I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Marco Pavone, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Charbel Farhat

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

J Gerdes

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Rock

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
Abstract

Dynamics models are used to develop control algorithms for many applications, including aerospace vehicles, self-driving cars, robots, industrial plants, and more. Physics-based models are particularly common, due to their interpretability and ability to efficiently encode knowledge of the system’s behavior. An ideal model accurately represents the system’s dynamics while not introducing significant complexity into the control algorithm. However, conflict between model accuracy and complexity often arises, and thus the two must be balanced to satisfy practical requirements. While low-dimensional models provide sufficient accuracy for some control applications, there are also many where high-dimensional models are required. One important application is the control of systems best modeled by partial differential equations (PDEs), such as systems with fluid flows or fluid-structure interaction (e.g. highly maneuverable or flexible aircraft), deformable structures (e.g. soft robots), and more. In practice, infinite-dimensional PDE models are generally semi-discretized to produce high-fidelity ordinary differential equation (ODE) models that are finite-dimensional. Since the dimension of these models can range from thousands to millions, standard model-based controller design can be extremely challenging.

In this thesis, we propose an approach for efficiently designing high-performing controllers based on high-dimensional models. Specifically, we develop a model predictive control (MPC) algorithm for solving constrained optimal control problems that leverages high-fidelity, but low-dimensional, reduced order approximations of the original model to satisfy practical computational requirements. In the linear setting, we combine existing ideas from tube MPC with novel approaches for controller synthesis and analysis to develop a reduced order MPC (ROMPC) scheme for solving robust, output feedback control problems, and we provide theoretical closed-loop performance guarantees that explicitly account for model reduction error. We also extend the ROMPC scheme to the nonlinear setting by exploiting piecewise-affine reduced order models. We motivate and validate the proposed approach through two case studies. First, we use a linear, coupled rigid-body/fluid dynamics model for aircraft control, where the high-dimensional computational fluid dynamics (CFD) model has over one million dimensions. Second, we use a nonlinear finite element model (FEM) with over ten thousand dimensions to control a soft robot. Simulation and hardware experiments are used in both studies to demonstrate the practicality and performance of ROMPC.
Acknowledgments

There are many family, friends, and colleagues who I would like to acknowledge for shaping my graduate school journey, and who certainly deserve so much more than a few words on a page.

First, I would like to acknowledge my Ph.D. advisor, Professor Marco Pavone. Transitioning from a student to a researcher is not a trivial task, but Marco was there to teach me how to ask the right questions and motivate me when the path was not clear. Beyond his one-on-one advising, Marco also spent considerable effort developing and maintaining the Autonomous Systems Lab, which I had the honor of calling my academic home and where I benefited significantly from many amazing colleagues and mentors.

Next, I would like to thank my collaborators for supplying their knowledge, insights, and time. In particular, I would like to thank Professor Charbel Farhat and Andrew McClellan for an amazing collaboration which led to a substantial portion of this thesis. Having the opportunity to work so closely with researchers in a completely different field has provided me with a valuable perspective and many lessons learned. I would also like to thank Sander Tonkens, Benoit Landry, Sumeet Singh, and Mo Chen, who, as collaborators and fellow members of the Autonomous Systems Lab, provided extremely valuable feedback on my research, provided constant outlets for brainstorming new ideas, and were by my side in the paper-writing trenches. Another special thanks are reserved for Ravi Haksar and Professor Mac Schwager, who mentored me for my very first research project at Stanford.

I would also like to acknowledge Professors Charbel Farhat, Stephen Rock, and J. Christian Gerdes for giving their time to be members of my dissertation reading committee, and for the valuable feedback and advice they provided. Professor Farhat, Rock, and Gerdes, along with Professor Monroe Kennedy III, were also gracious enough to panel my thesis defense, where they provided useful comments and thoughtful questions. Another acknowledgement must be made for the support of the Department of Defense National Defense Science and Engineering Fellowship (NDSEG) Program, which provided me with the funding to pursue the research presented in this thesis.

A huge amount of gratitude is also reserved for all of the members of the ASL, past and present, for providing me with a constant source of inspiration, knowledge, and friendship. I have great memories with every one of you, from lab outings such as beach picnics or ski trips, hanging out after the robotics seminar on Fridays, or even just random chats in our little oasis in the basement.
of Durand. Many other friends have also contributed to my happiness during my time at Stanford, including Benoit, Lyndsey, Lauren, and Miles, who have gifted me with many great memories centered around great food, hikes, and escapes from the Bay Area.

Along my entire educational path, ranging from my childhood in Fairborn to my undergraduate studies at Purdue, and continuing through my graduate studies at Stanford, my family has been the greatest source of love and support. I also completely contribute my passion for engineering and learning to my family, and particularly to Dad, Mom, Ben, Mommom, Bobpop, Grandma, and Grandaddy. They provided me with the most amazing childhood, cultivated within me an inquisitive spirit and provided the experiences and opportunities to satisfy it, and have never left my side. I have also had the fortune to be welcomed into the loving arms of the Bell family, who have provided me with many happy memories and a second place to call home. Finally, as a constant companion during my journey at Stanford, my loving wife Natalie has been there to celebrate the good times and pull me up in the tough times, and for that I will be forever grateful.
# Contents

Abstract iv

Acknowledgments v

1 Introduction 1

1.1 Control of Infinite-Dimensional Systems . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.1.1 Model Order Reduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.1.2 Other Approaches . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.2 Summary of Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

I Linear Reduced Order Model Predictive Control 8

2 Linear Reduced Order Model Predictive Control 9

2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2 Model Description and Problem Formulation . . . . . . . . . . . . . . . . . . . . . . 12

2.2.1 Linear Full Order Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.2.2 Constrained Optimal Control Problem . . . . . . . . . . . . . . . . . . . . . . . 13

2.2.3 Linear Reduced Order Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

2.3 Controller Definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.3.1 Reduced Order Controller and State Estimator . . . . . . . . . . . . . . . . . . 15

2.3.2 Reduced Order Optimal Control Problem . . . . . . . . . . . . . . . . . . . . . 15

2.3.3 Linear ROMPC Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

2.4 Closed-Loop Performance Guarantees . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.4.1 Closed-Loop Error Dynamics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.4.2 Closed-Loop Robust Constraint Satisfaction . . . . . . . . . . . . . . . . . . . . 19

2.4.3 Closed-Loop Stability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

2.4.4 Computational Challenges . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

2.5 Controller Synthesis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
6.2 Model Description and Problem Formulation ........................................ 84
  6.2.1 Nonlinear Full Order Model ..................................................... 84
  6.2.2 Constrained Optimal Control Problem ....................................... 85
  6.2.3 Piecewise-Affine Reduced Order Model ...................................... 86
6.3 Controller Definition ........................................................................... 88
  6.3.1 Reduced Order Optimal Control Problem ..................................... 88
  6.3.2 Reduced Order Controller and State Estimator ............................. 89
6.4 Future Work ....................................................................................... 90

7 Soft Robot Control Using FEM-Based MPC ........................................... 93
  7.1 Introduction ..................................................................................... 94
  7.2 Soft Robot Finite Element Model .................................................... 95
  7.3 Piecewise-Affine Reduced Order Model .......................................... 96
    7.3.1 Model Reduction Via Proper Orthogonal Decomposition .......... 96
    7.3.2 Nonlinear Model Reduction ..................................................... 97
    7.3.3 Algorithm for Constructing the Piecewise-Affine ROM ............ 98
  7.4 Simulation Experiments .................................................................. 100
    7.4.1 Controller Parameters .......................................................... 101
    7.4.2 Results ................................................................................. 102
  7.5 Hardware Experiments .................................................................... 105
    7.5.1 Experimental Setup ............................................................... 105
    7.5.2 Controller Parameters .......................................................... 106
    7.5.3 Results ................................................................................. 107
  7.6 Conclusion and Future Work ............................................................ 108

8 Conclusion ............................................................................................ 111
  8.1 Future Directions ............................................................................. 112
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Linear ROMPC Controller Synthesis Results</td>
<td>27</td>
</tr>
<tr>
<td>2.2</td>
<td>Linear ROMPC Controller Synthesis Computation Times</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>Linear ROMPC Error Bound Results</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Error Bounds: Distillation Column</td>
<td>47</td>
</tr>
<tr>
<td>4.2</td>
<td>Error Bounds: Supersonic Diffuser</td>
<td>53</td>
</tr>
<tr>
<td>5.1</td>
<td>CFD-based ROMPC Simulations, Longitudinal Motion Constraints</td>
<td>69</td>
</tr>
<tr>
<td>5.2</td>
<td>CFD-based ROMPC Simulations, Lateral Motion Constraints</td>
<td>69</td>
</tr>
<tr>
<td>5.3</td>
<td>CFD-based ROMPC Simulations, Control Constraints</td>
<td>70</td>
</tr>
<tr>
<td>7.1</td>
<td>Soft Robot Simulation Results: Example 1</td>
<td>102</td>
</tr>
<tr>
<td>7.2</td>
<td>Soft Robot Simulation Results: Example 2</td>
<td>102</td>
</tr>
<tr>
<td>7.3</td>
<td>Soft Robot Hardware Results: Example 1</td>
<td>108</td>
</tr>
<tr>
<td>7.4</td>
<td>Soft Robot Hardware Results: Example 2</td>
<td>108</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Linear ROMPC Block Diagram</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Linear ROMPC Tube-based Robust Constraint Satisfaction</td>
<td>18</td>
</tr>
<tr>
<td>3.1</td>
<td>Linear ROMPC Error Matrix Norm Decay</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Distillation Column Schematic</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>Distillation Column Simulation, Product Compositions</td>
<td>48</td>
</tr>
<tr>
<td>4.3</td>
<td>Distillation Column Simulation, Control Inputs</td>
<td>48</td>
</tr>
<tr>
<td>4.4</td>
<td>Distillation Column Simulation, Control Input Rates</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Chemical Reactor Schematic</td>
<td>49</td>
</tr>
<tr>
<td>4.6</td>
<td>Chemical Reactor Simulation, Temperature and Concentration Profiles</td>
<td>55</td>
</tr>
<tr>
<td>4.7</td>
<td>Chemical Reactor Simulation, Control Inputs</td>
<td>56</td>
</tr>
<tr>
<td>4.8</td>
<td>Chemical Reactor Simulation, ROMPC vs. LQR</td>
<td>56</td>
</tr>
<tr>
<td>4.9</td>
<td>Heat Transfer Problem: Flat Plate Layout</td>
<td>57</td>
</tr>
<tr>
<td>4.10</td>
<td>2D Heatflow Simulation, Setpoint Tracking</td>
<td>57</td>
</tr>
<tr>
<td>4.11</td>
<td>2D Heatflow Simulation, Control Input</td>
<td>57</td>
</tr>
<tr>
<td>4.12</td>
<td>Supersonic Diffuser Schematic</td>
<td>57</td>
</tr>
<tr>
<td>4.13</td>
<td>Supersonic Diffuser Simulation, Throat Mach Perturbation</td>
<td>58</td>
</tr>
<tr>
<td>4.14</td>
<td>Supersonic Diffuser Simulation, Inlet Bleed Control</td>
<td>58</td>
</tr>
<tr>
<td>5.1</td>
<td>Autonomous Carrier Landing Problem, Glideslope Trajectory</td>
<td>62</td>
</tr>
<tr>
<td>5.2</td>
<td>Autonomous Carrier Landing Problem, Inertial and Target Reference Frames</td>
<td>63</td>
</tr>
<tr>
<td>5.3</td>
<td>Autonomous Carrier Landing Problem, Target and Body-fixed Reference Frames</td>
<td>63</td>
</tr>
<tr>
<td>5.4</td>
<td>CFD-based ROMPC Simulations, mAEWing2</td>
<td>68</td>
</tr>
<tr>
<td>5.5</td>
<td>CFD-based ROMPC Simulations, Longitudinal Response</td>
<td>74</td>
</tr>
<tr>
<td>5.6</td>
<td>CFD-based ROMPC Simulations, Lateral Response</td>
<td>74</td>
</tr>
<tr>
<td>5.7</td>
<td>CFD-based ROMPC Simulations, Control</td>
<td>75</td>
</tr>
<tr>
<td>5.8</td>
<td>CFD-based ROMPC Flight Experiment Testbed</td>
<td>75</td>
</tr>
<tr>
<td>5.9</td>
<td>CFD-based ROMPC Flight Experiment Testbed, 3D Laser Scan</td>
<td>75</td>
</tr>
</tbody>
</table>
5.10 CFD-based ROMPC Flight Experiment Testbed, CFD Mesh ........................................ 76
5.11 CFD-based ROMPC Flight Experiment Testbed, CFD Mesh ........................................ 76
5.12 CFD-based ROMPC Flight Experiments, Pre-flight .................................................... 76
5.13 CFD-based ROMPC Flight Experiments, Ground Track .............................................. 77
5.14 CFD-based ROMPC Flight Experiments, Velocity ......................................................... 77
5.15 CFD-based ROMPC Flight Experiments, Position ....................................................... 78
5.16 CFD-based ROMPC Flight Experiments, Attitude ....................................................... 78
5.17 CFD-based ROMPC Flight Experiments, Control ....................................................... 79

7.1 “Diamond” Soft Robot: Hardware and Finite Element Mesh ........................................... 95
7.2 Soft Robot Simulation Results: Example 1 ................................................................. 103
7.3 Soft Robot Simulation Results: Example 2 ................................................................. 104
7.4 Soft Robot Hardware Setup ......................................................................................... 106
7.5 Soft Robot Hardware Results: Example 1 ................................................................. 109
7.6 Soft Robot Hardware Results: Example 2 ................................................................. 110
7.7 Soft Robot Hardware Experiment: Desired Trajectories ............................................. 110
Chapter 1

Introduction

Controlling dynamical systems is an engineering challenge that has been of intense interest for decades. Modern applications are widespread, and include the control and automation of vehicles, aircraft, industrial processes, electrical systems, power generation, medical robotics, and more. Research into novel control techniques has also continued to grow as the systems to be controlled become more diverse and complex. One important class of control techniques is optimal control, which broadly seeks to design controllers that explicitly optimize the system’s behavior with respect to some performance metric. These methods typically leverage a mathematical model of the system’s behavior, and are classified as either open-loop or closed-loop. In this introduction, we briefly discuss the advantages and disadvantages of these different approaches to model-based optimal control, and introduce the concept of model predictive control (MPC), which combines advantages from both open and closed-loop methods. We will then proceed to introduce some of the current challenges and limitations of model predictive control, associated with trade-offs between practical computational requirements and fidelity of the dynamics models that can be considered.

Open-loop optimal control methods, also referred to as trajectory optimization methods, generate a sequence of control inputs to be applied to the system from a particular initial condition. While efficient numerical methods have been developed for solving such problems, open-loop control is not robust against uncertainty due to the lack of a feedback mechanism to correct for deviations from the optimized trajectory. On the other hand, closed-loop methods, based on the Hamilton-Jacobi-Bellman equation or dynamic programming, generate control laws or policies that globally define optimal control inputs for the system at any state. Closed-loop methods therefore provide robustness against uncertainty, but in practice are often computationally intractable to synthesize. Model predictive control, also known as receding horizon control, combines the computational advantages from open-loop methods with robustness properties of closed-loop methods. This is accomplished by repeatedly solving (typically online) open-loop optimal control problems over a finite-horizon. This allows the controller to utilize the most recent information about the system’s state to incorporate
feedback.

Over the past few decades, MPC has become an extremely popular technique for constrained optimal control due to advances in algorithms for numerically solving optimal control problems with state and control constraints. These problems are of significant practical interest because the inclusion of constraints allows the controller to operate safely at higher levels of performance. However, while MPC algorithms offer advanced capabilities beyond classical control techniques, they are also burdened by unique challenges. The most significant and fundamental challenge comes from the real-time computational requirements imposed by the receding horizon approach. Even with increased computational resources and better algorithms for solving optimization problems, the complexity of the optimal control problem is still limited in practice. In the context of MPC, this limits the complexity and dimensionality of the dynamics model that can be used, which can negatively impact the controller’s performance if a low-fidelity model must be used. Previous applications of MPC have therefore been restricted to control problems where low-dimensional models are sufficiently accurate, such as when rigid-body assumptions can eliminate the need to model structural dynamics. Another approach is to limit the system’s operation to regions where low-dimensional models are accurate. For example, aircraft models parameterized by aerodynamic coefficients are low-fidelity but sufficiently accurate for many flight conditions.

Many other interesting control applications benefit significantly from the use of high-fidelity models that are high-dimensional. Infinite-dimensional systems (e.g. systems with fluid flows, deformable or flexible structures, chemical processes, heat transfer) are an important class of systems where low-dimensional models are often insufficient. Physics-based models of infinite-dimensional systems are expressed as partial differential equations, which are generally semi-discretized in space to generate finite-dimensional ordinary differential equation models. These ODE models are high-fidelity representations of the system dynamics, but are also generally extremely high-dimensional, often having dimensions ranging from thousands to millions. Model predictive control of infinite-dimensional systems is therefore challenging because the models are too high-dimensional to satisfy real-time computation requirements. In fact, these models can be so high-dimensional that even routine offline computations for controller synthesis (e.g. solving Lyapunov or Riccati equations) can become intractable. In this thesis, we enable constrained optimal control of infinite-dimensional systems by developing a computationally efficient model predictive control framework that can still leverage very high-dimensional models.

1.1 Control of Infinite-Dimensional Systems

As previously mentioned, semi-discretized PDE models of infinite-dimensional systems are generally very high-dimensional, making model-based controller design challenging. In this section, we introduce several approaches that have been proposed for controlling infinite-dimensional systems,
including perhaps the most widely used approach that leverages model order reduction techniques to generate low-dimensional approximate models before synthesizing the controller.

To begin, some approaches work directly with high-dimensional models by using numerical methods that exploit model sparsity for efficiency. For example, Benner (2004) suggests exploiting sparsity in some linear models to efficiently solve algebraic Riccati equations for infinite-horizon linear quadratic regulator (LQR) problems. However, this approach is still limited computationally to systems with relatively low-dimensional models, and does not extend well to model predictive control schemes that need real-time algorithms to run online. Mitchell and Overton (2015) also exploit sparsity within the context of $\mathcal{H}_\infty$ control of linear systems to efficiently design a low-order controller for a large-scale system. Within the context of receding horizon control of infinite-dimensional systems, some preliminary works have performed theoretical analysis without addressing practical computational issues. Ito and Kunisch (2002) propose a receding horizon strategy that guarantees system stability by adding terminal costs defined by control Lyapunov functions. Similar results have also been demonstrated without the use of the control Lyapunov functions, for example in (Grüne, 2009) for discrete-time systems, in (Altmüller et al., 2010) which focuses on minimal stabilizing horizons for the wave equation, and in (Azmi and Kunisch, 2016) for continuous-time analysis of Burgers’ equation.

Computationally practical approaches for synthesizing optimal controllers for PDE systems have also been developed. Perhaps the most prominent strategy is to leverage model order reduction techniques to define a high-fidelity, but low-dimensional, model that captures the dominant behavior of the spatially discretized PDE model. These low-dimensional models can then be used in place of the high-dimensional models within existing optimal control approaches, such as LQR control and model predictive control.

### 1.1.1 Model Order Reduction

A popular approach for model-based control of PDE systems is to leverage principled model order reduction techniques (Antoulas, 2005; Benner et al., 2015) to first reduce the dimensionality of the high-dimensional spatially-discretized PDE model. Model order reduction techniques have been proven to be effective for producing high-fidelity, but low-dimensional, models of PDE systems in a number of applications, including the simulation and control of fluid flows (Hovland et al., 2008a), fluid-structure interaction problems (Farhat et al., 2003), aircraft control (McClellan et al., 2020), Lorenzetti et al. (2020), simulation and control of soft robots (Goury and Duriez, 2018; Thieffry et al., 2019a), chemical process control (Agudelo et al., 2007; Marquez et al., 2013), finance (Schu, 2013), nuclear reactor control (Ou et al., 2011; Sartori et al., 2014), and more.

Many widely used model order reduction techniques are projection-based, where a pair of low-rank bases are identified in the high-dimensional state space and then used to compress the dynamics
model via an orthogonal or oblique projection. Balanced truncation is a model order reduction technique for linear systems that leverages the control theoretic notions of reachability and observability. In particular, this model reduction procedure seeks to retain only the model behavior that is the most reachable and observable and ignores the rest. Balanced truncation is appealing for control applications due to its strong theoretical properties, such as guaranteed stability preservation and bounds on the $H_\infty$-norm of the error, but is difficult to apply to systems with more than ten thousand dimensions with today’s computational resources since it involves computing the system’s reachability and observability Gramians. Proper orthogonal decomposition (POD) (also sometimes referred to as principal component analysis) is another widely used technique. POD is a data-driven model reduction technique that uses snapshots of the system’s state along trajectories to identify (via singular value decomposition) the low-dimensional subspace used for projection. This approach is particularly useful for nonlinear model order reduction and the reduction of extremely high-dimensional linear models, but which lacks rigorous \textit{a priori} guarantees. However, recent work has been done to augment POD approaches to also provide desirable theoretical properties \textit{a posteriori}, such as (Amsallem and Farhat 2012), which proposes a method to preserve stability properties of the system. Other useful approaches to model order reduction also exist, which are explored in detail in (Antoulas 2005; Benner et al. 2015). It is also important to note how model order reduction techniques differ from other methods for generating low-dimensional approximate models. In contrast to many data-driven system identification or machine learning methods (van Overschee and de Moor 1996; Benner et al. 2018a), projection-based model order reduction methods take advantage of the original high-fidelity model, retain the overall model structure, and also define an explicit, linear relationship between the original model states and the reduced order states. These properties are particularly useful for enabling system-theoretic error analysis to be performed, which this work leverages in the study of \textit{constrained} optimal control problems. Another option for generating lower dimensional approximate models of PDEs is to simply use a coarser spatial discretization. However, it has been shown that this is often not a good choice since model order reduction techniques can outperform in accuracy while also being orders of magnitude more computationally efficient.

Once an appropriate model order reduction technique has been applied to the high-dimensional model, the resulting low-dimensional model can be efficiently used within the context of existing control techniques (e.g. LQR/LQG, $H_\infty$, MPC). For example, an optimal feedback controller for PDE systems is synthesized by using POD to directly reduce the dimensionality of the Hamilton-Jacobi-Bellman equation in (Kunisch et al. 2004), and in (Leibfritz and Volkwein 2006) a robust output feedback controller is synthesized using nonlinear semidefinite programming techniques where the discretized PDE is also reduced using POD. The work by Li and Christofides (2008) also uses POD to generate reduced order models that are used to design LQR controllers for diffusion-convection-reaction processes. However, because the reduced order model is an approximation to
the high-dimensional model, it is not guaranteed that a controller designed with the ROM will provide the desired closed-loop behavior (e.g. stability). Thus, work has also been done to synthesize ROM-based controllers that also provide some form of performance guarantees. For example, POD reduction of the Hamilton-Jacobi-Bellman equation is again considered in (Alla et al., 2017), but additionally with an \textit{a priori} error analysis on the approximation of the value function in the reduced order problem with respect to the true solution in the high-dimensional case. Performance guarantees of ROM-based control have also been studied for the case of linear quadratic optimal control. Specifically, (Gubisch and Volkwein, 2017) review results on \textit{a posteriori} error analysis to quantify the sub-optimality of the LQR control due to model reduction errors, which utilize results from (Töltzscher and Volkwein, 2008), among others. This work particularly focuses on error analysis for POD-based model order reduction, as POD is a data-driven method that generally lacks \textit{a priori} error guarantees. Other model reduction approaches have also been used for linear quadratic optimal control, including balanced truncation (Antil et al., 2010) and the data-driven Loewner framework (Zhang, 2019). Both of these works also produce \textit{a posteriori} error results to bound how far the reduced order optimal control is from the true optimal control. Finally, (Benner et al., 2018b) use reduced order models for low-dimensional $H_\infty$ controller synthesis, and include a stability constraint on the high-dimensional closed-loop system to guarantee desired theoretical properties.

Model order reduction techniques have also been used to reduce the computational demand of \textit{receding horizon control} of infinite-dimensional systems. (Ghiglieri and Ulbrich, 2014) use POD for an optimal flow control problem, but do not provide stability guarantees. Asymptotic stability for a class of PDEs controlled by a POD-based MPC scheme is then presented in (Alla and Volkwein, 2015) which, similar to (Altmüller et al., 2010), determines a minimal stabilizing horizon. Work by (Altmüller, 2014) also considers model predictive control for PDE systems, and also uses horizon length to guarantee stability. While guaranteeing stability is an extremely desirable property, none of these proposed receding horizon control schemes consider problems with \textit{state constraints}, which are of practical interest in many engineering applications and is of particular interest in this thesis. Some ROM-based methods that consider \textit{constrained} optimal control problems have also been proposed, and these are reviewed in detail in Chapter [2].

1.1.2 Other Approaches

Several approaches for controlling infinite-dimensional systems exist beyond the use of model order reduction. Some are still model-based, but some are purely data-driven approaches. One alternative model-based approach is to design a controller based on potentially high-dimensional models, and then reduce the controller complexity \textit{a posteriori}. The advantage of such approaches is that they retain knowledge from the full model during the controller synthesis phase and still achieve online computational efficiency, but at the cost that the synthesis step becomes computationally intractable in extremely high-dimensional problems. This controller-order reduction approach is applied in
CHAPTER 1. INTRODUCTION

(Atwell et al., 2001; Atwell and King, 2004), which use POD to reduce the order of a high-dimensional linear controller. Singler and Batten (2009) apply balanced truncation methods to reduced the order of an infinite-dimensional dynamic controller instead of reducing the system dynamics and then designing the controller. In particular, they compare two techniques: one reduces the controller dynamics directly, and another reduces the Riccati equations that are used to compute the controller gains.

Machine learning or data-driven techniques have also been developed for controlling infinite-dimensional systems. While many approaches exist, here we mention just a few. Xie et al. (2012) and Xie et al. (2015) combine POD (also a data-driven procedure) with neural network modeling for nonlinear model predictive control applications. In another approach, data collected from computational fluid dynamics (CFD) simulations is used to develop a learned finite-dimensional model in Morton et al. (2018), which is then used to design an MPC controller to suppress vortex shedding. Bieker et al. (2020) extends ideas for learning models of fluid flows for MPC to more challenging flow control problems, and use input-output data rather than assuming full knowledge of the system’s state. An alternative learning-based approach is taken in Holl et al. (2020), which instead of learning a finite-dimensional model and then designing a controller, uses a differentiable PDE solver to directly learn a controller via end-to-end training. The main advantage of data-driven approaches is that they can be useful when modeling the system is challenging (e.g. for extremely chaotic fluid flows), yet in many cases these methods generate data through PDE model-based simulation and therefore still inherently rely on physics-based models. A disadvantage of machine learning-based approaches to modeling and control of PDE systems is that there are currently no easy ways to verify model accuracy except through empirical evidence. Data inefficiency can also be a major challenge, since simulations of PDE systems can be computationally expensive to perform. Nonetheless, there are many promising future avenues for data-driven control of infinite-dimensional systems, including those that combine data-driven methods to enhance existing model-based methods, including those that leverage model order reduction.

1.2 Summary of Contributions

Principled model order reduction techniques have proven to be effective at generating high-fidelity, but low-dimensional, models of PDE systems, and reduced order models have been applied in a variety of control applications. One particularly interesting application of reduced order models is to design model predictive controllers for solving constrained optimal control problems. The ROM’s high-fidelity enables better closed-loop performance, while the low-dimensionality satisfies practical computational requirements associated with the receding horizon nature of MPC. However, since ROMs are approximations to the high-dimensional discretized PDE models, approximation error must be accounted for in order to guarantee closed-loop performance.
In this thesis, we propose techniques for reduced order model predictive control of high-dimensional systems, such as those resulting from the semi-discretization of PDEs. The contributions of this thesis advance the state-of-the-art in several theoretical and practical ways:

1. In Chapters 2, 3, and 4, we propose a linear reduced order model predictive control (ROMPC) scheme that combines existing ideas from tube-based robust MPC with novel approaches to efficiently synthesize the controller in high-dimensional problem settings. The proposed approach exhibits several theoretical and practical advantages. First, both offline synthesis and online implementation are computationally efficient and can scale to extremely high-dimensional problems. Second, we consider ROMs generated through the broad and popular class of projection-based model reduction. Third, the approach considers output-feedback control, avoiding the need for limiting full-state knowledge assumptions. Fourth, in Chapter 5 we propose a computationally efficient method for quantifying the approximation errors associated with model order reduction, and use the results to provide theoretical guarantees on robust constraint satisfaction of the high-dimensional closed-loop system. This approach advances the state-of-the-art by simultaneously addressing the common issues of scalability, limitations on the type of ROM, reliance on impractical assumptions, and lack of guarantees for constrained optimal control.

2. We propose an extension of the linear ROMPC scheme to the nonlinear setting in Chapter 6. Nonlinear model order reduction is significantly more challenging than in the linear case, and in this work we leverage a linearization-based approach to generate nonlinear piecewise-affine reduced order model approximations of high-dimensional nonlinear models. The nonlinear ROMPC scheme enables output-feedback control by including a reduced order extended Kalman filter, and we use sequential quadratic programming to solve the optimal control problem.

3. We demonstrate the practicality and scalability of linear ROMPC in Chapter 5 where we use a computational fluid dynamics model with more than a million dimensions to model the aerodynamics of an unmanned aircraft. We then implement the linear ROMPC controller based on a coupled rigid-body/CFD model onboard a computationally-limited flight computer for real-time aircraft flight control. We also demonstrate the performance of the nonlinear ROMPC scheme in Chapter 7 where we use a finite element model with ten thousand dimensions to control a soft robot to perform dynamic trajectory tracking tasks.
Part I

Linear Reduced Order Model

Predictive Control
Chapter 2

Linear Reduced Order Model Predictive Control

Model predictive control (MPC) (Rawlings et al. 2017) is a powerful technique for addressing constrained optimal control problems, and has been widely studied in both theory and practice. MPC algorithms compute optimal control sequences in a receding horizon fashion by solving (typically online) constrained mathematical programs that optimize the system’s predicted future trajectory over a finite horizon. The predictive capability is obtained by embedding a mathematical model of the system’s dynamics as constraints in the optimization, and additional state and control constraints can also be included.

While MPC algorithms offer a number of advanced capabilities beyond classical control techniques, they are also burdened by unique challenges. One significant and fundamental challenge is the trade-off between the computational complexity of the optimization and the performance of the closed-loop system. In particular, the receding horizon nature of MPC requires solving the optimization problem in real-time, which precludes the use of overly complex dynamics models. Contrarily, high-fidelity dynamics models are required for good prediction accuracy and thus good closed-loop performance. Many early applications of MPC were for industrial process control where the system dynamics are slow, allowing for a slow controller update rate and thus the accommodation of higher fidelity models. More recently, increased computational capabilities and more advanced optimization algorithms have led to the application of MPC to systems with faster dynamics, such as automotive (Beal and Gerdes 2013) or aerospace (Eren et al. 2017) vehicles. However in these cases the dynamics models are generally low-dimensional.
2.1 Introduction

This chapter explores a computationally efficient MPC approach for linear systems whose models are high-dimensional. As mentioned in the introduction of this thesis, this problem setting is particularly motivated by cases where the high-dimensional model is defined by the semi-discretization (e.g. by a finite element or finite volume method) of a partial differential equation (PDE) model of an infinite-dimensional system. Such models arise in many engineering problems for modeling structural or fluid dynamics, heat flow dynamics, chemical reactions, electrochemical processes, and more. Example control applications involving these types of systems include soft robot control (Thieffry et al., 2019a; Tonkens et al., 2021), aircraft control (Lorenzetti et al., 2020), fluid flow control (Lassaux, 2002), chemical reaction processes (Agudelo et al., 2007), lithium-ion battery charging (Fan et al., 2018), among many others. In these settings, it is not uncommon for the model dimension of the semi-discretized PDE to reach from the thousands to millions, making standard model-based controller design (e.g. linear quadratic regulation, MPC) extremely challenging (Benner, 2004). In this context, one solution to the problem of computational complexity is to design controllers based on low-dimensional surrogate models. However, the use of low-fidelity models for controller design can lead to poor closed-loop performance, including sub-optimality, poor robustness to disturbances, or even instability. Therefore a low-dimensional but high-fidelity model is required.

Principled model order reduction techniques for deriving high-fidelity reduced order models (ROMs) from high-dimensional models have been extensively developed (Antoulas, 2005; Benner et al., 2015). These model reduction methods, such as balanced truncation and proper orthogonal decomposition (POD), have been successfully applied for model-based control of infinite-dimensional systems. For example, Marquez et al. (2013) use POD for controlling a nuclear reactor and McClellan et al. (2020) use POD for aircraft control. Performance of control systems based on reduced order models has also been analyzed from a theoretical perspective, including for the unconstrained linear quadratic optimal control problem (Gubisch and Volkwein, 2017; Antil et al., 2010; Zhang, 2019), as well as in receding horizon control (Ghiglieri and Ulbrich, 2014; Alla and Volkwein, 2015; Altmüller, 2014). While these works provide some theoretical results, such as bounds on the sub-optimality of the controller or guaranteed stability of the high-dimensional model, they do not consider optimal control problems with state constraints, which are of considerable practical interest due to their ability to enable safe control at the limits of performance. The unique challenge of using reduced order models for state-constrained MPC is that it is unclear a priori how model approximation error will affect closed-loop performance. In particular, model reduction error can lead to constraint violations by the controlled high-dimensional system, or even instability.

Previous work has begun to address the problem of state-constrained MPC with reduced order models. Early work (Hovland et al., 2008a,b) only considers soft state constraints, and therefore no guarantees on constraint satisfaction are provided. Other approaches provide hard constraint satisfaction guarantees by appropriately analyzing the model reduction error. In some work, certain
conditions are required to provide stability and constraint satisfaction guarantees, for example by restricting the analysis to ROMs derived using modal decomposition (Dubljevic et al. 2006; Dubljevic and Christofides 2006) or to problems where the constraints are only imposed on the reduced order state (Sopasakis et al. 2013). A less restrictive approach that considers projection-based ROMs and general state and control constraints is described by Löhnig et al. (2014). However, this work uses full-state feedback to ensure the model reduction error dynamics are stable, and for high-dimensional systems it is often not practical to assume that full-state measurements are available. This limitation is addressed by Kögel and Findeisen (2015a), which provides stability and constraint satisfaction guarantees while incorporating a reduced order state estimator into an output-feedback control scheme. Additionally, this work considers a robust control problem with bounded disturbances on the dynamics and measurement noise. The disadvantage of this approach is that the error analysis requires the solution to linear programs whose complexity is dependent on the dimension of the full order model. Thus for extremely high-dimensional problems (e.g. models with over several thousand degrees of freedom) this approach becomes less computationally efficient.

In summary, traditional MPC methods rely on computational tools that do not scale to extremely high-dimensional problems. While previous work has introduced reduced order models within the context of MPC to overcome online computational challenges, no work has simultaneously provided a method that (1) has theoretical performance guarantees, (2) is computationally efficient to synthesize, (3) is applicable to output feedback settings, and (4) can leverage general projection-based model order reduction methods; all are required or desirable properties for practical control settings.

In this chapter we present a reduced order model predictive control (ROMPC) scheme for controlling high-dimensional discrete-time linear systems. In particular, we consider a robust, output feedback, constrained optimal control problem where bounded disturbances affect the system dynamics and measured outputs, and where constraints are imposed on both states and control inputs. The proposed ROMPC scheme is defined by a constrained optimization problem that leverages a reduced order model, an ancillary feedback controller, and a reduced order state estimator. An efficient method for synthesizing the ancillary controller and estimator gains (based on $H_2$-optimal control techniques) is presented, and theoretical guarantees on the stability of the closed-loop system and robust constraint satisfaction are also provided. Additionally, the guarantees on constraint satisfaction require additional analysis techniques presented in Chapter 3 which quantify errors associated with the model order reduction, state estimation, and exogenous disturbances. Furthermore, discussion on the application of the proposed ROMPC scheme to setpoint tracking problems and also to high-dimensional continuous-time systems is presented. This work is based on several prior publications: (Lorenzetti et al. 2019; Lorenzetti and Pavone 2020a; Lorenzetti et al. 2021).

\footnote{https://github.com/StanfordASL/rompc}
2.2 Model Description and Problem Formulation

This section defines the linear discrete-time dynamics model, the mathematical formulation for the control problem, and the nominal reduced order model used to design the ROMPC scheme.

2.2.1 Linear Full Order Model

This work considers high-dimensional system models, such as from a finite approximation to an infinite-dimensional system (e.g. a semi-discretized PDE). Specifically, consider the linear, discrete-time, full order model (FOM):

\[
\begin{align*}
    x^f_{k+1} &= A^f x^f_k + B^f u_k + B^f w_k, \\
    y_k &= C^f x^f_k + v_k, \\
    z_k &= H^f x^f_k,
\end{align*}
\]

where the superscript \((\cdot)^f\) denotes a full order (high-dimensional) variable, \(x^f \in \mathbb{R}^{n_f}\) is the state, \(u \in \mathbb{R}^m\) is the control, \(y \in \mathbb{R}^p\) is the measured output, \(z \in \mathbb{R}^o\) are performance variables, \(w \in \mathbb{R}^{m_w}\) are disturbances acting on the system dynamics, and \(v \in \mathbb{R}^p\) is measurement noise. The disturbances \(w\) and \(v\) are assumed to be bounded by:

\[
w \in W, \quad v \in V,
\]

where \(W := \{ w \mid H_w w \leq b_w \}\) with \(H_w \in \mathbb{R}^{n_w \times m_w}\) and \(V := \{ v \mid H_v v \leq b_v \}\) with \(H_v \in \mathbb{R}^{n_v \times p}\) are convex polytopes.

Performance variable and control constraints are assumed to be defined by:

\[
z \in \mathcal{Z}, \quad u \in \mathcal{U},
\]

where \(\mathcal{Z} := \{ z \mid H_z z \leq b_z \}\) with \(H_z \in \mathbb{R}^{n_z \times o}\) and \(\mathcal{U} := \{ u \mid H_u u \leq b_u \}\) with \(H_u \in \mathbb{R}^{n_u \times m}\) are also convex polytopes. The following assumption is also made regarding the constraints and disturbances:

**Assumption 1.** The sets \(\mathcal{Z}, \mathcal{U}, W,\) and \(V\) are compact and \(\mathcal{Z}\) and \(\mathcal{U}\) contain the origin.

The performance variables \(z\) are defined because in many practical high-dimensional problems only a small subset of the states are relevant from a performance perspective. For example in rigid-body aircraft control problems leveraging CFD aerodynamics models (i.e. a fluid-structure interaction problem), the variables of interest are the aircraft rigid-body states (e.g. position, velocity, attitude) and not the fluid states.
2.2. MODEL DESCRIPTION AND PROBLEM FORMULATION

2.2.2 Constrained Optimal Control Problem

The control problem is to find an output feedback control scheme that guarantees that the FOM (2.1) satisfies the performance and control constraints (2.3) under any admissible disturbances, is stable, and also minimizes the cost function:

\[ J = \sum_{k=0}^{\infty} (x^f_k)^\top Q^f x^f_k + u_k^\top R u_k, \]  

where \( Q^f \in \mathbb{R}^{n^f \times n^f} \) is symmetric, positive semi-definite and \( R \in \mathbb{R}^{m \times m} \) is symmetric, positive definite. Note that we assume \( Q^f \) is positive semi-definite since a practical choice is \( Q^f = (H^f)^\top Q_z H^f \), where \( Q_z \) is a positive definite matrix that defines the cost with respect to the performance variable \( z \) (i.e. \( z^\top Q_z z \)) rather than the actual state \( x^f \).

2.2.3 Linear Reduced Order Model

Typical robust output feedback MPC schemes (Rawlings et al., 2017; Lorenzetti and Pavone, 2020b) would leverage the dynamics model (2.1) to solve the constrained optimal control problem. But with a high-dimensional model the resulting computational requirements may be excessive. An alternative is to use a reduced order model to design a computationally efficient controller.

In this chapter, we consider ROMs derived using projection-based model reduction methods, a general class of methods that includes popular techniques such as balanced truncation and proper orthogonal decomposition. These methods utilize either a Galerkin or a Petrov-Galerkin projection to project the model onto a reduced order subspace. In particular, a pair of basis matrices \( V, W \in \mathbb{R}^{n^f \times n^f} \) define the projection matrix \( P = V (W^\top V)^{-1} W^\top \), and the high-dimensional state can be projected to the reduced order state by \( x = (W^\top V)^{-1} W^\top x^f \) and can be approximately reconstructed by \( x^f \approx V x \).

The reduced order model is therefore defined using the Petrov-Galerkin projection by:

\[ x_{k+1} = Ax_k + Bu_k + B_w w_k, \]
\[ y_k = C x_k + v_k, \quad z_k = H x_k, \]  

Note \( V, W \) here are basis matrices, while script \( V, W \) define the disturbance bounds in (2.2).

If \( W = V \) the projection is referred to as a Galerkin projection, and some model reduction methods define the basis such that \( W^\top V = I \).
where \( x \in \mathbb{R}^n \) is the reduced order state and the ROM dynamics matrices are defined by:

\[
\begin{align*}
A &:= (W^TV)^{-1}W^TA^fV, \\
B &:= (W^TV)^{-1}W^TB^f, \\
B_w &:= (W^TV)^{-1}W^TB_{w}^f, \\
C &:= C^fV, \\
H &:= H^fV.
\end{align*}
\]

We also make the following assumption:

**Assumption 2.** The pair \((A, B)\) is controllable and the pairs \((A, C)\) and \((A, H)\) are observable.

As will be seen later, this assumption is required to guarantee performance of the ROMPC algorithm, and will also be used to bound errors induced by the model approximation in Chapter 3. Since this assumption is based on the lower dimensional ROM it is computationally easy to verify, but additional work must be done to provide conditions on the full order model or model reduction process that guarantee its satisfaction.

Additionally, the cost function (2.4) is approximated as:

\[
J \approx \sum_{k=0}^{\infty} x_k^TQx_k + u_k^TRu_k, \quad Q = V^TQ^fV.
\] (2.6)

### 2.3 Controller Definition

Model-based controller design with high-dimensional models is computationally challenging for both the controller synthesis stage (offline), as well as for online implementation. In this work, the ROM (2.5) is therefore leveraged to design an efficient reduced order model predictive control (ROMPC) scheme that consists of: (1) a reduced order linear feedback control law, (2) a reduced order state estimator, and (3) a reduced order optimal control problem (OCP). In particular, the proposed approach uses the reduced order OCP to optimize the trajectory of a nominal reduced order reference system:

\[
\begin{align*}
\bar{x}_{k+1} &= A\bar{x}_k + B\bar{u}_k, \\
\bar{z}_k &= H\bar{x}_k,
\end{align*}
\] (2.7)

where \( \bar{x} \in \mathbb{R}^n \), \( \bar{u} \in \mathbb{R}^m \), and \( \bar{z} \in \mathbb{R}^o \) are the state, control input, and performance output of the reference ROM, respectively. The feedback control law, in concert with the state estimator, then drives the real system to track this optimized reference trajectory.
2.3. CONTROLLER DEFINITION

2.3.1 Reduced Order Controller and State Estimator

The reduced order linear feedback control law that is used to control the full order system is given by:

\[ u_k = \bar{u}_k + K(\hat{x}_k - \bar{x}_k), \]  

(2.8)

where \((\bar{x}_k, \bar{u}_k)\) are the state and control values of the reference ROM, \(\hat{x}_k\) is the reduced order state estimate, and \(K \in \mathbb{R}^{m \times n}\) is the controller gain matrix. The reduced order state estimator is defined as:

\[ \hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y_k - C\hat{x}_k), \]  

(2.9)

where \(u_k\) and \(y_k\) are the control and (noisy) measurement from the full order system (2.1), and \(L \in \mathbb{R}^{n \times p}\) is the estimator gain matrix. Both gain matrices \(K\) and \(L\) are computed using Algorithm 3 in Section 2.5.

2.3.2 Reduced Order Optimal Control Problem

The optimal control problem (OCP) that defines the reference ROM trajectory optimizes over the finite-horizon sequences \(\bar{x}_k = \{\bar{x}_i|k\}_{i=k}^{k+N}\) and \(\bar{u}_k = \{\bar{u}_i|k\}_{i=k}^{k+N-1}\), where \(\bar{x}_i|k\) and \(\bar{u}_i|k\) denote the state and control variables at time step \(i\) in the OCP solved at time step \(k\). The OCP is given by:

\[
(\bar{x}^*_k, \bar{u}^*_k) = \arg\min_{\bar{x}_k, \bar{u}_k} \|\bar{x}_{k+N}|k\|^2 + \sum_{j=k}^{k+N-1} \|\bar{x}_j|k\|^2_Q + \|\bar{u}_j|k\|^2_R,
\]

subject to \(\bar{x}_{i+1}|k = A\bar{x}_i|k + B\bar{u}_i|k\), \(H\bar{x}_i|k \in \bar{Z}\), \(\bar{u}_i|k \in \bar{U}\), \(\bar{x}_{k+N}|k \in \bar{X}_f\), \(\bar{x}_k|k = \bar{x}_k\),

(2.10)

where \(i = k, \ldots, k+N-1\), the integer \(N\) is the planning horizon, \(\bar{x}_k\) is the current state of the reference ROM, and the constraint sets \(\bar{Z}\) and \(\bar{U}\) are tightened versions of the original constraints (2.3) such that \(\bar{Z} \subseteq Z\) and \(\bar{U} \subseteq U\). These tightened constraints are used to ensure robust constraint satisfaction and are defined in Section 2.4. The solution of the OCP (2.10) yields the optimized trajectory \((\bar{x}^*_k, \bar{u}^*_k)\), from which the reference ROM control input at time \(k\) is chosen as \(\bar{u}_k = \bar{u}^*_k|k\).

Since the next reference trajectory state \(\bar{x}_{k+1}\) is computed via (2.7) with input \(\bar{u}_k\), it also holds that \(\bar{x}_{k+1} = \bar{x}^*_{k+1}|k\).

The positive definite matrix \(P\) and the set \(\bar{X}_f\) define a terminal cost and terminal constraint that are designed to guarantee that the nominal reduced order reference trajectory with control \(\bar{u}_k = \bar{u}^*_k\) is exponentially stable such that \(\bar{x}_k \to 0\) and \(\bar{u}_k \to 0\), and that the OCP (2.10) is recursively feasible. Since methods for choosing \(P\) and \(\bar{X}_f\) to achieve exponential stability have been well established (Rawlings et al. 2017 Chapter 2, Lorenzetti et al. 2019 Section IV) they are not discussed here.
and instead we focus on leveraging those results in the context of ROMPC. However, for exponential stability we do note that the following assumption is required.

**Assumption 3.** The cost matrices $Q$ and $R$ are symmetric, positive definite.

In practice, the matrix $Q$ in (2.6) will be positive semi-definite if $Q^I$ is positive semi-definite. In this case the state cost in (2.6) can be made positive definite by using $Q + \gamma I$, where $\gamma \in \mathbb{R}$ is a small positive value.

### 2.3.3 Linear ROMPC Algorithm

The offline and online components of the ROMPC algorithm are now summarized, beginning with the offline controller synthesis procedure described in Algorithm 1. The overall approach is decomposed such that the computationally intensive components are performed in the offline phase. In particular, this includes the computation of error bounds $\Delta_z$ and $\Delta_u$ that are used to define the tightened constraints $\tilde{Z}$ and $\tilde{U}$.

**Algorithm 1** Linear ROMPC Synthesis (Offline)

1. **procedure** ROMPCSynthesis
2. Compute $K$, $L$ (Section 2.5, Algorithm 3)
3. Compute $\Delta_z$, $\Delta_u$ (Section 3.2, Algorithm 5)
4. Compute $\tilde{Z}$, $\tilde{U}$ (Section 2.5, Equation 2.12)
5. Compute $P$, $\tilde{X}_f$ (Section 2.3.2)

In the online portion of the ROMPC scheme (Algorithm 2) the reference ROM state $\bar{x}_k$ and reduced order state estimate $\hat{x}_k$ are initialized at time $k = 0$ to be $\bar{x}_0 = \hat{x}_0 = 0$, and it is assumed that the ROMPC controller takes over at time $k = k_0$. For times $0 \leq k < k_0$ it is assumed that a “startup” controller is used to give the state estimator and reference ROM enough time to converge from their initialized values. In practice, the startup controller could be as simple as a linear feedback controller designed around a steady state operating point that is sub-optimal, but conservative (i.e. with respect to the state constraints). Alternatively, in some cases the startup controller could even be a simple open-loop sequence. At time $k = k_0$ the ROMPC controller then takes over to drive the system to different operating points for better performance, while guaranteeing constraint satisfaction. It is also assumed that at time $k_0$ the OCP (2.10) is feasible.

For additional clarity, the architecture of the ROMPC control scheme (for times $k \geq k_0$) is shown in Figure 2.1. This diagram highlights the fact that there are two systems being controlled: the FOM (2.1) is the physical plant, and the ROM is used as a simulated reference system. The two systems are connected only by the controller (2.8), which is essentially trying to drive the FOM to track the optimized reduced order reference trajectory.

Additionally, when $k < k_0$ and the startup controller is applied to the FOM, the reference ROM
2.4 Closed-Loop Performance Guarantees

This section discusses two key properties of the closed-loop system’s performance, namely robust constraint satisfaction and stability. In the context of this work, a property is considered robust if...
it holds even in the presence of reduced order model approximation errors, state estimation errors, and admissible bounded disturbances (2.2).

Recall that the reduced order OCP (2.10) optimizes the trajectory of the reference ROM under the constraints that $\bar{z} \in \bar{Z}$ and $\bar{u} \in \bar{U}$. The control law (2.8) and state estimator (2.9) are then used to make the full order system (2.1) track this trajectory. However, perfect tracking is not possible due to disturbances, model reduction errors, and state estimation errors. Thus, choosing $\bar{Z} = Z$ and $\bar{U} = U$ will not guarantee constraint satisfaction for the full order system. A tube-based approach (Rawlings et al., 2017, Chapter 3.5) is therefore employed to robustly guarantee constraint satisfaction. In this approach, the worst-case tracking errors are quantified (using the approaches presented in Chapter 3) and used to define “tubes” around the nominal trajectories of $\bar{z}$ and $\bar{u}$. Since these tubes are guaranteed to contain the actual trajectories of the system, requiring that the entire tube satisfy the constraints will guarantee that the actual trajectory will also satisfy the constraints. This is implemented in practice by choosing $\bar{Z}$ and $\bar{U}$ to be tightened versions of the original constraints, as represented visually in Figure 2.2.

![Figure 2.2: Robust constraint satisfaction can be guaranteed by using error tubes and constraint tightening. The trajectory of the reference ROM (2.7), which is controlled by the OCP (2.10) and satisfies the tightened constraints $\bar{z} \in \bar{Z}$, is shown in pink. The performance variables, $z$, of the full order system (2.1) are shown in blue, and track $\bar{z}$ with bounded error. The tightened constraint $\bar{Z}$ guarantees that the entire tube, and thus $z$, satisfies the constraint $z \in Z$.](image)

2.4.1 Closed-Loop Error Dynamics

Quantifying the worst-case errors $\delta_z := z - \bar{z}$ and $\delta_u := u - \bar{u}$ requires an analysis of the dynamics of the state reduction error $e := x^f - V\bar{x}$ and control error $d = \dot{x} - \bar{x}$. A joint error state $\epsilon$ is defined as $\epsilon := [e^T, d^T]^T$ such that the closed-loop error dynamics are given by:

$$\epsilon_{k+1} = A_\epsilon \epsilon_k + B_\epsilon r_k + G_\epsilon \omega_k,$$

(2.11)

where:

$$A_\epsilon = \begin{bmatrix} A^f & B^fK \\ LC^f & A + BK - LC \end{bmatrix}, \quad B_\epsilon = \begin{bmatrix} P_\perp A^f V & P_\perp B^f \\ 0 & 0 \end{bmatrix}, \quad G_\epsilon = \begin{bmatrix} B_w^f & 0 \\ 0 & L \end{bmatrix}.$$
and with \( P_\perp = I - V(W^TV)^{-1}W^T \), \( r = [\bar{x}^T, \bar{u}^T]^T \), and \( \omega = [w^T, v^T]^T \). Note that \( \delta_z = H^t e \) and \( \delta_u = K d \) can be considered outputs of this error system, and the inputs to the system are the disturbances, \((w, v)\), and the trajectory of the reference ROM, \((\bar{x}, \bar{u})\). The influence of the trajectory \((\bar{x}, \bar{u})\) on the error is important, and in particular it should be noted that the error will be bounded only if the trajectory \((\bar{x}, \bar{u})\) is bounded.

### 2.4.2 Closed-Loop Robust Constraint Satisfaction

The closed-loop error dynamics (2.11) provide a means for analyzing the worst-case tracking errors \( \delta_z \) and \( \delta_u \). Of particular interest is how the tracking errors could lead to constraint violations. Since the constraint sets \( Z \) and \( U \) defined in (2.3) are polytopes expressed by a set of half-spaces, from the definition of the state reduction error \( e \) and control error \( d \) it follows that:

\[
\begin{align*}
z &\in Z \iff H_z(\bar{z} + \delta_z) \leq b_z, \\
u &\in U \iff H_u(\bar{u} + \delta_u) \leq b_u.
\end{align*}
\]

With a set of worst-case bounds \( \Delta_z \in \mathbb{R}^{n_z} \) and \( \Delta_u \in \mathbb{R}^{n_u} \) on the tracking error, such that \( H_z \delta_z \leq \Delta_z \) and \( H_u \delta_u \leq \Delta_u \), the OCP constraint sets \( \bar{Z}, \bar{U} \) can be defined as:

\[
\begin{align*}
\bar{Z} &:= \{ \bar{z} | H_z \bar{z} \leq b_z - \Delta_z \}, \\
\bar{U} &:= \{ \bar{u} | H_u \bar{u} \leq b_u - \Delta_u \},
\end{align*}
\]

which are tightened versions of \( Z \) and \( U \). With these tightened constraint sets, the nominal trajectory computed by the OCP (2.10) will account for the tracking error and therefore ensure constraint satisfaction. Two methods for computing the bounds \( \Delta_z \) and \( \Delta_u \) are presented in Chapter 3.

**Lemma 1 (Robust Constraint Satisfaction).** Suppose that at time \( k_0 \) the optimal control problem (2.10) is feasible and that \( H_z \delta_z,k \leq \Delta_z \) and \( H_u \delta_u,k \leq \Delta_u \) for all \( k \geq k_0 \). Then, under the proposed control scheme the full order system will robustly satisfy the constraints (2.3) for all \( k \geq k_0 \).

**Proof.** Since the optimal control problem is designed to be recursively feasible and is assumed to be feasible at time \( k_0 \), then the simulated reduced order system trajectory will satisfy \( \bar{z}_k \in \bar{Z} \) and \( \bar{u}_k \in \bar{U} \) for all \( k \geq k_0 \). Therefore, since \( H_z \bar{z}_k \leq b_z - \Delta_z \) and \( H_z \bar{z}_k \leq \Delta_z \) for all \( k \geq k_0 \), then \( H_z \bar{z}_k \leq b_z - H_z \bar{z}_k \leq \Delta_z \) for all \( k \geq k_0 \), and finally \( H_z \bar{z}_k \leq b_z \) for all \( k \geq k_0 \). The same analysis can also be applied for \( u_k \).

### 2.4.3 Closed-Loop Stability

In addition to robustly satisfying the constraints, the full order system (2.1) must be stable. In the disturbance free case (i.e. when \( w = 0 \) and \( v = 0 \)), the ROMPC scheme defined in Section 2.3 is guaranteed to make (2.1) stable and converge asymptotically to the origin. However, guaranteed
convergence to the origin is not possible when unpredictable disturbances affect the system. But, if the disturbances $w$ and $v$ are bounded, the controlled full order system can be shown to asymptotically converge to a compact set containing the origin (whose size depends on the disturbance bounds) under certain assumptions.

**Theorem 1 (Robust Stability).** Suppose that $A_c$ is Schur stable, that the OCP (2.10) is feasible at some time $k_0$, and that the OCP drives the reference ROM trajectory to the origin exponentially fast such that $\|\bar{x}_k\| \leq M\gamma^k$ for all $k \geq k_0$ and for some values $M > 0$ and $\gamma \in (0, 1)$. Then, the closed-loop system robustly and asymptotically converges to a compact set containing the origin.

**Proof.** Similarly to the closed-loop error dynamics (2.11), the closed-loop dynamics of the full order system and reduced order state estimator can be written:

$$
\xi_{k+1} = A_c\xi_k + B_c r_k + G_c \omega_k, \quad B_c = \begin{bmatrix} -BfK & Bf \\ -BK & B \end{bmatrix},
$$

where $\xi = [(x^f)^\top, \dot{x}^\top]^\top$. Therefore, writing the solution of this dynamics recursion in the form:

$$
\xi_k = A_c^k \xi_0 + \sum_{j=0}^{k-1} A_c^{k-1-j}(B_c r_j + G_c \omega_k),
$$

we can analyze the norm of $\xi$. Since $A_c$ is Schur stable there is guaranteed to exist values $M_1 \geq 1, \gamma_1 \in (0, 1)$ such that $\|A_c^k\| \leq M_1\gamma_1^k$ for all $k \geq 0$. Additionally, since $\mathcal{W} \times \mathcal{V}$ is a compact set by Assumption 1, there exists a constant $C$ such that $\|G_c \omega_k\| \leq C$ for all $k$. Therefore we can write:

$$
\|\xi_k\| \leq M_1\gamma_1^k\|\xi_0\| + \sum_{j=0}^{k-1} M_1\gamma_1^{k-1-j}(\|B_c r_j\| + C).
$$

Further, from the exponential stability of the simulated reduced order system there exists values $M_2 > 0$ and $\gamma_2 \in (0, 1)$ such that if the initial OCP is feasible then $\|r_k\| \leq M_2\gamma_2^k\gamma$ for all $k \geq 0$. Defining $\gamma = \max\{\gamma_1, \gamma_2\} \in (0, 1) \text{ and noting that } \sum_{j=0}^{k-1} \gamma^{k-1-j} \leq \frac{1}{1-\gamma}$ we can simplify the previous expression to:

$$
\|\xi_k\| \leq M_1\|\xi_0\|\gamma^k + M_1 M_2\|B_c\|\gamma^k + \frac{M_1 C}{1-\gamma}.
$$

Since $|\gamma| < 1$, then in the limit $\lim_{k \to \infty} \gamma^k = 0$ and $\lim_{k \to \infty} k\gamma^{k-1} = 0$ and therefore:

$$
\lim_{k \to \infty} \|\xi_k\| \leq \frac{M_1 C}{1-\gamma}.
$$

\[^4\text{Computation of this set is theoretically possible Kolmanovsky and Gilbert (1998), but is computationally intractable for high-dimensional systems and often not practically useful.}\]
Therefore the closed-loop system asymptotically converges to the compact set:

$$\Xi := \{\xi \mid \|\xi\| \leq \frac{M_1 C}{1-\gamma}\}.$$ 

\[ \square \]

**Corollary 1** (Stability). Suppose the conditions from Theorem 1 hold. Then, in the disturbance free case where \( \mathcal{W} = \{0\} \) and \( \mathcal{V} = \{0\} \), the origin is asymptotically stable for the closed-loop system.

**Proof.** In the absence of disturbances, the constant \( C \) in the proof of Theorem 1 can be chosen to be \( C = 0 \). Therefore:

$$\lim_{k \to \infty} \|\xi_k\| = 0,$$

and the closed-loop system asymptotically converges to the origin. \[ \square \]

The requirement that \( A_\epsilon \) is Schur stable is natural, and is satisfied by properly choosing the gain matrices \( K \) and \( L \). In fact, without the model order reduction step (i.e. \( A = A^f, B = B^f, C = C^f \)) the separation principle guarantees the stability of the error dynamics by the Schur stability of \( A + BK \) and \( A - LC \). Unfortunately, with the use of a reduced order model the separation principle cannot be applied since \( A^f \neq A \), and therefore no such guarantee on the stability of \( A_\epsilon \) can be made. A methodology for synthesizing the controller gains \( K \) and \( L \) such that \( A_\epsilon \) is Schur stable is presented in Section 2.5.

### 2.4.4 Computational Challenges

Tube-based approaches to robust MPC are simple to implement, but defining the tubes used for constraint tightening can be challenging. One common approach for linear robust MPC is to define a tube in the state space by computing a robust positively invariant (RPI) set for the system’s error dynamics (Mayne et al., 2006; Kögel and Findeisen, 2017; Lorenzetti and Pavone, 2020b). To compute an RPI set, bounds on the reference trajectory and external disturbances are assumed to be given by:

$$r_k \in \Delta_r, \quad \omega_k \in \Delta_\omega, \quad \forall k,$$

where \( \Delta_r \) and \( \Delta_\omega \) are compact and contain the origin in their interior. Then, an RPI set \( \mathcal{R} \) for the Schur stable closed-loop error dynamics (2.11) is a set that for all \( \epsilon_k \in \mathcal{R} \), and for all \( r_k \in \Delta_r \) and \( \omega_k \in \Delta_\omega \), the state \( \epsilon_{k+1} \) also satisfies \( \epsilon_{k+1} \in \mathcal{R} \). In other words, the RPI set \( \mathcal{R} \) defines a tube for the error system such that if the error starts within the tube then it will remain within the tube for all time, for any admissible disturbance.

Unfortunately, computation of RPI sets is notoriously difficult (Schulze Darup and Teichrib, 2019; Kolmanovsky and Gilbert, 1998; Rakovic et al., 2005; Lorenzetti and Pavone, 2020b) and scales poorly with the dimensionality of the system. Computing RPI sets for systems with as little as
one hundred dimensions can be numerically challenging with today’s algorithms and computational resources, let alone systems with thousands to millions of dimensions. The use of RPI sets would also be overly conservative in this application, since the definition of $\mathcal{R}$ assumes that $r_k$ can be chosen in a worst-case fashion at each time step. In fact, $r_k$ is defined by the reference ROM trajectory, which follows the dynamics (2.7). We therefore develop a novel and scalable approach for computing the bounds $\Delta_z$ and $\Delta_u$ that define the tubes for tightening the ROMPC constraints (2.12). This approach is defined in detail in Chapter 3.

Another significant computational challenge when synthesizing the ROMPC controller is computing the gain matrices $K$ and $L$ that ensure $A_\epsilon$ is Schur stable. Classical approaches to controller design, such as LQR, $\mathcal{H}_2$, or $\mathcal{H}_\infty$-control, require numerical methods that do not scale well to extremely high-dimensional systems. We therefore develop an approach to efficiently synthesize these gains that again leverages model order reduction, which is described next in Section 2.5.

### 2.5 Controller Synthesis

In this section, we present a methodology for synthesizing the gain matrices $K$ and $L$ that define the control law (2.8) and state estimator (2.9). The goal of the synthesis method is to find a set of gains that will ensure that the error dynamics are stable (i.e. $A_\epsilon$ is Schur stable), that keep the tracking errors $\delta_z$ and $\delta_u$ small, and that can be computed efficiently even for extremely high-dimensional problems.

In principle, stability is straightforward to verify since $A_\epsilon$ is Schur stable if all of its eigenvalues satisfy $|\lambda| < 1$. Assuming the full order system is sparse\footnote{Finite element/volume methods for semi-discretizing PDEs typically lead to sparse models.}, the largest magnitude eigenvalue can be computed with relative efficiency even in high-dimensional settings. The metric chosen to benchmark the tracking performance is the $\mathcal{H}_2$ system norm of the closed-loop error dynamics (2.11) with outputs:

$$\tilde{z}_k = H_\epsilon r_k, \quad H_\epsilon = \begin{bmatrix} W_z H_f & 0 \\ 0 & W_u K \end{bmatrix},$$

(2.13)

where $W_z$ and $W_u$ are user defined weighting matrices. Note that the outputs $\tilde{z} = [(W_z \delta_z)^T, (W_u \delta_u)^T]^T$ are simply the weighted tracking errors. Recall that the inputs to the closed-loop error dynamics system are the combined model reduction errors and disturbances, which can be concatenated as $\tilde{w} := [r^T, \omega^T]^T$.

The $\mathcal{H}_2$-norm of a closed-loop system with inputs $\tilde{w}$ and output $\tilde{z}$, denoted by $\|G\|_{\mathcal{H}_2}$ (where $G$ is the closed-loop transfer function from inputs $\tilde{w}$ to outputs $\tilde{z}$), satisfies the property:

$$\|\tilde{z}\|_{L_\infty} \leq \|G\|_{\mathcal{H}_2} \|\tilde{w}\|_{L_2},$$
where \( \| \tilde{z} \|_{L_\infty} = \sup_{k \geq 0} \| z_k \|_{\infty} \) and \( \| \tilde{w} \|_{L_2} = \sqrt{\sum_{k=0}^{\infty} \| \tilde{w}_k \|_2^2} \). In other words, the \( \mathcal{H}_2 \)-norm of the closed-loop system provides an upper bound on the maximum amplitude of the output \( \tilde{z} \) for inputs \( \tilde{w} \) of bounded energy. Since the outputs \( \tilde{z} \) in this problem correspond to the tracking errors \( \delta z \) and \( \delta u \), minimizing the \( \mathcal{H}_2 \)-norm can be used as a surrogate for minimizing the worst-case tracking error.

Controller synthesis for minimizing the closed-loop \( \mathcal{H}_2 \)-norm is commonly referred to as \( \mathcal{H}_2 \)-optimal control (Zhou et al., 1996; Petersson, 2013).

### 2.5.1 \( \mathcal{H}_2 \)-Optimal Controller Synthesis

The \( \mathcal{H}_2 \)-optimal controller synthesis problem is now written in a more standard form. First, the closed-loop dynamics (2.11) are algebraically rearranged into the equivalent system:

\[
\begin{align*}
\epsilon_{k+1} &= \tilde{A}\epsilon_k + \tilde{B}\tilde{u}_k + \tilde{B}_w\tilde{w}_k, \\
\tilde{y}_k &= \tilde{C}\epsilon_k + \tilde{D}_y\tilde{w}_k, \\
\tilde{z}_k &= \tilde{H}\epsilon_k + \tilde{D}_z\tilde{u}_k, \\
\tilde{u}_k &= \tilde{K}\tilde{y}_k,
\end{align*}
\tag{2.14}
\]

where:

\[
\tilde{A} := \begin{bmatrix} A^f & 0 \\ 0 & A \end{bmatrix}, \quad \tilde{B} := \begin{bmatrix} 0 & B^f \\ I & B \end{bmatrix}, \quad \tilde{C} := \begin{bmatrix} C^f & -C \\ 0 & I \end{bmatrix}, \quad \tilde{H} := \begin{bmatrix} W_z H^f & 0 \\ 0 & 0 \end{bmatrix}, \quad \tilde{K} := \begin{bmatrix} L & 0 \\ 0 & K \end{bmatrix},
\]

and with noise matrices:

\[
\tilde{B}_w := \begin{bmatrix} P_{\perp} A^f V & P_{\perp} B^f & B_w^f & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \tilde{D}_{yw} := \begin{bmatrix} 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \tilde{D}_{zu} := \begin{bmatrix} 0 & 0 \\ 0 & W_u \end{bmatrix}.
\]

With the requirement that \( \tilde{u} = \tilde{K}\tilde{y} \), it can be seen that the problem of choosing \( K \) and \( L \) to minimize the \( \mathcal{H}_2 \)-norm of the closed-loop error dynamics can be cast as solving an \( \mathcal{H}_2 \)-optimal control problem with a structured, static output feedback controller. This problem is compactly expressed as:

\[
\min_{K,L} \| G \|_{\mathcal{H}_2}, \tag{2.15}
\]

where \( G \) is the closed-loop transfer function of (2.14) (or equivalently of (2.11)) with inputs \( \tilde{w} \) and outputs \( \tilde{z} \).

There exist several approaches for solving such problems, including optimization methods based on convex approximations or non-convex optimization (Sadabadi and Peaucelle, 2016). However, for extremely high-dimensional problems these approaches may not be computationally tractable. One alternative to address the computational issues arising from high-dimensional problems is to approximate the objective function by a surrogate that is more efficient to evaluate. This approach is
used for \( H_\infty \)-optimal controller synthesis by Mitchell and Overton (2015) and Benner et al. (2018c). Of particular interest, Benner et al. (2018c) minimize the \( H_\infty \) norm of a reduced order system instead of the original high-dimensional one.

We propose to use a similar approach to Benner et al. (2018c): approximately solve the original problem by minimizing the \( H_2 \)-norm of a reduced order error system that approximates (2.14). Consider a reduced order approximation defined by:

\[
\begin{align*}
\epsilon_{r,k+1} &= \hat{A}_r \epsilon_{r,k} + \hat{B}_r \hat{u}_k + \hat{B}_{\tilde{w},r} \tilde{w}_k, \\
\tilde{y}_{r,k} &= \hat{C}_r \epsilon_{r,k} + \hat{D}_{\tilde{z} \tilde{u}} \hat{u}_k, \\
\tilde{z}_{r,k} &= \hat{H}_r \epsilon_{r,k} + \hat{D}_{\tilde{z} \tilde{u}} \hat{u}_k, \\
\hat{u}_k &= \hat{K} \tilde{y}_k,
\end{align*}
\]

where \( \hat{A}_r = (\hat{W}^\top \hat{V})^{-1} \hat{W}^\top \hat{A} \hat{V}, \hat{B}_r = (\hat{W}^\top \hat{V})^{-1} \hat{W}^\top \hat{B}, \hat{B}_{\tilde{w},r} = (\hat{W}^\top \hat{V})^{-1} \hat{W}^\top \hat{B}_{\tilde{w}}, \hat{C}_r = \hat{C} \hat{V}, \hat{H}_r = \hat{H} \hat{V}, \) and \( \hat{V} \) and \( \hat{W} \) are the reduced order basis that define a Petrov-Galerkin projection. The approximate \( H_2 \)-optimal control problem is:

\[
\min_{K,L} \| G_r \|_{H_2},
\]

where \( G_r \) is the closed-loop transfer function of the reduced order closed-loop error dynamics system (2.16). This problem is much more computationally tractable and is generally amenable to the previously mentioned techniques for computing structured, static output feedback controllers. Additionally, the basis matrices used to define the ROM (2.5) can be directly reused in (2.16) rather than performing an additional model reduction procedure. In Section 2.5.2 we show that this leads to further simplifications of the control synthesis problem.

### 2.5.2 Reduced Order Riccati Method

By reusing the basis matrices \( V \) and \( W \) that define the ROM (2.5) to define (2.16), the \( H_2 \)-optimal control problem (2.17) reduces to a classical \( H_2 \)-optimal control problem that only requires the solution of two Riccati equations. Specifically, the matrices \( \hat{V} \) and \( \hat{W} \) are chosen as:

\[
\hat{V} = \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}, \quad \hat{W} = \begin{bmatrix} W & 0 \\ 0 & I \end{bmatrix}.
\]
In this case the reduced order closed-loop error system (2.16) can be algebraically re-written as:

\[
\begin{align*}
e_{r,k+1} &= A e_{r,k} + B u_{r,k} + \begin{bmatrix} B_w & 0 \end{bmatrix} \omega_k, \\
y_{r,k} &= C e_{r,k} + \begin{bmatrix} 0 & I \end{bmatrix} \omega_k, \\
\tilde{z}_{r,k} &= \begin{bmatrix} W_z H \\ 0 \end{bmatrix} e_{r,k} + \begin{bmatrix} 0 & W_u \end{bmatrix} u_{r,k}, \\
d_{k+1} &= A d_k + B u_{r,k} + L(y_{r,k} - C d_k), \\
u_{r,k} &= K d_k,
\end{align*}
\] (2.18)

where \(e_r\) is the reduced order approximation of the state reduction error \(e\).

There are two interesting things to note about these dynamics. First, they are exactly the dynamics that would be used for controller synthesis in standard problems where there is no model order reduction and only the reduced order system was considered (i.e. \( \mathcal{A}^f = \mathcal{A}, \mathcal{B}^f = \mathcal{B}, \mathcal{C}^f = \mathcal{C}, \mathcal{H}^f = \mathcal{H} \)). In other words, this is a classical \(\mathcal{H}_2\)-optimal controller synthesis problem applied to the ROM. Second, for the simplified choice of \(\tilde{V}\) and \(\tilde{W}\) the solution to the optimization problem (2.17) is:

\[
\begin{align*}
K &= -(B^T X B + R)^{-1} B^T X A, \\
L &= A Y C^T (C Y C^T + I)^{-1},
\end{align*}
\] (2.19)

where \(X\) and \(Y\) are solutions of the discrete-time algebraic Riccati equations:

\[
\begin{align*}
X &= A^T X A - A^T X B (B^T X B + R)^{-1} B^T X A + Q_z, \\
Y &= A Y A^T - A Y C^T (C Y C^T + I)^{-1} C Y A^T + Q_w,
\end{align*}
\] (2.20)

where \(R = W_u^T W_u, Q_z = H^T W_z^T W_z H, Q_w = B_w B_w^T + \gamma I\) with \(\gamma > 0\), and \(R\) is assumed to be full rank. Note that the identity matrix in the choice of \(Q_w\) is a regularization term that is added to indirectly account for the model reduction disturbances. This is required because the use of \(V\) and \(W\) to simplify the \(\mathcal{H}_2\)-optimal synthesis problem leads to the model reduction disturbance terms to be completely ignored in (2.18).

The reduced order Riccati method is summarized in Algorithm 3. Note that no guarantees on the stability of \(\mathcal{A}_e\) are provided since the reduced order system is used to approximate the true problem. Therefore it is imperative to check the stability \textit{a posteriori}. Options to consider if Algorithm 3 fails to stabilize \(\mathcal{A}_e\) are discussed at the end of Section 2.5.3.
CHAPTER 2. LINEAR REDUCED ORDER MODEL PREDICTIVE CONTROL

Algorithm 3 Controller Gain Synthesis

1: procedure REDORDERRICCATIMETHOD(...) 
2: \( R \leftarrow W_u^\top W_u \) (\( R \) full rank) 
3: \( Q_z \leftarrow H^\top W_z^\top W_z H \) 
4: \( Q_w \leftarrow B_w B_w^\top + \gamma I \) 
5: \( X, Y \leftarrow \) solve Riccati equations (2.20) 
6: \( K = -(B^\top X B + R)^{-1} B^\top X A \) 
7: \( L = AY C^\top (CY C^\top + I)^{-1} \) 
8: if \( A_r \) is Schur stable then return \( K, L \)

2.5.3 Controller Synthesis Results

The reduced order Riccati method (Algorithm 3) solves the approximate \( H_2 \)-optimal control problem (2.17) as a surrogate for the full order problem (2.15). To benchmark the effectiveness of this method it is compared against an alternative technique that attempts to directly solve the original full order \( H_2 \)-optimal control problem (2.15). In particular, a quasi-Newton BFGS optimization method implemented in the package HIFOO (Arzelier et al., 2011) is used, which first minimizes the spectral radius until a stabilizing controller is found, and then minimizes the \( H_2 \)-norm. Since the \( H_2 \)-norm is non-convex and convergence to a global minimum is not guaranteed, HIFOO uses multiple random initial conditions and returns the best solution.

For the comparison, HIFOO\(^6\) is first used without any initial solution guess, it is then applied a second time with a warm-start by the solution from Algorithm 3. Results for each of the three methods (Riccati, HIFOO, and HIFOO with Riccati warm-start) are presented in Table 2.1 with the resulting computation times presented in Table 2.2. For these results, several example problems are considered. The “Small and Large Synthetic” systems are academic problems discussed in (Lorenzetti et al., 2021), and the remaining problems are derived from practical engineering settings, and are described in detail in Chapter 4. Specifically, the practical engineering applications include a distillation column, a chemical reaction process, and a heat transfer problem.

In Table 2.1 the values \( R_{H_2} \) are defined as:

\[
R_{H_2} := \frac{\|\Sigma - \Sigma_r\|_{H_2}}{\|\Sigma\|_{H_2}},
\]

(2.21)

where \( \Sigma \) is the transfer function from \( u \) to \( z \) of the full order model (2.1) and \( \Sigma_r \) is the transfer function from \( \bar{u} \) to \( \bar{z} \) of the reduced order model (2.7). This value can therefore be used as a metric for how good the reduced order model approximates the true model with respect to the \( H_2 \)-norm, where a small value of \( R_{H_2} \) indicates a good approximation. The remaining values in Table 2.1 are the values of the \( H_2 \)-norm, \( \|G\|_{H_2} \), of the full order closed-loop error system (2.14) defined with the

\(^6\)A modified version of the open-source HIFOO code is included in the repository accompanying this work, which is simplified to only solve the controller synthesis problem considered in this work and modified to handle discrete-time systems.
synthesized gains $K$ and $L$ using either HIFOO or HIFOO with Riccati warm-start, and expressed as a fraction of the $H_2$-norm resulting from the reduced order Riccati method.

These results show that the reduced order Riccati method performs comparably to direct optimization over a variety of problems. Additionally, from the computation times listed in Table 2.2 it can be seen that the reduced order Riccati method drastically outperforms in efficiency. In general, the Riccati method only depends on the ROM dimension and does not depend on the dimension of the full order model, which makes it scalable to even extremely high-dimensional ROMPC applications assuming $n \ll n_f$.

Table 2.1: Controller synthesis results comparing the reduced order Riccati method (Algorithm 3 with $\gamma = 0.001$) against a quasi-Newton approach applied directly to minimizing (2.15) (HIFOO), as well as a combined method where the Riccati method is used to warmstart HIFOO (HIFOO+Riccati). The values $R_{H_2} = \| \Sigma - \Sigma_r \|_{H_2}/\| \Sigma \|_{H_2}$ are used to quantify the model reduction error. The values corresponding to the different methods are the $H_2$-norms of the closed-loop error system, $\| G \|_{H_2}$, expressed as a ratio corresponding to the value computed using the reduced order Riccati method. For example a value of 0.95 implies a 5% smaller value of $\| G \|_{H_2}$ with respect to the Riccati method (Algorithm 3).

Table 2.2: Controller synthesis computation times for synthesizing the controll gains $K$ and $L$ using the reduced order Riccati method (Riccati), a quasi-Newton approach applied directly to minimizing (2.15) (HIFOO), as well as a combined method where the Riccati method is used to warmstart HIFOO (HIFOO+Riccati). The reduced order Riccati method is much faster because it is only dependent on the dimension $n$ of the reduced order model, and not the dimension $n_f$ of the FOM.

The primary disadvantage of the reduced order Riccati method is the lack of guarantees that the resulting controller will result in the closed-loop system being asymptotically stable. While good performance has been empirically demonstrated, similar results may not hold if the reduced order models are not sufficiently good approximations. In the case that the reduced order Riccati method

\footnote{The ROM dimension is a user-selected value, typically chosen with the aid of heuristics that are useful in the context of the specific model reduction method for quantifying model reduction error.}
fails to stabilize the full order system, two additional options should be considered. First, the accuracy of the reduced order model should be verified (e.g. by simulation). This is a worthwhile first step because even if a set of stabilizing gains $K$ and $L$ could be identified with a different approach, poor model approximation could still lead to large tracking errors and degraded performance for the closed-loop system. Second, for moderately-sized problems an optimization problem that incorporates the full order closed-loop error system (2.14) could be solved.

As can be seen from the previous results, directly optimizing the $\mathcal{H}_2$-norm of the full order closed-loop error system becomes challenging with high-dimensional problems. In particular, HIFOO becomes computationally challenging in these cases because each gradient evaluation of the $\mathcal{H}_2$-norm, $\|G\|_{\mathcal{H}_2}$, with respect to the controller gains $K$ and $L$ involves solving full order Lyapunov equations. A slightly more practical method, that would still guarantee stability of the closed-loop system, is to consider an optimization problem whose objective is the reduced order approximate $\mathcal{H}_2$-norm, $\|G_r\|_{\mathcal{H}_2}$, but is subjected to a stability constraint on the full order system. This is the approach proposed for reduced order $\mathcal{H}_\infty$ controller synthesis by [Benner et al. 2018c]. The advantage of this approach is that the full order system is only used to compute the spectral radius (i.e. to check stability) or compute a gradient of the spectral radius, and both of these operations can leverage efficient sparse eigenvalue methods.

### 2.6 Constant Setpoint Tracking

The ROMPC scheme presented in Section 2.3 can also be extended to setpoint tracking problems, where a subset of the performance variables $z$ are controlled to track a constant reference signal. However special care must be taken to avoid steady-state tracking errors that are induced by model reduction error.

Consider the tracking variables $z^r := Tz$ where the matrix $T \in \mathbb{R}^{t \times o}$ is defined by taking the $t$ rows of the identity matrix corresponding to the indices of the desired performance variables. The goal is to make $z^r$ converge to a constant reference signal $r$:

$$\lim_{k \to \infty} z^r_k = r.$$  

First, the FOM equilibrium corresponding to the setpoint $r$ can be computed by solving the linear system:

$$S^f \begin{bmatrix} x^f_{\infty} \\ u_{\infty} \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}, \quad S^f = \begin{bmatrix} A^f - I & B^f \\ TH^f & 0 \end{bmatrix}. \quad (2.22)$$

To ensure that (2.22) has a unique solution, it is assumed that the matrix $S^f$ is square (i.e., number of tracking variables $t$ is equal to the number of control inputs $m$), and full rank.
2.6. CONSTANT SETPOINT TRACKING

2.6.1 Constant Setpoint Tracking ROMPC

The ROMPC scheme defined in Section 2.3 can be modified for setpoint tracking with only a few slight changes to the OCP (2.10), namely in the cost function and terminal constraint set. First, the cost function in (2.10) should be changed to:

\[ \|\delta \bar{\bar{x}}_{k+N|k}\|^2_P + \sum_{j=k}^{k+N-1} \|\delta \bar{x}_j\|^2_Q + \|\delta \bar{u}_j\|^2_R, \]  

(2.23)

where \(\delta \bar{\bar{x}} = \bar{x} - \bar{x}_\infty\) and \(\delta \bar{u} = \bar{u} - \bar{u}_\infty\) and \((\bar{x}_\infty, \bar{u}_\infty)\) are the desired steady state targets for the reference ROM. Second, the terminal set \(\bar{X}_f\) should be computed using the same techniques discussed in Section 2.3.2, but should be designed around the target point \((\bar{x}_\infty, \bar{u}_\infty)\) rather than the origin (see (Lorenzetti et al., 2019, Section IV)). The combination of these changes is sufficient to guarantee that the reference ROM controlled by the reduced order OCP exponentially converges to the target point \((\bar{x}_\infty, \bar{u}_\infty)\) (rather than the origin).

Due to model reduction errors, the target point \((\bar{x}_\infty, \bar{u}_\infty)\) needs to be carefully chosen to ensure setpoint tracking for the controlled FOM. For a setpoint \(r\) and associated solution \((x_{\infty}^f, u_{\infty})\) to (2.22), the reduced order estimator steady state value (ignoring disturbances) is given by:

\[ \hat{x}_\infty = D(Bu_{\infty} + LC_f x_{\infty}^f), \]  

(2.24)

where \(D = (I - (A - LC))^{-1}\). Then, by requiring the controller (2.8) to also be at steady state, the target point \((\bar{x}_\infty, \bar{u}_\infty)\) that enables setpoint tracking is the solution to:

\[ S \begin{bmatrix} \bar{x}_\infty \\ \bar{u}_\infty \end{bmatrix} = \begin{bmatrix} 0 \\ K\hat{x}_\infty - u_{\infty} \end{bmatrix}, \quad S = \begin{bmatrix} A - I & B \\ K & -I \end{bmatrix}, \]  

(2.25)

where it is assumed that the square matrix \(S\) is full rank. Of course both \((x_{\infty}^f, u_{\infty})\) and \((\bar{x}_\infty, \bar{u}_\infty)\) must also satisfy the constraints \(z_{\infty} \in Z, u_{\infty} \in U, \bar{z}_\infty \in \bar{Z}, \bar{u}_\infty \in \bar{U}\).

2.6.2 Setpoint Tracking Closed-Loop Performance

The same closed-loop performance as in Section 2.4 is achieved, except that the system asymptotically converges to \(x_{\infty}^f\) rather than the origin. In particular Lemma 1 still holds, which guarantees robust constraint satisfaction.

**Theorem 2** (Robust Setpoint Tracking). Suppose that \(A_e\) is Schur stable, and that the OCP is feasible at time \(k_0\) and drives the reference ROM to the target point \((\bar{x}_\infty, \bar{u}_\infty)\) exponentially fast. Additionally, suppose the target point is defined for some setpoint \(r\) such that \((x_{\infty}^f, u_{\infty}), \bar{x}_\infty, \bar{u}_\infty\) are the unique solutions to the equations (2.22), (2.24), (2.25). Then, the closed-loop system robustly and asymptotically converges to a compact set containing \(x_{\infty}^f\) and the tracking variables...
$z^r$ converge to a compact set containing $r$.

**Proof.** Consider the combined errors $\delta \xi = [(\delta x^f)^\top, \delta \hat{x}^\top]^\top$ where $\delta x^f = x^f - x^f_\infty$ and $\delta \hat{x} = \hat{x} - \hat{x}_\infty$.

The dynamics of these errors are given by

$$
\delta \xi_{k+1} = A_\xi \delta \xi_k + B_\xi \delta r_k + G_\xi \omega_k,
$$

where $\delta r = [(\delta \bar{x})^\top, \delta \bar{u}^\top]^\top$ where $\delta \bar{x} = \bar{x} - \bar{x}_\infty$ and $\delta \bar{u} = \bar{u} - \bar{u}_\infty$. As shown in the proof of Theorem 1, since the simulated ROM converges exponentially to $(\bar{x}_\infty, \bar{u}_\infty)$, in the limit

$$
\lim_{k \to \infty} \|\delta \xi_k\| \leq B,
$$

where $B \in \mathbb{R}$ is finite. Further, since $S^f$ and $S$ are square and full rank the combined steady state of the controller system defined by the set of points $(x^f_\infty, u_\infty), \hat{x}_\infty$, and $(\bar{x}_\infty, \bar{u}_\infty)$ is unique and thus $x^f$ will converge to a vicinity of the state $x^f_\infty$. With the tracking error $\delta z^r := z^r - r = TH^f \delta x^f$ it must also be true that

$$
\lim_{k \to \infty} \|\delta z^r_k\| \leq \|TH^f\|B.
$$

Therefore the tracking variables also converge to a compact set containing the setpoint $r$. \qed

**Corollary 2** (Setpoint Tracking). Let the conditions from Theorem 2 hold. Then, if $W = \{0\}$ and $V = \{0\}$ the tracking variables $z^r$ converge asymptotically to the setpoint $r$.

**Proof.** Following the same approach as in the proof of Corollary 1 in the absence of disturbances the constant $B$ in the proof of Theorem 2 can be chosen to be $B = 0$. Therefore $\lim_{k \to \infty} \|\delta z^r_k\| = 0$. \qed

### 2.7 Continuous-Time Linear ROMPC

In some practical settings, a continuous time formulation of the control problem might be preferable, for example to avoid discretizing a high-dimensional continuous-time model in time. In this case, the time-discretization required for practical implementation would occur after the model has been reduced and the controller has been synthesized. For the benefit of the practitioner, this section includes a discussion on how the proposed ROMPC scheme can be synthesized and analyzed when using continuous-time systems, with only a few modifications.

First, the FOM (2.1) and ROM (2.7) would be expressed as ordinary differential equations instead of difference equations, and the cost functions (2.4) and (2.6) would be expressed as integrals.

Second, the continuous-time versions of the control law (2.8) and state estimator (2.9) are defined
by
\[
    u(t) = \bar{u}(t) + K(\dot{x}(t) - \bar{x}(t)),
\]
\[
    \dot{x}(t) = A\hat{x}(t) + Bu(t) + L(y(t) - C\hat{x}(t)).
\]

The reduced order Riccati method proposed in Section 2.5 can be applied to compute the gain matrices \(K\) and \(L\), but with the solution to the continuous algebraic Riccati equations:

\[
    A^T X + XA - XBR^{-1}B^T X + Q_z = 0,
\]
\[
    AY + YA^T - YC^TY + Q_w = 0,
\]

and choosing \(K = -R^{-1}B^T X\) and \(L = YC^T\).

Third, the continuous-time reference ROM is given by:

\[
    \dot{\bar{x}}(t) = A\bar{x}(t) + B\bar{u}(t),
\]
\[
    \bar{z}(t) = H\bar{x}(t),
\]

and is controlled by the reduced order optimal control problem:

\[
    (\bar{x}_k^*, \bar{u}_k^*) = \arg\min_{\bar{x}_k, \bar{u}_k} \|\bar{x}_{k+N}\|^2_P + \sum_{j=k}^{k+N-1} \|\bar{x}_j\|^2_Q + \|\bar{u}_j\|^2_R,
\]

subject to \(\bar{x}_{i+1} = A_d\bar{x}_i + B_d\bar{u}_i\),

\[
    H\bar{x}_i \in \bar{Z}, \quad \bar{u}_i \in \bar{U},
\]

\(\bar{x}_{k+N} \in \bar{X}_f, \quad \bar{x}_k = \bar{x}(t)\),

which is essentially the same as (2.10), but where the initial condition uses the continuous-time state \(\bar{x}(t)\) and the dynamics model is a discretized version of the ROM (2.29). In particular, a zero-order hold equivalent ROM with \(A_d := e^{A\Delta t}\) and \(B_d := \int_0^{\Delta t} e^{A\tau} B d\tau\) is used, where \(\Delta t\) is the discretization time. With this choice, the continuous-time reference ROM will match the solution to the OCP at the sampled time steps under a zero-order hold control. In other words, suppose the OCP was solved with \(\bar{x}_k \cdots k + 1\Delta t\) to give the optimal discrete-time trajectory sequence \((\bar{x}_i^*\}_{i=k}, \{\bar{u}_i^*\}_{i=k}^{k+N-1}\). Then, if the control sequence is applied as a zero-order hold to the reference ROM (2.29) the trajectories would be equivalent at the sampled times: \(\bar{x}(t+i\Delta t) = \bar{x}_i^*\) for \(i = 0, \ldots, N\). Note that the reduced order OCP is only solved every \(\Delta t\) seconds.

Finally, the error bounds used to compute the tightened constraint sets \(\bar{Z}\) and \(\bar{U}\) can be computed as discussed in Chapter 3. The overall method is summarized in Algorithm 4, where it is assumed that the OCP is feasible at \(t = t_0\) and where the control rate is given by \(\delta t\) (which is generally faster than \(\Delta t\)).
Algorithm 4: ROMPC Control (Online, Continuous-Time)

1: procedure ROMPC($\delta t$)
2: \hspace{1em} $t \leftarrow 0$, $\bar{x}(t) \leftarrow 0$, $\hat{x}(t) \leftarrow 0$
3: \hspace{1em} loop
4: \hspace{2em} if $t \geq t_0$ then
5: \hspace{3em} if $(t - t_0) \mod \Delta t = 0$ then
6: \hspace{4em} $\bar{u} \leftarrow \text{solveOCP}(\bar{x}(t))$
7: \hspace{4em} $u(t) \leftarrow \bar{u} + K(\hat{x}(t) - \bar{x}(t))$
8: \hspace{3em} else
9: \hspace{4em} $u(t) \leftarrow \text{startupControl}()$
10: \hspace{4em} $\bar{u}(t) \leftarrow u(t) - K(\hat{x}(t) - \bar{x}(t))$
11: \hspace{2em} $y(t) \leftarrow \text{getMeasurement}()$
12: \hspace{2em} applyControl($u(t)$)
13: \hspace{2em} $\bar{x}(t + \delta t) \leftarrow \text{integrate (2.29)}$
14: \hspace{2em} $\hat{x}(t + \delta t) \leftarrow \text{integrate (2.27)}$
15: \hspace{1em} $t \leftarrow t + \delta t$

2.8 Conclusion and Future Work

In this chapter, a reduced order model predictive control scheme is proposed for solving constrained optimal control problems for high-dimensional systems. This approach leverages reduced order models for computationally efficient controller design, and through the proposed synthesis (Section 2.5) and analysis (Chapter 3) techniques guarantees robust stability and constraint satisfaction for the controlled high-dimensional system. Applications of particular interest include systems whose models are derived from finite approximations to infinite-dimensional systems, such as semi-discretized PDEs.

There are several additional considerations of both theoretical and practical significance related to this work. First, an analysis on the sub-optimality of the ROMPC scheme arising from model reduction error would be of theoretical importance. Second, additional techniques for synthesizing the feedback controller gains that guarantee stability \textit{a priori} could be considered. Another theoretically important topic is the study of when and how model order reduction algorithms can preserve controllability and observability properties, which are important for controller design and which we have only assumed to hold in Assumption 2. One practical improvement would be to reformulate the optimal control problem to avoid the need to compute a terminal set $X_f$, since algorithms for computing invariant sets scale poorly with dimension and some ROMs may still have tens or hundreds of dimensions. Finally, and perhaps most importantly, the approach should be extended to the nonlinear setting to significantly expand the applications where high-performance ROM-based MPC would be most beneficial. Some preliminary work on extending the linear ROMPC approach to nonlinear settings is presented in Chapter 6 where we will also discuss more specific additional avenues of future work for nonlinear ROMPC.
Chapter 3

Linear ROMPC Error Analysis

Quantifying model reduction error is important for any application leveraging ROMs, including simulation, system design optimization, and control. Approaches for error analysis can be classified as either *a priori* or *a posteriori*, and are chosen based on needs of the particular application. In the context of control, (Negri et al., 2013) considers the linear quadratic regulator problem for unconstrained control of a PDE system, and provides *a posteriori* error bounds related to the optimal control solution. An improved method for *a posteriori* error bounding of LQR solutions is also discussed in (Kärcher and Grepl, 2014), which specifically computes bounds for both the control error and cost functional error. As another example, Gubisch and Volkwein (2017) derive an *a posteriori* estimate of the control error from a POD ROM-based constrained optimal control problem, and use the results to evaluate if the POD ROM should be adjusted. However, in the context of control it is also important to have *a priori* guarantees, which give confidence that the controller will perform as desired before it is implemented. In this chapter, we propose an *a priori* approach for efficiently analyzing model reduction error that is specifically targeted for the constrained optimal control problems considered in this thesis.

3.1 Introduction

Analyzing how model reduction error could lead to constraint violations is particularly important for reduced order model-based constrained optimal control problems. Recall from Chapter 2 that we impose constraints (2.3) on the high-dimensional performance outputs $z$ and the controls $u$:

$$z \in Z, \quad u \in U,$$

where $Z$ and $U$ are convex polytopes. Lemma 1 then states that these constraints will be satisfied by the controlled high-dimensional system, even in the presence of model reduction error, state
estimation error, and bounded disturbances, if a set of bounds on the worst-case tracking errors \( (\Delta_z, \Delta_u) \) can be used to compute tightened constraint sets \( \bar{Z} \) and \( \bar{U} \) for the ROMPC scheme. In this chapter, we propose an approach for efficiently computing the bounds \( (\Delta_z, \Delta_u) \) \textit{a priori}, such that robust constraint satisfaction can be guaranteed for the controlled high-dimensional system.

Two primary challenges associated with computing the bounds \( (\Delta_z, \Delta_u) \) are computational efficiency and the amount of conservatism introduced. Computational efficiency is important since the error dynamics system (2.11) is high-dimensional, and conservatism is important because excessive constraint tightening can lead to sub-optimal closed-loop performance. Unfortunately, since the error bounds must be computed \textit{a priori} (i.e. we must consider worst-case scenarios), a certain amount of conservatism is unavoidable.

Two approaches for \textit{a priori} model reduction error error analysis in the context of ROMPC have been previously proposed in (Löhnig et al., 2014; Kögel and Findeisen, 2015a; Lorenzetti et al., 2019). Linear programming is used in (Kögel and Findeisen, 2015a; Lorenzetti et al., 2019), but in both cases the full order dynamics model (2.1) is embedded in the constraints of the linear program, and thus these approaches do not scale well with problem dimension. The approach used in (Löhnig et al., 2014) produces a bound on the norm of the error state and is slightly more computationally efficient, but at the cost of being more conservative, as shown in (Lorenzetti and Pavone, 2020a). Similar error analyses using norms are discussed in (Haasdonk and Ohlberger, 2011; Hasenauer et al., 2012). The approach proposed in this chapter can be viewed as a blend of the two approaches, where we use linear programming and norm bounds to reduce conservatism and computational complexity. This work was originally published in (Lorenzetti and Pavone, 2020a; Lorenzetti et al., 2021).

### 3.2 Projection-Based Model Reduction Error Analysis

The error bounds \( \Delta_z \) and \( \Delta_u \) that are used to tighten the constraints in (2.3) are bounds on \( H_z \delta_z \) and \( H_u \delta_u \), which can also be written as \( E_z \epsilon \) and \( E_u \epsilon \) where:

\[
E_z = \begin{bmatrix} H_z H^f & 0 \end{bmatrix}, \quad E_u = \begin{bmatrix} 0 & H_u K \end{bmatrix}.
\]

Therefore, to compute worst-case bounds for these quantities the trajectories of the error dynamics (2.11) must be analyzed. Instead of using the recursive form of the dynamics, consider the equivalent form:

\[
\epsilon_k = A^{k-k_0} \epsilon_{k_0} + \sum_{j=k_0}^{k-1} A^{k-j} B \epsilon_{j+2} + G \omega_j,
\]

for any time \( k > k_0 \). Additionally, the time \( k_0 \) is defined as \( k_0 = k_0 - 2\tau \) where \( k_0 \) is the time when the ROMPC scheme takes control of the full order system and \( \tau \) is a user-defined time horizon (discussed at the end of Section 3.2.3). From this representation of the error, it can be seen that \( \epsilon_k \) is defined
for all \( k \geq k \) by the error \( \epsilon_k \), the trajectory of the reference ROM, \( r_k \), and the disturbances, \( \omega_k \). The disturbance terms have already been assumed to be bounded (Assumption 4), but the following additional assumptions on \( \epsilon_k \) and \( r_k \) are also made to ensure that \( \epsilon_k \) is bounded for all \( k \geq k \):

**Assumption 4.** The reference ROM trajectory satisfies \( \bar{u}_k \in \mathcal{U} \) and \( \bar{z}_k \in \mathcal{Z} \) for all \( k \geq k \).

**Assumption 5.** The initial error \( \epsilon_k \) is bounded by \( \| \epsilon_k \|_G \leq \eta_k \), where \( G \) is a positive definite weighting matrix such that \( \| \cdot \|_G = \sqrt{\langle \cdot, G \cdot \rangle} \).

Note that Assumption 4 is automatically satisfied for all \( k \geq k_0 \) since the reference ROM is controlled by the OCP (2.10) and \( \bar{Z} \subseteq \mathcal{Z} \) and \( \bar{U} \subseteq \mathcal{U} \). Further, Assumption 4 is easily verified in practice since the ROM is a reference (non-physical) system. Assumption 5 is hard to verify since it would require knowledge of the full order state \( x_f \). Nonetheless, as will be seen in the following sections, the value \( \eta_k \) can be chosen in an extremely conservative manner without having a major impact on the error bounds \( \Delta_z \) and \( \Delta_u \). This is accomplished by choosing the parameter \( \tau \) (which is the length of the time horizon under which Assumption 4 must hold before the starting time \( k_0 \)) to be sufficiently large, which will be shown to significantly reduce the influence of \( \epsilon_k \) on \( \epsilon_k \) for \( k \geq k_0 \).

In other words, increasing \( \tau \) increases the significance of Assumption 4 relative to Assumption 5. Additional discussion on the choice of \( \tau \) is presented in Section 3.2.3 and the choice of the matrix \( G \) is discussed in Section 3.2.2.

### 3.2.1 Computing Error Bounds

The proposed methodology for computing the error bounds \( \Delta_z \) and \( \Delta_u \) used to tighten the constraints in (2.12) is now presented. First, note that the error \( \epsilon_k \) is almost entirely defined by the most recent disturbances from \( r \) and \( \omega \) since the effects of older disturbances decay exponentially (when \( A \) is Schur stable). This fact is exploited to yield a methodology that is scalable to high-dimensional problems while not being overly conservative. In particular, using (3.2) the error \( \epsilon_k \) is divided into two components \( \epsilon_k = \epsilon_k^{(1)} + \epsilon_k^{(2)} \) where:

\[
\epsilon_k^{(1)} = A_k^{k-k} \epsilon_k + \sum_{j=k}^{k-\tau-1} A_k^{k-1-j} (B_{r_j} + G_{\omega_j}),
\]

\[
\epsilon_k^{(2)} = \sum_{j=k-\tau}^{k-1} A_k^{k-1-j} (B_{r_j} + G_{\omega_j}).
\]

The term \( \epsilon_k^{(2)} \) represents the contribution from the \( \tau \) most recent inputs, and the term \( \epsilon_k^{(1)} \) represents the contribution from everything prior. By choosing \( \tau \) to be sufficiently large, the error term \( \epsilon_k^{(1)} \) can be made negligible, and therefore can be bounded with more conservative techniques without making the total bound more conservative. The dominant error \( \epsilon_k^{(2)} \) is then analyzed using a worst-case optimization formulation that yields a tight bound.
CHAPTER 3. LINEAR ROMPC ERROR ANALYSIS

Bounding $\epsilon_{k}^{(1)}$:

This first term is bounded by considering the norm of the error, $\|\epsilon_{k}^{(1)}\|$. Using the triangle inequality and definition of induced matrix norms, an upper bound on the weighted norm of $\epsilon_{k}^{(1)}$ can be expressed as

$$
\|\epsilon_{k}^{(1)}\| G \leq \|A_{k}^{k-1}\| G \|\epsilon_{k}\| G + \sum_{j=k}^{k-\tau} \|A_{k}^{k-1-j}\| G \|B_{r_{j}}\| G + \sum_{j=k}^{k-\tau} \|A_{k}^{k-1-j}\| G \|G_{\omega_{j}}\| G,
$$

where $G$ is the same positive definite matrix used to define the weighted norm in Assumption 5.

Now, consider bounds $C_{r}$ and $C_{\omega}$ such that

$$
\|B_{r_{k}}\| G \leq C_{r} \quad \text{and} \quad \|G_{\omega_{k}}\| G \leq C_{\omega}
$$

for all $k \geq k$, and parameters $M \geq 1$ and $\gamma \in (0, 1)$ such that $\|A_{k}^{i}\| G \leq M \gamma^{i}$ for all $i \geq 0$. Since $A_{k}$ is Schur stable, parameters $M$ and $\gamma$ that meet these conditions are guaranteed to exist and $C_{r}$ and $C_{\omega}$ can be computed based on Assumptions 1 and 4. Methods for computing these parameters are discussed in Sections 3.2.2 and 3.2.2. These definitions lead to the bound

$$
\|\epsilon_{k}^{(1)}\| G \leq M \gamma^{k-k_{0}} \eta_{k} + M (C_{r} + C_{\omega}) \sum_{j=k}^{k-\tau} \gamma^{k-1-j}.
$$

Since $|\gamma| < 1$, for all $k \geq k_{0}$ the partial series is bounded by

$$
\sum_{j=k}^{k-\tau} \gamma^{k-1-j} = \sum_{j=0}^{k-\tau-1} \gamma^{j} \leq \sum_{j=0}^{\infty} \gamma^{j} = \frac{\gamma^{\tau}}{1-\gamma}.
$$

Additionally, since $k-k_{0} \geq 2\tau$ for all $k \geq k_{0}$ it holds that $\gamma^{k-k_{0}} \leq \gamma^{2\tau}$ for all $k \geq k_{0}$. Finally, the upper bound $\Delta^{(1)}$ on the weighted norm of $\epsilon_{k}^{(1)}$ can be expressed as:

$$
\|\epsilon_{k}^{(1)}\| G \leq \Delta^{(1)} := M \gamma^{2\tau} \eta_{k} + M \gamma^{\tau} (C_{r} + C_{\omega}) \frac{1}{1-\gamma},
$$

(3.3)

where $\eta_{k}$ is defined in Assumption 5. It is important to note that $\Delta^{(1)}$ is constant for all $k \geq k_{0}$.

Bounding $\epsilon_{k}^{(2)}$:

The dominant error term, $\epsilon_{k}^{(2)}$, is considered next, and it can be noted that of specific interest is the error corresponding to the terms $E_{z}\epsilon_{k}^{(2)}$ and $E_{u}\epsilon_{k}^{(2)}$. Consider a specific row of either $E_{z}$ or $E_{u}$, denoted by $\theta^{T}$. A bound, $\Delta^{(2)}(\theta)$, such that $\theta^{T} \epsilon_{k}^{(2)} \leq \Delta^{(2)}(\theta)$ for all $k \geq k_{0}$, can be computed via a
3.2. PROJECTION-BASED MODEL REDUCTION ERROR ANALYSIS

worst-case optimization formulation:

\[
\Delta^{(2)}(\theta) = \max_{\bar{x}, \bar{u}, \omega} \theta^T \sum_{j=0}^{\tau-1} A^T_{\tau-j} (B_r r_j + G_r \omega_j),
\]

subject to \( \bar{x}_{i+1} = A\bar{x}_i + B\bar{u}_i \),
\( r_i \in \bar{X} \times \mathcal{U} \), \( r_i = [\bar{x}_i^T, \bar{u}_i^T]^T \),
\( H\bar{x}_i \in \mathcal{Z} \), \( i \in [-\tau, \ldots, \tau-1] \),
\( \omega_i \in \mathcal{W} \times \mathcal{V} \), \( i \in [0, \ldots, \tau-1] \),

where \( \bar{X} := \{ \bar{x} \mid H\bar{x} \leq b\bar{x} \} \) is a compact polytopic set that is used to guarantee that the problem is bounded. A method for computing \( \bar{X} \) is defined in Section 3.2.2.

Defining \( \Delta_z \) and \( \Delta_u \):

The bounds on the terms \( \epsilon_k^{(1)} \) and \( \epsilon_k^{(2)} \) are now combined to give the final bounds \( \Delta_z \) and \( \Delta_u \). Again considering a row \( \theta^T \) of either \( E_z \) or \( E_u \):

\[
\theta^T \epsilon_k = \theta^T (\epsilon_k^{(1)} + \epsilon_k^{(2)}) \leq \|\theta^T G^{-1/2}\| \|\epsilon_k^{(1)}\| + \theta^T \epsilon_k^{(2)}.\]

Therefore, from the previously computed bounds the \( i \)-th element of vector \( \Delta_z \) and the \( j \)-th element of vector \( \Delta_u \) are defined as:

\[
\Delta_z,i := \Delta^{(1)}(e_{z,i}) + \Delta^{(2)}(e_{z,i}),
\]
\[
\Delta_u,j := \Delta^{(1)}(e_{u,j}) + \Delta^{(2)}(e_{u,j}),
\]

where \( i \in [1, \ldots, n_z] \), \( j \in [1, \ldots, n_u] \), \( e_{z,i}^T \) and \( e_{u,j}^T \) are the rows of \( E_z \) and \( E_u \) respectively, and:

\[
\Delta^{(1)}(\theta) = \|\theta^T G^{-1/2}\| \Delta^{(1)}.\]

3.2.2 Computing \( M, \gamma, G, C_r, C_\omega, \) and \( \bar{X} \)

The computation of several additional quantities are required to define the error bounds, namely \( M \), \( \gamma \), \( G \), \( C_r \), and \( C_\omega \) for \( \Delta^{(1)} \) in (3.3), and \( \bar{X} \) for \( \Delta^{(2)} \) in (3.4). Methods for defining these parameters are now presented.

Computing \( \bar{X} \):

The set \( \bar{X} := \{ \bar{x} \mid H\bar{x} \leq b\bar{x} \} \) is used to ensure the linear programs (3.4) are bounded, and will also be used in computing \( C_r \). For simplicity this set is chosen to be a hyper-rectangle where the rows
are the standard basis vectors in $\mathbb{R}^n$ (i.e. $H_\bar{x} = [I, -I]^T$). The $l$-th element of the vector $b_\bar{x}$ is then defined by the solution to the linear program:

$$
b_{\bar{x},l} = \maximize_{\bar{x},\bar{u}} \quad h_{\bar{x},l}^T \bar{x}_0,
\text{subject to} \quad \bar{x}_{i+1} = A\bar{x}_i + B\bar{u}_i, \\
\bar{u}_i \in \mathcal{U}, \quad i \in [0, \ldots, \bar{i} - 1] \\
H\bar{x}_i \in \mathcal{Z}, \quad i \in [0, \ldots, \bar{i}]
$$

where $h_{\bar{x},l}^T$ is the $l$-th row of $H_{\bar{x}}$ and $\bar{i} \geq n - 1$. By Assumption 2, specifically the observability of the pair $(A, H)$, we see that over time the constraints $\bar{z}_k \in \mathcal{Z}$ will restrict the admissible reduced order states $\bar{x}_k$ and therefore by choosing $\bar{i}$ large enough $b_{\bar{x},l}$ will be bounded. With this definition of $\bar{X}$ the following property holds:

**Proposition 1.** Suppose Assumptions 1, 2, and 4 hold. Then, the set $\bar{X}$ is compact and $\bar{x}_k \in \bar{X}$ for all $k \geq \bar{k}$.

*Proof.* We begin by showing $\bar{X}$ is compact. From Assumption 2 the matrix:

$$
\mathcal{O} := \begin{bmatrix} H^T & (HA)^T & \ldots & (HA^{n-1})^T \end{bmatrix}^T,
$$

is full rank. Since $\bar{x}_i = A^i \bar{x}_0 + \delta_i$ where $\delta_i = \sum_{j=0}^{i-1} A^{i-1-j}B\bar{u}_j$, the constraints $H\bar{x}_i \in \mathcal{Z}$ in (3.7) can be written as $HA^i \bar{x}_0 + H\delta_i \in \mathcal{Z}$. Further, the constraints of (3.7) enforce $\bar{u} \in \mathcal{U}$ and $\mathcal{U}$ is compact (Assumption 1) such that the terms $\delta_i$ are bounded. Therefore, with $\bar{i} \geq n - 1$ and $\mathcal{Z}$ compact (Assumption 1), the vector $\mathcal{O}\bar{x}_0$ is bounded, and since $\mathcal{O}$ is full rank $\bar{x}_0$ is bounded as well. Thus each element $b_{\bar{x},l}$ is bounded and by choice of $H_{\bar{x}}$ the set $\bar{X}$ is compact. To prove that $\bar{x}_k \in \bar{X}$ for all $k \geq \bar{k}$ we simply note that by the proposition assumptions the constraints of (3.7) are satisfied for all $k \geq \bar{k}$ and therefore $b_{\bar{x}}$ is a valid upper bound on $H_{\bar{x}}\bar{x}$.

**Computing $(C_r, C_\omega)$:**

The parameters $C_r$ and $C_\omega$ are used to bound the quantities $\|B_\epsilon r_k\|_G$ and $\|G_\omega \omega_k\|_G$, respectively. These bounds are computed by solving:

$$
C_r = \maximize_{\bar{x} \in \bar{X}, \bar{u} \in \mathcal{U}} \|B_\epsilon \bar{x}^T + \bar{u}^T\|_G, \\
C_\omega = \maximize_{w \in \mathcal{W}, v \in \mathcal{V}} \|G_\epsilon \bar{w}^T + \bar{v}^T\|_G,
$$

where $G$ is the same weighting matrix from Assumption 5. The optimization problems (3.8) can be solved by vertex enumeration (since the constraints are defined by convex polytopes) or upper bounded by convex relaxations (Mangasarian and Shiau 1986).
3.2. PROJECTION-BASED MODEL REDUCTION ERROR ANALYSIS

Proposition 2. Suppose Assumptions [1–3] hold. Then, $\|B_cr_k\|_G \leq C_r$ and $\|G_c\omega_k\|_G \leq C_\omega$ for all $k \geq \bar{k}$, and $C_r$ and $C_\omega$ are finite.

Proof. By Proposition 2 and Assumption 1 the sets $\bar{X}$, $\bar{U}$, $\bar{V}$, and $\bar{W}$ are compact, which guarantees $C_r$ and $C_\omega$ are finite. Additionally, by the bounded disturbance assumption it is guaranteed that $\omega_k \in W \times V$ for all $k$ and therefore $\|G_c\omega_k\|_G \leq C_\omega$ for all $k$. Similarly, by the proposition assumptions $r_k \in \bar{X} \times \bar{U}$ for all $k \geq \bar{k}$ such that $\|B_cr_k\|_G \leq C_r$ for all $k \geq \bar{k}$. \qed

Computing $(M, \gamma, G)$:

The parameters $M$ and $\gamma$ used in (3.3) are required to satisfy $M \geq 1$, $\gamma \in (0, 1)$, and $\|A_r\|_G \leq M\gamma^i$ for all $i \geq 0$. Two potential approaches for computing these parameters are discussed here. First, if $G$ is chosen such that $\|A_r\|_G < 1$ then $\gamma = \|A_r\|_G$ and $M = 1$ can be used since $\|A_r\|_G \leq \|A_r\|_G$. A value of $G$ for this approach can be computed by solving the discrete Lyapunov equation:

$$A_r^TGA_r - \eta^2G + I = 0,$$

where $\eta \in \mathbb{R}$ is some value such that $\eta \in (\max_j |\lambda_j(A_r)|, 1)$ where $\lambda_j(A_r)$ is the $j$-th eigenvalue of $A_r$. This approach will guarantee that $\gamma \leq \eta$ since $\|A_r\|_G = \|G^{1/2}A_rG^{-1/2}\|_2 = \max_i \sqrt{\lambda_i(\eta^2I - G^{-1})} < \eta$ where $\lambda_i(\cdot)$ denotes the $i$-th eigenvalue of the matrix.

Second, if $A_r$ is diagonalizable with eigenvalue decomposition $A_r = TDT^{-1}$ then $\gamma = \max_j |d_j|$ and $M = \|G^{1/2}T\|_2\|T^{-1}G^{-1/2}\|_2$ can be used, where $d_j$ denotes the $j$-th diagonal element of $D$. In this case $|\gamma| < 1$ is guaranteed by the Schur stability of $A_r$. Thus $G$ can be chosen to be any positive definite matrix, such as $G = (T^{-1})^T T^{-1}$ which would give $M = 1$ since $G^{1/2} = T^{-1}$. However, $G$ can also be more carefully selected to try to minimize the constant $M$ or more generally to decrease $\Delta^{(1)}$. This can be accomplished via an optimization-based approach, such as the geometric programming method described in [Lorenzetti and Pavone, 2020b] Section V). In practice, both the Lyapunov method and the diagonalization method involve operations of similar computational complexity, and thus it is generally suitable to just use the Lyapunov method. Unfortunately, the computational complexity of both methods scales as $O(n^3)$, which is not generally tractable for extremely high-dimensional problems where $n^f$ is in the ten-thousands or millions. Therefore we discuss some practical options to avoid computing these constants altogether in Remark 1.

3.2.3 Results and Practical Considerations

The procedure for computing the error bounds $\Delta_c$ and $\Delta_\omega$ that are used to tighten the constraints (2.12) is summarized in Algorithm 4. With the definition of these error bounds, Lemma 1 can be extended to the following result:
Algorithm 5: Linear ROMPC Error Bound Analysis

1: procedure ErrorBounds($\eta_k$, $\tau$, $\bar{\iota}$, ...) 
2: Compute $\bar{X}$ (Section 3.2.2) 
3: Compute $(C_\tau, C_u)$ (Section 3.2.2) 
4: Compute $(M, \gamma, G)$ (Section 3.2.2) 
5: Compute $\Delta^{(1)}$ (Equation 3.3) 
6: Compute $\Delta^{(2)}(\cdot)$ (Equation 3.4) 
7: Compute $\Delta Z, \Delta u$ (Equation 3.5) 
8: return $\Delta Z, \Delta u$

Theorem 3 (Robust Constraint Satisfaction). Suppose Assumptions 1-4 hold and that at time $k_0$ the OCP (2.10) is feasible. Then, under the proposed controller the full order system will robustly satisfy the constraints (2.3) for all $k \geq k_0$.

Proof. By design, under Assumptions 1-4 and from Propositions 1 and 2 the bounds $\|\epsilon^{(1)}_k\|_G \leq \Delta^{(1)}$ and $\theta^\top \epsilon^{(2)}_k \leq \Delta^{(2)}(\theta)$ hold for any $\theta$ and for all $k \geq k_0$. Therefore by definition $H_z \delta_{z,k} \leq \Delta z$ and $H_u \delta_{u,k} \leq \Delta u$ for all $k \geq k_0$. The final result therefore follows from Lemma 1.

We now present results from applying this error bounding procedure to several example problems. As in Chapter 2.5 we use the “Small and Large Synthetic” systems discussed in (Lorenzetti et al., 2021), as well as several practical problems described in detail in Chapter 4. In particular the following values are presented in Table 3.1 (for the disturbance free case):

1. $r = \max\{r_z, r_u\}$ where:
   $$r_{(z,u)} = \max_i 100 \times \frac{\Delta^{(1)}(e_{(z,u),i})}{b_{(z,u),i}}.$$  
2. $t_{(z,u),\text{max}} = \max_i \hat{\Delta}_{(z,u),i}$ and $t_{(z,u),\text{min}} = \min_i \hat{\Delta}_{(z,u),i}$ where:
   $$\hat{\Delta}_{(z,u),i} = 100 \times \frac{\Delta_{(z,u),i}}{b_{(z,u),i}}.$$  

The value $r$ defines the maximum percentage (over all constraints on $z$ and $u$) of constraint tightening induced by $\Delta^{(1)}$ with respect to the original constraint values $b_z$ and $b_u$ from (2.3). In other words, a value of $r = 1$ means that over all constraints, the $\Delta^{(1)}$ error would cause the constraint to be tightened by at most 1%. The values $\hat{\Delta}_{(z,u),i}$ define the percentage of total constraint tightening for each constraint, with respect to the constraint value. Therefore $t_{(z,u),\text{max}}$ is the maximum percentage of constraint tightening for either the constraints on $z$ or $u$, and similarly $t_{(z,u),\text{min}}$ is the minimum. For example, a value of $t_{z,\text{max}} = 1$ means that when defining the tightened constraint set $\bar{Z}$, the most a constraint is tightened by is 1% of its original value.

The values of $r$ are presented to demonstrate how the error bound component from $\Delta^{(1)}$ ends up being only a small fraction of the total error bound. This occurs because a large value of $\tau$ leads to
3.2. PROJECTION-BASED MODEL REDUCTION ERROR ANALYSIS

<table>
<thead>
<tr>
<th>Model</th>
<th>$\tau$</th>
<th>$r$</th>
<th>$t_{z,\text{max}}$</th>
<th>$t_{z,\text{min}}$</th>
<th>$t_{u,\text{max}}$</th>
<th>$t_{u,\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Synthetic</td>
<td>200</td>
<td>3.7e-6</td>
<td>13.3</td>
<td>3.3</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Large Synthetic</td>
<td>500</td>
<td>2.6e-3</td>
<td>4.1</td>
<td>4.1</td>
<td>6.7</td>
<td>6.7</td>
</tr>
<tr>
<td>Dist. Column</td>
<td>2000</td>
<td>3.7e-4</td>
<td>1.5</td>
<td>0.1</td>
<td>3.9</td>
<td>0.5</td>
</tr>
<tr>
<td>Tubular Reactor</td>
<td>1200</td>
<td>0.6</td>
<td>24.4</td>
<td>0.9</td>
<td>0.6</td>
<td>2.3e-3</td>
</tr>
<tr>
<td>Heatflow</td>
<td>1500</td>
<td>0.4</td>
<td>41.7</td>
<td>20.6</td>
<td>1.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Sup. Diffuser</td>
<td>2000</td>
<td>3.9e-9</td>
<td>0.2e-4</td>
<td>0.1e-4</td>
<td>0.4e-6</td>
<td>0.1e-6</td>
</tr>
</tbody>
</table>

Table 3.1: Error bound results for example problems described in Chapter 4 (and two synthetic/academic examples) with no external disturbances (i.e. model reduction error only). The parameter $\tau$ is the time horizon and in each case $\eta_k = 10^{-10}$ and $i = \tau$. The reported quantity $r$ is the maximum percentage of constraint tightening resulting from just the $\Delta^{(1)}$ term in (3.5). The quantities $t_{(z,u),\{\text{max/min}\}}$ are the max/min percentage of $z$ and $u$ constraint tightening resulting from the entire bounds $\Delta_z$ and $\Delta_u$ defined by (3.5).

As can be seen, for the values of $\tau$ in Table 3.1 the matrix norms decrease exponentially, and for $t = \tau$ are all less than $10^{-10}$. Therefore larger values of $\tau$ would have negligible effect on the error bounds. This analysis could also be used to provide a more automated way to choose the parameter $\tau$, for example by choosing $\tau$ such that $\|EA_t^t B_t\| \leq \epsilon_{\text{threshold}}$. 

Figure 3.1: Plot showing the decay of the matrix norm $\|EA_t^t B_t\|$, where $E = [E_z^T, E_u^T]^T$, for different values of $t \in [0, \ldots, \tau]$. For each example the value of $\tau$ is the same as presented in Table 3.1.

The vast majority of the error being captured by the bound $\Delta^{(2)}$. As mentioned before, this reduces conservatism since the norm bound approach used to define $\Delta^{(1)}$ is naturally more conservative. Another simple experiment to demonstrate this phenomenon is to study the change of the matrix norm $\|EA_t^t B_t\|$, where $E = [E_z^T, E_u^T]^T$, as $t$ increases, which is shown in Figure 3.1 for $t \in [0, \ldots, \tau]$. As can be seen, for the values of $\tau$ in Table 3.1 the matrix norms decrease exponentially, and for $t = \tau$ are all less than $10^{-10}$. Therefore larger values of $\tau$ would have negligible effect on the error bounds. This analysis could also be used to provide a more automated way to choose the parameter $\tau$, for example by choosing $\tau$ such that $\|EA_t^t B_t\| \leq \epsilon_{\text{threshold}}$. 

Computational Efficiency:

For extremely high-dimensional problems, computing the $\Delta^{(1)}$ term may be challenging or impossible (in particular computing the bounds on the norm of the matrix powers $\|A^t\|_C \leq M\gamma^t$). However, as is noted in Remark 1, it is significantly more important in practice to be able to compute the bounds $\Delta^{(2)}$. In fact, computing the bounds $\Delta^{(2)}$ remains computationally tractable since the complexity of the linear programs (3.4) only scale with the size of the reduced order model $n$, the number of control inputs $m$, the dimensions of the disturbances $m_w$ and $p$, and the time horizon parameter $\tau$. In many practical problems these parameters will be small enough that (3.4) can be solved efficiently with standard commercial solvers. Additionally, the number of linear programs that need to be solved only scales with the number of constraints that define $Z$ and $U$, which is typically not large.

One potentially challenging component to solving the linear programs (3.4) is computing the matrices $EA_i^tB$ and $EA_i^tG$ for $i = 0, \ldots, \tau$ (that define the objective functions) since the high-dimensional matrix $A_i$ is involved. However, the dimension of the resulting matrix products do not scale with $n^f$, and can be computed efficiently in a recursive fashion as long as the matrix-vector product operation can be efficiently performed (e.g. if $A_i$ is sparse). In fact, this is equivalent to simulating the dynamics from a particular initial condition.

Remark 1 (Practical Considerations for Extremely High-Dimensional Problems). Note that the preceding discussion and analysis (Figure 3.1) suggests that choosing $\tau$ to be large makes $\Delta^{(1)}$ negligible. Therefore, from a practical standpoint the proposed approach can be scaled to extremely high-dimensional problems. In particular, the computation of $\Delta^{(1)}$ could be foregone by choosing $\tau$ to be large enough such that $\|EA^\tau B\|$ is small.

Remark 2. Assumptions 4 and 5 are not very restricting. Assumption 4 is easily verifiable since it only considers the reference ROM trajectory. Additionally, as previously noted in Section 2.3.3, a “startup controller” is typically used to give time for the state estimator (and reference ROM system) to converge to meaningful values. This controller is likely designed to bring the system to a safe initial operating point that would satisfy the constraints, and thus also satisfy Assumption 4. Regarding Assumption 5 it is noted that the effect of the error term $\epsilon_k$ on the error $\epsilon_k$ (3.2) will be negligible for all $k \geq 0$ if the horizon parameter $\tau$ is chosen to be sufficiently large. Therefore the bound $\eta_k$ can be heavily conservative. For example, a value of $\eta_k = 10^{10}$ was chosen for all of the examples in Table 3.1.

3.2.4 Continuous-Time Error Analysis

In Section 2.7 we presented a continuous time formulation of the ROMPC controller that can be used when time-discretizing a high-dimensional continuous model is undesirable. In this section, we show how the error bounding analysis previously presented can also be used in continuous-time
settings. First, the continuous-time error dynamics can be written as:

$$\epsilon(t) = e^{A_e(t-t_\epsilon)}\epsilon(t_\epsilon) + \int_{t_\epsilon}^{t} e^{A_e(t-s)}(B_e r(s) + G_e \omega(s))ds,$$

where $t_\epsilon = t_0 - 2\tau$. Analogously to Assumptions 4 and 5 it is assumed that $\bar{u}(t) \in U$ and $\bar{z}(t) \in Z$ for all $t \geq t_\epsilon$ and that $\|\epsilon(t)\|_G \leq \eta_e$. The error is again divided into two components $\epsilon^{(1)}(t)$ and $\epsilon^{(2)}(t)$ with each term bounded separately. The bound on $\epsilon^{(1)}(t)$ is defined as:

$$\|\epsilon^{(1)}(t)\|_G \leq \Delta^{(1)}(\theta) := \beta e^{\alpha(2\tau)}\eta_e + \frac{\beta e^{\alpha\tau}(C_r + C_\omega)}{-\alpha},$$

(3.10)

where $\beta$ and $\alpha$ are defined such that $\|e^{A_e t}\|_G \leq \beta e^{\alpha t}$ for all $t \geq 0$ and the values $C_r$ and $C_\omega$ are bounds such that $\|B_e r(t)\|_G \leq C_r$ and $\|G_e \omega(t)\|_G \leq C_\omega$. The values $C_r$ and $C_\omega$ can be computed via (3.8) with $\bar{X}$ defined as in Section 3.2.2 except with the discrete-time $A_d$ and $B_d$.

The second error term is given by:

$$\epsilon^{(2)}(t) = \int_{t-\tau}^{t} e^{A_e(t-s)}(B_e r(s) + G_e \omega(s))ds,$$

which can be approximated via a quadrature scheme:

$$\epsilon^{(2)}(t) \approx \sum_{j=0}^{N_s} w_j e^{A_e(t-s_j)}(B_e r(s_j) + G_e \omega(s_j)),$$

where $N_s$ evenly spaced grid points are used such that $s_0 = t - \tau$, $s_1 = s_0 + \Delta t$ and so on. Therefore the bounds $\Delta^{(2)}(\theta)$ can be computed by solving the linear programs:

$$\Delta^{(2)}(\theta) = \max_{\bar{x}_{\epsilon}, \bar{u}, \omega} \theta^T \sum_{j=0}^{N_s} w_j e^{A_e (\tau-j\Delta t)}(B_e r_j + G_e \omega_j),$$

subject to

$$\bar{x}_{i+1} = A_d \bar{x}_i + B_d \bar{u}_i,$$

$$r_i \in \bar{X} \times U, \quad r_i = [\bar{x}_i^T, \bar{u}_i^T]^T,$$

$$H \bar{x}_i \in Z, \quad i \in [-N_s, \ldots, N_s],$$

$$\omega_i \in \bar{W} \times V, \quad i \in [0, \ldots, N_s].$$

Finally, the combined bounds $\Delta_\epsilon$ and $\Delta_u$ for constraint tightening are computed by (3.5).

Note that in this case the guarantees on constraint satisfaction are made without accounting for the numerical approximations error (e.g. from approximation of the integral when considering $\epsilon^{(2)}(t)$). Accounting for this error to derive exact guarantees is left for future work. However, in practice this analysis is still useful for approximately quantifying the worst-case error. The computational efficiency is consistent with the discrete-time case. In particular, the matrices that
define the cost functions in (3.11), namely $E e^{A_s} B_e$ and $E e^{A_s} G_e$ for $s = 0, \Delta t, 2\Delta t, \ldots, \tau$, can be efficiently computed. One approach would be to first compute the matrix exponential $e^{A_s \Delta t}$ and then to recursively compute the matrix products, but for extremely large problems another approach would be to use some integration scheme (e.g. implicit Euler scheme) to simulate the dynamics:

$$\dot{\epsilon} = A_s \epsilon, \quad \epsilon(0) = b_e, g_e,$$

where $b_e$ and $g_e$ are the columns of $B_e$ and $G_e$. 
Chapter 4

Linear ROMPC Example Problems

4.1 Control of a Distillation Column

In this example, we consider a process engineering problem where we are designing a control system for a high-purity distillation column (shown in Figure 4.1) (Skogestad and Morari, 1988), which is used to separate binary liquid mixtures into products with high-purity. The process is modeled using an equilibrium-stage concept where each stage is at a vapor-liquid equilibrium, and where the liquid moves downward among the stages and the vapor moves upward with a feed flow entering in the middle of the column. The vapor at the top of the column is then condensed and is referred to as the distillate. The liquid at the bottom is referred to as the bottom product. Some of the distillate and bottom product is reintroduced into the column as a control variable and is referred to as reflux and boilup, respectively. The liquid on each stage, including the condenser (top) and reboiler (bottom), is called the stage liquid holdup.

Model and Problem Definition

The model used in this problem is described by Skogestad and Postlethwaite (2005) and is also used as an example in Kögel and Findeisen (2015b) for reduced order control. The distillation column has 40 stages and separates a binary mixture into products of 99% purity. The controlled inputs are the reflux flow rate $L$ [kmol/min] and boilup flow rate $V$ [kmol/min], as well as the distillate flow rate $D$ [kmol/min] and bottom product flow rate $B$ [kmol/min]. The performance variables of interest include the composition of the products $(x_B, y_D)$ [mole fraction] and liquid holdups in the reboiler $M_B$ [kmol] and condenser $M_D$ [kmol], and it is assumed that the measured values include product compositions and the liquid holdup in the condenser, reboiler, and at 3 additional stages of the column. The model is linearized about an equilibrium corresponding to a constant feed flow rate and 99% distillate purity. Changes in the feed flow rate $F$ [kmol/min] and feed composition $z_F$
[mole fraction] are modeled as disturbances.

The original model is a continuous time model with $n_f = 82$ states, where the states include the compositions and liquid holdups in each of the 40 stages as well as the condenser (the reboiler is stage 1). The model is discretized in time assuming a zero-order hold input with a sampling time of $\Delta t = 1$ minute. This model is also not asymptotically stable as it has two integrating modes resulting from the liquid holdup in the reboiler and condenser.

The constraints on the composition of the products are $|x_B| \leq 0.01$ and $|y_D| \leq 0.01$, and constraints on the liquid holdup in the reboiler and condenser are $|M_B| \leq 1$ and $|M_D| \leq 1$. Control constraints are that $\|u\|_\infty \leq 1$ where $u = [L, V, D, B]^T$. It is assumed that the disturbances on the feed flow are bounded by $|\Delta F| \leq 0.01$ and $|\Delta z_F| \leq 0.005$ which correspond to 1% of the nominal values. It is also assumed that there is noise on the measurements bounded by $\|v\|_\infty \leq .0001$.

Following the example in (Kögel and Findeisen, 2015b) we also choose to include a constraint $\|u_{k+1} - u_k\|_\infty \leq 0.2$ which encodes the realistic constraint that the flow rates cannot be instantaneously
changed. This constraint can be added by making the control \( u_k \) a part of the state and redefining the input to the augmented system to be the control rates.

The discrete time high-dimensional model is reduced using balanced truncation, where the inputs and outputs are first scaled by their constraint values. Before model reduction, the system was first decomposed into stable and unstable modes, and only the stable subsystem was reduced. The reduced order model has dimension \( n = 12 \).

Controller Definition

The controller gains \( K \) and \( L \) were computed using the reduced order Riccati method described in Algorithm 3. The error bounds were then computed, where the norm weighting matrix \( G \) was computing using the Lyapunov method and the constants \( C_r \) and \( C_w \) were computed by vertex enumeration, and the values \( \gamma = 0.988 \) and \( M = 1 \) were used. Additionally, the time horizon for the error analysis was chosen to be \( \tau = 2000 \) time steps (i.e. 2000 minutes since the sampling time is \( \Delta t = 1 \) minute), which (ignoring the \( \eta_{k_0 - 2\tau} \) term) led to the error bound \( \Delta^{(1)} = 3.6e^{-6} \), which is negligibly small. The error bounds computed are presented in Table 4.1 as a percentage of the corresponding constraint value, for both cases with and without the external disturbances. It can be seen that the error arising due to the model order reduction accounts for a smaller percentage of the total error bound than error from disturbances. The ROMPC problem was then formulated using the tightened constraints and with a time horizon of \( N = 50 \) time steps.

| Error Bounds (%) | \( y_D \) | \( x_B \) | \( M_D \) | \( M_B \) | \( |L| \) | \( |V| \) | \( |D| \) | \( |B| \) | \( |\Delta L| \) | \( |\Delta V| \) | \( |\Delta D| \) | \( |\Delta B| \) |
|------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| MOR error        | 1.0     | 1.5     | 0.1     | 0.8     | 0.2     | 0.4     | 0.2     | 0.5     | 0.5     | 2.3     | 0.6     | 0.8     | 0.5     |
| MOR + noise      | 5.6     | 6.6     | 1.3     | 6.4     | 1.5     | 2.4     | 1.6     | 2.1     | 1.7     | 6.2     | 3.0     | 8.7     | 2.5     |

Table 4.1: Error bounds for the distillation column example, values are expressed as a percentage of the constraint. The first line shows the error bounds when considering only errors due to model order reduction, and the second line also includes errors induced by bounded disturbances.

Results

This system is simulated with an initial condition corresponding to an equilibrium point with a distillate and bottom product mole fraction composition \( y_D = -0.005 \) and \( x_B = -0.009 \), and molar quantities in the condenser and reboiler \( M_D = -0.75 \), and \( M_B = 0.25 \). This equilibrium satisfies the state and control constraints and therefore satisfies the assumptions required for the error bounds to hold. In addition to simulating the system with the ROMPC controller previously discussed, a reduced order LQR (ROLQR) controller was designed was also implemented for comparison purposes. The simulation results of closed loop systems for both controllers are shown in Figures 4.2, 4.3, and 4.4. It can be seen in Figure 4.2 that the LQR controller violates the constraint on the bottom product composition \( x_B \), but this constraint is satisfied with the ROMPC controller as expected.
While the control constraints are not particularly limiting, the control rate constraints are saturated at the beginning of the response.

![Product Compositions](image)

**Figure 4.2:** Controlled product compositions in the binary distillation column simulation (Section 4.1). Note that the reduced order LQR controller violates the constraints while the proposed ROMPC controller does not.

![Control Inputs](image)

**Figure 4.3:** Control input (reflux flow rate $L$, boilup flow rate $V$, distillate flow rate $D$, and bottom product flow rate $B$) responses from a ROMPC controller simulation of the binary distillation column (Section 4.1).

![Control Inputs Rate](image)

**Figure 4.4:** Control input rate responses in a ROMPC controller simulation of the binary distillation column (Section 4.1). These control input rates have also been constrained by adding the input $u$ as an additional state to the model. Note that the ROMPC controller satisfies the constraints, as expected.

### 4.2 Control of a Chemical Reaction Process

A tubular reactor is a type of chemical reactor where a continuous flow passes through the vessel where the reaction occurs. Assuming cylindrical geometries and uniform reactant compositions along radial directions, models of such reactors can be simplified to consider composition changes only as a function of the axial position. In this example, we consider the control of this type of reactor
4.2. CONTROL OF A CHEMICAL REACTION PROCESS

(see Figure 4.5), where the goal is to stabilize the reaction process about a desired steady state. To control the reaction process the reactor is surrounded by independently controlled thermal jackets, which can effect the axial temperature distribution and therefore the axial reaction rates. In this particular example, maximizing the concentration of reaction product at the exit of the reactor requires a steady state operating temperature that is close to the safe operating temperatures of the reactor. We therefore consider this to be a constrained control problem, and propose the use of model predictive control.

Dynamics models for tubular reactors are typically described by nonlinear partial differential equations, which in our case are linearized about the desired operating point. The spatial discretization (along the reactor axial direction) therefore can lead to high-dimensional systems of linear ordinary differential equations. In this example, we use the model and steady state operating point described by [Agudelo et al. 2007], who also considered a model predictive control scheme based on proper orthogonal decomposition but did not account for model reduction errors.

Figure 4.5: A schematic of the chemical reactor used in Section 4.2. This figure can be found in [Agudelo et al. 2007]. The quantities $T_J$, denote the temperatures of the thermal jackets that control the process, and $C$ and $T$ denote the reactant concentration and temperature at a given point along the axial direction.

Model and Problem Definition

For the linearized model described by [Agudelo et al. 2007], the reactor is discretized axially with $N = 300$ evenly spaced nodes and there are $m = 3$ thermal jackets whose temperatures $T_J$ can be controlled. The state at a given node $i$ is described by the reactant concentration $C_i$ [mol/L] and the temperature $T_i$ [K]. It is assumed that three temperature sensors are available, located at axial positions that are 16.5%, 50%, and 83.5% of the length of the reactor. We additionally want to ensure that $T \in [300K, 395K]$ at every point in the reactor. However, to implement this restriction with a reasonable (and finite) number of state constraints we impose the condition at ten equally spaced points along the length of the reactor. A limit on the control authority is also imposed by enforcing that $T_J \in [300K, 395K]$. The state $x^f \in \mathbb{R}^{n_f}$ of the continuous time linear model is defined with the combination of both $C_i$ and $T_i$ for each node $i = 1, \ldots, N$ such that $n_f = 600$, and are non-dimensionalized using a reference concentration of $C_{\text{ref}} = 0.02$ mol/L and temperature of $T_{\text{ref}} = 340$ K. The control $u \in \mathbb{R}^m$ consists of the thermal jacket temperatures that are also
non-dimensionalized by $T_{\text{ref}}$.

The continuous time high-dimensional model is then reduced using balanced truncation where the output of the system is assumed to be a concatenation of the sensor measurements as well as the temperature values at the nodes where constraints are imposed. The reduced order model is chosen to have dimension $n = 30$.

**Controller Definition**

Since the model is expressed in continuous-time, we use the approach described in Section 2.7. First, the controller gains $K$ and $L$ are computed using the reduced order Riccati method described in Algorithm 3, but where the continuous algebraic Riccati equations (2.28) are used. The ROMPC problem is then formulated using the tightened constraints based on the computed error bounds. The discrete-time ROM is computed using a zero-order with discretization time of $\Delta t = 0.05$ seconds (the time constant associated with the fastest eigenvalue is close to 0.3 seconds). The horizon for the ROMPC problem is $N = 50$ steps.

The constraints imposed in the ROMPC scheme include both the temperature constraints at ten equally spaced points along the reactor as well as the control constraints. A tightened set of constraints which accounts for the model reduction and state estimation error is computed using the previously proposed method for continuous time systems. In particular, the Lyapunov approach was used for computing the positive definite matrix $G$ that defines the norm. With the chosen ancillary feedback controller, state estimator, and weighted norm $\|\cdot\|_G$, the values $\beta = 1$ and $\alpha = -0.549$ were computed. Additionally, the matrix exponential was computed directly and the integral was approximated by splitting up the interval with $\Delta t = 0.05$ and using a trapezoidal quadrature scheme. The time horizon is defined by $\tau = 60s$. A value for $C_r$ can be computed very efficiently by computing an upper bound on the true optimal value using the method described by Mangasarian and Shiau (1986). We also compute $\hat{X}$ using the same value of $\tau$ and initialize $\eta = 10^8$ to be extremely conservative for all realistic initial conditions for the system. In order for the remaining assumptions on the error bounds to be satisfied the system should satisfy the state and control constraints for a period of two minutes before applying the control scheme. This is easily satisfied in practice as the startup of such a reactor at a different (constraint admissible) steady state operating point is likely.

With the computed error bounds, the resulting temperature constraints were tightened by a maximum of 2.03 K (5%) and a minimum of 0.90 K (2.3%) and the control constraints were all tightened by less than 0.04 K (0.1%). This amount of tightening is quite small and suggests that the reduced order model is a good approximation to the true system dynamics.

**Results**

The full order linear model of the system is simulated using an implicit Euler integration scheme with a time step of $\Delta t = 0.01$ seconds over a period of 15 seconds. The initial condition for the system is
4.3 Control of a 2D Temperature Profile

In this example we consider a 2D heat transfer process across a flat plate. The temperature profile is controlled using four distributed heat sources along with four temperature sensors for output feedback, and the goal is to drive the temperature of four distributed locations to constant setpoints. The heat sources and sensors are shown in blue in Figure 4.9 where the red squares denote the regions where we wish to control the temperature to track the constant setpoint.

Model and Problem Definition

This example uses a linear PDE model describing the two-dimensional transfer of heat across a flat plate. To make the problem more interesting from a control perspective, an artificial perturbation is also added which makes the dynamics unstable. This model is a modified version of the HF2D9 model from (Leibfritz 2006) and is the same as the model used in (Lorenzetti and Pavone 2020a), which has five distributed controllable heat sources ($m = 5$) and five distributed sensors ($p = 5$). The PDE model is spatially discretized on a rectangular grid to obtain a high-dimensional ODE model, which is then discretized in time with a sampling time of $\Delta t = 0.05$ seconds and has dimension $n_f = 3481$. A stable/unstable decomposition is performed on the ODE model and a reduced order model with $n = 21$ is defined by reducing only the stable subsystem via balanced truncation.

The goal of the controller is to drive the temperature of four distributed locations on the flat plate to constant setpoints while satisfying constraints. In particular, the constraints are given by $Z = \{ z \mid \| z \|_\infty \leq 2 \}$ and $U = \{ u \mid \| u \|_\infty \leq 100 \}$ where $z \in \mathbb{R}^4$ are the temperatures at the four locations and $u \in \mathbb{R}^5$ represent the five distributed heat sources.
Controller Definition

As in the previous examples the gain matrices are computed using Algorithm 3. The ROMPC problem is formulated with a time horizon of \(N = 30\) steps, and with modifications for setpoint tracking described in Section 2.6 (where the tracking variables \(z^r = z\)). Additionally, the error bounds used to tighten the constraints are computed with \(\tau = 1500\) steps and using the Lyapunov approach to compute the matrix \(G\), and the constants \(C_r\) and \(C_\omega\) are computed through vertex enumeration. The norm bound results from the Lyapunov approach yield the values \(\gamma = 0.9879\) and \(M = 1\). The resulting error bounds lead to a maximum tightening of the constraints of 41.7\% for the output variables \(z\) and a maximum constraint tightening of 1.5\% for the control variables \(u\).

Results

The heatflow model is simulated with the initial condition being the origin, and where the goal is for the performance outputs to track a constant setpoint \(z^r = [0, 1, -0.1, 0.1]^T\). This initial condition is trivially guaranteed to have satisfied all conditions for the error bounds to hold. The simulation results are shown in Figures 4.10 and 4.11. Note that the control constraints are active in Figure 4.11 at the beginning of the transient, where a small gap exists due to the result of the constraint tightening. These results also show the performance of the setpoint tracking modifications, which successfully account for the steady-state model reduction error to ensure perfect tracking.

4.4 Control of a Supersonic Diffuser

This example studies the control of a supersonic diffuser, described by Lassaux (2002) (see Figure 4.12). The model was retrieved from the University of Florida Sparse Matrix Collection (Davis and Hu, 2011), specifically from the Oberwolfach model reduction benchmark collection (Willcox and Lassaux, 2005). This problem was also studied in the context of reduced order control by Hovland et al. (2008a).

A supersonic diffuser is a structure that is designed to slow down and increase the pressure in compressible fluids moving at supersonic velocities. Such devices are used in jet engines to decelerate the incoming flow to subsonic speeds before entering the compressor section. In typical nominal operating conditions, a normal shock forms just downstream of the throat in the diffuser. However, changes in the operating condition can cause the engine to “unstart”, which is a highly undesirable condition where the normal shock moves upstream and severely disrupts the airflow through the engine. This problem proposes to design an active control system to prevent engine unstart by using an air bleed upstream of the throat in the diffuser. The goal is to leverage the bleed control to keep the throat Mach number greater than 1, which keeps the normal shock of a started engine downstream of the throat.
4.4. CONTROL OF A SUPersonic DIFFUSER

Model and Problem Definition

The flow through a supersonic diffuser operating at a steady state Mach number \( M = 2.2 \) is modeled by the unsteady, inviscid, compressible Euler equations. These partial differential equations are then discretized using a finite volume method with 3078 nodes to yield a CFD model with \( n_f = 11730 \). This model is then linearized about the nominal operating condition, which has a throat Mach number \( M_T \approx 1.36 \) and nominal bleed of \( b = 0.01 \) \([\dot{m}_{\text{bleed}}/\dot{m}_0]\).

The control inputs are the perturbations to the nominal air bleed, \( \Delta b \), and the performance (and measured) output is the change in nominal throat Mach number, \( \Delta M_T \). Constraints on these values are given by \(-0.01 \leq \Delta b \leq 0.04 \) and \(-0.26 \leq \Delta M_T \leq 0.44 \). A process disturbance is defined by perturbations in the incoming flow density that are assumed bounded \(|w| \leq 1.225 \times 10^{-3} \) (0.1% of sea level density), and the measurement noise is assumed bounded by \(|v| \leq 10^{-4} \).

Controller Definition

The controller gains are computed using the reduced order Riccati method (Algorithm 3) with \( W_z = 1 \) and \( W_u = 0.5 \), and the OCP was defined with a horizon \( N = 20 \).

The error bounds were computed using the Lyapunov approach to compute the matrix \( G \), a horizon of \( \tau = 2000 \) was used, and the bounds \( C_r \) and \( C_\omega \) were computed using vertex enumeration. Results for the error bounds with no noise are given in Table 3.1 and with noise in Table 4.2. As can be seen from the results without noise, the error due to model reduction is negligible (i.e. the ROM is a very good approximation). With disturbances the constraints on \( \Delta M_T \) are tightened by 0.6% and 1.0% for upper and lower bounds respectively, and the constraints on \( \Delta b \) are tightened by 0.01% and 0.05%, both of which are quite reasonable.

<table>
<thead>
<tr>
<th>Error Bounds (%)</th>
<th>( \Delta M_T \leq 0.44 )</th>
<th>( \Delta M_T \geq -0.26 )</th>
<th>( \Delta b \leq 0.04 )</th>
<th>( \Delta b \geq -0.01 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOR + noise</td>
<td>0.6</td>
<td>1.0</td>
<td>0.01</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 4.2: Error bounds for the supersonic diffuser example (Section 4.4), values are expressed as a percentage of the constraint. These bounds includes errors induced by bounded disturbances and model reduction error. Based on the MOR-only error in Table 3.1, we can see that the model order reduction errors are negligible, and the majority of these error bound are a result of the disturbances.

Results

Simulation results are shown in Figures 4.13 and 4.14 which compare the proposed ROMPC scheme and a simple reduced order LQR controller. As can be seen in Figure 4.14, the proposed ROMPC scheme is able to satisfy the control constraints while the LQR controller violates them. Since control constraints often represent physical actuator limits, controllers that are unable to account for such limitations may under-perform or even be unsafe. In this simulation the initial condition is a steady state for the system where Assumption 4 is satisfied. To additionally highlight the motivation for the
proposed approach, note that a MPC scheme designed based on the full order model \((n_f = 11730)\) is not practical due to extreme computational requirements.
4.4. CONTROL OF A SUPERSONIC DIFFUSER

Figure 4.6: Simulation of the full order chemical reactor model (Section 4.2) with the reduced order model predictive control scheme. The dashed lines show the desired operating point for the reactor, and the black dashed line in the temperature plots indicates the imposed constraint. The x-axis denotes axial length along the reactor. It can be seen that the ROMPC scheme drives the process to the desired steady state while satisfying the temperature constraint.
CHAPTER 4. LINEAR ROMPC EXAMPLE PROBLEMS

Figure 4.7: Control inputs computed by the reduced order model predictive control scheme for the chemical reactor simulation (Section 4.2) shown in Figure 4.6. The black dashed line indicates the constraint constraint boundary, which as expected are satisfied for all time.

Figure 4.8: Snapshot of the temperature and concentration profiles at $t = 4$ seconds in a simulation of the chemical reactor (Section 4.2) under both the reduced order model predictive control scheme and a LQR controller. It can be seen that the LQR controller violates the temperature constraint, demonstrating the importance of constrained techniques such as MPC for this problem.
4.4. CONTROL OF A SUPersonic DIFFUSER

Figure 4.9: Layout of the flat plate for the 2D temperature profile control problem discussed in Section 4.3. The blue squares denote the distributed heat sources that are controlled as well as the temperature sensors. The red squares denote the target areas where we wish to control the temperature to track a given constant setpoint.

Figure 4.10: Performance variable responses from a ROMPC controller simulation of the 2D heatflow problem (Section 4.3). The dashed lines represent the setpoints and the solid lines represent the controlled response. Note that perfect setpoint tracking is possible by accounting for model reduction error in Section 2.6.

Figure 4.11: Control input response from the ROMPC controller for the 2D heatflow problem (Section 4.3). Note that the control constraints (black dashed lines) are active at the beginning, where a slight gap exists due to the constraint tightening that accounts for worst-case model reduction error.

Figure 4.12: Schematic of the supersonic diffuser discussed in Section 4.4 (figure from Hovland et al., 2008a). The supersonic incoming flow is slowed to subsonic speeds by a converging-diverging nozzle. An upstream bleed valve can be used to control the position of the normal shock that forms downstream of the throat, ensuring stable operation and avoiding “unstart” situations where the shock moves upstream of the throat.
Figure 4.13: Simulation results for the supersonic diffuser example in Section 4.4. This plot shows the performance output (throat mach number perturbation) for two simulations, one with the ROMPC scheme and another with a reduced order LQR controller.

Figure 4.14: Simulation results for the supersonic diffuser example in Section 4.4. This plot shows the control input (inlet bleed variation) for two simulations, one with the ROMPC scheme and another with a reduced order LQR controller. Note that the LQR controller fails to satisfy the imposed actuation constraint.
Chapter 5

Aircraft Control Using CFD-Based MPC

Optimal control techniques have been studied within the context of aircraft control for decades (Lorenzetti et al., 1969). More recently, computational advances have also enabled model predictive control techniques to be applied to constrained optimal control in aerospace applications (Eren et al., 2017). The study of MPC for aircraft control is particularly important as technology surrounding unmanned aerial vehicles (UAVs) has also developed, and the tasks that they are expected to perform have become increasingly complex and require increasingly higher-performance control. Unlike unconstrained optimal control techniques, the ability of MPC to incorporate state and control constraints allows UAV systems to safely operate at their performance limits.

5.1 Introduction

In this chapter, we consider a fully autonomous aircraft carrier landing problem where constrained optimal control methods are desirable for enabling safe and high-performing controllers to land UAVs in high sea states, recover from bolters (unarrested landings), handle wave-offs, and minimize dispersions and sink rates prior to landing. Autonomous carrier landing systems currently in use, such as the Automatic Carrier Landing System (ACLS) and the Precision Approach and Landing System (PALS) (Urnes and Hess, 1985; Ellis, 2003), require significant human supervision or action, can only operate in nominal conditions, and are tailored to piloted aircraft. Existing approaches for fully autonomous carrier landings utilize robust linear control methods (Crassidis and Mook, 1992; Niewoehner and Kaminer, 1996), adaptive control, fuzzy logic, neural networks (Steinberg and Page, 2001), dynamic inversion (Steinberg and Page, 2001; Denison, 2007), linear quadratic gain-scheduling (Ramesh and Subbarao, 2016), and model reference adaptive control (Reed and Steck, 2000).
While these methods find success for nominal landing scenarios, they do not satisfactorily address practical higher-performance landing scenarios that require more extreme maneuvering due to rough seas, wind gusts, strong carrier wake turbulence, and bolter recovery. To address these complex considerations, we propose the use of a model predictive control (MPC) scheme.

Several existing works have also proposed the use of MPC for autonomous carrier landing applications. The approach in (Koo et al. 2015) addresses the glideslope tracking problem (i.e. tracking a nominal approach to the carrier) by designing an MPC controller based on a linearized longitudinal aircraft dynamics model and a linear carrier motion model, but does not consider the effect of disturbances. Then, in (Misra and Bai 2018) a stochastic MPC scheme is proposed which focuses on glideslope tracking in the presence of uncertain wind gusts, described by a Dryden turbulence model. This work was then extended in (Misra and Bai 2019) to consider measurement uncertainty, a carrier airwake model, and a carrier motion model. A dynamics model with slightly higher fidelity is used in (Ngo and Sultan 2016), which addresses a helicopter ship-based landing problem. Specifically, the helicopter model incorporates dynamics from rigid-body motion, a simplified rotor blade structural model, and a simplified rotor aerodynamics model. Additionally, a carrier airwake model built from computational fluid dynamics (CFD) data is included, but only by interpolation with piecewise polynomial functions. In these approaches, the inclusion of models beyond the nominal aircraft rigid-body dynamics reflects the fact that MPC control in such a high-performance environment requires models with increased sophistication. Yet, the computational requirements of MPC have still limited these approaches to relatively simple models, and in particular low-fidelity aerodynamics models.

Classical approaches for modeling aircraft aerodynamics for control applications have generally leveraged databases of coefficients that are determined experimentally or through computational fluid dynamics techniques. Some aerodynamic coefficients are static, meaning they are determined assuming no body motion and steady flow. Others, such as aerodynamic derivative coefficients, capture aerodynamic changes during aircraft maneuvers by making a quasi-static assumption that the motion of the aircraft is slow relative to the rate that the airflow reaches steady state. Aerodynamic coefficient-based modeling and quasi-static assumptions are sufficient for many aircraft control applications, since the majority of operational flight envelopes are restricted. In other situations, such as autonomous carrier landings, it might be beneficial or even necessary to model unsteady aerodynamics. For example, in rapid pitch-up maneuvers where the stall angle of attack is reached, an aircraft might generate more lift than predicted with steady aerodynamic assumptions due to a phenomenon known as dynamic lift (Stevens et al. 2015). Control algorithms for autonomous aircraft carrier landings would benefit from unsteady aerodynamics modeling since high-performance maneuverability is needed, but also because the aircraft must fly through the turbulent wake of the carrier. Other control applications that could benefit from unsteady aerodynamic modeling include flexible aircraft that exhibit significant aeroelastic behavior, or novel research aircraft that are...
designed with non-standard aerodynamic properties, such as flapping-wing UAVs.

We therefore propose to improve UAV performance (for autonomous carrier landing and beyond) by sidestepping the use of low-fidelity aerodynamics models and instead directly using high-fidelity computational fluid dynamics (CFD) models. Other works have also proposed this idea, such as (Bewley et al., 2001), which uses the Navier-Stokes equations for predictive control to reduce drag of a turbulent flow in a plane channel, but does not address the computational challenges associated with practical implementation. As in the rest of this thesis, we address the computational challenges associated with using CFD models for control by leveraging reduced order models. This approach has also been taken for simulation, optimization, and control of fluid flows. For example, Bergmann et al. (2005) use POD for optimal rotary control of a cylinder wake to reduced drag and Amsallem et al. (2013) use a CFD-based aeroelastic ROM to include flutter constraints for an aircraft trajectory optimization problem solved through dynamic programming. Within the context of model predictive control, a CFD ROM-based MPC controller is used in Hovland et al. (2008a) for shock position control in a supersonic jet engine inlet.

In this chapter, we leverage high-fidelity CFD models to synthesize the ROMPC scheme proposed in Chapter 2 for aircraft control problems, and in particular focus on autonomous carrier landing scenarios. The work in this chapter is the result of a collaboration with Andrew McClellan and Professor Charbel Farhat from Stanford University, who developed both the CFD models and the reduced order models used in the simulation and hardware results in Sections 5.4 and 5.5. Preliminary components to the results from this chapter were published in (Lorenzetti et al., 2020), but the results here include a more complex control problem (adding lateral dynamics in addition to longitudinal dynamics), and in this chapter we use even larger-scale CFD models. This chapter also includes results from hardware experiments which, to the best of our knowledge, constitute the first real-world use of a (reduced order) CFD model on embedded hardware for real-time control.

It is important to note that this chapter is not aiming to completely solve the aircraft carrier landing problem, but rather use the problem as a proving ground for the concept of CFD-based aircraft control. Since the scope of this chapter is restricted to the use of linear models, we use simulation results of a high-dimensional linear CFD model as a way of proving potential performance of the approach, but use real-world flight tests to prove the computational practicality of the approach. Therefore, the main contribution of this chapter is to provide initial results (both theoretical and practical) that suggest additional work in CFD-based control is well motivated. Specific suggestions for avenues of future work are discussed at the end of the chapter.

5.2 Aircraft Rigid-Body/Fluid Dynamics Model

The autonomous aircraft carrier landing scenario can be split into two segments. The first segment consists of glideslope trajectory tracking, where the aircraft lines up the approach and prepares for
touchdown. Second, once the aircraft is close to the target landing zone, the controller should try to minimize landing dispersion and sink rate. As a first step toward using CFD models within this application, we focus on the *glideslope tracking* problem in this chapter. Specifically, we develop a nonlinear coupled rigid-body/CFD model and then linearize this model about the desired glideslope trajectory. Future extensions that consider higher-performance flight scenarios (e.g. the touchdown segment of carrier landings) and leverage nonlinear models are discussed in Section 5.6.

Figure 5.1: The first stage of the autonomous carrier landing problem is glideslope trajectory tracking, which gives the aircraft a nominal approach to prepare for final touchdown. We model the dynamics via a nonlinear rigid-body dynamics model that is coupled with a nonlinear CFD aerodynamics model, and then linearized about the glideslope trajectory.

The nominal glideslope trajectory (fixed relative to Earth) is defined by a flight path angle $\gamma_0$ and a constant desired UAV velocity $V_0$. We assume the aircraft operates under constraints on its physical state (e.g. angle-of-attack, distance from glideslope) and with limited control authority (e.g. limited thrust), and that we have access to measurements of position, velocity, pitch angle, and pitch rate (i.e. the aircraft’s rigid-body state). Using a flat-earth assumption, we define a series of coordinate systems: (1) an Earth-fixed inertial frame $I$, (2) a body-fixed frame $B$, aligned with the principal axes of the aircraft, and (3) an (inertial) target frame $R$ that moves along the glideslope at the constant velocity $V_0$ and with a constant angular offset $\theta_0$ from $I$. The relationship between $I$ and $R$ is given in Figure 5.2, where the glideslope is defined by the flight path angle $\gamma_0$, and $\alpha_0$ is the equilibrium angle of attack for the UAV. In the definition of the angles in Figure 5.2 arrows indicate direction of increasing angle and an angle is zero when the vector is parallel to the horizontal, such that $\theta_0 = \alpha_0 + \gamma_0$.

Figure 5.3 shows the relationship between the target frame $R$ and the body-fixed frame $B$ as viewed along the lateral direction. The translational coordinates of interest in the glideslope tracking problem are the relative position $\delta p_x$, $\delta p_y$, $\delta p_z$, and the relative velocities $\delta \dot{p}_x$, $\delta \dot{p}_y$, $\delta \dot{p}_z$, which describe the translational errors of the UAV with respect to the target frame $R$. We will use an axis-angle representation of the aircraft rotational degrees of freedom when developing the dynamics,
5.2. AIRCRAFT RIGID-BODY/FLUID DYNAMICS MODEL

but we will also at times convert these quantities into the more common and interpretable Euler angle representations when showing results. Specifically, the Euler angle coordinates that describe the UAV’s orientation relative to the target frame $\mathcal{R}$ are $\delta\phi$ (roll), $\delta\theta$ (pitch), and $\delta\psi$ (yaw). Finally, rotational rates will be expressed in the dynamics model as time derivatives of the axis-angle parameters, but we will sometimes show results where these values are converted to the more standard aircraft body rates.

$$\delta p = \begin{bmatrix} \delta p_x \\ \delta p_y \\ \delta p_z \end{bmatrix}, \quad w \text{ is the fluid state vector,}$$

$$u^{\text{ctrl}} := \begin{bmatrix} T \\ M_x \\ M_y \\ M_z \end{bmatrix}^T \quad \text{is the vector of controls,}$$

$$T \text{ is the controlled thrust,}$$

$$M_{x,y,z} \text{ are the controlled moments in the body frame directions,}$$

$m$ is the UAV mass, $g$ is acceleration due to gravity, and $f^{\text{aero}}$ represents the aerodynamic force vector written in target frame coordinates. The vector $\delta\Theta = [\delta\theta_x, \delta\theta_y, \delta\theta_z]^T$ defines the orientation of the UAV relative to the target frame $\mathcal{R}$ using the axis-angle representation, and $R(\delta\Theta) \in \mathbb{R}^{3 \times 3}$ is the rotation matrix that maps UAV body frame coordinates to target frame coordinates.

5.2.1 Rigid-Body Dynamics Model

The target frame $\mathcal{R}$ moves at a constant velocity in the inertial frame $\mathcal{I}$ without rotating, and thus is also inertial. The translational equations describing the motion of the UAV relative to the target frame, written in target frame coordinates, are therefore given by:

$$M\ddot{\delta p} = f^{\text{aero}}(w) + f^{\text{gravity}} + R(\delta\Theta)f^{\text{ctrl}}(u^{\text{ctrl}}),$$

with:

$M := \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix}$, $f^{\text{gravity}} := \begin{bmatrix} mg\sin\theta_0 \\ 0 \\ -mg\cos\theta_0 \end{bmatrix}$, $f^{\text{ctrl}}(u^{\text{ctrl}}) := \begin{bmatrix} -T \\ 0 \\ 0 \end{bmatrix}$,
CHAPTER 5. AIRCRAFT CONTROL USING CFD-BASED MPC

The rotational equations of motion are given by:

\[ J \ddot{\omega} + \omega \times (J \omega) = M, \]

where \( J \) is the moment of inertia matrix with respect to the \( B \) frame, \( \omega = [p, q, r]^T \) is the body frame angular velocity vector, and \( M \) are external moments from aerodynamics and control actions in body frame coordinates. This equation is then transformed from body frame coordinates into target frame coordinates:

\[ R(\delta \Theta)(J \ddot{\omega} + \omega \times (J \omega)) = M^{aero}(w) + R(\delta \Theta)M^{ctrl}(u^{ctrl}), \]

where \( M^{aero} \in \mathbb{R}^3 \) is the vector of aerodynamic moments written in target frame \( R \) coordinates and \( M^{ctrl}(u^{ctrl}) = [M_x, M_y, M_z]^T \). Both the translational and rotational equations are written in \( R \) frame coordinates because this makes the formulation of the aerodynamics model simpler.

Finally, the body frame angular velocity vector \( \omega \) is rewritten in terms of the axis-angle parameters and their derivatives (Mäkinen, 2004). The axis-angle representation \( \delta \Theta \) defines the rotation angle as \( \| \delta \Theta \| \) and the rotation axis as \( e = \delta \Theta / \| \delta \Theta \| \). The rotation matrix can be derived from the axis-angle parameterization as:

\[ R(\delta \Theta) = \exp(\tilde{\delta} \Theta) = I + \frac{\sin \| \delta \Theta \|}{\| \delta \Theta \|} \tilde{\delta} \Theta + \frac{1 - \cos \| \delta \Theta \|}{\| \delta \Theta \|^2} \tilde{\delta} \Theta^2, \]

where \( \tilde{\delta} \Theta \in \mathbb{R}^{3 \times 3} \) is the skew-symmetric matrix such that, given \( v \in \mathbb{R}^3 \), \( \tilde{\delta} \Theta v = \delta \Theta \times v \). The body frame angular velocity vector and its derivative can be written as:

\[ \omega = T(\delta \Theta)\dot{\delta} \Theta, \quad \dot{\omega} = T(\delta \Theta)\ddot{\delta} \Theta + \dot{T}(\delta \Theta, \dot{\delta} \Theta)\dot{\delta} \Theta, \]

where:

\[ T(\delta \Theta) = \frac{\sin \| \delta \Theta \|}{\| \delta \Theta \|} \mathbf{I} - \frac{1 - \cos \| \delta \Theta \|}{\| \delta \Theta \|^2} \tilde{\delta} \Theta + \frac{\| \delta \Theta \|}{\| \delta \Theta \|^3} (\delta \Theta \otimes \delta \Theta). \]

Using the identity \( T(\delta \Theta)R(\delta \Theta) = T^T(\delta \Theta) \) and denoting by \( \widetilde{T(\delta \Theta)\dot{\delta} \Theta} \) the skew-symmetric matrix corresponding to the cross product operator (i.e. \( \widetilde{T(\delta \Theta)\dot{\delta} \Theta} v = T(\delta \Theta)\dot{\delta} \Theta \times v \)), a substitution and pre-multiplication by \( T(\delta \Theta) \) transforms the rotational equations of motion into:

\[ T^T(\delta \Theta)JT(\delta \Theta)\ddot{\delta} \Theta + \left( T^T(\delta \Theta)JT(\delta \Theta, \dot{\delta} \Theta) + T^T(\delta \Theta)\widetilde{T(\delta \Theta)\dot{\delta} \Theta} \right) \dot{\delta} \Theta = T(\delta \Theta)M^{aero}(w) + T^T(\delta \Theta)M^{ctrl}(u^{ctrl}). \]
5.2. AIRCRAFT RIGID-BODY/FLUID DYNAMICS MODEL

The nonlinear rigid-body dynamics model, consisting of both translational and rotational equations, is then given by:

\[ M_c(\delta \Theta) \ddot{\xi} + C_c(\delta \Theta, \dot{\delta \Theta}) \dot{\xi} = f_{aero}^c(w, \delta \Theta) + f_{gravity}^c + f_{ctrl}^c(\delta \Theta, u^{ctrl}), \]  \hspace{1cm} (5.1)

where \( \xi = [\delta p^\top, \delta \Theta^\top]^\top \) is the combined relative position and relative orientation vector, and where:

\[
M_c(\delta \Theta) = \begin{bmatrix}
M & 0 \\
0 & T^\top(\delta \Theta)JT(\delta \Theta)
\end{bmatrix},
\]

\[
C_c(\delta \Theta, \dot{\delta \Theta}) = \begin{bmatrix}
0 & 0 \\
0 & T^\top(\delta \Theta)JT(\delta \Theta) + \widehat{T}(\delta \Theta)\dot{\delta \Theta}JT(\delta \Theta)
\end{bmatrix},
\]

\[
f_{aero}^c(\delta \Theta) = \begin{bmatrix} f_{aero}(w) \\
T(\delta \Theta)M_{aero}(w)
\end{bmatrix},
\]

\[
f_{gravity}^c = \begin{bmatrix} f_{gravity} \\
0
\end{bmatrix},
\]

\[
f_{ctrl}^c(\delta \Theta, u^{ctrl}) = \begin{bmatrix}
R(\delta \Theta)f_{ctrl}(u^{ctrl}) \\
T(\delta \Theta)^\top M_{ctrl}(u^{ctrl})
\end{bmatrix}.
\]

5.2.2 CFD Aerodynamics Model

The computational fluid dynamics model assumes inviscid, compressible flow and is built with an Arbitrary Lagrangian-Eulerian (ALE) framework (Lesoinne et al., 2001). The fluid dynamics governing equations are:

\[
\frac{\partial(J_{\zeta} w)}{\partial t} \bigg|_{\zeta_{ref}} + J_{\zeta} \nabla \cdot \left( F(w) - \frac{\partial \zeta}{\partial t} w \right) = 0,
\]

where \( w = [\rho, \rho v_x, \rho v_y, \rho v_z, E]^\top \) is the fluid state vector, \( \rho \) is the air density, \( v_i \) are velocity components, and \( E \) is the total energy (i.e. internal plus kinetic) per unit volume. The vector \( \zeta \) describes the mesh position and \( \zeta_{ref} \) is the reference mesh configuration. Finally, \( J_{\zeta} = \det(d\zeta/d\zeta_{ref}) \) and \( F(w) \) is the inviscid flux function. Using a finite volume method, this model can be written in semi-discrete form as:

\[
\frac{\partial(A_{\zeta}(\zeta) w)}{\partial t} + F(w, \zeta, \dot{\zeta}) = 0,
\]

where \( A_{\zeta}(\zeta) \) is a diagonal matrix of control volumes and \( F \) is a numerical flux function. The dimension of this semi-discretized model is dependent on the size of the mesh. For example, a mesh of \( N = 200,000 \) nodes (for a vertex-based integration scheme) will lead to a CFD model with \( w \in \mathbb{R}^{n_{\text{CFD}}} \) where \( n_{\text{CFD}} = 5N = 1,000,000 \). Assuming rigid mesh motion we can write \( \zeta = \zeta(\xi) \), which allows us to simplify the semi-discretized CFD model to:

\[
A_{\zeta} \dot{\zeta} + F(w, \zeta, \dot{\zeta}) = 0,
\]  \hspace{1cm} (5.2)

where the control volume matrix \( A_{\zeta} \) is constant.
5.2.3 Linear Rigid-body/CFD Model

In this section, the coupled nonlinear rigid-body equations (5.1) and nonlinear fluid dynamics equations (5.2) are linearized about the nominal glideslope trajectory. The CFD model is first used to compute the UAV equilibrium angle of attack $\alpha_0$ associated with the desired glideslope trajectory. The angle of attack is then used to compute the equilibrium pitch angle $\theta_0 = \alpha_0 + \gamma_0$ that defines the orientation of the target frame $R$. The UAV relative position and orientation values must then satisfy $\xi = \dot{\xi} = \ddot{\xi} = 0$ at equilibrium since the rigid-body variables are already expressed relative to the target frame $R$, thus we do not bother defining a perturbation variable, $\delta\xi$. The equilibrium control $u_0 = [T_0, M_{x,0}, M_{y,0}, M_{z,0}]^\top$ corresponds to the thrust and applied moments required to maintain the equilibrium glideslope trajectory, and thus the full control is given by $u_{\text{ctrl}} = u_0 + u$ where $u = [\delta T, \delta M_x, \delta M_y, \delta M_z]^\top$ is the perturbation on the equilibrium control input. When computing the equilibrium we also make the assumption that altitude effects on the aerodynamics are negligible, which would require the equilibrium angle of attack to vary with altitude.

The linear, coupled rigid-body/CFD model associated with the nominal glideslope trajectory is given by:

$$
\dot{x}^f = A^f x^f + B^f u, \quad A^f = \begin{bmatrix} 0 & I & 0 \\ M_0^{-1}E_0 & 0 & M_0^{-1}P_0 \\ -A_\zeta^{-1}G_0 & -A_\zeta^{-1}C_0 & -A_\zeta^{-1}H_0 \end{bmatrix}, \quad B^f = \begin{bmatrix} 0 \\ M_0^{-1}B_0 \end{bmatrix},
$$

where the full order state is $x^f = [\xi^\top, \dot{\xi}^\top, \delta w^\top]^\top$ with dimension $n^f = n_{\text{CFD}} + 12$, the matrix $I$ is the appropriately sized identity matrix, and:

$$
M_0 = \begin{bmatrix} M & 0 \\ 0 & J \end{bmatrix}, \quad P_0 = \left. \frac{\partial f_{\text{aero}}}{\partial w} \right|_0, \quad E_0 = \left. \frac{\partial f_{\text{ctrl}}}{\partial \xi} \right|_0,
$$

$$
B_0 = \left. \frac{\partial f_{\text{ctrl}}}{\partial u} \right|_0, \quad G_0 = \left. \frac{\partial F}{\partial \xi} \right|_0, \quad C_0 = \left. \frac{\partial F}{\partial \dot{\xi}} \right|_0, \quad H_0 = \left. \frac{\partial F}{\partial w} \right|_0
$$

Additionally, note that in the definition of $E_0$ we have assumed the term $\left. \frac{\partial f_{\text{aero}}}{\partial \xi} \right|_0$ to be negligible. As previously mentioned, we assume that measurements of the rigid-body states $\xi$ and $\dot{\xi}$ are available, and therefore the measurement model is:

$$
y = C^f x^f, \quad C^f = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix},
$$

such that $y = [\xi^\top, \dot{\xi}^\top]^\top$. We also include constraints only on the rigid-body states, and therefore define the performance variables to be $z = H^f x^f$ where $H^f = C^f$. 


5.3 Reduced Order Model

A Petrov-Galerkin projection is defined to reduce the CFD portion of the dynamics model. This projection is defined by first computing a basis matrix $V \in \mathbb{R}^{n_{\text{CFD}} \times n_r}$ for a low-dimensional ($n_r \ll n_{\text{CFD}}$) subspace to approximate the fluid solution. We then write $w \approx Vw_r + w_{\text{ref}}$ where $w_r$ is the reduced order fluid state and $w_{\text{ref}}$ is a constant reference state used to help to better condition $w_r$ (the glideslope trajectory equilibrium fluid state $w_0$ is a good choice for $w_{\text{ref}}$). The perturbation quantity $\delta w$ is then approximated as simply $\delta w \approx V\delta w_r$. Substituting this approximation into the model (5.3) and left multiplying the fluid equations by the transpose of a left-basis matrix $W$ gives:

$$\dot{x} = Ax + Bu, \quad y =Cx,$$

(5.5)

where $x = [\xi^T, \dot{\xi}^T, \delta w_r^T]^T$ and:

$$A = \begin{bmatrix} 0 & I & 0 \\ M_0^{-1}E_0 & 0 & M_0^{-1}P_0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -M_0^{-1}B_0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \end{bmatrix}.$$  

One simple choice for the left basis matrix is $W = A_\xi^T V$. Additionally, the basis matrix $V$ can be chosen such that $V^T A_\xi V = I$ to further simplify these equations.

The most challenging part of model order reduction is determining an appropriate subspace to be defined by the range of $V$ (and sometimes $W$). Proper orthogonal decomposition (POD) is used in this chapter, where the snapshot dataset is chosen to include a combination of time and frequency-domain snapshots. This procedure is described in detail in [McClellan et al., 2020], which also includes a post-processing step to ensure the resulting ROM is stable by specially designing the left-basis matrix $W$.

5.4 Simulation Experiments

We now demonstrate the performance of the linear ROMPC scheme from Chapter 2 for the glideslope tracking problem. The mAEWing2 UAV (Figure 5.4) is used for simulations, where the CFD mesh contains $N = 199,786$ nodes and therefore has a dimension $n_{\text{CFD}} = 998,930$. Combined with the rigid-body dynamics model, the coupled rigid-body/CFD model has a dimension $n_f = 998,942$. The nonlinear model is then linearized about a glideslope with a flight path angle of $\gamma_0 = -3.5^\circ$, and velocity of $V_0 = 30.48 \text{ m/s}$, and an angle of attack of $\alpha_0 = 1.92^\circ$. A reduced order model is
generated using proper orthogonal decomposition (POD), and the reduced order CFD model has a dimension \( n_r = 11 \), such that the combined rigid-body/CFD ROM has a dimension \( n = 23 \).

The ROMPC controller is defined with a cost function that mainly penalizes the rigid-body states and controls, and considers the constraints on the rigid-body states shown in Tables 5.1 and 5.2 and constraints on the controls shown in Table 5.3. To avoid having to discretize the coupled rigid-body/CFD model in time before designing the controller, we use the continuous time ROMPC approach discussed in Section 2.7. Specifically, we compute the gains \( K \) and \( L \) using the continuous-time reduced order Riccati approach where we solve the equations (2.28). The error bounds that are used for constraint tightening are computing using the continuous-time approach discussed in Section 3.2.4. Due to the extreme high-dimensionality of the CFD model, it is not possible to compute the bound \( \Delta^{(1)} \) since it is infeasible to compute the parameter \( \beta \) needed to define the decay of the norm of the closed-loop error dynamics (see Section 3.2.4 for definition of \( \beta \)). However, by design of the error bounding approach, we can still compute highly accurate bounds on the total error by making sure that the time horizon \( \tau \) is chosen to be large enough that \( \Delta^{(1)} \) becomes insignificant. For this analysis, we chose a time horizon \( \tau = 70 \) seconds to be extremely conservative since the matrix norm \( \| E e^{A \tau} B \| \approx 10^{-24} \) (see Section 3.2.3 for a more thorough discussion). The resulting error bounds lead to constraint tightening reported in Tables 5.1, 5.2, and 5.3 where we report the amount of constraint tightening as a percentage of the original constraint. Therefore a smaller percentage means less constraint tightening was required, and therefore the error bound was smaller.

From these error bounds a couple of interesting things can be seen. First, the pitch rates and the roll rates seem to have a slightly higher error than the yaw rate, but this could be expected since the aircraft generally has more control authority in those axes. Second, the error associated with lateral deviations (\( \delta p_y \)) is significantly lower than in the longitudinal direction, perhaps again because there is more control authority in the longitudinal plane. However, it also possible that the POD training procedure for building the ROM was just better tuned to capture the lateral motion than the longitudinal motion. This would be interesting to explore in further detail.

The continuous-time version of the ROMPC scheme that we implemented for the simulation is given in Algorithm 4, where we used a time-step of \( \delta t = 0.01 \) seconds for the simulation and discretize...
the optimal control problem with a sampling time of $\Delta t = 0.05$ seconds. The UAV was started at an equilibrium condition with a positional offset in both longitudinal and lateral directions. Simulation results for the UAV longitudinal and lateral rigid-body state are shown in Figures 5.5 and 5.6, and the ROMPC control is shown in Figure 5.7. As expected, the ROMPC controller successfully drives the high-dimensional simulation of the UAV to track the desired glideslope trajectory (i.e. the states all converge to the origin). These figures show both the reference ROM trajectory (dashed lines) as well as the trajectory of the simulated high-dimensional system (solid lines). We note that the ROMPC controller does a good job of driving the high-dimensional system to track the reference ROM trajectory. This simulation also demonstrates the effectiveness of our proposed approach for error bounding, since the $\delta \dot{\theta}_y$ response activates the tightened constraint at the beginning of the simulation (right-hand plot of Figure 5.5). As we can see, constraint tightening based on the computed error-bounds successfully guarantees that the simulated system does not violate the original constraint, even though some tracking error is observed.

5.5 Flight Testing

Flight experiments were also performed to supplement the simulation results from the previous section to show that the proposed approach can be practically implemented for real-time control on embedded hardware. The fixed-wing UAV shown in Figure 5.8 was developed for the flight tests, and includes a Pixhawk 4 autopilot computer running the PX4 software stack for primary flight functions, as well as an Odroid XU4 companion computer that runs the CFD-based ROMPC control scheme (leveraging the Robot Operating System (ROS))\footnote{https://github.com/StanfordASL/asl_fixedwing}. The Odroid XU4 has a Samsung

| Original Constraint | $|\delta p_x|$ | $|\delta p_z|$ | $|\delta \theta_x|$ | $|\delta \theta_z|$ | $|\delta \dot{p}_x|$ | $|\delta \dot{p}_z|$ | $|\delta \dot{\theta}_x|$ | $|\delta \dot{\theta}_z|$ |
|---------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 20 m                | 20 m            | 10°             | 10 m/s          | 5 m/s           | 10°/s           |
| 4.4 m               | 3.9 m           | 3.0°            | 1.8 m/s         | 1.9 m/s         | 3.1°/s          |
| 22.0%               | 19.5%           | 30.3%           | 17.9%           | 37.8%           | 31.4%           |

Table 5.1: Constraints applied to the longitudinal rigid-body states for the mAEWing2 shown in Figure 5.4 for the simulation results discussed in Section 5.4. Bounds on tracking error induced by model reduction and state estimation error are also shown, as well as the percentage by which the original constraints were tightened based on the computed bounds.

| Original Constraint | $|\delta p_y|$ | $|\delta \theta_y|$ | $|\delta \theta_z|$ | $|\delta p_y|$ | $|\delta \theta_y|$ | $|\delta \theta_z|$ |
|---------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 20 m                | 10°             | 10°             | 10 m/s          | 10°/s           | 10°/s           |
| 0.3 m               | 2.4°            | 2°              | 0.1 m/s         | 1.5°/s          | 3.1°/s          |
| 1.3%                | 24.1%           | 20.2%           | 1.3%            | 15.2%           | 30.6%           |

Table 5.2: Constraints applied to the lateral rigid-body states for the mAEWing2 shown in Figure 5.4 for the simulation results discussed in Section 5.4. Bounds on tracking error induced by model reduction and state estimation error are also shown, as well as the percentage by which the original constraints were tightened based on the computed bounds.
CHAPTER 5. AIRCRAFT CONTROL USING CFD-BASED MPC

Table 5.3: Control constraints for the simulation results discussed in Section 5.4. Bounds on tracking error induced by model reduction and state estimation error are also shown, as well as the percentage by which the original constraints were tightened based on the computed bounds.

<table>
<thead>
<tr>
<th></th>
<th>δT</th>
<th>δM_y</th>
<th>δM_z</th>
<th>δM_z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Constraint</td>
<td>100 N</td>
<td>50 Nm</td>
<td>50 Nm</td>
<td>50 Nm</td>
</tr>
<tr>
<td>Error Bound</td>
<td>26.5 N</td>
<td>1.2 Nm</td>
<td>8.7 Nm</td>
<td>1.2 Nm</td>
</tr>
<tr>
<td>Constraint Tightening</td>
<td>26.5%</td>
<td>2.4%</td>
<td>17.5%</td>
<td>2.4%</td>
</tr>
</tbody>
</table>

Exynos5422 processor that features four Cortex-A15 cores running at 2.0GHz and four Cortex-A7 cores running at 1.4GHz, and has a total of 2GB RAM. The quadratic programming problem that defines the ROMPC optimal control problem was solved on the Odroid in real-time using the qpOASES solver [Ferreau et al., 2014]. The UAV is controlled via a pusher propeller, a pair of anti-symmetrical ailerons, an elevator, and a rudder, and has a mass of \( m = 2.47 \text{ kg} \). An onboard GPS, airspeed sensor, and IMU provide raw measurement data, which is fused to generate rigid-body state estimates by an EKF implemented in the PX4 software. We then use the resulting EKF rigid-body estimates as measurements for our ROMPC control scheme.

Development of the CFD model for this UAV consisted of several steps. First, a laser scan of the UAV was used to generate an accurate 3D model (Figure 5.9). Next, a mesh with \( N = 4,501,041 \) was generated to semi-discretize the PDE model (Figure 5.10). Finally, the semi-discretization lead to a CFD model with \( n_{\text{CFD}} = 27,006,246 \) degrees of freedom. For this application we have also slightly modified the CFD model outlined at the beginning of this chapter. Specifically, the CFD model was extended to include a Spalart-Allmaras turbulence model, which adds an additional degree of freedom to each node of the mesh (i.e. \( n_{\text{CFD}} = 6N \)). Figure 5.11 shows a characteristic velocity and vorticity plot resulting from the CFD model.

The rigid-body dynamics model was also modified for the flight experiments by removing the rotational dynamics equations, and using the control input vector \( u_{\text{ctrl}} = [T, \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z]^T \). In other words, the new model consists of the translation equations, plus the integrating kinematics of the attitude parameters \( \theta_x, \theta_y, \theta_z \), but it is assumed that attitude rate commands are applied to the aircraft. This modification was made because in practice it is not easy to apply body-moment commands (i.e. \( M_x, M_y, M_z \)) when the actual commanded quantities are the deflection angles of the aileron, elevator, and rudder. By switching the control input to be the attitude rates, a low-level PID controller implemented in the PX4 flight software can be used to convert the attitude rate commands into control surface deflection commands. However, additional work should be done to remove this modification by instead explicitly modeling the control surfaces in the CFD model, which would enable direct control of the UAV control surface deflection angles, but which requires more advanced CFD modeling and additional effort in generating good reduced order models. Another point to note is that the rigid-body/CFD model uses thrust inputs, therefore a thrust model must

---

\[2\text{Development of the CFD model was performed by Andrew McClellan and Professor Charbel Farhat, with some additional help on generating the CFD mesh from Charbel Bou-Mosleh.}\]
be used to map thrust commands to propeller motor speeds that can be implemented by the flight computer. Unfortunately, generating a thrust mapping can be quite difficult due to complex propeller aerodynamics. While future work could incorporate propeller aerodynamics in the CFD model, in this flight experiment we used a combination of static thrust-stand data and propeller manufacturer data to fit a simple thrust model of the form:

\[ T = c_{T,0}(1 - c_{T,V\omega} \frac{V}{u_T})u_T^2, \]

where \( V \) is the current airspeed seen by the propeller (assumed to be the UAV’s x-axis velocity), and \( u_T \) is a normalized motor command. This simple model is used to approximately account for the variation of thrust with airspeed and propeller spin rate, but certainly generates another source of error in the experiment.

The flight experiment presented in this thesis is a steady wings-level flight scenario, where the aircraft should fly in a straight line with constant altitude and velocity of \( V_0 = 15 \) m/s (i.e. a glideslope with flight path angle \( \gamma_0 = 0^\circ \)). The CFD model and the rigid-body dynamics model were both linearized about this trajectory, where the nonlinear CFD model was used to compute the equilibrium angle of attack of \( \alpha_0 \approx 2.0^\circ \). Proper orthogonal decomposition was again used to compute the reduced order CFD model, which had a dimension \( n_r = 74 \), such that the dimension of the combined rigid-body/CFD ROM was \( n = 83 \). The dimension of the ROM is significantly compressed, but additional computational speed-ups when solving the online quadratic programs is possible by reformulating the optimal control problem with a single-shooting approach, where only the control parameters are variables in the optimization problem. This enabled real-time control by enabling the optimal control problems to be solved by qpOASES in just a few milliseconds or less.

The ROMPC controller was defined with just control constraints to simplify the optimal control problem, and as in the simulation experiments, the gain matrices \( K \) and \( L \) were computed using the continuous time Riccati equations. Since the primary motivation and purpose of these flight experiments was to demonstrate the practicality of the approach (i.e. to show that ROMs could be used for MPC on resource-constrained hardware) the theoretical error bounds were not computed for this case. Flight data is shown in Figures 5.13, 5.14, 5.15, 5.16, and 5.17. First, Figure 5.13 shows the ground track of the aircraft starting from the moment the controller was engaged. As can be seen, the aircraft starts out with some yaw error that is corrected. Figures 5.14, 5.15, and 5.16 show the relative velocity, position, and Euler angle errors with respect to the nominal target trajectory. The relative velocity starts with a non-zero initial condition, but converges to near-zero values as desired. The position and Euler angle errors converge to small constant offsets, which is expected since any form of constant disturbance (such as wind during the flight or mismatch between predicted and actual aerodynamic drag) will lead to non-zero steady state error without the inclusion of disturbance estimation and integral action in the controller. Finally, Figure 5.17 shows the control commands applied to the aircraft. As mentioned before, the thrust commands are
mapped to motor commands via a simple nonlinear mapping, and the angular velocity commands
are converted into aileron, elevator, and rudder commands through a low-level PID control loop
implemented by the PX4 autopilot. Note that near steady state the commanded thrust value is
between 5 and 6 Newtons. The equilibrium thrust computed by the CFD model was $T_0 = 2.36$
Newtons, which suggests that there were a significant amount of disturbances affecting the system,
likely due to a combination of increased drag from wind\footnote{There was some wind during these flight tests but the actual wind speed was not known.} and errors in the thrust model or CFD
model. In particular, the thrust model is likely the main culprit, since propeller aerodynamics are
rather complex. Additional work could also look into incorporating a reduced order propeller CFD
model to avoid the need to perform a mapping from command to thrust, or even simply additional
experimental testing to better characterize the thrust model.

In summary, these flight tests have been used to demonstrate the \textit{computational practicality} of
the proposed approach. With the practicality proven, we can motivate future work that is necessary
to improve \textit{performance} beyond the state-of-the-art for the glideslope tracking problem, such as
incorporating wind estimation and better thrust modeling. Most importantly, these flight tests also
motivate continued work to look at nonlinear CFD and rigid-body dynamics models such that we
can truly take advantage of the unsteady aerodynamic modeling offered by CFD, which is where the
most significant advantages of this line of work will be seen.

\textbf{5.6 Conclusion and Future Work}

In this chapter, we have demonstrated the proposed linear ROMPC controller from Chapter\footnote{There was some wind during these flight tests but the actual wind speed was not known.} within
the context of autonomous aircraft control. This is a particularly exciting application of reduced
order modeling, since high-fidelity computational fluid dynamics models can be extremely high-
dimensional, but can enable increasingly higher-performance flight through unsteady aerodynamics
modeling. The performance of the linear ROMPC controller was demonstrated in both simulation
and flight experiments. Simulation experiments demonstrated the theoretical properties of the
control scheme (i.e. stability and constraint satisfaction guarantees, even in the presence of model
reduction error), while the flight experiments demonstrated the practicality of the approach to real-
world settings. To the best of our knowledge, the flight experiments presented here constitute the
first use of a (reduced order) CFD model for real-time flight control.

The flight experiments have also demonstrated the large amount of work that still needs to be
done to bridge the gap between theory and practice. Currently, the theoretical analysis has been
performed only for the \textit{linear} setting. Even though real-world systems, such as the UAV studied in
this chapter, are nonlinear, the theoretical developments based on linear assumptions are important
for two main reasons:
1. The error bounds provide assurance that the model order reduction process resulted in a high-quality approximation of the high-dimensional model. Specifically, they provide a control-oriented metric to describe how much is lost by model order reduction. While these bounds fail to hold in the nonlinear setting, they provide a useful validation that the linear controller should perform comparably to a high-dimensional linear controller.

2. They provide a good foundation to extend the results to the nonlinear setting, and to provide a realistic expectation of the types of properties that should be expected in the nonlinear case. Important lessons have also been learned by exploring which parts of the theoretical analysis pose the most significant challenges, as well as solutions that could be leveraged in future work.

Specifically for aircraft flight control, extensions to the use of nonlinear rigid-body and nonlinear CFD models are perhaps the most exciting, since this is necessary to control aircraft in high-performance flight settings where unsteady aerodynamics are more important to model. As mentioned in the discussion on the flight test results, modeling the UAV control surfaces and propeller dynamics in the CFD model is another important area of work. Finally, it would also be interesting to explore how the proposed CFD-based ROMPC scheme could be leveraged within the context of multi-disciplinary design optimization to optimize performance of novel aircraft (e.g. highly maneuverable or highly flexible aircraft) based on their ability to be controlled.
Figure 5.5: Longitudinal motion from the CFD-based simulation of the mAEWing2 aircraft discussed in Section 5.4. As can be seen, each of the longitudinal components of the rigid-body motion converge to zero, indicating the controller successfully tracks the desired glideslope trajectory. The dashed lines are associated with the reference ROM trajectory that is computed by the ROMPC controller, and the solid lines indicate the trajectory of the high-dimensional controlled system. Note that the constraint on $\dot{\theta}_y$ becomes active at the beginning of the simulation. The solid black line shows the original constraint of $10^2 \text{s}^{-1}$, and the black dashed line represents the tightened constraint that accounts for the potential tracking error. This demonstrates the effectiveness of our proposed approach for error bounding and constraint tightening. While the constraint tightening appears conservative in this case, this is generally expected since the error bounds are computed using a worst-case analysis.

Figure 5.6: Lateral motion from the CFD-based simulation of the mAEWing2 aircraft discussed in Section 5.4. As can be seen, each of the lateral components of the rigid-body motion converge to zero, indicating the controller successfully tracks the desired glideslope trajectory. The dashed lines are associated with the reference ROM trajectory that is computed by the ROMPC controller, and the solid lines indicate the trajectory of the high-dimensional controlled system.
5.6. CONCLUSION AND FUTURE WORK

As can be seen, the control actions converge to zero as the UAV successfully tracks the desired glideslope trajectory. The dashed lines are associated with the control actions corresponding to the reference ROM trajectory that is computed by the ROMPC controller, and the solid lines indicate the actual control applied to the high-dimensional system.

Figure 5.8: This UAV was used to perform flight tests of the CFD-based ROMPC control scheme.

Figure 5.9: The UAV shown in Figure 5.8 was laser scanned to generate an accurate 3D model, which was then used to developed the computational fluid dynamics model of the UAV’s aerodynamics.
Figure 5.10: The 3D model (Figure 5.9) of the UAV was used to generate a 3D mesh that semi-discretizes the PDE model. This particular mesh consists of \( N = 4,501,041 \) nodes. The image on the left shows the nodes of the mesh on the surface of the UAV, with a zoomed-in section showing how fine the mesh resolution is. On the right is an image showing a slice of the mesh down the centerline of the UAV, which shows how the mesh has higher density nearer the body where better resolution is needed for accuracy. Images courtesy of Andrew McClellan.

Figure 5.11: Images showing characteristic velocity (left) and vorticity (right) of the aerodynamics, as computed by the CFD model. Images courtesy of Andrew McClellan.

Figure 5.12: An image of the UAV just before flight.
5.6. CONCLUSION AND FUTURE WORK

Figure 5.13: Ground track results from the flight experiments of the CFD-based ROMPC controller. The target trajectory is a steady wings-level flight with constant altitude and speed. Here it is seen that when the controller was engaged there was an initial deviation that was then corrected.

Figure 5.14: Flight test data from a flight of the UAV in Figure 5.8 controlled by the CFD-based ROMPC scheme. This figure shows the relative velocity with respect to the target steady wings-level trajectory. The controller was able to regulate the system to drive the relative velocity close to zero.
CHAPTER 5. AIRCRAFT CONTROL USING CFD-BASED MPC

Figure 5.15: Flight test data from a flight of the UAV in Figure 5.8, controlled by the CFD-based ROMPC scheme. This figure shows the relative position with respect to the target steady wings-level trajectory. The system seems to converge with some steady-state errors that are likely caused by constant unmodeled effects such as wind, errors in the thrust model, or even asymmetries in the aircraft not captured by the CFD model. However, these errors are relatively small since the aircraft is moving at roughly the equilibrium velocity of 15 m/s.

Figure 5.16: Flight test data from a flight of the UAV in Figure 5.8, controlled by the CFD-based ROMPC scheme. This figure shows the Euler angles with respect to the target steady wings-level trajectory. The system seems to converge with some small steady-state errors that are likely caused by constant unmodeled effects such as wind, errors in the thrust model, or even asymmetries in the aircraft not captured by the CFD model. In particular, there is a small offset in the roll ($\delta \phi$) and yaw ($\delta \psi$).
Figure 5.17: Flight test data from a flight of the UAV in Figure 5.8 controlled by the CFD-based ROMPC scheme. This figure shows the total control (equilibrium + ROMPC control) applied to the system. The top plot shows the commanded thrust values (which are mapped to propeller motor commands via a simple thrust model), and the bottom plot shows the commanded body-axis angular velocities: roll rate ($p$), pitch rate ($q$), and yaw rate ($r$). These angular velocity commands are converted into aileron, elevator, and rudder deflection angles by a low-level PID controller implemented in the PX4 autopilot software stack.
Part II

Nonlinear Reduced Order Model
Predictive Control
Chapter 6

Nonlinear Reduced Order Model Predictive Control

Even though real-world engineering systems exhibit nonlinear dynamical behavior, the use of linear models is prevalent and sufficient in many control applications. However, there are also many settings where the system’s nonlinearities are significant enough to have an important impact on controller performance, and thus nonlinear models should be used. Constrained optimal control is one setting where nonlinear model predictive control is useful, since nonlinear effects are generally more significant when the system is operating at its performance limits. Even in settings where the dynamics model is low-dimensional, its nonlinearity generally makes the resulting optimization problem non-convex and thus more challenging to solve. In this chapter, we consider constrained optimal control problems where the system dynamics model is not only nonlinear, but is also high-dimensional (e.g. the model is the result of semi-discretizing a nonlinear PDE model). Within this problem setting, we propose a nonlinear output feedback model predictive control scheme that again leverages model order reduction techniques to address the issue of online computational complexity.

6.1 Introduction

Direct methods for solving optimal control problems in the context of linear MPC typically result in convex linear or quadratic programs, for which efficient and robust numerical methods can find the global optimum. In the case of nonlinear MPC, the discretized optimal control problems are generally non-convex nonlinear programs and numerical methods can only return local optima. Two popular numerical approaches to solving these optimization problems include sequential convex programming (SCP) and interior point methods. In this chapter, we use an SCP approach to solving the nonlinear optimal control problem that is used within the proposed MPC framework.
SCP algorithms solve nonlinear programs by sequentially forming and solving convex approximations (generally quadratic programs) to iteratively improve the solution \cite{NocedalWright2006}. While SCP algorithms have been shown to be useful within the context of nonlinear MPC and trajectory optimization \cite{Diehl2009, Lew2020, Liu2014, Mao2016, Bonalli2019}, they are generally only applicable to real-time control applications where the system dynamics model is relatively low-dimensional. This limitation arises because not only is the computational complexity for solving each convex approximation dependent on the state dimension, but so is the computation of the dynamics model Jacobians (which must be recomputed at each iteration to form the convex approximation). In fact, the online computation of the Jacobians alone could easily render an implementation of the nonlinear MPC scheme intractable for extremely high-dimensional systems. For example, a high-fidelity nonlinear semi-discretized PDE model may have thousands to millions of equations and it would be challenging to even store the model’s Jacobians in memory, let alone use them to formulate an optimization problem.

Similar to our proposed approach for linear ROMPC in Chapter 2, a computationally efficient SCP-based nonlinear ROMPC scheme can be synthesized in high-dimensional problem settings by using model order reduction techniques to generate an approximate low-dimensional reduced order model. As in the linear ROMPC case, we again consider Petrov-Galerkin projection-based methods, which as previously mentioned use left and right basis matrices, \( W \in \mathbb{R}^{n_f \times n} \) and \( V \in \mathbb{R}^{n_f \times n} \), to define a projection matrix \( P = V(V^TW)^{-1}W^T \). In the nonlinear setting, we use a slightly more general projection where the projection of the high-dimensional state \( x_f \in \mathbb{R}^{n_f} \) and its approximate reconstruction are given by:

\[
x = (V^TW)^{-1}W^T(x_f - x_f^{ref}), \quad x_f \approx Vx + x_f^{ref},
\]

where \( x \in \mathbb{R}^n \) is the reduced order state and \( x_f^{ref} \in \mathbb{R}^{n_f} \) is a constant reference state used to better condition the basis matrices. Unfortunately, the nonlinearity of the dynamics model again introduces new challenges that are not present in the linear model setting. Consider the nonlinear model:

\[
\dot{x}_f = f_f(x_f, u),
\]

where \( u \in \mathbb{R}^m \) is the control input. The application of the Petrov-Galerkin projection to this system yields the nonlinear ROM:

\[
\dot{x} = f(x, u) := (V^TW)^{-1}W^f f(Vx + x_f^{ref}, u).
\]

While technically this is a low-dimensional set of ordinary differential equations, the evaluation of the right-hand side requires reconstructing the high-dimensional state and then evaluating a high-dimensional function, which can negate any computational benefit from having performed the state
reduction. Similarly, the state Jacobian of the reduced order model is given by:

$$\nabla_x f(x, u) = (V^TW)^{-1}W^T \nabla_x f(Vx + x_{ref}, u)V,$$

which requires evaluation of the high-dimensional Jacobian at the reconstructed high-dimensional state, followed by a reduction via projection. Again, even though the Jacobian is low-dimensional, evaluating it still requires high-dimensional operations. Only reducing the state dimension is therefore not sufficient to enable the use of SCP for real-time nonlinear MPC in high-dimensional problem settings since many function and Jacobian evaluations are required for each solve.

This computational issue is addressed in nonlinear model order reduction methods by performing an additional approximation of the high-dimensional nonlinear function (in addition to the state approximation). Well known function approximation methods generally fall under two categories: hyperreduction methods and linearization methods. A number of hyperreduction methods have been proposed, including gappy POD (Everson and Sirovich [1995]), the discrete empirical interpolation method (DEIM) (Chaturantabut and Sorensen [2010]), the energy conserving sampling and weighting (ECSW) method (Farhat et al. [2015]), and others. These methods generally achieve computational efficiency improvements by evaluating a small number of entries of the high-dimensional nonlinear function and approximately reconstructing the rest. The disadvantage of such approaches is that an online implementation is software-intrusive (i.e. it generally requires specialized software for evaluating the high-dimensional function), which could be non-trivial for some high-fidelity computational models (e.g. models resulting from the semi-discretization of PDEs). On the other hand, linearization methods such as the trajectory piecewise-linear (TPWL) method (Rewieński and White [2006]) and the maniMOR method (Gu and Roychowdhury [2008]) select linearization points in the state space and precompute the reduced order function values and their Jacobians at these points. The use of these stored values generally results in a simpler online implementation than in hyperreduction methods, but with the cost that the accuracy of the model may suffer in regions of the state space far from the selected linearization points. Some data-driven or model-free approaches have also been proposed, including the use of dynamic mode decomposition (Alla and Kutz [2017]) for approximating the nonlinear function or by representing the nonlinear behavior through learned Elman neural networks (Xie et al. [2015]). The disadvantages of data-driven and model-free methods are that it is unclear how easily it would be to compute the Jacobians required for SCP, and that the modeling accuracy can suffer from the loss of the structure inherent in the original high-dimensional model.

Model order reduction techniques have also been applied to nonlinear model predictive control, however this field is significantly less developed than in the linear setting. Several approaches use data-driven nonlinear ROMs for MPC, (Xie et al. [2015] Friedl et al. [2018]), but as previously mentioned, the use of data-driven models introduces a new set of challenges. Work by (Rivotti et al. [2012]) uses model order reduction but skips the nonlinear function approximation step. They instead...
leverage explicit nonlinear MPC to shift all computationally demanding work offline and achieve fast online runtimes. However, without nonlinear function approximation (e.g. hyperreduction or linearization-based methods) the offline synthesis of the controller still suffers from scalability issues. A high-dimensional nonlinear structural dynamics model is used for flexible aircraft MPC in [Wang et al., 2016], but the nonlinear ROM used for MPC is just a second order Taylor expansion reduced via balanced truncation, and not a full nonlinear ROM. One approach that does use modern nonlinear model order reduction methods is [Xie et al., 2011], which uses linearization-based TPWL ROMs for nonlinear MPC. However, they do not consider state or control constraints and they also assume full state feedback, which is not generally a practical assumption for high-dimensional systems.

In this chapter, we propose an approach for output feedback nonlinear ROMPC that leverages a piecewise-affine reduced order model (i.e. uses the TPWL linearization approach) and solves the constrained optimal control problem by sequential convex programming. The piecewise-affine model reduction method was chosen for its ability to generate an online model whose Jacobians can be computed in a trivial fashion, greatly simplifying the implementation of the SCP algorithm. Output feedback control is enabled by including a state estimator based on the piecewise-affine ROM to generate reduced order state estimates from measurements. The optimal control formulation is capable of addressing a variety of dynamic control tasks including setpoint and trajectory tracking.

6.2 Model Description and Problem Formulation

Similar to Chapter 2, this chapter considers control applications where the system dynamics model is high-dimensional, such as when the model is a finite-dimensional approximation of an infinite-dimensional system. Unlike Chapter 2, this chapter considers nonlinear continuous-time models.

6.2.1 Nonlinear Full Order Model

We consider the continuous-time full order nonlinear dynamics model (FOM):

\[ \dot{x}^f = f^f(x^f, u), \quad y = C_y^f x^f, \quad z = C_z^f x^f, \]  

(6.1)

where \((\cdot)^f\) is used to denote a full order (high-dimensional) variable or function, \(x^f(t) \in \mathbb{R}^{n^f}\) is the full order state, \(u(t) \in \mathbb{R}^m\) is the control input, \(y(t) \in \mathbb{R}^p\) represents measured variables, and \(z(t) \in \mathbb{R}^o\) represents performance output variables. The measurements \(y\) are used to generate an estimate of the system’s state, and the performance variables are (not necessarily observable) quantities that will be used to define the system’s performance cost function and constraints. Note that in many practical high-dimensional problems the dimension \(o\) of the performance output variables will be much smaller than the state dimension \(n^f\). This is due to the fact that control problems are generally focused on specific aspects of the system rather than on the entire state. This is particularly
important for constrained optimal control as a smaller number of performance variables will require a smaller number of constraints in the optimal control problem. For simplicity in presentation we have assumed the measurement and output models are linear, but it is relatively straightforward to extend the approach to the nonlinear setting.

6.2.2 Constrained Optimal Control Problem

The control objective is to define an output feedback controller to minimize a cost function while ensuring the system satisfies constraints on the performance variables \( z \) and control \( u \). We consider a finite-horizon cost of the form:

\[
J^f = \| \delta z(t_f) \|_{Q_z}^2 + \int_{t_0}^{t_f} \| \delta z(t) \|_{Q_z}^2 + \| \delta u(t) \|_{R}^2 \, dt, \tag{6.2}
\]

where \([t_0, t_f]\) defines the time horizon, \( Q_z, P_z \in \mathbb{R}^{o \times o} \) are positive semi-definite performance variable cost matrices and \( R \in \mathbb{R}^{m \times m} \) is a positive definite control cost matrix. The terms \( \delta z(t) = z(t) - z_d(t) \) represent deviations of the performance variables with respect to a desired target trajectory \( z_d(t) \), and \( \delta u(t) = u(t) - u_d(t) \) represents deviations from a desired target input \( u_d(t) \). For example, in trajectory tracking tasks \( z_d(t) \) defines the reference trajectory (and typically \( u_d(t) = 0 \)), and in regulation tasks \( z_d(t) = z_0, u_d(t) = u_0 \) where \( z_0 \) and \( u_0 \) define the desired equilibrium point.

The set of constraints that are imposed on the performance outputs and the control inputs are defined by:

\[
u \in \mathcal{U}, \quad z \in \mathcal{Z}, \tag{6.3}
\]

where \( \mathcal{U} := \{ u \mid H_u u \leq b_u \} \) and \( \mathcal{Z} := \{ z \mid H_z z \leq b_z \} \) are convex polytopes with \( H_u \in \mathbb{R}^{m \times m} \) and \( H_z \in \mathbb{R}^{n_z \times o} \).

The full definition of the nonlinear constrained optimal control problem (OCP) is:

\[
\begin{align*}
\text{minimize} & \quad J^f, \\
\text{subject to} & \quad \dot{x}^f = f^f(x^f, u), \quad x^f(0) = x^f_0, \\
& \quad z = C^f z^f, \\
& \quad u \in \mathcal{U}, \quad z \in \mathcal{Z},
\end{align*} \tag{6.4}
\]

where \( x^f_0 \) defines an initial condition. As was previously mentioned, this OCP can be solved using direct methods by first transcribing it into a nonlinear programming problem (NLP), for example by using a multiple-shooting or direct collocation method. The nonlinear program can then be solved numerically by solving a sequential convex programming algorithm. If the OCP can be solved in

\footnote{Since we are using a sequential convex programming approach for solving the optimal control problem it is theoretically possible to extend the formulation to non-convex constraints as well.}
real-time, a receding horizon closed-loop feedback control scheme could be implemented. Yet, as in the linear ROMPC setting, when (6.1) is high-dimensional the OCP (6.4) suffers from computational challenges arising from the large number of decision variables and constraints. In fact, it would be impossible to solve the OCP (6.4) in real-time with existing state-of-the-art optimization algorithms. To address this computational issue, we propose to leverage a high-fidelity, but low-dimensional, piecewise-affine ROM to approximate the original high-dimensional FOM.

6.2.3 Piecewise-Affine Reduced Order Model

As discussed in Section 6.1, several nonlinear model order reduction techniques have been developed. In this chapter, we use a linearization-based method originally proposed by Rewienski and White (2006) that generates a piecewise-affine ROM whose Jacobians can be computed trivially. The ability to quickly compute the Jacobians is crucial for efficient use of numerical algorithms for solving optimal control problems, (e.g. sequential convex programming, which recomputes the Jacobians at every iteration to form a new convex approximation of the NLP). Nonetheless, interesting future work includes the exploration of alternative nonlinear model reduction techniques which could provide better modeling accuracy. Some future considerations on this topic are discussed in Section 6.4.

As was discussed in Section 6.1, a Petrov-Galerkin projection is first used to compress the state dimension such that a model with a reduced number of equations is given by:

\[
\begin{align*}
\dot{x} &= (V^T W)^{-1} W^T f(V x + x_{\text{ref}}, u), \\
y &= C_y x + y_{\text{ref}}, \\
z &= C_z x + z_{\text{ref}},
\end{align*}
\]

where \(x \in \mathbb{R}^n\) is the reduced order state, and the reduced order measurement and output model matrices are given by \(C_y = C_y^f V\) and \(C_z = C_z^f V\). The computational complexity of evaluating the nonlinear function \(f^f\) is then reduced by approximating the terms as piecewise-affine functions based on a set of linearization points. Specifically, the nonlinear functions are linearized around \(N_{\text{lin}}\) points, \((\tilde{x}_i, \tilde{u}_i)\), where the states, \(\tilde{x}_i\), are represented in the high-dimensional space as \(\tilde{x}_i^f = V \tilde{x}_i + x_{\text{ref}}^f\). Around each of these linearization points the nonlinear functions are approximated by a first-order Taylor expansion:

\[
f^f(x^f, u) \approx f^f(\tilde{x}_i^f, \tilde{u}_i) + \nabla_x f^f(\tilde{x}_i^f, \tilde{u}_i)(x^f - \tilde{x}_i^f) + \nabla_u f^f(\tilde{x}_i^f, \tilde{u}_i)(u - \tilde{u}_i).
\]

Thus in the vicinity of each linearization point \(\tilde{x}_i\):

\[
\dot{x} \approx A_i x + B_i u + d_i,
\]
where:

\[
A_i = (V^TW)^{-1}W^T \nabla_x f(\tilde{x}_i, \tilde{u}_i)V,
\]

\[
B_i = (V^TW)^{-1}W^T \nabla_u f(\tilde{x}_i, \tilde{u}_i),
\]

\[
d_i = (V^TW)^{-1}W^T f(\tilde{x}_i, \tilde{u}_i) - A_i \tilde{x}_i - B_i \tilde{u}_i,
\]

Combined, the \(N_{\text{lin}}\) linearization points provide a global approximation of the nonlinear dynamics via the piecewise-affine ROM:

\[
\dot{x} = A_ix + B_iu + d_i, \quad i = \arg \min_{j \in \{1, \ldots, N_{\text{lin}}\}} \|x - \tilde{x}_j\|_M,
\]

where \(M \in \mathbb{R}^{n \times n}\) is a positive semi-definite weighting matrix that defines a distance metric for determining the nearest linearization point. In [Rewieński and White (2006)], the piecewise-affine model uses an interpolation among the different linearization points, but we found that simply choosing the single nearest linearization point provided better good results ([Rewieński and White (2006) noted similar results]). The added benefit of not using interpolation is that computing \(i = \arg \min_{j \in \{1, \ldots, N_{\text{lin}}\}} \|x - \tilde{x}_j\|_M\) can be accomplished faster than computing \(\sum_{i=1}^{N_{\text{lin}}} w_i(x)A_i, \sum_{i=1}^{N_{\text{lin}}} w_i(x)B_i,\) and \(\sum_{i=1}^{N_{\text{lin}}} w_i(x)d_i\) where \(w_i \in \mathbb{R}\) are weights that sum to one.

In practice it is up to the engineer to determine the appropriate number of linearization points, \(N_{\text{lin}}\), as well as a method for choosing where in the state space they are located. A relatively straightforward approach is to simulate the system under a set of randomized and representative control inputs and choosing linearization points along the resulting trajectory. A specific algorithm for choosing the linearization points that follows this general approach is introduced in the context of finite element model-based soft robot control in Chapter 7. This automated approach is based on an iterative procedure that continues to collect linearization points whenever a heuristic for model error becomes too large, based on a user-defined threshold parameter. However, this approach still requires some guess work and iteration by the engineer to determine an appropriate threshold, since too small of a threshold will lead to many linearization points being collected, but too large of a threshold could lead to poor modeling accuracy.
CHAPTER 6. NONLINEAR REDUCED ORDER MODEL PREDICTIVE CONTROL

6.3 Controller Definition

We now leverage the discrete-time ROM (6.6) to formulate a reduced order OCP to approximately solve the original OCP (6.4). An output feedback control scheme is then defined which consists of three components: (1) the reduced order OCP, (2) a reduced order state estimator, and (3) a reduced order feedback control law. In this scheme the reduced order OCP optimizes a reduced order reference trajectory that the system should follow, and the state estimator incorporates measurements to provide an estimate of the system’s current (reduced order) state. The feedback control law then uses the state estimate to drive the system to track the optimized reference trajectory. Note that overall, the form of this control scheme parallels that of the linear ROMPC controller defined in Chapter 2.

6.3.1 Reduced Order Optimal Control Problem

A discretized, optimal (reduced order) reference trajectory \( \bar{x}_k^*, \bar{u}_k^* \) = \( \{ \bar{x}_i^* | k \leq i \leq k+N \}, \{ \bar{u}_i^* | k \leq i \leq k+N-1 \} \) is defined over a finite horizon, \( N \), by solving a reduced order approximation of the high-dimensional OCP (6.4):

\[
(\bar{x}_k^*, \bar{u}_k^*) = \arg \min_{\bar{x}_k, \bar{u}_k} \left\{ \sum_{j=k}^{k+N-1} \| \delta \bar{z}_{j+k} \|^2_R + \| \delta \bar{u}_{j+k} \|^2_R + \| \delta \bar{z}_{j+k} \|^2_Q \right\},
\]

subject to \( \bar{x}_{i+1+k} = g(\bar{x}_{i+k}, \bar{u}_{i+k}) \), \( \bar{x}_{k+k} = \bar{x}_k \),

\[
\bar{z}_{i+k} = C_{z} \bar{x}_{i+k} + z_{\text{ref}},
\]

\( \bar{u}_{i+k} \in U, \quad \bar{z}_{i+k} \in Z, \)

where \( i = k, \ldots, k+N-1 \) and \( \bar{x}_k \) is the initial state at time step \( k \). The cost function is now defined by the terms \( \delta \bar{z}_{j+k} = \bar{z}_{j+k} - z_{d,j} \) and \( \delta \bar{u}_{j+k} = \bar{u}_{j+k} - u_{d,j} \), where \( z_{d,j} \) and \( u_{d,j} \) are the desired target trajectories at time step \( j \). This finite horizon problem is solved in a receding horizon fashion to define the optimal reference trajectory over an arbitrarily long horizon. Specifically, the OCP is initialized at time \( k = 0 \) by setting \( \bar{x}_0 = \hat{x}_0 \), where \( \hat{x}_0 \) is the reduced order state estimate. The OCP is then recursively solved every \( N_r < N \) time steps (i.e. at time steps \( k = N_r, 2N_r, \ldots \)) over the receding horizon \( [k, k+N] \) by setting \( \bar{x}_k = \bar{x}_k^{*|k-N_r} \), where \( \bar{x}_k^{*|k-N_r} \) is the optimal state from the previous solution (computed at time step \( k-N_r \)).

As previously mentioned, the non-convex OCP (6.7) is solved using sequential convex programming by solving a sequence of quadratic program (QP) approximations until convergence. Note that the original cost function (6.2) is quadratic in \( x^f \) and \( u \) and remains quadratic in \( \bar{x} \) and \( \bar{u} \) in the reduced order OCP since the performance output model has been assumed to be linear. Therefore the only non-convexity of the reduced order OCP (6.7) comes from the piecewise-affine ROM \( g \), which by nature is readily linearized by simply selecting the appropriate \( A_i, B_i, \) and \( d_i \) from the database.
6.3. CONTROLLER DEFINITION

of linearizations. Of course it is also possible to include non-convex cost and constraint terms, but this formulation is sufficient for many control tasks and greatly simplifies the implementation of the SCP algorithm.

Note that many receding horizon control schemes choose to use the current state estimate to initialize each OCP problem (i.e. $\hat{x}_k = \hat{x}_k$). However, in our proposed control scheme we incorporate the state estimate $\hat{x}_k$ by combining the optimal open-loop trajectory from the OCP with an ancillary feedback control term (defined in Section 6.3.2). An advantage of this approach is that by interpolating the optimal reference trajectory computed by the OCP the overall control scheme can be run at a faster rate than the OCP can be solved. This is important for real-time control since even SCP algorithms using current state-of-the-art convex optimization solvers may be too slow to run at the desired control frequency. For example, even if the OCP was solved at a rate of 1-10 Hz the combined controller could run at 100 Hz or more.

6.3.2 Reduced Order Controller and State Estimator

The reduced order OCP (6.7) defines an optimized open-loop reference trajectory in the reduced order state space for the system to follow. A simple output feedback control scheme is now defined to drive the system to track this trajectory.

To estimate the system’s current reduced order state from measurements a reduced order state estimator is defined as:

$$\hat{x}_k = g(\hat{x}_{k-1}, u_{k-1}) + L_k(y_k - C_y g(\hat{x}_{k-1}, u_{k-1}) - y_{\text{ref}}), \quad (6.8)$$

where $\hat{x}_k \in \mathbb{R}^n$ is the current reduced order state estimate, $L_k \in \mathbb{R}^{n \times p}$ is the estimator gain, $y_k$ is the current system measurement, and $u_{k-1}$ is the previous control input. We choose the gain $L_k$ to be the extended Kalman filter gain based on the discrete-time ROM dynamics (6.6).

The control law applied to the system is then computed using a combination of open-loop optimal reference trajectory computed by the OCP and a reduced order linear feedback control law:

$$u_k = \bar{u}_k + G_k(\hat{x}_k - \bar{x}_k^*), \quad (6.9)$$

where $G_k \in \mathbb{R}^{m \times n}$ is the controller gain matrix, and $(\bar{x}_k^*, \bar{u}_k^*)$ is the optimized reference trajectory state-input pair defined by the most recent solution to the OCP (6.7). To define the controller gains $G_k$, we propose a simple and computationally efficient method based on gain scheduling:

$$G_k = G_i, \quad i = \arg \min_{j \in \{1, \ldots, N_{\text{lin}}\}} \|\bar{x}_k^* - \hat{x}_j\|_M, \quad (6.10)$$

where the gains $G_i$ are the discrete-time LQR gains at each linearization point $\hat{x}_i$, which can be computed a priori. In particular, the gains $G_i$ are computed using the discrete-time reduced order
dynamics matrices $A_{i,d}$ and $B_{i,d}$, and positive semi-definite cost matrix $Q = C_{z}^{T}Q_{z}C_{z} \in \mathbb{R}^{n \times n}$ and control cost matrix $R$. The offline and online procedures for the proposed nonlinear reduced order MPC scheme are summarized in Algorithms 6 and 7, respectively.

Algorithm 6 Nonlinear Output Feedback ROMPC Synthesis (Offline)
1: procedure NONLINEARROMPCSYNTHESIS
2: Compute model reduction basis matrices $W$ and $V$
3: Select $N_{\text{lin}}$ linearization points $(\tilde{x}_i, \tilde{u}_i)$ and compute ROM matrices $A_i, B_i, d_i$
4: Compute discrete-time ROM matrices $A_{i,d}, B_{i,d}, d_{i,d}$
5: Compute controller gain matrices $G_i$ for each linearization point $(\tilde{x}_i, \tilde{u}_i)$.

Algorithm 7 Nonlinear Output Feedback ROMPC Control (Online)
1: procedure NONLINEARROMPC
2: $k \leftarrow 0, k_{\text{solve}} \leftarrow 0$
3: $\hat{x}_{k-1} \leftarrow \text{initializeEstimator}()$
4: loop
5: if $k \mod N_r = 0$ then
6: $k_{\text{solve}} \leftarrow k$
7: if $k = 0$ then
8: $(\bar{x}_{k_{\text{solve}}}^{*}, \bar{u}_{k_{\text{solve}}}^{*}) \leftarrow \text{solveOCP}(\hat{x}_k)$
9: else
10: $(\bar{x}_{k_{\text{solve}}}^{*}, \bar{u}_{k_{\text{solve}}}^{*}) \leftarrow \text{solveOCP}(\bar{x}_{k-1}|N_r)$
11: $y_k \leftarrow \text{getMeasurement}()$
12: $\bar{x}_k = g(\bar{x}_{k-1}, u_{k-1}) + L_k(y_k - C_y g(\bar{x}_{k-1}, u_{k-1}) - y_{\text{ref}})$
13: $\bar{x}_k^{*} \leftarrow \bar{x}_{k_{\text{solve}}}^{*}, \bar{u}_k^{*} \leftarrow \bar{u}_{k_{\text{solve}}}^{*}$
14: $i = \arg \min_{j \in \{1, \ldots, N_{\text{lin}}\}} \|\bar{x}_k - \bar{x}_j\|_M$
15: $u_k \leftarrow \bar{u}_k^{*} + G_i(\bar{x}_k - \bar{x}_j)$
16: applyControl($u_k$)
17: $k \leftarrow k + 1$

6.4 Future Work

There are many interesting future directions for this work, both on the design of the control scheme and on the consideration of different types of nonlinear reduced order modeling approaches. Regarding the control scheme, the current approach lacks rigorous guarantees on performance of the closed-loop system. While the use of a reduced order EKF for state estimation and gain-scheduled feedback control are computationally efficient and generally perform well, the approach cannot guarantee stability of the system. In fact it is possible that in highly non-linear systems one or both of these components could perform poorly. Therefore additional work could be done to leverage more state-of-the-art nonlinear control techniques within this framework to improve reliability. Robustness against external disturbances could also be considered, perhaps using a tube-based approach.
as was done for the linear ROMPC approach from Chapter 6.4. Finally, modifications could be made to the reduced order OCP (6.7) to provide guarantees on recursive feasibility, which is necessary to guarantee closed-loop stability of the MPC scheme.

Additional improvements could be made to the piecewise-affine model order reduction method to improve modeling accuracy. First, more rigorous methods for determining the linearization points could be proposed. Second, it is well-known that using global reduced order basis matrices (i.e., global \((W,V)\)) is not ideal and can lead to larger than necessary ROM dimensions. This issue could be alleviated using by using local reduced order basis matrices at each linearization point, however additional care must be taken since the individual reduced order bases may not correspond to the same generalized coordinates \(\text{[Amsallem and Farhat 2011]}\). The work in \(\text{[Choi et al. 2020]}\) uses this approach within the context of PDE-constrained optimization by using a database of linear ROMs associated with different parameters and interpolating to find the linear ROM at new parameter points. Similar to the piecewise-affine ROM (TPWL) method, the maniMOR approach \(\text{[Gu and Roychowdhury 2008]}\) also leverages a database of linearized ROMs with varying bases to define a nonlinear ROM.

Another important extension would be to leverage state of the art hyperreduction methods for function approximation in place of the linearization (piecewise-affine) approach used in this chapter. While hyperreduction methods are software-intrusive in the online implementation, they do not suffer from approximation error induced by deviations from the linearization points since the approximations are done ad hoc at each state. In particular, the general idea behind hyperreduction methods is that the nonlinear function can be approximated through a sampling of the entries of the nonlinear function followed by a reconstruction step. For example, in gappy POD \(\text{[Everson and Sirovich 1995]}\) and DEIM \(\text{[Chaturantabut and Sorensen 2010]}\), the nonlinear function in the reduced order model:

\[
\dot{x} = (V^{\top}W)^{-1}W^{\top}f(Vx + x_{\text{ref}}, u),
\]

is approximated\(^2\) as:

\[
f(Vx + x_{\text{ref}}, u) \approx V_f(P^{\top}V_f)^{\dagger}P^{\top}f(Vx + x_{\text{ref}}, u)
\]

where \(V_f \in \mathbb{R}^{k \times n_f}\) is a basis matrix (e.g., generated through an application of POD to snapshots of the nonlinear function \(f\)) and \(P \in \mathbb{R}^{n_f \times s}\) is a “mask” matrix of ones and zeros such that \(P^{\top}f(Vx + x_{\text{ref}}, u)\) outputs specific rows of \(f(Vx + x_{\text{ref}}, u)\). Thus the ROM can be written as:

\[
\dot{x} = f(x, u) := MP^{\top}f(Vx + x_{\text{ref}}, u),
\]

where \(M = (V^{\top}W)^{-1}W^{\top}V_f(P^{\top}V_f)^{\dagger} \in \mathbb{R}^{n \times s}\) is a small matrix that can be computed \textit{a priori} and

\(^2\)In the DEIM method the pseudoinverse \(\dagger\) actually ends up being the inverse based on the resulting matrix rank.
the product $P^\top f^f(Vx+x^f_{\text{ref}}, u)$ is determined by evaluating \textit{only} $s$ elements of the nonlinear function $f^f$, thereby making the evaluation of the function and its Jacobians significantly more efficient. This approach has been leveraged within the context of design optimization in (Amsallem et al., 2015). The ECSW method (Farhat et al., 2015) takes a similar overall approach, but is tailored to nonlinear models that arise from finite element methods and also focuses on guaranteeing additional theoretical properties, such as preservation of structure and numerical stability.
Chapter 7

Soft Robot Control Using FEM-Based MPC

Soft robots are an emerging class of robots that leverage natural compliance through continuous deformation to simplify tasks such as object manipulation, moving in complex environments, safely interacting with humans, and even assisting in surgical procedures (Rus and Tolley, 2015; Thuruthel et al., 2018). However, significant challenges in modeling, simulation, and control have limited their practical use. One fundamental challenge is that continuously deforming soft robots are infinite-dimensional systems that can exhibit significant nonlinear behavior during structural deformation. Another challenge is that diversity among soft robot designs makes it challenging to develop modeling techniques that are generalizable.

One approach to soft robot modeling and control is to hand-engineer simplified models of the robot’s motion by making approximations. For example, piecewise constant curvature models (Westerm, III and Jones, 2010), beam models (Gravagne et al., 2003), and Cosserat models (Renda et al., 2018) have been used. However, these low-fidelity approximations are typically tailored to specific robot geometries and often model only the robot’s kinematics, making it challenging to design controllers for dynamic tasks.

Data-driven methods have also been developed to generate models directly from input-output data. This approach has been used to develop both kinematic and dynamic controllers. For example, (Bern et al., 2020) learns a differentiable kinematics model for solving inverse kinematics problems, and (Gillespie et al., 2018; Thuruthel et al., 2019) learn neural network dynamics models to develop closed-loop controllers. Another data-driven approach is proposed in (Bruder et al., 2019), which uses Koopman operator theory to build a dynamics model that is used for model predictive control. While data-driven methods are generalizable to different types of soft robots, they fail to leverage physics-based principles and there is no systematic procedure for developing them. Additionally,
reliance on experimental data for model identification and controller validation precludes the use of data-driven modeling approaches in the design process.

Alternatively, finite element methods (FEMs) provide a systematic, physics-based approach to soft robot modeling. These approaches can be used to develop high-fidelity models for a wide variety of soft robots and can be directly incorporated into the design process. While some techniques directly use FEM models for inverse kinematics based control (Duriez, 2013) or trajectory optimization (Bern et al., 2019), the high-dimensionality of FEM models (e.g. thousands to tens of thousands of degrees of freedom) makes the design of real-time dynamic controllers challenging.

### 7.1 Introduction

In this chapter, we employ the nonlinear reduced order model predictive control scheme presented in Chapter 6 to design a controller that leverages high-fidelity finite element models to control cable-actuated soft robots (Tonkens et al., 2021). In particular, the challenge of computational efficiency is addressed by using proper orthogonal decomposition to compress the high-dimensional FEM model without significant loss in modeling accuracy.

Previous work has also leveraged reduced order finite element models within the context of soft robotics. Specifically, linearized reduced order FEM models have been used to design regulating output-feedback controllers for cable-actuated (Thieffry et al., 2019a) and pneumatically-actuated (Katzschmann et al., 2019) soft robots, as well as trajectory tracking controllers (Thieffry et al., 2019b). However, using a linearized FEM model is not sufficient for many dynamic control problems due to the significant nonlinearities that can arise from the soft robot’s deformation. Nonlinear FEM model reduction has only been used in the context of soft robotics to design inverse kinematics-based controllers (Goury and Duriez, 2018), which are not sufficient for dynamic control tasks such as trajectory tracking. In summary, due to the use of either linearized models or kinematics-based controllers, existing soft robot control approaches that leverage reduced order FEM models do not adequately address dynamic soft robot control tasks.

This chapter proposes an approach for constrained optimal control of soft robots that leverages high-fidelity nonlinear finite element models. We handle the computational challenges induced by the high-dimensionality of the FEM models by exploiting an existing linearization-based model order reduction technique in the design of our control scheme, as described in detail in Chapter 6. We demonstrate the performance of the nonlinear reduced order model predictive control scheme in both simulation and through hardware experiments, using the “Diamond” soft robot whose FEM model is shown in Figure 7.1. The FEM model used to design the controller for this robot has a dimension of 9768. We perform all simulations in SOFA (Coevoet et al., 2017), an open-source FEM software toolkit. Our implementation of the control scheme, as well as an interface to the SOFA simulator,
is available online\footnote{github.com/StanfordASL/soft-robot-control}. For the hardware experiments, we use the soft silicone “Diamond” robot shown in Figure 7.1 which is actuated by four torque-control enabled servos, and use the same controller that is used in simulation. Data is collected (and used for feedback) through the use of five motion capture markers attached to the robot.

Figure 7.1: The soft silicone “Diamond” robot. The cables attached at the “elbows” are routed to four torque-control enabled servos that are used to apply specific cable tensions. (Left) The hardware setup for the robot (servos not pictured). The grey dots attached to the top and “elbows” are motion capture markers that are tracked in real-time. (Right) Finite element mesh with $N = 1628$ mesh nodes. The black lines denote the actuation cables.

### 7.2 Soft Robot Finite Element Model

Finite element methods are a numerical approach for solving partial differential equations (PDEs) and have been used for physics-based modeling and simulation in many domains, including fluid and structural mechanics as well as soft robotics (Coevoet et al., 2017). These methods solve PDEs by performing a spatial discretization with a finite number of elements defined by a mesh, which results in a finite set of ordinary differential equations corresponding to the state of each mesh node.

In the context of soft robotics, the state of each node consists of the node’s position and velocity, resulting in six degrees of freedom. Therefore, for a spatial discretization with $\mathcal{N}$ nodes the resulting FEM model will have a total dimension of $n^f = 6\mathcal{N}$. In particular, the soft robot FEM model is defined as in (Goury and Duriez, 2018) using Newton’s second law:

\begin{equation}
M^f \ddot{q}^f = P^f - F^f(q^f, \dot{q}^f) + H^f(q^f)u,
\end{equation}

\begin{equation}
\dot{q}^f = v^f,
\end{equation}

where $q^f(t) \in \mathbb{R}^{3\mathcal{N}}$ is the vector of node positions, $v^f(t) \in \mathbb{R}^{3\mathcal{N}}$ is the vector of node velocities, and $u(t) \in \mathbb{R}^m$ is the control input. Additionally, $M^f \in \mathbb{R}^{3\mathcal{N} \times 3\mathcal{N}}$ is a positive definite constant mass
matrix, \( P_f \in \mathbb{R}^{3N} \) represents constant external forces (e.g. gravity), \( F_f(q^f, v^f) : \mathbb{R}^{3N} \times \mathbb{R}^{3N} \rightarrow \mathbb{R}^{3N} \) is a nonlinear function that defines the internal forces (e.g. stresses from deformation), and \( H_f(q^f) : \mathbb{R}^{3N} \rightarrow \mathbb{R}^{3N \times m} \) is a nonlinear function that specifies the input matrix. The internal forces \( F_f(q^f, v^f) \) can be modeled in several ways depending on the material properties of the robot. A classic example is to use a linear-elastic model, which assumes a linear stress-strain relationship through Hooke’s law. Note that the FEM model can be written in the general form of (6.1) from Chapter 6 by defining a combined state vector \( x_f = [v^f \top, q^f \top] \top \in \mathbb{R}^{n_f} \) and by multiplying through the first equation by the inverse of the mass matrix.

As in Chapter 6 we assume the measurement and performance output models are linear:

\[
y = C_y x_f, \quad z = C_z x_f,
\]

where \( y(t) \in \mathbb{R}^p \) is the measurement and \( z(t) \in \mathbb{R}^o \) are the performance variables. In FEM-based control problems only a small set of variables may be of particular interest (e.g. end-effector position and velocity) and thus the number of performance outputs is generally small (\( o \ll n_f \)).

### 7.3 Piecewise-Affine Reduced Order Model

As discussed in Chapter 6, principled model order reduction techniques have been developed to derive high-fidelity reduced order models (ROMs) from high-dimensional models. For nonlinear systems like the soft robot FEM model, the model order reduction process consists of two successive steps. The first is to project the high-dimensional model onto a reduced-order subspace, and the second is to define an efficient approach for approximately evaluating the model’s high-dimensional nonlinear terms.

#### 7.3.1 Model Reduction Via Proper Orthogonal Decomposition

We first use proper orthogonal decomposition to reduce the nonlinear FEM model (7.1). The POD method defines a Galerkin projection specified by a projection matrix \( UU^\top \), where \( U \in \mathbb{R}^{3N \times r} \) is an orthogonal basis matrix and \( r \) is the dimension of the reduced-order subspace (typically \( r \ll 3N \)). This projection is applied to the high-dimensional FEM position and velocity vectors to define reduced order position and velocity vectors:

\[
q = U^\top (q^f - q^f_{\text{ref}}), \quad v = U^\top (v^f - v^f_{\text{ref}}),
\]

and to approximately reconstruct the original states by:

\[
q^f \approx Uq + q^f_{\text{ref}}, \quad v^f \approx Uv + v^f_{\text{ref}}.
\]
where $q_{ref}^f, v_{ref}^f \in \mathbb{R}^{3N}$ are the constant reference states that are used to better condition the basis matrix $U$. In particular, we select $q_{ref}^f$ and $v_{ref}^f$ to correspond to a static equilibrium (i.e. $v_{ref}^f = 0$) of the soft robot.

The reduced order model is then defined by projecting the FEM model (7.1) onto the reduced order subspace:

$$
\begin{align*}
M\dot{v} &= P - U^T F^f (Uq + q_{ref}^f, Uv + v_{ref}^f) + U^T H^f (Uq + q_{ref}^f)u, \\
\dot{q} &= v + v_{ref},
\end{align*}
$$

(7.3)

with $M = U^T M_f U$, $P = U^T P_f$, and $v_{ref} = U^T v_{ref}^f$. Additionally, with the combined reduced order state $x = [v^T, q^T]^T \in \mathbb{R}^n$ (where $n = 2r$) and reference state $x_{ref}^f = [v_{ref}^f, q_{ref}^f]^T \in \mathbb{R}^n$, the reduced order measurement and performance models are given by:

$$
\begin{align*}
y &= C_y x + y_{ref}, \\
z &= C_z x + z_{ref},
\end{align*}
$$

(7.4)

where $C_y = C_y^f V$ and $C_z = C_z^f V$ with $V$ defined as $V = \text{blkdiag}(U, U)$, and $y_{ref} = C_y x_{ref}^f$ and $z_{ref} = C_z x_{ref}^f$ are constants. Crucially, this projection results in a ROM (7.3) with combined dimension $n$, which can be orders of magnitude smaller than the original FEM model (7.1) of dimension $n_f$.

### 7.3.2 Nonlinear Model Reduction

While the ROM (7.3) has a reduced dimension, evaluation of the nonlinear terms is still computationally expensive because of their dependence on the high-dimensional state (e.g. the evaluation of the internal force $U^T F^f (Uq + q_{ref}^f, Uv + v_{ref}^f)$ scales as $O(n_f)$). As discussed in Chapter 6, we address this by approximating the nonlinear terms by reduced order piecewise-affine functions, resulting in a piecewise-affine ROM.

Specifically, the piecewise-affine approximations are generated by considering $N_{\text{lin}}$ linearization points $(\tilde{q}_i, \tilde{v}_i)$, which are represented in the high-dimensional space by $\tilde{q}_i^f = U\tilde{q}_i + q_{ref}^f$ and $\tilde{v}_i^f = U\tilde{v}_i + v_{ref}^f$. Note that the FEM model (7.1) is control-affine, and therefore no linearization about the control $u$ is needed. Around each of these linearization points, the internal forces $F^f$ are approximated by a first-order Taylor expansion:

$$
F^f(q^f, v^f) \approx F^f(\tilde{q}_i, \tilde{v}_i) + \frac{\partial F^f}{\partial q^f}(\tilde{q}_i, \tilde{v}_i)(q^f - \tilde{q}_i) + \frac{\partial F^f}{\partial v^f}(\tilde{q}_i, \tilde{v}_i)(v^f - \tilde{v}_i).
$$

Thus the internal force terms from (7.3) can be written as:

$$
U^T F^f (Uq + q_{ref}^f, Uv + v_{ref}^f) \approx F_i + K_i (q - \tilde{q}_i) + D_i (v - \tilde{v}_i),
$$
CHAPTER 7. SOFT ROBOT CONTROL USING FEM-BASED MPC

with reduced order terms:

\[ F_i = U^T F^f(q^f_i, \dot{q}^f_i), \quad K_i = U^T \frac{\partial F^f}{\partial q^f}(\tilde{q}^f_i, \tilde{v}^f_i)U, \quad D_i = U^T \frac{\partial F^f}{\partial v^f}(\tilde{q}^f_i, \tilde{v}^f_i)U. \]

The input matrix \( H^f \) in (7.3) is then simply approximated by the zeroth-order Taylor series expansion:

\[ H^f(Uq + q^f_{\text{ref}}) \approx H^f(q^f_i). \]

With these simplifications the ROM (7.3) can be approximated in the vicinity of \( \tilde{x}_i = [\tilde{v}^T_i, \tilde{q}^T_i]^T \) by:

\[ \dot{x} = A_i x + B_i u + d_i, \quad (7.5) \]

where:

\[
A_i = \begin{bmatrix}
-M^{-1}D_i & -M^{-1}K_i \\
I & 0
\end{bmatrix}, \quad B_i = \begin{bmatrix} M^{-1}H_i \\ 0 \end{bmatrix}, \\
d_i = \begin{bmatrix} M^{-1}(P - F_i + K_i\tilde{q}_i + D_i\tilde{v}_i) \\ \dot{v}_{\text{ref}} \end{bmatrix}.
\]

This collection of \( N_{\text{lin}} \) linearization points yields the piecewise-affine ROM in the form of (6.5) in Chapter 6:

\[ \dot{x} = A_i x + B_i u + d_i, \quad i = \arg \min_{j \in \{1, \ldots, N_{\text{lin}}\}} \| x - \tilde{x}_j \|_M. \quad (7.6) \]

We choose the weighting matrix \( M \) to be \( M = \text{blkdiag}(0, I) \) where \( I \) is the appropriately sized identity matrix since the internal forces of the soft robot in Figure 7.1 are modeled as heavily dependent on the robot’s configuration \( q^f \). Additionally, we define a discrete-time version of this piecewise-affine ROM in the form of (6.6) by assuming a zero-order hold control input, which is used in the control scheme.

7.3.3 Algorithm for Constructing the Piecewise-Affine ROM

This section proposes an automated approach for the development of the piecewise-affine ROM (7.6), specifically the computation of the reduced order basis matrix \( U \) and the selection of the set of linearization points \( P = \{(\tilde{q}_i, \tilde{v}_i)\}_{i=1}^{N_{\text{lin}}} \). This process is computationally expensive since it relies on simulations of the high-dimensional FEM model (7.1), but can be done entirely offline.

As mentioned in Section 7.3.1 the basis matrix \( U \) that defines the reduced order subspace is computed via POD. In this work, POD is implemented by simulating the FEM model (7.1) to collect a set of “snapshots”, which can include the soft robot’s configuration \( q^f \), velocity \( v^f \), and acceleration \( \dot{v}^f \), and implicitly defines a basis that characterizes the robot’s behavior. A principal component
7.3. PIECEWISE-AFFINE REDUCED ORDER MODEL

The analysis of the snapshots is then used to identify the reduced order subspace. In particular, the subspace is selected by defining a “snapshot matrix” $S$ with columns corresponding to snapshots, and then computing a singular value decomposition $S = \bar{U}\bar{\Sigma}\bar{V}$. The basis matrix $U$ is then defined by taking the $r$ columns of $\bar{U}$ associated with the $r$ largest singular values. The dimension of the subspace, $r$, is typically chosen to be as small as possible while still providing good approximation accuracy. In practice, a commonly used heuristic for quantifying the approximation accuracy is the “energy” of the truncated singular values (Antoulas, 2005).

The linearization points used to define the piecewise-affine ROM (7.6) are also determined via an offline data-driven procedure. In particular, at each time step of a FEM simulation a linearization point is added to the ROM if the reduced order state predicted by the ROM diverges too significantly from the FEM result. In other words, the ROM is built incrementally over the course of the simulation. For good ROM accuracy, the FEM simulation that is used for POD snapshot collection and linearization point selection should sufficiently cover the full range of possible robot motions. To ensure sufficient data is collected, a simple yet effective approach is to apply an open-loop control sequence in the FEM simulation that approximately spans the range of possible actuation for the robot. Specifically, we choose this sequence through Latin hypercube sampling of the soft robot’s admissible actuation. A summary of the procedure for developing the piecewise-affine ROM is given in Algorithm 8, where FEM simulates the FEM model (7.1) over one time step, POD computes the reduced order basis $U$, and ROM corresponds to simulating the piecewise-affine ROM (7.6).

```
Algorithm 8 Defining Piecewise-Affine ROM (Offline)

1: procedure DEFINE_ROM($T_{sim}$, $r$, $M_q$, $M_v$, $\eta$)
2: \{$u_k\}_{k=0}^{T_{sim}} \leftarrow \text{LatinHypercubeSample}(T_{sim})$
3: $X^f = \{(q_0^f, v_0^f, 0)\}$
4: for $k = \{0, \ldots, T_{sim}\}$ do
5: \{$q_{k+1}^f, v_{k+1}^f\} \leftarrow \text{FEM}(u_k, q_k^f, v_k^f)$
6: $X^f \leftarrow X^f \cup \{(q_{k+1}^f, v_{k+1}^f, v_{k+1}^f - v_k^f)\}$
7: $U \leftarrow \text{POD}(X^f, r)$
8: $P = \{(q_0^f, v_0^f)\}$
9: for $k = \{0, \ldots, T_{sim}\}$ do
10: \{$q_{k+1}^f, v_{k+1}^f\} \leftarrow \text{FEM}(u_k, q_k^f, v_k^f)$
11: \{$q_{k+1}, v_{k+1}\} \leftarrow (U^\top(q_{k+1}^f - q_{k+1}^f), U^\top(v_{k+1}^f - v_{k+1}^f))$
12: \{$q_{k+1}, v_{k+1}\} \leftarrow \text{ROM}(u_k, q_k, v_k, P)$
13: if $\|q_{k+1} - \bar{q}_{k+1}\|_{M_q} + \|v_{k+1} - \bar{v}_{k+1}\|_{M_v} > \eta$ then
14: $P \leftarrow P \cup \{(q_k^f, v_k^f)\}$

return $U, P$
```
CHAPTER 7. SOFT ROBOT CONTROL USING FEM-BASED MPC

7.4 Simulation Experiments

We now compare our method against two alternative approaches. First, we use the linear ROMPC scheme presented in Chapter 2 synthesized with a *linearized* FEM model to demonstrate the significant benefits of using a nonlinear model in our proposed approach. Second, we demonstrate that our method performs comparably to a data-driven Koopman operator-based MPC scheme [Bruder et al., 2019] without suffering common drawbacks of data-driven methods, such as a loss in generality from building problem or task-specific models. These comparisons are discussed in more detail in Section 7.4.2.

We compare these approaches in simulation using the “Diamond” soft robot shown in Figure 7.1. This robot is fixed at the base and is actuated by controlling the tension in four cables that are attached at the robot’s “elbows”. The “top” of the robot will be referred to as the “end effector”. We use the open-source SOFA framework [Faure et al., 2012; Coevoet et al., 2017] for finite element model-based simulations, and the mesh used in our experiments can be found in the SOFA Soft Robots plugin[^2] and in our online repository[^3]. The FEM model used to simulate the Diamond robot consists of *N* = 1628 nodes (i.e. *n* = 9768), the nonlinear internal forces *F* (*q*, *v*) are defined by a linear stress-strain law, and gravity is assumed to be the only external force *P*.*F*. The Young’s modulus is chosen to be *E* = 175 MPa, the Poisson’s ratio *ν* = 0.45, and a Rayleigh damping model:

\[
\frac{∂F_f}{∂v_f}(q_f, v_f) = \alpha M_f + \beta \frac{∂F_f}{∂q_f}(q_f, v_f),
\]

is used with \(\alpha = 2.5\) and \(\beta = 0.01\). The total mass of the robot was set to be *m* = 0.424 kilograms. These physical quantities were chosen experimentally to provide good agreement with the physical robot discussed next in Section 7.5.

The measurement model includes the position and velocity of the end effector and the four “elbows” of the robot (i.e. *y* ∈ \(\mathbb{R}^30\)). We consider a control application where the performance variable is the position of the robot’s end effector (i.e. *z* = \([x_{ee}, y_{ee}, z_{ee}]^T\)). In particular, a constrained optimal control problem is formulated to drive the end effector position \((x_{ee}, y_{ee}, z_{ee})\) to track a desired trajectory \((x_{d,ee}(t), y_{d,ee}(t), z_{d,ee}(t))\) subject to position constraints. Two desired trajectories are considered:

1. **Example 1:** An “infinity” or “figure 8” shape trajectory in the *x*<sub>ee</sub> and *y*<sub>ee</sub> end-effector coordinates. A constraint on the *y*<sub>ee</sub> coordinate is also included.

2. **Example 2:** A “circle” trajectory in the *y*<sub>ee</sub> and *z*<sub>ee</sub> end-effector coordinates.

Note that these types of *dynamic* trajectory tracking problems may not be addressed well with

[^2]: github.com/SofaDefrost/SoftRobots
[^3]: github.com/StanfordASL/soft-robot-control
kinematics-based controllers unless the motion is slow enough that dynamic effects are not significant, which highlights one advantage of the proposed nonlinear FEM-based ROMPC approach over existing techniques. In addition, the use of the constraint in Example 1 precludes the use of many data-driven control methods, highlighting the generality of our approach.

7.4.1 Controller Parameters

In addition to our proposed nonlinear FEM-based ROMPC scheme, two additional baseline control schemes are implemented to evaluate relative performance. Here we provide the details related to the definition of each controller:

1. **Nonlinear FEM-based ROMPC**: For our proposed approach we construct a piecewise-affine ROM for the Diamond robot using the procedure in Algorithm 8. In particular, we use acceleration snapshots, $\dot{v}_f$, in the POD step to define a reduced order subspace with dimension $r = 21$ (i.e. $n = 42$), we choose $v_{f\text{ref}} = 0$, and select $q_{f\text{ref}}$ to be the equilibrium configuration of the robot with no actuation. For selecting the linearization points we choose the threshold parameter $\eta$ (with $M_q = 0$, $M_v = I$) such that a total of $N_{\text{lin}} = 274$ points $(\tilde{q}_i, \tilde{v}_i)$ are selected. Additionally, the piecewise-affine ROM is defined with $M = \text{blkdiag}(0, I)$, such that only the robot’s configuration (and not velocity) is used to select the nearest linearization point. The reduced order OCP (6.7) is then defined with a horizon of $N = 5$, a rollout horizon of $N_r = 3$, and the ROM is discretized in time by assuming a zero-order hold control with a sampling time of $h = 0.1$ seconds. To solve the OCP we use a slightly modified version of the SCP algorithm proposed by Bonalli et al. (2019). By interpolating the optimal trajectory computed by the OCP, the feedback controller (6.9) and state estimator (6.8) operate with a sampling time of 0.01 seconds.

2. **Linearized FEM-based ROMPC**: This controller is an implementation of the linear ROMPC controller defined in Chapter 2 based on a ROM generated by linearizing the FEM model (7.1) around the point $x^{f\text{ref}}$ and then reusing the POD projection from our proposed controller. We consider the same planning horizon $N$ and sampling time $h$ as in our controller and use a rollout horizon $N_r = 1$. The ROMPC feedback controller and state estimator also operate with a sampling time of 0.01 seconds by interpolating the optimized trajectory.

3. **Koopman operator-based MPC (Bruder et al., 2019)**: In this control approach a dynamic model of the input-output behavior of the system is defined through a data-driven method. To summarize, a set of nonlinear basis functions are used to transform the collected input-output data from the nonlinear system into a “lifted” state space. A linear model is then identified from the lifted data, which is used to formulate a linear MPC scheme. In this experiment the outputs for the model are the performance variables $z$, and (using the notation from Bruder et al. (2019)) we build the Koopman model with a delay $d = 1$ and a sampling
time of $h = 0.1$ seconds. We choose the basis functions to include all monomials of maximum degree 2, which leads to a linear model with a lifted state of dimension $N_K = 66$. The data used to build this model came from the same FEM model simulation used to collect data for the POD portion of our proposed approach.

### 7.4.2 Results

Simulation results of the Diamond FEM model are presented for each controller in Figures 7.2 and 7.3. Figure 7.2 shows an example where the target trajectory is the “figure 8” pattern and a constraint is imposed on the end-effector’s $y_{ee}$ coordinate, and the desired trajectory in Figure 7.3 is the “circle” trajectory in the $y - z$ coordinate plane. As can be seen, the ROMPC scheme, which uses a linearized ROM, offers poor tracking and severely violates the constraints. In contrast, our approach and the Koopman controller offer good tracking performance and generally satisfy the constraints, highlighting the importance of capturing the soft robot’s nonlinear behavior that is induced by the structural deformation combined with the geometry of the robot.

<table>
<thead>
<tr>
<th>Method</th>
<th>Tracking Error MSE (mm$^2$)</th>
<th>Computation Time</th>
<th>Mean (ms)</th>
<th>Min. (ms)</th>
<th>Max. (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear ROMPC</td>
<td>23.62</td>
<td></td>
<td>10</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>Koopman MPC</td>
<td>1.48</td>
<td></td>
<td>66</td>
<td>9</td>
<td>241</td>
</tr>
<tr>
<td><strong>Nonlinear ROMPC</strong></td>
<td><strong>0.83</strong></td>
<td></td>
<td><strong>33</strong></td>
<td><strong>12</strong></td>
<td><strong>68</strong></td>
</tr>
</tbody>
</table>

Table 7.1: Example 1 (Figure 8 Trajectory): A comparison of the mean square error (MSE) and the time spent solving QPs (on a 2.5 GHz Intel Core i5 processor with 8GB of RAM) in each controller. Both ROMPC controllers used the Gurobi [Gurobi Optimization, LLC, 2020] solver while the Koopman MPC controller used the OSQP [Stellato et al., 2017] due to numerical issues when using Gurobi (possibly a result of the numerical conditioning of the Koopman model). Note that the reported value for the nonlinear ROMPC method considers the cumulative sum of all QP solve times in the SCP algorithm. These results show our FEM-based optimal control scheme achieves state-of-the-art tracking performance in terms of the MSE and is real-time capable.

<table>
<thead>
<tr>
<th>Method</th>
<th>Tracking Error MSE (mm$^2$)</th>
<th>Computation Time</th>
<th>Mean (ms)</th>
<th>Min. (ms)</th>
<th>Max. (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear ROMPC</td>
<td>24.54</td>
<td></td>
<td>9</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Koopman MPC</td>
<td>3.22</td>
<td></td>
<td>7</td>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td><strong>Nonlinear ROMPC</strong></td>
<td><strong>0.94</strong></td>
<td></td>
<td><strong>43</strong></td>
<td><strong>10</strong></td>
<td><strong>86</strong></td>
</tr>
</tbody>
</table>

Table 7.2: Example 2 (Circular Trajectory): A comparison of the mean square error (MSE) and the time spent solving QPs (on a 2.5 GHz Intel Core i5 processor with 8GB of RAM) in each controller. Both ROMPC controllers used the Gurobi [Gurobi Optimization, LLC, 2020] solver while the Koopman MPC controller used the OSQP [Stellato et al., 2017] due to numerical issues when using Gurobi (possibly a result of the numerical conditioning of Koopman model). Note that the reported value for the nonlinear ROMPC method considers the cumulative sum of all QP solve times in the SCP algorithm. These results show our FEM-based optimal control scheme achieves state-of-the-art tracking performance in terms of the MSE and is real-time capable.

To demonstrate the computational requirements of each controller we report the amount of time spent solving QPs (which is the most significant computational component) in Tables 7.1 and
7.4. SIMULATION EXPERIMENTS

Figure 7.2: Example 1 (Figure 8 Trajectory): FEM model simulation results for the Diamond soft robot with the linear ROMPC controller (Chapter 2), the Koopman MPC controller (Bruder et al., 2019), and our proposed nonlinear FEM-based ROMPC controller. The red line represents a constraint, and the black dashed lines indicate the desired trajectory (which is partially infeasible due to the constraint). Note that in addition to the advantages discussed in Section 7.4.2, our proposed approach performs comparably to the Koopman approach with respect to tracking and constraint satisfaction. The poor performance of linear ROMPC motivates the need for nonlinear model-based control.

Note that the linear ROMPC and Koopman controllers only solve one QP at a time while the proposed SCP approach may require multiple QPs to be solved. These results show that the proposed FEM model-based control scheme is real-time capable.

In terms of tracking performance, the proposed nonlinear ROMPC scheme slightly improves on the performance of the Koopman operator-based controller in both simulations (see MSE results in Tables 7.1 and 7.2). Our proposed approach also offers several additional advantages over the Koopman MPC approach. First, the Koopman approach only models the behavior of a prespecified choice of the performance variables \( z \), while the FEM model in our approach captures the robot’s entire state (independent of the choice of \( z \)) and can therefore be used for any number of different control problems. In contrast to our approach, the Koopman controller is also restricted to only...
consider outputs $z$ that are a subset of the measured variables $y$. Second, the dimension of the Koopman model does not scale well with the number of measured variables. For example, including all the measured variables $y$ (i.e. setting $z = y \in \mathbb{R}^{30}$) would result in a model of dimension $N_K = 2145$ (using the same delay $d = 1$ and all monomials with maximum degree 2). Third, the Koopman controller must operate at whatever frequency the model is discretized at (and the frequency the QP is solved at), while our controller can be operated at much higher frequencies by subsampling the optimized trajectory.
7.5 Hardware Experiments

In this section we demonstrate our proposed FEM-based MPC control scheme through hardware experiments using the “Diamond” soft robot shown in Figure 7.1. As in the previous section we compare against the linear ROMPC scheme from Chapter 2 that uses a linearized FEM model, as well as the data-driven, Koopman operator-based MPC approach developed by Bruder et al. (2019).

7.5.1 Experimental Setup

The experimental setup consists of several main components:

1. Torque-control Servos: Four Dynamixel XM430-W350-T servos are used to control the tension in the robot’s four actuation cables through a torque-feedback controller (technically current-feedback) implemented on the servo. A simple linear system identification step was used to identify a relationship between the normalized command and the resulting torque/cable tension. Commands are sent to the servos using the DynamixelSDK ROS library.

2. Motion Capture System: An OptiTrack motion capture system is used to track the motion capture markers fixed to the robot in real-time. Each marker’s position and velocity are transformed into the robot’s local reference frame and then streamed as ROS topics.

3. Computer: The control scheme is implemented as a ROS package and run on a Dell XPS laptop with a 2.50 GHz Intel Core i5 processor and 8 GB of memory. The entire control scheme is implemented in Python, and the optimization problems were solved using CVXPY (Diamond and Boyd, 2016) to model the problem and Gurobi (Gurobi Optimization, LLC, 2020) as the solver.

System identification of the FEM model required tuning the linear stress-strain law parameters (i.e. Young’s modulus $E$ and Poisson’s ratio $\nu$), the Rayleigh damping parameters $\alpha$ and $\beta$, and the robot mass $m$. Calibration was also performed to ensure the normalized servo commands generated the expected torque values and to identify the appropriate coordinate transformation from the motion capture system reference frame to the local robot reference frame. Once the servos and motion capture system were calibrated, system identification of the FEM model consisted of several steps. First, the mass of the robot was measured. Second, the Young’s modulus $E$ and Poisson’s ratio $\nu$ were tuned while at the robot’s rest configuration such that the SOFA simulation matched the hardware (since at rest the damping terms are zero). Third, several step inputs were used to elicit a dynamic response, and the Rayleigh damping parameters $\alpha$ and $\beta$ were tuned. Finally, a linear scaling of the original servo calibration was defined to better calibrate the steady-state deformation of the robot with the FEM model. This last calibration was used to help account for additional effects (e.g. friction) affecting the actuation system once it was hooked up to the robot.
CHAPTER 7. SOFT ROBOT CONTROL USING FEM-BASED MPC

Figure 7.4: Hardware setup for soft robot experiments. The robot is equipped with motion capture markers that are tracked in real-time using an OptiTrack motion capture system. The robot is cable-actuated, where the tension in the cables is controlled through torque-enabled Dynamixel servos.

7.5.2 Controller Parameters

1. **Nonlinear FEM-based ROMPC**: For our proposed approach we use the same piecewise-affine ROM for the Diamond robot that was defined for the simulation experiments in Section 7.4 (which also used the FEM model whose parameters were tuned to match the hardware). We additionally keep all of the hyperparameters of the control scheme the same (e.g. the controller operates at 100 Hz, the ROM is discretized with a sampling time of $h = 0.1$ seconds, and an OCP is solved every 0.3 seconds), except that we additionally give the system 5 seconds of no control input to allow the state estimator to converge before the control task begins. The measurement and output model matrices $C^f_y$ and $C^f_z$ are also slightly different in these experiments since the motion capture markers are offset slightly from the robot. In particular we model the position of the motion capture markers as a function of the FEM model state as:

$$p_m = D \frac{p_n - p_0}{\|p_n - p_0\|} + p_n,$$

where $p_m \in \mathbb{R}^3$ is the position of the marker, $p_n \in \mathbb{R}^3$ is the position of the point on the robot where the marker is attached, $D \in \mathbb{R}$ is the offset distance from the robot’s surface to the marker center, and $p_0 \in \mathbb{R}^3$ a static position that acts as a “pivot point” to define a direction from $p_0$ to $p_n$ that we assume the marker to lie on. The position $p_n$ can be expressed as a linear function of the FEM state as $p_n = C^f_{pn} q^f$ and the nonlinear function for computing
7.5. HARDWARE EXPERIMENTS

$p_m$ is linearized with respect to $p_n$ to give a new linear measurement and output model. The model for the velocity of the motion capture markers is computed by taking the derivative of the position model:

$$v_m = \frac{D}{\|p_n - p_0\|^3} \left( \|p_n - p_0\|^2 I - (p_n - p_0)(p_n - p_0)^\top \right) v_n + v_n,$$

where $v_n \in \mathbb{R}^3$ is the velocity $v_n = \dot{p}_n$ and can be expressed as a linear function of the FEM state as $v_n = C_{vn}^f v^f$. This nonlinear model for marker velocity is linearized for use in our control scheme.

2. Linearized FEM-based ROMPC: We again use the same linearized FEM model as in the simulation experiments in Section 7.4, use the same control scheme hyperparameters (e.g. 100 Hz operating frequency), and also give the state estimator 5 seconds to converge before the control task begins. This controller also uses the new linear measurement and output models for the motion capture marker position and velocity that are used for the nonlinear ROMPC scheme.

3. Koopman operator-based MPC: For the Koopman operator-based MPC scheme (Bruder et al., 2019) we use the same hyperparameters to compute the Koopman model as in Section 7.4. However, in this case the data used to build the model was collected from the physical robot rather than the FEM simulator. In particular the data was collected from a five minute long series of ramp and step inputs that were randomly sampled using Latin hypercube sampling. The model is discretized with a sampling time of $h = 0.1$ seconds and therefore this controller operates at a frequency of 10 Hz.

7.5.3 Results

Experimental results for the “figure 8” target trajectory are shown in Figure 7.5 and for the “circular” target trajectory in Figure 7.6, and the desired trajectories are visually depicted relative to the robot in Figure 1. Additionally, the mean-squared tracking error is reported for each target trajectory in Tables 7.3 and 7.4. As noted from the simulation results, we see that the linear ROMPC controller provides the worst tracking performance, validating that the use of nonlinear models is beneficial. We also see that again our proposed nonlinear FEM-based ROMPC scheme performs similarly to the Koopman MPC, and in fact slightly better in terms of MSE tracking error. Based on the increase in performance as well as the other advantages of our proposed controller discussed in Section 7.4.2, we believe these results provide validation of the advantages of using FEM model-based controllers in terms of both performance and generality.

We would additionally like to note an additional observation from these experiments that we believe could lead to further improvements in performance of our control scheme. We believe that
one of the most significant sources of error in these experiments (related to our proposed control scheme) is the result of inaccurate actuation. Tension control is challenging, and we noted in testing and setup that the servo’s torque-controllers were not extremely precise. What seemed to be one of the biggest issues was an observed hysteresis in the actual servo torque when the commanded torque was linearly increased and then decreased. The impact of this hysteresis on the motion of the soft robot was observed in terms of a perceived “lag” in the motion when the commanded cable tensions were lowered. This impacted our proposed model-based controller since the FEM model and controller assumes the cable tension is perfectly applied. This is a general weakness for any model-based controller, namely that it is challenging to model all potential influences on the system. In contrast, the data-driven Koopman MPC approach should not suffer from this problem since the model is trained on input-output data that captures any nonlinearities arising from the actuation. Thus it should in theory be able to implicitly learn a model of the actuator dynamics. One interesting avenue of future research associated with this challenge is to explore how a high-fidelity FEM-based control scheme could be augmented with a data-driven component to account for unmodeled effects.

<table>
<thead>
<tr>
<th></th>
<th>Linear ROMPC</th>
<th>Koopman MPC</th>
<th>Nonlinear ROMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracking MSE (mm²)</td>
<td>50.60</td>
<td>19.41</td>
<td>10.36</td>
</tr>
</tbody>
</table>

Table 7.3: Example 1 (Figure 8 Trajectory): A comparison of the tracking mean square error (MSE) for each controller.

<table>
<thead>
<tr>
<th></th>
<th>Linear ROMPC</th>
<th>Koopman MPC</th>
<th>Nonlinear ROMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracking MSE (mm²)</td>
<td>84.71</td>
<td>49.76</td>
<td>42.25</td>
</tr>
</tbody>
</table>

Table 7.4: Example 2 (Circular Trajectory): A comparison of the tracking mean square error (MSE) for each controller.

7.6 Conclusion and Future Work

In this chapter we leverage the nonlinear reduced order model predictive control scheme defined in Chapter 6 for model-based optimal control of soft robots based on high-fidelity nonlinear finite element models. Notably, computational efficiency is achieved by defining a reduced order piecewise-affine model to approximate the high-dimensional FEM model. Both simulation and hardware results are used to demonstrate the performance of the proposed approach in comparison to a state-of-the-art data-driven method and the linear FEM model-based control method from Chapter 2. These results demonstrate an improvement over both existing methods in tracking performance for a pair of dynamic trajectory tracking tasks. We also discuss other advantages of the proposed model-based approach over data-driven methods, such as the generality of the FEM-model compared to task/problem specific data-driven methods.

Future work includes the use of more advanced nonlinear model reduction techniques (e.g. the
energy-conserving sampling and weighting (ECSW) method (Farhat et al., 2015), which is specifically formulated for FEM model reduction), application to different types of soft robots (e.g. pneumatically actuated robots, which have a distributed actuation force), the ability to handle scenarios where the robot makes and breaks contact with the environment, and whether data-driven techniques could be used to augment the current approach. Work by Balajewicz et al. (2016) could be a potential starting point for modeling contact within a reduced order model framework.
CHAPTER 7. SOFT ROBOT CONTROL USING FEM-BASED MPC

Figure 7.6: Example 2 (Circular Trajectory): FEM model simulation results for the Diamond soft robot with the linear ROMPC controller (Chapter 2), the Koopman MPC controller (Bruder et al., 2019), and our proposed nonlinear FEM-based ROMPC controller. The black dashed lines indicate the desired trajectory.

Figure 7.7: Desired trajectories for the Diamond soft robot hardware experiments. On the left is the “circular” desired trajectory in the $y - z$ plane (Example 2) and the “figure 8” trajectory in the $x - y$ plane (Example 1) is shown on the right. The constraint imposed on the “figure 8” trajectory is represented by the white colored “keep out region” in the right side image.
Chapter 8

Conclusion

Many interesting engineering systems, such as those with fluid flows or fluid-structure interaction (e.g. highly maneuverable or flexible aircraft) or flexible/deforming structures (e.g. soft robots), have dynamics that evolve over a continuum and thus are considered infinite-dimensional. High-fidelity semi-discretized partial differential equation models for these systems can be extremely high-dimensional, potentially ranging from tens of thousands to millions of ordinary differential equations. In this thesis, we explored techniques for model-based controller design for these systems that can efficiently exploit these high-fidelity, but high-dimensional, models without sacrificing closed-loop performance. We focused on the development of a model predictive control scheme, which can address constrained optimal control problems, to extend the state-of-the-art for infinite-dimensional system control and enable these systems to be operated in ways not possible with current techniques.

A key technique used in this thesis is model order reduction, which enables computationally efficient use of extremely high-dimensional models. While model order reduction methods have been well studied, this thesis exploited these approaches in a novel way. The majority of our work focused on the linear setting, where we developed a linear reduced order MPC (ROMPC) scheme that enjoys strong theoretical guarantees but, unlike existing approaches, does not require sacrifices in practicality, scalability, or generality. We also extend our ideas to the nonlinear setting by leveraging linearization-based methods for nonlinear model order reduction, and use sequential quadratic programming for real-time optimization in the MPC scheme. We demonstrate the performance of both linear and nonlinear ROMPC through extensive simulation and hardware experiments, focusing primarily on the applications of aircraft control through the use of linear computational fluid dynamics models for aerodynamics modeling, and soft robot control through the use of nonlinear finite element models of the robot’s continuously deforming structure.
8.1 Future Directions

The research presented in this thesis is motivated by the power of physics-based modeling to design controllers to enable safe and high-performing autonomous systems. Unfortunately, physics model-based control has been restricted primarily to systems where accurate models are low-dimensional, such as structurally rigid aerospace or ground vehicles where unsteady aerodynamic effects are not significant, robots with a finite number of rigid components, or other more complex systems where model fidelity is sacrificed to achieve low-dimensionality, leading to limitations on controller performance. This thesis shows that modern and high-performing model-based control techniques such as MPC can also be applied to systems with high-dimensional models in a principled way, as long as model reduction techniques can effectively compress the model dimension (as has been demonstrated in many useful settings, including those requiring semi-discretized PDE models).

To highlight the relevance and potential future impacts of this work in more specific detail, we again turn to the examples of aircraft and soft robot control. For aircraft control, the ability to model unsteady aerodynamic effects through high-fidelity computational fluid dynamics (CFD) models could have significant impacts on flight performance capabilities, allowing autonomous flight in operating regimes that are currently avoided for safety reasons. CFD-based controllers could also revolutionize future aircraft design, for example by allowing the synthesis of the controller to be directly embedded within multi-disciplinary design optimization problems, or enabling more complex aircraft designs (e.g. highly-flexible aircraft with significant aeroelastic effects, flapping-wing UAVs, morphing-body aircraft) that would be challenging to control without accurate computational aerodynamic or structural models. Research and development of soft robots is also becoming popular, as their physical interactions with the world (and with humans) are inherently safer due to the robot’s natural compliance, leading to significant advantages in contexts such as medical care, home-use, and even agriculture, where delicate handling of produce is required. Our work could therefore have significant impact for soft robotic applications since high-fidelity finite element models of structurally complex soft robots can be generated in systematic ways. In other words, the ability to use finite element models for controller design could lead to both higher-performing robots as well as novel and more complex robot designs. Lastly, there are also many applications that have not been explored in detail in this thesis, but which could similarly benefit from our research, such as fluid flow control (e.g. heating, ventilation, and air condition systems) or flexible structure control (e.g. spacecraft, buildings), heat flow control, chemical reaction control, lithium-ion battery charging, nuclear reactor control, and even finance.

However, there is still a significant amount of work to be done to fully tap the potential of reduced order model-based control. One of the most important areas of future work is the continued development of nonlinear reduced order MPC, since nonlinear models are crucial for many of the exciting applications of MPC schemes where the system is operating at the limits of performance. While many of the contributions of this thesis are related to the linear setting, they provide a
8.1. FUTURE DIRECTIONS

strong foundation for future work in the nonlinear setting as well. For example, our work developing theoretical guarantees on closed-loop performance in the linear ROMPC setting provides a basis for understanding challenges and potential solutions related to developing theoretical guarantees in the nonlinear setting as well. Additionally, our preliminary work in nonlinear ROMPC, where we leverage linearization-based methods for nonlinear model order reduction, also motivates future work that leverages more state-of-the-art hyper-reduction methods (which have the potential to provide significant improvements in ROM accuracy). It would also be beneficial to continue exploring the benefits of reduced order MPC for specific applications, including additional work on aircraft and soft robot control, and explore how reduced order model-based controllers can be augmented with data-driven techniques for improved performance. To summarize, areas of future consideration for reduced order model-based control of high-dimensional systems include:

1. Theoretical analysis of nonlinear ROMPC – to find conditions that guarantee desirable closed-loop properties such as stability and constraint satisfaction. Similar to the linear setting from Chapter 2, this will require analysis of model reduction error and again one of the main challenges will be related to computational efficiency.

2. Use of state-of-the-art hyper-reduction nonlinear ROMs – to take advantage of their improved accuracy over the linearization-based nonlinear ROMs used in this thesis. However, current hyper-reduced ROMs generally have more intensive software implementation requirements, which could make it challenging to satisfy real-time controller requirements on resource-constrained hardware.

3. Exploration of novel applications for ROM-based control – and additionally exploring how ROM-based controller synthesis methods could be embedded within multidisciplinary design optimization.

4. Explore how data-driven methods could be used to augment ROM-based controllers – to get the best of physics-based modeling and data-driven approaches. These approaches could focus on identifying specific weaknesses of ROM-based controllers that could be strengthened with machine learning, or they could leverage physical insight from the model reduction process to make data-driven control methods more efficient.
Bibliography


