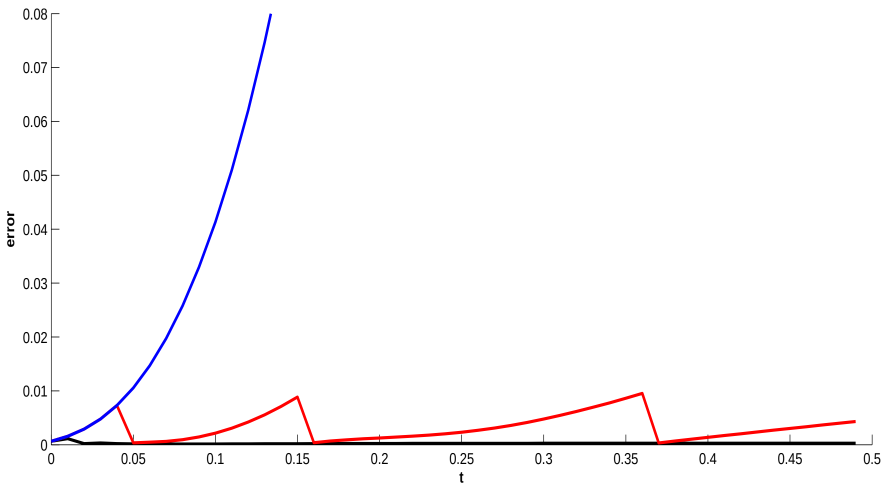


Figure 4.3: Error of the model for Algorithm 4.13 (blue), Algorithm 4.14 (black) and Algorithm 4.15 (red)

endless loop can occur. Otherwise, this might be possible if the reduced model is too bad to undercut the model error in the first step. Furthermore, it allows us to choose $\varepsilon = 0$, see Section 6.3.

4.3 Adaptive Horizon MPC

An important parameter for the computational effort of the MPC algorithm is given by the length of the optimization horizon N. As we will see in Chapter 6 the computing cost grows with the horizon length. Therefore, we have a tradeoff between a higher degree of suboptimality (see Section 1.2) and the computing time. In Chapter 3 we presented results that allow for estimating the minimal horizon that satisfies a desired suboptimality degree. Although the results are useful for a qualitative analysis they are quite conservative for a concrete example. Furthermore, it seems to be reasonable that an appropriate horizon depends on the current state and, thus, changes during the runtime of the algorithm. The idea of adaptive horizon algorithms can be found in [38], [77] and [45]. Since these algorithms are formulated for arbitrary metric spaces, they are directly applicable to infinite dimensional systems.

However, to our knowledge the presented algorithms were not applied to PDE systems yet. One well known drawback of these algorithms is the possibility that the computational burden for the additional optimization problems is much higher than the benefit of shorter horizons. Surprisingly, this fact makes adaptive algorithms interesting for the application to PDEs. In Section 4.2 we demonstrate that the effort for solving PDE constrained optimal control problems can be significantly reduced by using low dimensional models. These concepts can also be used for solving the optimization problems arising in adaptive horizon MPC.
In Section 4.3.1 we take a look at some ideas of the adaptive horizon methods described in [77] and [38]. In the ensuing Section 4.3.2 we present a new algorithm to combine adaptivity with hierarchical grid methods.

4.3.1 Algorithms for Adaptive Horizon MPC

The algorithms we introduced in this section are essentially based on the relaxed Lyapunov inequality $V_N(y) \geq V_N(f(y, F_N(y)) + \alpha l(y, F_N(y))$ we already used in Theorem 1.20. In this theorem it was required that the inequality holds for all $y \in Y$. Hence, this α is also called *global suboptimality degree.* Since this leads in general to quite conservative values of α it was suggested by [77] to only consider those points of the state space Y which are visited by the trajectory. This leads to the following theorem, whose proof can be found in [77]:

Theorem 4.6
Consider a feedback law $F_N \colon Y \to U$ and its associated trajectory $y(\cdot)$ with initial value $y(0) = y_0 \in Y$. If there exists a function $V_N \colon Y \to \mathbb{R}$ satisfying

$$V_N(y(n)) \geq V_N(y(n+1)) + \alpha l(y(n), F_N(y(n)) \tag{4.22}$$

for some $\alpha \in (0, 1]$ and all $n \in \mathbb{N}_0$ then

$$\alpha V_\infty(y(n)) \leq \alpha V_\infty^{F_N}(y(n)) \leq V_\infty(y(n)) \tag{4.23}$$

The corresponding α value is called the *closed-loop suboptimality degree.* Note that $V_N(y(n))$ and $l(y(n), F_N(y(n)))$ are byproducts of the optimization problem and they are known at time n. However, since $V_N(y(n+1))$ is not available at time n the condition (4.22) cannot be checked online. Nevertheless, by using a posteriori information, i.e. at time $n+1$ we determine the α value from the previous step, (4.22) can be used for adapting the horizon, cf. [77] for details. The drawback of these algorithms is that they cannot ensure the desired suboptimality for the closed loop system. In principle, it is possible to compute $V_N(y(n+1))$ at time n by solving an additional optimal control problem. However, due to measurement and modelling errors we cannot expect to know $y(n+1)$ exactly (see also the remarks in [77]). Furthermore, this implementation would double the computational burden. To overcome these problems there are some approaches for a priori estimates. Since this is not the main topic of this section, we do not go into detail. Elaborate

information can be found in [77].
In this section we follow the ansatz of [38]. In order to indicate that the horizon is no longer constant, we use the notation $N(n)$ for the horizon at time n. The idea is to compute an upper bound for the unknown $V_{N(n+1)}(y(n+1))$ that is available at time n, i.e.

$$\bar{V}_{N(n+1)}(y(n+1)) \geq V_{N(n+1)}(y(n+1)). \tag{4.24}$$

Since this upper bound can be quite far away from $V_{N(n+1)}(y(n+1))$, it is possible to obtain quite conservative values for α. Now, the key step is to introduce a so called *slack variable* $s(n)$ that relaxes inequality (4.22) by the conservatism from the previous steps. This leads to the following theorem which proof can be found in [38] and provides the basis for Algorithm 4.16.

Theorem 4.7
Consider a closed loop trajectory $y(\cdot)$ according to (1.29) *and suppose that for a pre-specified $\bar{\alpha} \in (0,1)$ we can find a control horizon $N(n) \in \mathbb{N}_2$ such that*

$$V_{N(n)}(y(n)) \geq \bar{V}_{N(n+1)}(y(n+1)) + \bar{\alpha} l(y(n), F_{N(n)}(y(n))) + s(n) \tag{4.25}$$

where

$$s(n) = s(n-1) + \alpha l(y(n-1), F_{N(n-1)}(y(n-1))) + \bar{V}_{N(n)}(y(n)) - \bar{V}_{N(n-1)}(y(n-1)) \tag{4.26}$$

and $s(0) = 0$ hold for all $n \in \mathbb{N}_0$. Then

$$\alpha V_\infty^{F_N}(y(0)) \leq V_\infty(y(0)). \tag{4.27}$$

This theorem can directly be used in the following algorithm, which was developed in [38]:

```
   Input: $y(0), N_{\min}, \bar{\alpha}$
 1 Calculate $V_{N(n)}(y(n))$;
 2 Calculate $\bar{V}_{N(n+1)}(y(n+1))$ ;
 3 while condition (4.25) is violated do
 4     Set $N(n) := N(n) + 1$;
 5     Calculate $V_{N(n)}(y(n))$;
 6     Calculate $\bar{V}_{N(n+1)}(y(n+1))$ ;
 7 end
 8 Apply $u(n)$ and get $y(n+1)$ ;
 9 Set $N(n+1) := \max(N(n) - 1, N_{min})$ ;
10 Set $n := n + 1$ and go to 1
```

Algorithm 4.16: Adaptive Horizon MPC

The procedure is simple: We enlarge the horizon until condition (4.25) is satisfied (step 3-7). If an appropriate horizon is found, the control is applied to the system

(8) and the horizon is reduced (9). In this algorithm the optimization horizon can be maximally reduced by one in each MPC step. In Section 6.4 it turns out that for a semilinear PDE with distributed control step (9) is not useful: Since the horizon stays constant for a long time the algorithm has to increase and decrease the horizon in each step. To overcome this problem we introduce an additional criterion to reduce the horizon depending on the current α value. The numerical examples in Chapter 6 indicate that $\alpha > \rho\bar{\alpha}$ with a prescribed $\rho > \frac{3}{2}$ yields good results.
One possibility to define $\bar{V}_{N(n+1)}(y(n+1))$ is given by

$$\bar{V}_{N(n+1)}(y(n+1)) = V_N(y(n)) - l(y(n), u(n)) - l(y_e(n), u_e(n)) + V_{N_{max}-N+1}(y_e(n))$$

where the first three terms are already known after the evaluation of $V_N(y(n))$, cf. [38]. N_{max} denotes a horizon, where the closed loop performance of $\bar{\alpha}$ can be guaranteed. $y_e(n)$ is the prediction at time n of the value $y(n+N-1)$. To ensure that this idea actually reduces the computational effort, the condition $N_{\min} \geq N_{max}/2$ should be satisfied, cf. [38].
The advantage of this approach is that the conservatism of the upper bound does not play an important role due to the slack variable. Furthermore, Theorem 4.7 guarantees a closed loop suboptimality degree of at least $\bar{\alpha}$ for Algorithm 4.16.

4.3.2 Multigrid Methods in Adaptive MPC

As already mentioned, the effort for solving the additional optimal control problems in adaptive horizon MPC is often higher than the benefit. This leads to the idea to combine the adaptive algorithms from the previous section with model order reduction. One concept originates from the multigrid method for solving PDEs and uses hierarchical grids for the spatial discretization. The method is based on the numerical observation in Section 6.2 that the minimal stabilizing horizon is not very sensitive with respect to the spatial discretization.

	Distributed		Boundary	
Grid	Horizon	Time (s)	Horizon	Time (s)
0.1	6	0.07	9	0.45
0.04	6	0.24	8	0.92
0.02	6	0.80	8	1.98
0.01	6	3.88	8	4.59

Table 4.1: Minimal stabilizing horizon for distributed and boundary control with varying spatial discretizations determined by numerical simulations of the MPC closed loop

In Table 4.1 we see the numerical results for the minimal stabilizing horizon for a nonlinear one dimensional PDE with either distributed or boundary control depending on the mesh size h. The details concerning the equation, optimizer and parameter can be found in Section 6.2. It is remarkable that in the distributed case the minimal stabilizing horizon is the same for each grid size. Except for the coarsest grid, we see the same behaviour for the boundary controlled case. Furthermore, the computing time for 100 MPC steps is displayed. Obviously, the computational burden is significantly reduced for the coarse grid compared to the finest one. However, as mentioned before the difference with respect to the optimization horizon is small. This observation is used for the construction of Algorithm 4.17. In step (1) the current state $y(n)$ is restricted to the state on the coarse grid $y_h(n)$. In a finite element implementation $h > 0$ is typically given by the diameter of the triangulation. I^h denotes the well known *restriction operator* from multigrid theory, cf. [42], and $V^h_{N(n)}(y_h(n))$ is the optimal value function for the discretized system. Based on the coarse grid model an appropriate horizon N is determined similarly to Algorithm 4.16, step (2)-(9). Table 4.1 indicates that these steps can be computed quite fast. In step 10 the full optimal control problem is solved with the priorly determined horizon. Afterwards, the control $u(n)$ is applied to obtain the next state $y(n+1)$. From the theoretical point of view the single purpose of the *while loop* is to determine the horizon. In the practical implementation one can use the information that the optimal control problem with the current horizon was already solved on the coarse grid. Therefore, it is possible to obtain with $u(n) = I_h u_h(n)$ a good initial guess for the full optimization problem. I_h denotes the so called *prolongation operator*. This idea is investigated in Section 6.4.

```
   Input: $y(0), N_{\min}, \bar{\alpha}, h$
1  $y_h(n) = I^h(y(n))$ ;
2  while $N(n) \le N_{\max}$ do
3      Calculate $V^h_{N(n)}(y_h(n))$;
4      Calculate $\bar{V}^h_{N(n+1)}(y_h(n+1))$ ;
5      if condition (4.25) holds then
6          BREAK;
7      end
8      Set $N(n) := N(n) + 1$;
9  end
10 Calculate $V_{N(n)}(y(n))$ ;
11 Apply $u(n)$ and get $y(n+1)$ ;
12 Set $N(n+1) := \max(N(n) - 1, N_{min})$;
13 Set $n := n + 1$ and GOTO 1
```

Algorithm 4.17: Hierarchical grids in adaptive horizon MPC

Of course, the algorithm only yields reasonable results if the α value on the coarse grid is close to that of the full system. A justification that this the case is given in Theorem 4.8. The goal is to compare at time n the local degree of suboptimality for the full system with that for the system on the coarse grid.

Theorem 4.8

Let $y(n)$ be a closed loop trajectory and $y_h(n) = I^h y(n)$ the restriction to the dis-

cretized grid. Assume that $|V(y) - V_h(y_h)| < \varepsilon$ and $V(y) < \infty$ for all $y \in Y$. Furthermore, let $l(y(n), 0) \neq 0$. Then it holds

$$|\alpha - \alpha_h| < C(y)\varepsilon \tag{4.28}$$

where $\alpha_h := \frac{V_N^h(y_h(n)) - V_N^h(y_h(n+1))}{l(y_h(n), u_h(n))}$ is the local α value for the discretized problem.

Proof. To shorten the notation we define $x = y(n)$ and $y = y(n+1)$. From the boundedness of $V(x)$ and the assumption $|V(x) - V_h(x_h)| < \varepsilon$ we directly obtain the boundedness of $V(y), l(x,u), l(x_h, u_h), V_h(x_h)$ and $V_h(y_h)$. With the definition of α we get

$$\begin{aligned}
|\alpha - \alpha_h| &= \left| \frac{V_N(x) - V_N(y)}{l(x,u)} - \frac{V_N^h(x_h) - V_N^h(y_h)}{l(x_h,u_h)} \right| \\
&= \left| \frac{l(x_h,u_h)V_N(x) - l(x_h,u_h)V_N(y) - l(x,u)V_N^h(x_h) + l(x,u)V_N^h(y_h)}{l(x,u)l(x_h,u_h)} \right. \\
&\quad - \left. \frac{l(x_h,u_h)V_N^h(x_h) - l(x_h,u_h)V_N^h(x_h) - l(x_h,u_h)V_N^h(y_h) + l(x_h,u_h)V_N^h(y_h)}{l(x,u)l(x_h,u_h)} \right| \\
&\leq \frac{l(x_h,u_h)\left|V_N(x) - V_N^h(x)\right| + V_N^h(x_h)\left|l(x_h,u_h) - l(x,u)\right|}{l(x,u)l(x_h,u_h)} \\
&\quad + \frac{l(x_h,u_h)\left|V_N^h(y) - V_N(y)\right| + V_N^h(y_h)\left|l(x,u) - l(x_h,u_h)\right|}{l(x,u)l(x_h,u_h)} \\
&\leq \frac{4M(y)\varepsilon}{l(x,u)l(x_h,u_h)} = C(y)\varepsilon
\end{aligned}$$

with $M(y) = \max\{V_h(x_h), V_h(y_h)\}$. □

Remark 4.9
It is important to note that the estimate (4.28) becomes arbitrary bad if the state is close to the equilibrium. In order to overcome this problem it is useful to introduce the concept of practical suboptimality. This means that the suboptimality inequality only holds outside an ε ball around the equilibrium. See [44] for a precise definition and further remarks.

For a linear PDE with quadratic cost functional and a discretization with linear finite elements it is known, cf. [72], that $|V(y) - V_h(y_h)| < ch^2$ and, thus, we get $|\alpha - \alpha_h| = \mathcal{O}(h^2)$. Estimates concerning the discretization error of the optimal value function for more general PDEs and cost functionals can be found in [74].
It should be mentioned that the previous theorem in general cannot be used to estimate the maximal deviation between the determined horizon for the coarse grid and that one of the full system. On the one hand, the generic constants from Theorem 4.8 are in general unknown, as well as, the constants C and σ from the exponential controllability condition. On the other hand, solving the α formula (1.35) for the horizon N is a nontrivial task and in general impossible.

Remark 4.10

Note, that the notation $V_N(x)$ in Algorithm 4.17 suggests that we solve the optimal control problem exactly without a spatial discretization. This is of course not true and was only done to simplify the notation. In the real implementation we have not only the coarse grid with mesh size h but also a fine grid with mesh size H. The restriction operator is then given by I_H^h. However, Theorem 4.8 is also applicable for this case.

Remark 4.11

It is a natural idea to replace the coarse grid in Algorithm 4.17 by a reduced POD model. Actually, the resulting algorithm has similar properties as Algorithm 4.17. However, the numerical investigations indicate that the additional time for generating the reduced POD model in each MPC step is higher than the benefit caused by the adaptive horizon. The idea to combine Algorithm 4.15 with the adaptive horizon method seems to be more interesting. However, in the numerical simulation we observe the following tradeoff behaviour: Algorithm 4.15 is successful if the system does not change that much and, thus, the POD model remains constant for some MPC steps. In contrast to that, the adaptive horizon algorithms are reasonable if the system behaviour changes.

5 Numerical Implementation

The content of this chapter is twofold: In Section 5.1 we introduce some test examples and describe their stability behaviour. In detail we want to motivate the *Schlögl* equation and the model of the *catalytic rod*. The nonlinearity of Schlögl type originates from the theoretical description of chemical reactions. Because of the non-monotonicity of the resulting system, this equation is of particular interest for theoretical as well as for practical reasons. In the subsequent motivation we follow [70] and [20]. The catalytic rod equation is a frequently used model in chemical engineering, cf. [23], in order to describe the temperature distribution of an exothermic catalytic reaction.
In the second part of this chapter we present an overview of the practical contribution of this thesis. The modular structure of the *C++* code is introduced and the application to an example is displayed. The implementation is based on the algorithms introduced in Chapter 4.

5.1 Examples

5.1.1 Schlögl Equation

The first considered system was discussed by Schlögl [83] as a model for an autocatalytic reaction scheme

$$A_1 + 2X \underset{k_1^-}{\overset{k_1^+}{\rightleftharpoons}} 3X, \qquad\qquad X \underset{k_2^-}{\overset{k_2^+}{\rightleftharpoons}} A_2.$$

with velocity constants $k_1^+, k_1^-, k_2^+, k_2^-$. A_1 and A_2 are chemicals with known constant concentrations c_1 and c_2 respectively. The constancy of the concentrations can be assured by continuously feeding the reactor where the reaction takes place. Then the reaction kinetics of the chemical X is given by the cubic polynomial

$$f(y) = -k_1^- y^3 + k_1^+ y^2 - k_2^+ y + k_2^- c_2$$

where y denotes the concentration of X. For suitable velocity constants, $f(y) = 0$ has three real-valued roots and we can rewrite $f(y)$ as

$$f(y) = -k(y - y_1)(y - y_2)(y - y_3)$$

with constant parameters k and y_i $(i = 1, 2, 3)$. By combining the reaction term with the diffusion of X we obtain the PDE for the concentration field $y(x, t)$

$$y_t(x,t) = D\Delta y(x,t) - k(y(x,t) - y_1)(y(x,t) - y_2)(y(x,t) - y_3) \tag{5.1}$$

where D denotes the diffusion coefficient. For unbounded domains or large domains with Neumann boundary conditions it is known, cf. [20], that the solution of (5.1) has a wave type behaviour. This can lead to numerical difficulties in the optimal control problem. In our examples with small domains the travelling waves do not play a role.

In the following we consider the case $y_1 = -1$, $y_2 = 0$ and $y_3 = 1$. Then the nonlinearity reduces to $f(y) = -\mu y(y+1)(y-1) = \mu(y - y^3)$ where $k = \mu$ is the reaction parameter. The resulting reaction diffusion equation is in the literature also known as *Chaffee Infante equation.* Since this model is an easy representative of an equation with a non monotone nonlinearity, it is well suited as a benchmark example for optimization algorithms. Our first test example is a one dimensional system with distributed control. We look at

$$\begin{aligned} y_t(x,t) &= y_{xx}(x,t) + \mu(y(x,t) - y(x,t)^3) + u(x,t) && \text{in} \quad (0,1)\times(0,\infty) && (5.2a)\\ y(0,t) &= y(1,t) = 0 && \text{in} \quad (0,\infty) && (5.2b)\\ y(x,0) &= y_0(x) && \text{in} \quad (0,1) && (5.2c) \end{aligned}$$

where we impose a homogeneous Dirichlet condition on the boundary. As already discussed in Chapter 3 the origin ($y \equiv 0$) is an unstable equilibrium for $\mu \geq \pi^2$. Moreover, as we can see in Figure 5.1 (for $\mu = 15$), there are in addition two stable equilibria (solid lines). It can be proven, cf. [21], that for a positive initial function $y_0(x)$ the solution converges to the positive equilibrium for $t \to \infty$ and vice versa for negative initial data. Furthermore, any solution is global and there is no blow up in finite time, see also Remark 3.2. The solution trajectory of the uncontrolled equation can be found in Figure 6.1 with $y_0(x) = 0.2\sin(\pi x)$.

Although there exist models in application where a distributed control in the whole spatial domain makes sense, it is often more reasonable that the control can only act on parts of the boundary. Therefore, our next test example is the one dimensional Schlögl equation with Neumann boundary control on the right side

$$\begin{aligned} y_t(x,t) &= y_{xx}(x,t) + \mu(y(x,t) - y(x,t)^3) && \text{in} \quad (0,1)\times(0,\infty) && (5.3a)\\ y(0,t) &= 0 && \text{in} \quad (0,\infty) && (5.3b)\\ y_x(1,t) &= u(t) && \text{in} \quad (0,\infty) && (5.3c)\\ y(x,0) &= y_0(x) && \text{in} \quad (0,1). && (5.3d) \end{aligned}$$

On the left side we impose an homogeneous Dirichlet condition. Since in many chemical engineering problems only the flux and not the value on the boundary can be controlled, the use of Neumann control seems to be reasonable. In Chapter 3 we have seen that the uncontrolled equation is unstable for $\mu \geq \frac{\pi}{4}$. In this case we have again two stable and one unstable equilibrium. The three equilibria of the PDE (5.3) are displayed in Figure 5.2 for $\mu = 15$.

In the last example for the Schlögl model we consider a two dimensional equation

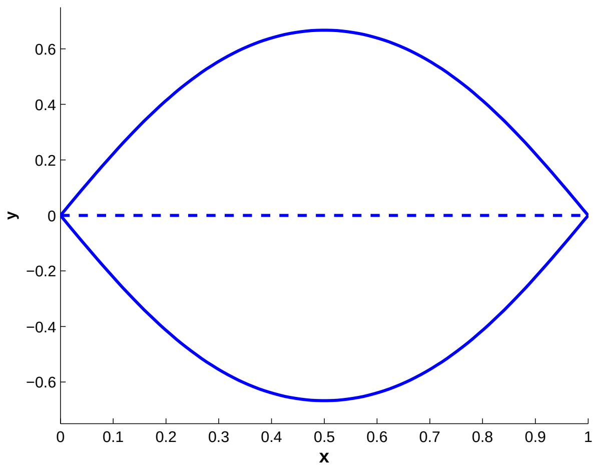


Figure 5.1: Stable equilibria (solid lines) and unstable equilibrium (dashed line) of the uncontrolled one dimensional Schlögl equation (5.2) with $\mu = 15$.

with Neumann control on some parts of the boundary

$$\begin{aligned} y_t(x,t) &= \Delta y(x,t) + \mu(y(x,t) - y(x,t)^3) && \text{in} \quad \Omega \times (0,\infty) && (5.4a)\\ \partial_\nu y(x,t) &= u(x,t) && \text{on} \quad \Gamma_{N_c} \times (0,\infty) && (5.4b)\\ \partial_\nu y(x,t) &= 0 && \text{on} \quad \Gamma_{N_0} \times (0,\infty) && (5.4c)\\ y(x,0) &= y_0(x) && \text{in} \quad \Omega && (5.4d) \end{aligned}$$

The domain is given by the unit square with circular hole, see Figure 5.3 (a) for the detailed description. The boundary $\partial\Omega$ is divided into two parts: The boundary of the square Γ_{N_0} and the boundary of the inner circle Γ_{N_c}. On Γ_{N_0} we impose a homogeneous Neumann condition while the Neumann control acts on Γ_{N_c}.
If we interpret the solution $y(x,t)$ as a temperature field, the homogeneous Neumann condition on the boundary of the square can be seen as a perfect isolation of the domain. The circle inside the domain can be considered as a cooling device to control the temperature in the reactor by the heat flux.
For $\mu \neq 0$ it is obvious that the uncontrolled equation has the three equilibria $y \equiv -1, y \equiv 0$ and $y \equiv 1$. (In the special case $\mu = 0$ we get that $y \equiv c$ is an equilibrium for each $c \in \mathbb{R}$.) The equilibrium $y \equiv 0$ is again unstable for arbitrary $\mu > 0$ while $y \equiv -1$ and $y \equiv 1$ are stable.

In order to solve the PDE (5.4) we discretize the spatial variable by the finite element method introduced in Section 2.3.1. We use the *Matlab PDE Toolbox* to generate a mesh of triangular piecewise linear finite elements. The assembling is also done by *Matlab*. Since we do not utilize adaptive algorithms in space these steps can be done offline before the MPC algorithm starts. For the simulations in Chapter 6 we consider the three different meshes displayed in Figure 5.3 (b)-(d).

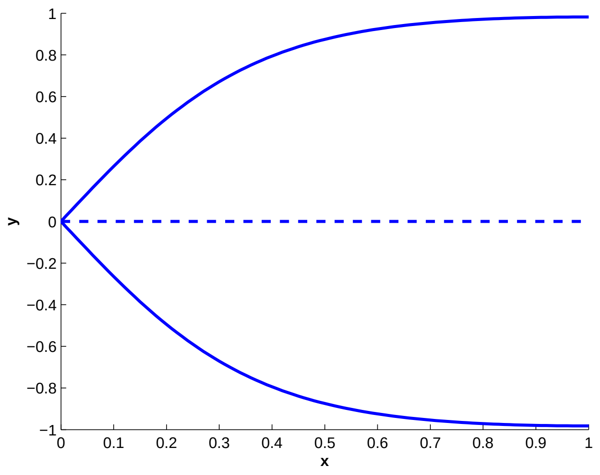


Figure 5.2: Stable equilibria (solid lines) and unstable equilibrium (dashed line) of the uncontrolled one dimensional Schlögl equation (5.3) with $\mu = 15$.

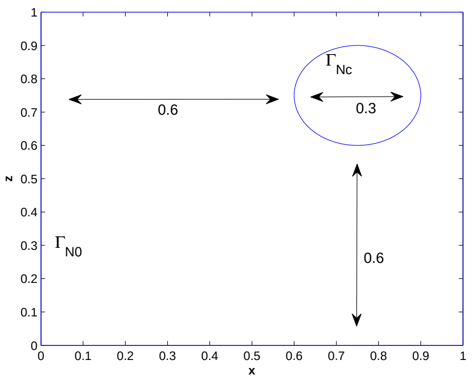
The corresponding information about the triangulation can be found in Table 5.1. With $h_{\max}$ we denote the maximum diameter of the triangles. The corresponding number of nodes is given by M_x. Since we use linear finite elements, M_x coincides with the number of state variables. The number of control variables (in this case the number of nodes on the circle) is denoted by M_u. Obviously, the control dimension grows much slower than the dimension of the state. This is due to the fact that we consider boundary control.

	$h_{\max}$	M_x	M_u
Grid I	0.10	183	16
Grid II	0.05	652	24
Grid III	0.02	4089	48

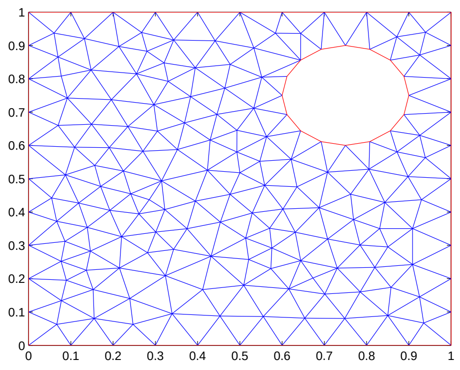
Table 5.1: Parameter values for the triangulations in Figure 5.3 (b)-(d) with maximum triangle diameter $h_{\max}$, number of state variables M_x and control variables M_u.

5.1.2 Catalytic Rod

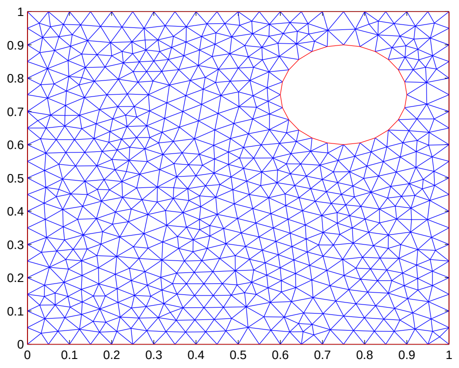
The catalytic rod model is a frequently used example in chemical engineering, cf. [23]. It describes an exothermic catalytic reaction $A \rightarrow B$ which takes place in a long thin rod, see Figure 5.4. Since the reaction is exothermic the task is to control the temperature of the rod. Controlling inside the domain is for this model in principle possible, because the rod is coated by a cooling medium. By assuming constant



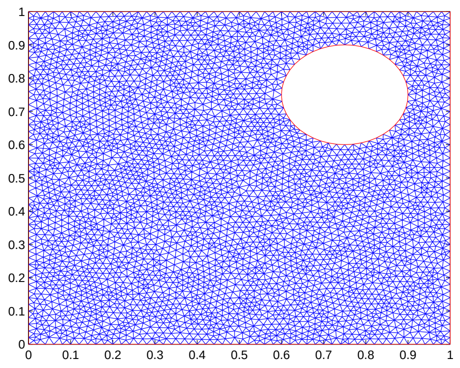
(a) Geometry



(b) Grid I



(c) Grid II



(d) Grid III

Figure 5.3: Geometry of the domain (a) and triangulation for the FEM solver.

density, heat capacity, conductivity and a constant temperature on the boundary, the dimensionless rod temperature $y(x,t)$ can be described as

$$y_t(x,t) = y_{xx}(x,t) + \beta_T(e^{-\frac{\gamma}{1+y(x,t)}} - e^{-\gamma}) + \beta_U(v(x,t) - y(x,t)) \tag{5.5a}$$

$$y(0,t) = y(\pi,t) = 0 \tag{5.5b}$$

$$y(x,0) = y_0(x). \tag{5.5c}$$

The constants are given by the dimensionless heat of reaction β_T, the dimensionless activation energy γ and the dimensionless heat transfer coefficient β_U. We want to point out that in this model the control can generally not act in each spatial point independently. In [23] the author suggest to use $v(x,t) = b(x)u(t)$ where $b(x)$ is a given shape function, i.e., we have only a one dimensional control variable $u(t)$. This assumption simplifies the control problem and it is more realistic than $v(x,t) = u(x,t)$. One possibility of the actuator distribution is given by $b(x) = \sqrt{\frac{2}{\pi}}\sin(x)$, cf. [23]. This idea takes into account that the *hot-spot* is in the middle of the rod and

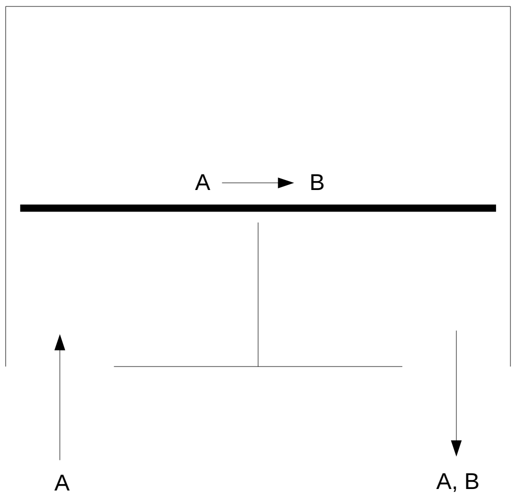


Figure 5.4: Catalytic rod.

we need the maximum cooling at this position. A further possibility is that we only can control some parts of the domain, cf. [11]. In this case we have $b(x) = \mathbb{1}_\omega$, where $\mathbb{1}$ denotes the indicator function and $\omega \subseteq (0, \pi)$. If we consider a model with several actuators we obtain $v(x,t) = \sum_{i=1}^{k} b_i(x)u_i(t)$ with control functions $u_i(t), i = 1 \dots k$. The typically used process parameters for this model, cf. [23], are given by

$$\beta_T = 50.0, \quad \beta_U = 2.0, \quad \gamma = 4.0.$$

In Figure 5.5 the non-negative equilibria of the uncontrolled equation (5.5) are displayed. For these parameters the origin is an unstable equilibrium (dashed line). In Section 6.1 we will see that stabilization with MPC is not a problem from the optimization point of view and thus does not qualify as a benchmark example for the optimal control algorithms. However, we will observe that the catalytic rod model requires different minimal stabilizing horizons for different initial distributions. This makes the example interesting for the adaptive horizon algorithms in Section 6.4.

5.2 Implementation

In this section we present an overview of the *C++* implementation for the algorithms presented in Chapter 4. In the first part we consider the plain MPC Algorithm. Since our program has a modular object-oriented structure, replacing single components like PDE models or optimization algorithms is an easy task. We begin the section with a flow diagram of the program before we consider the corresponding classes in detail. It should be mentioned that we will not explain each internal method of the abstract class but only those which are relevant for the user. Furthermore, we only

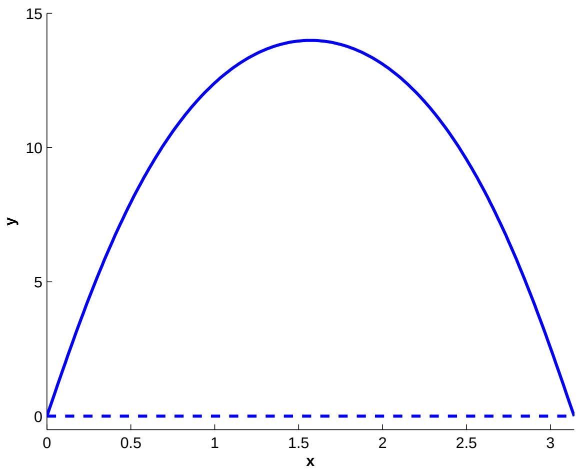


Figure 5.5: Stable (solid line) and unstable equilibrium (dashed line) of the uncontrolled catalytic rod model (5.5).

display fragments of the source code which are meaningful. A complete example can be found in Appendix A.
The implementation of the *MPC-POD* algorithm is presented in Section 5.2.2. Here, the aim is to explain the interaction between the components rather than investigating the source code in detail. We complete the chapter with the presentation of the adaptive horizon MPC algorithms.

5.2.1 Plain MPC Algorithm

The principle structure of the implementation is displayed in Figure 5.6. The abstract classes are written in black while the example subclass is red. On the bottom of the pyramid we have the abstract class Model. It contains information about the problem setting like PDE and cost functional. The class has to provide predictions for state and adjoint equation as well as gradient information and possibly control constraints for the optimal control solver. Since we have to discretize the PDE for the simulation, this class generally requires an external ODE solver for time integration and a method to discretize the spatial variable, e.g. a FEM class. However, in the case of discrete time systems or if the spatial variable is directly discretized by finite differences these classes are not necessary. An example subclass is given by CatalyticRod.
On the next level we have the abstract class Optimize. This is the centerpiece of the implementation: For the current state $y(n)$ it solves the optimal control problem and provides the next state $y(n+1)$ as well as the optimal control sequence $u^\star$. For this purpose it requires the information from the Model class. An example subclass is given by BFGS which is an implementation of Algorithm 4.9.

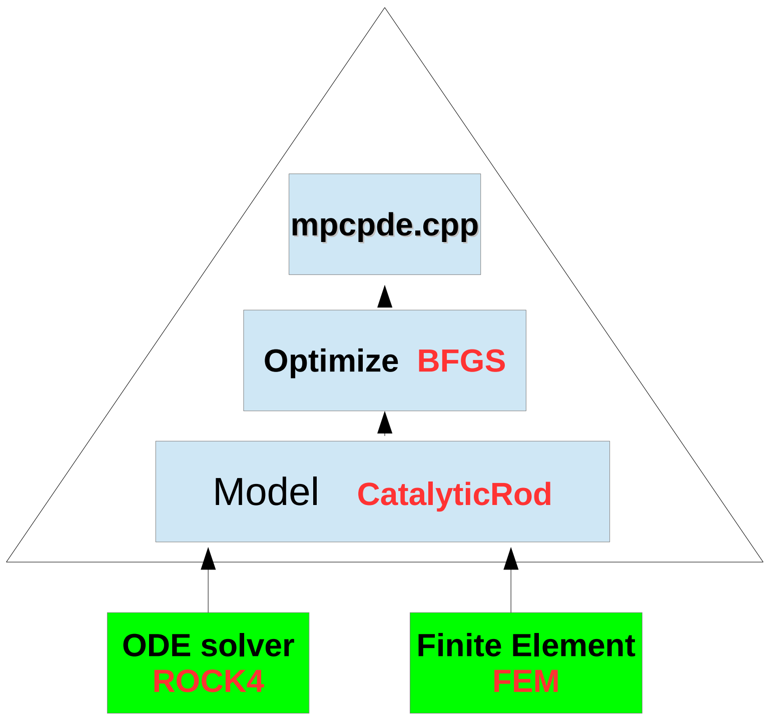


Figure 5.6: Program structure.

On the top of the pyramid we have the *main* program mpcpde.cpp. Here the actual MPC loop takes place. Moreover, the concrete instances of the abstract classes Model and Optimize are created and memory for the variables is allocated.

Class Model

Now, we present the abstract class Model and the concrete realization ReactDiff in detail. As already said the class provides simulations for state and adjoint equation. Furthermore, the optimizer requires information how the gradient looks like. In Table 5.2 the abstract methods of class Model are displayed, i.e., these methods are obligatory for the derived subclass.
The method predictState simulates the state equation with initial state x from initial time t to $t+h$ for a given control u. The solution at time $t+h$ is stored in x. Moreover, the simulation for the adjoint equation is done by the method predictAdjoint. The simulation runs from t to $t+h$ and the solution is again stored in x. Since the adjoint equation requires information about the current state, the solution of the state equation is provided by the variable $rpar$. Note that we introduce a time transformation to make sure that the equation runs forward in time. Therefore, the

given state has to run backwards in time what is taken into account in the internal implementation. The last obligatory method is computeGradient which computes the current gradient by the adjoint p and the control u. The result is stored in *gradient*.

In the next step we want to demonstrate the structure of the presented method

Function	Description	variables
predictState	Simulate state equation	t: Initial time x: Current state u: Current control h: Simulation time
predictAdjoint	Simulate adjoint equation	t: Initial time x: Current adjoint $rpar$: Current state h: Simulation time
computeGradient	Compute the gradient	p: Current adjoint u: Current control $gradient$: Current gradient $lambda$: Regularization parameter N: Horizon

Table 5.2: Methods of the abstract class Model which are obligatory for every derived subclass.

for a concrete example. We investigate the derived subclass ReactDiff which is a prototype class for general reaction diffusion equations. As a concrete example we look at the one dimensional Schlögl model with distributed control (5.2). In order to provide a simulation for the state equation, we have to discretize the PDE. For this easy example we use a finite difference method for the spatial variable. The method stateEquation of the class ReactDiff provides the semidiscretized PDE for the ODE solver. In Listing 5.1 the implementation of the semidiscretization is displayed. The central finite differences are visible in the *for* loop. For the first and the last point the homogeneous Dirichlet condition has to take into account. The method semiFunction computes the nonlinearity of this semilinear PDE. Moreover, we observe that we have a distributed control because u acts in each discretization point independently.

Listing 5.1: Semidiscretized state equation

```
void ReactDiff::stateEquation (int *n, double *t, double *x,
                               double *dx, double *u, int *ipar)
{
    int dim=_dimension_state;
    double hx=LENGTH/double(dim+1);
    dx[0]=(- 2.0* x[0] + x[1])/pow(hx,2)
          +semiFunction(x[0]) + u[0];

    for(int i=1;i<dim-1;i++)
    {

            dx[i]=(x[i-1]-2.0*x[i]+x[i+1])/pow(hx,2)
                  +semiFunction(x[i]) + u[i];
    }
    dx[dim-1]=(x[dim-2] -2.0* x[dim-1] )/pow(hx,2)
              +semiFunction(x[dim-1]) + u[dim-1];
}

double ReactDiff::semiFunction(double x)
{
        f=MU*(x-pow(x,3));
        return f;
}
```

Again, we want to mention that we simplified the example to clarify the structure. In the next step we present the semidiscretization of the adjoint equation. In Listing 5.2 the method adjointEquation is displayed. It can be seen that we use again central finite differences. Furthermore, we have the discrete representation of $f'(y)p$, where $f'(y)$ is computed by the method deriFunction. The last term originates from $J_y = y - y_d$. In our example we look at $y_d \equiv 0$.

Listing 5.2: Semidiscretized adjoint equation

```
void ReactDiff::adjointEquation(int *n, double *t, double *x,
                                double *dx, double *rpar, int *ipar)
{
    int dim=_dimension_state;
    double hx=LENGTH/double(dim+1);

    dx[0]=( -2.0*x[0] + x[1])/pow(hx,2)
          +x[0]*deriFunction(rpar[0])+rpar[0]

    for(int i=1;i<dim-1;i++)
    {

            dx[i]=(x[i-1]-2.0*x[i]+x[i+1])/pow(hx,2)
                  +x[i]*deriFunction(rpar[i])+rpar[i];
    }
    dx[dim-1]=(x[dim-2] -2.0* x[dim-1] )/pow(hx,2)
              +x[dim-1]*deriFunction(rpar[dim-1])+rpar[dim-1];
}

double ReactDiff::deriFunction(double x)
{
        f=MU*(1.-3.*pow(x,2));
        return f;
}
```

We want to point out that the functions described so far are only auxiliary methods for the simulation of the PDEs. The last step is to solve the semidiscretized PDE with an appropriate ODE solver. This is done by the obligatory methods predictState and predictAdjoint, see Listing 5.3. First, the used ODE solver has to be declared in the constructor of ReactDiff (not shown in the listing) before it can be initialized by init(t,x). Afterwards the calculation is done with calc(t+h,u).

Listing 5.3: Obligatory methods predictState and predictAdjoint

```
void ReactDiff::predictState ( double t, double *x,
                               double *u, double h )
{
        _odesolver->init(t,x);
        _odesolver->calc(t+h, u);
}

void ReactDiff::predictAdjoint ( double t, double *x,
                                 double *rpar, double h )
{
        _odesolveadjoint->init(t,x);
        _odesolveadjoint->calc(t+h, rpar);
}
```

The remaining obligatory method is computeGradient, where the current gradient is determined. In our example ReactDiff the gradient is given by $J' = p(x,t) + \lambda u(x,t)$, see Section 2.2. The resulting discretized version is displayed in Listing 5.4. We compute the gradient by control u and adjoint p while the result is stored in *gradient*.

Listing 5.4: Obligatory method computeGradient

```
void ReactDiff::computeGradient ( double *p, double *u,
                double *gradient, double _lambda, int _horizon)
{
    for( int j=0;j<_horizon;j++)
    {
      for( int i=0;i<_ctrl_dimension;i++)
      {
          gradient[j*_ctrl_dimension+i]
          =(p[j*_ctrl_dimension+i]+_lambda*u[j*_ctrl_dimension+i]);
      }
    }
}
```

After the presentation of the obligatory methods we want to pay attention to the control constrained case. Bounds on the control can be incorporated by setting the variables _control_lb and _control_ub to the lower and upper bound. If the bounds are not defined in the derived subclass, the variables have the default values -10^{19} and 10^{19} respectively.
Finally, we want to consider the special case where we use the Newton-CG method as optimization algorithm. In Section 4.1.2 we discussed that the Newton method

requires second order information. This information can also be included in the subclass ReactDiff. By defining $\_linearizedPDE = true$ one can define the methods predictLinear where the linearized state equation is computed and secondDerivative where the second derivative of the semilinear term is given. Otherwise the relevant functions are determined by numerical differentiation.
The constructor of a Model subclass requires the state ($DIMENSION\_X$) and the control dimension ($DIMENSION\_U$). In Listing 5.5 the generation of an instance is displayed.

Listing 5.5: Initialization of a class ReactDiff.

```
Model *model = new ReactDiff(DIMENSION_U, DIMENSION_X);
```

The Model subclasses for the PDE examples presented in Section 5.1 can be found in Table 5.3.

Class	Description
ReactDiff	One dimensional Schlögl model with distributed control (5.2)
SemiLinBC	One dimensional Schlögl model with boundary control (5.3)
MODELFEM	FEM Schlögl model with boundary control (5.4)
CatalyticRod	Catalytic rod model (5.5)

Table 5.3: Example subclasses from the superclass Model.

Class Optimize

In this section we introduce the abstract class Optimize which is the superclass for our optimization algorithms. Again, we only present the *public* methods which are relevant for the user. In Table 5.4 these methods are displayed. The most important function is given by calc(x,u). The input variables are the current state x and an initial guess for the optimal control problem u, e.g., the shifted optimal control sequence from the previous MPC step. After its computation, the state at the next time step is stored in x while u contains the current control sequence. If the methods setTolerance and setMaxIterations are not specified then the algorithm uses the default values $tol = 10^{-6}$ and $maxsteps = 500$. The current values of the stage cost and the objective function are especially of interest for adaptive horizon MPC. The same holds for the method resizeHorizon, where the optimization horizon can be changed in each MPC step.
In Listing 5.6 we see the initialization of a subclass *BFGS*. The description of the required variables is displayed in Table 5.5. If the optimization horizon changes during the MPC algorithm, the variable N has to be set to the maximum horizon.

Listing 5.6: Initialization of the class BFGS.

```
OPTIMIZE *optimize=new BFGS(model, yd, N, lambda,T);
```

Function	Description	variables
calc	Calculation of the optimal control	x: Current state u: Current control
setTolerance	Set prescribed accuracy	tol: Tolerance
setMaxIterations	Set maximum number of iterations	$maxsteps$: Maximum steps
getStageCost	Return stage cost	
getObjective	Return objective function value	
resizeHorizon	Resize optimization horizon	N: New horizon

Table 5.4: Public methods of the abstract class Optimize.

Type	Name	Description
Model	*model*	Example model
double*	y_d	Desired state
int	N	Optimization horizon
double	*lambda*	Regularization parameter
double	T	Sampling time

Table 5.5: Required variables for the constructor of a derived subclass.

In Table 5.6 we see the currently implemented subclasses of the abstract class OPTIMIZE. The details concerning the optimization algorithms can be found in Section 4.1 while the corresponding numerical results are presented in Chapter 6. In Section

Class	Description
PGM	Projected Gradient Method
NCG	Nonlinear Conjugate Gradient Method
BFGS	BFGS Method
BFGSINV	Inverse BFGS Method
BFGSMF	Meshfree BFGS Method
NEWTONCG	Newton-CG method

Table 5.6: Example subclasses from the superclass Optimize.

6.2 we distinguish between the algorithms *BFGSINV I* and *BFGSINV II*. Both are variants of BFGSINV with different initial approximations for the inverse Hessian, see Section 6.2 for details. By using the method setHessian one can choose the unit matrix (*false*) or the approximation from the previous MPC step (*true*). The default value is *true* because this variant turned out to be the faster one.

5.2.2 MPC-POD Algorithm

In this section we give an overview about the implementation of the algorithms introduced in Section 4.2 which combine POD with MPC. The emphasis is on the program flow and the connection between the involved classes. We will not go into implementation details or display the source code. The interested reader can find the concrete example on the enclosed CD-ROM.

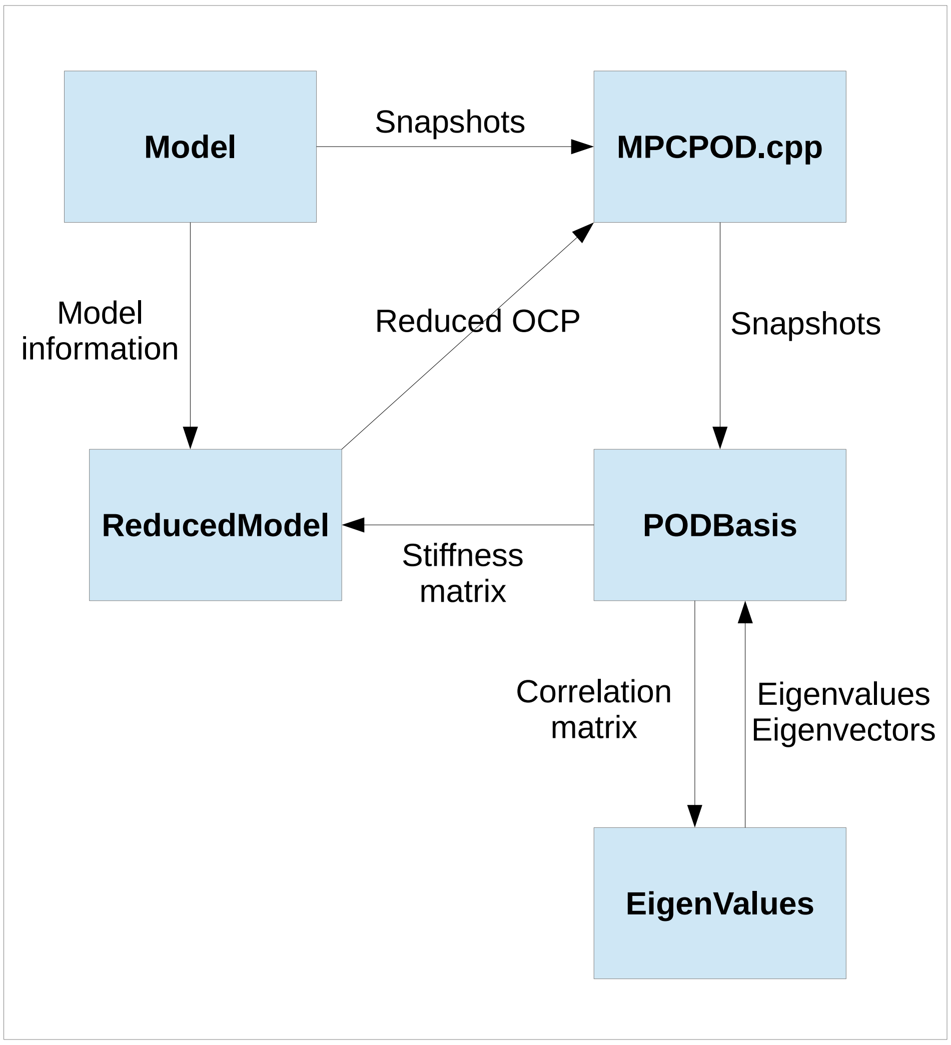


Figure 5.7: Program structure of MPCPOD.

In Figure 5.7 the structure of the *MPCPOD* code is displayed. Each box represents a class and the arrows indicate the information flow between the classes. The file MPCPOD.cpp is the *main* program where the MPC algorithm takes place. Furthermore, the snapshots to determine the reduced model are created here. For this

purpose it requires the simulations of the full model which are provided by the class Model. The abstract class Model is the one we already introduced in Section 5.2.1. Depending on the type of snapshots (see the discussion in Section 2.3.1) it has to provide solutions of the state and adjoint equation for the high dimensional model. Based on the snapshots the class PODBasis builds up the correlation matrix. The corresponding eigenvalues and eigenvectors are computed in the class EigenValues. By using the eigenvalues and eigenvectors the class PODBasis determines the dimension of the reduced model and calculate the reduced stiffness and mass matrices.
The class ReducedModel contains the small dimensional POD model. It requires the corresponding matrices from class PODBasis and model information, e.g. the nonlinearity and the cost functional, from class Model. In our implementation ReducedModel is a subclass of the *yane* model superclass. The MPC library *yane* is well suited for these small dimensional problems. However, the implementation of other ODE optimal control solvers is also possible. Finally, the reduced OCP is provided to *MPCPOD*.
Note that the presented structure only describes the generation of the reduced POD model. The actual implementation of the algorithms presented in Section 4.2 takes place in MPCPOD.cpp.

5.2.3 Adaptive Horizon MPC

In this section we describe the implementation of the adaptive horizon algorithms presented in Section 4.3.1. The focus is on the class AdaptiveHorizon which is a realization of the hierarchical grids Algorithm 4.17. The plain adaptive horizon Algorithm 4.16 also fits into this setting if only the finest spatial grid is investigated. The main task of this class is to compute an appropriate horizon which guarantees a certain quality of the MPC closed loop solution. According to Algorithm 4.17 this step is done for the discretized PDE on a coarse grid. Therefore, the actual optimization of the full problem has to be done afterwards. This step can be omitted if the horizon is calculated from the full model (Algorithm 4.16). In this case the optimal control sequence is already known.

Listing 5.7: Initialization of the class AdaptiveHorizon.

```
AdaptiveHorizon *adaptivehorizon=new AdaptiveHorizon(optimize,
                     maxN, N0, dalpha, usecontrol, resty, restu)
```

In Listing 5.7 we see the initialization of the class AdaptiveHorizon. An explanation of the required input variables can be found in Table 5.7. It should be mentioned that the required optimization algorithm is that for the PDE on the coarse grid. Depending on the used algorithm the maximum horizon $maxN$ and the initial horizon $N0$ play an important role for the performance of the adaptive horizon method. The parameter *dalpha* denotes the prescribed degree of suboptimality. With the parameter *usecontrol* one can decide whether the control on the coarse grid (*true*) or the shifted control from the previous MPC step ($false$) should be used as initial

Type	Name	Description
OPTIMIZE	*optimize*	Optimization algorithm on coarse grid
int	*maxN*	Maximum horizon
int	*N0*	Initial horizon
double	*dalpha*	Desired α
bool	*usecontrol*	Use control
int	*resty*	Quotient of fine and coarse grid (state)
int	*restu*	Quotient of fine and coarse grid (control)

Table 5.7: Required variables for the constructor of class AdaptiveHorizon.

guess for the optimization algorithm. This choice is nontrivial because it depends on the considered example and on the fineness of the coarse grid, see Section 6.4 for details. If the horizon is determined by the full model one should use the option *true*, because in this case the following optimization can be omitted. The remaining variables *resty* and *restu* denote the quotient of the fine and the coarse grid for the state and control variable respectively. These information are required by the internal restriction and prolongation methods.
The actual calculation of the horizon is done by the method calcAdaptive, see Listing 5.8. This method is called in each MPC step before the optimization algorithm starts. The input variables are the current state x, the shifted control from the previous MPC step *Control* and the current horizon N. After execution of the method, N contains the new horizon. If the parameter *usecontrol* is set to *true*, the variable *Control* contains the optimal control on the coarse grid prolongated on the fine grid. Otherwise, it is the same as before. Afterwards, the horizon of the full problem has to be set to the previously computed one. This can be done with the method resizeHorizon(N).

Listing 5.8: Initialization of the class AdaptiveHorizon.

```
adaptivehorizon->calcAdaptive( x, Control, N);
optimize->resizeHorizon(N);
```

Finally, we want to take a note on the internal restriction and prolongation methods. Their task is to convert variables on different meshes. For the case of spatially one dimensional systems the corresponding operators can be found in textbooks about multigrid methods, e.g. [19]. It is important to mention that the conversion matrices are not build up explicitly. For the general case with higher dimensional unstructured domains the construction of restriction and prolongation methods is nontrivial and not yet implemented in our algorithm.

6 Numerical Results

In this chapter we present the main numerical results. In Section 6.1 we start with some prototype closed loop simulations for the example PDEs introduced in Section 5.1. Afterwards in Section 6.2 we investigate the optimization algorithms presented in Section 4.1 in detail. The focus is laid on the suitability for the MPC algorithm and the dependence on parameters like spatial discretization, horizon length, regularization and reaction values. In the ensuing Section 6.3 we analyse the performance of the algorithms in which we combine MPC and POD concepts, see Section 4.2. The advantages and limitations of the adaptive horizon algorithms introduced in Section 4.3 are presented in Section 6.4. Throughout this chapter, all presented results have been computed on a machine *Intel(R) Core(TM)2 Duo CPU E6850 @ 3.00GHz* with 4 Gbyte internal memory. The operating system is *openSUSE 11.1 (x86_64)*.

6.1 Numerical Examples

6.1.1 One dimensional Schlögl equation with distributed control

In this section we present numerical simulations for the one dimensional heat equation with a nonlinearity of Schlögl type and distributed control. On the boundary we impose a homogeneous Dirichlet condition. In detail we look at

$$
\begin{aligned}
y_t(x,t) &= y_{xx}(x,t) + 15(y(x,t) - y(x,t)^3) + u(x,t) && \text{in} \quad (0,1)\times(0,\infty) && \text{(6.1a)}\\
y(0,t) &= y(1,t) = 0 && \text{in} \quad (0,\infty) && \text{(6.1b)}\\
y(x,0) &= 0.2\sin(\pi x) && \text{in} \quad (0,1). && \text{(6.1c)}
\end{aligned}
$$

with stage cost

$$
l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(0,1)} + \frac{\lambda}{2}\|u(\cdot,nT)\|^2_{L^2(0,1)}. \tag{6.2}
$$

In Figure 6.1 we observe that for the uncontrolled equation the equilibrium $y \equiv 0$ is unstable and the solution converges to a different equilibrium. This behaviour is to be expected because $\mu = 15 > \pi^2 = \lambda_1$. Next, we want to stabilize the PDE with the MPC feedback. In the numerical simulation we choose a sampling time $T = 0.025$ and a regularization parameter $\lambda = 0.01$. The spatial discretization is given by $M_x = 100$. We consider $n = 40$ MPC steps and, thus, we have $t \in [0,1]$. The solution of the semidiscretized PDE is obtained by the *ROCK4* ODE solver

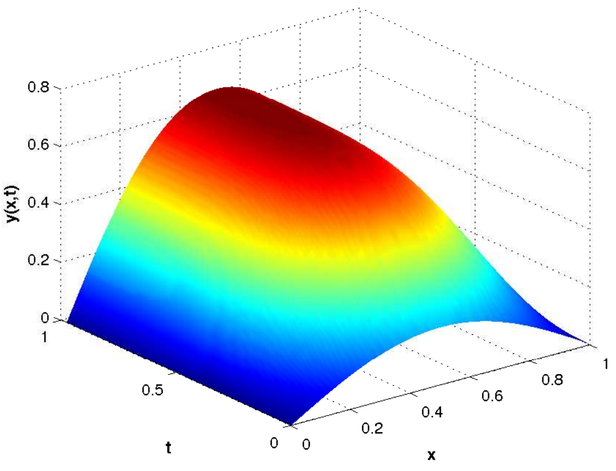
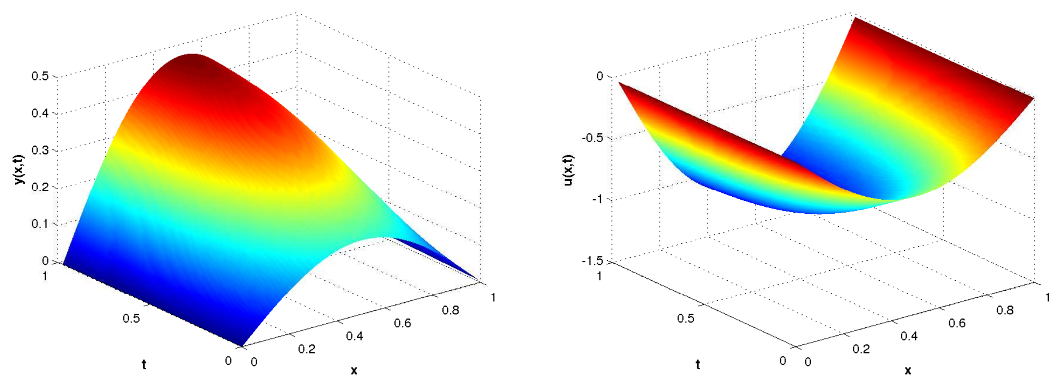
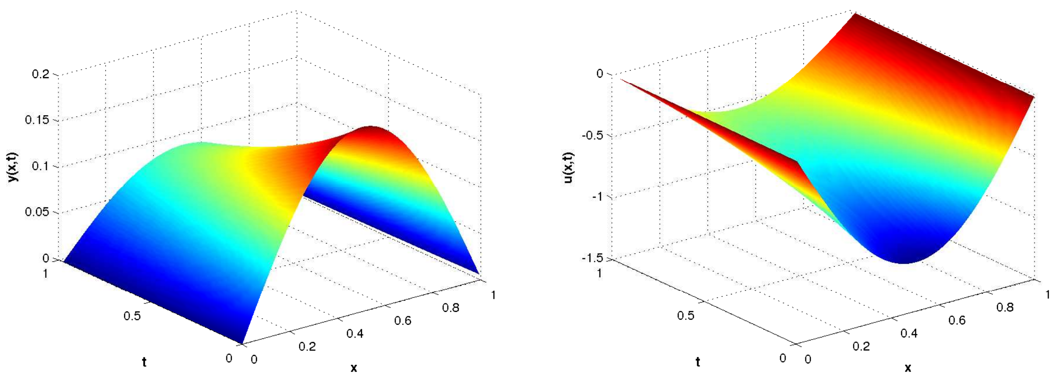


Figure 6.1: Solution trajectory of the uncontrolled one dimensional Schlögl equation

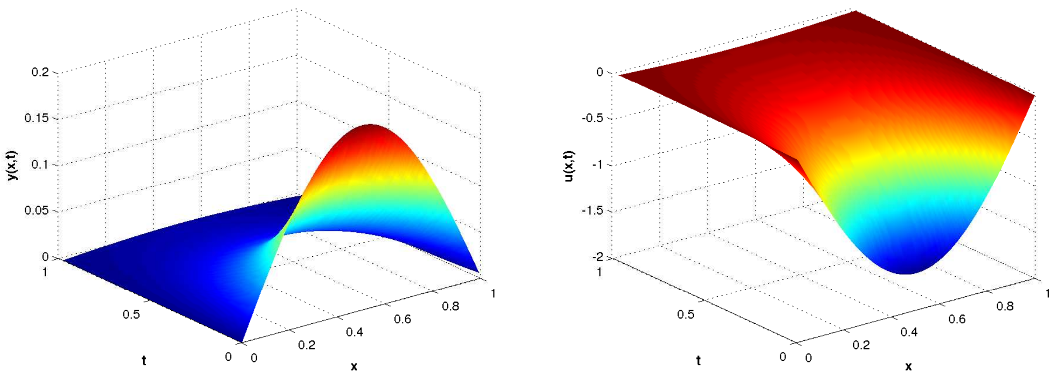
with a tolerance $tol_{ODE} = 10^{-8}$. This high accuracy seems to be exaggerated, because the error caused by the spatial discretization is much higher. However, since discretization errors caused by time discretization are not considered in this thesis, it is desirable to keep them small (see also Remark 1.8).
The solution of the PDE constrained optimization problem is obtained by the *Newton-CG* algorithm presented in Section 4.1 with tolerance $tol_{opt} = 10^{-6}$. In Figure 6.2 the solution trajectory (left) and the corresponding control (right) of the MPC closed loop for different optimization horizons are displayed. First, we observe that the MPC feedback with an optimization horizon $N = 2$ (a) does not yield a stabilizing behaviour. The solution converges to an equilibrium which is different to that of the uncontrolled equation. A similar behaviour is observable for the control: After some steps it remains constant in time. For $N = 3$ (b) we observe that the control is able to stabilize the system. The state as well as the control converge to the origin. The same behaviour is also visible for the optimization horizon $N = 4$ (c). However, the convergence is significantly faster. Furthermore, it can be seen that the control acts stronger in the first MPC steps for larger horizons. Afterwards, it converges faster to zero than for shorter horizons.
In Figure 6.3 we look at the distance (measured in the $L^2(0,1)$-norm) of the state to zero. It can be observed (left) that the solution of the uncontrolled equation and for $N = 2$ converge to different equilibria. Moreover, we see (right) that a longer horizon directly leads to a faster stabilization. In the stabilizing case ($N = 3, 4, 5$) the norm $\|y(\cdot,t)\|_{L^2}$ is strictly monotonically decreasing. This behaviour is strongly related to our theoretical investigations in Section 3.3.2. The distributed control is able to immediately reduce the norm of the state. It is reasonable that the system can be stabilized with $N = 2$, if we choose the regularization parameter λ small enough, i.e., if we do not penalize the control effort. (In the considered example the



(a) $N = 2$



(b) $N = 3$



(c) $N = 4$

Figure 6.2: Solution trajectory (left) and corresponding control (right) of the MPC closed loop for the one dimensional Schlögl equation with distributed control and different optimization horizons

value of λ is too large to ensure stability with a horizon $N = 2$.)

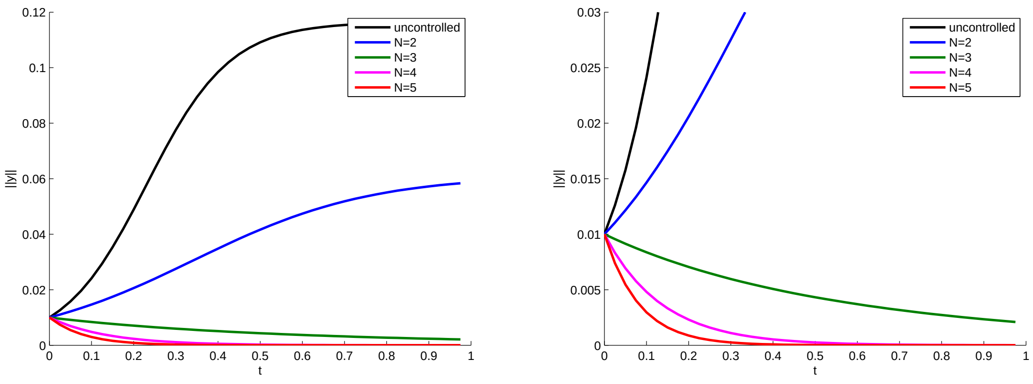


Figure 6.3: Evolution of $\|y(\cdot,t)\|_{L^2}$ for different optimization horizons N

Finally, it is worth mentioning that the computing time for the considered $n = 40$ MPC steps is between 0.12 seconds ($N = 2$) and 0.28 seconds ($N = 5$). Therefore, the average computing time for each single optimal control problem is only between 3ms and 7ms! The optimal control problem in each MPC step was solved with the *Newton-CG* algorithm. These results indicate that real time feedback control with MPC is in principle possible.

6.1.2 One dimensional Schlögl equation with boundary control

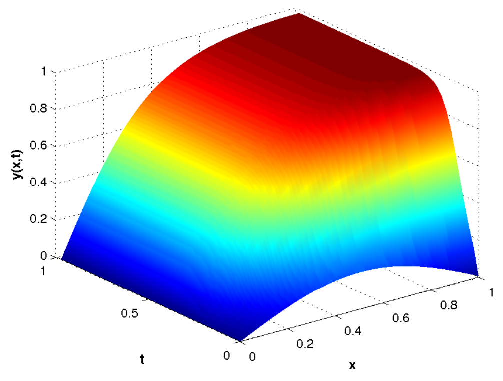
In this section we look at numerical simulations of the boundary controlled one dimensional heat equation with Schlögl nonlinearity. On the left boundary we impose a homogeneous Dirichlet condition and on the right side we have Neumann control. Therefore, we investigate

$$\begin{aligned}
y_t(x,t) &= y_{xx}(x,t) + 15(y(x,t) - y(x,t))^3 && \text{in} \quad (0,1)\times(0,\infty) && (6.3a)\\
y(0,t) &= 0 && \text{in} \quad (0,\infty) && (6.3b)\\
y_x(1,t) &= u(t) && \text{in} \quad (0,\infty) && (6.3c)\\
y(x,0) &= 0.2\sin(\pi x) && \text{in} \quad (0,1). && (6.3d)
\end{aligned}$$

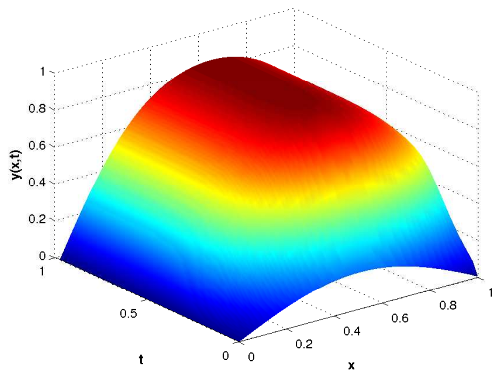
with stage cost

$$l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(\Omega)} + \frac{\lambda}{2}|u(nT)|^2. \tag{6.4}$$

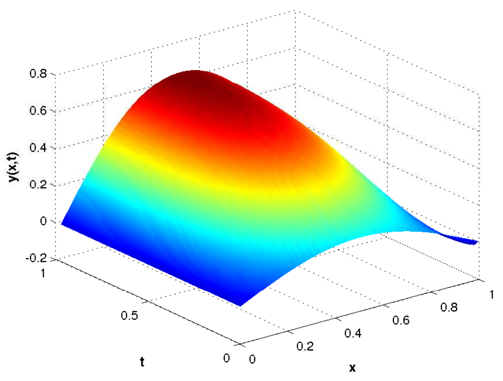
The origin is an unstable equilibrium for the uncontrolled equation, because $\mu = 15 > \pi^2/4 = \lambda_1$. This behaviour can be observed in Figure 6.4 (a). Furthermore, we see that for a horizon of $N = 2$ (b) and $N = 3$ (c) the MPC feedback is not able to stabilize the equation. The solution converges to an equilibrium that is not the desired one. The first horizon where we observe stability is given by $N = 4$ (d).



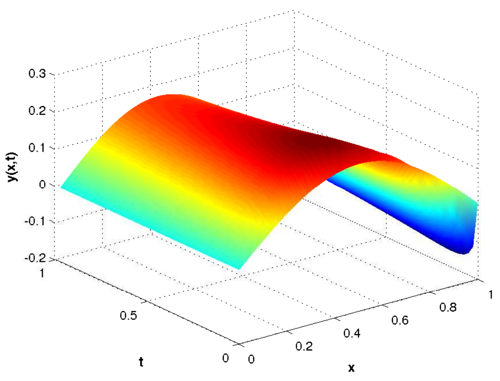
(a) uncontrolled



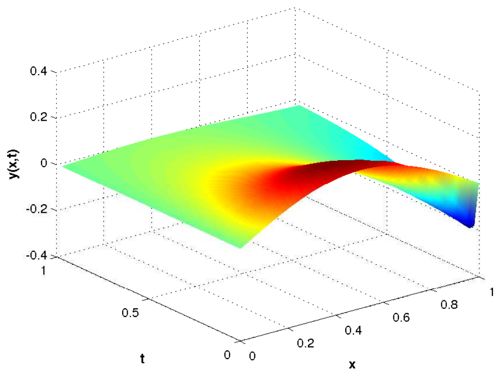
(b) $N = 2$



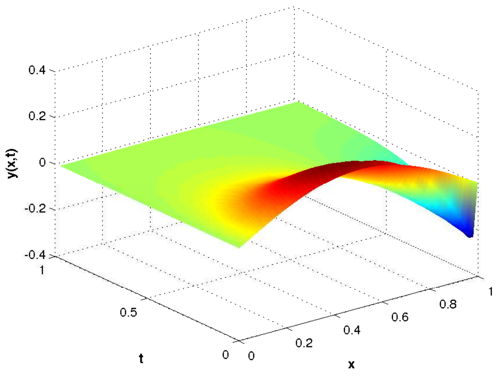
(c) $N = 3$



(d) $N = 4$



(e) $N = 5$



(f) $N = 6$

Figure 6.4: Solution trajectory of the MPC closed loop for the one dimensional Schlögl equation with no control (a) and boundary control with different optimization horizons (b)-(f).

For an increasing horizon we obtain, as expected, a faster convergence to zero (e)-(f).

Similar to the case of distributed control we see that the control acts in a stronger way in the first steps if we enlarge the horizon. However, the value of the control is bounded, even for $N \to \infty$. The stability behaviour is also visible in Figure 6.5 (left) where the norm of the state is displayed. We observe convergence to zero for a horizon $N \geq 4$. A more interesting property can be seen if we look at the stabilizing horizons in detail, see Figure 6.5 (right). The norm of the state is not strictly monotonically decreasing, but it increases in the first steps. This behaviour is reasonable, because for stabilization we need a large value on the right boundary, which temporarily produces a larger norm of the state. This observation also fits to our theoretical investigations in Chapter 3: We cannot expect to obtain stability with a horizon $N = 2$ if the sampling time is too short, because it is possible that the MPC controller does not 'see' that it is beneficial to control. This can occur even without penalizing the control effort, i.e., for $\lambda = 0$.

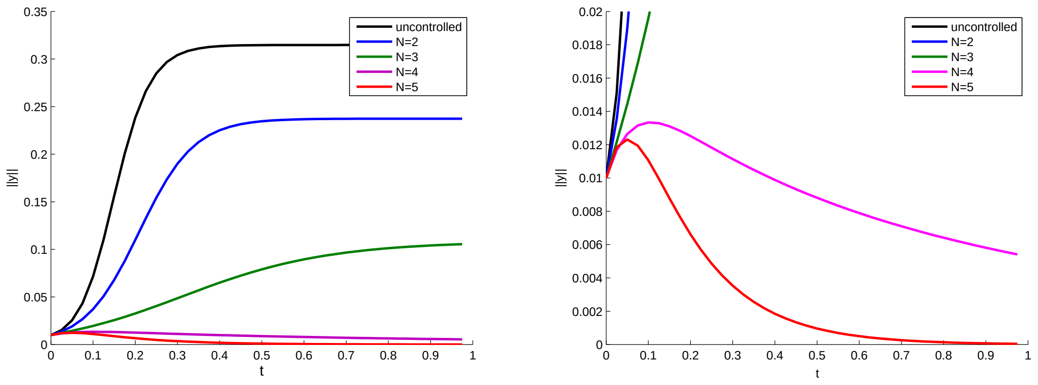


Figure 6.5: Evolution of $\|y(\cdot,t)\|_{L^2}$ for different optimization horizons N.

6.1.3 Two dimensional Schlögl equation with boundary control

In this section we investigate numerical simulations of the two dimensional FEM example introduced in Section 5.1. The nonlinearity is of Schlögl type and we have Neumann control on some parts of the boundary. Therefore, the PDE is given by

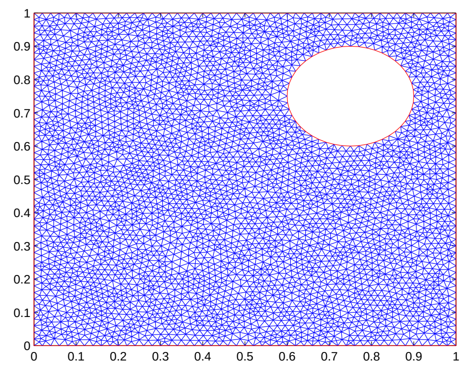
$$\begin{aligned} y_t(x,t) &= \Delta y(x,t) + 8(y(x,t) - y(x,t)^3) && \text{in} \quad \Omega \times (0,\infty) && (6.5a)\\ \partial_\nu y(x,t) &= u(x,t) && \text{on} \quad \Gamma_{N_c} \times (0,\infty) && (6.5b)\\ \partial_\nu y(x,t) &= 0 && \text{on} \quad \Gamma_{N_0} \times (0,\infty) && (6.5c)\\ y(x,0) &= 0.5 && \text{in} \quad \Omega && (6.5d) \end{aligned}$$

The boundary sets Γ_{N_c}, Γ_{N_0} as well as the geometrical details are specified in Section 5.1. The stage costs for this problem are given by

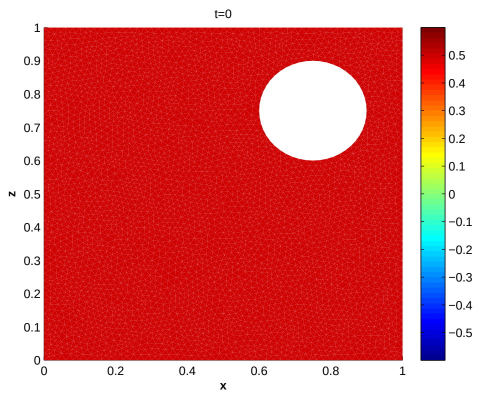
$$l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|_{L^2(\Omega)} + \frac{\lambda}{2}\|u(\cdot,nT)\|_{L^2(\Gamma_{N_c})}. \tag{6.6}$$

As already discussed in Section 5.1 the uncontrolled equation with initial condition $y_0(x) \equiv 0.5$ converges to the stable equilibrium $y_e(x) \equiv 1$. In the MPC closed loop simulation we choose a sampling time $T = 0.01$ and a small regularization parameter $\lambda = 0.001$. The semidiscretized PDEs are again solved by the *ROCK4* ODE solver with a tolerance of $tol_{ODE} = 10^{-8}$. The number of state variables caused by the spatial discretization is given by $M_x = 4098$ (see Section 5.1 for details). For mesh generation and assembling we use the *MATLAB PDE toolbox*.

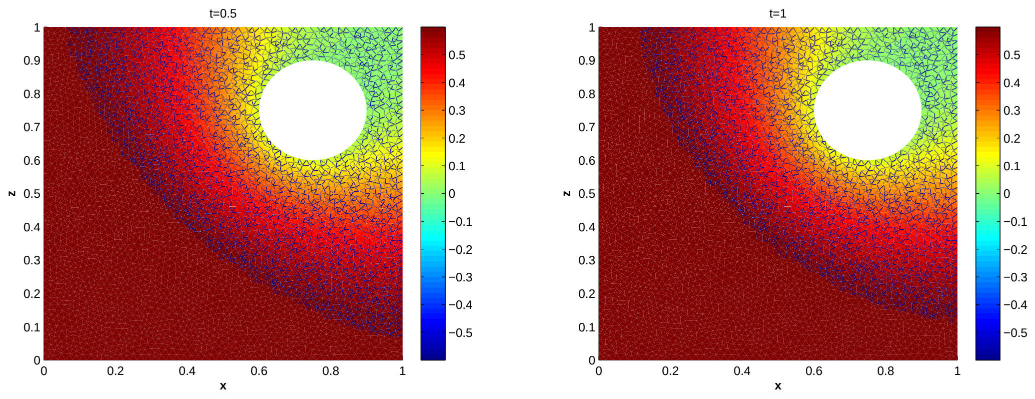
In Figure 6.6 the used Finite Element grid (a) and the initial distribution (b)



(a) FEM mesh



(b) Initial distribution



(c) $N = 2$

Figure 6.6: FEM mesh (a), initial distribution $y_0(x) \equiv 0.5$ (b) and MPC closed loop with horizon $N = 2$ at time snapshots $t = 0.5$ and $t = 1$.

$y_0(x) \equiv 0.5$ are displayed. Moreover, we see the solution trajectory of the MPC controlled PDE with an optimization horizon $N = 2$ at time $t = 0.5$ and $t = 1$. It can be observed that this horizon is too short to stabilize the system and the trajectory converges to a stationary state different from zero. Next, we enlarge the optimization horizon to $N = 10$. In Figure 6.7 we display the solution at different time snapshots. It is clearly visible that the MPC feedback stabilizes the equation to the desired equilibrium $y \equiv 0$.

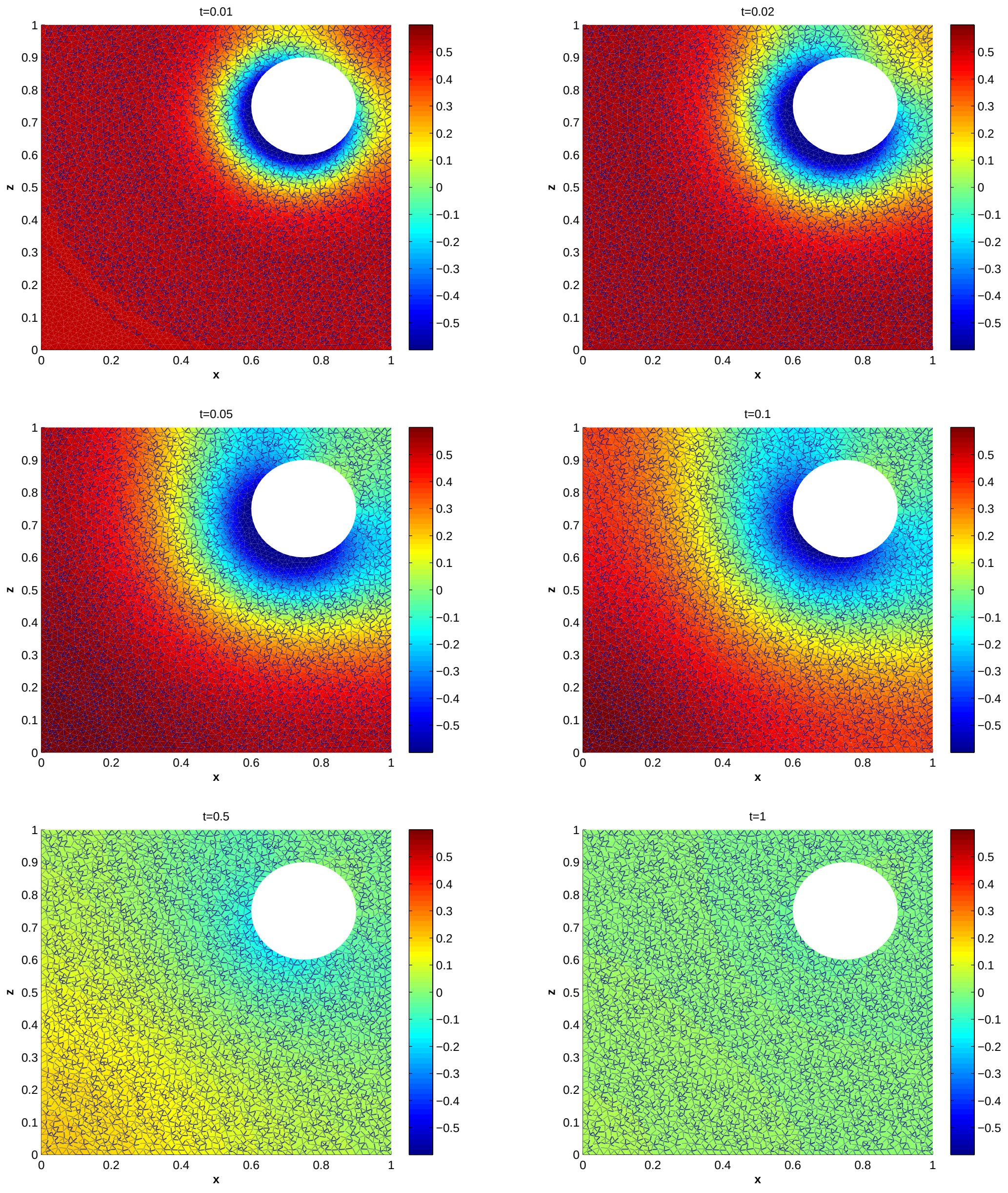


Figure 6.7: Solution trajectory of the Finite Element example with horizon $N = 10$ at different time snapshots.

6.1.4 Catalytic Rod

In this section we look at the numerical simulations for the catalytic rod model (5.5). We consider the case where we have a time dependent control function $u(t)$ that acts in the whole domain (see also the discussion in Section 5.1.2), i.e., the PDE is given by

$$y_t(x,t) = y_{xx}(x,t) + 50(e^{-\frac{4}{1+y(x,t)}} - e^{-4}) + 2(b(x)u(t) - y(x,t)) \tag{6.7a}$$
$$y(0,t) = y(\pi,t) = 0 \tag{6.7b}$$
$$y(x,0) = y_0(x). \tag{6.7c}$$

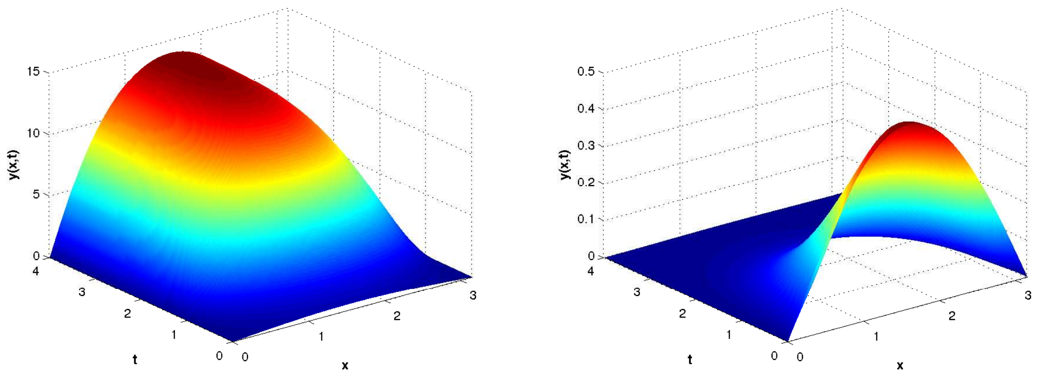
A natural choice for the stage cost is given by

$$l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(0,\pi)} + \frac{\lambda}{2}\|b(x)u(nT)\|^2_{L^2(0,\pi)}.$$

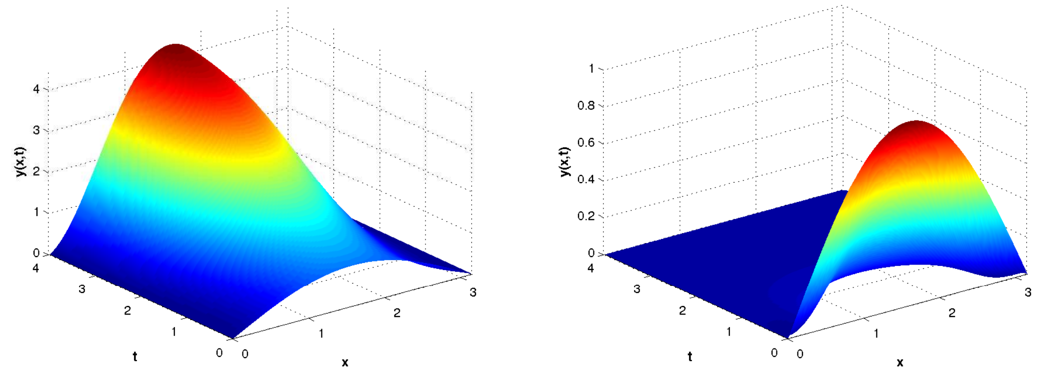
In our simulation we assume the shape function to be constant on the whole interval, i.e., we consider $b(x) \equiv \frac{1}{2}$ for $x \in (0,\pi)$. In this easy case the stage cost simplifies to

$$l(y(n),u(n)) = \frac{1}{2}\|y(\cdot,nT)\|^2_{L^2(0,\pi)} + \frac{\lambda}{2}\frac{\pi}{4}|u(nT)|^2.$$

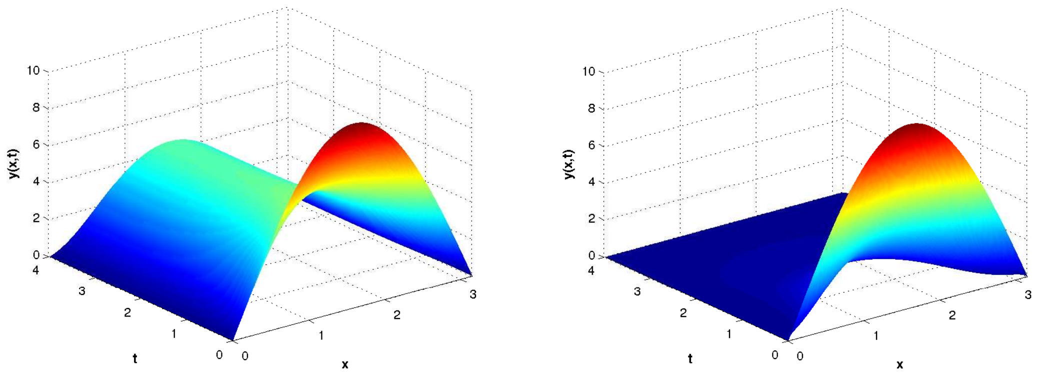
In the MPC closed loop simulation we choose a sampling time $T = 0.01$, regularization parameter $\lambda = 0.01\frac{4}{\pi}$ and a spatial discretization $M_x = 200$. The *ROCK4* ODE solver that we used in the previous examples was not able to solve the semidiscretized PDE with a prescribed tolerance of $tol_{ODE} = 10^{-8}$. Therefore, for this equation we use the fully implicit ODE solver *RADAU5*. In order to investigate the long term behaviour we consider $n = 400$ MPC steps.
As already remarked, the initial function $y_0(x)$ plays an important role for the stabilization of (6.7) with MPC. In the first step we analyse $y_0(x) = 0.5\sin(x)$. In Figure 6.8 (a) (left) we observe that the uncontrolled equation converges to the stable equilibrium with maximum value ≈ 14. Furthermore, it can be seen (right) that an optimization horizon $N = 2$ is sufficient to obtain stability for this initial function. This changes if we modify the initial distribution to $y_0(x) = \sin(x)$, see Figure 6.8 (b). While we observe a fast stabilization to the origin with an optimization horizon $N = 3$ (right), the closed loop trajectory converges to a *steady state* with maximum value ≈ 4.5 for $N = 2$ (left). The 'turning point' of the stability behaviour for $y_0(x) = a\sin(x)$ is given by $a \approx 0.76$, i.e., for $a < 0.76$ we obtain stability with a horizon $N = 2$ while for $a > 0.76$ we need at least $N = 3$. In the last step we look at the initial function $y_0(x) = 10\sin(x)$. For $N = 2$ we observe in Figure 6.8 (c) (left) that the solution converges to the same upper steady state as before (however, in this case from above). On the right figure we see the fast convergence to the desired state $y \equiv 0$ for an optimization horizon $N = 3$.
Since the qualitative behaviour is also of particular interest for the adaptive horizon algorithms in Section 6.4 we shortly summarize the numerical findings: Starting point is again the initial function $y_0(x) = a\sin(x)$. For $a > 0$ the solution converges to the stable equilibrium displayed in Figure 5.5. For $0 < a < 0.76$ the MPC algorithm stabilizes the equation with $N = 2$. For $0.76 < a < 4.5$ (and $N = 2$) the solution converges from below and for $a > 4.5$ from above to the steady state with maximum value ≈ 4.5. With a horizon $N = 3$ we observe stabilization to zero for each investigated positive initial function.



(a) Solution of the uncontrolled catalytic rod model (left) and the MPC closed loop trajectory with $N = 2$ (right) for initial function $y_0(x) = 0.5\sin(x)$.



(b) MPC closed loop trajectory with optimization horizon $N = 2$ (left) and $N = 3$ (right) for initial function $y_0(x) = \sin(x)$.



(c) MPC closed loop trajectory with optimization horizon $N = 2$ (left) and $N = 3$ (right) for initial function $y_0(x) = 10\sin(x)$.

Figure 6.8: MPC closed loop trajectory for the catalytic rod model (6.7) with different initial functions and different optimization horizons.

6.2 Optimization Algorithms

In the previous section we have seen that real time MPC is in principle possible for the example PDEs introduced in Section 5.1. Next, we want to investigate the performance of the different optimization algorithms, presented in Section 4.1, in detail. We focus on the influence of the spatial discretization, horizon length and warm start behaviour. In order to keep the errors caused by initialization (memory allocation et cetera) small, we consider parameter settings where the computational time dominates these effects. Throughout this section we use the following abbreviations: The Projected Gradient Method 4.4 is denoted by *PGM*, the Nonlinear Conjugate Gradient algorithm 4.5 by *NCG* and the BFGS algorithm 4.9 by *BFGS*. Furthermore, we compare two variants of the BFGS method, where the inverse of the Hessian is approximated: In algorithm *BFGSINV I* we use update formula (4.11) with the unit matrix in the initialization step. In the variant *BFGSINV II* the approximation of the inverse Hessian from the previous MPC step is used for the initial step. The matrix free BFGS algorithm 4.10 is denoted by *BFGS MF* and the Newton-CG method 4.8 by *NEW CG*.

Remark 6.1
A further method to solve MPC problems with PDEs is to semidiscretize the PDE and use an existing MPC solver for ODEs, e.g., the yane library. Although this algorithm is quite fast for small systems it is not suitable for semidiscretized PDEs. The numerical simulations indicate that we have a factor between 10 *and* 200 *in the computing time compared to the algorithms in Section 4.1. However, due to the easy implementation (no adjoint and gradient information are required) the library is very helpful for test examples and highly nonlinear PDEs. Furthermore, we use this algorithm in combination with POD, because the number of ODEs is in this case quite small, see Section 6.3.*

Spatial discretization

In the following, we investigate the influence of the spatial discretization for the Schlögl equation with distributed control (5.2) presented in Section 5.1. Since we look at a distributed control problem, the total number of optimization variables is given by $(M_x \cdot N)$ where M_x denotes the number of spatial grid points and N the optimization horizon. Thus, for algorithms which build up the (approximate) Hessian explicitly (*BFGS, BFGSINV I, BFGSINV II*) the number of optimization variables is bounded by $\sqrt{\frac{4\cdot 10^9}{8}} \approx 22360$ due to the limitation of 4 Gbyte internal memory. In the example we use an optimization horizon $N = 10$ with a sampling time $T = 0.025$ and a regularization parameter $\lambda = 0.01$. The stopping criterion is given by $tol_{opt} = 10^{-6}$ first order optimality tolerance. We evaluate the semidiscretized PDE by the *ROCK4* ODE solver with tolerance $tol_{ODE} = 10^{-8}$. The initial function is given by $y_0(x) = \sin(\pi x)$.
The computational time in seconds for the different optimization algorithms and

spatial discretizations is displayed in Table 6.1. We consider $n = 20$ MPC steps, i.e., the presented value is the overall time for the 20 optimal control problems. With a discretization of $M_x = 2000$ and a horizon of $N = 10$ we obtain 20000 control variables and, thus, we are close to the memory limit for the algorithms, where the Hessian is built up explicitly. Of course the optimizer as well as the ODE solver also require additional memory. The stepsize in the optimization method is determined by the inexact line search method presented in Section 4.1.1.

M_x	PGM	NCG	BFGS	BFGS INV I	BFGS INV II	BFGS MF	NEW CG
50	0.36	0.22	0.66	0.47	0.34	0.33	0.2
100	1.32	0.82	2.77	1.90	1.29	1.24	0.67
200	7.83	4.53	12.04	8.52	5.91	6.12	3.36
400	54.45	31.29	60.80	47.74	36.18	38.01	22.16
800	416.52	239.89	354.53	308.60	225.00	269.25	167.35
2000	6432.60	3332.00	4213.40	4563.6	3057.60	3657.30	3373.00

Table 6.1: Computation time in seconds for $n = 20$ MPC steps of the Schlögl equation with distributed control (5.2). The parameters are given by $N = 10, T = 0.025, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$.

First, we observe that all optimizers are able to solve the 20 optimal control problems with $M_x = 50$ (this corresponds to 500 optimization variables) within one second, even for the high accuracy setting of the optimizer and the ODE solver. This statement is still true for the NCG method and the Newton-CG algorithm for $M_x = 100$. It should be mentioned that initialization effects can yield distorted results on the short time scale. Comparing the first order methods, we observe that the *NCG* algorithm delivers significantly better results than the pure gradient method *PGM* for all displayed values of M_x. This observation is consistent with results in the literature. Moreover, we recognize the expected fact that approximating the inverse Hessian (*BFGSINV I+II*) yields better results than approximating the Hessian itself. This procedure avoids the necessity to solve a linear system of equations. It is observable that using the shifted Hessian from the previous MPC step (*BFGSINV II*) instead of the unit matrix (*BFGSINV I*) leads to a beneficial behaviour. Overall we observe that the Newton-CG method (*NEW CG*) yields the best results for most of the considered parameters. However, for not too large values of M_x the nonlinear conjugate gradient method seems to be a serious alternative. It can be seen that *BFGSINV II* yields the best performance for the finest discretization (20000 optimization variables). However, in contrast to *NEW CG* and *NCG* we run into memory problems in the case of a finer grid.
It is visible that the computation time scales quadratically in M_x for moderate values of the spatial discretization. This behaviour is to be expected due to the effort

of matrix vector products. Obviously, for larger values of M_x the computing time grows faster than quadratic. This can be explained by the behaviour of the ODE solver. A fine discretization leads to a very stiff system and the effort significantly grows.

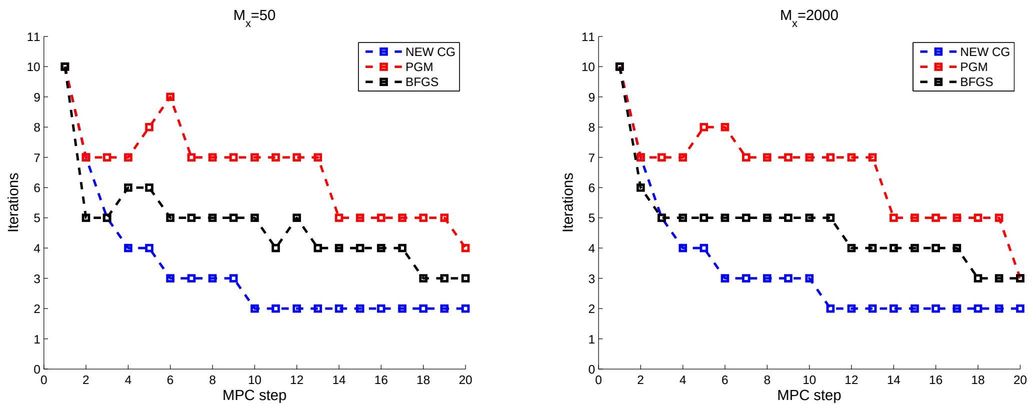


Figure 6.9: Number of iterations in each MPC step for different optimization algorithms with spatial discretization $M_x = 50$ (left) and $M_x = 2000$ (right).

Next, we want to investigate the influence of the spatial discretization on the number of iteration steps of the optimization algorithms. In Figure 6.9 the number of iterations in each MPC step is displayed. For reasons of clarity we restrict ourselves to the algorithms *PGM, BFGS* and *NEW CG*. However, for the other methods we observe similar results. In Figure 6.9 we compare the number of iterations of the coarsest grid $M_x = 50$ (left) with that of the finest grid $M_x = 2000$ (right). It is remarkable that although the number of optimization variables changes from 500 to 20000 the number of iterations remains nearly the same. This observation still holds for the different spatial discretizations and different optimizer. Furthermore, it can be seen that the number of iterations is essentially decreasing. This can be explained by the stability behaviour. The MPC algorithm with horizon $N = 10$ stabilizes the PDE and, thus, the solution converges to the equilibrium where less optimization steps are required. In the following section, in which we analyse the influence of the horizon length, we will look at this problem in more detail. Moreover, one should mention that the first MPC step plays a particular role: In contrast to the further steps no appropriate initial guess is available. Especially for larger horizons this fact becomes visible. The beneficial effect of a good initial guess will be discussed later. One should be careful comparing the algorithms in terms of the iterations. This can easily be seen, e.g., by comparing the Newton-CG method with the projected gradient algorithm: In each step of *NEW CG* a linear system of equations has to be solved, where the computation of Hessian vector products requires additional PDE solutions. In contrast to that the main effort in *PGM* is given by determining an appropriate step length. A better indicator is the number of PDEs which have to be solved.

In Figure 6.10 we display the total number of solved PDEs during the 20 optimal

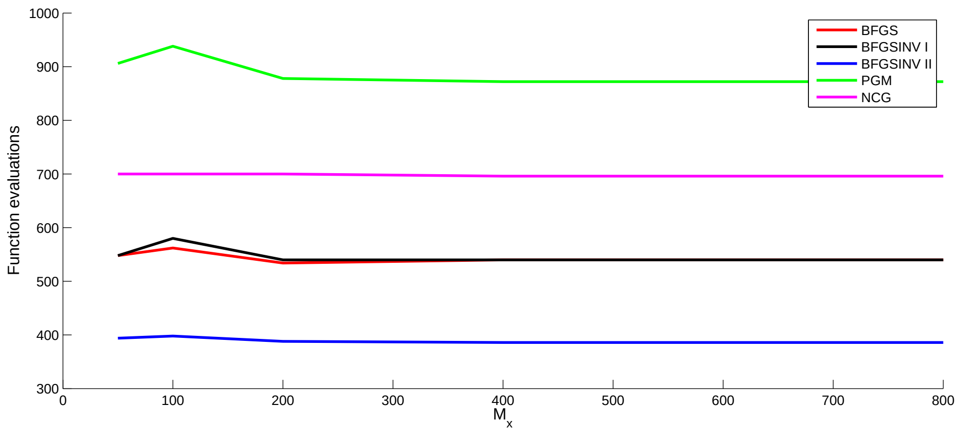


Figure 6.10: Total number of solved PDEs for different optimization algorithms dependent on the spatial discretization. The parameters are given by $n = 20, N = 10, T = 0.025, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$

control problems. It is remarkable that the spatial discretization plays no important role independently of the used algorithm. The relatively small values for the different BFGS algorithms should not be overestimated: The main effort of these algorithms lies in the computation of dyadic products and matrix vector evaluations. (In addition in *BFGS* a linear system of equations has to be solved.)

Next, we look at the finite element example presented in Section 5.1. Thus, we investigate a PDE with Neumann boundary control on different finite element grids. The nonlinearity of Schlögl type is given by $f(y) = 5(y - y^3)$. In Table 6.2 the computation time for $n = 20$ MPC steps is displayed. We set the parameters to $N = 10, T = 0.01$ and $\lambda = 0.001$. Note that this small regularization parameter generally leads to more iteration steps and, thus, to a longer computing time. As before, we choose $tol_{opt} = 10^{-6}$ and $tol_{ODE} = 10^{-8}$ as tolerance for the optimizer and ODE solver respectively. The number of grid points is denoted by M_x. Since we use linear finite elements, the number M_x coincides with the degrees of freedom. Note that the number of points where the control acts is much smaller than the total number of grid points due to the boundary control. The number of control variables and grid information corresponding to the values of M_x can be found in Section 5.1.

Since the number of control variables grows only moderately compared to the total number of grid points, it is more difficult to separate the different influences on the computing time. However, it is reasonable that the effort for solving one PDE

M_x	PGM	NCG	BFGS	BFGS INV I	BFGS INV II	BFGS MF	NEW CG
183	5.74	2.55	2.51	2.52	1.25	2.48	2.18
652	28.70	10.12	10.97	10.93	5.26	11.05	8.73
4089	919.53	299.85	356.51	353.40	158.84	363.14	321.51

Table 6.2: Computation time in seconds for $n = 20$ MPC steps of the finite element example (5.4) with boundary control. The parameters are given by $N = 5, T = 0.01, \lambda = 0.001$ and $tol_{opt} = 10^{-6}$.

grows quickly. In contrast to that the computational burden within the optimizer increases not that much because the control space is comparatively small. Therefore, it seems to be desirable to keep the number of solved PDEs small. In Figure 6.10 we have already seen that the BFGS based algorithms are well suited concerning this property. In contrast to the distributed case the computation of the dyadic and scalar products plays no important role. This explains why these algorithms perform well for this boundary control problem. Furthermore, it is observable that for these parameters the computing time for *BFGS* and *BFGSINV I* is essentially the same. Since the numbers of iterations are quite similar, one explanation for the observed behaviour is that the computation of the additional dyadic products for *BFGSINV I* is not much faster than solving the comparatively small linear system of equations in *BFGS*. As we will see in the next section this behaviour changes for larger optimization horizons.

In the last example we look at the one dimensional heat equation with Schlögl nonlinearity $f(y) = 8(y - y^3)$. The parameters for the simulation are given by $N = 10, T = 0.025, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$. In Table 6.3 we display the computing time for $n = 20$ MPC steps. Similar to the previous examples we observe

M_x	PGM	NCG	BFGS	BFGS INV I	BFGS INV II	BFGS MF	NEW CG
50	1.97	0.83	0.81	0.76	0.35	0.71	0.42
100	11.17	2.53	2.46	2.24	0.96	2.38	1.20
200	21.17	7.27	7.23	6.38	3.17	6.29	3.74
400	100.57	40.48	34.89	31.00	15.92	31.01	20.47

Table 6.3: Computation time in seconds for $n = 20$ MPC steps of the boundary controlled Schlögl equation (5.3). The parameters are given by $N = 10, T = 0.025, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$.

that the algorithms *BFGSINV II* and *NEW CG* perform very well. Moreover, it can be seen that the algorithm *NCG* is not as competitive as in the case of distributed control. This can be explained by the high number of required PDE solutions (see Figure 6.10). Again the worst result can be observed for the *PGM* algorithm. It is remarkable that, although the number of control variables is much smaller for boundary control, the computing time is not so different to the distributed controlled case.

Horizon Length

In this section we investigate the influence of the horizon length on the computing time in detail. Obviously, a larger optimization horizon directly leads to more control variables and one expects that the computational time increases. However, beside the number of optimization variables, we have to take two additional effects into account: As already mentioned in the previous section, a longer horizon often yields a faster convergence to the equilibrium. For most of the considered algorithms the number of optimization steps decreases if the current state is close to the desired one. Moreover, we will see that for some algorithms the first MPC step (where no initial guess is available) is quite sensitive with respect to the horizon.
Again we consider the distributed controlled Schlögl equation with parameters $T = 0.025, \lambda = 0.01, \mu = 15$ and $M_x = 200$. In Table 6.4 the computational time for $n = 20$ MPC steps with varying optimization horizon is displayed.

N-1	PGM	NCG	BFGS	BFGS INV I	BFGS INV II	BFGS MF	NEW CG
2	0.75	0.67	0.96	0.90	0.72	0.69	0.78
4	2.12	1.51	3.49	2.97	1.87	2.09	1.51
8	6.10	3.30	10.18	7.37	4.03	4.90	2.49
16	23.43	8.29	34.72	22.64	13.47	17.88	4.31

Table 6.4: Computation time in seconds for $n = 20$ MPC steps of the distributed controlled Schlögl equation. The parameters are given by $M_x = 200, T = 0.025, \lambda = 0.01, \mu = 15$ and $tol_{opt} = 10^{-6}$.

Obviously, the behaviour of the algorithms with regard to the horizon length is quite different: While the computing time for the *NEW CG* algorithm does not even grow linearly we observe a nearly quadratic increase in the *PGM* algorithm. For the further algorithms a superlinear growth is visible. To investigate the convergence properties we especially look at the number of iterations in the first MPC step. For the horizon $N = 3$ the *PGM* algorithm requires 4 iterations and the other methods 3 iterations. By enlarging the optimization horizon to $N = 17$ the number of iterations is 77 for *PGM*, 20 for the BFGS algorithms (in the first step it is the same for these methods) and 12 for $NEWCG$. This observation explains the sensitivity of the *PGM* algorithm with respect to the optimization horizon.

Furthermore, we look at the property that a larger horizon leads to a faster stabilization. In Figure 6.11 we display the number of iterations in each MPC step for the algorithms *NEW CG* (left) and *BFGS* (right). For both methods we observe that a larger horizon requires more iterations at the beginning. This can be explained by the higher number of optimization variables. Moreover, it is visible that for larger horizons the decrease of iterations is quite fast, while for shorter horizons the number remains nearly constant. In the Newton method for instance, the number of

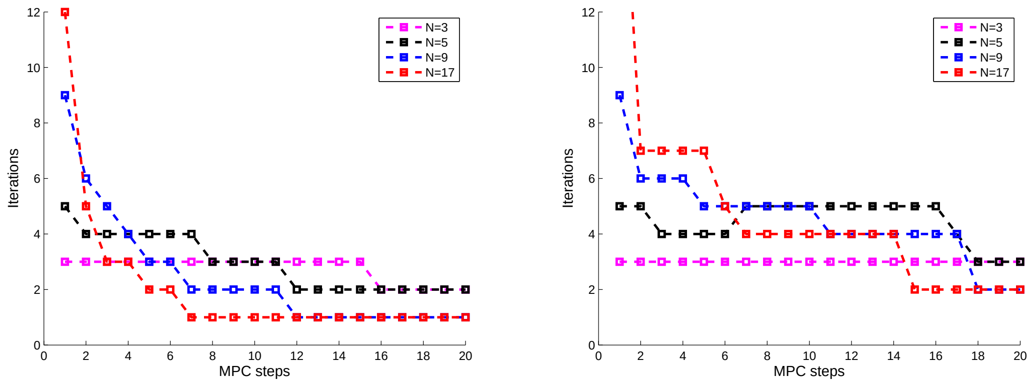


Figure 6.11: Number of optimization steps in each MPC step with different horizons for the algorithms *NEW CG* (left) and *BFGS* (right).

iterations for $N = 17$ is below or equal that for the smaller horizons after two MPC steps. A similar behaviour can be observed for the other algorithms. However, as we see for the *BFGS* algorithm (Figure 6.11 (right)), it is possible that this property becomes visible later. Of course, this beneficial behaviour in terms of the iterations is only observable if N is large enough to stabilize the system. It should be mentioned that this effect strongly depends on the used algorithm and the considered PDE.

Warm start behaviour and parameter dependence

In this section we investigate the influence of an appropriate initial guess on the optimization algorithm. Moreover, we consider the effect of different reaction values μ and different regularization parameters λ. In contrast to our earlier investigations, the number of control variables is not affected by these parameters.

As already discussed, one advantage of the MPC algorithm is the knowledge of a good initial guess for the optimal control problem. Only for the first MPC step and in the case of instantaneous control ($N = 2$) no information is present. (One possibility to obtain an initial guess for these cases is the application of a known (not optimal) feedback, cf. [77].)

In the numerical simulation we compare the case in which the initial guess is obtained from the previous MPC step with that in which we initialize the optimizer with the zero sequence. This means for the first method that we use $u_{initial} = (u_1^*, u_2^*, \dots, u_{N-1}^*, 0)$, where $u^* = (u_0^*, u_1^*, \dots, u_{N-1}^*)$ is the optimal control sequence from the previous MPC step. An alternative choice for the last control is given by a simple copy of u_{N-1}^*. More elaborate techniques to obtain a guess for the last control are suggested in [77]. (However, these methods require the solution of additional control problems.) In the second method we use $u_{initial} = (0, \dots, 0)$. Since there is no appropriate initial guess available for the first MPC step, we begin our consideration with the second step. This ensures that in the first method all optimization problems have an appropriate initial guess. We again consider the example of the Schlögl model (6.1) with parameters $N = 10, \mu = 15, T = 0.025, M_x = 200, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$. The computing time for $n = 20$ MPC steps is displayed in Table

	PGM	NCG	BFGS	BFGS INV I	BFGS INV II	BFGS MF	NEW CG
ws	6.73	3.41	12.21	8.38	4.48	4.81	2.73
zero	12.71	5.53	28.17	15.18	9.58	14.48	4.45

Table 6.5: Computation time in seconds for $n = 20$ MPC steps of the Schlögl model (6.1). The algorithm which uses the MPC initial guess is denoted by *ws*, while zero is the method with zero initialization. The parameters are given by $N = 10, T = 0.025, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$.

6.5. It can be seen that the warm start method essentially halves the computation time for most algorithms. For the meshfree BFGS method this effect is even stronger. This can be explained by the fact that a good initial guess significantly reduces the number of iterations (see Figure 6.12). However, the *BFGS MF* algorithm only performs well if the number of optimization steps is comparatively small due to the recursive formulas (4.12) and (4.13).

In the next step we investigate the influence of the regularization parameter λ on the optimal control problem. In Chapter 3 we have seen that reducing the value of λ generally leads to smaller stabilizing horizons and faster stabilization. In contrast to this it is well known, cf. [1], that a small value of λ requires more optimization steps and numerical errors can occur. The following investigations show that these contradicting effects are actually observable. We consider the example of the Schlögl model (6.1) with parameters $N = 8, T = 0.025, M_x = 200, \mu = 15$ and $tol_{opt} = 10^{-6}$. In Table 6.6 the computing time for $n = 20$ MPC steps with varying regularization parameter is displayed.

It is remarkable that the computing time for most of the algorithms is not monotonically in λ. This is caused by similar reasons we already discussed in the investigation of the horizon length: In Figure 6.13 (left) the L^2-norm of the state for the different

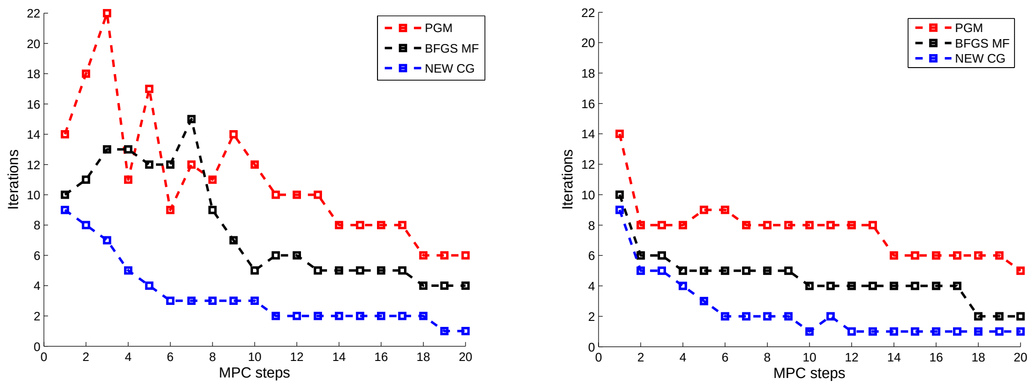


Figure 6.12: Number of optimization steps in each MPC step for different optimization algorithms with zero initialization (left) and with initial guess from the previous MPC step (right).

λ	PGM	NCG	BFGS	BFGS INV I	BFGS INV II	BFGS MF	NEW CG
0.001	0.67	0.65	12.21	3.74	3.21	6.88	0.99
0.01	5.14	2.93	8.67	6.57	3.80	4.39	2.72
0.1	2.25	2.22	6.01	4.78	2.66	2.67	2.31

Table 6.6: Computation time in seconds for $n = 20$ MPC steps of the Schlögl model (6.1). The parameters are given by $N = 10, T = 0.025, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$

values of λ is presented. Obviously, for $\lambda = 0.001$ we obtain a faster stabilization than for $\lambda = 0.01$. If the penalization of the control effort is too strong ($\lambda = 0.1$), the MPC algorithm does not stabilize the system. The stability behaviour is directly visible in the number of iterations (see Figure 6.13 (right)). For smaller values of the regularization parameter the number of iterations in the first MPC step increases dramatically. This observation is consistent with results in the literature, cf. [1]. However, after a few steps the beneficial effect of the faster stabilization is visible and the number of iterations becomes small. Since this property can compensate the higher effort for the first steps, for most of the algorithms the required computing time increases from $\lambda = 0.001$ to $\lambda = 0.01$. An exception is given by the *BFGS MF* algorithm. This behaviour can again be explained by the bad performance of this algorithm if the number of iterations is comparatively large. The advantage that the iteration number is smaller after a few steps cannot compensate the computational burden for the first step. In the unstable case $\lambda = 0.1$ we observe once more that a large value of λ yields few iterations for the optimization algorithm and, therefore, a small computation time in the first MPC step. Since the unstable equation converges to an equilibrium (though not the desired one) the number of iterations remains constant during the simulation.

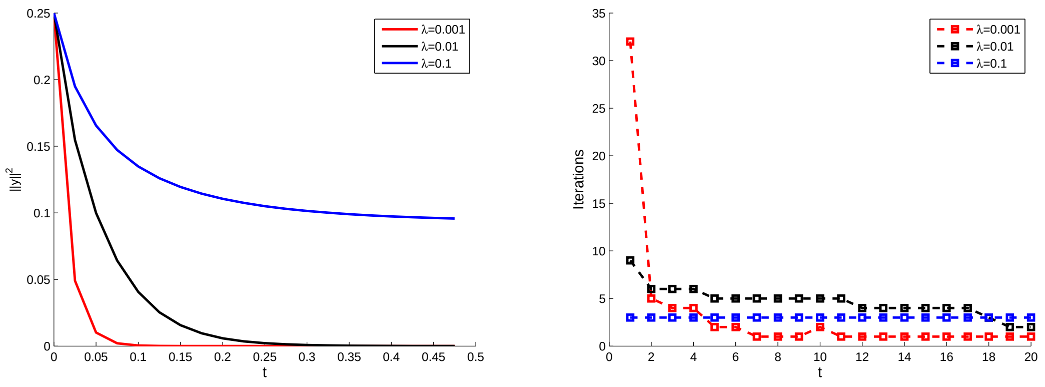


Figure 6.13: Evolution of $\|y(\cdot,t)\|_{L^2}^2$ for different values of λ (left). Number of optimization steps in each MPC step in dependence of the regularization parameter λ (right).

Finally, we investigate the influence of the reaction parameter μ on the optimization algorithms. Note that this parameter comes from the physical model and is not a discretization parameter (such as the spatial discretization M_x) or a design parameter (such as the horizon N or regularization λ) that can be chosen by the user. From the theory in Chapter 3 we know that a larger value of μ requires a larger control effort for compensation. Furthermore, the required horizon to obtain stability increases. In order to investigate the influence of the reaction parameter we look at the example of the one dimensional Schlögl equation with distributed control (5.2). The parameters are given by $N = 10, T = 0.025, M_x = 200, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$.

μ	PGM	NCG	BFGS	BFGS	BFGS	BFGS	NEW CG
				INV I	INV II	MF	
11	5.91	3.07	10.32	7.66	3.61	4.23	2.08
16	6.70	3.46	11.66	8.52	4.43	5.11	2.76
21	7.02	3.49	11.67	4.78	4.88	5.66	3.12

Table 6.7: Computation time in seconds for $n = 20$ MPC steps of the Schlögl equation. The parameters are given by $N = 10, T = 0.025, \lambda = 0.01$ and $tol_{opt} = 10^{-6}$.

In Table 6.7 we display the computing time for $n = 20$ MPC steps with varying reaction parameter μ. It is visible that the computing time is increasing for growing parameter μ. This effect could be caused by several reasons: It is possible that the time for solving a single PDE increases, while the number of iterations in the optimization algorithm remains constant. This behaviour, for instance, was observable in Section 6.2, in which we investigated the influence of the spatial discretization. Furthermore, it is to be expected that the number of optimization steps increases because the stability behaviour changes with varying reaction parameter. In Figure 6.14 it is visible that the latter is actually true. We display the number of iterations for the *BFGS* algorithm in each MPC step in dependence of μ. It can be seen that especially in the first steps the number of iterations is higher for a larger reaction parameter. This behaviour can also be observed for the other optimization algorithms.

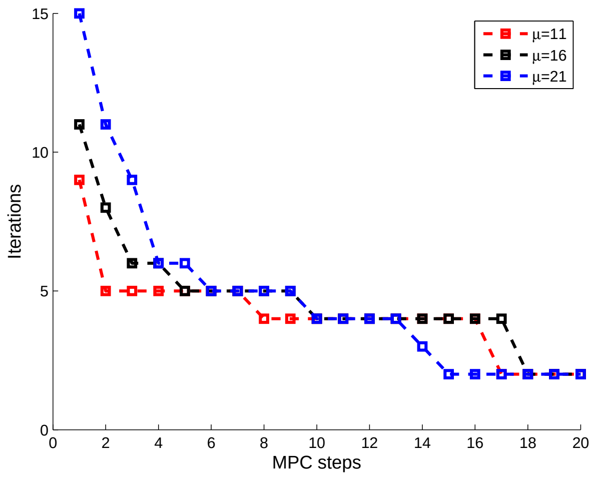


Figure 6.14: Number of optimization steps in each MPC step for different reaction parameters μ.

6.3 MPC-POD Algorithms

Next, we investigate the algorithms which combine MPC and POD methods, presented in Section 4.2. We want to point out that Algorithm 4.13 and 4.14 can be interpreted as a variant of Algorithm 4.15. The crucial parameter in Algorithm 4.15 is the local model error e. If this value exceeds a prescribed bound ε, the reduced POD model is updated. By setting $\varepsilon = \infty$ we end up in Algorithm 4.13 where the reduced model is built up once before the MPC algorithm starts. In the following we denote this method as *POD I*. For $\varepsilon = 0$ the POD model is updated in each MPC step. This is exactly the procedure in Algorithm 4.14 (denoted by *POD III*). These two variants are compared with Algorithm 4.15 for $\varepsilon = 0.01$ (denoted by *POD II*).

This means that the local model error in each MPC step is below 1%.

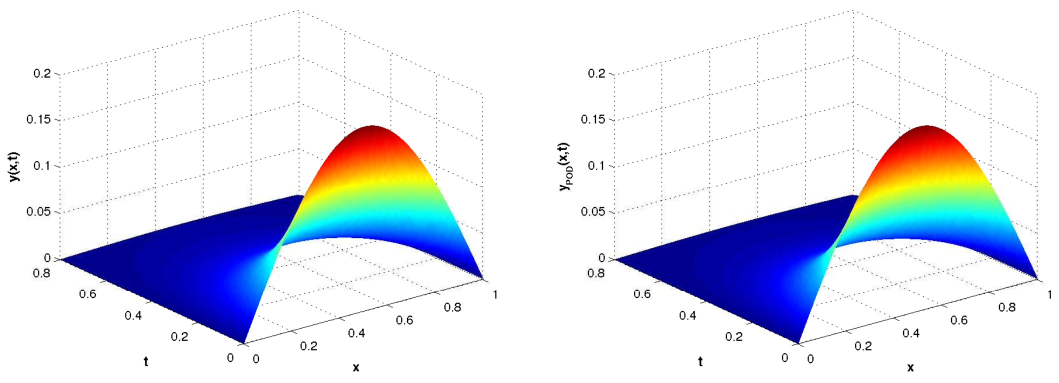
Remark 6.2
Once again, we want to point out that the model error describes the quality of the reduced model. Especially, it is not possible to make a statement about the accuracy of the closed loop trajectory with POD control. In order to estimate this error we would need information about the error between the POD control and the control obtained from the full model, which are generally not available.

In the following we compare the described variants of Algorithm 4.15. In the first MPC step, where no initial guess is available, we create the snapshots by using the initial control $u = 0$. This choice is motivated by the numerical results in [3]. The snapshots in the further steps are created by using the control from the previous MPC step. Our numerical results indicate that this method yields better POD models than using the zero initialization in each step. A discussion concerning this topic can be found in Section 4.2. In the snapshot ensemble we include state information as well as information about the adjoint equation. It turns out that this method yields better results than only using the snapshots of the state. The same observation can be found in [27]. Similar improvements can be observed by including the time derivative of the state instead of the adjoint, cf. [63]. The reduced POD model is built up as described in Section 2.3.1. In order to decide how many POD modes N_{POD} to incorporate in the reduced model we use the frequently used criterion

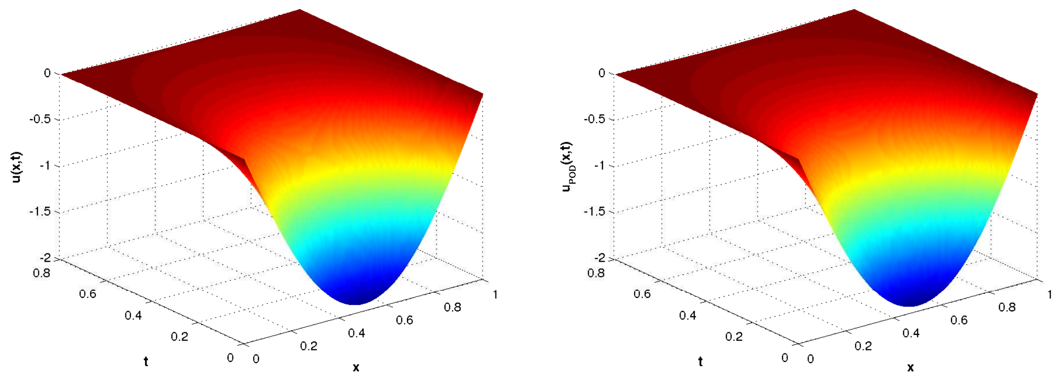
$$\mathcal{E}(N_{POD}) = \frac{\sum_{k=1}^{N_{POD}} \lambda_k}{\sum_{k=1}^{l} \lambda_k} \geq 0.999 \tag{6.8}$$

where l denotes the dimension of the snapshot ensemble. This means that the POD basis contains at least 99.9% of the information of the ensemble.
The optimal control problem of the reduced model is solved by the MPC library *yane*, which is quite fast and well suited for these low dimensional systems. The tolerance for the underlying optimization algorithm, *NLPQLP* by K. Schittkowski, cf. [82], is given by $tol_{opt} = 10^{-8}$.

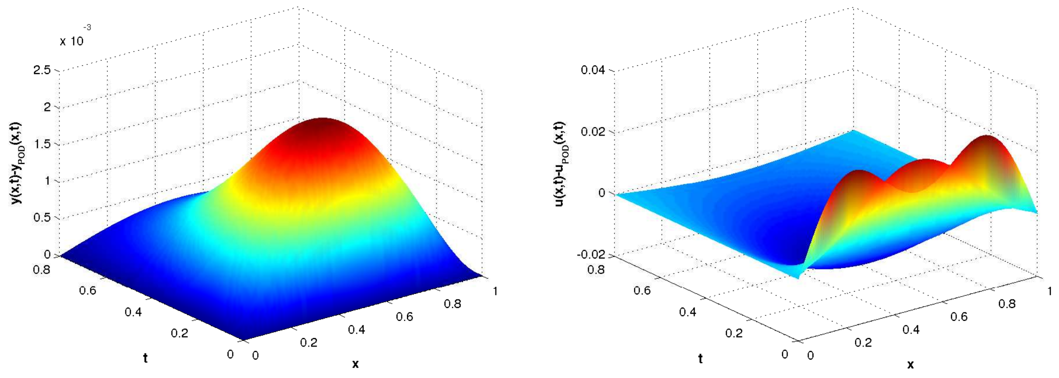
We look at the one dimensional Schlögl model (5.2) with distributed control. The parameters for our numerical closed loop simulation are given by $\mu = 15, \lambda = 0.01, T = 0.01$ and $N = 10$. The semidiscretized full model is solved with the *ROCK4* ODE solver with tolerance $tol_{ODE} = 10^{-8}$ and the low dimensional POD model with the explicit *DoPri853* ODE solver with tolerance $tol_{ODE} = 10^{-8}$. We choose 10 snapshots of the state and 10 snapshots of the adjoint equation. Since we consider a horizon of $N = 10$ the snapshots coincide with the sampling grid. In our examples a higher number of snapshots (and thus a finer resolution of the grid) did not significantly improve the results. (For the presented example we also considered simulations with up to 200 snapshots.) Note that the choice of a good snapshot ensemble is a nontrivial issue. Some results concerning the optimal snapshot location can be found in [64].



(a) Solution trajectory with control computed by the full model (left) and by the reduced POD model (right)



(b) Control function computed by the full model (left) and by the reduced model (right)



(c) Differences between the reduced and the full model for state (left) and control (right)

Figure 6.15: Accuracy of the reduced model compared to the full system.

In the first simulation we will show that the combination of MPC and POD yields good results. The initial distribution is given by $y_0(x) = 0.2\sin(\pi x)$. Later we will see that $y_0(x)$ plays an important role for the performance of the algorithm. In

Figure 6.15 (a) the solution trajectory is displayed with the control computed by the full model (left) and the control obtained by the POD model (right). We want to point out that $y_{POD}(x,t)$ is not the state of the reduced model but the state of the full system with the control computed by POD. Obviously, both solutions are quite similar. Furthermore, we observe the same behaviour by comparing the control (see Figure 6.15 (b)). It is visible that the control determined by the reduced model (right) is a very good approximation of the true control (left). Finally, in Figure 6.15 (c) we display the differences $y(x,t) - y_{POD}(x,t)$ (left) and $u(x,t) - u_{POD}(x,t)$ (right). It is visible that the maximum difference in the state is approximately $2 \cdot 10^{-3}$. The maximum error for the control is ≈ 0.02. This leads to a relative error between the true control and the control of the reduced model of about 1%. Thus, the model reduction technique yields quite good results for this example.
After this motivation, we want to investigate the presented example in more detail. In addition to our three POD algorithms we also look at the *NEW CG* method, which is the fastest of the 'classical' optimization algorithms presented in Section 4.1 for this example. In our numerical simulation we consider $n = 20$ MPC steps. First we want to analyse how often the POD model has to build up for the different algorithms. In Table 6.8 we display the number of 'model updates' in dependence of the prescribed model error ε. The corresponding values for $\varepsilon = \infty$ and $\varepsilon = 0$ are obvious: In algorithm *POD I* we build up the POD model only once before the first MPC step. The opposite procedure, where we update the model in each MPC step, is given in algorithm *POD III*. Thus, we have up= 20. The interesting value is up= 2 for algorithm *POD II*. This means that it is sufficient to build up the reduced model only two times to ensure a local model error below 1%.
This is not surprising if we take a look at the dimension of the reduced POD models: Each POD model is a one (!) dimensional system. If we take into account that the initial distribution $y_0(x)$ is the first eigenfunction of the linearized PDE, this behaviour is reasonable. In Figure 6.15 (a) it can be seen that in each time step the profile is similar to the initial function with a different height. The second model update directly occurs after the first MPC step. This is also reasonable because at this moment we can simulate the snapshots with an appropriate control.

	POD I	POD II	POD III
ε	∞	0.01	0
up	1	2	20

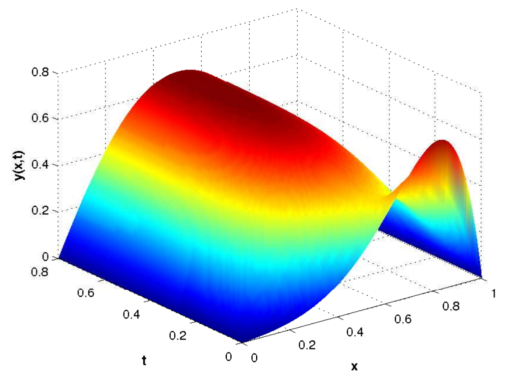
Table 6.8: Number of model updates for the different POD algorithms.

In Table 6.9 the computing time for the three POD algorithms and the *NEW CG* method for $n = 20$ MPC steps is displayed. The spatial discretization varies between $M_x = 50$ and $M_x = 800$. The first observation is that for a coarse spatial discretization the *NEW CG* algorithm is much faster than the POD methods. This is reasonable because in this case one simulation is cheap and the effort for build-

M_x	POD I	POD II	POD III	NEW CG
50	0.42	0.45	0.58	0.09
100	0.61	0.63	0.89	0.24
200	0.93	0.99	1.51	1.16
400	1.66	1.78	3.23	6.88
800	3.59	4.05	9.97	48.73

Table 6.9: Computation time in seconds for $n = 20$ MPC steps of the one dimensional Schlögl equation. The parameters are given by $N = 10, T = 0.01, \lambda = 0.01$ and $\mu = 15$.

ing the POD system (snapshot generation, eigenvalue computation, creation of the reduced model) is needless. However, at the latest when $M_x = 400$ all POD algorithms outperform the *NEW CG* method. For a spatial discretization of $M_x = 800$ we have a speed up between 4.9 (*POD III*) and 13.6 (*POD I*). We want to point out that the presented computing time for *POD I* includes the effort for building up the one reduced model. If we take the view that this step can be done offline before the MPC algorithm starts, algorithm *POD I* becomes even faster. In Figure 6.15 we have already observed that the POD approximation is quite good. Furthermore, it can be seen that *POD I* is the fastest of the POD algorithms whereas *POD III* is the slowest one. By taking into account that each model update requires computing time, this result seems to be obvious. However, as we will see in the next example, this is not true in general.



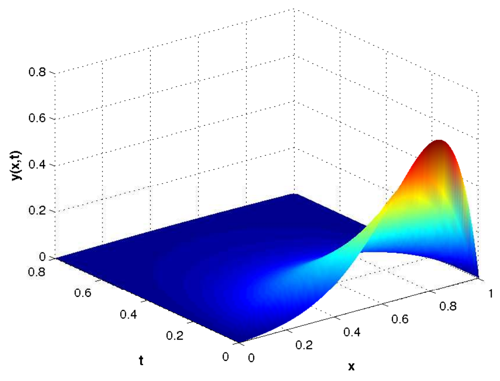


Figure 6.16: Solution trajectory of the uncontrolled equation (left) and with MPC control (right) for the initial distribution $y_0(x) = 0.02 \sin(\pi x) \exp(5x)$.

For the next example we use the same parameters as before but we change the initial distribution to $y_0(x) = 0.02 \sin(\pi x) \exp(5x)$. The solution trajectory for the uncontrolled equation can be found in Figure 6.16 (left). It is clearly visible that the unsymmetrical initial function is first diffused to the centre before it converges to the stable equilibrium. Therefore, it is reasonable that more POD modes are required in order to describe the dynamics. The MPC closed loop system is presented in Figure 6.16 (right).

In Table 6.10 we display the computing time for $n = 20$ MPC steps. First, we see

M_x	POD I	POD II	POD III	NEW CG
50	0.69	0.71	0.95	0.10
100	1.24	1.03	1.49	0.34
200	2.21	1.88	2.82	1.46
400	4.02	3.58	5.41	8.68
800	8.81	8.34	14.26	62.43

Table 6.10: Computation time in seconds for $n = 20$ MPC steps of the one dimensional Schlögl equation. The parameters are given by $N = 10, T = 0.01, \lambda = 0.01$ and $\mu = 15$.

that we obtain similar results as before: For small values of the spatial discretization the *NEW CG* algorithm yields significantly better results than the POD methods. This behaviour changes visibly for finer discretizations (e.g. $M_x = 400$), where solving a PDE requires much more time. In the simulation we observe up= 4 model updates for algorithm *POD II*. (For *POD I* and *POD III* we have again up= 1 and up= 20 respectively.) Furthermore, it is visible that (except for $M_x = 50$) *POD II* is the fastest of the POD algorithms. At the first glance this is surprising, because in *POD II* we have more model updates than in *POD I*. In order to explain this observation we have to take a closer look at the dimension of the reduced model. In Figure 6.17 the dimension of the reduced model in each MPC step is displayed. In the first step we see that 4 POD modes are needed to obtain an appropriate approximation of the dynamics. Since *POD I* (red line) build up the reduced model only once, we have a four dimensional system in each MPC step. For Algorithm *POD III* (blue line), where the model is rebuilt in each step (indicated by the squares), we observe that the number of required POD modes decreases. This is reasonable because for larger t the shape does not change significantly. Finally, we consider Algorithm *POD III* (black line). The four MPC steps where the reduced model is built are marked by a black circle. It is visible that we have three model updates at the beginning where the dynamic strongly changes and one after ten MPC steps. This observation explains why *POD II* yields the best performance. Algorithm *POD I* only requires one update but for all MPC steps we have to solve a system of dimension 4. In the opposite case of *POD III* we have (except the first step) less POD modes but a model update (including time consuming simulations) in each step. Algorithm *POD II* yields a good tradeoff between the number of updates and the dimension of the system for this example. We want to point out

that the performance of the different algorithms strongly depends on the effort for one simulation and on the system dynamics. If the solution for a PDE is expensive one should keep the number of updates small. In this case it is reasonable to use the algorithms *POD I* and *POD II*. For a system where the dynamic strongly changes it is useful (and often necessary for keeping the model error small) to regularly rebuild the reduced model. Finally, we compare the error of the state trajectory for *POD II* and *POD III*. In Figure 6.18 we display the difference between $y(x,t)$ (where the control is obtained from the full system) and $y_{POD}(x,t)$ where the control is computed by *POD II* (left) and *POD III* (right). Although we only have 4 model updates in Algorithm *POD II*, the results are quite similar. The presented examples show that the combination of model predictive control and reduction techniques can be very efficient. We observe a visible speed up compared to the fastest optimizer from Section 4.1, whereas the computed control is quite similar.
As already discussed in Section 2.3.1 the nonlinearity of the reduced system still requires operations in the high dimensional space. This problem can be avoided by using the Discrete Empirical Interpolation Method (DEIM), which is a method that builds new basis functions upon the nonlinear term. It is to be expected that the use of DEIM will accelerate the optimization of the reduced system, cf. [3].

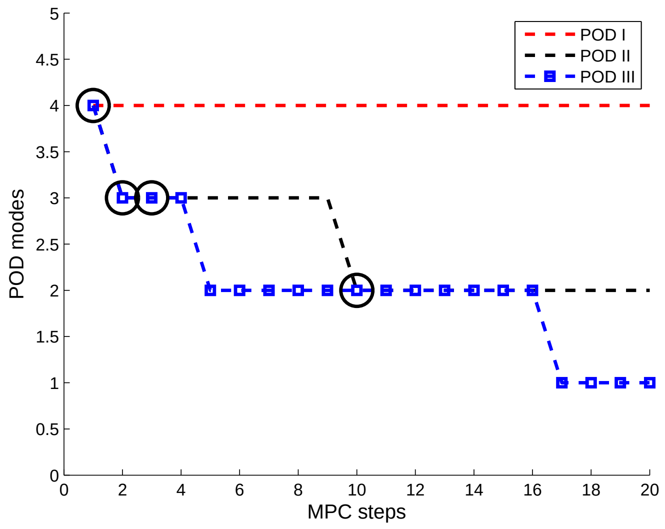


Figure 6.17: Dimension of the reduced POD model in each MPC step. The black circles and the blue squares indicate the model updates for *POD II* and *POD III* respectively.

Remark 6.3
In the presented examples we have seen that only very few POD modes are required to observe good results. This is a well known phenomenon for the heat equation and

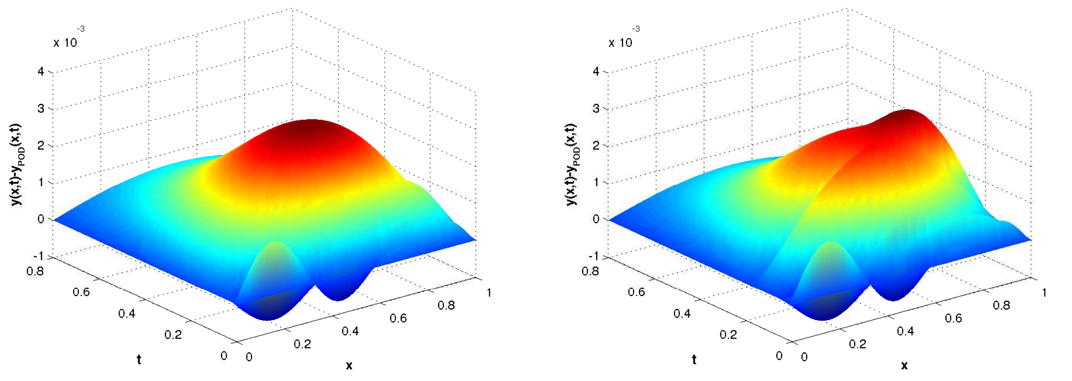


Figure 6.18: Difference of the closed loop trajectory between the control computed using the full model and the control determined by POD algorithm *POD III* (left) and *POD II* (right).

can be explained by the fast decay of the eigenvalues. This changes drastically if an additional convection term come into play. For these equations more elaborate techniques are required, cf. [2]. The same problems occur for hyperbolic systems like the wave equation.
Furthermore, we want to remark that the speed up due to POD strongly depends on the number of POD modes that are necessary to obtain an appropriate model. In [20], for instance, the authors consider a travelling wave model where 15 *POD basis functions are required. It is demonstrated that the computing time for solving the* 15 *dimensional reduced system with dense matrix is much higher than for the* 300 *dimensional Finite Element space with a sparse structure.*
Finally, it should be mentioned that the theoretical background of POD is not yet fully understood, see also the remarks in Section 2.3.1.

6.4 Adaptive Horizon Algorithms

In this section we investigate the adaptive horizon algorithms presented in Section 4.3.1. We start with the example of the catalytic rod (6.7) in order to demonstrate that adaptive horizon MPC can be a useful method to reduce the computational burden. Afterwards we consider the algorithms in more detail. We close the section by demonstrating the disadvantages of adaptive horizon MPC by means of the boundary controlled Schlögl equation (6.3).
In Section 6.1.4 we already discussed the dependence of the minimal stabilizing horizon on the initial function $y_0(x) = a\sin(x)$. With regard to adaptive horizon MPC the numerical simulations indicate the following behaviour for $a > 4.5$: With $N = 2$ we can reduce the distance to zero until we are close to the upper steady state, see Figure 6.8 (c) (left). Afterwards, we use the horizon $N = 3$ to bring the

state close to the origin from where we can stabilize the system with $N = 2$, see Figure 6.8 (a) (right). In our numerical example we will see that this behaviour can be indeed observed. The parameters of the MPC algorithm are the same we already used in Section 6.1.4 and the initial function is given by $y_0(x) = 10\sin(x)$. Since we are essentially interested in stabilization we choose a small value $\bar{\alpha} = 0.05 > 0$ as the desired suboptimality degree. Furthermore, we only reduce the horizon if the condition $\alpha > \rho\bar{\alpha}$ holds, where α is the local suboptimality degree in the current step, see Algorithm 4.16. In our numerical simulation we choose $\rho = 3$ which seems to be a good tradeoff between oscillations and conservatism of the horizon.
In Figure 6.19 the MPC closed loop trajectory is displayed for $N = 3$ (left) and for the adaptive horizon algorithm (right). It is visible that both methods stabilize the problem. The adaptive horizon algorithm has an average horizon of $N = 2.54$ for the $n = 300$ MPC steps. However, we see that there is no big difference in the performance of the algorithms. Again, we want to point out that we do not observe stability with a horizon $N = 2$, see Figure 6.8 (c) (left). The time for computing the 300 MPC steps is given by $31.04s$ for $N = 3$ and by $20.59s$ for the adaptive horizon algorithm. (The details can be found in Table 6.11.) This convincing result shows that adaptive horizon MPC can be a useful tool in the context of PDEs.
In the next step we want to investigate the behaviour of adaptive horizon MPC for

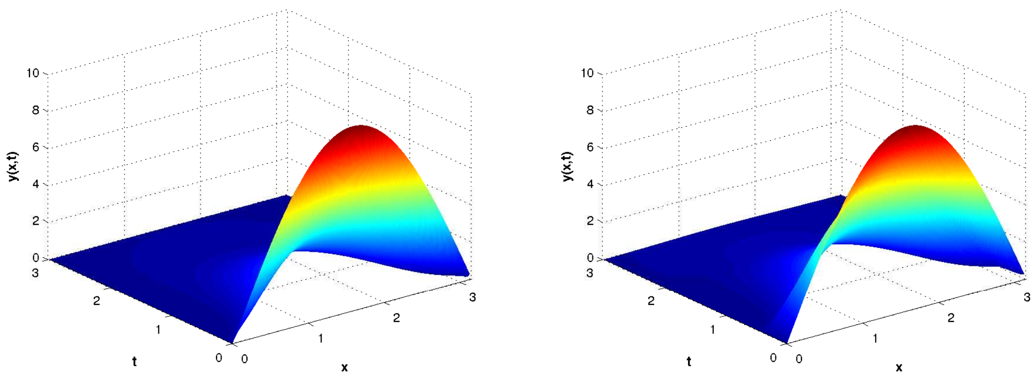


Figure 6.19: MPC closed loop trajectory for the catalytic rod model (6.7) with optimization horizon $N = 3$ (left) and with the adaptive horizon Algorithm 4.17 (right).

this example in more detail. In Figure 6.20 (left) we see the applied horizon in each MPC step. Obviously, we obtain exactly the expected behaviour: At the beginning a horizon of $N = 2$ guarantees a sufficient large decrease of the cost functional. At $t = 0.4$ we enlarge the horizon to $N = 3$ to overcome the controlled steady state. Finally, at time $t = 2.02$ we can reduce the horizon to $N = 2$. It can be seen, that the horizon rarely changes and except for $t = 0.32$ no oscillations are observable. This is due to the fact that we choose a large value $\rho = 3$. (Actually, for $\rho = 2$ the algorithm is even a bit faster, but we obtain much more oscillations.) In Figure 6.20

(right) we see the time evolution of the maximum value of $y(\cdot,t)$. The influence of the horizon $N = 3$ between $t = 0.4$ and $t = 2.02$ is clearly visible.

Now, we want to investigate this example in more detail for different parameter

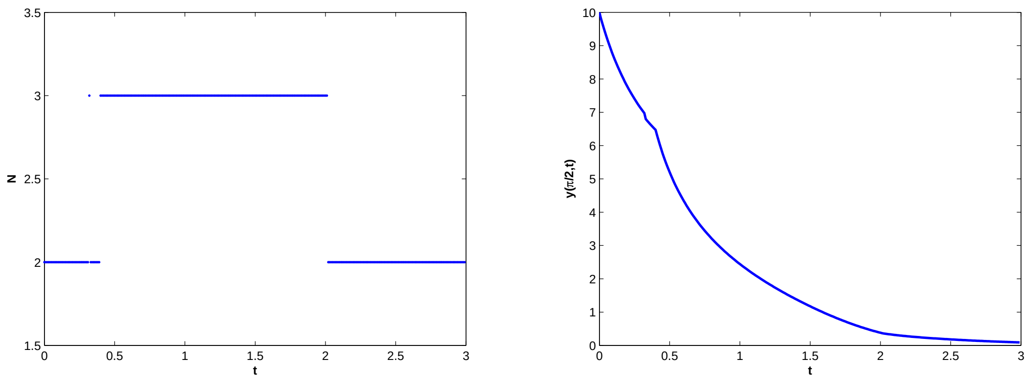


Figure 6.20: Horizon that is applied in each MPC step (left) and time evolution of the maximum value of $y(\cdot,t)$ (right).

settings of the adaptive horizon algorithm. In the MPC closed loop simulation we choose a sampling time $T = 0.01$, regularization parameter $\lambda = 0.01\frac{4}{\pi}$ and $M_x = 200$ as the spatial discretization of the full model. The semidiscretized PDE is solved by the *RADAU5* ODE solver with tolerance $tol_{ODE} = 10^{-10}$. For solving the optimal control problem we use Algorithm *BFGSINV II* with a tolerance $tol_{opt} = 10^{-6}$. Motivated by the numerical results we choose a maximum horizon $N_{max} = 5$ and an initial horizon $N_0 = 2$ for the adaptive horizon algorithm. In Table 6.11 we display the computation time for $n = 300$ MPC steps. The value M_h denotes the discretization of the coarse grid. According to our implementation in Section 5.2.3 the parameter $usecontrol$ determines whether the shifted control from the previous MPC step is used ($false$) or the interpolation of the coarse grid control ($true$). With t_1 we denote the computing time for determining an appropriate horizon. The time for solving the optimal control problem on the fine grid is given by t_2. Thus, the overall time of the algorithm is $t_1 + t_2$. In order to interpret the results one should have in mind that an MPC controller with the stabilizing horizon $N = 3$ requires $31.04s$. The MPC algorithm with a horizon $N = 2$ needed $20.02s$. However, for this horizon we do not observe stability.
In the first step we consider the case where we compute the horizon with the full model, i.e., we have $M_h = M_x = 200$. This corresponds to Algorithm 4.16. We observe a computing time for determining the horizon of $t_1 = 49.12s$. Since the optimal control sequence for the full model is already computed, we obtain $t_2 = 0$. However, we see that the effort for this algorithm is much higher than using the plain MPC method with the horizon $N = 3$. Although the catalytic rod model is well suited for adaptive horizon methods, Algorithm 4.16 is not useful for this problem. This observation also holds for the other considered examples. The additional

computational burden is always higher than the benefit of the shorter horizon.
In the next step we double the mesh size, i.e., we consider $M_h = 100$. It can

	usecontrol = *false*			*usecontrol* = *true*			
M_h	t_1	t_2	$t_1 + t_2$	t_1	t_2	$t_1 + t_2$	$\bar{N}$
200	49.12	0	49.12	49.12	0	49.12	2.57
100	12.98	26.38	39.36	12.90	20.35	33.25	2.57
50	3.33	26.37	29.70	3.15	21.44	24.59	2.56
25	0.83	26.09	26.92	0.61	19.98	20.59	2.56

Table 6.11: Computation time for the adaptive horizon Algorithm 4.17, where M_h denotes the dimension of the coarse grid, t_1 the computing time to determine the horizon and t_2 the time for solving the optimal control problem. The comparative values are $t = 31.04s$ for $N = 3$ and $t = 20.02s$ for $N = 2$.

be seen that we can determine the appropriate horizon in $t_1 = 12.98s$ while the actual optimization requires $t_2 = 26.38s$. However, the overall computing time with $t_1 + t_2 = 39.36s$ is higher than the plain MPC algorithm. This changes for $M_h = 50$ where we observe a total computing time of $t_1 + t_2 = 29.70s$. It is visible that the time t_1 scales quadratically with the mesh size. In view of our results from Section 6.2 this behaviour is reasonable. Since the algorithm only provides the horizon, it is clear that the optimization time t_2 nearly remains constant. We can interpret t_2 as the 'pure' optimization time of the adaptive horizon algorithm, i.e., we can estimate the maximum benefit of the adaptive horizon method. If the computational time to determine the horizon t_1 is smaller than this benefit, adaptive horizon MPC can be useful. In the last row ($M_h = 25$) we observe that the time for determining the horizon is less than one second. In this case the adaptive horizon algorithm is $\approx 4s$ ($\approx 13\%$) faster than the MPC method with $N = 3$.
In the previous investigations the only task of the adaptive horizon algorithm was to provide an appropriate horizon. Now, we consider the case where we use the coarse grid solution as initial guess for the full problem, i.e., we set *usecontrol* = *true*. It is clear that the values of t_1 are almost the same as before, because this part of the algorithm is the same for both variants. The important values are those of t_2: We observe that the computing time for solving the optimal control problems is smaller than in the previous case. The explanation is that the interpolated control from the coarse grid yields a better initial guess than the classical MPC shift. With this variant of the algorithm and $M_h = 25$ we are able to compute the 300 MPC steps within $20.59s$, which is a notable speed up ($\approx 34\%$) compared to $31.04s$ for the MPC algorithm with horizon $N = 3$. In the last column of Table 6.11 we observe that the determined horizon is very robust with regard to the spatial discretization. This observation provides an important justification for the presented algorithm. Finally, we want to point out that the advantage of the second variant is not as clear as it

seems. In the following example we will observe a different behaviour.

Next, we shortly present two examples where the adaptive horizon algorithm does not yield convincing results. The first example is the one dimensional Schlögl equation with distributed control (6.1). We use the parameter $T = 0.01, \lambda = 0.01$ and $M_x = 200$. The optimal control problem is solved by the *BFGSINV II* method with tolerance $tol_{opt} = 10^{-6}$. For solving the semidiscretized PDE we use the *ROCK4* ODE solver with tolerance $tol_{ODE} = 10^{-8}$.
The numerical simulations show that the minimal stabilizing horizon is given by $N = 6$, see Figure 6.21. We choose the initial condition $y_0(x) = \sin(x)$ which is above the controlled steady state for $N = 5$. The parameters for the adaptive horizon algorithm are given by $N_{max} = 8, N_0 = 2, \rho = 3$ and $\bar{\alpha} = 0.05$. We consider again $n = 300$ MPC steps. The computing time for the minimal stabilizing horizon $N = 6$ is given by $11.25s$.
In Table 6.12 we observe a similar behaviour as for the example before: The time

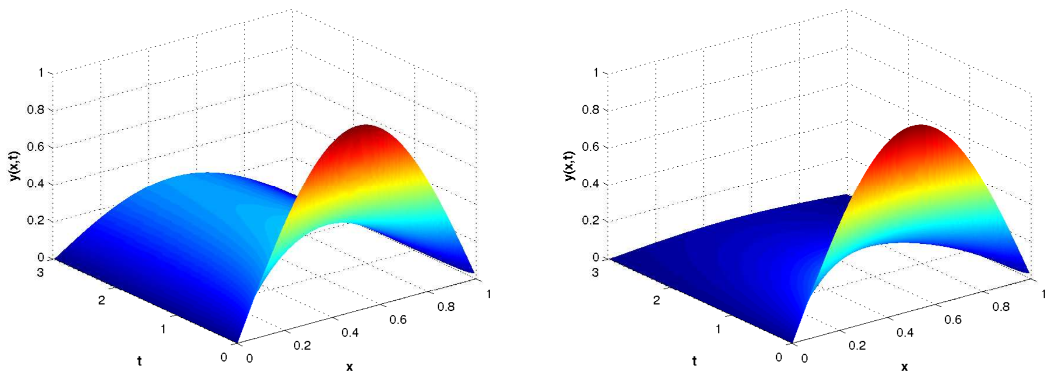


Figure 6.21: MPC closed loop trajectory for the Schlögl equation (6.1) with optimization horizon $N = 5$ (left) and $N = 6$ (right).

for determining the horizon t_1 scales quadratically with the grid size and the time for solving the optimal control problems nearly remains constant. Furthermore, we see that the adaptive horizon method only works beneficial if the dimension of the reduced system is small enough. The important difference to the catalytic rod model is the observation that the classical MPC shift variant yields an initial guess that is as good as the coarse grid interpolation method. This result is indicated by the similar values of t_2 for both variants. For the coarsest grid the first variant is even faster. This behaviour is reasonable because the quality of the initial guess decreases for a coarser grid. Moreover, we see again that the calculated horizon is not sensitive with regard to the spatial discretization. It is remarkable that the average horizon is smaller than $N = 5$ for which we do not observe stability for the plain MPC.
A comparison between the adaptive horizon algorithm and the plain MPC method by means of the computing time indicates the success of the adaptive method. In

	usecontrol = *false*			*usecontrol* = *true*			
M_h	t_1	t_2	$t_1 + t_2$	t_1	t_2	$t_1 + t_2$	$\bar{N}$
200	16.90	0	16.90	16.90	0	16.90	4.87
100	4.44	7.99	12.43	4.59	7.92	12.51	4.89
50	1.56	8.06	9.62	1.45	8.26	9.71	4.91
25	0.43	8.15	8.58	0.63	8.47	9.10	4.91

Table 6.12: Computation time for the adaptive horizon Algorithm 4.17, where M_h denotes the dimension of the coarse grid, t_1 the computing time to determine the horizon and t_2 the time for solving the optimal control problem. The comparative value is $t = 11.25s$ for $N = 6$.

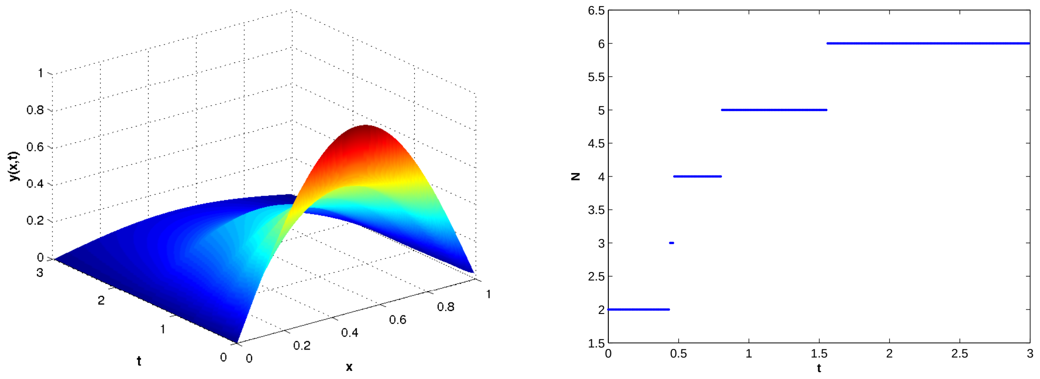


Figure 6.22: Adaptive horizon MPC closed loop trajectory for the Schlögl equation (6.1) (left) and the applied horizon in each MPC step (right).

the fastest variant this algorithm requires $8.58s$ while the plain MPC needs $11.25s$. However, this speed up is bought by the performance of the MPC closed loop solution. We observe a much slower convergence to the equilibrium (see Figure 6.22 (left)) compared with the plain MPC algorithm with horizon $N = 6$. Thus, for this example it is not clear whether the use of the adaptive horizon algorithm is reasonable. In Figure 6.22 (right) the applied horizon in each MPC step is displayed. In contrast to the catalytic rod example we observe a monotonically increasing behaviour until the minimal stabilizing horizon is reached.

We close this section by investigating the boundary controlled Schlögl equation (6.3). The parameters for the MPC simulation are given by $T = 0.01, \lambda = 0.01$ and $M_x = 200$. We use the *ROCK4* ODE solver with tolerance 10^{-8} for solving the semidiscretized PDE. The optimal control sequence is determined by the Algorithm *BFGSINV II*. For the considered parameters the minimal stabilizing horizon is given by $N = 9$. The MPC algorithm with this horizon requires $13.49s$ for $n = 300$ MPC

steps.
For the adaptive horizon algorithm we use the parameters $N_{max} = 11, N_0 = 2, \rho = 4$ and $\bar{\alpha} = 0.05$. In Table 6.13 the computing time of the adaptive horizon algorithm is displayed. The interesting value is the computational time for solving the optimal control problems t_2. It can be seen that only for the variant *usecontrol* = *true* and a fine spatial discretization a benefit is theoretically possible. However, in these cases the time to determine the horizon is too large. Thus, the overall time is for each adaptive horizon variant notable higher than for the plain MPC algorithm with horizon $N = 9$. Furthermore, the stabilization behaviour is much better for the non-adaptive MPC algorithm.

The bad performance of the adaptive horizon MPC is probably caused by two

	usecontrol = *false*			*usecontrol* = *true*				
M_h	t_1	t_2	$t_1 + t_2$	t_1	t_2	$t_1 + t_2$	$\bar{N}$	*stability*
200	27.96	0	27.96	27.96	0	27.96	8.13	y
100	9.39	15.00	24.39	9.18	11.29	20.47	8.04	y
50	3.99	16.79	20.78	4.11	12.31	16.42	7.65	y
25	1.78	16.16	17.94	1.54	16.3	17.84	6.79	n

Table 6.13: Computation time for the adaptive horizon Algorithm 4.17, where M_h denotes the dimension of the coarse grid, t_1 the computing time to determine the horizon and t_2 the time for solving the optimal control problem. The comparative value is $t = 13.49s$ for $N = 9$.

reasons. The first one becomes obvious by considering the computing time for different optimization horizons. In Table 6.14 we observe that the computing time is essentially decreasing for larger horizons. This fact is caused by the reason that a larger optimization horizon leads to a faster stabilization which is beneficial for the optimizer close to the equilibrium. (This effect reverses for $N \geq 15$.) Therefore, the main assumption of the adaptive horizon paradigm is not satisfied. While this problem occurs for all adaptive horizon methods, the second reason for the bad performance concerns the hierarchical grid method. In Table 6.13 it can be seen that the average horizon significantly changes with the spatial discretization. This contradicts our assumption that the coarse grid provides an appropriate horizon for the full problem which is the crucial condition for this algorithm. In the last column we display whether the determined horizon stabilizes the equation. It can be seen that the horizon determined on the coarsest grid is not sufficient to guarantee stability.

N	7	8	9	10	11	12	13	14
t	17.20	15.01	13.49	14.57	13.50	12.47	11.65	11.51

Table 6.14: Computing time for the non-adaptive MPC algorithm for different optimization horizons.

It should be mentioned that the findings in this section are also observable for different values of $\bar{\alpha}$.
The three presented examples show that a general statement about the efficiency of adaptive horizon algorithms is not possible. The performance strongly depends on the investigated system and the corresponding parameters. This observation was already done in the context of MPC for ODEs, cf. [45].

Appendix A

The following program code represents a simple implementation example of the Schlögl model with distributed control (5.2). The model classes are given by *react-diff.cpp* and *reactdiff.h*. The main program is *mpcpde.cpp*.

Listing A.1: reactdiff.cpp

```
#include "reactdiff.h"

#define MU 15.       // reaction parameter
#define LENGTH 1.    // interval length


ReactDiff::ReactDiff(int dimension_u, int dimension_x)
          : Model (dimension_u, dimension_x)

{
    _ctrl_dimension =dimension_u;  // dimension of control space
    _state_dimension=dimension_x;  // dimension of state space

    // control bounds
    _control_lb = new double[_ctrl_dimension];
    _control_ub = new double[_ctrl_dimension];
    for ( int i = 0; i < _ctrl_dimension; i++ )
    {
       _control_ub[i] =  10.;
       _control_lb[i] = -10.;
    }

    //ODE solver initialization
    _odefunc   =new yane::OdeSolve
                ::OdeFunction(stateEquation, _state_dimension);
    _odeadjoint=new yane::OdeSolve
                ::OdeFunction(adjointEquation, _state_dimension);

    _odeconfig=new yane::OdeSolve::OdeConfig();
    _odeconfig->setTolerance(1e-8, 1e-8);

    _odesolver=new yane::OdeSolve::ROCK4();
    _odesolver->reset(_odefunc, _odeconfig);
    _odesolveadjoint=new yane::OdeSolve::ROCK4();
    _odesolveadjoint->reset(_odeadjoint, _odeconfig);
}

double ReactDiff::semiFunction(double x)
{
        f=MU*(x-pow(x,3));
        return f;
}

double ReactDiff::deriFunction(double x)
{
        f=MU*(1.-3.*pow(x,2));
        return f;
}
//Semidiscretization of the state equation
void ReactDiff::stateEquation (int *n, double *t, double *x,
                               double *dx, double *u, int *ipar)
{
    int dim=_dimension_state;
    double hx=LENGTH/double(dim+1);

    dx[0]=(- 2.0* x[0] + x[1])/pow(hx,2)
          +semiFunction(x[0]) + u[0];

    for(int i=1;i<dim-1;i++)
    {

            dx[i]=(x[i-1]-2.0*x[i]+x[i+1])/pow(hx,2)
                  +semiFunction(x[i]) + u[i];
    }
    dx[dim-1]=(x[dim-2] -2.0* x[dim-1] )/pow(hx,2)
              +semiFunction(x[dim-1]) + u[dim-1];
}
//Semidiscretization of the adjoint equation
void ReactDiff::adjointEquation(int *n, double *t, double *x,
                                double *dx, double *rpar, int *ipar)
{
    int dim=_dimension_state;
    double hx=LENGTH/double(dim+1);

    dx[0]=( -2.0*x[0] + x[1])/pow(hx,2)
          +x[0]*deriFunction(rpar[0])+rpar[0]

    for(int i=1;i<dim-1;i++)
    {

            dx[i]=(x[i-1]-2.0*x[i]+x[i+1])/pow(hx,2)
                  +x[i]*deriFunction(rpar[i])+rpar[i];
    }
    dx[dim-1]=(x[dim-2] -2.0* x[dim-1] )/pow(hx,2)
              +x[dim-1]*deriFunction(rpar[dim-1])+rpar[dim-1];
}

void ReactDiff::predictState ( double t, double *x,
                               double *u, double h )
{
        _odesolver->init(t,x);
        _odesolver->calc(t+h, u);
}

void ReactDiff::predictAdjoint ( double t, double *x,
                                 double *rpar, double h )
{
        _odesolveadjoint->init(t,x);
        _odesolveadjoint->calc(t+h, rpar);
}
//Evaluation of the gradient
void ReactDiff::computeGradient ( double *p, double *u,
                 double *gradient, double _lambda, int _horizon)
{
    for( int j=0;j<_horizon;j++)
    {
      for( int i=0;i<_ctrl_dimension;i++)
      {
          gradient[j*_ctrl_dimension+i]
          =(p[j*_ctrl_dimension+i]+_lambda*u[j*_ctrl_dimension+i]);
      }
    }
}
```

Listing A.2: reactdiff.h

```
#ifndef REACTDIFF_H
#define REACTDIFF_H

#include <cmath>
#include <model.h>
#include <yane/odesolve.h>      //required ODE solver
/**
        @author Nils Altmueller <btmb07@btm5x6>
*/
class ReactDiff : public Model{
  public:

        ReactDiff(int dimension_u, int dimension_x);
        ~ReactDiff();

        static void stateEquation(int * n, double * t, double * x,
                                  double * dx, double * rpar, int * ipar );
        static void adjointEquation(int * n, double * t, double * x,
                                  double * dx, double * rpar, int * ipar );

        static double semiFunction(double x);
        static double deriFunction(double x);


        void predictState ( double t, double * x,
                            double * u, double h ) ;
        void predictAdjoint ( double t, double * x,
                              double * u, double h );
        void computeGradient ( double *p, double *u,
                double *gradient, double _lambda, int _horizon );

        yane::OdeSolve::OdeFunction *_odefunc;
        yane::OdeSolve::OdeFunction *_odeadjoint;
        yane::OdeSolve::OdeConfig *_odeconfig;
        yane::OdeSolve::OdeSolveFirst *_odesolver;
        yane::OdeSolve::OdeSolveFirst *_odesolveadjoint;

  private:

        int _ctrl_dimension;
        int _state_dimension;
};

#endif
```

Listing A.3: mpcpde.cpp

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <cmath>

#include "optimizer.h"          //optimal control algorithms
#include "testexample.h"        //examples from Chapter 5

#define HORIZON 10
#define T 0.01          //sampling time
#define LAMBDA 0.01     //regularization parameter
#define tol 1e-6        //tolerance for the optimizer
#define STEPS 300       //computed MPC steps

using namespace std;

int main(int argc, char *argv[])
{
        int DIMX=200;           //state dimension
        int DIMENSION=DIMX;     //control dimension

        Model *model= new ReactDiff(DIMENSION, DIMX);

        double *x0=new double[DIMX];            //state
        double *yd=new double[DIMX];            //desired state
        //optimal control sequence
        double *Control=new double[HORIZON*DIMENSION];

        for( int i = 0 ; i < DIMX; i++ )
        {
                //initial condition
                x0[i]=0.2*sin(M_PI*double(i+1)/double(DIMX+1));
                yd[i]=0.0;
        }

        for(int i=0;i<HORIZON;i++)
        {
                for(int j=0;j<DIMENSION;j++)
                {
                        Control[i*DIMENSION+j]=0.0;
                }
        }
        //initialization of the optimal control problem
        OPTIMIZE *optimize=new BFGSINV(model,yd,HORIZON,LAMBDA,T);
        optimize->setTolerance(tol);

        //MPC loop
        for(int k=0;k<STEPS;k++)
        {
            //solve optimal control problem
            optimize->calc(x0,Control);

            //shift the optimal control sequence
            for(int i=0;i<HORIZON-1;i++)
            {
              for( int j = 0 ; j < DIMENSION; j++ )
              {
                Control[i*DIMENSION+j]=Control[(i+1)*DIMENSION+j];
              }
            }
            for( int j = 0 ; j < DIMENSION; j++ )
            {
                Control[(HORIZON-1)*DIMENSION+j]=0.;
            }
        }

        delete optimize;
        delete model;

  return EXIT_SUCCESS;
}
```

Bibliography

[1] Dirk Abbeloos, Moritz Diehl, Michael Hinze, and Stefan Vandewalle. Nested multigrid methods for time-periodic, parabolic optimal control problems. *Computing and visualization in science*, 14(1):27–38, 2011.

[2] Alessandro Alla and Maurizio Falcone. An adaptive POD approximation method for the control of advection-diffusion equations. In *Control and Optimization with PDE Constraints*, pages 1–17. Springer, 2013.

[3] Alessandro Alla and Stefan Volkwein. Asymptotic stability of POD based model predictive control for a semilinear parabolic PDE. *arXiv preprint arXiv:1312.2145*, 2013.

[4] Nils Altmüller and Lars Grüne. Distributed and boundary model predictive control for the heat equation. *GAMM-Mitteilungen*, 35(2):131–145, 2012.

[5] Nils Altmüller and Lars Grüne. A comparative stability analysis of Neumann and Dirichlet boundary MPC for the heat equation. In *Control of Systems Governed by Partial Differential Equations*, volume 1, pages 133–138, 2013.

[6] Nils Altmüller, Lars Grüne, and Karl Worthmann. Instantaneous control of the linear wave equation. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, Budapest, Hungary*, pages 1895–1899, 2010.

[7] Nils Altmüller, Lars Grüne, and Karl Worthmann. Performance of NMPC schemes without stabilizing terminal constraints. In *Recent Advances in Optimization and its Applications in Engineering*, pages 289–298. Springer, 2010.

[8] Nils Altmüller, Lars Grüne, and Karl Worthmann. Receding horizon optimal control for the wave equation. In *49th IEEE Conference on Decision and Control (CDC)*, pages 3427–3432, 2010.

[9] Nils Altmüller, Lars Grüne, and Karl Worthmann. Improved stability estimates for MPC without terminal constraints applied to reaction diffusion PDEs. *Preprint*, 2012.

[10] Mario Annunziato and Alfio Borzì. A Fokker–Planck control framework for multidimensional stochastic processes. *Journal of Computational and Applied Mathematics*, 237(1):487–507, 2013.

[11] Antonios Armaou and Panagiotis D Christofides. Dynamic optimization of dissipative PDE systems using nonlinear order reduction. *Chemical Engineering Science*, 57(24):5083–5114, 2002.

[12] Giles Auchmuty. Optimal coercivity inequalities in $W^{1,p}(\Omega)$. *Proceedings of the Royal Society of Edinburgh*, 135(5):915–933, 2005.

[13] Giles Auchmuty, Behrouz Emamizadeh, and Mohsen Zivari. Dependence of Friedrichs' constant on boundary integrals. *Proceedings of the Royal Society of Edinburgh*, 135(5):935–940, 2005.

[14] Roland Becker, Dominik Meidner, and Boris Vexler. Efficient numerical solution of parabolic optimization problems by finite element methods. *Optimisation Methods and Software*, 22(5):813–833, 2007.

[15] Alain Bensoussan, Michel C Delfour, Giuseppe Da Prato, and Sanjoy K Mitter. *Representation and Control of Infinite Dimensional Systems (vol. 1)*. Birkhäuser Verlag, Basel, 1992.

[16] Thomas R Bewley, Parviz Moin, and Roger Temam. DNS-based predictive control of turbulence: an optimal benchmark for feedback algorithms. *Journal of Fluid Mechanics*, 447(2):179–225, 2001.

[17] Alfio Borzì and Karl Kunisch. A multigrid method for optimal control of time-dependent reaction diffusion processes. In *Fast solution of discretized optimization problems*, pages 50–57. Birkhäuser Verlag, Basel, 2001.

[18] Alfio Borzì and Volker Schulz. *Computational Optimization of Systems Governed by Partial Differential Equations*. SIAM, Philadelphia, 2012.

[19] Dietrich Braess. *Finite elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2001.

[20] Rico Buchholz, Harald Engel, Eileen Kammann, and Fredi Tröltzsch. On the optimal control of the Schlögl-model. *Computational Optimization and Applications*, 56(1):153–185, 2013.

[21] Thierry Cazenave and Alain Haraux. *An Introduction to Semilinear Evolution Equations*, volume 13. Oxford University Press on Demand, 1998.

[22] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[23] Panagiotis D Christofides. *Nonlinear and Robust Control of PDE Systems: Methods and Applications to Transport Reaction Processes*. Birkhäuser Verlag, Boston, 2001.

[24] Philippe G Ciarlet. *Linear and Nonlinear Functional Analysis with Applications.* SIAM, Philadelphia, 2013.

[25] Jean-Michel Coron. *Control and Nonlinearity.* Number 136. AMS Bookstore, 2009.

[26] Ruth F Curtain and Hans Zwart. *An Introduction to Infinite-Dimensional Linear Systems Theory.* Springer Verlag, New-York, 1995.

[27] Franz Diwoky and Stefan Volkwein. Nonlinear boundary control for the heat equation utilizing proper orthogonal decomposition. In *Fast Solution of Discretized Optimization Problems*, pages 73–87. Springer, 2001.

[28] Stevan Dubljevic and Panagiotis D Christofides. Predictive control of parabolic PDEs with boundary control actuation. *Chemical Engineering Science*, 61(18):6239–6248, 2006.

[29] Stevan Dubljevic, Nael H El-Farra, Prashant Mhaskar, and Panagiotis D Christofides. Predictive control of parabolic PDEs with state and control constraints. *International Journal of robust and nonlinear control*, 16(16):749–772, 2006.

[30] Stevan Dubljevic, Prashant Mhaskar, Nael H El-Farra, and Panagiotis D Christofides. Predictive control of transport-reaction processes. *Computers & chemical engineering*, 29(11):2335–2345, 2005.

[31] Heinz W. Engl. *Integralgleichungen.* Springer, Wien, 1997.

[32] Lawrence Evans. *Partial Differential Equations.* American Mathematical Society, Providence, RI, 1998.

[33] E Fernández-Cara. Null controllability of the semilinear heat equation. *ESAIM: Control, Optimisation and Calculus of Variations*, 2:87–103, 1997.

[34] Carl Geiger and Christian Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben.* Springer, Heidelberg, 1999.

[35] Carl Geiger and Christian Kanzow. *Theorie und Numerik Restringiererter Optimierungsaugaben.* Springer, Heidelberg, 2002.

[36] Matthias Gerdts, Günter Greif, and Hans Josef Pesch. Numerical optimal control of the wave equation: optimal boundary control of a string to rest in finite time. *Mathematics and Computers in Simulation*, 79(4):1020–1032, 2008.

[37] Jane Ghiglieri and Stefan Ulbrich. Optimal flow control based on POD and MPC and an application to the cancellation of Tollmien-Schlichting waves. *Optimization Methods and Software*, 29, Iss:5, 2014.

[38] Pontus Giselsson. Adaptive nonlinear model predictive control with suboptimality and stability guarantees. In *49th IEEE Conference on Decision and Control (CDC), 2010*, pages 3644–3649. IEEE, 2010.

[39] Roland Griesse. *Parametric Sensitivity Analysis for Control-Constrained Optimal Control Problems Governed by Systems of Parabolic Partial Differential Equations.* PhD thesis, Universität Bayreuth, Germany, 2003.

[40] Roland Griesse. A reduced SQP algorithm for the optimal control of semilinear parabolic equations. In *System Modeling and Optimization XX*, pages 239–253. Springer, 2003.

[41] Roland Griesse and Andrea Walther. Evaluating gradients in optimal control: continuous adjoints versus automatic differentiation. *Journal of optimization theory and applications*, 122(1):63–86, 2004.

[42] Christian Grossmann and Hans-Görg Roos. *Numerical Treatment of Partial Differential Equations.* Springer, Heidelberg, 2007.

[43] Lars Grüne. Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems. *SIAM Journal on Control and Optimization*, 48(2):1206–1228, 2009.

[44] Lars Grüne and Jürgen Pannek. Practical NMPC suboptimality estimates along trajectories. *Systems & Control Letters*, 58(3):161–168, 2009.

[45] Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control.* Springer, London Heidelberg New York, 2011.

[46] Lars Grüne, Jürgen Pannek, Martin Seehafer, and Karl Worthmann. Analysis of unconstrained nonlinear MPC schemes with time varying control horizon. *SIAM Journal on Control and Optimization*, 48(8):4938–4962, 2010.

[47] Martin Gugat, Günter Leugering, and G Sklyar. L^p-optimal boundary control for the wave equation. *SIAM Journal on Control and Optimization*, 44(1):49–74, 2005.

[48] Sabine Hein. *MPC/LQG-Based Optimal Control of Nonlinear Parabolic PDEs.* PhD thesis, Technische Universität Chemnitz, 2009.

[49] Roland Herzog and Karl Kunisch. Algorithms for PDE-constrained optimization. *GAMM-Mitteilungen*, 33(2):163–176, 2010.

[50] Michael Hinze and Karl Kunisch. Three control methods for time-dependent fluid flow. *Flow, Turbulence and Combustion*, 65(3-4):273–298, 2000.

[51] Michael Hinze and Karl Kunisch. Second order methods for optimal control of time-dependent fluid flow. *SIAM Journal on Control and Optimization*, 40(3):925–946, 2001.

[52] Michael Hinze, Rene Pinnau, Michael Ulbrich, and Stefan Ulbrich. *Optimization with PDE constraints, volume 23 of Mathematical Modelling: Theory and Applications*. Springer, New York, 2009.

[53] Michael Hinze and Stefan Volkwein. Analysis of instantaneous control for the Burgers equation. *Nonlinear Analysis: Theory, Methods & Applications*, 50(1):1–26, 2002.

[54] Svein Hovland, Jan Tommy Gravdahl, and Karen E Willcox. Explicit model predictive control for large-scale systems via model reduction. *Journal of guidance, control, and dynamics*, 31(4):918–926, 2008.

[55] Ralf Hundhammer and Günter Leugering. *Online Optimization of Large Scale Systems*, chapter Instantaneous Control of Vibrating String Networks, pages 229–249. Springer, 2001.

[56] Willem Hundsdorfer and Jan G Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations.* Springer, Heidelberg, 2003.

[57] Kazufumi Ito and Karl Kunisch. Receding horizon optimal control for infinite dimensional systems. *ESAIM: control, optimisation and calculus of variations*, 8(1):741–760, 2002.

[58] Kazufumi Ito and Karl Kunisch. The primal-dual active set method for nonlinear optimal control problems with bilateral constraints. *SIAM Journal on Control and Optimization*, 43(1):357–376, 2004.

[59] Kazufumi Ito and Karl Kunisch. *Lagrange Multiplier Approach to Variational Problems and Applications.* SIAM, Philadelphia, 2008.

[60] Dante Kalise and Axel Kröner. Reduced-order minimum time control of advection-reaction-diffusion systems via dynamic programming. *To appear in MTNS14 Proceedings, 8 pp.*, 2014.

[61] Peter Knabner and Lutz Angerman. *Numerical Methods for Elliptic and Parabolic Partial Differential Equations*, volume 44. Springer Verlag, New York, 2003.

[62] Miroslav Krstic and Andrey Smyshlyaev. *Boundary Control of PDEs: A Course on Backstepping Designs.* Society for Industrial and Applied Mathematics, Philadelphia, 2008.

[63] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische Mathematik*, 90(1):117–148, 2001.

[64] Karl Kunisch and Stefan Volkwein. Optimal snapshot location for computing POD basis functions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(03):509–529, 2010.

[65] Karl Kunisch, Stefan Volkwein, and L Xie. HJB-POD-based feedback design for the optimal control of evolution problems. *SIAM Journal on Applied Dynamical Systems*, 3(4):701–722, 2004.

[66] John Lagnese and Günter Leugering. *Domain Decomposition Methods in Optimal Control of Partial Differential Equations.* Birkhäuser, Basel, 2004.

[67] Irena Lasiecka and Roberto Triggiani. *Control Theory for Partial Differential Equations: Volume 1, Abstract Parabolic Systems: Continuous and Approximation Theories*, volume 1. Cambridge University Press, 2000.

[68] Jacques Louis Lions. *Optimal Control of Systems Governed by Partial Differential Equations.* Springer-Verlag, Berlin, 1971.

[69] Weijiu Liu. Boundary feedback stabilization of an unstable heat equation. *SIAM journal on Control and Optimization*, 42(3):1033–1043, 2003.

[70] Jakob Löber, Rhoslyn Coles, Julien Siebert, Harald Engel, and Eckehard Schöll. Control of chemical wave propagation. *arXiv preprint arXiv:1403.3363*, 2014.

[71] Dominik Meidner. *Adaptive Space-Time Finite Element Methods for Optimization Problems Governed by Nonlinear Parabolic Systems.* PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2007.

[72] Dominik Meidner and Boris Vexler. A priori error estimates for space-time finite element discretization of parabolic optimal control problems part i: Problems without control constraints. *SIAM Journal on Control and Optimization*, 47(3):1150–1177, 2008.

[73] Pekka Neittaanmäki and Dan Tiba. *Optimal Control of Nonlinear Parabolic Systems: Theory, Algorithms, and Applications.* Marcel Dekker, New York, 1994.

[74] Ira Neitzel and Boris Vexler. A priori error estimates for space–time finite element discretization of semilinear parabolic optimal control problems. *Numerische Mathematik*, 120(2):345–386, 2012.

[75] Vesna Nevistic and James A Primbs. Receding horizon quadratic optimal control: Performance bounds for a finite horizon strategy. In *Proc. European Control Conference.* Citeseer, 1997.

[76] J Nocedal and SJ Wright. *Numerical Optimization.* Springer New York, 1999.

[77] J. Pannek. *Receding Horizon Control: A Suboptimality-based Approach.* PhD thesis, University of Bayreuth.

[78] A Pazy. *Semigroups of Linear Operators and Applications to Partial Differential Equations.* Springer-Verlag, New York, 1983.

[79] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[80] J. Rawlings and D. Mayne. *Model Predictive Control: Theory and Design.* Madison, WI, Nob Hill Publishing, 2009.

[81] Jean Pierre Raymond and Housnaa Zidani. Hamiltonian Pontryagin's principles for control problems governed by semilinear parabolic equations. *Applied Mathematics and Optimization*, 39(2):143–177, 1999.

[82] Klaus Schittkowski. NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems. *Annals of operations research*, 5(2):485–500, 1986.

[83] Friedrich Schlögl. Chemical reaction models for non-equilibrium phase transitions. *Zeitschrift für Physik*, 253(2):147–161, 1972.

[84] Andrey Smyshlyaev and Miroslav Krstic. Backstepping observers for a class of parabolic PDEs. *Systems & Control Letters*, 54(7):613–625, 2005.

[85] Eduardo D Sontag. Smooth stabilization implies coprime factorization. *Automatic Control, IEEE Transactions on*, 34(4):435–443, 1989.

[86] Eduardo D Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems.* Springer, New York, 1998.

[87] Renato Spigler and Marco Vianello. Convergence analysis of the semi-implicit Euler method for abstract evolution equations. *Numerical functional analysis and optimization*, 16(5-6):785–803, 1995.

[88] Walter A Strauss. *Partial Differential Equations: An Introduction.* Wiley, New York, 1992.

[89] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems.* Springer-Verlag, Berlin, 1984.

[90] Roberto Triggiani. Boundary feedback stabilizability of parabolic equations. *Applied Mathematics and Optimization*, 6(1):201–220, 1980.

[91] Fredi Tröltzsch. *Optimal Control of Partial Differential Equations, volume 112 of Graduate Studies in Mathematics.* American Mathematical Society, Providence, 2010.

[92] Fredi Tröltzsch and Stefan Volkwein. POD a-posteriori error estimates for linear-quadratic optimal control problems. *Computational Optimization and Applications*, 44(1):83–115, 2009.

[93] Michael Ulbrich. *Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*. SIAM, Philadelphia, 2011.

[94] Harald Voit. *MPC Schemata mit variablem Kontrollhorizont*. Masters thesis, University of Bayreuth, 2008.

[95] Stefan Volkwein. Nonlinear conjugate gradient methods for the optimal control of laser surface hardening. *Optimization Methods and Software*, 19(2):179–199, 2004.

[96] Stefan Volkwein. Model reduction using proper orthogonal decomposition. *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. see http://www. uni-graz. at/imawww/volkwein/POD. pdf*, 2011.

[97] Gregory Von Winckel and Alfio Borzì. Computational techniques for a quantum control problem with H^1-cost. *Inverse Problems*, 24(3):034007, 2008.

[98] Martin Weiser, Tobias Gänzler, and Anton Schiela. A control reduced primal interior point method for a class of control constrained optimal control problems. *Computational Optimization and Applications*, 41(1):127–145, 2008.

[99] Karl Worthmann. *Stability Analysis of Unconstrained Receding Horizon Control Schemes*. PhD thesis, University of Bayreuth, 2011.

[100] Weiguo Xie, Ioannis Bonis, and Constantinos Theodoropoulos. Off-line model reduction for on-line linear MPC of nonlinear large-scale distributed systems. *Computers & Chemical Engineering*, 35(5):750–757, 2011.

[101] Enrique Zuazua. Optimal and approximate control of finite-difference approximation schemes for the 1-d wave equation. *Rend. Mat. Appl.(7)*, 24(2):201–237, 2004.

[102] Enrique Zuazua. Controllability of partial differential equations. *HAL Id: cel-00392196, see https://cel.archives-ouvertes.fr/cel-00392196*, 2006.

Ehrenwörtliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die von mir vorgelegte Dissertation mit dem Thema

„Model Predictive Control for Partial Differential Equations“

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Zudem erkläre ich, dass

- ich diese Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe und
- Hilfe von gewerblichen Promotionsberatern bzw. -vermittlern oder ähnlichen Dienstleistern von mir weder in Anspruch genommen wurde noch künftig in Anspruch genommen wird.

Bayreuth, den ....

Nils Altmüller