Computational Strategies for Nonlinear Model Predictive Control

by

Matthew Jeremy Tenny

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

(Chemical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2002

Acknowledgments

During my time at the University of Wisconsin-Madison, I have been fortunate to work with several outstanding people. Foremost among them is my advisor, Prof. James B. Rawlings. His insight into which are the important issues of the field was an invaluable asset. He has the ability to distill a problem to its essence to determine the key aspects that must be addressed. His guidance and advice allowed me to keep my focus even during difficult research moments.

After a two near-miss encounters, I finally met up with Prof. Stephen J. Wright at Argonne National Laboratory in the summer of 1999, and continued collaborating with him after his arrival as a professor two years later. Working with Steve has been incredible – his knowledge of optimization is unmatched, and his assets as a human being are greater still. Steve is incredibly patient, and probably the hardest working person I have met. I owe many of my accomplishments to positive interactions with him.

My research benefitted greatly from the constructive feedback of Dr. Thomas A. Badgwell and Prof. David Q. Mayne. Their input greatly helped the exposition of some of the ideas in this study. Some of the examples in this dissertation were the result of collaborations with Thorsten Stuetzle, Prof. John W. Mitchell, Prof. William A. Beckman, Eric L. Haseltine, and Dr. Rahul Bindlish. Their help allowed me to focus my research more directly on theory and algorithmic design, rather than on developing examples.

I would next like to acknowledge my colleagues in the Rawlings research group. When I started in the group, Scott, Chris, and Rahul were like the big brothers I never had. Daniel was always ready for a game of darts, and near the end of his time in Madison he became a good outlet for my frequent sanity checks. Gabriele was a better dart player than Daniel, and the two had much to talk about regarding their love of U2. I will not hold that against them. Gabriele's cooking was amazing, and his friendship is both valued and missed. Jenny was always a cheerful presence in the office. Brian always knew the importance of a free lunch. Eric was the group workhorse, and he will never, ever win Nethack. Aswin was always available for lunch trips and coffee breaks. Finally, I thank John Eaton for answering my frequent stupid questions.

A big part of my life is my amazing friends. I would like to first mention the friends I have made in the department. I thank Rafael and Bridget Alcalá for being supportive and for the many delicious dinners I have had the opportunity to enjoy with them. I am grateful to Len Switzer for our frequent lunch trips. Richie and Anne Peters are good people, hands down. I also enjoyed the frequent music exchanges with Rob Sever.

My rock climbing friends have been a great diversion from work. I would like to thank the staff of Boulders climbing gym for their friendship. I would like to specifically mention James Lapotka, with whom I have enjoyed the tradition of thirsty Thursdays. I also want to thank Martha Montague who was always willing to climb, even at conferences. Finally, I am grateful to Katie Cadwell for our Scrabble and Boggle games and movies.

My main support system has been my friends from the Chicago area. I have known many of them for over a decade. Jason Hoos has always been a dependable source of phone numbers. Joe Sprenger likes his hot dogs with green stuff on it. Jason Weiss spent a lot of time consoling me on the telephone. Mike Kellen is a wild man. Smo is fat, and Val is fired.

Finally, I would like to thank my family for their support over the years. My parents tried their hardest to help me out, even when I didn't want help. My sister, Laurel, has always been one of my closest friends and confidants. My brother-in-law, Andy, was always useful to talk to when I needed a reality check.

I conclude by thanking all the people I have known over the years that made it possible for me to continue my educational pursuits.

Matthew Jeremy Tenny

University of Wisconsin–Madison

October 2002

Computational Strategies for Nonlinear Model Predictive Control

Matthew Jeremy Tenny

Under the supervision of Professor James B. Rawlings

At the University of Wisconsin–Madison

The primary purpose of this study is to develop a computationally efficient, numerically robust method for implementing model predictive control in real-time using nonlinear models. In addition to developing the algorithmic framework for the solution of these on-line optimization problems, the key differences between nonlinear and linear dynamic models are delineated in the context of optimization and closed-loop performance. A secondary goal of this research is to develop a freely available software package that incorporates the ideas of this work, as well as several relevant nonlinear models for benchmarking purposes.

The work presented in this thesis advocates a regulator without a terminal constraint, but with a terminal penalty based on the linear quadratic regulator at the target. The regulator is able to stabilize many difficult nonlinear examples.

Moving horizon estimation is used to estimate states from a history of input and output data. This strategy is shown to converge to the true state of the plant even when the current industrial standard, the extended Kalman filter, fails. Further,

an approximate smoothing covariance is developed for nonlinear models that removes the periodic behavior of the filtering covariance update.

Both the regulator and state estimator employ specially tailored sequential quadratic programming algorithms to efficiently solve their respective nonlinear programs. The feasibility-perturbed SQP (FP-SQP) algorithm is introduced, which has the property of maintaining feasibility with respect to the state evolution equation and hard constraints at every iteration. The solution of the quadratic programs at each iteration is performed by a structured solver that scales linearly with horizon length, rather than cubically. The presented nonlinear programming method is demonstrated to be faster and more efficient than standard commercial software.

The closed-loop behavior of the nonlinear model predictive control system is investigated using the regulator, the estimator, and a nonlinear target calculation. Insight into the stability of disturbance models is gained, and superior performance to linear control is documented. Further, local minima are demonstrated for both the regulator and estimator. These minima may be avoided by shortening the horizon in the regulator, or by applying constraints in the estimator.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

For a successful technology, reality must take precedence over public relations, for Nature cannot be fooled.

— Richard Feynman

Over the last two decades, model predictive control (MPC) has emerged as a major advanced control technology. Among its strengths are the ability to handle constraints directly in its framework and satisfaction of some optimal performance criteria by solving on-line minimization problems. At the heart of the MPC controller is the model, which is used not only to forecast the effects of future inputs, but also to estimate the current state of the plant given the history of past measurements and controls.

Because the optimization problems are solved on-line, the model is typically linear since the optimization problems that arise from linear models are easily solved quadratic programs. Process dynamics, on the other hand, are rarely linear. Nonlinearities enter chemical processes through higher-order reactions, temperature dependence of rate constants through the exponential relationship in the Arrhenius equation, and thermodynamics of non-ideal mixtures.

For certain chemical processes, linear models approximate the dynamics of the true system well and linear MPC is adequate. Systems in which little noise is present and operating conditions rarely change are often excellent candidates for this type of control. However, in processes in which disturbances are present, or in which the system must be steered through various operating regimes, such as during start up or grade transitions, linear MPC may perform poorly or even fail. For these reasons, nonlinear model predictive control (NMPC) is increasingly viewed as a desirable technology.

The obstacles associated with implementing NMPC are nontrivial. The naïve approach of attaching a generic nonlinear programming software package to a model obtained by numerical integration of the model's differential equations usually results in either failure or performance too slow to operate in real time. Further, the effects of model nonlinearities on the solution of the NMPC optimization problems and on the behavior of the closed-loop system are not yet well-studied. The purpose of this study is to develop an efficient solution method for nonlinear programs in NMPC and to examine the effects of nonlinearities on such a control system.

1.1 MPC Structure

For the purposes of our study, we consider an MPC system composed of three parts, as shown in Figure 1.1. At each time k, the regulator determines the value of the manipulated variable u_k to inject into the plant based on the current state estimate x_k and the steady-state state and input targets $x_{t,k}$ and $u_{t,k}$. The estimator uses the measured outputs y_k and the manipulated inputs to produce an estimate of the current state x_k and the integrated disturbance p_k. Based on the value of p_k and the

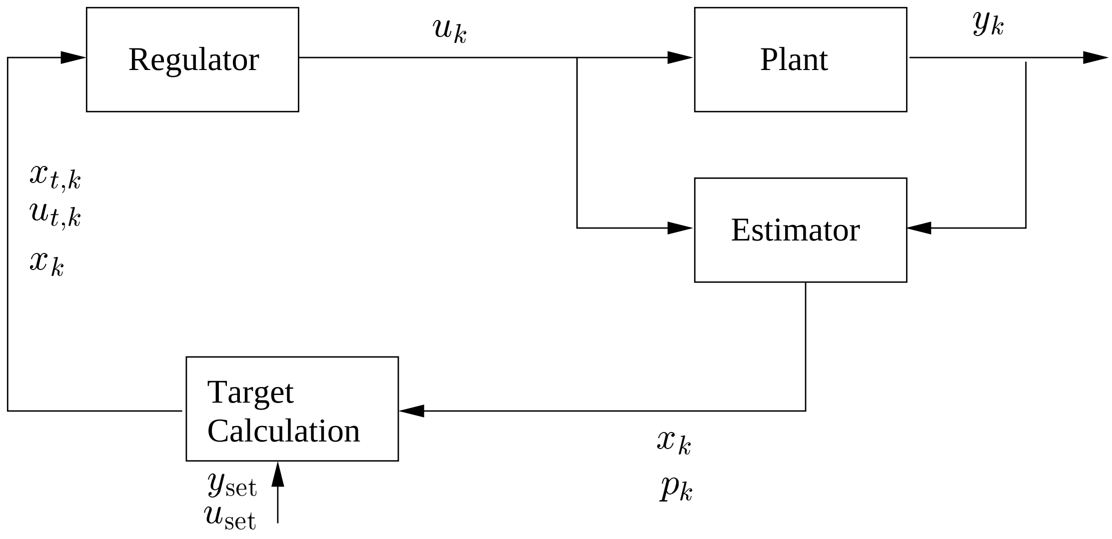

Figure 1.1: Schematic of MPC system

output and input set points y_{set} and u_{set}, the target calculation determines new state and input targets for the regulator. The process repeats at each time instant.

As might be expected, the amount of time between each measurement, called the sampling time, is a critical factor for this type of control system. If the sampling time is too short, then the required computations may not finish quickly enough to be useful. If the sampling time is too long, then the dynamics of the process could lead to undesirable controller performance. We assume that the sampling time can be chosen appropriately throughout this thesis.

In all three components of the MPC scheme shown in Figure 1.1, the algorithm must solve an on-line optimization problem in real-time. The regulator determines an optimal future input profile that brings the system from its current estimated state to the state and input targets. The general framework for this optimization problem is to minimize the deviation from the targets at *all times* in the future. While closed-form solutions to this type of optimization problem exist for certain types of systems, e.g., unconstrained linear systems, finding a solution for general constrained nonlinear models over an infinite prediction horizon is impractical; numerical methods to solve these problems can handle only finitely many variables and their computational times scale according to the number of variables considered. For this reason, we consider the moving horizon (also called receding horizon) regulator in which the optimization problem is performed over a prediction horizon, and the cost of the prediction after the end of the horizon is approximated by a terminal penalty.

The ideal estimator would minimize the variance of all the measured data from the model predictions. As the number of measurements increase during plant operation, the number of variables in the on-line minimization problem also increase. Again, for linear unconstrained systems, the Kalman filter is the closed-form solution for generating optimal state estimates. However, such a solution does not exist for our general class of models. Instead, we rely on on-line optimization. For this reason, we require the number of variables over which the cost function is minimized to be finite, and preferably small. Moving horizon state estimation considers a finite window of measurements at each sampling time and summarizes past measurements as an arrival cost.

Finally, the target calculation determines the state and input targets that are the closest steady states to the set points. This optimization problem is much smaller than the estimator and regulator problems since it considers only the steady-state problem and not trajectories of inputs, disturbances, or states through time. The target calculation must respond to changes in estimated disturbances as well as external changes in set points, which can result from plant-wide economic considerations.

1.2 Dissertation Overview

The remainder of this dissertation is organized as follows. Chapter 2 reviews the literature for nonlinear model predictive control and nonlinear moving horizon state estimation. We include references for applications, theory, and computational approaches as well as a historical overview of model predictive control.

The following two chapters focus on formulating the estimation and control problems studied for the remainder of this dissertation. Chapter 3 outlines the development of the MPC regulator formulation we use. We motivate the structure of the optimization problem to be solved in later chapters. Chapter 4 discusses the moving horizon estimation (MHE) problem. We begin by presenting the formulation of the problem. Since the optimization problem has a probabilistic interpretation, we motivate and derive a smoothing covariance update to summarize the arrival cost. We compare the performance of MHE with a smoothing covariance update to MHE with a filtering update and to the industrially popular extended Kalman filter (EKF).

The next portion of our study concerns the practical application of the MPC regulator and moving horizon estimation in real-time. Due to the dual nature of the regulator and the moving horizon estimator, the two techniques can be solved by similar methods. Chapter 5 derives solution methods for the structured quadratic programs that arise during iterates of a sequential quadratic programming algorithm. These QP subproblems are equivalent to the regulation and estimation problems for linear time-varying systems and are solved by a primal-dual interior-point method with the property that computation time scales linearly with horizon length rather than cubically.

Chapter 6 presents a new method for solving the nonlinear optimization problems in the MPC regulator and estimator. This approach, called feasibility perturbed sequential quadratic programming (FP-SQP), maintains feasibility of the iterates of the NLP with respect to the model equation and hard constraints on the inputs for the regulator. Among the highlights of this approach are that the algorithm may terminate early if a stabilizing suboptimal solution is desired, the method is robust to online failures, and it has proven convergence properties. Also, because it is an SQP method, it solves a quadratic subproblem at each iterate. These quadratic programs are specially structured and are solved efficiently by the method described in Chapter 5.

Chapter 7 explores the effects of nonlinear models on the closed-loop behavior of an NMPC controller operating on a plant. The results of this chapter incorporate the strategies and findings of each of the previous chapters. We identify characteristics of the performance of nonlinear models, some of which are improvements over the linear models, and others that represent minor obstacles to overcome. We suggest strategies to surmount the difficulty of finding local minima in the optimization problems of the regulator and estimator.

Chapter 8 summarizes the contributions of this study and recommends potential future work.

Chapter 2

Literature Review

For every expert, there is an equal and opposite expert.
— Arthur C. Clarke

2.1 Model Predictive Control

The industrial implementation of model predictive control was first proposed in 1976 and formally described in 1978 by Richalet et al. [100] in their commercial software IDCOM. They utilized a finite prediction horizon, an input response model, a reference trajectory, and constraints. Independently, Shell Oil developed a method in the early 1970s called dynamic matrix control (DMC) that Cutler and Ramaker [22] presented at the 1979 National AIChE meeting. The early version had no ability to handle constraints and no guarantee of stability. It did, however, require only the inversion of a matrix, and no explicit solution of an optimization problem. The ability to handle constraints appeared in 1986 when Garcia and Morshedi [30] applied DMC to a constrained linear system. The authors referred to the resulting methodology as quadratic dynamic matrix control (QDMC) because the formulation required the solution of a quadratic program. In 1984, Garcia presented an application of QDMC on a batch process [29]; many application papers followed. In 1989, Peterson et al. [88] reported the application of an NMPC scheme to a simulated semi-batch free radical polymerization of methylmethacrylate process. In this work, they chose to extend DMC to a nonlinear system with no constraints. However, they actually linearized the nonlinear system and applied DMC to the linear model. This result was an early example of a recurrent theme – the extension of linear MPC in the development of NMPC.

2.1.1 Model formulation

In order to solve the full NMPC problem, a nonlinear model must be utilized. The research surrounding model formulation for NMPC applications has been widely varied. Many models are chosen for their ease of use. Among these are the neural network, which has been used in NMPC by several researchers [50, 20, 113, 120]. However, the models achieved through neural networks are often inexact and require a large amount of data to properly capture the complete dynamics of a nonlinear system.

Another easily implementable model form is the fuzzy model, which rely on heuristic rules to change the type of model used depending on the region of operating space in which the state of the plant is located. This method has been utilized in NMPC applications [20, 128] and has been a popular method in industry where empirical models and heuristic methods are commonly employed. However, due to its heuristic nature, it has no stability guarantees or theoretical properties.

Closely related to fuzzy models is the concept of gain scheduling. In this approach, the nonlinear process is modeled as a system of interconnected linear models in state space. Then, linear MPC can be performed, switching back and forth between linear models as each region is encountered. This idea has been explored extensively by Ling and Edgar [62, 63] for their water-gas shift reactor. Again, heuristics are employed to determine where and how to organize the boundaries between each linear operating region. For highly nonlinear plants, one expects the linear regions to get asymptotically small, which poses a significant challenge for this method.

Other models are chosen for the facility of the optimization problem to be solved. Among these are polynomial models, for which the optimization step is reduced to simply locating the roots of a polynomial. Hernandez and Arkun [39] utilized an autoregressive moving average (ARMA) polynomial model for their CSTR with reversible first-order reaction. Jang and Wang [47] used collocation polynomials for their packed-bed distillation column, while Maner and Doyle [65] used a best-fit sixth-order polynomial to represent their polymerization reactor process. Another popular polynomial model is the Taylor series expansion, which has appeared in several NMPC applications [87, 15, 78]. However, in order to use a Taylor series model, it is assumed that the derivatives of the plant are either known or can be measured. Another popular polynomial model is the Volterra model [66, 31], which is commonly a second-order polynomial. The limits of the polynomial models are that model identification requires sizeable data sets, and model accuracy is sacrificed by limiting the order of the model. The more highly nonlinear the plant, the more important higher order accuracy becomes.

Some methods are linear MPC techniques, adapted to accept nonlinear plant models. Among these are gain scheduling (as previously mentioned) and feedback linearization. Feedback linearization relies on a geometric transformation of a nonlinear system into an equivalent linear system in transformed space. See [44] for a discussion of differential geometric techniques in estimation and control. Linear MPC is performed on the linear system and then the inputs are transformed back into the original space. There are a number of challenges associated with this method. First, only a specific subclass of problems are transformable into a linear system, making the method applicable to only a relatively small percentage of problems. Secondly, linear constraints in the nonlinear system may be transformed to nonlinear non-convex constraints in the equivalent linear system. Finally, many of the systems are only locally linearizable, resulting in regions of the state space that are not properly addressed. Linear MPC is not equipped to handle non-convex constraints. However, some researchers have applied this method to specific NMPC problems [109, 55].

The most popular models in literature are ordinary differential equations (ODE), partial differential equations (PDE) and differential algebraic equations (DAE). The benefits of these approach are that the model is based on theoretical foundations, and therefore exhibits the same type of behavior as the plant, only a few parameters need to be fit to the plant data (kinetic rate constants, etc.), and the controller can be designed for better robustness. The difficulty, however, is that the model must be numerically integrated to predict the behavior of the plant, which may be computationally expensive.

2.1.2 Computational approaches

Model reduction is one of the main approaches to reducing the computational burden of numerically integrating differential models. Among the methods of model reduction investigated for NMPC applications are wavelets [43] and orthogonal collocation [83, 82, 90]. Model reduction is only applicable for large state systems, like distillation columns.

To improve the stability of the integration step, Bock and co-workers [8] have been investigating direct multiple shooting methods, in which the solution to the ODE is determined at all points simultaneously. This method has been demonstrated to be more numerically stable than the traditional method of integration in series and has been demonstrated to be effective in efficient MPC formulations [79, 28, 25]. Significant research also has been performed on the optimization approach utilized by the regulator. Some investigators have focused on global optimization, specifically genetic algorithms [110, 81, 99]. However, this approach tends to be slow, and thus not implementable in real-time.

Others have chosen to increase the speed of popular local optimization methods by tailoring them to take advantage of the specific structure of the MPC formulation. The approach uses an interior point method to solve a sequential quadratic programming problem (SQP) that, due to the causal structure of the problem, has a banded structure [98, 1]. This method has been successful in easing the computational burden of optimizing large systems, and has recently come into favor with academic researchers. Also, Bartlett et al. [2] presented a comparison of interior-point and active-set optimization methods for NMPC.

A third approach has been to simplify the optimization problem in an effort to decrease the computation time of each calculation. This type of strategy has been employed in several ways. Zheng [129] chooses to optimize only on the first move of the prediction horizon at each sampling time. However, this approach does not yield favorable results, as it is closely related to a finite horizon optimization with a one-step ahead prediction. Additional approaches are to use distinct prediction and control horizons, as discussed by Magni et al. [64] and the so-called triple-mode MPC approach of Cannon and co-workers [11] in which the prediction horizon is split into three distinct parts to aid the computation.

Another method of simplification is to decentralize the control operations of a plant so that each sub-controller is responsible for a smaller subset of the plant. This approach makes sense for plant-wide control, where each unit operation could have its own controller. However, the interacting dynamics between control systems can lead to unexpected and unfavorable results. Ricker [102] investigated decentralized control of the Tennessee Eastman challenge process using NMPC techniques. In an earlier work, Ricker and Lee [101] used NMPC as a way to tune slave PI controller sub-loops using a cascade control approach.

A final approach is adaptive control. In this method, the parameters of the model are adjusted as the behavior of the closed-loop system is observed. Therefore, the model adapts to the plant. This strategy does not always work well, however, since unmeasured disturbances may greatly alter the model parameters when the model may not be in error. Adaptive NMPC has nonetheless found a place in the literature [39, 124, 90].

2.1.3 Theoretical advances

In an effort to guarantee stability properties for the nonlinear model predictive controller, the theoreticians had to approach the issue of implementing a finite horizon instead of an infinite horizon. The stability guarantee came first in the form of a terminal equality constraint [54, 68]. However, this solution was somewhat unnatural. If the prediction horizon was not sufficiently long, the method could fail. Also, this approach required a large horizon for the controller to yield comparable results to those achieved with an infinite horizon controller.

To ease the computation, a terminal region was proposed. The original idea was to have a dual-mode controller. Once the trajectory entered the terminal region, the controller would switch to linear MPC [70]. It was further stated that there was to be a quadratic terminal penalty added to the cost function to approximate the cost of the linear infinite portion of the trajectory after the end of the prediction horizon [85]. The method was further advanced by not requiring a strictly quadratic penalty and removing the requirement that a linear control law be implemented, instead using the linear feedback policy as an initial approximation to the optimal solution [23]. Methods outlining the theoretical properties of the terminal region are outlined by Chen and Allgöwer [16, 19], however, the efficient computation of such regions, online or offline, is not addressed in any work so far.

Less stringent criteria for stability were developed by Scokaert et al. [107] and Chen and Allgöwer [18], who require only a decrease in the cost function at every sampling time for stability, resulting in a suboptimal control law. Other work has included inverse optimality [67] where the feedback law is chosen first, and then the cost function that is optimal for the feedback law is developed. The theory, however, still relies on a terminal region that is identical to those described by other authors. Other important work has been done to eliminate the requirement of the terminal state constraint. Jadbabaie et al. [45] introduce a method that eases such a constraint, while also detailing several illuminating properties of the terminal region. Other recent work by Hu and Linnemann [42] removes the terminal constraint but requires a nonstandard terminal penalty.

2.1.4 Applications

The types of problems studied in the literature for the application of NMPC have been wide-ranging. Popular processes include continuous stirred tank reactors (CSTRs), polymerization reactions, and distillation columns. Some chemical engineering processes have included a paper machine headbox that was modeled and controlled by Lee and Ricker [58], and the Kamyr digester, which makes pulp from wood chips, studied by Wisnewski and Doyle [123]. However, there are also novel processes to which NMPC has been applied. Gibbs and Weber examined the control of a supercritical power plant [33, 34]. Chen and Weigand looked at the baker yeast fermentation process [20]. The process of drying cooked rice was investigated by Ramesh et al. [92]. Hong et al. examined microalgae fermentation [41] while Langrish et al. used NMPC for the drying of grey ironbark timber [57, 74]. The use of NMPC in the biomedical community appeared recently with the maintenance of glucose levels in the human body [120], the regulation of local anesthetic in human muscle tissue [60], and controlling the motion of a human arm [61]. NMPC has also been found in the control of transient pollution of urban wastewater systems [99] and in the aerospace industry, optimizing real-time guidance systems [7, 48, 91]. A recent article by Yu et al. [127] compares several nonlinear control methods including a variety of nonlinear MPC to the Caltech ducted fan. With such a wide range of applications, the importance of an efficient method for performing NMPC is essential.

2.2 Moving Horizon Estimation

Optimal state estimation for linear models became practical in the 1960's with the work of Kalman [51] and Kalman and Bucy [52] in their development of the well-known Kalman filter. This filtering method is the iterative closed-form optimal solution to the linear unconstrained MHE problem [10], and is implemented in a wide variety of applications. However, this approach is not applicable for nonlinear models nor constrained systems. Commonly, a simple approach for state estimation of nonlinear processes is to employ the extended Kalman filter (EKF). In this approach, a nonlinear system is linearized along a trajectory and Kalman filtering is performed on the linearized system [49]. The EKF is a popular in industrial settings and often performs well. Its theoretical properties remain largely unproven, though. Neither the Kalman filter nor the extended Kalman filter rely on on-line optimization, and neither handles constraints.

Moving horizon estimation was first presented for unconstrained linear systems by Thomas [119] and Kwon et al. [56]. The first use of MHE for nonlinear systems was not published until 1986 by Jang et al. [46]. In their work, however, the model did not account for disturbances or constraints. The stability of constrained linear MHE was developed by Muske and Rawlings [76, 77] and Rao, Rawlings, and Lee [96]. Robertson and Lee considered the probabilistic interpretation of constraints in the linear MHE problem [103, 105, 104]. The groundwork for constrained nonlinear MHE was developed by Rao et al. [94, 97].

Applications of MHE in industry have been reported by Russo and Young [106] and Gesthuisen and Engell [32]. Both papers considered state estimation of polymerization processes.

Chapter 3

MPC Regulator

To get anywhere, or even to live a long time, a man has to guess, and guess right, over and over again, without enough data for a logical answer.

— Robert Heinlein

While the field of linear MPC is considered by many to have reached maturity, there are still many issues in NMPC that have not been addressed. The most common criticism is that NMPC will never be able to operate in real-time. The aim of this document is to present a prototype method that represents a significant step toward the goal of stable real-time implementation of model predictive control for nonlinear processes.

One of the strengths of model predictive control is its ability to handle constraints in the control structure. A typical ordinary differential equation model is employed throughout the remainder of this document. The process model has the form:

$$\frac{dx}{dt} = f(x, u, t) \tag{3.1}$$

Here, the system is in state-space, where the variable x is a state and the variable u is an input or manipulated variable. It is common practice to transform x and u into deviation variables so that $f(0,0) = 0$. Therefore, the desired steady state occurs at the origin.

The true state of the plant is not often available, either because the states are internal variables and are impossible to measure, are expensive to obtain, or cannot be measured quickly enough to apply feedback control. Therefore, dynamic state estimation is necessary to recover an approximate state. Chapter 4 focuses on the solution to this problem. The outputs y are represented as follows:

$$y = g(x,t) \tag{3.2}$$

The MPC framework uses the state estimate as the basis for predicting the optimal control profile.

Often, the process also has constraints. Input constraints are based on physical limits of the process. For instance, valves are restricted to being open or closed no more than their physical limits, and inlet flowrates are not allowed to flow out of a system instead of into it. Because of the physical interpretation of these constraints, we refer to them as *hard constraints.* Other constraints are based on operating considerations, such as avoiding safety hazards that result if a process runs too hot, or environmental damage if a concentration becomes too high. These constraints on the states, if not strictly enforced, can be physically violated by the evolution of the system. Operating constraints that can be exceeded, but are present as guidelines for good operating practice are *soft constraints.* In later development, we will explore the mathematical interpretations of these two types of constraints. We present the constraints in the following way:

$$x \in X \tag{3.3}$$

$$u \in U \tag{3.4}$$

Model predictive control relies on computers to generate inputs based on measurements. While computers can perform calculations quickly, they do not function in continuous time. Instead, they operate on clock cycles. Therefore, it is necessary to represent the process model in a discrete time setting:

$$x_{k+1} = F(x_k, u_k) = x_k + \int_{t_k}^{t_{k+1}} f(x(\tau), u_k, \tau) d\tau \tag{3.5}$$

Here, u_k is held constant in the interval (t_k, t_{k+1}); this input profile is referred to as a *zero-order hold*. Also, the initial condition $x(t_k) = x_k$ is used. The discretization process does not alter the output equation nor the constraints:

$$\begin{aligned} &\text{For all } k: \\ &\quad y_k = g(x_k, t) \\ &\quad x_k \in X \\ &\quad u_k \in U \end{aligned} \tag{3.6}$$

In general, $t_{k+1} = t_k + \Delta t$. We refer to Δt as the *sampling time*. The choice of sampling time is a crucial step in the control of a process. If the sampling time is too short, it may not be possible to measure or control the process due to limitations of the physical equipment or computer processor. On the other hand, if the sampling time is too large, important aspects of the the dynamics of the process may go unnoticed, yielding poor control. In general, the sampling time is chosen small enough that it can capture the fastest dynamics of the process but large enough that the sampling of outputs and calculation of control moves can be accomplished within the allotted time.

In model predictive control, we seek to minimize a cost function that is based on the deviation of the states and the inputs from their targets. To guarantee stability and to achieve the best possible performance, the cost function takes into account all the behavior of the process from the current time onward. Therefore, we seek the sequence $\pi = \{u_0, u_1, u_2, \cdots\}$ that minimizes the following cost function:

$$\Phi = \frac{1}{2}\sum_{k=0}^{\infty} L(x_k, u_k) \tag{3.7}$$

In general, the function $L(x_k, u_k)$ is based on the relative magnitude of x and u and on the importance the control engineer places on each variable. In this document, and in many industrial settings, $L(x_k, u_k)$ is defined as:

$$L(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + (u_k - u_{k-1})^T S (u_k - u_{k-1}) \tag{3.8}$$

Here, Q is the penalty on the states, R is the penalty on the inputs, and S in the penalty on the rate of change of the inputs. R and S must be positive definite, while Q must be positive semi-definite. We refer to (3.8) as the *stage cost* at time k.

3.1 Terminal Equality Constraints

Since a computational optimization scheme is to be employed, it is natural to seek a cost function that does not rely on an infinite number of decision variables as in (3.7). Keerthi and Gilbert [54] proposed a solution with guaranteed stability that utilizes a finite horizon. In their method, the minimization in (3.7) is altered to include a terminal constraint:

$$\begin{aligned} &\min_{\pi} \frac{1}{2} \sum_{k=0}^{N} L(x_k, u_k) \\ \text{subject to: } & x_{k+1} = F(x_k, u_k) \\ & x_k \in X, \qquad u_k \in U, \qquad x_N = 0 \end{aligned} \tag{3.9}$$

The terminal equality constraint, however, poses difficulty to the optimization algorithm. Choosing the horizon length N may be challenging because if N is not chosen to be long enough, (3.9) may not have a feasible solution. Another shortfall of the terminal equality constraint is that the solution to the minimization problem often takes unusual or unnatural paths in order to satisfy the constraint, rather than yield a solution similar to that of the infinite horizon problem.

In order to develop the theory for an improved finite horizon formulation, we first take an excursion into unconstrained linear model predictive control.

3.1.1 Unconstrained linear MPC

Consider the discrete-time linear plant described by

$$x_{k+1} = Ax_k + Bu_k \tag{3.10}$$

$$y_k = Cx_k. \tag{3.11}$$

Now, using the cost function described by (3.8), a new, equivalent cost function is presented by augmenting the state with the previous input, yielding

$$L(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + 2x_k^T M u_k \tag{3.12}$$

in which

$$x_k \leftarrow \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \qquad Q \leftarrow \begin{bmatrix} Q & 0 \\ 0 & S \end{bmatrix} \qquad R \leftarrow \begin{bmatrix} R + S \end{bmatrix} \qquad M \leftarrow \begin{bmatrix} 0 \\ -S \end{bmatrix}. \tag{3.13}$$

By dynamic programming arguments (see [4]), it is straightforward to show that the optimal input that minimizes (3.7) for this linear unconstrained system is the linear feedback equation

$$u_k = K x_k \tag{3.14}$$

in which

$$K = -\left[R + B^T \Pi B\right]^{-1} \left(M^T + B^T \Pi A\right) \tag{3.15}$$

and

$$\Pi = Q + K^T R K + M K + K^T M^T + [A + BK]^T \, \Pi \, [A + BK]\,. \tag{3.16}$$

Equation 3.16 is referred to as the discrete-time algebraic Riccati equation. The solution to this equation may be obtained by an iterative procedure or by a Schur decomposition. By simple algebra, the optimal value of the cost function from any point x_k is

$$\Phi^* = \frac{1}{2} x_k \Pi x_k. \tag{3.17}$$

We now have the necessary fundamentals to return to the general nonlinear case.

3.2 Terminal Inequality Constraint

In practice, satisfying the terminal equality constraint in (3.9) leads to difficulties. For instance, the horizon length N may not be long enough to satisfy the constraint from all states. Without knowing in what operating regime the plant will operate, the operator cannot reliable choose a horizon length that is suitably long but not long enough to be computationally prohibitive. Secondly, the equality constraint for shorter horizons causes the open-loop estimated state trajectory and closed-loop performance to be dissimilar, resulting in unsatisfactory performance. For these reasons, Michalska and Mayne [70] introduced the concept of the terminal region. The final state in the trajectory is constrained to lie inside the terminal region, which is constructed such that a linear control law is sufficient to stabilize the system. The remaining cost from the terminal state is summarized as the cost-to-go or terminal cost term $V(x)$. The inequality constraint can be viewed as a relaxation of the equality constrained case. Since the terminal state may lie anywhere inside the terminal region, this formulation is easier to solve from a practical standpoint. In the regulator with terminal inequality constraint, we solve

$$\begin{aligned} &\min_{\pi} \frac{1}{2} \sum_{k=0}^{N-1} L(x_k, u_k) + \frac{1}{2} V(x_N) \\ \text{subject to: } & x_{k+1} = F(x_k, u_k) \\ & x_k \in X, \qquad u_k \in U, \qquad x_N \in W_\alpha \end{aligned} \tag{3.18}$$

3.2.1 Terminal invariant region

The theoretical fundamentals of the terminal region are based on the optimal linear feedback control law. The formulation is as follows.

Consider a nonlinear system as outlined earlier. We obtain a linear approximation to the system at the origin

$$x_{k+1} = Ax_k + Bu_k \tag{3.19}$$

in which

$$A = \frac{\partial F}{\partial x^T}(0,0) \qquad B = \frac{\partial F}{\partial u^T}(0,0)$$

Using the A and B matrices as determined, along with Q, R, and M from (3.12), we can apply (3.16) and (3.15) to yield values of Π and K that are valid only at the origin. We make the argument that there exists a neighborhood of the origin for which the linear feedback control law is asymptotically stabilizing. This neighborhood serves as the terminal region for the finite horizon controller. We now prove our claim about this region, which we refer to as W.

First, define

$$F_K(x) = F(x, Kx). \tag{3.20}$$

Assume that $\frac{\partial F_K}{\partial x^T}$ is Lipschitz continuous with Lipschitz constant k in a neighborhood $N_x := \{x| \quad \|x\| \leq \epsilon\}$ of the origin.

Now, compare the linear approximation of $F_K(x)$ to the nonlinear function:

$$F_K(x) = F_K(0) + \frac{\partial F_K}{\partial x^T}(0)x + e(x). \tag{3.21}$$

By Lemma 4.1.12 in Dennis and Schnabel [24],

$$\|F_K(x) - F_K(0) - \frac{\partial F_K}{\partial x^T}(0)x\| \leq \frac{k}{2}\|x\|^2 \qquad \forall x \in N_x. \tag{3.22}$$

So, we have

$$\|e(x)\| \leq \frac{k}{2}\|x\|^2 \qquad \forall x \in N_x \tag{3.23}$$

Now, define the Lyapunov function

$$V(x) = x^T \Pi x \tag{3.24}$$

in which Π is defined by (3.16). Then, for all $x \in N_x$:

$$V(F_K(x)) - V(x) = V([A + BK]x) - V(x) + e_1(x) \tag{3.25}$$

in which

$$e_1(x) = e(x)^T \Pi e(x) + 2x^T (A + BK)^T \Pi e(x). \tag{3.26}$$

By (3.23), we can bound $e_1(x)$ with a new constant $k_2 > 0$:

$$\|e_1(x)\| \leq k_2\|x\|^3 \qquad \forall x \in N_x. \tag{3.27}$$

Now, substitute (3.16) into (3.25) to yield

$$V(F_K(x)) - V(x) = -x^T[Q + K^T RK + MK + K^T M^T]x + e_1(x) \tag{3.28}$$

$$\leq -x^T[Q + K^T RK + MK + K^T M^T]x + k_2\|x\|^3 \qquad \forall x \in N_x \tag{3.29}$$

The interpretation of (3.29) is simple; the left side of the inequality is the decrease of the cost-to-go from some point x to the point resulting from applying the optimal linear feedback control law at the origin. If the right side is negative, then asymptotic stability is guaranteed. Since $Q + K^T RK + MK + K^T M^T$ is positive definite, the right side is guaranteed to be non-positive as long as $\|x\|$ is small enough. Therefore, the goal is to construct some region such that (3.29) holds. Define

$$W_\alpha := \{x| \quad x^T \Pi x \leq \alpha\}. \tag{3.30}$$

We choose α small enough that (3.29) is true. We can choose α at its largest (so that the decrease in the cost-to-go is small) or more conservatively for better cost function improvement at each step. For the purposes of this document, we choose α such that the improvement is one half the stage cost improvement of the linear system.

$$V(F_K(x)) - V(x) \leq -\frac{1}{2}x^T[Q + K^T RK + MK + K^T M^T]x \qquad \forall x \in W_\alpha \tag{3.31}$$

We can then shrink α, if necessary, so that all x in W_α obey the state and input constraints under linear feedback control and such that $W_\alpha \subset N_x$, provided that these constraints do not pass through the origin. □

Now, we prove the existence of a non-empty feasible set about the origin. We must state that the origin is feasible in the sense that it violates neither state nor input constraints. Without this assumption, there is no sense in targeting such a point. Therefore, at the very least, the terminal region can collapse to just the origin, and we revert to the terminal equality constraint as described earlier. In fact, this is the case if any of the constraints pass through the origin. That is a special case that we do not address in this document. Restrict the sets U and X to be closed. Therefore, if U and X are closed, $0 \in X$ and $0 \in U$, and 0 is not on the boundary of X or U, then the origin is in the interior of X and U. That means there is some non-empty region $W_c \in X$, $KW_c \in U$ around the origin. $\Box$

The calculation of α for a given nonlinear model is a nontrivial issue. In fact, the resulting optimization problem is a global and nonconvex, so local optima are not sufficient to guarantee the properties described above. It is possible to approach this optimization problem in an intelligent framework, however. We now aim to devise a reliable method for the calculation of the region W_α, as defined by Equation 3.30. The optimization method is global, but it takes advantage of the inherent structure of the problem.

First, let us begin by stating the problem we wish to solve:

$$\max_{x \in W_\alpha} \{V(F_K(x)) - V(x) + \frac{1}{2} x^T [Q + K^T R K + MK + K^T M^T] x\} \leq 0 \tag{3.32}$$

The meaning of Equation 3.32 is that for every $x \in W_\alpha$, the application of the linear feedback control law results in an improvement of at least one-half the improvement expected from a linear plant. Note that there is a trivial maximum to this problem located at the origin, where the function is exactly equal to zero. This extra solution causes the optimizer to return the answer at the origin, which we never wish to use, unless there are constraints that pass through the origin. Therefore, we alter the problem slightly to remove the solution at the origin, but keep the other solutions intact.

Let us refer to the expression on the left side of Equation 3.32 as $h(x)$. The method for removing the solution at the origin is simple; we divide $h(x)$ by a positive definite factor $p(x)$ that has the following properties:

1. $\lim_{x \to 0} \frac{h(x)}{p(x)} < 0$.

2. If $h(x) = 0$, $x \neq 0$, then $\frac{h(x)}{p(x)} = 0$.

3. If $h(x) > 0$, then $\frac{h(x)}{p(x)} > 0$.

4. If $h(x) < 0$, then $\frac{h(x)}{p(x)} < 0$.

We require the first property so that the solution at the origin is ignored. The second property is required so that all other solutions that make $h(x) = 0$, which we refer to as *critical solutions*, remain after the division term is inserted. The final two properties are necessary to maintain the proper relationship between the "max" operator and the inequality in Equation 3.32.

We now suggest the proposed term be used:

$$p(x) = x^T[Q + K^T RK + MK + K^T M^T]x \tag{3.33}$$

Now, we check our properties as listed previously. Firstly,

$$\lim_{x \to 0} \frac{h(x)}{p(x)} = -\frac{1}{2} < 0 \tag{3.34}$$

The last three conditions are true since for all $x \neq 0$, $p(x) > 0$.

We rearrange the problem as a minimization, which is customary for numerical optimization:

$$\min_{x \in W_\alpha} -\frac{h(x)}{p(x)} \geq 0 \tag{3.35}$$

Since the largest possible W_α is desired, we are interested the critical solutions, i.e., $\frac{h(x)}{p(x)} = 0$. Since there is a neighborhood about the origin for which Equation 3.35 is satisfied, we choose the critical solution that yields the smallest value of $\alpha = x^T \Pi x$. This solution determines the size of the region W_α. We must not neglect the constraints, however, as no point in the region may violate a constraint.

We approach this problem by finding the largest possible region that satisfies the constraints, and then shrinking the region by a methodology that takes advantage of the structure of the problem. The final result is the largest terminal region.

Calculation of the largest ellipsoidal region that fits within a linearly constrained set

In most applications, the constraints on the process are linear. We now wish to find the largest ellipsoidal region that fits within a linearly constrained set. This problem is related to a famous related class of problems: finding the largest sphere that fits in a linearly constrained set. This type of problem can be formulated as a linear program [27]; we need only extend the theory to ellipsoids. Figure 3.1 shows the size of the ellipsoid for which we are aiming. The linear constraints can be expressed as:

$$M_x x \geq m_x \qquad M_u u_i \geq m_u \tag{3.36}$$

Recall that in the terminal region, a linear feedback control law is utilized. Therefore, under linear constraints, the constraints are adjusted to:

$$\begin{bmatrix} M_x \\ M_u K \end{bmatrix} x \geq \begin{bmatrix} m_x \\ m_u \end{bmatrix} \tag{3.37}$$

The system is now linearly transformed to scale the ellipsoid to a sphere. In transforming the ellipsoid, the constraints are also transformed. We start by performing an eigenvalue decomposition on Π:

$$\Pi = V \Lambda V^{-1} \tag{3.38}$$

Since Π is positive definite and symmetric, all of the eigenvalues of Π are positive and real. Therefore, Λ is a diagonal matrix with positive elements on the diagonal; we

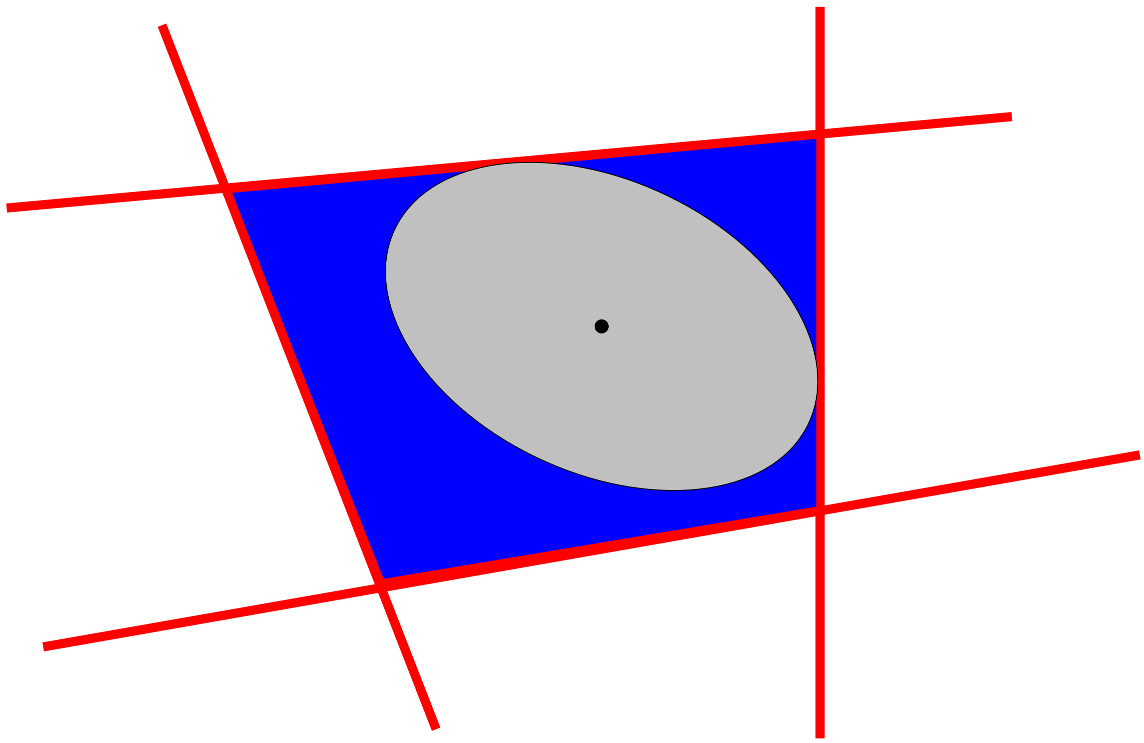

Figure 3.1: The largest ellipsoid centered at the origin that is contained in the feasible region

can take the square root of it easily, and it is invertible. We define a new coordinate system:

$$z = \Lambda^{1/2} V^{-1} x \tag{3.39}$$

So, we can redefine x in terms of z:

$$x = V \Lambda^{-1/2} z \tag{3.40}$$

We now have the familiar problem of finding the largest sphere centered at the origin that satisfies the following constraints:

$$\begin{bmatrix} M_x V \Lambda^{-1/2} \\ M_u K V \Lambda^{-1/2} \end{bmatrix} z \geq \begin{bmatrix} m_x \\ m_u \end{bmatrix} \tag{3.41}$$

Grouping terms, one arrives at the general inequality

$$M_z z \geq m_z \tag{3.42}$$

The goal is to express this problem as a linear program. First, it is necessary to remark that for an arbitrary point c, the distance from c to the plane $M_z^k z = m_z^k$ is given by

$$\frac{|M_z^k c - m_z^k|}{\|M_z^k\|_2} \tag{3.43}$$

where the superscript k refers to the kth row of the system of equations. To simplify the process, we normalize each row of M_z such that $\|M_z^k\|_2 = 1$.

In this problem, the center of the hypersphere is already fixed at the origin, so $c = 0$. We wish to maximize the radius of the sphere, ρ, subject to the ρ being smaller than or equal to the distance from the origin to each plane. This formulation results in:

$$\begin{aligned} &\max_{\rho} && \rho \\ &\text{subject to:} && \rho + m_z^k \leq 0 \qquad k = 1, 2, \ldots, N_c \end{aligned} \tag{3.44}$$

Here, N_c is the number of constraint equations. This linear program is straightforward to solve. A linear program is not necessary, however; an alternative method is to select the smallest (most negative) value of m_z^k, and then the largest radius would be the additive inverse of that value.

We now have a value for the largest ellipsoid that fits in a set of linear constraints:

$$\alpha_0 = x^T \Pi x = z^T z = \rho^2 \tag{3.45}$$

3.2.2 A global optimization technique for calculating the size of the terminal region

The largest ellipsoid that fits within the constraints is now known. It is desired to find the largest ellipsoid that satisfies Equation 3.35, which may be smaller. In order to shrink the ellipsoid, there must be a critical solution in the interior of the ellipsoid. The aim is to find such points, while taking advantage of the structure of the problem. The optimization is global, so there is no guarantee of finding the global solution; there is, however, a high probability of finding it. The optimization technique is stochastic, based loosely on the work of Kan and Timmer [53].

First, a point is selected randomly from the interior of the ellipsoid $x^T \Pi x = \alpha_0$ using a random number generation scheme based on the acceptance-rejection technique of von Neumann [122]. The system is transformed to a hypersphere, as shown earlier. Then, a hypercube with sides of length 2ρ is constructed such that the hypersphere is inscribed in it. A random point is chosen inside the hypercube. If it is within the hypersphere, it is accepted. If it is outside the hypersphere, it is rejected and a new random point is chosen. This method guarantees a uniform distribution of random points inside the hypersphere. The point is then transformed back into the coordinates of the ellipsoid.

Using the random point as an initial guess to a nonlinear program (NLP) solver, the following NLP is solved:

$$\begin{aligned} \min_{x} \quad & x^T \Pi x \\ \text{subject to:} \quad & \frac{h(x)}{p(x)} = 0 \end{aligned} \tag{3.46}$$

The interpretation of this optimization is that we are looking for the point nearest the center of the ellipsoid for which a critical solution exists. Since there may be many local minima for this problem, a random starting point is chosen inside the ellipsoid. Depending on the location of the starting point, a different local minimum may be found.

Suppose that the local minimum occurs at $\bar{x}$. We must now check the value of $\bar{\alpha} = \bar{x}^T \Pi \bar{x}$. If $\bar{\alpha} \geq \alpha_0$, then $\bar{x}$ lies on or outside the original ellipsoid. In this case, we have found a critical solution that is not inside the ellipsoid, so we have not located a critical solution that suggests the need to reduce the size of the region. Therefore, we must reject this solution and we do not alter the size of the ellipsoid.

If $\bar{\alpha} < \alpha_0$, then we have found a critical point in the interior of the ellipsoid. In this case, we shrink the size of the ellipsoid so that it is of size $\bar{\alpha}$. If we find a critical solution closer to the origin, it must be inside this new ellipsoid. Therefore, when we continue, we pick random points from inside this smaller ellipsoid. In this way, we take advantage of the structure of the problem. We do not needlessly sample regions that are known not to have a better solution. For a stochastic optimization such as this, the probability of finding the global optimum increases with such a tactic.

Note that the linear input and state constraints have been dropped from the problem. The reason for removing these constraints is that any point in the ellipsoid,

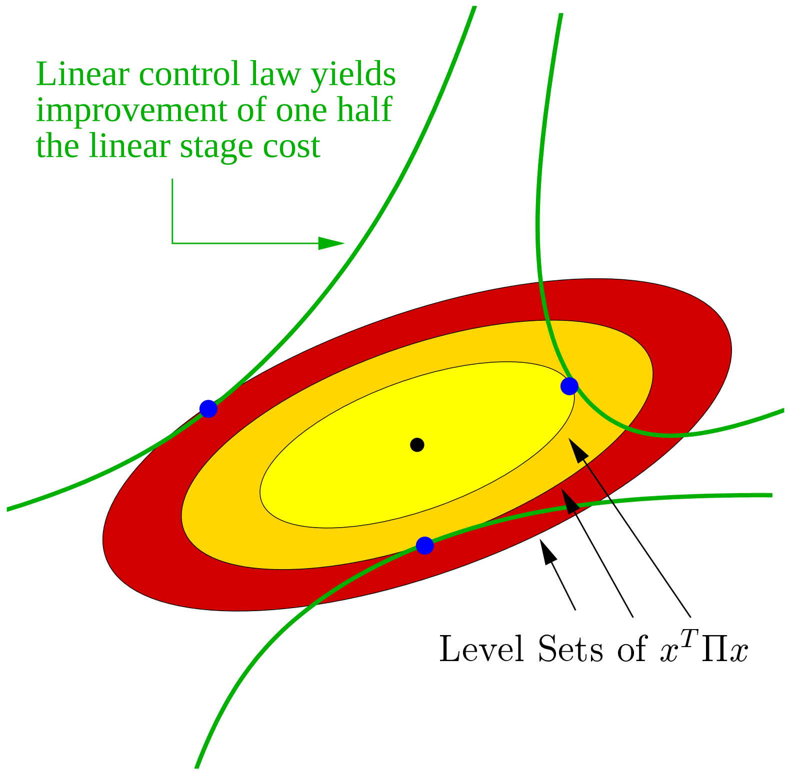

Figure 3.2: Process of reducing the size of an ellipsoid to generate α

including the random point selected, satisfies these constraints. If the local minimum of Equation 3.46 lies inside the ellipsoid, then it satisfies the state and input constraints also. If it lies outside the ellipsoid, the point is rejected. In special cases, the constraints should be left in the problem to exclude any points that may cause the integrator to fail (e.g., negative concentrations or negative absolute temperatures in rate expressions).

Repeating the selection of a random point, solving an NLP, and adjusting the size of the ellipsoid, a solution emerges. This process is outlined in Figure 3.2.2. A question arises regarding the necessary number of random points required to achieve a reliable solution. In computational studies, often ten iterations are adequate. However, since this computation is to be performed off-line, time is not an important consideration. More iterations may be performed when necessary, but there is no guarantee that a global minimum is found. Nevertheless, chemical process dynamics are generally such that a local solution has a large enough basin of attraction to be found easily by a stochastic process.

If the process is to be operated at several set points, it is necessary to calculate a value of α for each target. In such a case, it may be advantageous to build an interpolative solution based on splines to predict the size and orientation of the terminal region from a few known values of α and Π. This method requires further investigation.

Figure 3.3 shows the computed terminal regions for Example 6.9.1 about various steady-state targets. The manifold of possible targets is displayed as the solid line in Figure 3.3. The dashed lines represent the ellipsoidal W_α regions about each target listed in Table 3.1. See Appendix A for a more complete study of this model.

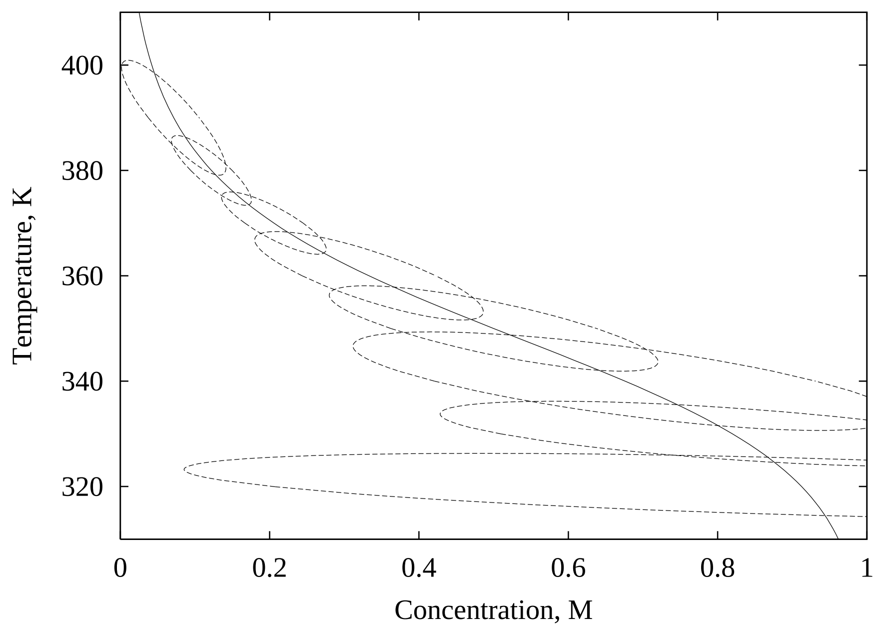

Figure 3.3: Terminal regions for steady-state targets of the CSTR in Example 6.9.1

Target Temperature	**Target Concentration**	α
320 K	0.912 M	534.7
330 K	0.820 M	656.0
340 K	0.676 M	2021
350 K	0.500 M	1879
360 K	0.333 M	2143
370 K	0.206 M	889.6
380 K	0.122 M	659.2
390 K	0.071 M	898.6

Table 3.1: Terminal region data for CSTR in Example 6.9.1

3.3 NMPC Formulation

While the terminal inequality constraint reduces the computational complexity of the nonlinear MPC regulator problem, it still shares the major drawbacks of the terminal equality constraint formulation, to a lesser degree – the suitable choice of a prediction horizon length and the mismatch between open-loop predictions and closed-loop behavior still complicate the approach due to the presence of the terminal constraint. The terminal inequality constraint has the additional complications of determining the terminal region, in most cases off-line, and adding an elliptical constraint that must be approximated in the optimization method. Experience dictates that, provided the terminal penalty is large enough or the prediction horizon is long enough, the terminal constraint is not required at all. For this reason, we choose to keep the terminal penalty but not enforce the terminal constraint. Therefore, the simplified formulation we wish to solve is

$$
\begin{aligned}
&\min_{\pi} \frac{1}{2}\sum_{k=0}^{N-1} L(x_k, u_k) + \frac{1}{2}V(x_N) \\
&\text{subject to: } x_{k+1} = F(x_k, u_k) \\
&\qquad x_k \in X, \qquad u_k \in U.
\end{aligned} \tag{3.47}
$$

While (3.47) is easier to solve than the previously presented methods, it still has one major complication. When constraints are present in the regulator, a solution to the optimization problem is not guaranteed. This difficulty arises from the inability of the optimizer to satisfy the state constraints $x_k \in X$, due to system dynamics, constraints on the inputs, or because the initial state already violates the constraint. Since the regulator is designed to be an on-line implementation, infeasibility of the optimization problems cannot be tolerated. For this reason, we introduce soft constraints.

In the soft constraint formulation, a penalty term is added to the objective function. This term penalizes violations of the constraints. If the penalty term is large enough, then the constrained solution is returned, when possible. Otherwise, the solution that minimizes the constraint violation while still maintaining the cost function at low value is returned. For ease of presentation, we now assume that the state and input constraints are linear, and of the form

$$
D_k u_k - G_k x_k \le d_k, \qquad H_k x_k \le h_k. \tag{3.48}
$$

The formulation of (3.47) with soft constraints is then

$$
\begin{aligned}
&\min_{\pi} \frac{1}{2}\sum_{k=0}^{N-1} L(x_k, u_k) + \frac{1}{2}V(x_N) + \sum_{k=0}^{N} \Xi(\eta_k) \\
&\text{subject to: } x_{k+1} = F(x_k, u_k) \\
&\qquad D_k u_k - G_k x_k \le d_k, \qquad H_k x_k - \eta_k \le h_k, \qquad \eta_k \ge 0
\end{aligned} \tag{3.49}
$$

in which $V(x_N)$ is defiined in (3.24), $L(x_k, u_k)$ is as presented in (3.8) or equivalently (3.12), and

$$\Xi(\eta_k) = \frac{1}{2}\eta_k^T Z_k \eta_k + z_k^T \eta_k \tag{3.50}$$

for positive definite Z_k and positive, and usually large, z_k.

The formulation in (3.49) is the one we consider for the remainder of this study. To demonstrate the ability of this regulation method, we present the following examples, in which the states are directly measured. In later chapters, we will explore other examples in which the states must be estimated using moving horizon estimation.

3.4 Regulation Examples

Example 3.4.1 *Consider a jacketed non-isothermal continuously stirred tank reactor in which the van der Vusse reaction is taking place [17]. The mechanism, sometimes referred to as the van der Vusse reaction, is*

$$\begin{aligned} A \xrightarrow{k_1} B \xrightarrow{k_2} C \\ 2A \xrightarrow{k_3} D \end{aligned} \tag{3.51}$$

For this specific example, Species A represents the reactant cyclopentadiene, B is the product cyclopentenol, Species C is the unwanted side product cyclopentanediol, and Species D is dicyclopentadiene, the product of an undesirable parallel reaction. We operate the reactor at the point of optimal yield of cyclopentenol (see Table 3.2). The flowrate of the feed ($\frac{\dot{V}}{V_R}$) and the heat removal rate ($\dot{Q}_K$) are the manipulated variables, and we measure the concentrations of A and B, the jacket temperature, and the reactor temperature to achieve our desired operating point. In this example, all *the states are measured outputs. The differential equations for this system are*

$$\begin{aligned}
\dot{c}_A &= \frac{\dot{V}}{V_R}(c_{A_0} - c_A) - k_1(T)c_A - k_3(T)c_A^2 \\
\dot{c}_B &= -\frac{\dot{V}}{V_R}c_B + k_1(T)c_A - k_2(T)c_B \\
\dot{T} &= \frac{\dot{V}}{V_R}(T_0 - T) - \frac{1}{\rho C_p}\left(k_1(T)c_A\Delta H_{R_{AB}} + k_2(T)c_B\Delta H_{R_{BC}} + k_3(T)c_A^2\Delta H_{R_{AD}}\right) \\
&\quad + \frac{k_w A_R}{\rho C_p V_R}(T_K - T) \\
\dot{T}_K &= \frac{1}{m_K C_{PK}}\left(\dot{Q}_k + k_w A_R(T - T_K)\right)
\end{aligned}$$

and the parameter values are listed in Table 3.3. The rate constants $k_i(T)$ are determined by the formula

$$k_i(T) = k_{i,0}e^{-\frac{E_i}{T+273.15}}. \tag{3.52}$$

Variable	**Set-point Value**
$\frac{\dot{V}}{V_R}$	14.19 h^{-1}
$\dot{Q}_k$	-1113.5 $\frac{\mathrm{kJ}}{\mathrm{h}}$
c_A	2.14 $\frac{\mathrm{mol}}{\mathrm{L}}$
c_B	1.09 $\frac{\mathrm{mol}}{\mathrm{L}}$
T	114.2 °C
T_K	112.9 °C

Table 3.2: Set-point for the jacketed CSTR example

We demonstrate a start-up operation for this process. We assume the initial conditions are such that the reactor contains no species A or B. The jacket and reactor temperature are both equal to the feed temperature (104.9 $°C$). The problem is characterized by a change of sign in the steady-state gain at the operating point, meaning linear controllers cannot stabilize this reactor without sacrificing performance. Figures 3.4 through 3.9 show the results of applying the nonlinear MPC algorithm to this problem with a sampling time of 20 seconds and a horizon length of 30 time steps. The weighting matrices for this example are

$$Q = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix} \qquad R = \begin{bmatrix} 0.5 & 0 \\ 0 & 5 \times 10^{-7} \end{bmatrix} \tag{3.53}$$

The inputs of the system are constrained such that the normalized flow rate must be between 3 and 35 hr^{-1} and the heat removal rate is between -9000 and 0 kJ/hr.

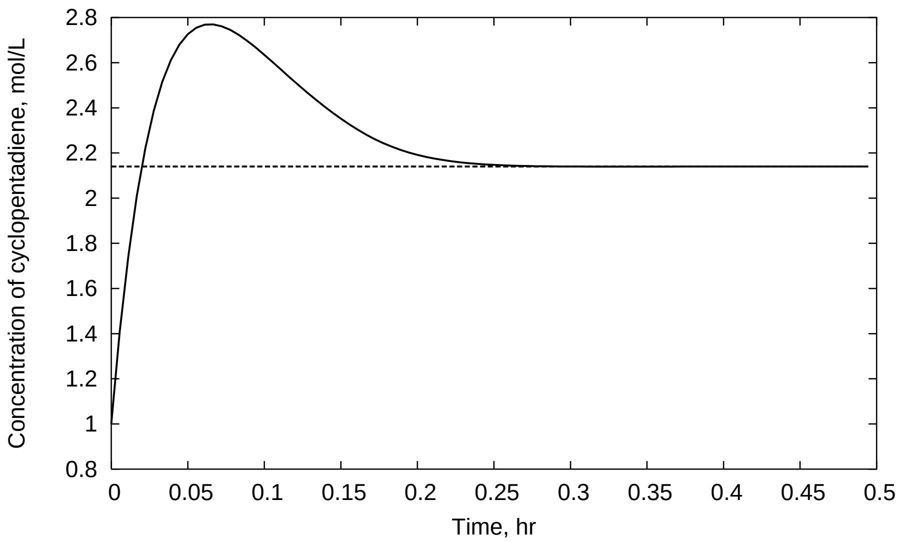

Figure 3.4: Measured cyclopentadiene concentration in Example 3.4.1

Note that the regulator is able to stabilize the process after saturating one of the input constraints for the first third of the simulation. Overall, the performance of the NMPC regulator on this example is excellent. We now turn to the next example.

Parameter	Value	Parameter	Value
c_{A_0}	$5.1 \ \frac{\text{mol}}{L}$	ρ	$0.9342 \ \frac{\text{kg}}{\text{m}^3}$
T_0	$104.9\ ^\circ\text{C}$	C_p	$3.01 \ \frac{\text{kJ}}{\text{kg}\cdot\text{K}}$
$\Delta H_{R_{AB}}$	$4.2 \ \frac{\text{kJ}}{\text{kg}\cdot\text{K}}$	k_w	$4032 \ \frac{\text{kJ}}{\text{h}\cdot\text{m}^2\cdot\text{K}}$
$\Delta H_{R_{BC}}$	$-11.0 \ \frac{\text{kJ}}{\text{kg}\cdot\text{K}}$	A_R	$0.215\ \text{m}^2$
$\Delta H_{R_{AD}}$	$-41.85 \ \frac{\text{kJ}}{\text{kg}\cdot\text{K}}$	V_R	10.0 L
C_{PK}	$2.0 \ \frac{\text{kJ}}{\text{kg}\cdot\text{K}}$	m_K	5.0 kg
$k_{1,0}$	$1.287 \times 10^{12}\ \text{h}^{-1}$	E_1	-9758.3 K
$k_{2,0}$	$1.287 \times 10^{12}\ \text{h}^{-1}$	E_2	-9758.3 K
$k_{3,0}$	$9.043 \times 10^{9} \ \frac{\text{L}}{\text{h}\cdot\text{mol}}$	E_3	-8560 K

Table 3.3: Model parameters for the jacketed CSTR in Example 3.4.1

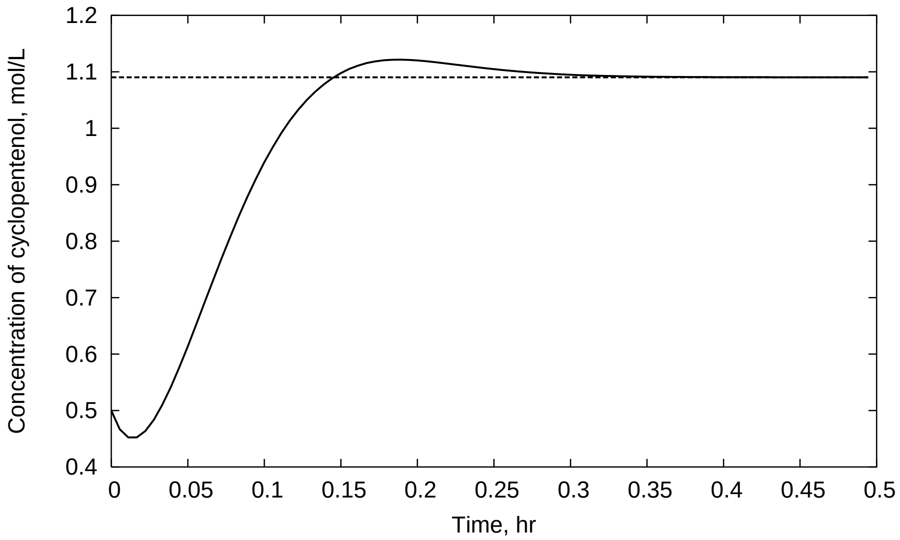

Figure 3.5: Measured cyclopentenol concentration in Example 3.4.1

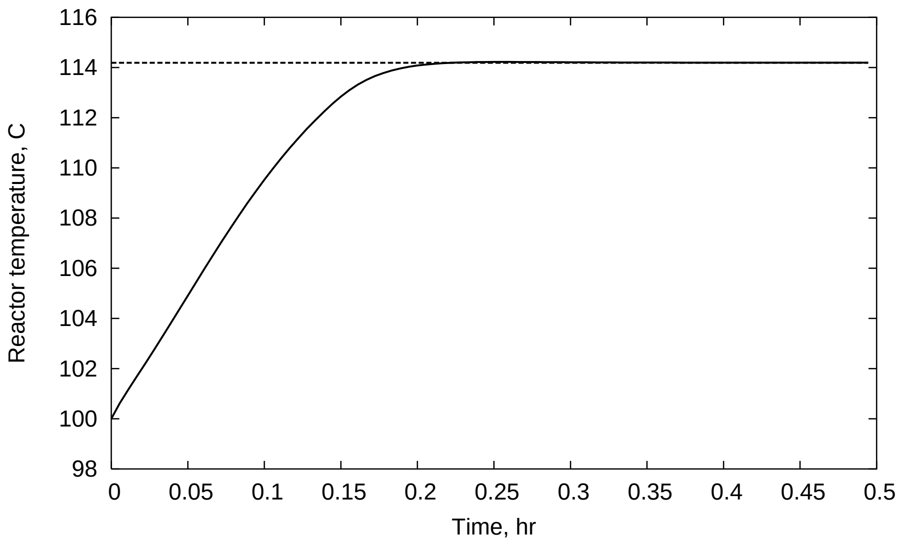

Figure 3.6: Measured reactor temperature in Example 3.4.1

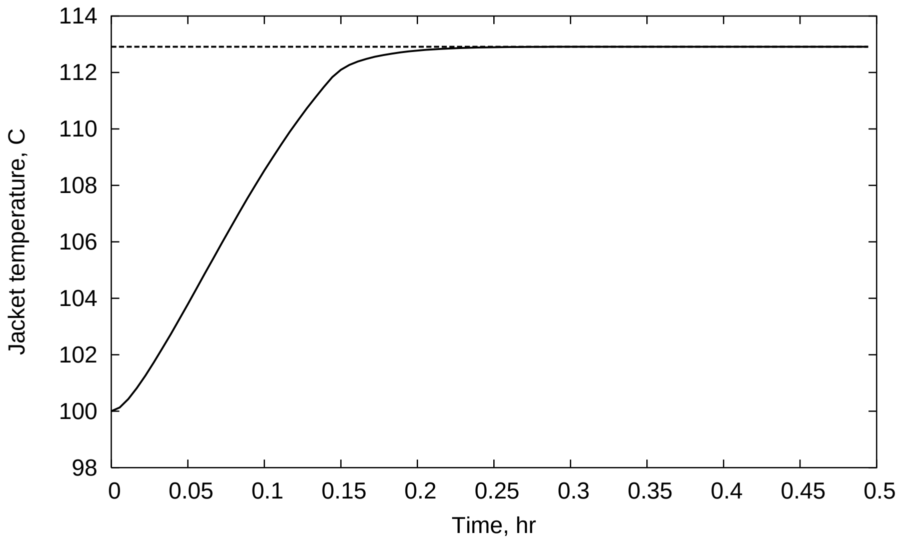

Figure 3.7: Measured jacket temperature in Example 3.4.1

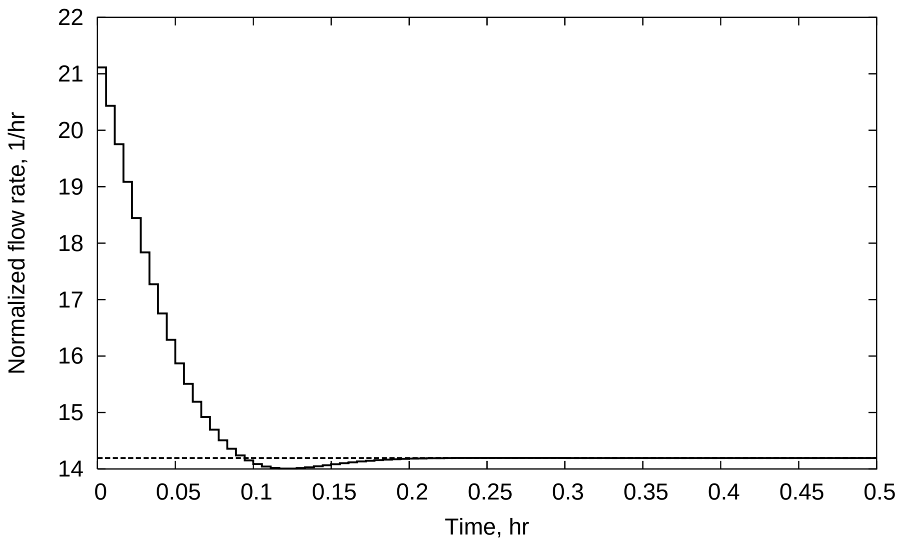

Figure 3.8: Manipulated flow rate in Example 3.4.1

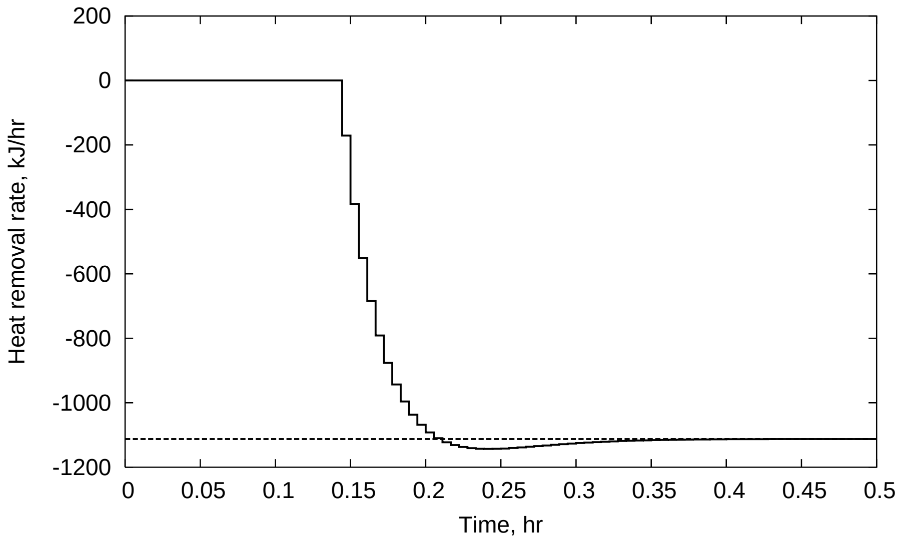

Figure 3.9: Manipulated heat removal rate in Example 3.4.1

Example 3.4.2 *Hauser and Osinga [37] present an inverted pendulum that is attached to a cart. The pendulum must be kept vertical by adjusting the external force applied to the cart. The system is modeled by the equations*

$$\begin{aligned} \dot{x_1} &= x_2 && (3.54) \\ \dot{x_2} &= \frac{\frac{g}{l}\sin(x_1) - \frac{1}{2}m_r x_2^2 \sin(2x_1) - \frac{m_r}{ml} u \cos(x_1)}{\frac{4}{3} - m_r \cos^2(x_1)} && (3.55) \end{aligned}$$

in which

$$m_r = \frac{m}{m+M} \tag{3.56}$$

and $g = 9.8\ m/s^2$, $m = 2\ kg$, $M = 8\ kg$, and $l = 0.5\ m$. The states x_1 and x_2 are the angular velocity and angular acceleration, in units of 1/s and $1/s^2$, respectively. The input u is the external force, in N. The set point of this system is the origin.

The system in Example 3.4.2 is sampled at a rate of 0.01 seconds^{-1} with a prediction horizon of 0.3 seconds. Although with current technology, this system is sampled too fast to expect real-time performance, the key features of an unstable set point and nonlinear dynamics are still appropriate for this study. The initial state of the system is $\begin{bmatrix} -3.5 & 5.9 \end{bmatrix}$. The regulator is tuned as

$$Q = \begin{bmatrix} .1 & 0 \\ 0 & .05 \end{bmatrix}, \qquad R = .01 \tag{3.57}$$

The results of the nonlinear MPC regulator on this system are shown in Figures 3.10 to 3.11. It is evident that the controller is easily able to stabilize the open-loop unstable set point of this example.

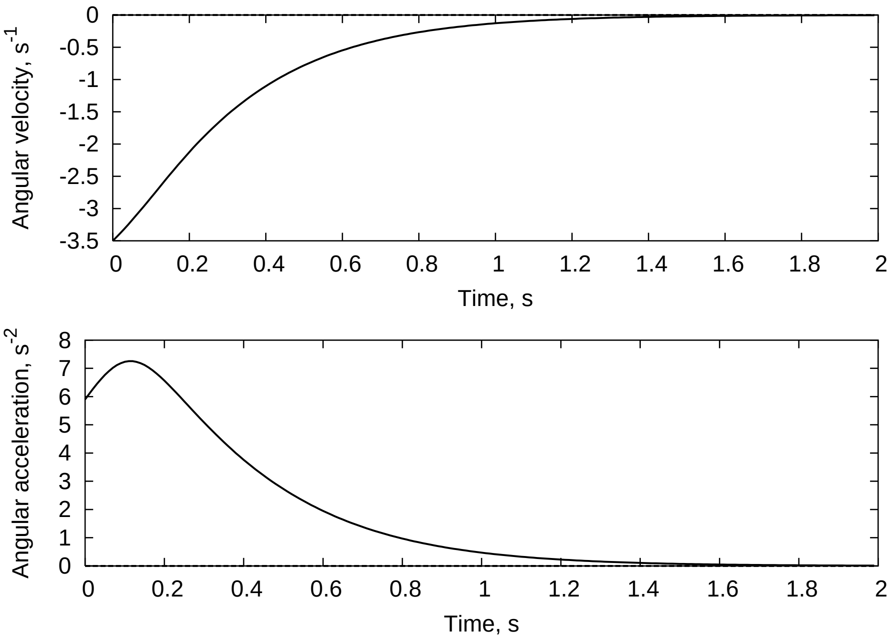

Figure 3.10: Measured outputs for inverted pendulum in Example 3.4.2

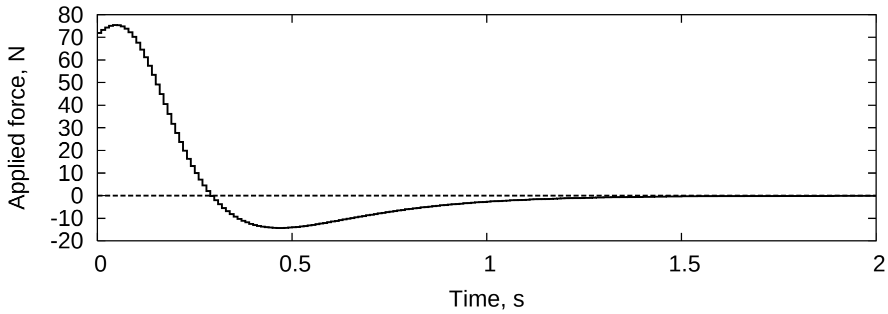

Figure 3.11: Inputs for inverted pendulum in Example 3.4.2

3.5 Conclusions

In this chapter, we present the evolution of the MPC regulator for nonlinear models from the infinite horizon formulation to the form we consider for the remainder of our study, the unconstrained regulator with terminal state penalty. Along the way, we present a study of terminal regions, including theory,a structured global optimization approach to determine the size of such regions, and an example of these regions on a simple problem. Using the regulator formulation as presented, we demonstrate excellent controller performance on a few simple nonlinear examples.

Chapter 4

Moving Horizon Estimation [1]

There are three kinds of lies: lies, damned lies, and statistics.
— Mark Twain

In recent years, researchers have investigated computational strategies for model predictive control, leaving the dual problem of moving horizon state estimation (MHE) largely untouched. Several recently published papers present efficient nonlinear MPC regulators [2, 28, 79, 86, 116, 11]. The nonlinear state estimation problem often is assumed solved by the extended Kalman filter (EKF). However, the EKF has been shown to fail on simple examples [14]. On the other hand, moving horizon estimation, handles constraints and nonlinear models naturally in its framework, and has been successful in estimating states of nonlinear models for which the EKF fails [114]. In this chapter, we address the practical issues of implementing MHE on a constrained nonlinear system and how the state estimation problem is tied to the nonlinear model predictive control framework. We present the formulation of the state estimation problem and derive two types of covariance updates for the *a priori* state estimate.

[1] **Portions of this chapter appear in Tenny, Rawlings, and Bindlish [116] and Tenny and Rawlings [114, 115]**

We handle the interactions between the state estimator and the MPC regulator by augmenting the system for integrated disturbances and nonlinear target calculations. We conclude the chapter with simple examples to highlight the features of the approach.

4.1 MHE Formulation

We consider the problem of estimating the state of systems modeled by

$$\begin{aligned} \dot{x} &= f(x, u, w, t) \\ y &= g(x, t) + v. \end{aligned} \tag{4.1}$$

Discretizing this model by numerical integration yields

$$\begin{aligned} x_{k+1} &= F(x_k, u_k, w_k, t_k) \\ y &= g(x_k, t_k) + v_k \end{aligned} \tag{4.2}$$

in which the states and disturbances satisfy

$$Hx_k \leq h, \qquad Sw_k \leq s, \qquad \Gamma v_k \leq \gamma. \tag{4.3}$$

We assume that the origin satisfies the constraints on w_k and v_k.

The MHE formulation we study is

$$\min \quad \Gamma_e(\rho) + \frac{1}{2} \sum_{k=j-N_e}^{j} \mathcal{L}_e(w_k, v_k) \tag{4.4a}$$

subject to:

$$x_0 = \bar{x}_0 + \rho \tag{4.4b}$$

$$x_{k+1} = F(x_k, u_k, w_k, t_k) \tag{4.4c}$$

$$y_k = g(x_k, t_k) + v_k \tag{4.4d}$$

$$Hx_k \leq h, \qquad Sw_k \leq s, \qquad \Gamma v_k \leq \gamma \tag{4.4e}$$

in which $\mathcal{L}_e(w_k, v_k) = w_k^T Q_w w_k + v_k^T R_v v_k$ and $\Gamma_e(\rho) = \frac{1}{2}\rho^T P_e \rho + p_e^T \rho$. In this formulation, R_v^{-1} is the covariance of the output noise v_k and Q_w^{-1} is the covariance of the state noise w_k. The term $\Gamma_e(\rho)$ approximates the arrival cost, which summarizes the effects of past information before $k = 0$. The weighting term P initially represents the inverse of the covariance of the *a priori* state estimate $\bar{x}_0$, and p_e starts as zero. The variable t_k represents the time at instant k. A few strengths of the MHE formulation are that constraints are incorporated into the framework of the problem, the resulting estimates are optimal and the method possesses excellent stability properties [97]. In the full-information problem, the horizon length N_e increases with j at each sampling time such that the two are equal as both tend toward infinity. However, in MHE, N_e grows to a specified horizon length N_T, at which point $\bar{x}_0$, P_e, and p_e are updated to reflect previous estimates of x_1. By selecting the arrival cost penalties P_e and p_e appropriately, MHE can approximate the full-information problem.

4.1.1 Filtering and smoothing MHE.

The standard way to implement MHE is as a filter. In this formulation, at time $N_T + 1$, we set $\bar{x}_0$ to $x_{1|0}$, the estimate of x_1 given measurements at time 0, and update $P_e = \Pi_{1|0}^{-1}$ accordingly. In the filtering update, $p_e = 0$. Note that when starting MHE, the *a priori* estimate is updated to a new estimate based at first on only one data point. For unconstrained linear systems, this method is equivalent to the Kalman filter.

The smoothing formulation takes advantage of more information by including more data in the $\bar{x}_0$ update. In this scenario, $\bar{x}_0$ becomes $x_{1|N_T}$, the estimate of x_1 at time N_T. The formula for updating P_e and p_e for time invariant linear systems was presented by Rao, et al. [96]. For unconstrained time invariant linear systems, the smoothing update is also equivalent to the Kalman filter. We derive a corresponding formula for nonlinear dynamical systems based on approximating the nonlinear model as a time-varying linear function. We make use of the following definitions:

$$A_k = \frac{\partial F}{\partial x_k^T}, \qquad G_k = \frac{\partial F}{\partial w_k^T}, \qquad C_k = \frac{\partial g}{\partial x_k^T}. \tag{4.5}$$

Also, we define d_k and f_k such that

$$x_{k+1} = A_k x_k + G_k w_k + f_k \tag{4.6a}$$

$$y_k = C_k x_k + v_k + d_k \tag{4.6b}$$

The derivation of the smoothing update is as follows. First, we draw the connection between the MHE least squares problem and its probabilistic interpretation. Namely, the act of minimizing the MHE cost function is the same as calculating the maximum likelihood estimate. Consider the vector x with distribution $p(x) \sim \mathcal{N}(m, P)$. For this system, $p(x)$ is expressed as

$$p(x) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} \exp\left[-\frac{1}{2}(x-m)^T P^{-1}(x-m)\right]. \tag{4.7}$$

Since we are interested in the value of x that maximizes $p(x)$, we want

$$\begin{aligned} x^* &= \arg\max_x \quad p(x) && (4.8)\\ &= \arg\max_x \quad \log(P) \qquad \text{(by monotonicity of log(P))} && (4.9)\\ &= \arg\min_x \quad -\log(P) && (4.10)\\ &= \arg\min_x \quad \frac{1}{2}(x-m)^T P^{-1}(x-m) && (4.11) \end{aligned}$$

Naturally, when choosing the penalties for w_k and v_k in the MHE formulation, their respective means are 0, and the penalties P^{-1} are Q_w and R_v, respectively. To choose the appropriate penalty for the deviation from the *a priori* state estimate x_0, we must first derive its covariance by probabilistic arguments. We begin by revisiting the interpretation of the smoothing problem for MHE introduced by Rao [93]. Assume that the modeled system is Markov, meaning that the current state depends only on the prior state. Define the notation

$$\begin{aligned} Y_1^n &= \{y_1, y_2, \ldots, y_{n-1}, y_n\} && (4.12)\\ X_1^n &= \{x_1, x_2, \ldots, x_{n-1}, x_n\}. && (4.13) \end{aligned}$$

In MHE, we are interested in the conditional probability $p(X_{j-N_e}^T|Y_0^j)$, the probability of the states x within an estimation horizon of length N given all the measurements up to time j.

$$
\begin{aligned}
&p(X_{j-N_e}^{j+1}|Y_0^j) \\
&\quad = p(x_{j-N_e}|Y_0^j)p(X_{j-N_e+1}^{j+1}|x_{j-N_e}, Y_0^j) && (4.14)\\
&\quad = p(x_{j-N_e}|Y_0^j)\frac{p(X_{j-N_e+1}^{j+1}, Y_0^j|x_{j-N_e})}{p(Y_0^j|x_{j-N_e})} && (4.15)\\
&\quad = \frac{p(x_{j-N_e}|Y_0^j)}{p(Y_0^j|x_{j-N_e})}p(X_{j-N_e+1}^{j+1}|x_{j-N_e})p(Y_0^j|X_{j-N_e}^{j+1}) && (4.16)\\
&\quad = \frac{p(x_{j-N_e}|Y_0^j)}{p(Y_0^j|x_{j-N_e})}p(Y_0^{j-N_e-1}|X_{j-N_e}^{j+1}, Y_{j-N_e}^j)p(X_{j-N_e+1}^{j+1}|x_{j-N_e})p(Y_{j-N_e}^j|X_{j-N_e}^{j+1}) && (4.17)\\
&\quad = \frac{p(x_{j-N_e}|Y_0^j)}{p(Y_0^j|x_{j-N_e})}p(Y_0^{j-N_e-1}|X_{j-N_e}^{j+1}, Y_{j-N_e}^j)\left[\prod_{k=j-N_e}^{j} p(x_{k+1}|x_k)p(y_k|x_k)\right] && (4.18)\\
&\quad = \frac{p(x_{j-N_e}|Y_0^j)}{p(Y_{j-N_e}^j|x_{j-N_e})}\frac{p(Y_0^{j-N_e-1}|X_{j-N_e}^{j+1}, Y_{j-N_e}^j)}{p(Y_0^{j-N_e-1}|x_{j-N_e}, Y_{j-N_e}^j)}\left[\prod_{k=j-N_e}^{j} p(x_{k+1}|x_k)p(y_k|x_k)\right] && (4.19)\\
&\quad = \frac{p(x_{j-N_e}|Y_0^j)}{p(Y_{j-N_e}^j|x_{j-N_e})}\frac{p(X_{j-N_e+1}^{j+1}|x_{j-N_e}, Y_0^j)}{p(X_{j-N_e+1}^{j+1}|x_{j-N_e}, Y_{j-N_e}^j)}\left[\prod_{k=j-N_e}^{j} p(x_{k+1}|x_k)p(y_k|x_k)\right] && (4.20)\\
&\quad = \frac{p(x_{j-N_e}|Y_0^j)}{p(Y_{j-N_e}^j|x_{j-N_e})}\left[\prod_{k=j-N_e}^{j} p(x_{k+1}|x_k)p(y_k|x_k)\right] && (4.21)
\end{aligned}
$$

From the smoothing formulation in (4.21), we can recover the filtering formulation by manipulating the first term:

$$\frac{p(x_{j-N_e}|Y_0^j)}{p(Y_{j-N_e}^j|x_{j-N_e})} = \frac{p(x_{j-N_e}, Y_0^j)}{p(Y_{j-N_e}^j|x_{j-N_e})p(Y_0^j)} \tag{4.22}$$

$$= \frac{p(x_{j-N_e}, Y_{j-N_e}^j|Y_0^{j-N_e-1})p(Y_0^{j-N_e-1})}{p(Y_{j-N_e}^j|x_{j-N_e})p(Y_0^j)} \tag{4.23}$$

$$= \frac{p(Y_{j-N_e}^j|x_{j-N_e}, Y_0^{j-N_e-1})p(x_{j-N_e}|Y_0^{j-N_e-1})p(Y_0^{j-N_e-1})}{p(Y_{j-N_e}^j|x_{j-N_e})p(Y_0^j)} \tag{4.24}$$

$$= \frac{p(Y_{j-N_e}^j|x_{j-N_e})p(x_{j-N_e}|Y_0^{j-N_e-1})p(Y_0^{j-N_e-1})}{p(Y_{j-N_e}^j|x_{j-N_e})p(Y_0^j)} \tag{4.25}$$

$$= \frac{p(x_{j-N_e}|Y_0^{j-N_e-1})p(Y_0^{j-N_e-1})}{p(Y_0^j)} \tag{4.26}$$

$$= \frac{p(x_{j-N_e}|Y_0^{j-N_e-1})}{p(Y_{j-N_e}^j|Y_0^{j-N_e-1})}. \tag{4.27}$$

Substituting (4.27) into (4.21) yields the filtering formulation of MHE:

$$p(X_{j-N_e}^{j+1}|Y_0^j) = \frac{p(x_{j-N_e}|Y_0^{j-N_e-1})}{p(Y_{j-N_e}^j|Y_0^{j-N_e-1})}\left[\prod_{k=j-N_e}^{j} p(x_{k+1}|x_k)p(y_k|x_k)\right]. \tag{4.28}$$

The smoothing and filtering forms are equivalent for linear unconstrained models, and are equal to the Kalman filter. For nonlinear systems, and constrained linear systems, the two probabilities are not equal.

We represent the nonlinear model as a time varying linear model as shown in (4.6). Assume that the states and all noise sequences are uncorrelated and Gaussian with the distributions

$$w_k \sim \mathcal{N}(0, Q_w^{-1}) \tag{4.29}$$

$$v_k \sim \mathcal{N}(0, R_v^{-1}) \tag{4.30}$$

and that we define W and V in a way consistent with (4.12). Then the MHE formulation gives rise to a least-squares problem subject to the model (4.6). For the smoothing formulation of MHE, this problem corresponds to

$$\max_{X_{j-N_e}^{j+1}} \log p(X_{j-N_e}^{j+1}|Y_0^j)$$

$$= \max_{X_{j-N_e}^{j+1}} \log p(x_{j-N_e}|Y_0^j) - \log p(Y_{j-N_e}^j|x_{j-N_e}) + \sum_{k=j-N_e}^{j} [\log p(x_{k+1}|x_k) + \log p(y_k|x_k)] \tag{4.31}$$

$$= \min_{X_{j-N_e}^{j+1}} -\log p(x_{j-N_e}|Y_0^j) + \log p(Y_{j-N_e}^j|x_{j-N_e}) + \frac{1}{2}\sum_{k=j-N_e}^{j} \left[w_k^T Q_w w_k + v_k^T R_v v_k\right]. \tag{4.32}$$

4.1.2 Approximation of $p(Y_{j-N_e}^j|x_{j-N_e})$ for the Smoothing Update for Nonlinear Models

While the statistics of the term $p(x_{j-N_e}|Y_0^j)$ in (4.32) are known to be $\mathcal{N}(\bar{x}_{j-N_e|j}, \Pi_{j-N_e|j})$, the other probability must be derived. Therefore, we must calculate $p(Y_{j-N_e}^j|x_{j-N_e})$. Define

$$\mathcal{O} = \begin{bmatrix} C_{j-N_e+1} \\ C_{j-N_e+2}A_{j-N_e+1} \\ C_{j-N_e+3}A_{j-N_e+2}A_{j-N_e+1} \\ \vdots \\ C_jA_{j-1}A_{j-2}\dots A_{j-N_e+1} \end{bmatrix} \qquad \mathcal{Y} = \begin{bmatrix} y_{j-N_e+1} \\ y_{j-N_e+2} \\ y_{j-N_e+3} \\ \vdots \\ y_j \end{bmatrix} \tag{4.33}$$

$$\mathcal{F} = \begin{bmatrix} d_{j-N_e+1} \\ d_{j-N_e+2} + C_{j-N_e+2} f_{j-N_e+1} \\ d_{j-N_e+3} + C_{j-N_e+3}(A_{j-N_e+2} f_{j-N_e+1} + f_{j-N_e+2}) \\ d_{j-N_e+4} + C_{j-N_e+4}(A_{j-N_e+3} A_{j-N_e+2} f_{j-N_e+1} + A_{j-N_e+3} f_{j-N_e+2} + f_{j-N_e+3}) \\ \vdots \end{bmatrix}. \tag{4.34}$$

The matrix $\mathcal{M}$ is defined for $k, l \leq N_e$ by

$$\mathcal{M}_{k,l} = \begin{cases} 0 & \text{if } l \geq k, \\ C_{k+j-N_e} G_{l+j-N_e} & \text{if } l = k-1, \\ C_{k+j-N_e} A_{k+j-N_e-1} A_{k+j-N_e-2} \ldots A_{l+j-N_e+1} G_{l+j-N_e} & \text{otherwise.} \end{cases} \tag{4.35}$$

Then we have the relationship

$$\begin{bmatrix} Y_{j-N_e}^{j} \\ x_{j-N_e} \\ W_{j-N_e}^{j-1} \end{bmatrix} = \begin{bmatrix} I & \mathcal{O} & \mathcal{M} \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} V_{j-N_e}^{j} \\ x_{j-N_e} \\ W_{j-N_e}^{j-1} \end{bmatrix} + \begin{bmatrix} \mathcal{F} \\ 0 \\ 0 \end{bmatrix}. \tag{4.36}$$

The distribution for $p(V_{j-N_e}^{j}, x_{j-N_e}, W_{j-N_e}^{j-1})$ is

$$p(V_{j-N_e}^{j}, x_{j-N_e}, W_{j-N_e}^{j-1}) \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \bar{x}_{j-N_e} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathcal{R} & & \\ & \Pi_{j-N_e} & \\ & & \mathcal{Q} \end{bmatrix} \right) \tag{4.37}$$

in which

$$\mathcal{R} = \operatorname{diag}(R_v^{-1}), \qquad \mathcal{Q} = \operatorname{diag}(Q_w^{-1}). \tag{4.38}$$

Using (4.36) and (4.37), we solve for the distribution of $p(Y_{j-N_e}^{j}, x_{j-N_e}, W_{j-N_e}^{j-1})$:

$$p(Y_{j-N_e}^{j}, x_{j-N_e}, W_{j-N_e}^{j-1}) \sim \mathcal{N}\left(\begin{bmatrix} \mathcal{O}\bar{x}_{j-N_e} \\ \bar{x}_{j-N_e} \\ 0 \end{bmatrix} + \begin{bmatrix} \mathcal{F} \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathcal{R} + \mathcal{O}\Pi_{j-N_e}\mathcal{O}^T + M\mathcal{Q}M^T & \mathcal{O}\Pi_{j-N_e} & M\mathcal{Q} \\ \Pi_{j-N_e}\mathcal{O}^T & \Pi_{j-N_e} & 0 \\ \mathcal{Q}M^T & 0 & \mathcal{Q} \end{bmatrix}\right) \tag{4.39}$$

From (4.39), we obtain the marginal probability density $p(Y_{j-N_e}^{j}, x_{j-N_e})$:

$$p(Y_{j-N_e}^{j}, x_{j-N_e}) \sim \mathcal{N}\left(\begin{bmatrix} \mathcal{O}\bar{x}_{j-N_e} \\ \bar{x}_{j-N_e} \end{bmatrix} + \begin{bmatrix} \mathcal{F} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathcal{R} + \mathcal{O}\Pi_{j-N_e}\mathcal{O}^T + M\mathcal{Q}M^T & \mathcal{O}\Pi_{j-N_e} \\ \Pi_{j-N_e}\mathcal{O}^T & \Pi_{j-N_e} \end{bmatrix}\right) \tag{4.40}$$

Hence the desired conditional probability is

$$p(Y_{j-N_e}^{j}|x_{j-N_e}) \sim \mathcal{N}\left(\mathcal{O}x_{j-N_e} + \mathcal{F}, \mathcal{R} + M\mathcal{Q}M^T\right). \tag{4.41}$$

Define $\mathcal{W} = \mathcal{R} + M\mathcal{Q}M^T$. Then

$$\log p(Y_{j-N_e}^{j}|x_{j-N_e}) = \frac{1}{2}(\mathcal{Y} - \mathcal{O}x_{j-N_e} - \mathcal{F})^T\mathcal{W}^{-1}(\mathcal{Y} - \mathcal{O}x_{j-N_e} - \mathcal{F}) \tag{4.42}$$

in which $\mathcal{Y}$ is defined by (4.33).

4.1.3 Smoothing Implementation

For the implementation of the smoothing update, we start with the covariance of our *a priori* estimate given no current measurement, $\Pi_{j-N_e|j-N_e-1}$. This covariance is initially supplied by the user at startup as $\Pi_{0|-1}$ and updated after $j = N_e$. At each subsequent time, we propagate the covariance forward to yield the covariance $\Pi_{j|j}$

using the formulas

$$\Pi_{k|k} = \Pi_{k|k-1} - \Pi_{k|k-1} C_k^T \left(R + C_k^T \Pi_{k|k-1} C_k\right)^{-1} C_k \Pi_{k|k-1} \tag{4.43}$$

$$\Pi_{k+1|k} = G_k Q G_k^T + A_k \Pi_{k|k} A_k^T. \tag{4.44}$$

Since the smoothing *a priori* estimate is $x_{j-N_e+1|j}$, the required covariance is $\Pi_{j-N_e+1|j}$. Using the values derived above, and starting with $\Pi_{j|j}$ at $k = j-1$, the smoothing covariance is calculated by

$$\Pi_{k|j} = \Pi_{k|k} + \Pi_{k|k} A_k^T \Pi_{k+1|k}^{-1} \left(\Pi_{k+1|j} - \Pi_{k+1|k}\right) \Pi_{k+1|k}^{-1} A_k \Pi_{k|k}. \tag{4.45}$$

As described earlier, the penalty P used in the optimization is not $\Pi_{j-N_e+1|j}^{-1}$, since doing so results in counting the effects of the measurements in the data window twice. For this reason, a second term is required to remove these influences from the penalty term. This term is exactly (4.42).

We then conclude that the penalties on the *a priori* state estimates at time $j+1$ for the smoothing problem are

$$P = \Pi_{j-N_e+1|j}^{-1} - \mathcal{O}^T \mathcal{W}^{-1} \mathcal{O} \tag{4.46}$$

$$p = \mathcal{O}^T \mathcal{W}^{-1} \left(\mathcal{Y} - \mathcal{O} x_{j-N_e+1|j} - \mathcal{F}\right). \tag{4.47}$$

4.2 Integrating Disturbances

In cases such as plant/model mismatch and unmeasured nonzero mean disturbances, it is necessary to shift the state and input targets such that the resulting system does not exhibit offset due to prediction error. To accomplish this goal, we introduce an integrated disturbance term into the estimator and subsequently perform a nonlinear target calculation. The estimator formulation in Equation 4.4 becomes

$$\min \quad \Gamma_e(\rho, \rho_p) + \sum_{k=j-N_e}^{j} \mathcal{L}_e(w_k, v_k, \xi_k) \tag{4.48a}$$

subject to:

$$x_{j-N_e} = \bar{x}_{j-N_e} + \rho, \quad p_{j-N_e} = \bar{p}_{j-N_e} + \rho_p \tag{4.48b}$$

$$x_{k+1} = F(x_k, u_k + X_u p_k, w_k, t_k) \tag{4.48c}$$

$$p_{k+1} = p_k + \xi_k \tag{4.48d}$$

$$y_k = g(x_k, t_k) + X_y p_k + v_k \tag{4.48e}$$

$$Hx_k \leq h, \quad Sw_k \leq s, \quad \Gamma v_k \leq \gamma. \tag{4.48f}$$

in which we have introduced the integrated disturbances p_k. The disturbance enters the system through some combination of the inputs and the outputs by way of the matrices X_u and X_y, respectively. The current estimate of the integrated error in the system is represented by $\bar{p}_{j-N_e}$ with estimates of the future evolution of p_k influenced only by the presumed random, zero-mean disturbance ξ_k. The correction to $\bar{p}_{j-N_e}$ is denoted by ρ_p. The integrated disturbance model above differs from integral controllers because the term we integrate in this formulation is model error and not distance from the set point. Therefore, the system composed of the state estimator, target calculation, and MHE regulator does not exhibit windup.

Once an estimate of the integrated disturbance has been obtained, new targets for the states and inputs must be calculated. In the target calculation phase, we seek steady-state values of the inputs and states for the system with an integrated disturbance model, assuming the estimated integrating error $\bar{p}_{j-N_e}$ remains constant in the future. Ideally, the steady-state values of the states $x_{t,j}$ are such that the required outputs, augmented by the disturbance, are at the output targets y_{set}. However, if it is impossible to find such states, we try to find states which come closest to reaching the output targets. Also, the states and steady-state inputs $u_{t,j}$ must satisfy the regulation constraints $Du_{t,j} \leq d$ and $Hx_{t,j} \leq h$. In mathematical form, the formulation we solve in the nonlinear target calculation is

$$\min_{x_{t,j}, u_{t,j}, \eta} \frac{1}{2}\eta^T \bar{Q}\eta + \bar{q}^T\eta + \frac{1}{2}\left(u_{t,j} - u_{t,j-1}\right)^T \bar{R}\left(u_{t,j} - u_{t,j-1}\right) \tag{4.49a}$$

subject to:

$$x_{t,j} = F(x_{t,j}, u_{t,j} + X_u p_j, 0, t_j) \tag{4.49b}$$

$$-\eta \leq g(x_{t,j}, t_j) + X_y p_j - y_{\text{set}} \leq \eta \tag{4.49c}$$

$$Du_{t,j} \leq d, \qquad Hx_{t,j} \leq h, \qquad \eta \geq 0. \tag{4.49d}$$

In the formulation of Equation 4.49a, we can make a few observations. First, it is not necessary to satisfy the steady-state constraint (Equation 4.49b) as written if using a continuous time model. In such a case, it does not make sense from a computational standpoint to keep a numerical integration inside an optimization routine unnecessarily. Instead, the constraint can be replaced by $f(x_{t,j}, u_{t,j}, 0, t_j) = 0$.

Secondly, we address the common case in which there are fewer output targets than there are output variables. In such cases, we can reduce the output constraint in Equation 4.49c to include only those rows that have output targets. The variable η is a slack variable that represents the relaxation of the output target constraint. Consequently, the relative importance of nearly satisfying the output target equality constraint is governed by the choice of the $\bar{Q}$ and $\bar{q}$ matrices.

Finally, the penalty on the steady-state inputs in Equation 4.49a is included to guarantee a unique solution to the target calculation problem. Here, $u_{t,j-1}$ is the former input target. The resulting formulation therefore distinguishes among the possible solutions by choosing the one that is closest to the prior input target.

4.3 Examples

We now present some simple examples to highlight the benefits of the moving horizon estimation approach.

Example 4.3.1 *Consider the simple mathematical example [14]*

$$\dot{x}_1 = k_1 x_1^2 x_2^5 - k_2 x_1^3 x_2^4$$
$$\dot{x}_2 = -k_1 x_1^2 x_2^5 + k_2 x_1^3 x_2^4$$

with output equation

$$y = x_1^4 x_2.$$

We choose Q and R to be the identity matrix and P to be the identity matrix multiplied by ten. The true initial state of the plant is $[0.3, 4.7]$, *but the* a priori *estimate is* $[19, 6]$.

The simulation results are presented in Figure 4.1. Note that the EKF converges to the wrong state estimate, even though the model and plant are identical. This example shows that although the EKF is commonly used in industry, its performance properties are not adequate for all nonlinear models.

Example 4.3.2 *Consider an isothermal gas-phase reactor in which the reversible reaction $2A \rightleftharpoons B$ is taking place. An initial amount of A and B are charged to the reactor, but the composition of the original mixture is not known accurately. A pressure gauge measures the total pressure of the system as the species react. The true initial state of the system and the* a priori *estimate are*

$$x_0 = \begin{bmatrix} P_A \; P_B \end{bmatrix}^T = \begin{bmatrix} 3 \; 1 \end{bmatrix}^T, \qquad \bar{x}_0 = \begin{bmatrix} 0.1 \; 4.5 \end{bmatrix}^T.$$

The system is modeled by

$$\dot{x}_1 = -2k_1 x_1^2 + 2k_2 x_2, \qquad \dot{x}_2 = k_1 x_1^2 - k_2 x_2 \tag{4.50}$$

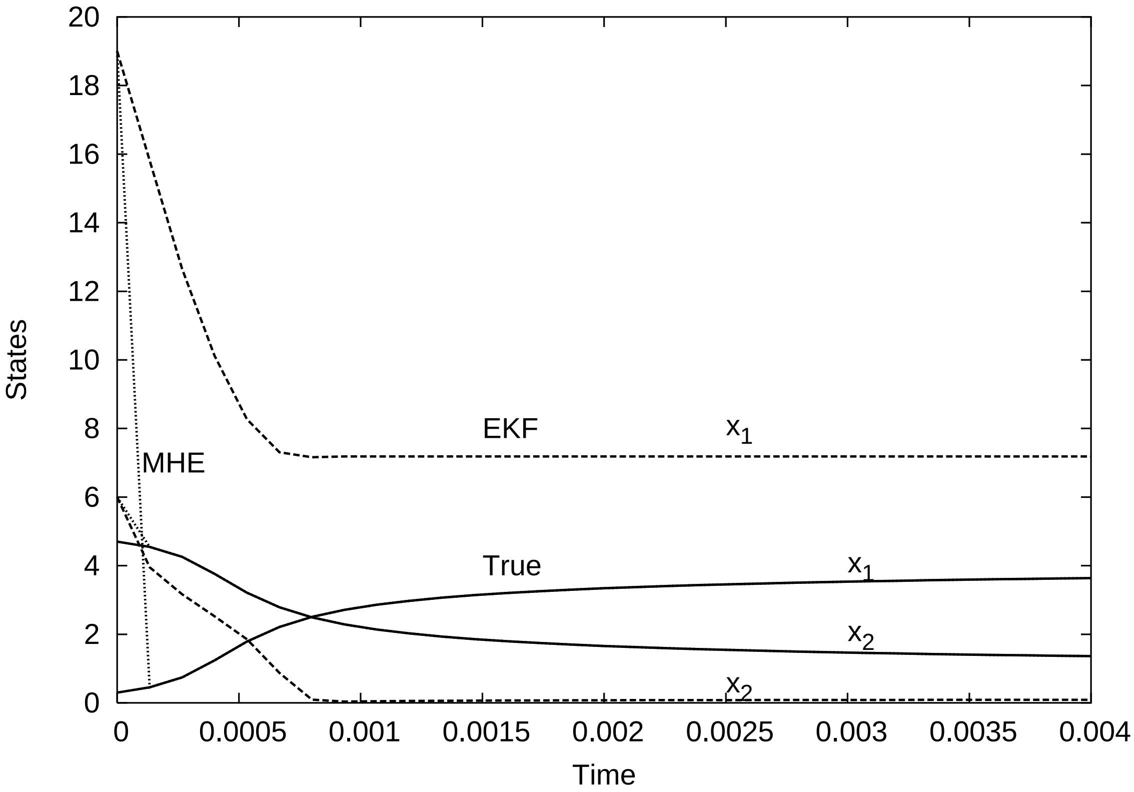

Figure 4.1: EKF vs. MHE performance on Example 4.3.1

in which $k_1 = 0.16\ min^{-1}\ atm^{-1}$, $k_2 = 0.0064\ min^{-1}$.

The results of the MHE estimator for a horizon length of ten is presented in Figure 4.2 for both the filter and smoothing updates. Note the periodic behavior of the estimates obtained using the filter update. The period is equal to the horizon length. This behavior is attributed to the propagation of estimation error from a poor *a priori* estimate. From a control standpoint, if a state estimate is inaccurate, then the regulatory action will most likely be inappropriate. Therefore, we advocate the use of the smoothing update for nonlinear control applications. The EKF fails for this example as well. In general, the EKF fails on systems with nonlinear measurements that have the property that more than one set of equilibrium states may satisfy the output equation.

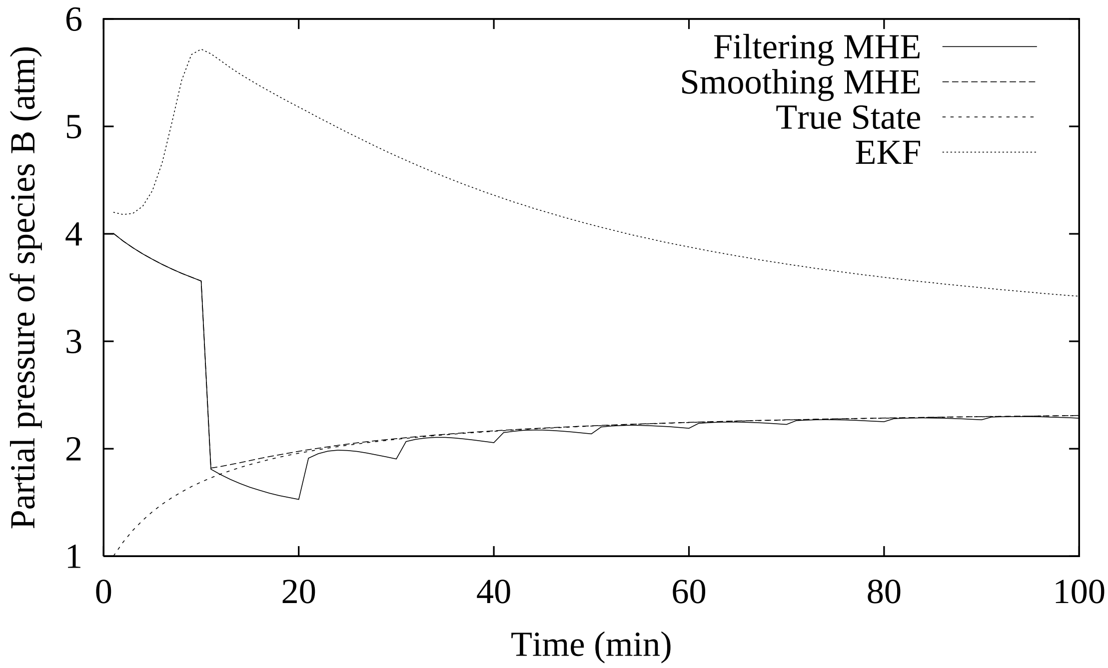

Figure 4.2: State estimates of gas reactor in Example 4.3.2

We now investigate some of the closed-loop performance properties of MHE on both linear and nonlinear models.

Example 4.3.3 *We investigate the process of MMA-VA copolymerization as described by Bindlish [6], which consists of a well-mixed reactor followed by a product separator. Several grades of polymer, characterized by the mole fraction of monomer A (MMA) in the copolymer product and the intrinsic viscosity of the copolymer product, are manufactured using this process. Depending on the demands of the market, the desired copolymer viscosity and composition are varied during operation. The feed to the reactor consists of the monomers (MMA and VA), initiator (AIBN), transfer agent (acetaldehyde) and inhibitor (m-dinitrobenzene) dissolved in a solvent (benzene). To remove heat released by the polymerization reaction, a coolant is employed. The polymer product is then separated from the unreacted hydrocarbons in a down-*

stream separator. The reactor and separator are represented by a physical model based on first principles. The model has 15 states, 3 inputs, and 7 output variables, of which only three have set-points. The sampling time for this process is 5 minutes and the prediction horizon is 20 time steps (100 minutes). We wish to switch from product grade A to grade B, with output set-points as shown in Table 4.1. For grade transitions, Bindlish notes that linear MPC is unstable. The measurements are subject to 5% noise and the a prioir *estimate of the state is incorrect by 5% for all the states.*

Figure 4.3 shows the output variables during the simulation for the nonlinear controller and moving horizon estimation, and Figure 4.4 shows the corresponding inputs. The quality of the state estimator is measured by the norm of the difference between the true states and the state estimates. As more data are collected, this value should decrease until it reaches a constant value. Figure 4.5 shows the evolution of this reconstruction error during the simulation of the copolymerization example. Each five minute sample time requires approximately 45 seconds for the combined estimation and regulation problems.

Grade	**Polymer production rate (kg/min)**	**MMA Mole fraction in copolymer**	**Copolymer viscosity ($10^{-6}m^3/kg$)**
A	0.3	0.60	35000
B	0.25	0.75	36725

Table 4.1: Product grades for MMA-VA copolymerization

4.4 Conclusions

In this chapter, we demonstrate that the moving horizon estimation problem is well-suited to reconstructing the states of nonlinear models from dynamic input and output data. Although widely used in industry, the extended Kalman filter fails to converge to the true states of the system on simple examples in which the initial state estimate is poor. In contrast, the moving horizon approach, through appropriate use of the nonlinear model, is able to converge to the correct states.

We present a derivation for an approximate nonlinear smoothing covariance update. We show that the while both the smoothing and filtering updates converge to the true state of nonlinear systems when the extended Kalman filter does not, the filtering update induces periodic behavior with a period length equal to the estimation horizon. We further present that the smoothing update avoids this undesirable feature.

Finally, the nonlinear moving horizon estimator with smoothing covariance update is applied to a closed-loop example with incorrect state estimate and sensor noise during a product grade transition. We present results of good control despite poor initial state estimates. We also show that the state estimates converge to the true states by examining the reconstruction error.

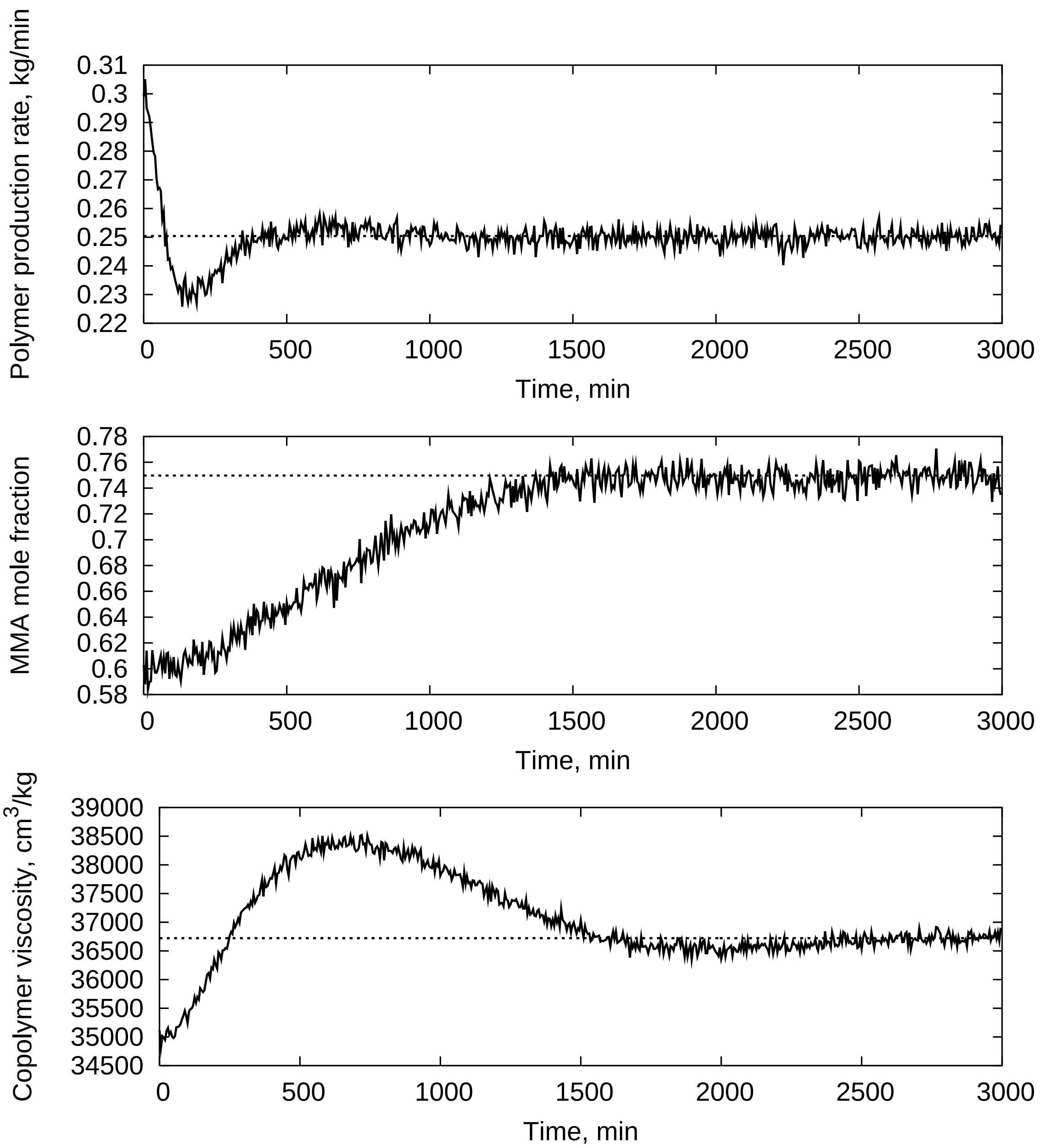

Figure 4.3: Measured polymer production rate in Example 4.3.3

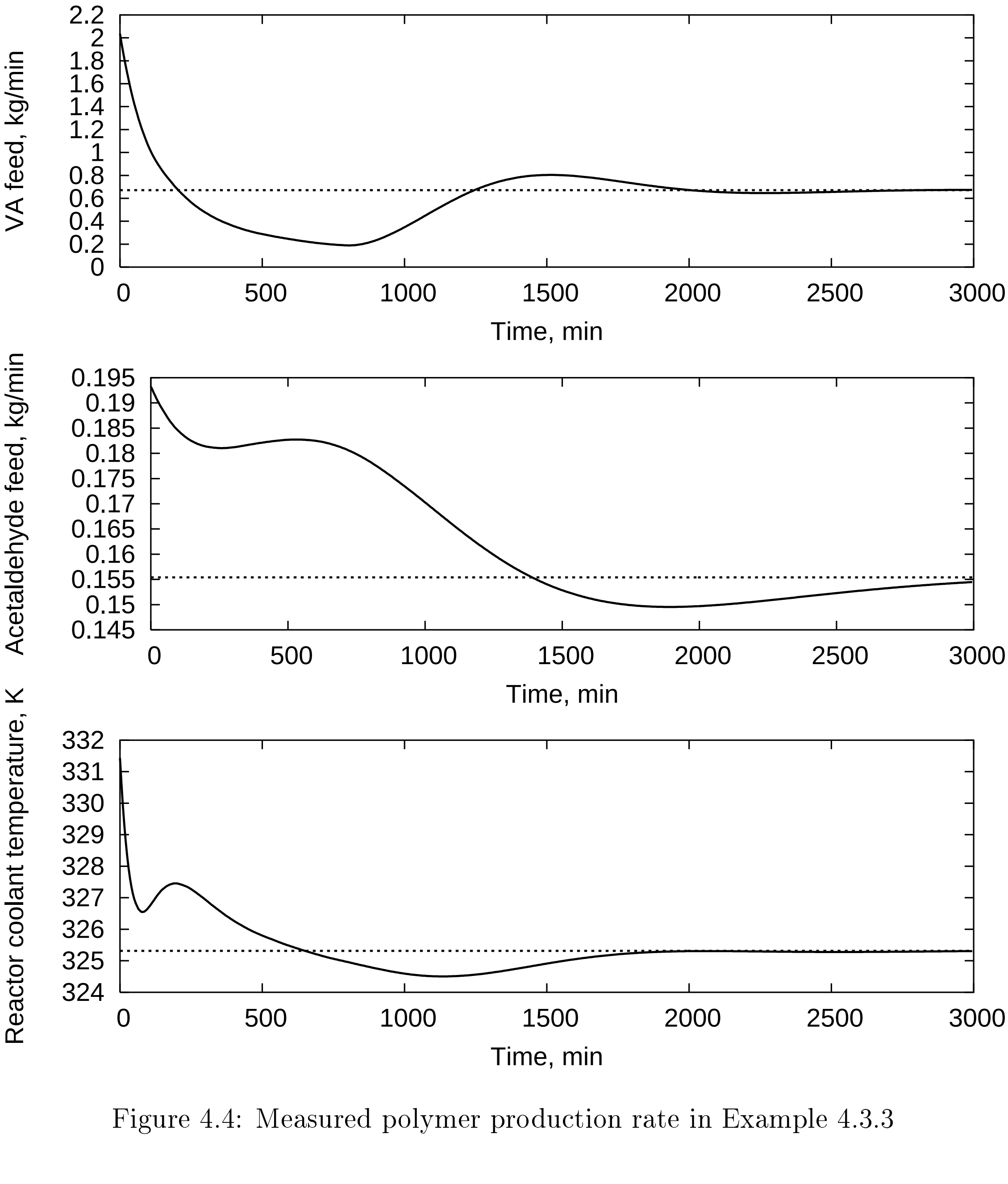

Figure 4.4: Measured polymer production rate in Example 4.3.3

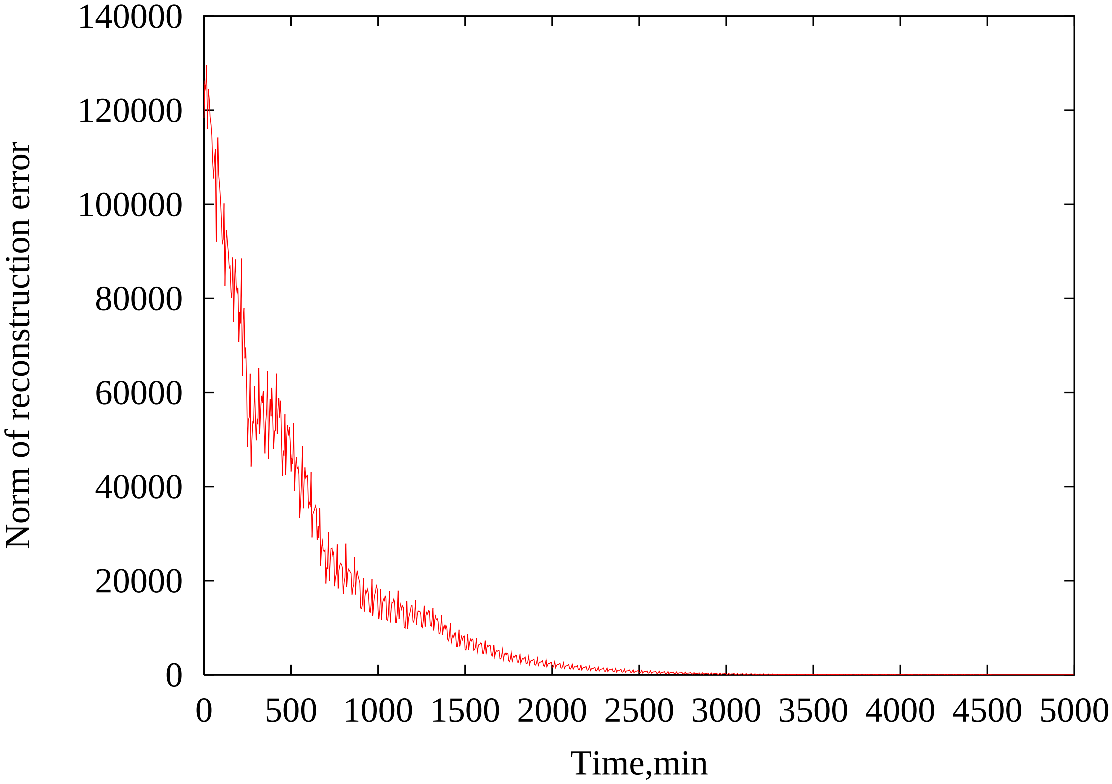

Figure 4.5: State reconstruction error for Example 4.3.3

Chapter 5

Structured Quadratic Programs for Linear, Time-Varying Systems

*Ah! I see you have the machine that goes *ping*!*

— Monty Python – The Meaning of Life

In linear model predictive control, as well as linear moving horizon state estimation, the optimal inputs or state estimates are achieved by solving quadratic programs (QPs). For nonlinear models, however, the optimization problems are no longer QPs, but are nonlinear programs (NLPs). To solve the NLPs, we employ a variant of a sequential quadratic programming (SQP) algorithm. The details of this algorithm are presented in Chapter 6. At every iteration of an SQP method, a quadratic program is formed by approximating the NLP as a QP about the current iterates. For nonlinear model predictive control and moving horizon estimation, these quadratic subproblems are equivalent to solving the respective problems for linear time-varying models. In this chapter, we present a method for solving these quadratic programs efficiently.

To solve the quadratic subproblem that arises in moving horizon estimation and model predictive control, we employ a primal-dual interior point method. The predictor corrector method used in this algorithm was first proposed by Mehrotra [69]. In this chapter, we present the details of a primal-dual interior point method for the solution of quadratic programs of the structure found in our specific applications. For more details on primal-dual interior point methods, see [126]. The basic framework for the development of this method is similar to the linear MPC regulator formulation found in [98], the highlight of which is the property that the QP solution time is a linear function of the prediction horizon length, rather than a cubic relationship.

5.1 Primal-Dual Interior-Point Methods

Consider the convex quadratic program

$$\min_{w} \Phi(w) = \frac{1}{2} w^T Q w + c^T w, \quad \text{s.t. } Fw = g,\, Cw \leq d \tag{5.1}$$

in which the matrix Q is positive semidefinite. The Karush-Kuhn-Tucker (KKT) conditions for optimality are that there exist Lagrange multipliers π and λ such that the following conditions hold:

$$\begin{gathered} Qw + F^T \pi C^T \lambda + c = 0 \\ -Fw + g = 0 \\ 0 \leq -Cw + d \perp \lambda \geq 0. \end{gathered}$$

The convex nature of the objective guarantees that these conditions are necessary and sufficient for optimality. It is convenient to replace the third condition with the equivalent, but easier to handle, system

$$\begin{bmatrix} Qw + F^T\pi + C^T\lambda + c \\ -Fw + g \\ -Cw + d - t \\ T\Lambda e \end{bmatrix} = 0, \quad (\lambda, t) \geq 0 \tag{5.2}$$

in which t is the vector of slack variables, Λ and T are matrices with λ and t on the diagonals, respectively, and e is a vector of ones.

Primal-dual interior-point methods solve the quadratic program by iterating from an initial guess to the optimal one. Using the notation in (5.2), the iterates are determined by solving the linear system

$$\begin{bmatrix} Q & F^T & C^T & \\ -F & & & \\ -C & & & -I \\ & & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta\pi \\ \Delta\lambda \\ \Delta t \end{bmatrix} = \begin{bmatrix} r_Q \\ r_F \\ r_C \\ r_t \end{bmatrix} \tag{5.3}$$

for specific choices of the right hand side variables. In the Mehrotra predictor-corrector algorithm, we solve (5.3) twice; the first phase is the predictor or affine-scaling step, while the second phase is the centering-corrector direction.

The first step for the solution of the QP is to first determine the duality gap

$$\mu = \frac{\lambda^T t}{m} \tag{5.4}$$

in which m is the number of rows in C. The duality gap is a measure of the feasiblity of the solution. As the algorithm proceeds, the duality gap converges to zero, as required by the fourth equation of (5.2). The second step is to solve (5.3) with the affine-scaling step right-hand side

$$\begin{bmatrix} r_Q \\ r_F \\ r_C \\ r_t \end{bmatrix} = \begin{bmatrix} -Qw - F^T\pi - C^T\lambda - c \\ Fw - g \\ Cw + t - d \\ -T\Lambda e \end{bmatrix}. \tag{5.5}$$

We denote the solution to the predictor step by $(\Delta w_{\text{aff}}, \Delta\pi_{\text{aff}}, \Delta\lambda_{\text{aff}}, \Delta t_{\text{aff}})$. The next step is to determine the maximum affine-scaling step length by

$$\alpha_{\text{aff}} = \arg\max\{\alpha \in [0,1] \,|\, (\lambda, t) + \alpha(\Delta\lambda_{\text{aff}}, \Delta t_{\text{aff}}) \geq 0\}. \tag{5.6}$$

The scalar α_{aff} determines the duality gap for the full step to the boundary

$$\mu_{\text{aff}} = \frac{(\lambda + \alpha_{\text{aff}}\Delta\lambda_{\text{aff}})^T(\lambda + \alpha_{\text{aff}}\Delta\lambda_{\text{aff}})}{m}. \tag{5.7}$$

Using this value, we define

$$\sigma = (\mu_{\text{aff}}/\mu)^3. \tag{5.8}$$

The next step is to solve for the corrector step. We determine $(\Delta w_{\text{cc}}, \Delta\pi_{\text{cc}}, \Delta\lambda_{\text{cc}}, \Delta t_{\text{cc}})$ by solving (5.3) with

$$\begin{bmatrix} r_Q \\ r_F \\ r_C \\ r_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\Delta T_{\text{aff}}\Delta\Lambda_{\text{aff}} e + \sigma\mu e \end{bmatrix}. \tag{5.9}$$

The overall search direction is then defined by

$$(\Delta w, \Delta\pi, \Delta\lambda, \Delta t) = (\Delta w_{\text{aff}}, \Delta\pi_{\text{aff}}, \Delta\lambda_{\text{aff}}, \Delta t_{\text{aff}}) + (\Delta w_{\text{cc}}, \Delta\pi_{\text{cc}}, \Delta\lambda_{\text{cc}}, \Delta t_{\text{cc}}). \tag{5.10}$$

To keep (λ, t) strictly nonnegative, we determine

$$\alpha_{\max} = \arg\max\{\alpha \in [0,1] \,|\, (\lambda, t) + \alpha(\Delta\lambda, \Delta t) \geq 0\} \tag{5.11}$$

and define the new iterate as

$$(w, \pi, \lambda, t)^{+} = (w, \pi, \lambda, t) + \alpha_{\max}\gamma(\Delta w, \Delta\pi, \Delta\lambda, \Delta t) \tag{5.12}$$

in which the heuristic factor γ is chosen to be .995. The process is repeated until convergence to within a specified tolerance is achieved.

In our algorithm, we perform block elimination on (5.3) and solve the more condensed system

$$\begin{bmatrix} Q & F^T & C^T \\ -F & & \\ -C & & \Lambda^{-1}T \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta\pi \\ \Delta\lambda \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} \hat{r}_Q \\ \hat{r}_F \\ \hat{r}_C \end{bmatrix}, \tag{5.13}$$

noting that

$$\Delta t = \Lambda^{-1}(r_t - T\Delta\lambda). \tag{5.14}$$

In practice, the linear programs in (5.6) and (5.11) are not solved by a linear programming method. Instead, we present the more convenient solution

$$\alpha = \min\left\{\frac{-1}{\min\{(\Delta\lambda, \Delta t)/(\lambda, t)\}}, 1\right\}, \tag{5.15}$$

which is valid unless all elements of $(\Delta\lambda, \Delta t)$ are nonnegative, in which case $\alpha = 1$.

5.2 Application to moving horizon estimation

We now focus on the solution of (5.13) for the specialized case of moving horizon state estimation. In this application, the quadratic program we wish to solve is of the form

$$\min_{x,w,v,\epsilon,\xi,\rho} \frac{1}{2}\rho^T P\rho + p^T\rho + \sum_{k=0}^{N-1}\left(\frac{1}{2}w_k^T Q_k w_k + q_k^T w_k + \frac{1}{2}v_k^T R_k v_k + r_k^T v_k\right) +$$
$$\sum_{k=0}^{N-1}\left(\frac{1}{2}x_k^T \Theta_k x_k + \theta_k^T x_k + x_k^T L_k w_k + \frac{1}{2}\epsilon_k^T Z_k \epsilon_k + z_k^T \epsilon_k + \frac{1}{2}\xi_k^T \Psi_k \xi_k + \psi_k^T \xi_k\right) +$$
$$\frac{1}{2}v_N^T R_N v_N + r_N^T v_N + \frac{1}{2}\epsilon_N^T Z_N \epsilon_N + z_N^T \epsilon_N + \frac{1}{2}\xi_N^T \Psi_N \xi_N + \psi_N^T \xi_N \tag{5.16a}$$

subject to:

$$x_0 = \bar{x}_0 + \rho \tag{5.16b}$$

$$x_{k+1} = A_k x_k + G_k w_k + f_k \tag{5.16c}$$

$$0 = C_k x_k + v_k + d_k \tag{5.16d}$$

$$H_k x_k - \epsilon_k \leq h_k \tag{5.16e}$$

$$\epsilon_k \geq \bar{\epsilon}_k \tag{5.16f}$$

$$\Gamma_k v_k - \xi_k \leq \gamma_k \tag{5.16g}$$

$$\xi_k \geq \bar{\xi}_k \tag{5.16h}$$

$$S_k w_k + F_k x_k \leq s_k \tag{5.16i}$$

$$S_\rho \rho \leq s_\rho. \tag{5.16j}$$

One of the keys to efficiently solving (5.13) is to order the variables in w in a special order. To arrive at the particular blockwise diagonal structure desired, we introduce the following definition of w:

$$w^T = \left[\rho^T \ \epsilon_0^T \ x_0^T \ v_0^T \ \xi_0^T \ w_0^T \ \epsilon_1^T \ x_1^T \ v_1^T \ \xi_1^T \cdots\right]. \tag{5.17}$$

The variables are arranged stagewise, which is an important point. Since the moving horizon problem does not have any terms, with the exception of the state equation, that rely on more than one stage, the resulting matrix in (5.13) is sparse. More importantly, however, is that by a combination of proper variable ordering and rearrangement of the order of the equations to be solved, the overall matrix is overlapping block diagonal. This structure enables the efficient solution of the primal-dual interior-point step.

The Lagrange multipliers π for the equality constraints are denoted as π_ρ^p for (5.16b), π_k^p for (5.16c), and π_k^q for (5.16d). The multipliers for the inequalities λ are denoted as λ_ρ^S for (5.16j), λ_k^S for (5.16i), λ_k^H for (5.16e), λ_k^I for (5.16f), λ_k^Γ for (5.16g), and λ_k^J for (5.16h).

The first step to solving (5.13) is to eliminate by substitution the soft constraint penalties ϵ_k and ξ_k and the inequality Lagrange multipliers. To do so, we introduce the notation

$$\Lambda T^{-1} = \Sigma^{-1}. \tag{5.18}$$

The matrix T is invertible since it is square and nonsingular; the slack variables and Lagrange multipliers in an interior point method are always strictly positive, as enforced by (5.6) and (5.11). The values for t and λ are only exact in the limit towards infinite iterations.

Solving for $\Delta\epsilon_k$ yields

$$\Delta\epsilon_k = \widetilde{Z}_k^{-1}({\Sigma_k^H}^{-1} H_k \Delta x_k + \hat{r}_k^\epsilon + {\Sigma_k^H}^{-1}\hat{r}_k^H + {\Sigma_k^I}^{-1}\hat{r}_k^I) \tag{5.19}$$

in which

$$\widetilde{Z}_k = Z_k + {\Sigma_k^H}^{-1} + {\Sigma_k^I}^{-1}. \tag{5.20}$$

Similarly,

$$\Delta\xi_k = \widetilde{\Psi}_k^{-1}({\Sigma_k^\Gamma}^{-1}\Gamma_k \Delta v_k + \hat{r}_k^\xi + {\Sigma_k^\Gamma}^{-1}\hat{r}_k^\Gamma + {\Sigma_k^J}^{-1}\hat{r}_k^J) \tag{5.21}$$

in which

$$\widetilde{\Psi}_k = \Psi_k + {\Sigma_k^\Gamma}^{-1} + {\Sigma_k^J}^{-1}. \tag{5.22}$$

For the development of the algorithm, we present the following definitions:

$$\overline{P} = P + S_\rho^T {\Sigma_\rho^S}^{-1} S_\rho \tag{5.23a}$$

$$\overline{M}_k = \Theta_k + H_k^T {\Sigma_k^H}^{-1} H_k + F_k^T {\Sigma_k^S}^{-1} F_k - H_k^T {\Sigma_k^H}^{-1} \widetilde{Z}_k {\Sigma_k^H}^{-1} H_k \tag{5.23b}$$

$$\overline{L}_k = L_k + F_k^T {\Sigma_k^S}^{-1} S_k \tag{5.23c}$$

$$\overline{R}_k = R_k + \Gamma_k^T {\Sigma_k^\Gamma}^{-1} \Gamma_k - \Gamma_k^T {\Sigma_k^\Gamma}^{-1} \widetilde{\Psi}_k {\Sigma_k^\Gamma}^{-1} \Gamma_k \tag{5.23d}$$

$$\overline{Q}_k = Q_k + S_k^T {\Sigma_k^S}^{-1} S_k \tag{5.23e}$$

and for the special case of $k = N$,

$$\overline{M}_N = H_N^T {\Sigma_N^H}^{-1} H_N - H_N^T {\Sigma_N^H}^{-1} \widetilde{Z}_N {\Sigma_N^H}^{-1} H_N. \tag{5.24}$$

The right-hand sides for the condensed problem are

$$\tilde{r}_k^\epsilon = \hat{r}_k^\epsilon + {\Sigma_k^H}^{-1} \hat{r}_k^H + {\Sigma_k^I}^{-1} \hat{r}_k^I \tag{5.25a}$$

$$\tilde{r}_k^\xi = \hat{r}_k^\xi + {\Sigma_k^\Gamma}^{-1} \hat{r}_k^\Gamma + {\Sigma_k^J}^{-1} \hat{r}_k^J \tag{5.25b}$$

$$\tilde{r}_k^p = -\hat{r}_k^p \tag{5.25c}$$

$$\tilde{r}_k^q = -\hat{r}_k^q \tag{5.25d}$$

$$\tilde{r}^\rho = \hat{r}^\rho - S_\rho^T {\Sigma_\rho^S}^{-1} \hat{r}_\rho^S \tag{5.25e}$$

$$\tilde{r}_k^x = \hat{r}_k^x - H_k^T {\Sigma_k^H}^{-1} \hat{r}_k^H - F_k^T {\Sigma_k^S}^{-1} \hat{r}_k^S + H_k^T {\Sigma_k^H}^{-1} \widetilde{Z}_k^{-1} \tilde{r}_k^\epsilon \tag{5.25f}$$

$$\tilde{r}_k^v = \hat{r}_k^v - \Gamma_k^T {\Sigma_k^\Gamma}^{-1} \hat{r}_k^\Gamma + \Gamma_k^T {\Sigma_k^\Gamma}^{-1} \widetilde{\Psi}_k^{-1} \tilde{r}_k^\xi \tag{5.25g}$$

$$\tilde{r}_k^w = \hat{r}_k^w - S_k^T {\Sigma_k^S}^{-1} \hat{r}_k^S \tag{5.25h}$$

and for $k = N$,

$$\tilde{r}_N^x = \hat{r}_N^x - H_N^T {\Sigma_N^H}^{-1} \hat{r}_N^H + H_N^T {\Sigma_N^H}^{-1} \widetilde{Z}_N^{-1} \tilde{r}_N^\epsilon. \tag{5.26}$$

The resulting system to be solved is the positive semidefinite symmetric system

$$\begin{bmatrix} \overline{P} & I & & & & & & & \\ I & & -I & & & & & & \\ & -I & \overline{M}_0 & & C_0^T & \overline{L}_0 & A_0^T & & \\ & & & \overline{R}_0 & I & & & & \\ & & C_0 & I & & & & & \\ & & \overline{L}_0^T & & & \overline{Q}_0 & G_0^T & & \\ & & A_0 & & & G_0 & & -I & \\ & & & & & & -I & \overline{M}_1 & \\ & & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \Delta\rho \\ \Delta\pi_\rho^p \\ \Delta x_0 \\ \Delta v_0 \\ \Delta\pi_0^q \\ \Delta w_0 \\ \Delta\pi_0^p \\ \Delta x_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \tilde{r}^\rho \\ \tilde{r}_\rho^p \\ \tilde{r}_0^x \\ \tilde{r}_0^v \\ \tilde{r}_0^q \\ \tilde{r}_0^w \\ \tilde{r}_0^p \\ \tilde{r}_1^x \\ \vdots \end{bmatrix} \tag{5.27}$$

in which the blocks from $\tilde{r}^x$ to $\tilde{r}^p$ repeat along the diagonal of the matrix until the last stage. Here,

$$\begin{bmatrix} \ddots & & & & & & & & \\ -I & \overline{M}_{N-1} & & C_{N-1}^T & \overline{L}_{N-1} & A_{N-1}^T & & & \\ & & \overline{R}_{N-1} & I & & & & & \\ & C_{N-1} & I & & & & & & \\ & \overline{L}_{N-1}^T & & & \overline{Q}_{N-1} & G_{N-1}^T & & & \\ & A_{N-1} & & & G_{N-1} & & -I & & \\ & & & & & -I & \overline{M}_N & & C_N^T \\ & & & & & & & \overline{R}_N & I \\ & & & & & & C_N & I & \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta\pi_{N-2}^p \\ \Delta x_{N-1} \\ \Delta v_{N-1} \\ \Delta\pi_{N-1}^q \\ \Delta w_{N-1} \\ \Delta\pi_{N-1}^p \\ \Delta x_N \\ \Delta v_N \\ \Delta\pi_N^q \end{bmatrix} = \begin{bmatrix} \vdots \\ \tilde{r}_{N-2}^p \\ \tilde{r}_{N-1}^x \\ \tilde{r}_{N-1}^v \\ \tilde{r}_{N-1}^q \\ \tilde{r}_{N-1}^w \\ \tilde{r}_{N-1}^p \\ \tilde{r}_N^x \\ \tilde{r}_N^v \\ \tilde{r}_N^q \end{bmatrix} \tag{5.28}$$

We solve this system by starting at the last stage and working backwards. At stage

N,

$$\begin{bmatrix} -I & \overline{M}_N & & C_N^T \\ & \overline{R}_N & I & \\ C_N & I & & \end{bmatrix} \begin{bmatrix} \Delta\pi_{N-1}^p \\ \Delta x_N \\ \Delta v_N \\ \Delta\pi_N^q \end{bmatrix} = \begin{bmatrix} \tilde{r}_{N-1}^p \\ \tilde{r}_N^x \\ \tilde{r}_N^v \\ \tilde{r}_N^q \end{bmatrix}, \tag{5.29}$$

which can be reduced to

$$\Pi_{N-1}\Delta x_N - \Delta\pi_{N-1}^p = \pi_{N-1}. \tag{5.30}$$

in which

$$\Pi_{N-1} = \overline{M}_N + C_N^T \overline{R}_N C_N \tag{5.31a}$$

$$\pi_{N-1} = \tilde{r}_N^x - C_N^T \tilde{r}_N^v + C_N^T \overline{R}_N \tilde{r}_N^q. \tag{5.31b}$$

Now examine the system

$$\begin{bmatrix} -I & \overline{M}_k & & C_k^T & \overline{L}_k & A_k^T & \\ & \overline{R}_k & I & & & & \\ C_k & I & & & & & \\ \overline{L}_k^T & & & & \overline{Q}_k & G_k^T & \\ A_k & & & & G_k & & -I \\ & & & & & -I & \Pi_k \end{bmatrix} \begin{bmatrix} \Delta\pi_{k-1}^p \\ \Delta x_k \\ \Delta v_k \\ \Delta\pi_k^q \\ \Delta w_k \\ \Delta\pi_k^p \\ \Delta x_{k+1} \end{bmatrix} = \begin{bmatrix} \tilde{r}_{k-1}^p \\ \tilde{r}_k^x \\ \tilde{r}_k^v \\ \tilde{r}_k^q \\ \tilde{r}_k^w \\ \tilde{r}_k^p \\ \pi_k \end{bmatrix}. \tag{5.32}$$

This system is condensed in much the same way, yielding

$$\begin{aligned} \Pi_{k-1} = & \overline{M}_k + C_k^T \overline{R}_k C_k + A_k^T \Pi_k A_k - \\ & (\overline{L}_k + A_k^T \Pi_k G_k)(\overline{Q}_k + G_k^T \Pi_k G_k)^{-1}(\overline{L}_k + A_k^T \Pi_k G_k)^T \end{aligned} \tag{5.33a}$$

$$\begin{aligned} \pi_{k-1} = & \tilde{r}_k^x - (\overline{L}_k + A_k^T \Pi_k G_k)(\overline{Q}_k + G_k^T \Pi_k G_k)^{-1}(\tilde{r}_k^w + G_k^T \Pi_k \tilde{r}_k^p + G_k \pi_k) - \\ & C_k^T(\tilde{r}_k^v - \overline{R}_k \tilde{r}_k^q) + A_k^T \Pi_k \tilde{r}_k^p + A_k^T \pi_k. \end{aligned} \tag{5.33b}$$

Therefore, the system can be solved by recursion on the stages, yielding a solution method that scales linearly with the number of stages. The iteration continues until the first stage is reached; then the system looks like

$$\begin{bmatrix} \overline{P} & I & \\ I & & -I \\ & -I & \Pi_0 \end{bmatrix} \begin{bmatrix} \Delta\rho \\ \Delta\pi_\rho^p \\ \Delta x_0 \end{bmatrix} = \begin{bmatrix} \tilde{r}^\rho \\ \tilde{r}_\rho^p \\ \tilde{r}_0^x \end{bmatrix}, \tag{5.34}$$

yielding

$$\Delta\rho = (\overline{P} + \Pi_0)^{-1}(\tilde{r}^\rho + \pi_0 + \Pi_0 \tilde{r}_\rho^p) \tag{5.35a}$$

$$\Delta x_0 = \Delta\rho - \tilde{r}_\rho^p \tag{5.35b}$$

$$\Delta\pi_\rho^p = \Pi_0 \Delta x_0 - \pi_0 \tag{5.35c}$$

The back-substitution continues stagewise down the matrix:

$$\Delta v_k = -C_k \Delta x_k + \tilde{r}_k^q \tag{5.36a}$$

$$\Delta\pi_k^q = -\overline{R}_k \Delta v_k + \tilde{r}_k^v \tag{5.36b}$$

$$\Delta w_k = (\overline{Q}_k + G_k^T \Pi_k G_k)^{-1}(\tilde{r}_k^w - (\overline{L}_k^T + G_k^T \Pi_k A_k)\Delta x_k + G_k^T(\pi_k + \Pi_k \tilde{r}_k^p)) \tag{5.36c}$$

$$\Delta x_{k+1} = A_k \Delta x_k + G_k \Delta w_k - \tilde{r}_k^p \tag{5.36d}$$

$$\Delta\pi_k^p = \Pi_k \Delta x_{k+1} - \pi_k \tag{5.36e}$$

To find the step for the inequality Lagrange multipliers and the soft constraint variables, use the formulas

$$\Delta\epsilon_k = \widetilde{Z}_k^{-1}(\tilde{r}_k^{\epsilon} + \Sigma_k^{H^{-1}} H_k \Delta x_k) \tag{5.37a}$$

$$\Delta\xi_k = \widetilde{\Psi}_k^{-1}(\tilde{r}_k^{\xi} + \Sigma_k^{\Gamma^{-1}} \Gamma_k \Delta v_k) \tag{5.37b}$$

$$\Delta\lambda_\rho^S = \Sigma_\rho^{S^{-1}}(\hat{r}_\rho^S + S_\rho \Delta\rho) \tag{5.37c}$$

$$\Delta\lambda_k^H = \Sigma_k^{H^{-1}}(\hat{r}_k^H + H_k \Delta x_k - \Delta\epsilon_k) \tag{5.37d}$$

$$\Delta\lambda_k^I = \Sigma_k^{I^{-1}}(\hat{r}_k^I - \Delta\epsilon_k) \tag{5.37e}$$

$$\Delta\lambda_k^\Gamma = \Sigma_k^{\Gamma^{-1}}(\hat{r}_k^\Gamma + \Gamma_k \Delta v_k - \Delta\xi_k) \tag{5.37f}$$

$$\Delta\lambda_k^J = \Sigma_k^{J^{-1}}(\hat{r}_k^J - \Delta\xi_k) \tag{5.37g}$$

$$\Delta\lambda_k^S = \Sigma_k^{S^{-1}}(\hat{r}_k^S + F_k \Delta x_k + S_k \Delta w_k). \tag{5.37h}$$

Finally, the slack variable step Δt can be recovered easily through (5.14).

The method presented for solving for the step in the primal-dual interior point method is applicable to both the predictor and corrector steps, since the right hand side variables $\hat{r}$ are not defined in the above arguments *a priori*. We now present the values for these vectors for both steps.

For the affine-scaling step, the elements of $\hat{r}_Q$ in (5.13) are

$$\hat{r}^\rho = -(P\rho + \pi_\rho^p + S_\rho^T \lambda_\rho^S + p) \tag{5.38a}$$

$$\hat{r}_k^\epsilon = -(Z_k \epsilon_k - \lambda_k^H - \lambda_k^I + z_k) \tag{5.38b}$$

$$\hat{r}_k^x = -(\Theta_k x_k + L_k w_k - \pi_{k-1}^p + C_k^T \pi_k^q + A_k^T \pi_k^p + H_k^T \lambda_k^H + F_k^T \lambda_k^S + \theta_k) \tag{5.38c}$$

$$\hat{r}_k^v = -(R_k v_k + \pi_k^q + \Gamma_k^T \lambda_k^\Gamma + r_k) \tag{5.38d}$$

$$\hat{r}_k^\xi = -(\Psi_k \xi_k - \lambda_k^\Gamma - \lambda_k^J + \psi_k) \tag{5.38e}$$

$$\hat{r}_k^w = -(L_k^T x_k + Q_k w_k + G_k^T \pi_k^p + S_k^T \lambda_k^S + q_k) \tag{5.38f}$$

except for k=N, when

$$\hat{r}_N^x = -(-\pi_{N-1}^p + C_N^T \pi_N^q + H_N^T \lambda_N^H). \tag{5.39}$$

The vectors in $\hat{r}_F$ are defined by

$$\hat{r}_\rho^p = \rho - x_0 - \bar{x}_0 \tag{5.40a}$$

$$\hat{r}_k^q = C_k x_k + v_k + d_k \tag{5.40b}$$

$$\hat{r}_k^p = A_k x_k + G_k w_k - x_{k+1} - f_k \tag{5.40c}$$

and the elements of $\hat{r}_C$ are

$$\hat{r}_\rho^S = S_\rho \rho - s_\rho \tag{5.41a}$$

$$\hat{r}_k^H = H_k x_k - \epsilon_k - h_k \tag{5.41b}$$

$$\hat{r}_k^I = -\epsilon_k + \bar{\epsilon}_k \tag{5.41c}$$

$$\hat{r}_k^\Gamma = \Gamma_k v_k - \xi_k - \gamma_k \tag{5.41d}$$

$$\hat{r}_k^J = -\xi_k + \bar{\xi}_k \tag{5.41e}$$

$$\hat{r}_k^S = S_k w_k + F_k x_k - s_k. \tag{5.41f}$$

For the case of the center-correcting step, both $\hat{r}_Q$ and $\hat{r}_F$ are zero. Therefore, only $\hat{r}_C$ is presented:

$$\hat{r}_\rho^S = -\Lambda_\rho^{S^{-1}} \left[(\Delta T_\rho^S)_{\text{aff}} (\Delta \lambda_\rho^S)_{\text{aff}} + \sigma \mu e \right] \tag{5.42a}$$

$$\hat{r}_k^H = -\Lambda_k^{H^{-1}} \left[(\Delta T_k^H)_{\text{aff}} (\Delta \lambda_k^H)_{\text{aff}} + \sigma \mu e \right] \tag{5.42b}$$

$$\hat{r}_k^I = -\Lambda_k^{I^{-1}} \left[(\Delta T_k^I)_{\text{aff}} (\Delta \lambda_k^I)_{\text{aff}} + \sigma \mu e \right] \tag{5.42c}$$

$$\hat{r}_k^\Gamma = -\Lambda_k^{\Gamma^{-1}} \left[(\Delta T_k^\Gamma)_{\text{aff}} (\Delta \lambda_k^\Gamma)_{\text{aff}} + \sigma \mu e \right] \tag{5.42d}$$

$$\hat{r}_k^J = -\Lambda_k^{J^{-1}} \left[(\Delta T_k^J)_{\text{aff}} (\Delta \lambda_k^J)_{\text{aff}} + \sigma \mu e \right] \tag{5.42e}$$

$$\hat{r}_k^S = -\Lambda_k^{S^{-1}} \left[(\Delta T_k^S)_{\text{aff}} (\Delta \lambda_k^S)_{\text{aff}} + \sigma \mu e \right]. \tag{5.42f}$$

5.3 Application to the MPC regulator

The quadratic subproblem solved for the nonlinear model predictive control is developed in a similar way to the estimation problem discussed in the previous section. We begin by presenting the structure of the quadratic program solved in the regulator

$$\min_{x,u,\epsilon} \sum_{k=0}^{N-1} \left(\frac{1}{2} x_k^T Q_k x_k + q_k^T x_k + \frac{1}{2} u_k^T R_k u_k + r_k^T u_k + x_k^T M_k u_k + \frac{1}{2} \epsilon_k^T Z_k \epsilon_k + z_k^T \epsilon_k \right) +$$
$$\frac{1}{2} x_N^T Q_N x_N + q_N^T x_N + \epsilon_N^T Z_N \epsilon_N + z_N^T \epsilon_N$$

subject to:

$$x_{k+1} = A_k x_k + B_k u_k + f_k \tag{5.43a}$$

$$D_k u_k - G_k x_k \leq d_k \tag{5.43b}$$

$$H_k x_k - \epsilon_k \leq h_k \tag{5.43c}$$

$$\epsilon_k \geq \bar{\epsilon}_k \tag{5.43d}$$

in which x_0 is constant. The terminal penalty is represented in this formulation by Q_N and q_N.

As in the estimator, we order w in (5.13) so that the resulting system has a particular blockwise diagonal structure. In the regulator subproblem, we define w as

$$w^T = \begin{bmatrix} u_0^T \ \epsilon_1^T \ x_1^T \ u_1^T \ \epsilon_2^T \ x_2^T \cdots \end{bmatrix}. \tag{5.44}$$

The variables are grouped according to prediction stage, so the resulting matrix in (5.13) is sparse. As in the estimator subproblem, the equations are rearranged such that the overall matrix is overlapping block diagonal. This structure enables the efficient solution of the primal-dual interior-point step.

The Lagrange multipliers π for the equality constraints are denoted as π_k^p for (5.43a). The multipliers for the inequalities λ are denoted as λ_k^D for (5.43b), λ_k^H for

(5.43c), and λ_k^I for (5.43d).

We begin by eliminating the soft constraint penalties ϵ_k and the inequality Lagrange multipliers by substitution. To do so, recall the notation from the previous section:

$$\Lambda T^{-1} = \Sigma^{-1}. \tag{5.45}$$

Solving for $\Delta\epsilon_k$ yields

$$\Delta\epsilon_k = \widetilde{Z}_k^{-1}({\Sigma_k^H}^{-1} H_k \Delta x_k + \hat{r}_k^{\epsilon} + {\Sigma_k^H}^{-1} \hat{r}_k^H + {\Sigma_k^I}^{-1} \hat{r}_k^I) \tag{5.46}$$

in which

$$\widetilde{Z}_k = Z_k + {\Sigma_k^H}^{-1} + {\Sigma_k^I}^{-1}. \tag{5.47}$$

For the development of the algorithm, we present the following definitions:

$$\overline{R}_k = R_k + D_k^T {\Sigma_k^D}^{-1} D_k \tag{5.48a}$$

$$\overline{M}_k = M_k - G_k^T {\Sigma_k^D}^{-1} D_k \tag{5.48b}$$

$$\overline{Q}_k = Q_k + G_k^T {\Sigma_k^D}^{-1} G_k + H_k^T {\Sigma_k^H}^{-1} H_k - H_k^T {\Sigma_k^H}^{-1} \widetilde{Z}_k {\Sigma_k^H}^{-1} H_k \tag{5.48c}$$

and for the special case of $k = N$,

$$\overline{Q}_N = Q_N + H_N^T {\Sigma_N^H}^{-1} H_N - H_N^T {\Sigma_N^H}^{-1} \widetilde{Z}_N {\Sigma_N^H}^{-1} H_N. \tag{5.49}$$

The right-hand sides for the condensed regulator problem are

$$\tilde{r}_k^{\epsilon} = \hat{r}_k^{\epsilon} + {\Sigma_k^H}^{-1} \hat{r}_k^H + {\Sigma_k^I}^{-1} \hat{r}_k^I \tag{5.50a}$$

$$\tilde{r}_k^p = -\hat{r}_k^p \tag{5.50b}$$

$$\tilde{r}_k^x = \hat{r}_k^x - H_k^T {\Sigma_k^H}^{-1} \hat{r}_k^H + G_k^T {\Sigma_k^D}^{-1} \hat{r}_k^D + H_k^T {\Sigma_k^H}^{-1} \widetilde{Z}_k^{-1} \tilde{r}_k^{\epsilon} \tag{5.50c}$$

$$\tilde{r}_k^u = \hat{r}_k^u - D_k^T {\Sigma_k^D}^{-1} \hat{r}_k^D \tag{5.50d}$$

and for $k = N$,

$$\tilde{r}_N^x = \hat{r}_N^x - H_N^T \Sigma_N^{H^{-1}} \hat{r}_N^H + H_N^T \Sigma_N^{H^{-1}} \widetilde{Z}_N^{-1} \tilde{r}_N^{\epsilon}. \tag{5.51}$$

The resulting system to be solved is the positive semidefinite symmetric system

$$\begin{bmatrix} \overline{R}_0 & B_0^T & & & & & & & & \\ B_0 & & -I & & & & & & & \\ & -I & \overline{Q}_1 & \overline{M}_1 & A_1^T & & & & & \\ & & \overline{M}_1^T & \overline{R}_1 & B_1^T & & & & & \\ & & A_1 & B_1 & & -I & & & & \\ & & & & -I & \overline{Q}_2 & \overline{M}_2 & A_2^T & & \\ & & & & & \overline{M}_2^T & \overline{R}_2 & B_2^T & & \\ & & & & & A_2^T & B_2^T & \ddots & & \\ & & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \Delta u_0 \\ \Delta \pi_0^p \\ \Delta x_1 \\ \Delta u_1 \\ \Delta \pi_1^p \\ \Delta x_2 \\ \Delta u_2 \\ \Delta \pi_2^p \\ \vdots \end{bmatrix} = \begin{bmatrix} \tilde{r}_0^u \\ \tilde{r}_0^p \\ \tilde{r}_1^x \\ \tilde{r}_1^u \\ \tilde{r}_1^p \\ \tilde{r}_2^x \\ \tilde{r}_2^u \\ \tilde{r}_2^p \\ \vdots \end{bmatrix} \tag{5.52}$$

in which the blocks from $\tilde{r}^x$ to $\tilde{r}^p$ repeat along the diagonal of the matrix until the last stage. Here,

$$\begin{bmatrix} \ddots & & & & & \\ -I & \overline{Q}_{N-1} & \overline{M}_{N-1} & A_{N-1}^T & & \\ & \overline{M}_{N-1}^T & \overline{R}_{N-1} & B_{N-1}^T & & \\ & A_{N-1} & B_{N-1} & & -I & \\ & & & -I & \overline{Q}_N \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta \pi_{N-2}^p \\ \Delta x_{N-1} \\ \Delta u_{N-1} \\ \Delta \pi_{N-1}^p \\ \Delta x_N \end{bmatrix} = \begin{bmatrix} \vdots \\ \tilde{r}_{N-2}^p \\ \tilde{r}_{N-1}^x \\ \tilde{r}_{N-1}^u \\ \tilde{r}_{N-1}^p \\ \tilde{r}_N^x \end{bmatrix} \tag{5.53}$$

We solve this system by starting at the last stage and working backwards. First define

$$\Pi_N = Q_N, \qquad \pi_N = \tilde{r}_N^x \tag{5.54}$$

Therefore, the last row of (5.53) is

$$\Pi_N \Delta x_N - \Delta \pi_{N-1}^p = \pi_N. \tag{5.55}$$

Now examine the system

$$\begin{bmatrix} \ddots & & & & & \\ -I & \overline{Q}_k & \overline{M}_k & A_k^T & & \\ & \overline{M}_k^T & \overline{R}_k & B_k^T & & \\ & A_k & B_k & & -I & \\ & & & -I & \Pi_{k+1} & \end{bmatrix} \begin{bmatrix} \vdots \\ \Delta\pi_{k-1}^p \\ \Delta x_k \\ \Delta u_k \\ \Delta\pi_k^p \\ \Delta x_{k+1} \end{bmatrix} = \begin{bmatrix} \vdots \\ \tilde{r}_{k-1}^p \\ \tilde{r}_k^x \\ \tilde{r}_k^u \\ \tilde{r}_k^p \\ \pi_{k+1} \end{bmatrix}, \tag{5.56}$$

which is equivalent to the final stage when $k = N - 1$. We condense this system in a similar way to the substitution shown for the estimation problem. The resulting recursive formulas are

$$\begin{aligned} \Pi_k =& \overline{Q}_k + A_k^T \Pi_{k+1} A_k - \\ & (\overline{M}_k + A_k^T \Pi_{k+1} G_k)(\overline{R}_k + B_k^T \Pi_{k+1} B_k)^{-1} (\overline{M}_k + A_k^T \Pi_{k+1} B_k)^T \end{aligned} \tag{5.57a}$$

$$\begin{aligned} \pi_k =& \tilde{r}_k^x + A_k^T \Pi_{k+1} \tilde{r}_k^p + A_k^T \pi_{k+1} - \\ & (\overline{M}_k + A_k^T \Pi_{k+1} B_k)(\overline{R}_k + B_k^T \Pi_{k+1} B_k)^{-1} (\tilde{r}_k^u + B_k^T \Pi_{k+1} \tilde{r}_k^p + B_k \pi_{k+1}). \end{aligned} \tag{5.57b}$$

As shown in the solution to the estimation quadratic subproblem, we apply a recursion method, updating Π_k and π_k until we obtain their respective values at $k = 1$. At this point, the remaining equations are

$$\begin{bmatrix} \overline{R}_0 & B_0^T & \\ B_0 & & -I \\ & -I & \Pi_1 \end{bmatrix} \begin{bmatrix} \Delta u_0 \\ \Delta \pi_0^p \\ \Delta x_1 \end{bmatrix} = \begin{bmatrix} \tilde{r}_0^u \\ \tilde{r}_0^p \\ \pi_1 \end{bmatrix}, \tag{5.58}$$

yielding

$$\Delta u_0 = (\overline{R}_0 + B_0^T \Pi_1 B_0)^{-1} (\tilde{r}_0^u + B_0^T \pi_1 + B_0^T \Pi_1 \tilde{r}_0^p) \tag{5.59a}$$

$$\Delta x_1 = B_0 \Delta u_0 - \tilde{r}_0^p \tag{5.59b}$$

$$\Delta \pi_0^p = \Pi_1 \Delta x_1 - \pi_1 \tag{5.59c}$$

The back-substitution continues from the first stage to the last:

$$\Delta u_k = (\overline{R}_k + B_k^T \Pi_{k+1} B_k)^{-1} (\tilde{r}_k^u - (\overline{M}_k^T + B_k^T \Pi_{k+1} A_k)\Delta x_k + B_k^T(\pi_{k+1} + \Pi_{k+1}\tilde{r}_k^p)) \tag{5.60a}$$

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta w_k - \tilde{r}_k^p \tag{5.60b}$$

$$\Delta \pi_k^p = \Pi_{k+1}\Delta x_{k+1} - \pi_{k+1} \tag{5.60c}$$

To find the step for the inequality Lagrange multipliers and the soft constraint variables, we introduce the formulas

$$\Delta \epsilon_k = \widetilde{Z}_k^{-1}(\tilde{r}_k^\epsilon + \Sigma_k^{H^{-1}} H_k \Delta x_k) \tag{5.61a}$$

$$\Delta \lambda_k^H = \Sigma_k^{H^{-1}}(\hat{r}_k^H + H_k \Delta x_k - \Delta \epsilon_k) \tag{5.61b}$$

$$\Delta \lambda_k^I = \Sigma_k^{I^{-1}}(\hat{r}_k^I - \Delta \epsilon_k) \tag{5.61c}$$

$$\Delta \lambda_k^D = \Sigma_k^{D^{-1}}(\hat{r}_k^D - G_k \Delta x_k + D_k \Delta u_k). \tag{5.61d}$$

As noted earlier, the slack variable step Δt is obtained through (5.14).

The right hand side variables $\hat{r}$ for the regulator quadratic subproblems for the affine-scaling and center-correcting step are now presented. For the affine-scaling step, the elements of $\hat{r}_Q$ in (5.13) are

$$\hat{r}_k^\epsilon = -(Z_k \epsilon_k - \lambda_k^H - \lambda_k^I + z_k) \tag{5.62a}$$

$$\hat{r}_k^x = -(Q_k x_k + M_k u_k - \pi_{k-1}^p + A_k^T \pi_k^p + H_k^T \lambda_k^H - G_k^T \lambda_k^D + q_k) \tag{5.62b}$$

$$\hat{r}_k^u = -(M_k^T x_k + R_k u_k + B_k^T \pi_k^p + D_k^T \lambda_k^D + r_k) \tag{5.62c}$$

except for k=N, when

$$\hat{r}_N^x = -(Q_N x_N - \pi_{N-1}^p + H_N^T \lambda_N^H + q_N). \tag{5.63}$$

The vectors in $\hat{r}_F$ are defined by

$$\hat{r}_k^p = A_k x_k + B_k u_k - x_{k+1} - f_k \tag{5.64}$$

and the elements of $\hat{r}_C$ are

$$\hat{r}_k^H = H_k x_k - \epsilon_k - h_k \tag{5.65a}$$

$$\hat{r}_k^I = -\epsilon_k + \bar{\epsilon}_k \tag{5.65b}$$

$$\hat{r}_k^D = D_k u_k - G_k x_k - d_k. \tag{5.65c}$$

For the case of the center-correcting step, both $\hat{r}_Q$ and $\hat{r}_F$ are zero. Therefore, only $\hat{r}_C$ is presented:

$$\hat{r}_k^H = -\Lambda_k^{H^{-1}} \left[(\Delta T_k^H)_{\text{aff}} (\Delta \lambda_k^H)_{\text{aff}} + \sigma \mu e \right] \tag{5.66a}$$

$$\hat{r}_k^I = -\Lambda_k^{I^{-1}} \left[(\Delta T_k^I)_{\text{aff}} (\Delta \lambda_k^I)_{\text{aff}} + \sigma \mu e \right] \tag{5.66b}$$

$$\hat{r}_k^D = -\Lambda_k^{D^{-1}} \left[(\Delta T_k^D)_{\text{aff}} (\Delta \lambda_k^D)_{\text{aff}} + \sigma \mu e \right]. \tag{5.66c}$$

A comparison of computational times for the structured solvers presented in this chapter and QPSOL, a standard unstructured commercial quadratic programming package is presented in Figure 5.1. It is evident from the figure that structured solutions become more important for longer horizon lengths.

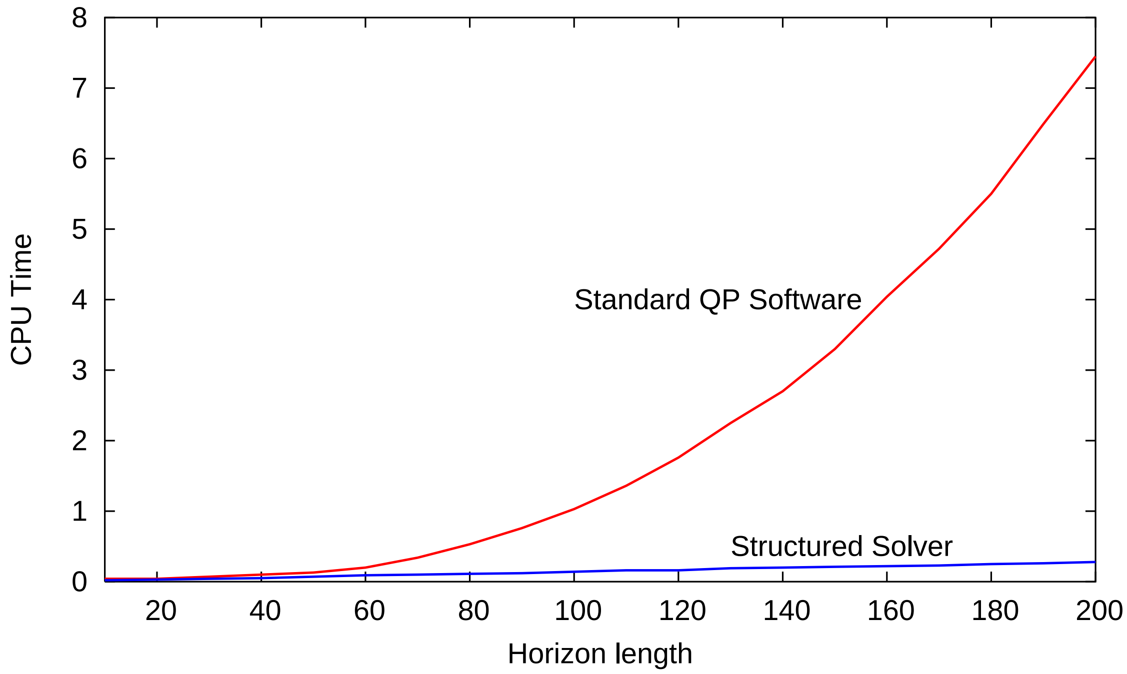

Figure 5.1: Comparison of computational times for structured QP solver and standard method

Chapter 6

Nonlinear Model Predictive Control via Feasibility-Perturbed Sequential Quadratic Programming [1]

Things should be made as simple as possible, but not any simpler.

— Albert Einstein

6.1 Introduction

From an algorithmic point of view, nonlinear model predictive control requires the repeated solution of nonlinear optimal control problems. At certain times during the control period, the state of the system is estimated, and an optimal control problem is solved over a finite time horizon (commencing at the present time), using this state

[1] **This chapter appears in modified form as Tenny, Wright, and Rawlings [118]**

estimate as the initial state. The control component at the current time is used as the input to the system. Algorithms for nonlinear optimal control, which are often specialized nonlinear programming algorithms, can therefore be used in the context of nonlinear model predictive control, with the additional imperatives that the problem must be solved in "real time," and that good estimates of the solution may be available from the state and control profiles obtained at the previous timepoint.

The linear MPC problem is well studied from an optimization standpoint. It gives rise to a sequence of optimal control problems with quadratic objectives and linear dynamics, which can be viewed as structured convex quadratic programming problems. These problems can be solved efficiently by algorithms that exploit the structure. We developed such an algorithm for the general case of linear time-varying systems in Chapter 5. The nonlinear MPC problem has been less widely studied, and is a topic of recent interest.

Albuquerque et al. [1] have used a sequential quadratic programming (SQP) approach, using a primal-dual interior-point method to solve the quadratic programming subproblems. They used a general-purpose solved for sparse symmetric indefinite systems to solve the linear systems arising at each interior-point iteration, together with finite-difference methods to estimate the Hessian terms. They assumed that a good solution estimate was available for each nonlinear problem, so the algorithm contains no techniques to ensure global convergence. An example shows that a large application can be controlled successfully with their approach, in that the optimal control problem can be solved in the interval between changes to the system inputs. Later work of Bartlett, Wächter, and Biegler [2] describes the use of active-set quadratic programming solvers as possible alternatives to the interior-point solvers of [1], and concludes that for problems with few degrees of freedom, the active-set approach may be more efficient. Cervantes et al. [13] described application of an interior-point method directly to the nonlinear optimal control formulation, of the type that has recently been proposed for general nonlinear programming problems. This method generates steps by applying a modified SQP method to a barrier-function reformulation of the nonlinear program, and also makes use of a line search, a merit function, and reduced-space quasi-Newton Hessian approximations. Biegler, Cervantes, and Wächter [5] modify this approach by using a preconditioned conjugate gradient approach to solve the linear equations, allowing them to use finite-difference approximations to the exact Hessian-vector products in place of the explicit quasi-Newton reduced Hessian approximations. Rather than a single merit function, they also use a "filter" criterion to select the line search parameter, an approach that has been the subject of much recent (practical and theoretical) investigation in the context of general nonlinear programming.

Diehl et al. [25] consider the solution of the nonlinear optimal control problem in the specific context of nonlinear MPC. An SQP framework is used, with the Hessians in the quadratic program being either the Hessian of the objective function in the nonlinear MPC problem, or a partitioned quasi-Newton approximation introduced by Bock and Plitt [9]. A line search approach based on a nonsmooth penalty function is used to ensure global convergence; additional details are given in Leineweber et al. [59]. Their method also include an "initial value embedding" strategy, in which approximate derivative information (based on the previous iteration, or a reference trajectory) is used to generate the first SQP step cheaply at each new timepoint.

The works described above have a number of common features. First, they use multiple-shooting or collocation techniques to formulate the underlying continuous-time problem as a problem with finitely many variables that is suitable for solution by structured nonlinear programming techniques. Second, their iterates consist of both control variables and state variables, which are not required to be consistent with respect to the state dynamics. (In the language of nonlinear programming, the iterates are infeasible.) This "simultaneous" approach is known to have advantages in terms of stability; open-loop unstable systems have been observed to give rise to instability in algorithmic approaches that use the model equation to eliminate the states to produce a formulation involving only the control variables. As a result of the infeasible iterates, these algorithms construct a merit function (typically based on the objective function value and the sum of constraint violations) to assess the worth of different points, or else use a filter approach based on the objective function and constraint violations separately. Third, the methods typically use a line search approach to curtail steps that appear to be unacceptable.

In this chapter, we describe an approach that differs from those above in two fundamental ways. First, it computes iterates containing both state and control components, but perturbs these to retain feasibility with respect to the constraints at every interation. Second, it replaces the line-search globalization approach with a scaled trust-region approach. The algorithm is essentially the feasibility-perturbed SQP method described by Wright and Tenny [125], adapted to the nonlinear MPC context. The quadratic programming subproblems are identical to the quadratic programs for linear, time-varying systems, and are solved by using the specialized interior-point approach described in Chapter 5.

By retaining feasibility of all iterates, the algorithm gains several significant advantages. First, the objective function can be used as a merit function, greatly simplifying the description of the algorithm. Algorithms that allow infeasible iterates must construct a merit function from some combination of objective function, constraint infeasibilities, Lagrange multiplier estimates, and various parameters. Since we insist on consistency of the model equation at every iteration, it may be objected that we run the risk of encountering the stability problems that attend the inputs-only formulation. However, by using a change of variables within the feasibility perturbation strategy, we avoid such problems. A second advantage of the feasible approach is that the latest iterate can be used as a (suboptimal) feasible solution, if it is necessary to terminate the solution process early. Third, feasibility may allow a more natural formulation of the problem than may be needed when infeasible iterates are allowed. For example, when one of the states is a concentration of a chemical, a naturally nonnegative variable, algorithms that allow infeasible points may allow (nonphysical) negative values of this variable to be considered at some iterations, leading to unpredictable behavior of the algorithm. This may be remedied by introducing an additional nonnegativity constraint and insisting on feasibility with respect to this constraint, or by a change of variables. A feasible-point algorithm, on the other hand, will not produce nonphysical values of this variable at any iteration, hence allowing a more natural formulation and obviating any additional nonnegativity constraints.

Our computational experience also shows that our algorithm requires fewer iterates than other SQP-type approaches on a wide range of problems. We believe this performance is due in large part to the retention of feasibility, and the avoidance of unpredictable algorithmic behavior that comes with allowing infeasible points.

We note that other features of the approach of Diehl et al. [25] that are specific to the nonlinear MPC context, including the initial-value embedding, can also be incorporated into our approach. We omit further discussion of these issues, however, and focus on the solution of the nonlinear optimal control problem that arises at each timepoint.

An earlier version of the algorithm of this chapter was applied to a copolymerization reactor by Tenny, Rawlings, and Bindlish [116]. Because of the analysis in companion work [125], the algorithm is now on a more solid footing. We also add several enhancements, including the use of a stabilizing change of variables during the feasibility perturbation stage and the choice of a trust region scaling matrix.

The remainder of the chapter is structured as follows. In Section 6.2, we describe briefly the feasibility-perturbed SQP algorithm, Algorithm FP-SQP, of Wright and Tenny [125]. Section 6.3 introduces our formulation of the nonlinear MPC problem we solve, while the SQP subproblem arising from this formulation is presented in Section 6.4. The specific elements of Algorithm FP-SQP are presented in the subsequent sections. Section 6.5 describes the perturbation technique used to maintain feasibility of the iterates. Section 6.6 presents the various Hessian approximations that can be used in the SQP subproblem. Our trust-region scaling matrix for the NMPC problem is derived in Section 6.7. Finally, Section 6.9 contains computational results on a variety of nonlinear control models.

6.2 The Trust-Region Projected Sequential Quadratic Programming Algorithm

We describe the trust-region feasibility-perturbed sequential quadratic programming algorithm, which we refer to as Algorithm FP-SQP, with respect to the following general formulation of a constrained nonlinear optimization problem:

$$\min f(z) \;\text{ subject to }\; c(z) = 0, \;\; d(z) \leq 0, \tag{6.1}$$

where $z \in \mathbb{R}^n$ is the vector of variables, $f : \mathbb{R}^n \to \mathbb{R}$, $c : \mathbb{R}^n \to \mathbb{R}^m$, and $d : \mathbb{R}^n \to \mathbb{R}^r$ are smooth (twice continuously differentiable) functions. We denote by $\mathcal{F}$ the set of feasible points for (6.1). A full description of this algorithm and derivation of its convergence properties appears in [125].

Algorithm FP-SQP generates a sequence of feasible iterates $\{z^j\}_{j=0,1,2,\ldots}$, where a step from the current iterate z to the next iterate is obtained by first solving the following quadratic programming subproblem for Δz:

$$\min_{\Delta z} \; m(\Delta z) \stackrel{\text{def}}{=} \nabla f(z)^T \Delta z + \tfrac{1}{2} \Delta z^T H \Delta z \;\text{ subject to} \tag{6.2a}$$

$$c(z) + \nabla c(z)^T \Delta z = 0, \;\; d(z) + \nabla d(z)^T \Delta z \leq 0, \tag{6.2b}$$

$$\|D \Delta z\|_p \leq \Delta, \tag{6.2c}$$

where (6.2b) represents linearization of the constraints $c(\cdot)$ and $d(\cdot)$ around the current iterate z, (6.2c) is a trust-region constraint (where $p \in [1, \infty]$ denotes the choice of norm), and D is a scaling matrix for the trust region. We make the assumption (Assumption 1 below) that (6.2b) and (6.2c) together bound the size of the full vector Δz. This assumption holds trivially if D in (6.2c) is uniformly nonsingular, but we are interested also in cases in which D has zero eigenvalues. The matrix H in (6.2a) is assumed to be symmetric but not necessarily positive semidefinite. For best local convergence behavior, H should be a good approximation to the Hessian of the Lagrangian for the problem (6.1) on the nullspace of the gradients of the constraints active at the solution z^*.

Having solved (6.2) for Δz, we obtain a candidate step $\widetilde{\Delta z}$ by perturbing Δz in such a way as to satisfy two conditions. First, *feasibility*:

$$z + \widetilde{\Delta z} \in \mathcal{F}. \tag{6.3}$$

Second, *asymptotic exactness*: There is a continuous monotonically increasing function $\phi : \mathbb{R}^+ \to \mathbb{R}^+$ with $\phi(0) = 0$ such that

$$\|\Delta z - \widetilde{\Delta z}\|_2 \leq \phi(\|\Delta z\|_2)\, \|\Delta z\|_2. \tag{6.4}$$

The acceptability of a candidate step $\widetilde{\Delta z}$ depends on a "sufficient decrease" test, which makes use of the ratio ρ_j defined at iteration j as follows:

$$\rho_j = \frac{f(z^j) - f(z^j + \widetilde{\Delta z}^j)}{-m_j(\Delta z^j)}, \tag{6.5}$$

where z^j is the current iterate, Δz^j solves (6.2), and $\widetilde{\Delta z}^j$ satisfies (6.3) and (6.4). If ρ_j exceeds a small positive quantity η, we accept the step and set $z^{j+1} = z^j + \widetilde{\Delta z}^j$, and possibly adjust the trust-region radius Δ and scaling matrix D in preparation for the next iteration. Otherwise, we set $z^{j+1} = z^j$, decrease Δ, and calculate a new candidate step.

We specify the algorithm formally as follows.

Algorithm 6.2.1 (FP-SQP)

Given starting point z_0, *trust-region upper bound* $\bar{\Delta} > 0$, *initial radius* $\Delta_0 \in (0, \bar{\Delta})$, $\eta \in [0, 1/4)$, *and* $p \in [1, \infty]$;

for $j = 0, 1, 2, \cdots$

Obtain Δz^j *by solving (6.2);*

Seek $\widetilde{\Delta z}^j$ *with the properties (6.3) and (6.4);*

if *no such* $\widetilde{\Delta z}^j$ *can be found;*

$\Delta_{j+1} \leftarrow (1/2)\|D_j \Delta z^j\|_p$;

$z^{j+1} \leftarrow z^j$; $D_{j+1} \leftarrow D_j$;

else

Calculate ρ_j *using (6.5);*

if $\rho_j < 1/4$

$\Delta_{j+1} \leftarrow (1/2)\|D_j \Delta z^j\|_p;$

else **if** $\rho_j > 3/4$ *and* $\|D_j \Delta z^j\|_p = \Delta_j$

$\Delta_{j+1} \leftarrow \min(2\Delta_j, \bar{\Delta});$

else

$\Delta_{j+1} \leftarrow \Delta_j;$

if $\rho_j > \eta$

$z^{j+1} \leftarrow z^j + \widetilde{\Delta z}^j;$

choose new scaling matrix D_{j+1};

else

$z^{j+1} \leftarrow z^j;\ D_{j+1} \leftarrow D_j;$

end (for).

Before stating the main convergence results for this algorithm, which are proved in the companion paper [125], we introduce some notation and assumptions.

The Lagrangian function for (6.1) is

$$\mathcal{L}(z, \mu, \lambda) \stackrel{\text{def}}{=} f(z) + \mu^T c(z) + \lambda^T d(z), \tag{6.6}$$

where $\mu \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^r$ are Lagrange multipliers. The Karush-Kuhn-Tucker conditions for (6.1) are as follows:

$$\nabla_z \mathcal{L}(z, \mu, \lambda) = \nabla f(z) + \nabla c(z)\mu + \nabla d(z)\lambda = 0, \tag{6.7a}$$

$$c(z) = 0, \tag{6.7b}$$

$$d(z) \leq 0 \perp \lambda \geq 0, \tag{6.7c}$$

where $\perp$ indicates that $\lambda^T d(z) = 0$. The Mangasarian-Fromovitz constraint qualification (MFCQ) at a feasible point z, which ensures that the linearization of the constraints (6.2b) adequately captures the local geometry of $\mathcal{F}$ near z, requires that

$$\nabla c(z) \text{ has full column rank; and} \tag{6.8a}$$

$$\text{there exists a vector } v \in \mathbb{R}^n \text{ such that}$$

$$\nabla c(z)^T v = 0 \text{ and } v^T \nabla d_i(z) < 0 \text{ for all } i \in \mathcal{A}(z). \tag{6.8b}$$

If z is a stationary point for (6.1) at which (6.8) is satisfied, then there exist vectors μ and λ such that (6.7) is satisfied by the triplet (z, μ, λ).

The level set L_0 for the starting point z^0 of Algorithm FP-SQP is defined as follows:

$$L_0 \stackrel{\text{def}}{=} \{z \,|\, c(z) = 0,\ d(z) \leq 0,\ f(z) \leq f(z^0)\} \subset \mathcal{F}.$$

Our assumption on the trust-region bound (6.2c) is as follows:

Assumption 1 *There is a constant δ such that for all points $z \in L_0$ and all positive definite scaling matrices D used by the algorithm, we have for any Δz satisfying the constraints*

$$c(z) + \nabla c(z)^T \Delta z = 0, \quad d(z) + \nabla d(z)^T \Delta z \leq 0$$

that

$$\delta^{-1} \|\Delta z\|_2 \leq \|D \Delta z\|_p \leq \delta \|\Delta z\|_2. \tag{6.9}$$

In this assumption, the constant that relates $\|\cdot\|_2$ with the equivalent norms $\|\cdot\|_p$ (for $p \in [1, \infty]$) is absorbed into the constant δ.

Our assumption on the boundedness of the set containing the iterates and on smoothness of the functions f, c, and d can be stated as follows.

Assumption 2 *The level set L_0 is bounded, and the functions f, c, and d in (6.1) are twice continuously differentiable in an open neighborhood $\mathcal{N}(L_0)$ of this set.*

Note that L_0 is certainly closed, so that if Assumption 2 holds, it is also compact.

For the third assumption, which bounds the distance from an infeasible point z to the feasible set $\mathcal{F}$ in terms of values of the constraint functions c and d, we define $\mathcal{B}(z,t)$ to be the open Euclidean ball about z:

$$\mathcal{B}(z,t) \stackrel{\text{def}}{=} \{y \mid \|y-z\| < t\}.$$

Here and in all subsequent formulae, an omitted subscript on $\|\cdot\|$ denotes the Euclidean norm.

Assumption 3 *For every point $\hat{z} \in L_0$, there are positive quantities ζ and $\hat{\Delta}_3$ such that for all $z \in \mathrm{cl}(\mathcal{B}(\hat{z},\hat{\Delta}_3))$ we have*

$$\min_{v\in\mathcal{F}} \|v-z\| \le \zeta\left(\|c(z)\| + \|[d(z)]_+\|\right), \tag{6.10}$$

where $[d(z)]_+ = [\max(d_i(z),0)]_{i=1}^r$.

In the companion report [125], we proved two global convergence results for Algorithm FP-SQP. These results differ in their assumptions on the series of Hessian matrices H_j that appear in the quadratic programming subproblem (6.2). The first result makes an assumption on these matrices that is typically satisfied by quasi-Newton updating schemes.

Theorem 6.1 *Suppose that Assumptions 1, 2, and 3 are satisfied, and assume that all limit points of the algorithm satisfy MFCQ. Suppose in addition that the approximate Hessians H_j satisfy the bound $\|H_j\|_2 \le \sigma_0 + j\sigma_1$, for some nonnegative constants σ_0 and σ_1. Then the algorithm has a stationary limit point.*

The second convergence results assumes uniform boundedness of the Hessians H_j. Such a bound would hold if for instance each H_j were taken to be the Hessian of the Lagrangian (6.6), for suitable definitions of the Lagrange multiplier estimates μ and λ, or some positive definite modification of this matrix.

Theorem 6.2 *Suppose that Assumptions 1, 2, and 3 are satisfied, and that the Hessian approximations H_j satisfy $\|H_j\| \leq \sigma$ for all j and some constant σ. Then the algorithm cannot have a limit point $\bar{z}$ at which the MFCQ condition (6.8) holds but the KKT conditions (6.7) are not satisfied.*

Under additional assumptions on the algorithm and the solution of the problem, rapid local convergence can also be proved. We refer the interested reader to [125, Section 4] for details.

6.3 Model Predictive Control: Problem Definition

For the purposes of our study, we examine discrete-time nonlinear systems in which the control moves are injected and measurements are taken at every sampling time. Usually, the system variables are shifted so that the desired value of the state variable is zero, and the fundamental aim of the control strategy is to drive the state variables to zero. At each sampling time, an open-loop optimal control problem is solved over a finite horizon N, with the aim of identifying the input u_0 that should be injected into the system at the present time. This quantity can be obtained by solving a nonlinear optimal control problem whose initial state is the current state of the system. Because u_0 must be calculated in real time, efficient algorithms for solving this nonlinear optimal control problem are desirable. Algorithm FP-SQP is well suited for this type of application, in part because each iterate it generates is feasible and therefore can be used as a suboptimal solution, if it is necessary to terminate the algorithm prior to convergence. This property is especially important in systems with faster sampling rates.

We consider the following formulation of this N-step finite-horizon MPC problem:

$$\min_{x,u,\eta} \Phi(x,u,\eta) \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} \left[\mathcal{C}(x_k,u_k) + \Xi(\eta_k)\right] + \Gamma(x_N) + \Xi(\eta_N) \quad \text{s.t.} \tag{6.11a}$$

$$x_0 \text{ given}, \quad x_{k+1} = F(x_k,u_k), \quad Du_k \le d, \quad Gx_k - \eta_k \le g, \quad \eta_k \ge 0, \tag{6.11b}$$

where x, u, and η denote the sequences of vectors representing states, inputs, and state constraint violations, respectively; that is,

$$x = (x_1, x_2, \ldots, x_N),$$

$$u = (u_0, u_1, \ldots, u_{N-1}),$$

$$\eta = (\eta_1, \eta_2, \ldots, \eta_N).$$

Changing the use of m and n from Section 6.2, we suppose that $x_k \in \mathbb{R}^n$, $k = 0, 1, \ldots, N$ and $u_k \in \mathbb{R}^m$, $k = 0, 1, \ldots, N-1$. The stage cost $\mathcal{C}(x_k, u_k)$ and the terminal penalty $\Gamma(x_N)$ in (6.11a) are convex. Moreover, they are typically quadratic and *strictly* convex with respect to the input variables u_k in their arguments. The model function $F(x_k, u_k)$ is usually the source of nonlinearity in the optimization problem, making (6.11) a nonlinear programming problem. The model equation is usually obtained by integrating a DAE model.

A more natural formulation of the MPC problem might impose the state constraints $Gx_k \le g$ explicitly, but the formulation (6.11b) represents a "softening" of these constraints by introducing the violation variables η and including penalty terms $\Xi(\cdot)$ on the violations in the objective function (6.11a). We assume, as is usual, that these penalties also are quadratic; that is,

$$\Xi(\eta_k) = \eta_k^T \Psi \eta_k + \psi^T \eta_k. \tag{6.12}$$

Softening of state constraints often makes sense in terms of the problem formulation. The input constraints $Du_k \le d$ represent physical restrictions on the available control action (for instance, a limit on available power, flow rate, or voltage), so it is natural to make the constraints "hard" in the formulation (6.11b). By contrast, state constraints often represent desired values for profitability, safety, or convenience, so that violation of these constraints is a condition to be discouraged rather than forbidden in the formulation. Since the penalty terms for the constraint violations η_k is quadratic, and since these terms are defined by linear inequalities, they do not contribute to the "nonlinear" nature of the problem (6.11), and play little part in the development of this section.

If the constraints $Du_k \le d$, $k = 0, 1, \ldots, N-1$ are feasible, then the full set of constraints (6.11b) is feasible. From any input sequence u that satisfies $Du_k \le d$, we recover the states x_k and violationsη_k by setting $x_{k+1} = F(x_k, u_k)$ and $\eta_k = \max(Gx_k - g, 0)$ for all k.

For the purposes of this chapter, we consider the problem (6.11) in isolation, as a single nonlinear program that we wish to solve. To put this problem in context, however, we give a brief description of MPC methodology. In MPC, a sequence of problems of the form (6.11) are solved, one at each sampling point. A starting point for the input vector sequence $\{u_k\}$ in (6.11) can be constructed by shifting the input vectors obtained at the solution form the previous sampling point forward by one stage, and using an educated guess of the remaining value u_{N-1}, based on the solution of the discrete-time linear quadratic regulator solution. A full feasible point for (6.11) can then be obtained in the manner described above.

At the very first sampling time, no previous input trajectory is available for defining the starting point. A poor choice of this initial point may cause the optimizer to find undesirable local minima. Our approach is to construct u_k and x_{k+1} sequentially for $k = 0, 1, \ldots, N-1$ as follows. We define a starting guess for u'_k by setting $u'_k = u_{k-1}$, and linearize the model F about (x_k, u'_k). We then calculate the locally optimal linear infinite horizon feedback law for the resulting linear model with quadratic objective. Application of this law yields an improved estimate u_k, which we accept as the initial guess. (In this development, we assume that the dynamics of the model do not change greatly between each sampling time.) Application of the model equation in (6.11b) now yields x_{k+1}. The first value u_0 can be attained by starting with an initial guess u'_0, linearizing the system about x_0 and the guess, and determining the optimal feedback gain K_0. This value of K_0 generates a new value for $u_0 = K_0 x_0$. The linearization and recalculation of K_0 is repeated until the guess for u_0 converges to within a tolerance. If the initial guess violates the hard constraints on the inputs, the result is adjusted to a feasible input; see (6.21).

6.4 The SQP Subproblem

In the problem (6.11), inputs and states at the current sampling time k directly affect future states $x_{k+1}, x_{k+2}, \ldots,$ but have no effect on states and inputs at previous times. Due to this causal nature, the optimization problems are highly structured. For an appropriate "interleaved" ordering of the components of u, x, and η, the Hessian of the objective and the Jacobian of the constraints in (6.11) are block-banded. Rao, Wright, and Rawlings [98] exploit this structure for the case of linear model F and convex quadratic $\mathcal{C}$ and Φ by developing a customized primal-dual interior-point method for the resulting quadratic programming problem. With some alterations, this method can be used to solve the subproblems arising in the SQP algorithm of Section 6.2, applied to (6.11). We devise other features of the SQP algorithm to ensure that this structure can be exploited at the level of the quadratic programming subproblem; for example, by choosing the approximate Hessians H in (6.2a) to have the same structure as the exact Hessians.

For the remainder of the MPC discussion, it is convenient to introduce the following definitions, based on the formulation in (6.11):

$$Q_k = \frac{\partial^2 \Phi}{\partial x_k^2} = \frac{\partial^2 \mathcal{C}(x_k, u_k)}{\partial x_k^2}, \tag{6.13a}$$

$$R_k = \frac{\partial^2 \Phi}{\partial u_k^2} = \frac{\partial^2 \mathcal{C}(x_k, u_k)}{\partial u_k^2}, \tag{6.13b}$$

$$M_k = \frac{\partial^2 \Phi}{\partial u_k \partial x_k} = \frac{\partial^2 \mathcal{C}(x_k, u_k)}{\partial u_k \partial u_k}, \tag{6.13c}$$

$$A_k = \frac{\partial F(x_k, u_k)}{\partial x_k}, \tag{6.13d}$$

$$B_k = \frac{\partial F(x_k, u_k)}{\partial u_k} \tag{6.13e}$$

In the SQP approach applied to (6.11), the subproblem has a structure similar to (6.11), except that the model equation is linearized, and the objective is replaced by a quadratic whose second-order terms are approximations to the Hessian of the Lagrangian function for (6.11). To be precise, the subproblem is as follows:

$$\begin{gathered}\min_{\Delta x, \Delta u, \Delta \eta} \tfrac{1}{2} \Delta u_0^T \tilde{R}_0 \Delta u_0 + r_0^T \Delta u_0 + \\ \sum_{k=1}^{N-1} \left\{ \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \right\} \\ + \tfrac{1}{2} \Delta x_N^T \tilde{Q}_N \Delta x_N + q_N^T \Delta x_N + \sum_{k=1}^{N} \Xi(\eta_k + \Delta \eta_k)\end{gathered} \tag{6.14}$$

subject to

$$\Delta x_0 = 0, \tag{6.15a}$$

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k, \quad k = 0, 1, \ldots, N-1, \tag{6.15b}$$

$$D(u_k + \Delta u_k) \le d, \quad k = 0, 1, \ldots, N-1, \tag{6.15c}$$

$$G(x_k + \Delta x_k) - (\eta_k + \Delta \eta_k) \le g, \quad k = 1, 2, \ldots, N, \tag{6.15d}$$

$$\eta_k + \Delta \eta_k \ge 0, \quad k = 1, 2, \ldots, N, \tag{6.15e}$$

$$\|\Sigma_k \Delta u_k\|_\infty \le \Delta, \quad k = 0, 1, \ldots, N-1. \tag{6.15f}$$

Note that feasibility of the current iterate x, u, η (in particular, $x_{k+1} = F(x_k, u_k)$) is exploited in defining the linearization of the model equation (6.15b). The blocks that make up the Lagrangian Hessian approximation are denoted by $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ in (6.14). We discuss various choices for these approximations in Section 6.6. We have assumed that the constraint violation penalties $\Xi(\cdot)$ are quadratic, as in (6.12). Note that the trust-region constraint (6.15f) is applied only to the $\{\Delta u\}$ components. We discuss the choice of scaling matrices Σ_k in Section 6.7.

6.5 Feasibility Perturbation

In this section, we describe the perturbation technique that is used to recover a feasible step $\widetilde{\Delta x}, \widetilde{\Delta u}, \widetilde{\Delta \eta}$ from the trust-region SQP step $\Delta x, \Delta u, \Delta \eta$. The perturbed step should satisfy the asymptotic exactness condition (6.4).

The easiest way to perturb the step is to set $\widetilde{\Delta u} = \Delta u$, and then recover the new states from the model equation (6.11b); that is, $\widetilde{\Delta x}$ satisfies

$$x_{k+1} + \widetilde{\Delta x}_{k+1} = F(x_k + \widetilde{\Delta x}_k, u_k + \Delta u_k), \quad k = 0, 1, \ldots, N-1, \tag{6.16}$$

(with $\widetilde{\Delta x}_0 = \Delta x_0 = 0$). The perturbed constraint violations $\{\widetilde{\Delta \eta}\}$ are recovered from (6.15d) and (6.15e) by setting

$$\eta_k + \widetilde{\Delta \eta}_k = \max\left(G(x_k + \widetilde{\Delta x}_k) - g, 0\right), \quad k = 1, 2, \ldots, N. \tag{6.17}$$

This scheme is obvious, and often works well, but it has potential problems when the system is open-loop unstable at the set-point; that is, $\partial F / \partial x(0, 0)$ has spectral radius greater than 1. The system is ill-conditioned in the sense that the forward integration used to recover $\widetilde{\Delta x}$ in (6.16) contains "increasing modes", which amplify perturbations and cause large differences between Δx_k and $\widetilde{\Delta x}_k$ as k grows. In this case, it is unlikely that $x + \widetilde{\Delta x}$ will be close to a true projection of $x + \Delta x$ onto the feasible set for (6.11b).

A more stable perturbation technique is obtained by using a change of variable, motivated by linear MPC theory, to stabilize the forward integration. We define a set of stagewise variables v as follows:

$$v_k = (u_k + \Delta u_k) - K_k(x_k + \Delta x_k), k = 0, 1, \ldots, N-1, \tag{6.18}$$

where K_k is a feedback gain matrix constructed such that $|\text{eig}(A_k + B_k K_k)| < 1$. Several choices of K_k are possible. One method is to use *pole placement*, choosing the poles of the closed loop system arbitrarily inside the unit circle, and then choosing K_k such that the eigenvalues of $A_k + B_k K_k$ coincide with these poles. A second method is to obtain K_k by solving the optimal infinite horizon linear quadratic regulator problem at each (A_k, B_k) pair. This K_k is then guaranteed to stabilize the linear system defined by (A_k, B_k). It is often possible to reuse the same K_k over a number of different stages k, provided the eigenvalues of $(A_k + B_k K_k)$ stay inside the unit circle, and thereby reduce the computational load associated with calculating each K_k online. In many cases, a single stabilizing "time-invariant" K_k can be used at all stages, usually by solving the optimal infinite-horizon linear-quadratic regulator problem on the linearized system at the target.

We use a third approach to select the feedback gain matrices K_k, based on solution of a finite-horizon discrete-time linear-quadratic regulator for time-varying systems. This approach is guaranteed to yield K_k for which $|\text{eig}(A_k + B_k K_k)| < 1$ for many, but not all systems. The procedure for defining K_k is developed from dynamic programming arguments, starting from the end of the prediction horizon and working towards the beginning. The result is the discrete-time Riccati equation [4]. First, define $\Pi_N = Q_N$, and then apply the following recursions for $k = N-1, N-2, \ldots, 1$:

$$\begin{aligned}
K_k &= -\left(R_k + B_k^T \Pi_{k+1} B_k\right)^{-1} \left(M_k^T + B_k^T \Pi_{k+1} A_k\right) \\
\Pi_k &= Q_k + K_k^T R_k K_k + M_k K_k + K_k^T M_k^T + \left(A_k + B_k K_k\right)^T \Pi_{k+1} \left(A_k + B_k K_k\right).
\end{aligned}$$

If one of the K_k matrices generated by this technique does not stabilize (A_k, B_k), then K_k may be selected instead from one of the other techniques mentioned above. However, experience has shown that it is often better to use the K_k from the above approach because it most closely resembles the solution of the quadratic subproblem. The feedback gains K_k obtained from the discrete Riccati scheme are consistent with the optimal solution for the unconstrained time-varying linear system along the trajectory of the current guess. This scheme for calculating the K_k is not as computationally intensive as the first two approaches described above, and it represents the system at the current iterate well.

Given the stabilizing feedback gain matrices K_k and the SQP step, we can define v from the formulae (6.18). We may view v as the deviation from a stable closed-loop trajectory. We then determine the perturbed components $\widetilde{\Delta x}$ and $\widetilde{\Delta u}$ as follows:

$$\widetilde{\Delta u}_k = K_k(x_k + \widetilde{\Delta x}_k) - u_k + v_k, \quad k = 0, 1, \ldots, N-1, \tag{6.19a}$$

$$\widetilde{\Delta x}_{k+1} = F(x_k + \widetilde{\Delta x}_k, u_k + \widetilde{\Delta u}_k) - x_k, \quad k = 0, 1, \ldots, N-1. \tag{6.19b}$$

By combining (6.18) with (6.19a), we see immediately that

$$\widetilde{\Delta u}_k - \Delta u_k = K_k(\widetilde{\Delta x}_k - \Delta x_k). \tag{6.20}$$

Note $\widetilde{\Delta x}_0 = \Delta x_0 = 0$ in (6.19a) and (6.20), so from the latter equation we have in particular that $\widetilde{\Delta u}_0 = \Delta u_0$.

When there is a constraint $Du_k \leq d$ on the inputs, we modify the procedure above by solving the following subproblem after the calculation of each $\widetilde{\Delta u}_k$ from (6.19a):

$$\min_{\widehat{\Delta u}_k} (\widehat{\Delta u}_k - \widetilde{\Delta u}_k)^T R_k (\widehat{\Delta u}_k - \widetilde{\Delta u}_k) \text{ subject to } D(u_k + \widehat{\Delta u}_k) \leq d. \tag{6.21}$$

We then make the replacement $\widetilde{\Delta u}_k \leftarrow \widehat{\Delta u}_k$, and proceed with (6.19b).

The definition (6.17) can again be used to recover the perturbed step in the η components.

In Section 6.5.1, we show that this feasibility projection approach satisfies the asymptotic exactness condition (6.4) under reasonable assumptions, and also suggest why the stabilization scheme improves the results.

In the case of open-loop unstable models, the SQP subproblem itself may be ill-conditioned. The change of variables in (6.18) may also be used in the QP to improve the conditioning. In fact, the re-conditioning can be performed by using the same values for K_k as we derived above for the perturbation.

6.5.1 Asymptotic Exactness of Feasibility-Perturbed Step

In this section, we show that the feasibility perturbation scheme proposed in Section 6.5 satisfies the asymptotic exactness condition (6.4). For this purpose, we assume that the constraints $Du_k \leq d$ are not present (although the proof can be extended to handle this case).

The key conditions relating the perturbed steps $\widetilde{\Delta u}_k$, $k = 0, 1, \ldots, N-1$ and $\widetilde{\Delta x}_k$, $k = 1, 2, \ldots, N$ to their original SQP-step counterparts Δu_k and Δx_k are (6.19b) and (6.20). By setting $k = 0$ in (6.20), we see that

$$\widetilde{\Delta u}_0 = \Delta u_0. \tag{6.22}$$

To analyze the relationship between the original and perturbed SQP steps, we simplify the notation as follows:

$$y = (x_1, x_2, \ldots, x_N), \quad w = (u_1, u_2, \ldots, u_{N-1}), \quad w_0 = u_0,$$
$$c(y, w, w_0) = [x_{k+1} - F(x_k, u_k)]_{N-1}^{k=0}, \quad K = \begin{bmatrix} K_1 & & & & 0 \\ & K_2 & & & 0 \\ & & \ddots & & \vdots \\ & & & K_{N-1} & 0 \end{bmatrix},$$

so that (6.20) becomes

$$\widetilde{\Delta w} - \Delta w = K(\widetilde{\Delta y} - \Delta y), \tag{6.23}$$

while (6.19b) becomes

$$c(y + \widetilde{\Delta y}, w + \widetilde{\Delta w}, w_0 + \widetilde{\Delta w}_0) = 0. \tag{6.24}$$

Because $(\Delta y, \Delta w, \Delta w_0)$ is an SQP step, we have

$$c_y(y, w, w_0)\Delta y + c_w(y, w, w_0)\Delta w + c_{w_0}(y, w, w_0)\Delta w_0 = 0. \tag{6.25}$$

The feasibility-perturbed step $(\widetilde{\Delta y}, \widetilde{\Delta w})$ satisfies the algebraic conditions (6.23) and (6.24), which can be viewed a parametrized system of nonlinear equations, with parameter $\widetilde{\Delta w_0} = \Delta u_0$. Note that this system is "square;" that is, the number of equations equals the number of unknowns. Hence, we can obtain the asymptotic exactness result by applying the implicit function theorem. The Jacobian of this system at $(\widetilde{\Delta y}, \widetilde{\Delta w}) = (0,0)$ is

$$\begin{bmatrix} c_y(y,w,w_0) & c_w(y,w,w_0) \\ K & -I \end{bmatrix}, \tag{6.26}$$

while its residual at the point $(\widetilde{\Delta y}, \widetilde{\Delta w}) = (\Delta y, \Delta w)$ is

$$\begin{aligned} &\begin{bmatrix} c(y+\Delta y, w+\Delta w, w_0+\Delta w_0) \\ K(\Delta y - \Delta y) - (\Delta w - \Delta w) \end{bmatrix} \\ &= \begin{bmatrix} c_y \Delta y + c_w \Delta w + c_{w_0} \Delta w_0 \\ 0 \end{bmatrix} + O(\|(\Delta y, \Delta w, \Delta w_0)\|^2) \\ &= O(\|(\Delta y, \Delta w, \Delta w_0)\|^2), \end{aligned} \tag{6.27}$$

where we used smoothness of c and (6.25). If we assume that K is chosen so that the matrix (6.26) is nonsingular, and if $(\Delta y, \Delta w, \Delta w_0)$ is sufficiently small, the implicit function theorem together with (6.27) implies that

$$(\Delta y - \widetilde{\Delta y}, \Delta w - \widetilde{\Delta w}) = O\left\|(\Delta y, \Delta w, \Delta w_0)\right\|^2.$$

It follows immediately from (6.22) that

$$\frac{\left\|(\Delta y, \Delta w, \Delta w_0) - (\widetilde{\Delta y}, \widetilde{\Delta w}, \widetilde{\Delta w_0})\right\|}{\|(\Delta y, \Delta w, \Delta w_0)\|} = O\left(\|(\Delta y, \Delta w, \Delta w_0)\|\right) = o(1), \tag{6.28}$$

as required.

Note that the choice $K = 0$ (no stabilization) is sufficient to satisfy the assumptions above, provided that $c_y(y, w, w_0)$ is nonsingular. However, the nonzero choice of K in Section 6.5 is designed to ensure that (6.26) is much better conditioned than $c_y(y, w, w_0)$, and thus that the constant in the $O(\cdot)$ term in (6.28) is smaller.

Finally, we show that asymptotic exactness also holds for the η_k components. From the SQP step condition, we have similarly to (6.17) that

$$\eta_k + \Delta\eta_k = \max(G(x_k + \Delta x_k) - g, 0).$$

By comparing this expression with (6.17), we obtain

$$\|\Delta\eta_k - \widetilde{\Delta\eta}_k\| \leq \|G(\Delta x_k - \widetilde{\Delta x}_k)\| = O(\|\Delta x_k - \widetilde{\Delta x}_k\|).$$

Asymptotic exactness in these components now follows immediately from (6.28).

6.6 Approximate Hessians

We now describe various ways to choose the matrices $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ that appear in the objective of the SQP subproblem (6.14). Since the variables η enter the constraints of (6.11) linearly and the objective quadratically, the terms in the Hessian of the quadratic involving these variables are constant, so it is not necessary to seek approximations of these terms. Hence, for clarity, we omit these variables from the formulation considered in this section and the next, although our implementations described in Section 6.9 solve the full problem (6.11).

By omitting the soft state constraints from (6.11), we obtain

$$\min_{x,u} \Phi(x, u) \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} \mathcal{C}(x_k, u_k) + \Gamma(x_N) \quad \text{s.t.} \tag{6.29a}$$

$$x_0 \text{ given}, \quad x_{k+1} = F(x_k, u_k), \quad Du_k \leq d, \quad k = 0, 1, \ldots, N-1, \tag{6.29b}$$

while the SQP subproblem has the following form:

$$\min_{\Delta x,\Delta u} \tfrac{1}{2}\Delta u_0^T \tilde{R}_0 \Delta u_0 + r_0^T \Delta u_0 + \tfrac{1}{2}\Delta x_N^T \tilde{Q}_N \Delta x_N + q_N^T \Delta x_N + \sum_{k=1}^{N-1} \left\{ \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \right\} \tag{6.30}$$

subject to

$$\Delta x_0 = 0, \tag{6.31a}$$

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k, \quad k = 0,1,\ldots,N-1, \tag{6.31b}$$

$$D(u_k + \Delta u_k) \le d, \quad k = 0,1,\ldots,N-1, \tag{6.31c}$$

$$\|\Sigma_k \Delta u_k\|_\infty \le \Delta, \quad k = 0,1,\ldots,N-1. \tag{6.31d}$$

The Lagragian for the problem (6.29) is as follows:

$$\begin{aligned} \mathcal{L}(x,u,\lambda,\mu) &= \Phi(x,u) + \sum_{k=0}^{N-1} \lambda_k^T (F(x_k,u_k) - x_{k+1}) + \mu_k^T (Du_k - d) \qquad (6.32) \\ &= \sum_{k=0}^{N-1} \left[\mathcal{C}(x_k,u_k) + \lambda_k^T (F(x_k,u_k) - x_{k+1}) + \mu_k^T (Du_k - d)\right] + \Gamma(x_N). \end{aligned}$$

We can decompose the Lagrangian in a stagewise fashion, as follows:

$$\mathcal{L}(x,u,\lambda,\mu) = \mathcal{L}_0(u_0,\lambda_0,\mu_0) + \sum_{k=1}^{N-1} \mathcal{L}_k(x_k,u_k,\lambda_{k-1},\lambda_k,\mu_k) + \mathcal{L}_N(x_N,\lambda_{N-1}), \tag{6.33}$$

where

$$\begin{aligned} \mathcal{L}_0(u_0,\lambda_0,\mu_0) &= \mathcal{C}(x_0,u_0) + \lambda_0^T F(x_0,u_0) + \mu_0^T (Du_0 - d), \\ \mathcal{L}_k(x_k,u_k,\lambda_{k-1},\lambda_k,\mu_k) &= \mathcal{C}(x_k,u_k) + \lambda_k^T F(x_k,u_k) - \lambda_{k-1}^T x_k + \mu_k^T (Du_k - d), \\ \mathcal{L}_N(x_N,\lambda_{N-1}) &= \Gamma(x_N) - \lambda_{N-1}^T x_N. \end{aligned}$$

Note that each x_k and u_k appear only in $\mathcal{L}_k$, and that each $\mathcal{L}_k$ depend on just a few of the Lagrange multiplier components.

Using the definitions above, we now discuss various options for choosing the Hessian terms in the SQP subproblem (6.30), (6.31).

6.6.1 Exact Hessian

The first option is to use the exact Hessians of the Lagrangian; that is,

$$\hat{Q}_k = \frac{\partial^2 \mathcal{L}_k}{\partial x_k^2}, \quad \hat{R}_k = \frac{\partial^2 \mathcal{L}_k}{\partial u_k^2}, \quad \hat{M}_k = \frac{\partial^2 \mathcal{L}_k}{\partial u_k \partial x_k}. \tag{6.34}$$

The full Hessian of $\mathcal{L}$ with respect to the variables x and u, using a stagewise interleaving of these variables, has the following block-banded structure:

$$\nabla^2_{x,u}\mathcal{L} = \begin{bmatrix} \hat{R}_0 & & & & & & & & \\ & \hat{Q}_1 & \hat{M}_1 & & & & & & \\ & \hat{M}_1^T & \hat{R}_1 & & & & & & \\ & & & \hat{Q}_2 & \hat{M}_2 & & & & \\ & & & \hat{M}_2^T & \hat{R}_2 & & & & \\ & & & & & \ddots & & & \\ & & & & & & \hat{Q}_{N-1} & \hat{M}_{N-1} & \\ & & & & & & \hat{M}_{N-1}^T & \hat{R}_{N-1} & \\ & & & & & & & & \hat{Q}_N \end{bmatrix} \tag{6.35}$$

Note in particular that the Hessian of the Lagrangian is structured identically to the Hessian of the objective in (6.29), and to the Hessian of the linear MPC problem, because there are no coupled interactions between future and past states and inputs other than through the model equality constraint.

We can also use a finite-difference approximation to the terms in (6.34). Note that each $\hat{Q}_k$ consists of Q_k from (6.13a) added to a contribution involving λ_k and the second partial derivatives of the model function F. If the model is linear, we have $\hat{Q}_k = Q_k$. Similar comments apply for $\hat{R}_k$ and $\hat{M}_k$.

These choices for the Hessians of the SQP subproblem have the advantage of rapid local convergence, under appropriate assumptions. The disadvantage is that second derivatives of the model F may be difficult to compute by hand, and time-consuming to approximate with a difference approximation. Moreover, the block-diagonal matrices

$$\mathcal{H}_k \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} = \begin{bmatrix} \hat{Q}_k & \hat{M}_k \\ \hat{M}_k^T & \hat{R}_k \end{bmatrix} \tag{6.36}$$

may not be positive definite, in which case the SQP subproblem will not be a convex quadratic program. Since the solver described in Chapter 5 requires a convex quadratic objective, these matrices may need to be modified in order for the solver to work.

6.6.2 Hessians of the Objective

An approximation that is often effective is to simply ignore the contributions from F and use

$$\tilde{Q}_k = Q_k, \quad \tilde{R}_k = R_k, \quad \tilde{M}_k = M_k. \tag{6.37}$$

This is known as the "Gauss-Newton" choice in some papers. Although this choice is not asymptotically equivalent to the ideal choice of Section 6.6.1 unless the model function F is linear, these matrices often have the right scale. Moreover, the matrix

$$\mathcal{H}_k \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} = \begin{bmatrix} Q_k & M_k \\ M_k^T & R_k \end{bmatrix} \tag{6.38}$$

is constant and positive semidefinite when (as is usually the case) the cost function $\mathcal{C}$ in (6.29) is quadratic and convex.

6.6.3 Full Quasi-Newton Approximation

Quasi-Newton variants of SQP for nonlinear programming have been the subject of extensive research; see [80, Chapter 18]. Indeed, most existing implementations of

SQP use quasi-Newton Hessian approximations, either to the full Langrangian Hessian, or to the projection of this Hessian onto the nullspace of the active constraints. Here, we consider only methods of the former kind. The Hessian approximations are updated after each iteration, using information about the step just taken and the difference in the Lagrangian first derivatives between the current iterate and the previous one. Specifically, we have the step vector s defined by

$$s = \begin{bmatrix} x^+ - x \\ u^+ - u \end{bmatrix}, \tag{6.39}$$

where x^+ and u^+ denote the new iterates and x and u the current iterates; and

$$\begin{aligned} y &= \begin{bmatrix} \nabla_x \mathcal{L}(x^+, u^+, \lambda, \mu) - \nabla_x \mathcal{L}(x, u, \lambda, \mu) \\ \nabla_u \mathcal{L}(x^+, u^+, \lambda, \mu) - \nabla_u \mathcal{L}(x, u, \lambda, \mu) \end{bmatrix} \\ &= \begin{bmatrix} \nabla_x \Phi(x^+, u^+) - \nabla_x \Phi(x, u) + \sum_{k=0}^{N-1} (A_k(x_k^+, u_k^+) - A_k(x_k, u_k))^T \lambda_k \\ \nabla_u \Phi(x^+, u^+) - \nabla_u \Phi(x, u) + \sum_{k=0}^{N-1} (B_k(x_k^+, u_k^+) - B_k(x_k, u_k))^T \lambda_k \end{bmatrix}, \end{aligned} \tag{6.40}$$

where we have used the notation (6.13) to define A_k and B_k.

The damped BFGS approach (see Powell [89] and Nocedal and Wright [80, p. 540]) maintains a positive definite approximation H to the full Lagrangian Hessian (6.35), and updates it after each step according to the following rule:

Algorithm 6.6.1 (damped BFGS)

if $s^T y < 0.2 s^T H s$

define $\theta = 0.8 s^T H s / (s^T H s - s^T y)$

set $y \leftarrow \theta y + (1 - \theta) H s$

end if

update H *as follows:*

$$H \leftarrow H - \frac{Hss^T H}{s^T H s} + \frac{yy^T}{y^T s}. \tag{6.41}$$

A sensible starting initial guess for H is the matrix (6.35) with the approximations (6.37), possibly with the addition of a multiple of the identity matrix to ensure strict positive definiteness.

A naive application of this procedure to our problem is not practical, since after just one update H becomes in general a fully dense matrix. Since it does not preserve the block structure of (6.35), we cannot use the efficient quadratic programming technique to solve the SQP subproblem.

6.6.4 Sparsified Quasi-Newton Approximation

An ad-hoc alternative to the approach just discussed is to impose the desired sparsity pattern (that is, the pattern in (6.35)) on the Hessian approximation H after each step. In other words, we carry out the procedure above to update H, and then zero out all parts of H that are outside the block-diagonal band in (6.35). It can be shown that this approach maintains positive definiteness of the approximations (given a positive definite initial approximation to H). However, approaches that enforce specific sparsity patterns in this fashion do not enjoy a good reputation, even for unconstrained problems.

6.6.5 Partitioned Quasi-Newton Approaches

We now consider approaches that maintain separate approximations to the Hessians of each of the component Lagrangians $\mathcal{L}_k$ in (6.33). Griewank and Toint [36] proposed methods of this type for partially separable nonlinear systems, and these methods were first applied to nonlinear control problems by Bock and Plitt [9]. In our problem, the Lagrangian in (6.33) is in fact *completely* separable in the state and input variables. Each pair (x_k, u_k) appears only in the term $\mathcal{L}_k$, and the coupling between stages comes only in the model equation.

We apply this approach by defining "stagewise" versions of the s and y vectors in (6.39) and (6.40). For $k = 1, 2, \ldots, N-1$, we have

$$s_k = \begin{bmatrix} x_k^+ - x_k \\ u_k^+ - u_k \end{bmatrix}, \tag{6.42}$$

and

$$\begin{aligned} y_k &= \begin{bmatrix} \frac{\partial}{\partial x_k}\mathcal{L}_k(x_k^+, u_k^+, \lambda_{k-1}, \lambda_k, \mu_k) - \frac{\partial}{\partial x_k}\mathcal{L}_k(x_k, u_k, \lambda_{k-1}, \lambda_k, \mu_k) \\ \frac{\partial}{\partial u_k}\mathcal{L}_k(x_k^+, u_k^+, \lambda_{k-1}, \lambda_k, \mu_k) - \frac{\partial}{\partial u_k}\mathcal{L}_k(x_k, u_k, \lambda_{k-1}, \lambda_k, \mu_k) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial}{\partial x_k}\mathcal{C}(x_k^+, u_k^+) - \frac{\partial}{\partial x_k}\mathcal{C}(x_k, u_k) + (A_k(x_k^+, u_k^+) - A_k(x_k, u_k))^T\lambda_k \\ \frac{\partial}{\partial u_k}\mathcal{C}(x_k^+, u_k^+) - \frac{\partial}{\partial u_k}\mathcal{C}(x_k, u_k) + (B_k(x_k^+, u_k^+) - B_k(x_k, u_k))^T\lambda_k \end{bmatrix}, \end{aligned} \tag{6.43}$$

where we again used the notation (6.13) for A_k and B_k. For the initial and final stages, we have

$$\begin{aligned} s_0 &= u_0^+ - u_0, \\ y_0 &= \frac{\partial}{\partial u_0}\mathcal{C}(x_0^+, u_0^+) - \frac{\partial}{\partial u_0}\mathcal{C}(x_0, u_0) + (A_0(x_0^+, u_0^+) - A_0(x_0, u_0))^T\lambda_0 \\ s_N &= x_N^+ - x_N \\ y_N &= \frac{\partial}{\partial x_N}\Gamma(x_N^+) - \frac{\partial}{\partial x_N}\Gamma(x_N). \end{aligned} \tag{6.44}$$

We now use s_k and y_k to maintain an approximation H_k to the kth diagonal block in (6.35), which for all but the initial and final stages has the form

$$H_k = \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \tag{6.45}$$

A damped-BFGS variant of the partitioned approach uses the update strategy of Algorithm 6.6.1, applied to each stage k separately. To be precise, we apply Algorithm 6.6.1 to each stage $k = 0, 1, \ldots, N$, with s_k, y_k, and H_k replacing s, y, and H, respectively. We then obtain decompose the approximate Hessians H_k according to (6.45), to obtain the matrices $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ to be used in (6.30).

As well as being efficient, this approach maintains positive semidefiniteness of the diagonal blocks, so the SQP subproblem can be passed to the convex quadratic programming solver without complications.

An alternative approach is to use the symmetric rank-1 (SR1) update (see [80, Section 8.2]) in place of the damped BFGS approach. The update formula for the stage-k block is as follows:

$$H_k \leftarrow H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k}, \tag{6.46}$$

where the update is skipped if the denominator in (6.46) is too small; that is, if the following criterion is satisfied:

$$\left| (y_k - H_k s_k)^T s_k \right| < 10^{-6} \|y_k - H_k s_k\| \, \|s_k\|. \tag{6.47}$$

The Hessian approximations $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ are then recovered from the decomposition (6.45).

The symmetric rank-one (SR1) update is in a sense more natural than BFGS, as it does not maintain positive definiteness (and there is no reason to expect the diagonal blocks of (6.35) to be positive definite). However, the indefinite approximations may cause problems for the quadratic programming solver. We handle this by passing to the quadratic programming solver a version of H_k in which the negative eigenvalues have been replaced by zero. That is, we form the eigenvalue decomposition

$$H_k = V_k \Lambda_k V_k^T,$$

where Λ_k is a diagonal matrix containing the eigenvalues, and V_k^T is orthogonal. We then redefine H_k to be

$$H_k \leftarrow V_k \Lambda_k^+ V_k^T,$$

where Λ_k^+ is obtained from Λ_k by replacing the negative diagonals by zero. Since the matrices H_k are small in our applications, the cost of performing these eigenvalue decompositions is relatively trivial.

6.7 Trust Region Scaling

In our problem (6.11), the states x and the constraint violations η are fully determined by the inputs u and the constraints (6.11b). Therefore, rather than apply a trust region constraint to all the variables in the SQP method, we define the trust region only in terms of the inputs. In this section, we discuss the issue of *scaling* the trust region constraint; that is, choosing the matrices Σ_k in (6.15f). It is well known that scaling can have a significant impact on the practical performance of trust-region algorithms.

In trust-region algorithms for unconstrained problems, the subproblems usually have the following form:

$$\min_{\Delta z} \nabla f(z)^T \Delta z + \frac{1}{2} \Delta z^T H \Delta z, \quad \text{subject to} \quad \|D\Delta z\| \leq \Delta. \tag{6.48}$$

Often, D is chosen to be a diagonal matrix whose diagonal elements are related to the diagonals of H. When the trust-region constraint is a Euclidean norm, the optimality conditions for (6.48) yield

$$(H + \xi D^T D)\Delta z = -\nabla f(z), \tag{6.49}$$

for some Lagrange multiplier $\xi \geq 0$. Hence, a common choice is to make each diagonal of D the square root of the corresponding diagonal of H. Similar choices of D are appropriate for other norms, as we discuss at the end of this section.

Motivated by this choice, we use the constraints in the our SQP subproblem to eliminate all but the input variables, and obtain a subproblem of the form (6.48) in which Δz is made up of $\Delta u_0, \Delta u_1, \ldots, \Delta u_{N-1}$. We then base the scaling matrix on the diagonal blocks of the Hessian in this subproblem.

For simplicity, we work with a special case of (6.11) in which neither the input constraints $Du_k \leq d$ nor the soft state constraints are present. (Our derivation is exactly the same as in the general case, but less cluttered.) To be specific, our MPC problem is as follows:

$$\min_{x,u,\eta} \Phi(x,u,\eta) \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} \left[\mathcal{C}(x_k,u_k) + \Xi(\eta_k)\right] + \Gamma(x_N) + \Xi(\eta_N) \quad \text{s.t.} \tag{6.50a}$$

$$x_0 \text{ given}, \quad x_{k+1} = F(x_k,u_k), \quad k = 0,1,\ldots,N-1. \tag{6.50b}$$

With a Euclidean-norm trust region, the corresponding SQP subproblem is

$$\min_{\Delta x,\Delta u} \tfrac{1}{2}\Delta u_0^T \tilde{R}_0 \Delta u_0 + r_0^T \Delta u_0 + \tfrac{1}{2}\Delta x_N^T \tilde{Q}_N \Delta x_N + q_N^T \Delta x_N + \sum_{k=1}^{N-1} \left\{ \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \right\} \tag{6.51}$$

subject to

$$\Delta x_0 = 0, \tag{6.52a}$$

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k, \quad k = 0,1,\ldots,N-1, \tag{6.52b}$$

$$\|\Sigma_k \Delta u_k\|_2 \leq \Delta, \quad k = 0,1,\ldots,N-1. \tag{6.52c}$$

We aggregate the variables in this subproblem as follows:

$$\Delta x = (\Delta x_1, \Delta x_2, \ldots, \Delta x_N), \quad \Delta u = (\Delta u_0, \Delta u_1, \ldots, \Delta u_{N-1}),$$

and also aggregate the data matrices as follows:

$$A = \begin{bmatrix} -I & & & & \\ A_1 & -I & & & \\ & A_2 & -I & & \\ & & \ddots & \ddots & \\ & & & A_{N-1} & -I \end{bmatrix}, \quad B = \begin{bmatrix} B_0 & & & \\ & B_1 & & \\ & & \ddots & \\ & & & B_{N-1} \end{bmatrix}, \tag{6.53a}$$

$$M = \begin{bmatrix} 0 & & & & \\ \tilde{M}_1 & 0 & & & \\ 0 & \tilde{M}_2 & 0 & & \\ \vdots & & & \ddots & \\ 0 & 0 & \dots & \tilde{M}_{N-1} & 0 \end{bmatrix}, \quad \tilde{Q} = \begin{bmatrix} \tilde{Q}_1 & & & \\ & \tilde{Q}_2 & & \\ & & \ddots & \\ & & & \tilde{Q}_N \end{bmatrix}, \tag{6.53b}$$

$$\Sigma = \begin{bmatrix} \Sigma_0 & & & \\ & \Sigma_1 & & \\ & & \ddots & \\ & & & \Sigma_{N-1} \end{bmatrix}, \quad \tilde{R} = \begin{bmatrix} \tilde{R}_0 & & & \\ & \tilde{R}_1 & & \\ & & \ddots & \\ & & & \tilde{R}_{N-1} \end{bmatrix}, \tag{6.53c}$$

$$q = (q_1, q_2, \dots, q_N), \quad r = (r_0, r_1, \dots, r_{N-1}). \tag{6.53d}$$

Using this notation, the objective (6.51) can be written in a more compact form:

$$\frac{1}{2} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix}^T \begin{bmatrix} \tilde{Q} & \tilde{M}^T \\ \tilde{M} & \tilde{R} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix}^T \begin{bmatrix} q \\ r \end{bmatrix}, \tag{6.54}$$

as can the constraints (6.52b):

$$A\Delta x + B\Delta u = 0. \tag{6.55}$$

Since A is square and nonsingular, we can use (6.55) to eliminate Δx, and write the objective (6.54) in terms of Δu alone, as follows:

$$\frac{1}{2}\Delta u^T \hat{Q} \Delta u + \hat{r}^T \Delta u, \tag{6.56}$$

where

$$\hat{Q} = \tilde{R} + B^T A^{-T} \tilde{Q} A^{-1} B - (\tilde{M} A^{-1} B + B^T A^{-T} \tilde{M}^T)$$
$$\hat{r} = r - B^T A^{-T} q.$$

Following the motivation above, we choose Σ to be a block-diagonal matrix such that the diagonal blocks of $\Sigma^T \Sigma = \Sigma^2$ are identical to those of $\hat{Q}$. Since from (6.53), A is block lower triangular and B is block diagonal, A^{-1} and $A^{-1}B$ are both block lower triangular. From the structure of $\tilde{M}$ in (6.53b), we have that the diagonal blocks of $(\tilde{M} A^{-1} B + B^T A^{-T} \tilde{M}^T)$ are zero. Hence, the diagonal blocks of $\hat{Q}$ are simply those of $\tilde{R} + B^T A^{-T} \tilde{Q} A^{-1} B$, which are as follows:

$$\begin{aligned}
\hat{Q}_{11} &= \tilde{R}_0 + B_0^T \tilde{Q}_1 B_0 + B_0^T A_1^T \tilde{Q}_2 A_1 B_0 + \cdots + B_0^T A_1^T \ldots A_{N-1}^T \tilde{Q}_N A_{N-1} \ldots A_1 B_0, \\
\hat{Q}_{22} &= \tilde{R}_1 + B_1^T \tilde{Q}_2 B_1 + B_1^T A_2^T \tilde{Q}_3 A_2 B_1 + \cdots + B_1^T A_2^T \ldots A_{N-1}^T \tilde{Q}_N A_{N-1} \ldots A_2 B_1, \\
&\vdots \\
\hat{Q}_{NN} &= \tilde{R}_{N-1} + B_{N-1}^T \tilde{Q}_N B_{N-1}.
\end{aligned}$$

Therefore, we can construct the scaling matrices for the Euclidean-norm trust region by the following recursive relationship: Define $\mathcal{G}_N = \tilde{Q}_N$, and then apply the following formula for $k = N, N-1, \ldots, 2$:

$$\mathcal{G}_{k-1} = \tilde{Q}_{k-1} + A_{k-1}^T \mathcal{G}_k A_{k-1}.$$

Then we have

$$\hat{Q}_{kk} = \tilde{R}_{k-1} + B_{k-1}^T \mathcal{G}_k B_{k-1}, \quad k = 1, 2, \ldots, N,$$

so our choice for the scaling matrices is

$$\Sigma_{k-1} = \hat{Q}_{kk}^{1/2}, \quad k = 1, 2, \ldots, N. \tag{6.57}$$

In practice, the use of a 2-norm trust region introduces a nonlinear constraint into the subproblem, making it no longer a quadratic program. Rather than use the same scaling matrices Σ_k for the ∞-norm as for the 2-norm, we construct tangent planes to the ellipsoidal 2-norm constraint using an eigenvalue decomposition. For each k, we calculate the orthogonal V_k and positive diagonal Λ_k such that

$$\Sigma_k^T \Sigma_k = \Sigma_k^2 = V_k \Lambda_k^2 V_k^T.$$

We then define the ∞-norm constraint as follows:

$$\|\Lambda_k V_k^T \Delta u\|_\infty \leq \Delta, \quad k = 0, 1, \ldots, N-1.$$

This constraint defines the smallest multi-dimensional box that circumscribes the Euclidean-norm trust region.

6.8 Adapting FP-SQP to Nonlinear MHE

In Chapter 4, we developed a formulation for MHE problems. In this section, we adapt algorithm FP-SQP to solve the minimization in (4.48a). In this case, we soften the output disturbance and state constraints. Our algorithm is

1. Initialize with guesses $\{w_k\}$, ρ, and ρ_p, $Sw_k \leq s$.

2. Substitute $\{w_k\}$, ρ into (4.4b)- (4.4d) to yield $\{v_k\}$ and $\{x_k\}$. Determine cost Φ (4.4a).

3. Obtain linear approximations of state and output equations, as in (4.6a) and (4.6b).

4. Solve quadratic program using a trust region method, scaled as described in Section 6.8.1, with structured solver.

5. Perform the perturbation (see Section 6.8.2) and repeat Step 2 to yield the new cost $\bar{\Phi}$.

6. Check $\bar{\Phi} < \Phi$. If so, let $\Phi = \bar{\Phi}$ and go to Step 3. If not, shrink the trust region and go to Step 4. Repeat until convergence criteria are achieved.

As with the regulator, the quadratic subproblems solved in Step 4 are highly structured due to the causal nature of the model. This causality results in a special banded structure that can be solved using a structured interior point solver analogous to that presented by Rao, et al., for the linear MPC regulator [98]. Further discussion of the solution of these quadratic programs is presented in Chapter 5.

6.8.1 Trust Region Scaling

As before, it is not necessary to apply a trust region constraint to all the variables in the MHE problem. The states and output disturbances are determined uniquely by the input disturbance w_k and by ρ. For this reason, we need only constrain these variables. By similar arguments to Section 6.7, we can derive a trust region scaling matrix.

Therefore, we can construct the scaling matrices for the Euclidean-norm trust region by the following recursive relationship: Define $\mathcal{G}_j = C_j^T R_v C_j$, and then apply the following formula for $k = j, j-1, \ldots, j-N_e+1$:

$$\mathcal{G}_{k-1} = C_k^T R_v C_k + A_{k-1}^T \mathcal{G}_k A_{k-1}.$$

Then we have

$$\hat{Q}_{kk} = Q_w + G_{k-1}^T \mathcal{G}_k G_{k-1}, \quad k = j-N_e+1, j-N_e+2, \ldots, j.$$

For the trust region on the ρ term, we have

$$\hat{Q}_\rho = P_e + \mathcal{G}_{j-N_e}$$

so our choice for the scaling matrices is

$$\Sigma_{k-1} = \hat{Q}_{kk}^{1/2}, \quad k = j - N_e + 1, j - N_e + 2, \ldots, j \tag{6.58}$$

and

$$\Sigma_\rho = \hat{Q}_\rho^{1/2}.$$

We construct tangent planes to the ellipsoidal 2-norm constraint using the same eigenvalue decomposition as in Section 6.7.

6.8.2 Feasibility Perturbation

In this section we adapt a similar approach to finding a perturbation matrix (or set of matrices) that stabilize the open-loop system (A_k, G_k) for the purpose of feasibility perturbations. We showed in Section 6.5 how and why the perturbation matrix is necessary for the MPC regulator. We now present the analog of this approach for the MHE problem.

While the inputs are used to generate the states in the MPC regulator by perturbation, the MHE problem uses the input disturbances w_k to generate both the states and the output disturbances. Therefore, we seek a matrix K_k such that all $|\text{eig}(A_k + G_k K_k)| \leq 1$, and we will use this matrix as the perturbation matrix after solving each QP. Note that in the estimator, this approach is applicable only when input disturbances are present, either directly from the model or from adding an integrating disturbance model. Otherwise, the states and output disturbances are determined solely by ρ. By the same arguments as for the regulator in Section 6.5, we present this perturbation matrix.

$$K_k = -(Q_w + G_k^T \Pi_{k+1} G_k)^{-1} (G_k^T \Pi_{k+1} A_k) \tag{6.59}$$

in which

$$\Pi_j = C_j^T R_v C_j \tag{6.60}$$

and Π_k is updated by

$$\Pi_k = Ck^T R_v C_k + K_k^T Q_w K_k + (A_k + G_k K_k)^T \Pi_{k+1} (A_k + G_k K_k). \tag{6.61}$$

We then use the same approach as the regulator, and re-parameterize the input disturbances as

$$w_k = K_k x_k + \omega_k. \tag{6.62}$$

After solving each quadratic program, the resulting ω_k are used to regenerate (w_k, v_k, x_k) from the nonlinear model. This perturbation step is important in keeping the feasible solutions close to the solutions from the quadratic program. To account for constraints on w_k, the perturbed step is clipped in a way similar to (6.21).

6.9 Computational Results

In this section, we describe computational experience with Algorithm FP-SQP applied to nonlinear MPC. We describe in detail two examples that are typical of problems that arise in industrial practice, showing how the various mechanisms described in Sections 6.5, 6.6, and 6.7 contribute to the effectiveness of the approach. We then summarize computational experience with these two examples and three others.

Our first example involves a continuously stirred tank reactor (CSTR).

Example 6.9.1 *Consider a CSTR in which the exothermic reaction $A \longrightarrow B$ is taking place. The temperature of the reactor is reduced by adjusting the temperature*

q	*100 L/min*	E/R	*8750 K*
C_{Af}	*1 mol/L*	k_0	7.2×10^{10} min^{-1}
UA	5×10^4 $J/min \cdot K$	V	*100 L*
ρ	*1000 g/L*	T_f	*350 K*
C_p	*0.239 J/g · K*	ΔH	-5×10^4 J/mol

Table 6.1: Parameters for the CSTR Model of Example 6.9.1.

of the coolant fluid in a heat exchange coil inside the vessel. The goal is to determine the optimal trajectory of future coolant temperatures such that the system reaches a desired steady state.

The equations governing this system are as follows:

$$\dot{C}_A = \frac{q}{V}(C_{Af} - C_A) - k_0 \exp\left(-\frac{E}{RT}\right) C_A, \tag{6.63a}$$

$$\dot{T} = \frac{q}{V}(T_f - T) + \frac{(-\Delta H)}{\rho C_p} k_0 \exp\left(-\frac{E}{RT}\right) C_A + \frac{UA}{V\rho C_p}(T_c - T), \tag{6.63b}$$

where C_A and T describe the state of the system, T_c is the input, and the remaining quantities are parameters whose values, as presented in [38], are given in Table 6.1. The variable C_A is the concentration of species A in mol/L, T is the temperature of the reactor in K, and T_c is the temperature of the coolant.

Given target values $C_{A,\mathrm{target}}$ and T_{target} for the states and $T_{c,\mathrm{target}}$ for the input, we define the states and input for this system as follows:

$$x = \begin{bmatrix} C_A - C_{A,\mathrm{target}} \\ T - T_{\mathrm{target}} \end{bmatrix}, \quad u = T_c - T_{c,\mathrm{target}}.$$

We use the target values $C_{A,\mathrm{target}} = 0.5$ M, $T_{\mathrm{target}} = 350$ K, and $T_{c,\mathrm{target}} = 300$ K. The initial state of the system is $x_0 = (1.0, 350)^T$.

We obtain a discrete-time problem from this model by defining a sampling interval of $\Delta t = 0.05$ minutes, and performing numerical integration of the equations

(6.63) between sampling times using the LSODE code [40]. We define the components of the cost function as follows:

$$\mathcal{C}(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k, \quad \Gamma(x_N) = x_N^T P x_N \tag{6.64}$$

where $Q = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$ and $R = 2$. The matrix P is the Lyapunov penalty associated with the discrete linear quadratic regulator problem with penalties Q and R on the linearized system at the set point. It has the following value:

$$P = \begin{bmatrix} 99164.7 & 2104.17 \\ 2104.17 & 73.2818 \end{bmatrix}.$$

The prediction horizon N is 60 time steps, or 3 minutes. The input u is constrained so that the coolant temperature may not be colder than 230 K; that is, $T_c \geq 230$.

For this example, the choice of target operating point is open-loop unstable. For temperatures just above the set point, the plant ignites, converting most of the reactant to product, and releasing more heat than the coolant can remove. Although we do not explicitly impose a "soft" state constraint to discourage this situation, it is certainly undesirable to steer the system through an ignition.

We find from solving the formulation (6.11) that the problem has several local solutions, which can be identified by starting the algorithm from different starting points. Figure 6.1 shows the profiles for the input variable u at three selected local solutions, along with the final objective value corresponding to each profile. The trajectory with the lowest objective function is believed to be the global optimum, arrived at by using the initial guess procedure described in Section 6.3. The other two minima plotted here are much less desirable, as they result from ignition. The controller in both these cases saturates at the minimum coolant temperature (that is, T_c remains at its lower bound of 230 for several successive sampling times) but is incapable of preventing ignition. These solutions were obtained by adding random noise at the level of 10% to the initial guess used to obtain the global solution.

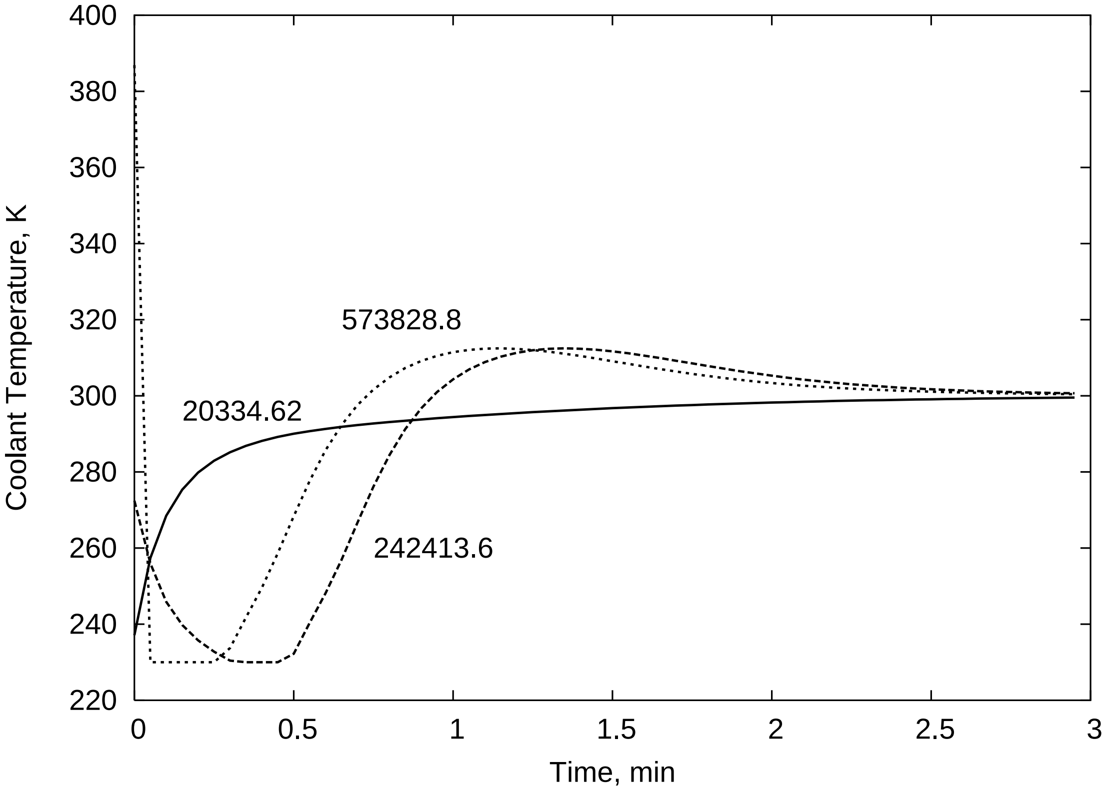

Figure 6.1: Input profiles and objective function values of local minima of CSTR example

An interesting property about the input profiles in the figure is that they all asymptotically converge to the steady-state input coolant temperature of 300 K. Each local solution still stabilizes the system, even though ignition occurs. Though ignition makes these solutions unacceptable in this case, the stabilization indicates that the controller is well designed. Local solutions may give acceptable control performance in other cases.

The existence of local solutions leads us to conclude that, as in other nonconvex optimization problems, the choice of starting point may be crucial to the quality of the computed solution. This observation has implications for the way we generate starting points for solving the MPC problem (6.11) at subsequent sampling times. The technique proposed in Section 6.3 of shifting the solution of (6.11) at the previous timepoint to obtain a starting point for the current timepoint may not be adequate. For instance, if $F(x, u)$ is not an adequate model of the true system, the state x_1 obtained by solving (6.11) at the previous timepoint may be distant from the estimate of x_0 obtained by applying a state estimation procedure at the current timepoint. Likewise, if the state and input targets change between sampling times, either due to a change in setpoint or as a result of rejecting non-zero mean disturbances, the shifting procedure may produce a poor initial guess. This could lead the algorithm to produce a potentially undesirable local optimum for (6.11). We can avoid these difficulties by repeating the initialization procedure in Section 6.3.

A key component in the formulation of Algorithm FP-SQP is the use of stabilization in the feasibility perturbation procedure, described in Section 6.5. Omission of stabilization can result in state trajectories $\widetilde{\Delta x}_k$ that diverge wildly from the SQP steps Δx_k. This phenomenon was noted for Example 6.9.1, as illustrated in Figure 6.2. This figure is a phase portrait, with the first component C_A and second component T of the system state plotted on the horizontal and vertical axes, respectively. From the initial guess indicated on the plot, the first iteration of Algorithm FP-SQP yielded a step Δx for the subproblem, where $x + \Delta x$ is labelled as "QP Solution" in the figure. When we recovered a feasible step $\widetilde{\Delta x}$ by means of the stabilized feasibility perturbation procedure described in Section 6.5, we obtained a perturbed step $\widetilde{\Delta x}$ that is quite close in this phase portrait to the original SQP step; see the curve labelled "Stabilized Projection." If, however, we omit the stabilization (setting $K_k = 0$ for all k), we obtain a perturbed step $\widetilde{\Delta x}$ that is distant from the original

SQP step; see the curve labelled "Naive Projection," for which some of the points are off the scale. In fact, the unstabilized perturbation predicts an ignition of the system. The unstabilized perturbation leads to poor algorithmic performance in this example—the Algorithm FP-SQP takes hundreds of iterations to converge. The trust region radius Δ must be shrunk to a very small value before an acceptable perturbed step is generated by the naive perturbation approach.

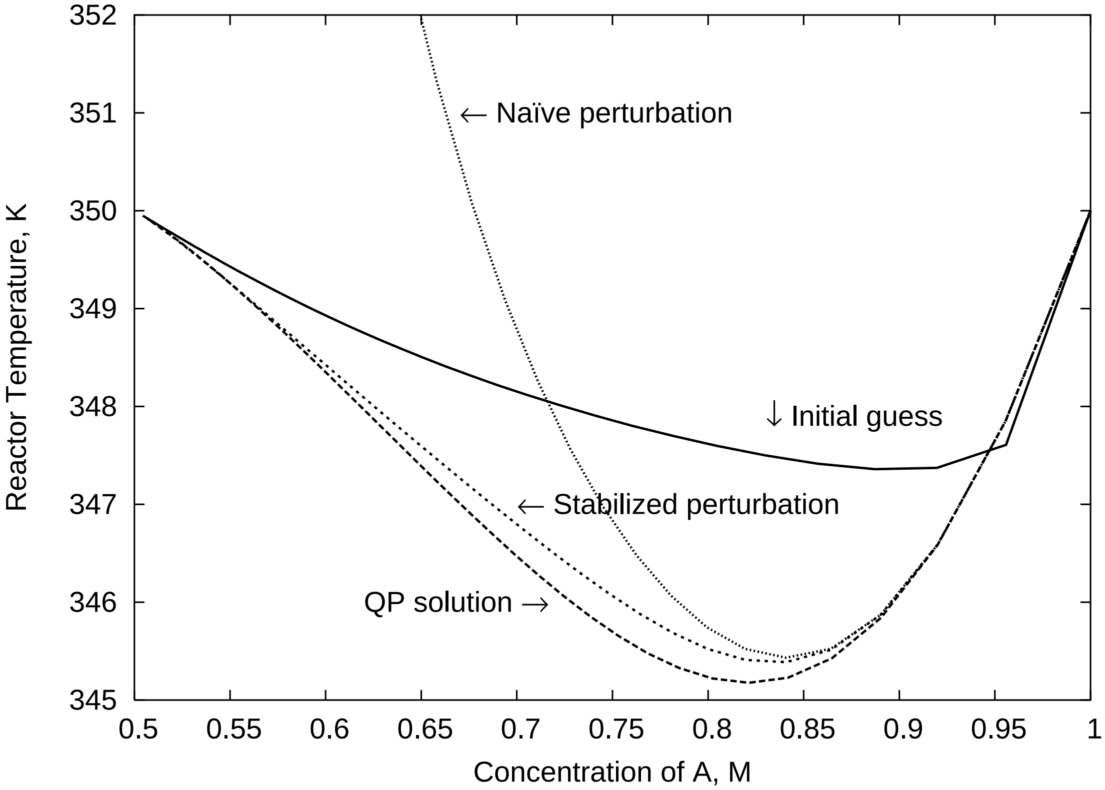

Figure 6.2: Phase portrait of states during the first iteration of the CSTR example

We now turn to our second example.

Example 6.9.2 *An electromagnetically actuated mass spring damper system is gov-*

erned by the equations

$$\dot{p} = v \tag{6.65}$$

$$\dot{v} = -\frac{k}{m}p - \frac{c}{m}v + \frac{\alpha}{m}\frac{C}{(d_0 - p)^{\gamma}} \tag{6.66}$$

where the states p and v represent position and velocity, respectively, and the input C is a function of the current applied to the coil (see [71]). The parameters in this model take on the following values: $\alpha = 4.5 \times 10^{-5}$, $\gamma = 1.99$, $c = 0.6590$, $k = 38.94$, $d_0 = 0.0102$, and $m = 1.54$. We apply the constraint $0 \leq C \leq 3$ to the input.

Given target values p_{target} and v_{target} for the states and C_{target} for the inputs, we define the state variable x and the input u as follows:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} p - p_{\text{target}} \\ v - v_{\text{target}} \end{bmatrix}, \quad u = C - C_{\text{target}}.$$

We used the target values $p_{\text{target}} = .0074$ and $v_{\text{target}} = 0$ (that is, we try to steer the mass to a specified position at rest); and the target input is $C_{\text{target}} = .0532$. The initial state of the system is $p_0 = 0$, $v_0 = .012$.

The sampling time is .01 and the prediction horizon is 100 time steps. We choose the penalty matrices Q, R, and P as in (6.64), to be

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1, \quad P = \begin{bmatrix} 18776.1 & 1746.93 \\ 1746.93 & 67.751 \end{bmatrix}.$$

To demonstrate the evolution from initial guess to final solution, we present in Figure 6.3 the input profile at each iteration of Algorithm FP-SQP, where the Hessian of the objective was used to approximate the Lagrangian Hessian, as in (6.37). Because all iterates of FP-SQP are feasible, we could use any of these iterates as a feasible suboptimal solution. In this case, Algorithm FP-SQP converges in 10 iterations, but the input profiles become almost indistinguishable after iteration 6.

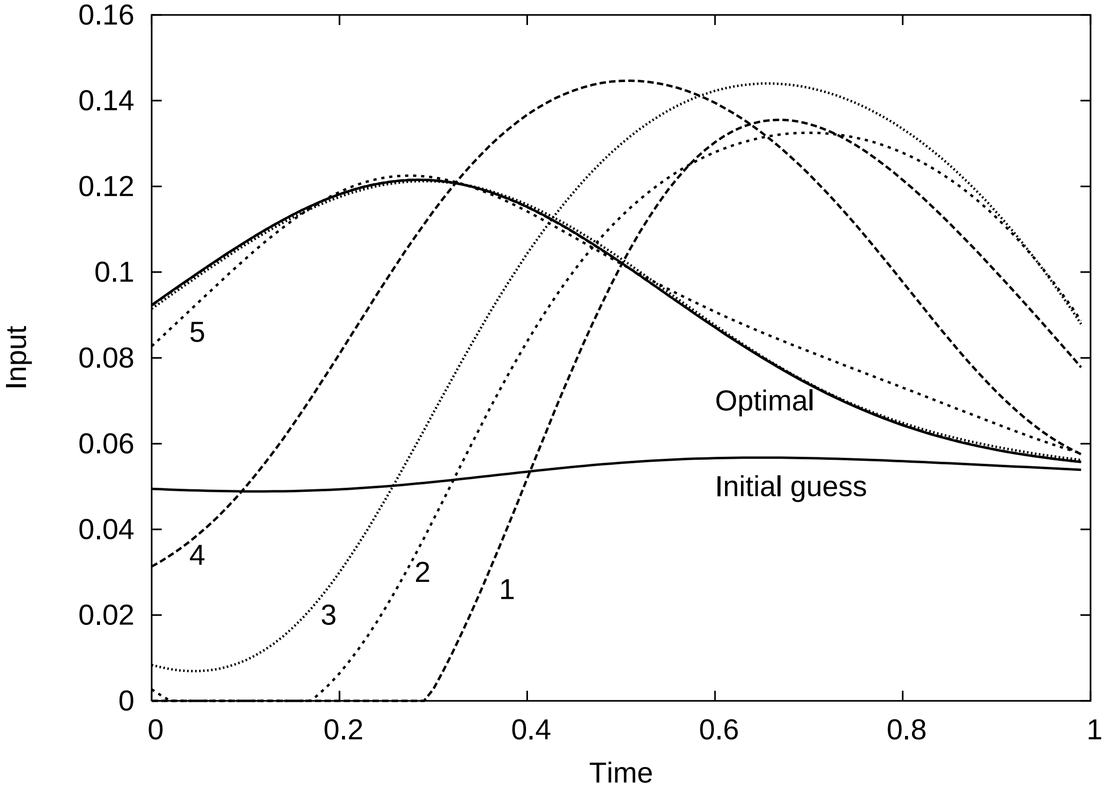

Figure 6.3: Input profiles at each iteration of Example 6.9.2

Returning to the feasibility perturbation scheme of Section 6.5, we show that the property (6.4) holds for the perturbation scheme presented here, when applied to Example 6.9.2. In Table 6.2, we tabulate the ratio

$$\|(\widetilde{\Delta x}, \widetilde{\Delta u}) - (\Delta x, \Delta u)\|_2 / \|(\Delta x, \Delta u)\|_2$$

at each iteration, where $(\Delta x, \Delta u)$ is the SQP step and $(\widetilde{\Delta x}, \widetilde{\Delta u})$ is the feasibility-perturbed SQP step. We also show the objective value at each iteration. From the table, it is clear that as the algorithm iterates, the discrepancy between the SQP step and its feasibility-perturbed variant vanishes as the iterates near the solution, so that (6.4) is satisfied for the perturbation scheme used here. Similar results were obtained on other examples.

Iteration	**Objective value**	$\frac{\Vert\Delta z - \widetilde{\Delta z}\Vert_2}{\Vert\Delta z\Vert_2}$
Initial	1.48171132254992	
1	1.03826539433288	7.980E-2
2	0.986956571853988	1.293E-2
3	0.896987789509686	4.476E-2
4	0.356871962432300	6.654E-1
5	0.217035426442347	2.300E-2
6	0.214667335503278	2.703E-3
7	0.214630294390683	2.399E-5
8	0.214629782597039	7.499E-7
9	0.214629774955025	8.200E-9
10	0.214629774925470	1.280E-9

Table 6.2: Asymptotic Exactness of Feasibility perturbation in Example 6.9.2

The Hessian of the Lagrangian in this example is indefinite, which indicates that the Hessian update schemes must take action to approximate this Hessian with a positive definite matrix. In the case of the BFGS approaches, the modified update mechanism is invoked more frequently. For the finite-difference-Hessian and the SR1 schemes, the resulting negative eigenvalues are set to zero, as described in Section 6.6, so that the subproblem may be solved by the convex QP solver of Chapter 5.

In Table 6.3, we compare variants of the Algorithm FP-SQP that use the various Hessian approximation strategies from Section 6.6. We also show results obtained with the commercial nonlinear optimization package NPSOL [35]. NPSOL is an SQP code that uses a BFGS approximation to the Hessian of the Lagrangian, as in Section 6.6.3. It uses dense linear algebra, and therefore is unable to take advantage of the structure of MPC problems. However, since the total number of variables in our examples is not particularly large (at most a few hundred), our problems can still be solved by this code in reasonable time. NPSOL does not restrict itself to feasible iterates and uses an augmented Lagrangian merit function to determine whether to accept steps. We applied NPSOL to the MPC problem in two ways. The first approach, denoted by NPSOLz in Table 6.3, refers to the application of NPSOL directly to the formulation (6.11). The second approach, denoted by NPSOLu, is the equivalent optimization problem that results from substituting the model equality constraints directly into the objective function, thereby eliminating the states x_k from the problem and yielding an optimization over only the input variables u_k and the constraint violations η_k.

Algorithm FP-SQP was implemented in Octave [26]. As mentioned earlier, the code LSODE was called from Octave to perform the numerical integrations between sampling times, necessary to obtain a discrete model. DDASAC [12] was used to determine the parametric sensitivities of the model equation with respect to their states and inputs. In the finite-difference-Hessian variant of FP-SQP (Section 6.6.1) finite differencing over the gradients calculated by DDASAC at perturbed points was used to obtain an approximate Lagrangian Hessian. Our platform was a 1.2 GHz AMD Athlon running Debian Linux. Initial points were calculated by the procedure described in Section 6.3. For Algorithm FP-SQP, these initial guesses were good enough that the trust region did not become active on any of the problems for which convergence was reported.

The parameter values used to initialize Algorithm 6.2.1 are $\eta = 0$ and $\Delta_0 = 1000$. For optimality, one of two tolerances must be satisfied. The absolute tolerance condtion requires $\|u^j - u^{j-1}\|_\infty \leq 1 \times 10^{-8}$. The relative tolerance is satisfied when $\|u^j - u^{j-1}\|_\infty / (\|u^{j-1}\|_\infty + \epsilon) \leq 1 \times 10^{-6}$, in which ϵ is defined as the smallest real number representable in floating point arithmentic.

We apply these codes to five test problems. The first two are Examples 6.9.1 and 6.9.2 described above. The third example, CSTR4, is a benchmark tank reactor model described in [17]. This system has four states, two inputs, and a prediction horizon of thirty samples. The fourth example, PEND, is the inverted pendulum system studied by Hauser and Osinga [37]. In this nonlinear model, a moving cart with an inverted pendulum must be steadied by appropriate changes to the velocity of the cart. The system has two states and one input, and the horizon length is thirty samples. The fifth example, COPOLY, is a copolymerization reaction and separation developed by Bindlish [6]. This model has fifteen states and three inputs, and the prediction horizon is twenty sample times.

Table 6.3 shows the number of iterations and the run times for NPSOL and Algorithm FP-SQP applied to each of the five examples. The run times contain limited information. Because Octave is a interpreted language, there is significant overhead reflected in the CPU times for the implementation of FP-SQP. Because the quasi-Newton approximation used in the NPSOL variants does not exploit the structure of the problem, its runtimes serve in part to show the importance of exploting this structure. Finally, because the sensitivity calculations are so expensive in our implementation, the variant of FP-SQP that used a finite-difference Hessian required an inordinate amount of time. The use of a more efficient sensitivity code, or evaluation of the gradients in parallel on a multiprocessor system, could be used to improve the runtime for this variant.

In normal operation, the initial guess to the optimizer would incorporate the solution to the NMPC problem at the previous time step, rather than the procedure described at the end of Section 6.3, which was used in Table 6.3. Barring large disturbances, the initial guess from the previous timepoint would usually be solved in fewer iterations than the iteration counts shown in Table 6.3. Since robust behavior in the presence of disturbance is a necessary property of a practical control system, however, the performance of the algorithm without prior information on the starting point is germane to our evaluation.

The results for NPSOL show that the original formulation (6.11) (the NPSOLz variant) is more robust than the one in which the states are eliminated for Example 6.9.1, because of the open-loop instability of this problem. We note that the variants of FP-SQP had little difficulty with this problem, indicating that our stabilized perturbation procedure described in Section 6.5 was effective. The NPSOLz variant generally has longer runtimes that NPSOLu, because the number of variables (and therefore the sizes of the dense matrices to be factored) is greater. More noteworthy are the observations that NPSOL fails altogether on Examples 6.9.1 and COPOLY, and takes appreciably more iterations than the FP-SQP variants on Examples 6.9.2 and PEND. We believe that the maintenance of feasibility in FP-SQP gives it an advantage in robustness over NPSOL.

Method	**Ex. 6.9.1**	**Ex. 6.9.2**	**CSTR4**	**PEND**	**COPOLY**
Finite-Difference Hessian	4 18.34	13 246.93	4 39.24	5 78.56	3 196.60
Objective Hessian	9 11.67	10 50.75	7 7.04	8 26.48	5 20.27
Partitioned BFGS	6 8.37	FAIL N/A	8 9.92	8 26.73	4 16.54
Sparsified BFGS	7 10.06	10 52.64	7 7.45	8 27.15	4 17.19
Paritioned SR1	6 8.07	12 60.78	7 8.86	11 36.62	4 16.13
NPSOLu	FAIL N/A	50 2279.73	3 16.30	12 127.53	FAIL N/A
NPSOLz	23 6783.61	>100 162989.12	4 4870.23	16 7834.80	FAIL N/A

Table 6.3: Comparison of iterations and CPU time to optimality tolerance

Table 6.3 also reveals that different variants of FP-SQP often take a similar number of iterations, with one notable exception. The partitioned BFGS procedure fails to converge on Example 6.9.2. This failure is a result of poor scaling of the Hessian approximation; the trust region remains inactive, but the steps become smaller and smaller, to the point at which the Hessian is so badly scaled that no further progress can be made. As noted earlier, the finite-difference Hessian variant is not competitive, because of the expense of the repeated sensitivity calculations with DDASAC needed to assemble the approximate Hessian.

In the application to MPC, both efficiency and robustness are necessary. The nonlinear programming algorithm should be efficient enough to find a reasonable approximate solution in the limited time available (the time interval between sampling points), while being robust enough to deal with possibly large disturbances in the system. Given the results of this section, we can draw the following conclusions.

- It is essential to exploit the stagewise structure of the problem.
- For models of the type used in our examples, which are typical of many industrial models, methods that require finite-difference calculation of Hessian approximations are generally too slow.
- Structured quasi-Newton schemes are frequently efficient, but sometimes fail in unpredictable ways.
- The simple scheme of using the objective Hessian as an approximation to the Lagrangian Hessian often works well, even when the objective Hessian is singular.
- An effective practical methodology may be to use the objective Hessian scheme

in concert with the structured quasi-Newton schemes, switching from one scheme to another when a failure occurs.

Although the algorithms seemed not to have much difficulty identifying the global solutions on these five examples, there may be situations in which we are led to a local solution, particularly when there is a model-plant mismatch, or after a disturbance. If additional time is available between sampling points, it may be well spent running these FP-SQP variants from a number of different initial guesses, and checking that the original computed minimizer is at least as good as the minimizers obtained from these speculative starting points.

To gain more insight into the behavior of the update schemes, Figure 6.4 shows the local convergence behavior of each algorithm on Example 6.9.1. In this plot, the relative error, defined as the ratio of the difference of the current and final objective to the final objective value, at iteration j (horizontal) is plotted against the relative error at iteration $j + 1$. Convergence rates can be inferred by determining the slope of the line near the solution (the lower left corner) in this log-log plot. A slope of 2 indicates quadratic convergence, while a slope of 1 indicates linear convergence, It is clear from the figure that the finite-difference Hessian variant yields a quadratic convergence rate. The variants based on the Hessian of the objective or the sparsified BFGS update produce nearly linear convergence. The partitioned BFGS variant is definitely superlinear, while the partitioned SR1 variant is slightly better than linear. The two partitioned quasi-Newton updates have somewhat jagged convergence profiles, however, possibly foreshadowing their occasional failure on other examples.

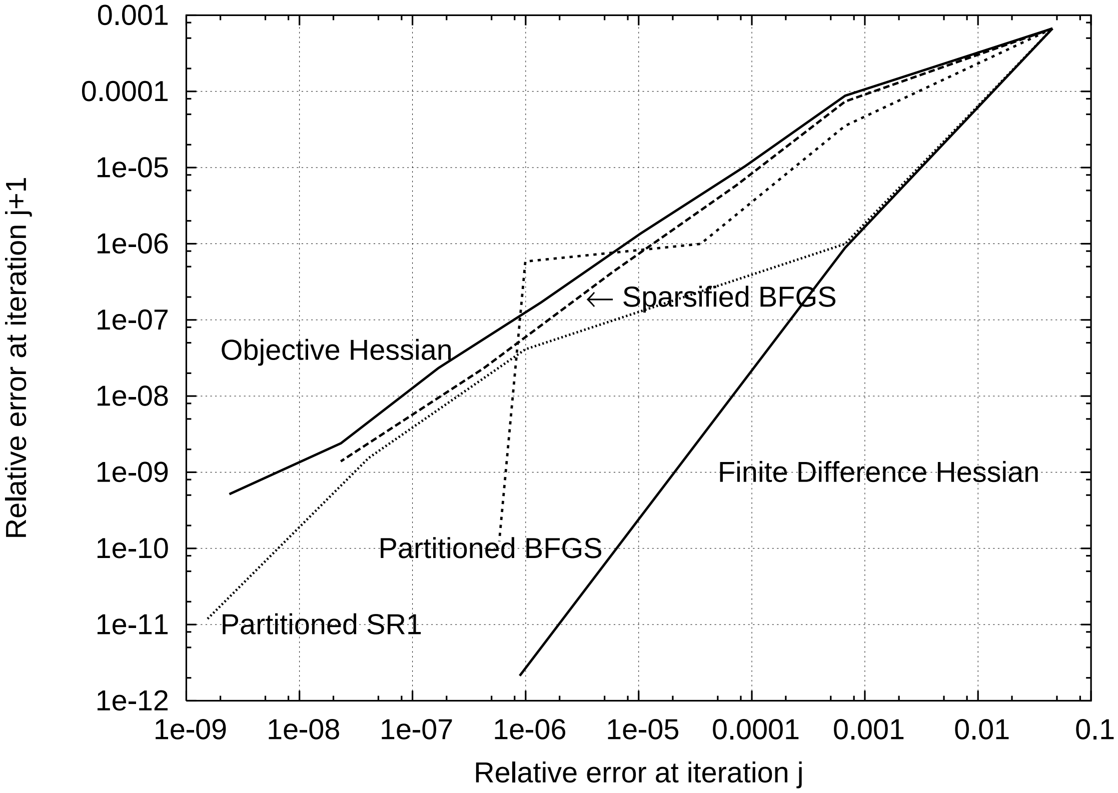

Figure 6.4: Comparison of convergence rates for Example 6.9.1

Chapter 7

Closed-loop Behavior of Nonlinear Model Predictive Control [1]

There is a theory which states that if ever anyone ever discovers exactly what the universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened.

— Douglas Adams

7.1 Introduction

In previous chapters, we presented the fundamental problem formulations and computational frameworks for the solution of the nonlinear regulation, estimation, and target calculation problems. We showed that the performance of each component for nonlinear systems was superior to other existing technologies, such as linear MPC and the extended Kalman filter. While these components perform well in isolation, the practical implementation of these parts necessarily involves interaction amongst

[1] **This chapter appears as Tenny, Rawlings, and Wright [117]**

the individual MPC elements. When the three components are connected to form a closed-loop system, the nonlinear model can exhibit different behavior than is possible using a linear model. Although the behavior of the closed-loop linear MPC system is well-studied, the behavior of the corresponding nonlinear system is not well characterized.

In this chapter, we consider the nonlinear model predictive control problem with moving horizon state estimation and a nonlinear steady-state target calculation. We focus on the aspects of the resulting closed-loop system that are specific to nonlinear models. One such issue is the effect of choosing integrating disturbance models to account for unmeasured nonzero mean disturbances. We demonstrate that the choice of disturbance models is crucial for a certain class of nonlinearity, and illustrate our findings with two simple examples. We compare our findings to the results of linear MPC on the same system.

The second part of this chapter concentrates on the appearance of local minima in the optimization problems in the MPC framework. Due to nonconvexity of the nonlinear process model constraints, multiple optima may be present for the regulator, estimator, and target calculation problems. We present examples of such local minima for the regulator and estimator problems and comment on their impact on closed-loop performance. We also motivate methods for avoiding undesired local minima by constraining the optimization problems. Also, in the case of the regulator problem, we show that an unnecessarily long prediction horizon may lead to local minima that are not globally optimal.

The remainder of this chapter is organized as follows. In Section 7.2, we present the framework for the nonlinear MPC closed-loop system. Section 7.3 highlights the behavior and potential pitfalls associated with choosing an integrating disturbance model. Section 7.4 investigates the effects of finding local minima of the nonlinear control and estimation problems of Section 7.2. We summarize our results and outline points of possible future investigation in Section 7.5.

7.2 Formulation

We consider a control system composed of three parts. The first part, the MPC regulator, is responsible for finding the best control profile given steady-state targets for the states and inputs. In some formulations, the regulator may instead be used to track a dynamic output trajectory. The second part, the state estimator, determines an approximate current state of the system, knowing the history of injected inputs and measured outputs. The state estimator is also used to estimate the integrating disturbance state. The final part is the steady-state target calculation, which adjusts the state and input targets to account for the integrated disturbance.

We begin by introducing the discrete time model

$$x_{k+1} = F(x_k, u_k + X_u p_k, w_k, t_k) \tag{7.1}$$

in which the index k represents the current sampling time, x_k is the state of the system, u_k is the input, w_k is the stochastic noise variable, and t_k is the time. We assume that w_k is normally distributed and has a zero mean. In our formulation, we enforce a zero-order hold (constant value between sampling times) on the u_k and w_k. The term $X_u p_k$ is the integrated input disturbance. In cases of plant/model mismatch or nonzero mean disturbances, this term is nonzero; however, in the nominal case in which the plant and model are identical, this term vanishes.

The discrete time model can be identified directly, or, more often, is the result of integrating a first-principles DAE model. In linear MPC, the function F is linear with respect to its arguments. In our study, however, F is any twice continuously differentiable function.

The outputs of the system are modeled as

$$y_k = g(x_k, t_k) + X_y p_k + v_k \tag{7.2}$$

in which y_k is the measurement at time k and v_k is a stochastic Gaussian zero-mean noise term. Again, g is assumed twice continuously differentiable. The term $X_y p_k$ is the integrating output disturbance. The integrating disturbance evolves as

$$p_{k+1} = p_k + \xi_k \tag{7.3}$$

in which ξ_k is a normally distributed zero-mean vector.

7.2.1 Regulation

To solve the regulation problem, we first assume that the stochastic variables w_k, v_k, and ξ_k take on their mean values. Since these means are zero, we note by (7.3) that the forecast of the integrating disturbance term p_k is constant at each sampling time. Suppose that the system is currently at time j.

Consider the following formulation of this N-step finite-horizon MPC problem:

$$\min_{x,u} \Gamma(\tilde{x}_{j+N}) + \sum_{k=j}^{j+N-1} \mathcal{L}(\tilde{x}_k, \tilde{u}_k) \tag{7.4a}$$

subject to:

$$x_{k+1} = F(x_k, u_k + X_u p_k, 0, t_k) \tag{7.4b}$$

$$Du_k \leq d, \qquad Gx_k \leq g \tag{7.4c}$$

in which x_0 is the current state estimate and x and u denote the sequences of vectors representing states and inputs, respectively; that is,

$$x = (x_{j+1}, x_{j+2}, \ldots, x_{j+N}),$$
$$u = (u_j, u_{j+1}, \ldots, u_{j+N-1}).$$

We define the deviation variables

$$\tilde{x}_k = x_k - x_{t,j}, \qquad \tilde{u}_k = u_k - u_{t,j} \tag{7.5}$$

in which the values $x_{t,j}$ and $u_{t,j}$ are the steady-state targets for the states and inputs at time j. The inequality constraints on the inputs and states may not yield a feasible solution for all problems, and for this reason, we may consider soft constraints on the states; see Chapter 6 for more details.

The stage cost that we consider here is

$$\mathcal{L}(\tilde{x}_k, \tilde{u}_k) = \tilde{x}_k^T Q \tilde{x}_k + \tilde{u}_k^T R \tilde{u}_k + (\tilde{u}_k - \tilde{u}_{k-1})^T S (\tilde{u}_k - \tilde{u}_{k-1}) \tag{7.6}$$

in which Q is positive semidefinite, and one of R and S is positive definite and the other is positive semidefinite.

We use the terminal penalty

$$\Gamma(\tilde{x}_{j+N}) = \tilde{x}_{j+N}^T P \tilde{x}_{j+N} \tag{7.7}$$

in which P is the solution to the discrete-time linear quadratic regulator problem for the linearized system at the current state and input targets [21]. For the case in which S is nonzero in (7.6), the terminal penalty is on the augmented state $[\tilde{x}_{j+N}; \tilde{u}_{j+N-1}]$.

The terminal state x_{j+N} is usually required to be inside some invariant region of the state target [70, 107]. Enforcing such a constraint is unnatural and impractical, however, as it is difficult to determine how long a horizon length N is required such that the terminal point may reach this set [45]. In practice, the horizon length is often chosen to be suitably long such that the terminal state x_N lies within the invariant region without explicitly constraining the state [116].

It is straightforward to test whether x_N satisfies the required properties of the invariant region; namely that a linear control law stabilizes the system after time N. We perform the following check:

Define

$$\tilde{u}_{j+N} = K\tilde{x}_{j+N}. \tag{7.8}$$

Provided $\tilde{u}_{j+N}$ does not lie on an active constraint, we calculate $\tilde{x}_{j+N+1}$ using (6.11b). If this new state does not violate any state constraints, we calculate the terminal stage cost decrease ratio

$$\alpha = \frac{\Gamma(\tilde{x}_{j+N}) - \Gamma(\tilde{x}_{j+N+1})}{\mathcal{L}(\tilde{x}_{j+N}, \tilde{u}_{j+N})}. \tag{7.9}$$

For linear systems, α will be exactly one. For nonlinear systems, any positive α value is acceptable. In cases in which this ratio is negative, or when constraints are active on the final inputs or states, the horizon length is not long enough. By satisfying the positivity of α, we are guaranteed nominal closed-loop stability.

The value of x_j comes from the state estimator. The first input u_j of the optimal input trajectory that results from the optimization problem (7.4) is injected into the plant.

7.2.2 Estimation

The control system obtains measurements from the plant at each sampling time. The goal of the state estimator is to determine the optimal approximation to the state evolution based on current and past inputs and measurements. The extended

Kalman filter (EKF) is a popular industrial choice for nonlinear models. Moving horizon estimation (MHE) has emerged as an alternative to the EKF [105, 70]. In recent work, moving horizon estimation has been shown to possess superior estimation properties compared to the EKF [115]. We consider the moving horizon estimation problem at time index j of

$$\min_{\rho,\rho_p,x,p,w,v,\xi} \quad \Gamma_e(\rho,\rho_p) + \sum_{k=j-N_e}^{j} \mathcal{L}_e(w_k, v_k, \xi_k) \tag{7.10a}$$

subject to:

$$x_{j-N_e} = \bar{x}_{j-N_e} + \rho \tag{7.10b}$$

$$p_{j-N_e} = \bar{p}_{j-N_e} + \rho_p \tag{7.10c}$$

$$x_{k+1} = F(x_k, u_k + X_u p_k, w_k, t_k) \tag{7.10d}$$

$$p_{k+1} = p_k + \xi_k \tag{7.10e}$$

$$y_k = g(x_k, t_k) + X_y p_k + v_k \tag{7.10f}$$

$$Hx_k \leq h, \qquad Sw_k \leq s, \qquad \Gamma v_k \leq \gamma \tag{7.10g}$$

in which the current output is denoted as y_j. The *a priori* estimate $\bar{x}_{j-N_e}$ is given in this formulation. The current state estimate x_j is the desired result that is used as the given initial state in the regulator. Also, the current integrated disturbance p_j is estimated and used in the target calculation.

The estimator stage cost is defined as

$$\mathcal{L}_e(w_k, v_k, \xi_k) = w_k^T Q_w w_k + v_k^T R_v v_k + \xi_k^T Q_\xi \xi_k \tag{7.11}$$

in which the penalty matrices Q_w, R_v, and Q_ξ are the inverse of the covariances of w, v, and ξ, respectively.

A few strengths of the MHE formulation are that constraints are incorporated into the framework, the resulting estimates are optimal and the method possesses excellent stability properties [97]. In the full-information problem, the horizon length N_e increases at each sampling time, tending toward infinity. In MHE, however, N_e grows to a specified horizon length N_T, at which point $\bar{x}_0$ and $\Gamma_e(\rho, \rho_p)$ are updated to reflect previous estimates of x_1. By selecting the arrival cost penalties appropriately, MHE approximates the full-information problem.

The arrival cost is approximated as

$$\Gamma_e(\rho, \rho_p) = \begin{bmatrix} \rho \\ \rho_p \end{bmatrix}^T P_{e,k} \begin{bmatrix} \rho \\ \rho_p \end{bmatrix} + p_{e,k}^T \begin{bmatrix} \rho \\ \rho_p \end{bmatrix} \tag{7.12}$$

in which the penalty matrices $P_{e,k}$ and $p_{e,k}$ and the *a priori* estimate $\bar{x}_{j-N_e}$ are updated according to the nonlinear smoothing covariance update given in Chapter 4, which is based on approximating the nonlinear system as a linear time-varying system. The smoothing update is based on the corresponding work of Rao et al. [96] for linear time invariant systems.

The initial values for the penalty matrices are

$$P_{e,0} = \begin{bmatrix} \Pi_0 & 0 \\ 0 & \Pi_{p,0} \end{bmatrix}, \qquad p_{e,0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7.13}$$

in which Π_0 represents the inverse of the initial covariance of the *a priori* state estimate and $\Pi_{p,0}$ is the inverse of the initial covariance of the *a priori* disturbance estimate. These matrices are not changed until the estimation horizon length reaches N_T, at which point the updating strategy is initialized.

7.2.3 Target Calculation

The goal of the target calculation is to find a steady state of the model that yields an output at the set point unless no such steady state exists. In circumstances where no steady-state targets can be found corresponding to the set point, we require the output target to be the closest output to the set point for which a steady state exists. If there are multiple steady-state inputs and states that satisfy the set point condition, then the state and input targets are selected as those that are nearest the previous input targets. At each time instant j, a new target must be calculated to account for the integrated disturbnce p_j. We formulate these notions as the following optimization problem, based on the linear case of Rao and Rawlings [95]:

$$\min_{x_{t,j},u_{t,j},\eta} \frac{1}{2}\eta^T \bar{Q}\eta + \bar{q}^T\eta + \frac{1}{2}\left(u_{t,j} - u_{t,j-1}\right)^T \bar{R}\left(u_{t,j} - u_{t,j-1}\right) \tag{7.14a}$$

subject to:

$$x_{t,j} = F(x_{t,j}, u_{t,j} + X_u p_j, 0, t_j) \tag{7.14b}$$

$$g(x_{t,j}, t_j) + X_y p_j - \eta \leq y_{\text{set}} \leq g(x_{t,j}, t_j) + X_y p_j + \eta \tag{7.14c}$$

$$Du_{t,j} \leq d, \qquad Hx_{t,j} \leq h, \qquad \eta \geq 0. \tag{7.14d}$$

In this nonlinear program, the variable η is a relaxation of the requirement that the state and input targets send the output to the set point when the set point is not feasible. Infeasibility of the set point occurs due to constraints or the dynamics of the nonlinear system. The vector $u_{t,j-1}$ is the input target from the previous sampling time. In general, $\bar{q}$ is chosen to be relatively large and strictly positive, and both $\bar{Q}$ and $\bar{R}$ are positive definite.

By shifting the state and input targets, the target calculation accounts for modelling error and adjusts the model to remove offset from the closed-loop system. In nonlinear MPC, it is possible that (7.14) does not have a solution. This problem may occur due to large disturbances or difficult constraints, such as tight bounds on the inputs. Infeasible target problems can also arise from large mismatch between plant and model. While relatively rare, failure of the target calculation is a serious issue that requires further study.

7.3 Disturbance Models

Integrating disturbances, as described earlier, are useful for removing offset in the output caused by unmodeled nonzero mean disturbances and plant/model mismatch. Provided the appropriate number of integrating disturbances have been added, the addition of integrating disturbances either removes offset or does not allow the closed-loop system to reach a stable operating point (see [84, 75] for details on the linear MPC case).

The effects of adding integrating disturbances are specific to the application, and determining where to add the integrating disturbance term for nonlinear models is an open question. The distribution of the integrating disturbance term among the inputs and outputs in nonlinear model predictive control is important since it may determine whether the closed-loop system reaches its target.

For cases of plant/model mismatch in linear MPC, Muske and Badgwell [75] recommend the input disturbance model. This choice is also justified by Shinskey [108], who notes that pure output disturbances are unlikely to occur in industry, but rather enter a system upstream of a dominant time constant, and usually at the manipulated variables. Morari and Lee [73] further note that DMC, which uses output disturbance models, does not quickly reject slow disturbances. Despite the preponderance of literature advocating input disturbance models for linear MPC, a study of the steady-state behavior of the plant is necessary before choosing a disturbance model for nonlinear MPC. We begin by examining the effects of disturbance models on two different systems.

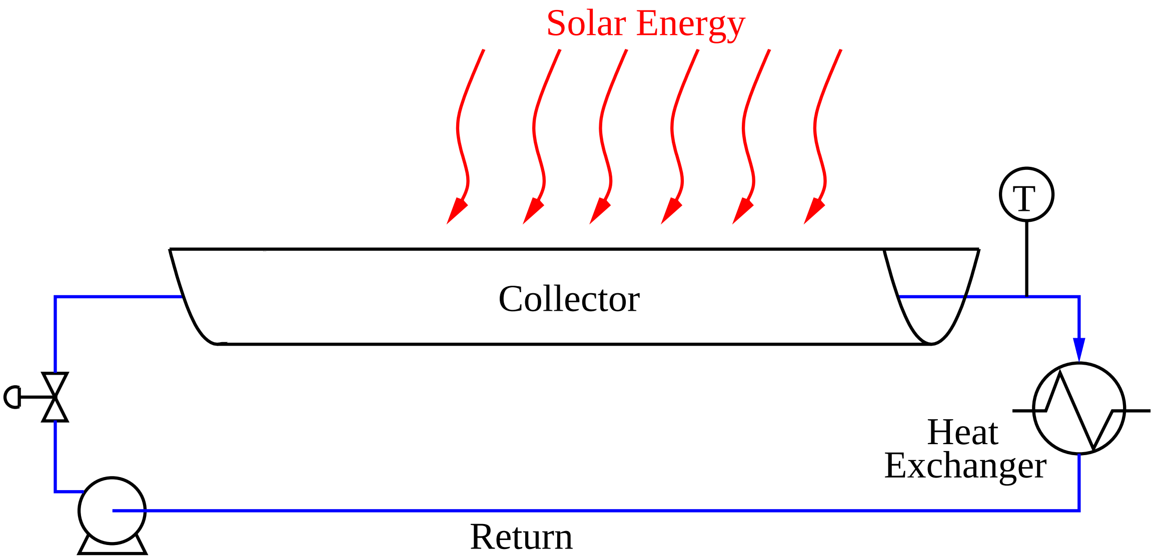

Figure 7.1: Schematic of solar collector

Example 7.3.1 *Consider a solar collector plant in which a fluid is used to absorb focussed solar energy along 790 meters of specially designed pipe [112, 111]. The fluid is circulated to a heat exchanger, where the energy is removed, and the cooled fluid is recirculated 790 meters back to the collector. The process is regulated by a pump that controls the flow rate of fluid through the system. A schematic of the solar collector is presented in Figure 7.1. Since the system is constructed to harness the sun's energy, the outlet temperature after the collector must be relatively high for the process to operate properly. In the case of this example, the desired outlet temperature is 543 K. Cloud cover, changes in ambient temperature, and other local weather effects act as unmeasured disturbances to the system, and must be rejected.*

The system is modeled by discretizing the collector and recycle loop spatially along their respective lengths. For ease of modeling, the temperatures at these spatial discretization points are states in the model, and are located close enough to each *other physically that to a good approximation*

$$\frac{\partial T}{\partial z} \approx \frac{T_i - T_{i-1}}{\Delta z}$$

in which T is the temperature at location z at which state i is measured. In this example, the temperatures along the length of the collector are modeled as the first twenty states, the next state models the temperature in the exchanger, and the final twenty states represent the temperature along the length of the recycle tube. The governing equations for this system are

$$\dot{T}_i = \alpha_i(F)(T_{i-1} - T_i) + \beta_i(T_{amb,i} - T_i) + \gamma_i \tag{7.15}$$

for $i = 1, 2, \ldots, 41$. For the case of $i = 1$, $T_{i-1} = T_{41}$. In this equation, the $\alpha_i(F)$ term represents the energy entering location i of the system from the previous location and leaves the current location by mass transport. In most cases, this term will depend on the mass flowrate F; however in the heat exchanger, we assume the exchange is fast enough that it does not depend on the flowrate of the circulating fluid. The term with β_i governs the energy exchanged between the system and the surroundings, such as the outside air for the collector, or the water in the heat exchanger. The γ_i term is the energy that enters the system from solar radiation. The values of the parameters for this model are listed in Table 7.1. Note that the input parameter F is included in $\alpha_i(F)$ for all states but the heat exchanger. Therefore, the $\alpha_i(F)$ terms are the source of nonlinearity in the system, since it multiplies the input with the states. Without disturbances, the steady-state input for the plant is 5.7 kg/s.

We examine the solar collector in Example 7.3.1 during an unmeasured 40% decrease in the solar power input (γ_i). Since this disturbance is unmodeled, an integrating disturbance model is added. We investigate the performance of the pure input

Parameter	Collector ($1 \leq i \leq 20$)	Exchanger ($i = 21$)	Recycle ($22 \leq i \leq 41$)
$\alpha_i(F)$ (s^{-1})	$(8.22 \times 10^{-3}\mathrm{kg}^{-1})\ F$	1.0	$(8.22 \times 10^{-3}\mathrm{kg}^{-1})\ F$
β_i (s^{-1})	1.19×10^{-3}	-5.0	0
γ_i (K/s)	5.41×10^{-1}	0	0
$T_{\mathrm{amb},i}$ (K)	303.15	375.15	——

Table 7.1: Parameters for the solar collector model for Example 7.3.1.

disturbance model ($X_u = 0.02$, $X_y = 0$) and pure output disturbance model ($X_u = 0$, $X_y = 1$) cases. We have chosen the relative scaling of X_u and X_y for input and output disturbance models such that the magnitudes of the integrating disturbances for both cases are similar. By maintaining the same penalties in the estimator, this disturbance tuning yields a fair comparison between disturbance models. The system has the following tunings for the regulator:

$$Q = I_{41}, \qquad R = 0, \qquad S = 1 \times 10^{-6}.$$

The input F is constrained to be between 0.8 and 8 kg/s. The system is sampled once per minute, and the prediction horizon is ten minutes. The estimator matrices are

$$Q_w = 0, \qquad R_v = 1 \times 10^6, \qquad \Pi_0 = (1 \times 10^6) I_{41},$$

$$Q_\xi = 1, \qquad \Pi_{p,0} = 1.$$

The system has forty-one states, and the output is the temperature at the outlet of the collector, which is the twentieth state.

The disturbance to the system occurs after five minutes, and since the solar input is unmeasured, the model and process no longer agree. The inputs and outputs of the system for the input and output disturbance models for both linear and non-linear MPC are shown in Figures 7.3 and 7.2. The model for linear MPC is obtained

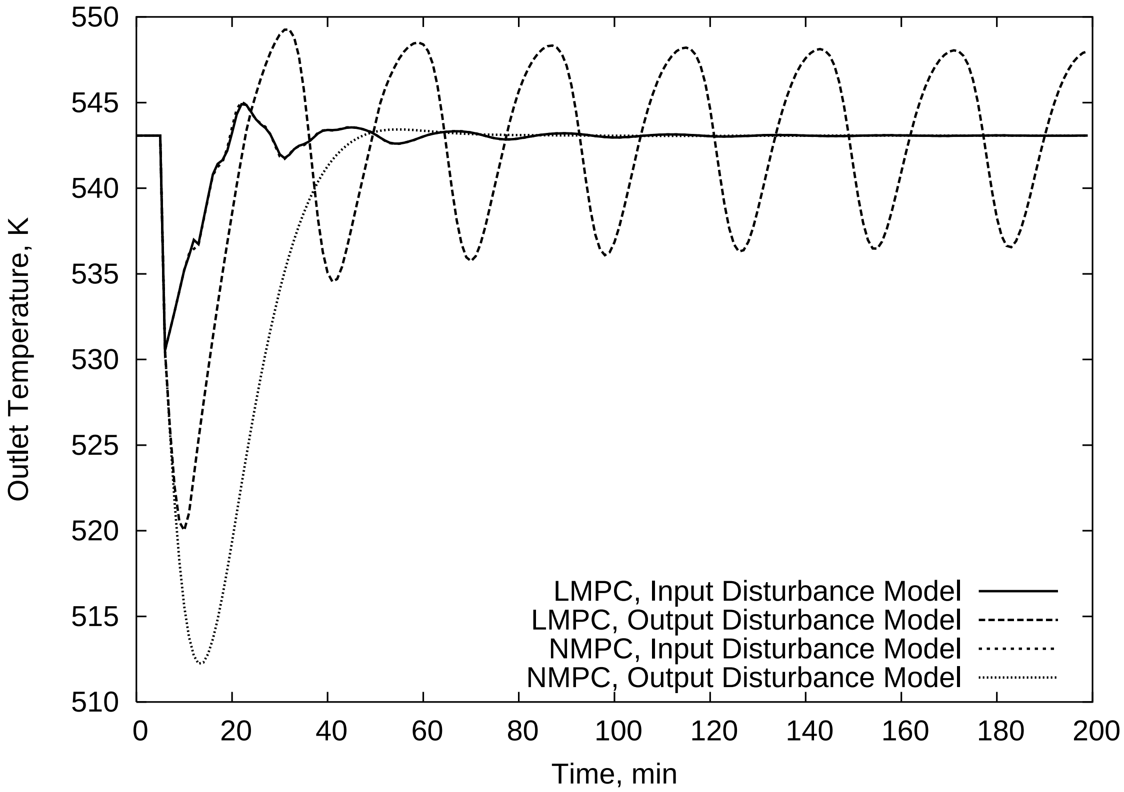

Figure 7.2: Disturbance model output performance for solar collector in Example 7.3.1

by linearizing the model about the original set point. We choose this linearization point because the process is originally stable at this point and the disturbance is unmeasured; it would be impossible to linearize the system at the final target since it is not known in advance.

It is evident from Figure 7.2 that the input disturbance model handles the sudden decrease in solar power more effectively than the output disturbance model for both linear and nonlinear models. In fact, the output disturbance model for the linear model does not steer the system to the set point within the observed time frame. The input disturbance models for linear and nonlinear MPC are nearly identical for this example. One reason the input disturbance model is better is that the modelled

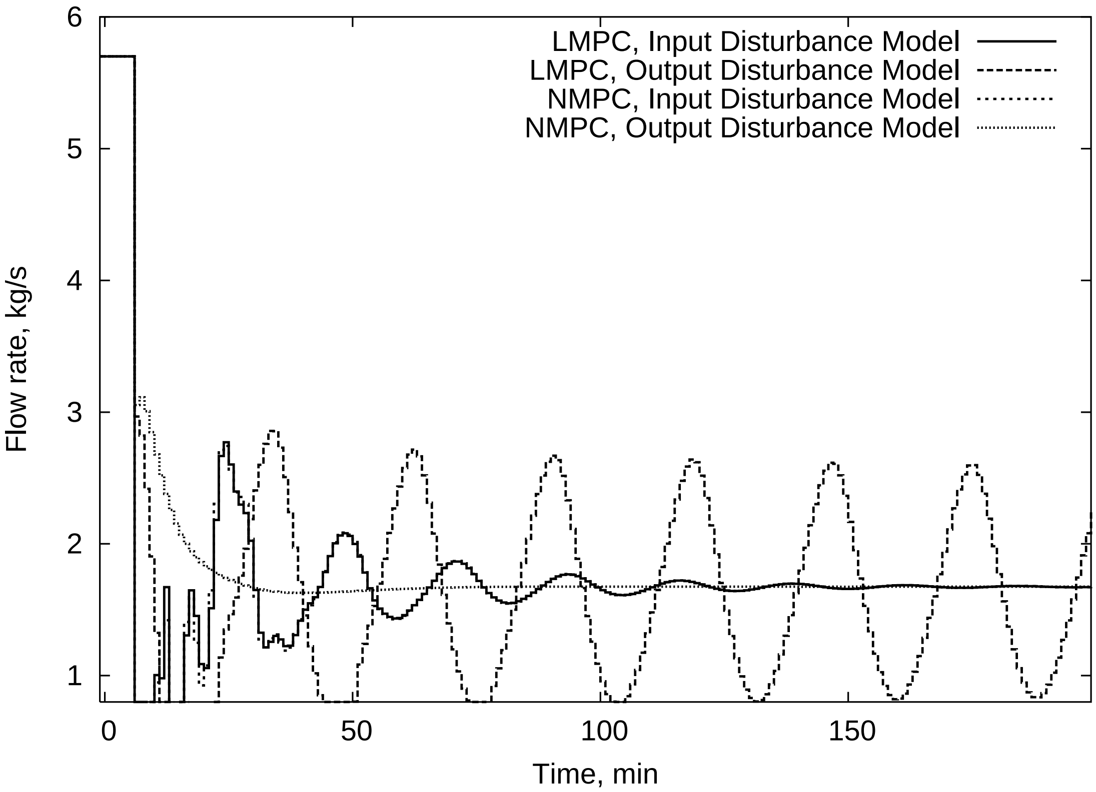

Figure 7.3: Disturbance model input performance for solar collector in Example 7.3.1

disturbance directly affects the states of the system, just as the true disturbance does. The output disturbance model, on the other hand, does not directly impact the states. Rather, it tries to find a new input and state target such that the original dynamics yield a system that accounts for the difference in observed and predicted outputs. In general, the input disturbance model is expected to perform better on systems with disturbances that alter the evolution of the states. We could conclude from this study that the input disturbance model is a better choice for mismodeled systems. However, such a conclusion would be premature, as we show in the following example.

Example 7.3.2 *Consider a continuously stirred tank reactor (CSTR) in which the irreversible reactions $A \to B \to C$ are taking place. The feed stream to the reactor is pure species A, and the maximum conversion to product B is desired. The concentration of the product B is measured and the process is regulated by adjusting the temperature of the reactor directly by a cascaded control system. The system is governed by the equations*

$$\dot{C}_A = \frac{F}{V}(C_{Af} - C_A) - k_1 C_A e^{-E_1/RT} \tag{7.16}$$

$$\dot{C}_B = k_1 C_A e^{-E_1/RT} - k_2 C_B e^{-E_2/RT} - \frac{F}{V} C_B \tag{7.17}$$

in which the concentrations C_A and C_B are the state variables and the temperature T is the manipulated variable. The values of the plant and model parameters are listed in Table 7.2.

In Example 7.3.2, the plant and the model do not agree since the activation energy in the plant is slightly lower than its model counterpart. This mismatch is realistic due to the difficulty in identifying accurate activation energies from experiments. In fact, the modeling error is relatively small; Figure 7.4 shows the locus of

F	100 L/min	C_{Af}	1 mol/L
V	100 L	E_1/R	8750 K
k_1	7.2×10^{10} min^{-1}	E_2/R (plant)	9700 K
k_2	5.2×10^{10} min^{-1}	E_2/R (model)	9750 K

Table 7.2: Parameters for the CSTR Model for Example 7.3.2.

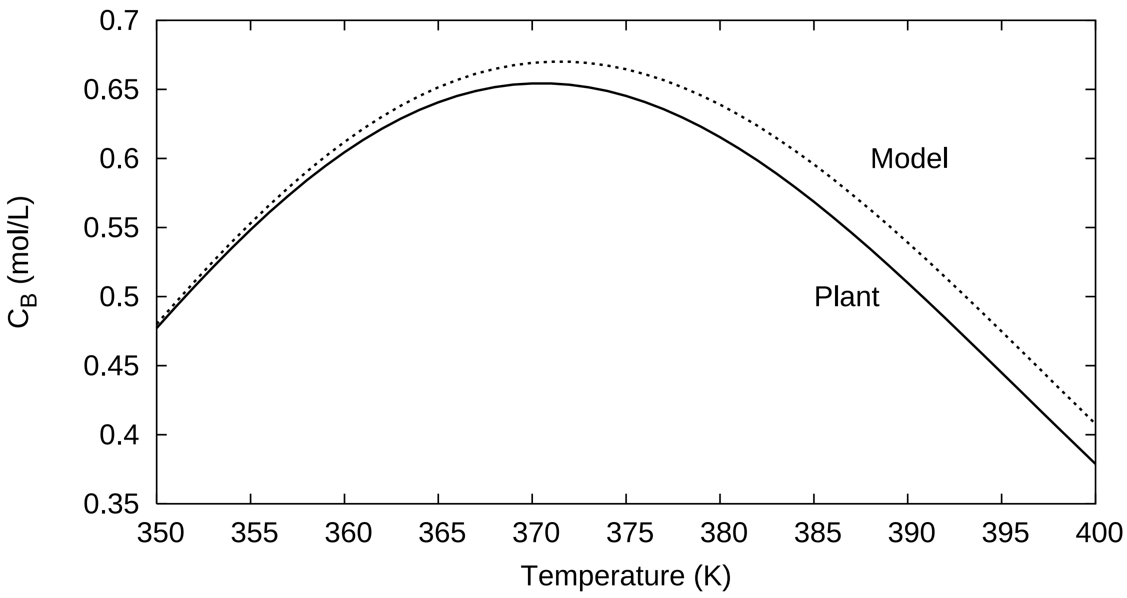

Figure 7.4: Locus of steady states of CSTR in Example 7.3.2

steady states for the model and the plant. These curves are similar in nature, with the exception that the maximum yield predicted by the model is higher than that of the plant. The model yield of species B is 0.670 mol/L, while the plant can achieve only 0.654 mol/L.

We now investigate the consequences of this small modelling discrepancy on closed-loop performance. We wish to operate the plant at its point of maximum yield. Therefore, the output set point in the target calculation is defined as the maximum yield of the model. Consider an MPC regulator with the penalty matrices

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 400 \end{bmatrix}, \qquad R = 2, \qquad S = 0.$$

a sampling time of 0.05 minutes, and a prediction horizon of 3 minutes. The estimator is tuned using

$$Q_w = 0, \qquad R_v = 1 \times 10^6, \qquad \Pi_0 = \begin{bmatrix} 1 \times 10^6 & 0 \\ 0 & 1 \times 10^6 \end{bmatrix},$$

$$Q_\xi = 1, \qquad \Pi_{p,0} = 1 \times 10^6,$$

and we begin by deciding what values of X_u and X_y to use. In the case of a pure input disturbance model, $X_u = 1$ and $X_y = 0$. For the output disturbance model $X_u = 0$ and $X_y = 1$. The closed-loop simulations for both cases are shown in Figures 7.5 and 7.6 for linear MPC and Figures 7.7 and 7.8 for nonlinear MPC. In these figures, the dashed lines represent the target values, while the solid lines represent the actual inputs and outputs of the plant. The linear model is obtained by linearizing the model at the original desired state and input targets.

The performance of linear MPC on this example is poor. The closed-loop system with the input disturbance model converts the reactant completely to the unwanted product C. The output disturbance model cannot determine an output target that recovers any appreciable yield of product B. Linear MPC is inappropriate for this system.

From examining Figures 7.7 and 7.8, it is clear that the output disturbance model in nonlinear MPC performs quite well on this example, bringing the closed-loop system to a steady-state value. The yield of the product B is not the maximum yield of the plant, but it is close. The output target has shifted down from its original unreachable value to a lower value that can be attained by the plant.

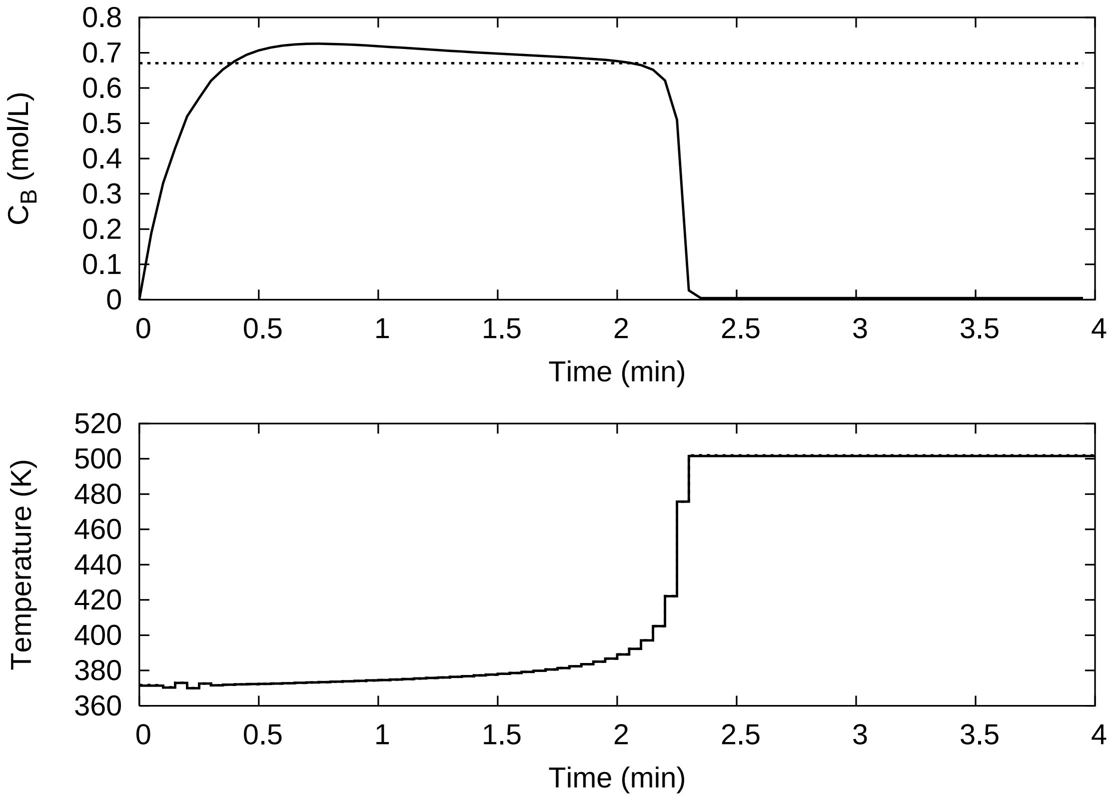

Figure 7.5: Closed-loop performance of input disturbance model on CSTR in Example 7.3.2 under linear MPC

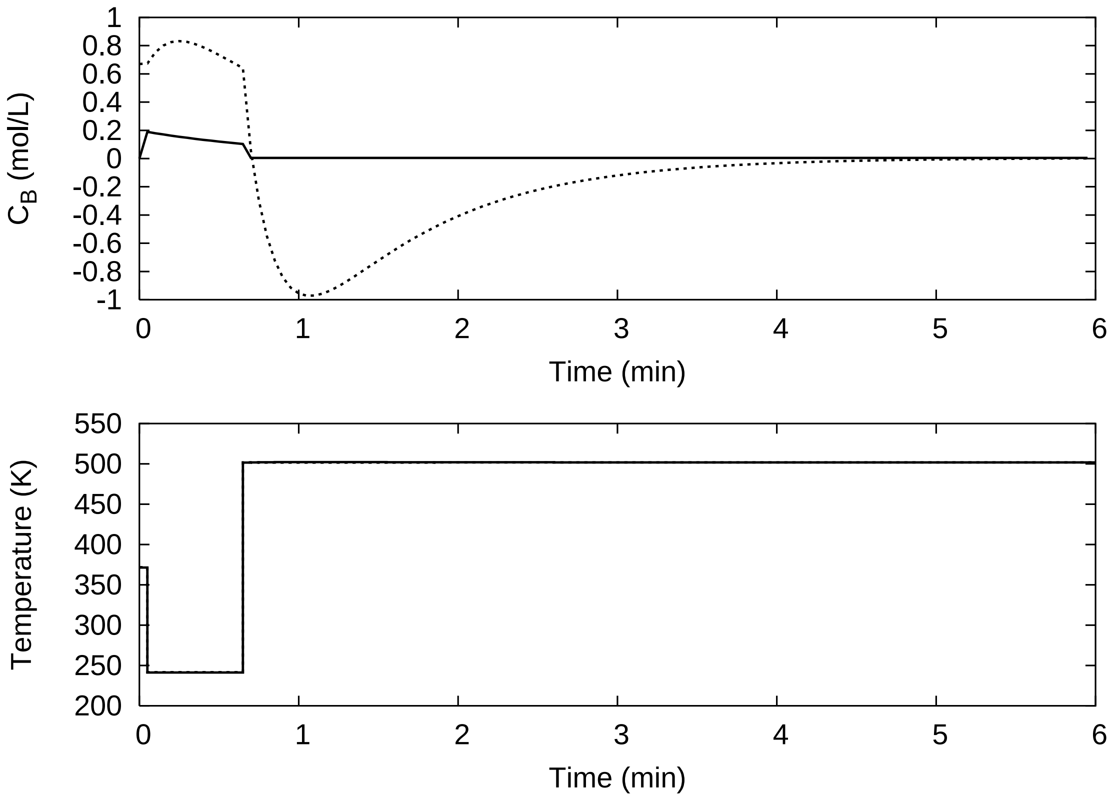

Figure 7.6: Closed-loop performance of output disturbance model on CSTR in Example 7.3.2 under linear MPC

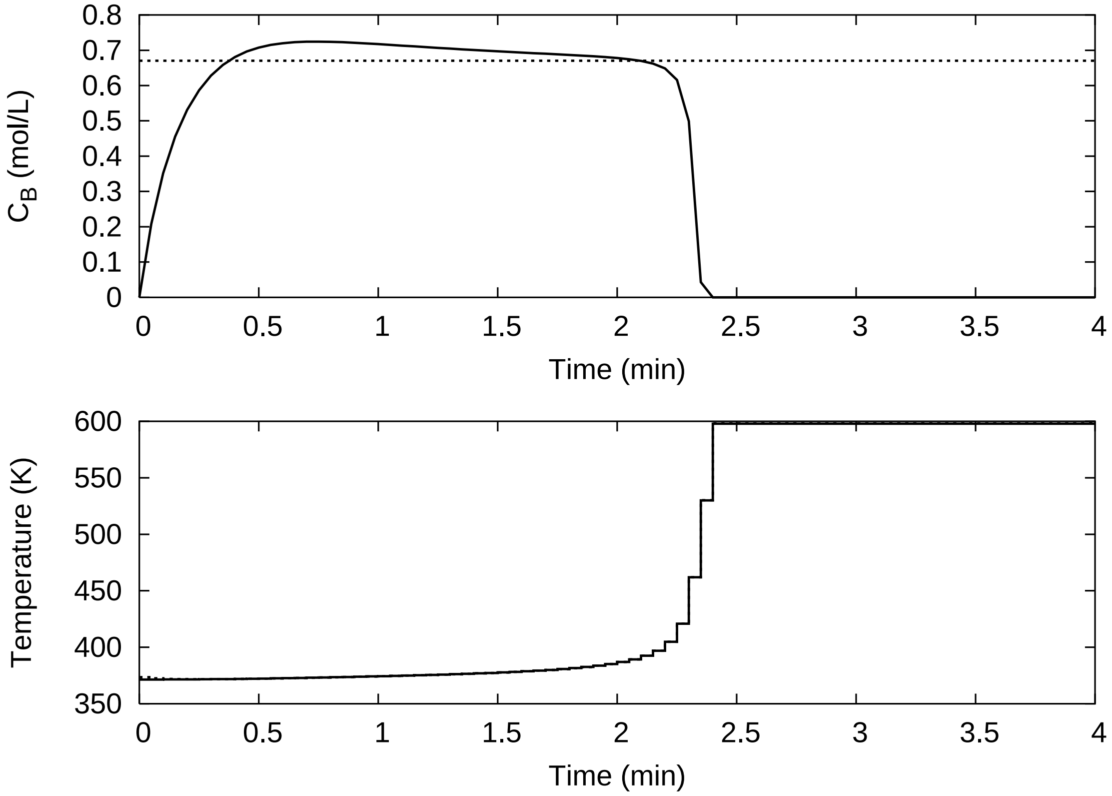

Figure 7.7: Closed-loop performance of input disturbance model on CSTR in Example 7.3.2 under nonlinear MPC

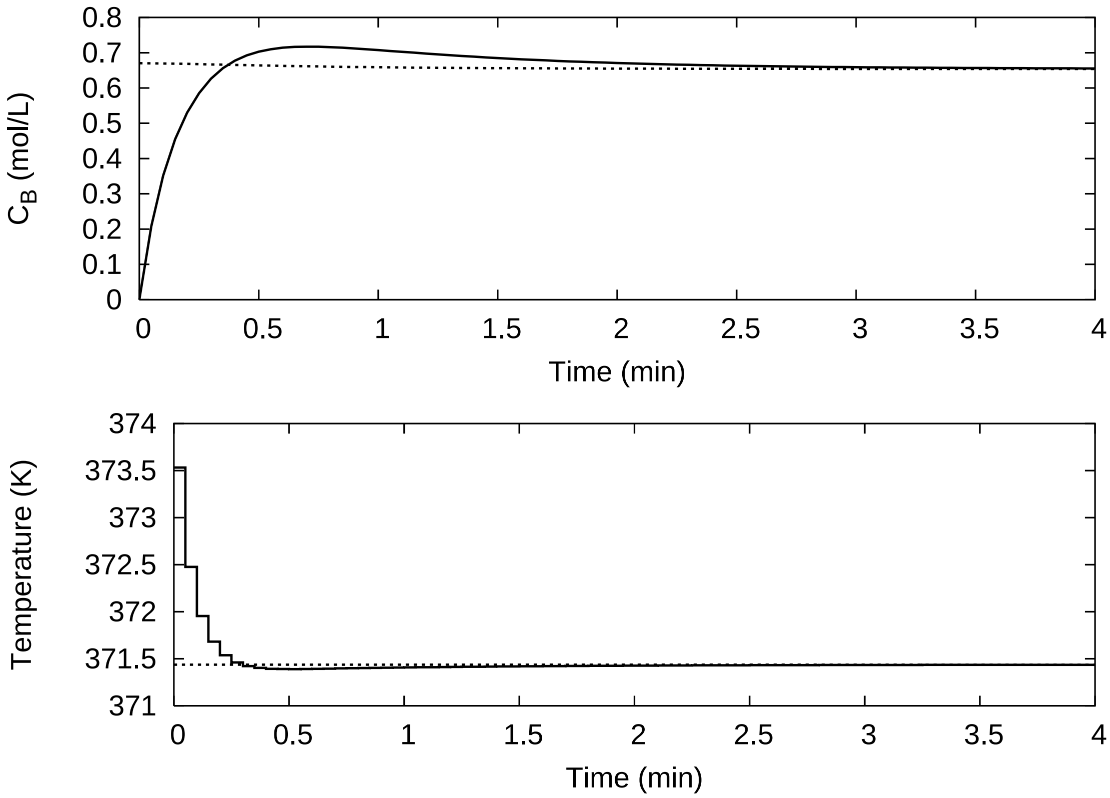

Figure 7.8: Closed-loop performance of output disturbance model on CSTR in Example 7.3.2 under nonlinear MPC

In the case of the input disturbance model, as with linear MPC, the system cannot be stabilized. While the input tracks the input target almost exactly, the target itself diverges. As a result, the reactor converts the reactant A completely to the undesired product C by increasing the temperature.

In this example, it is evident that an input disturbance model is not sufficient to control the system with plant/model mismatch. The effects of disturbance models on the MPC system must be explored to explain the cause of this phenomenon. Figure 7.9 is a graphical representation of the steady-state effects of adding an integrating disturbance to a system. The curves in the figure are the loci of steady states for the model and the plant. In this example, our aim is to maximize the output, so the target always resides at the maximum of the model curve. The input disturbance model, since it adds a bias term to the input, shifts the steady-state curve of the model horizontally in the figure. Similarly, the output disturbance model may adjust the model curve only in the vertical direction. For a closed-loop steady state to occur, the model curve, the target, and the plant curve must all meet at one point. When the set point and the model do not cross, the output target shifts to the steady state of the model nearest the set point. In this example, the desired steady state is at the maximum point of the model, so the target and model move together, intersecting only at this point. It is obvious that this intersection point cannot lie on the plant curve solely by moving horizontally, which explains why the input disturbance model cannot stabilize Example 7.3.2. On the other hand, when the model curve is allowed to move down to intersect the plant at the maximum of the model, as happens with the output disturbance model, the system can reach a steady state. Finally, we note that the value of the steady state reached by using a pure output disturbance model is predicted by finding the output value of the plant at which the input value of the

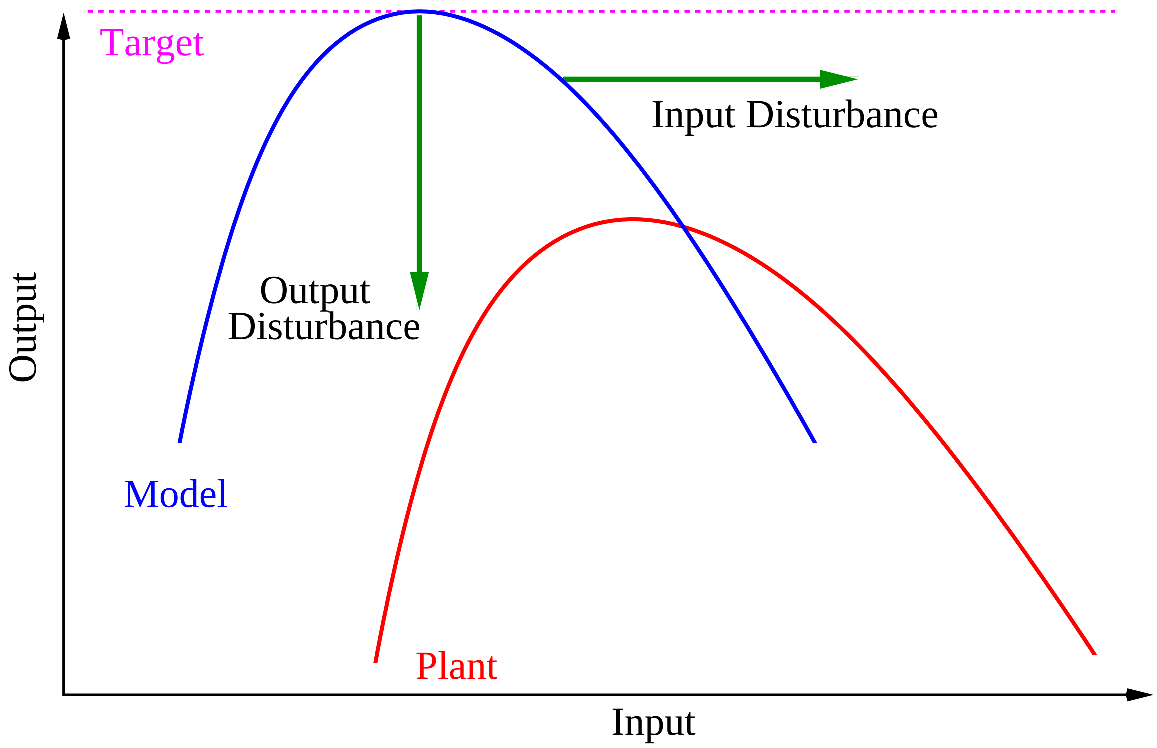

Figure 7.9: Effects of disturbance models on model steady states

model attains its maximum output.

The ill effects of using an input disturbance model in this example are due to two factors. First, the plant has a maximum value for the steady-state output, and second, when an infeasible set point is specified, the target calculation is incapable of finding an attainable output target. A possible solution to this problem, while still using an input disturbance model, is to not specify that the process operate at the maximum value, but rather choose an appropriately conservative operating point that is known to be a steady state. The location of such an operating point may not be available *a priori*, and is a limiting weakness of the input disturbance model. In general, the steady-state behavior as shown in Figure 7.9 dictates whether closed-loop stability is possible; whether the closed-loop system is actually stabilized, however, is determined by the combined dynamics of the estimator, regulator, and target calculation.

7.4 Local Minima

Because the model equations are nonlinear, the equality constraints in the optimization problems of the estimator, regulator, and target calculation may be noncovex, yielding local minima that are not globally optimal. This phenomenon is not observed with linear models because the linear constraints that represent the process dynamics are always convex. The local optima of the nonlinear programs may be undesirable due to physical considerations, or they may be unacceptable from a performance standpoint. Finding the appropriate locally optimal solution can often be achieved through appropriate initial guesses to the optimizer and the use of realistic constraints. We begin by discussing how local optima arise in the NMPC framework.

In related work, we describe the appearance of local minima for an exothermic stirred tank reactor (see Chapter 6). In that example, the local minima arise from varying the initial guesses to the optimization algorithm by adding noise to the initial guess for the inputs. At least seven local minima exist in the regulator optimization problem for the simple reactor example, all of which asymptotically reach the state and input targets. However, all but the globally optimal solution cause the process to ignite, converting nearly all the reactant to product at a high temperature.

Local minima are possible in the regulator, estimator, and target calculation problems when using nonlinear process models. The different solutions that result from the a nonlinear problem are a function of the initial guess to the optimization algorithm. For NMPC strategies that rely upon solutions from previous sample times to initialize the optimization problems, local solutions can appear in the regulator when the current state estimates are not close enough to the predicted states from the previous time step. This disagreement between the open-loop prediction of the regulator and the state estimate based on plant data can result from plant/model mismatch, inaccurate state estimates, unmeasured disturbances, or even system noise. In general, an educated guess for the initial starting point may often yield the global solution to the nonlinear program, but there are no guarantees. Occasionally, additional steps can be taken to prevent convergence to an undesired local solution. We present the following example to illustrate how local minima arise and how to avoid such solutions in a nonlinear MPC regulator.

Example 7.4.1 *An electromagnetically actuated mass spring damper system is governed by the equations*

$$\begin{aligned}\dot{p} &= v \\ \dot{v} &= -\frac{k}{m}p - \frac{c}{m}v + \frac{\alpha}{m}\frac{C}{(d_0 - p)^\gamma}\end{aligned}$$

where the states p and v represent position and velocity, respectively, and the input C is a function of the current applied to the coil (see [72]). The physical parameters of this model are: $\alpha = 4.5 \times 10^{-5}$, $\gamma = 1.99$, $c = 0.6590$, $k = 38.94$, $d_0 = 0.0102$, and $m = 1.54$. We apply the constraint $0 \leq C \leq 3$ to the input.

The goal is to steer the system such that it is at rest ($v = 0$) at the position $p = .0074$. The corresponding steady-state input for this position is $C = .0532$. The initial state of the system is $p_0 = 0$, $v_0 = .012$.

In Example 7.4.1, we use the initial guess to the optimizer as described in Chapter 6. The regulator has a sampling time of .01 and the following cost function weighting matrices:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1, \quad S = 0.$$

We begin by examining the case in which the prediction horizon for the regulator is 200 time steps, or 2 time units. The initial guess to the optimizer and the resulting open-loop optimal control profile is shown in Figure 7.10, and the evolution of the position state is displayed in Figure 7.11. The locally optimal input profile oscillates once before settling; we show later that this sequence of inputs is not globally optimal. Even though the initial guess is relatively good, the global solution to this regulation problem cannot be found as currently formulated.

We now concentrate on ways to force the system to the globally optimal profiles. In the same figures, we present the effects of using a shorter prediction horizon of 100 time steps. The periodic behavior of the input profile is not present with the shorter horizon because the horizon length is not long enough to complete a full period. This result runs counter to the commonly held belief that increasing the prediction horizon length can only improve controller performance. Here, a long prediction horizon can lead to one – or several – periods before approaching the set point. Because we use a terminal penalty to approximate the infinite horizon cost to go, decreasing the prediction horizon does not prevent the controller from having an accurate forecast of the infinite horizon cost. The shorter prediction horizon does prevent the controller from approaching the steady state by a path with unnecessary periodic behavior.

We also present the results of using the original horizon length of 200 time steps but constraining the velocity of the system to be nonnegative. These results are also in Figures 7.10 and 7.11. Note that this case also removes the periodic result and that the additional constraint, although in the formulation, is not active

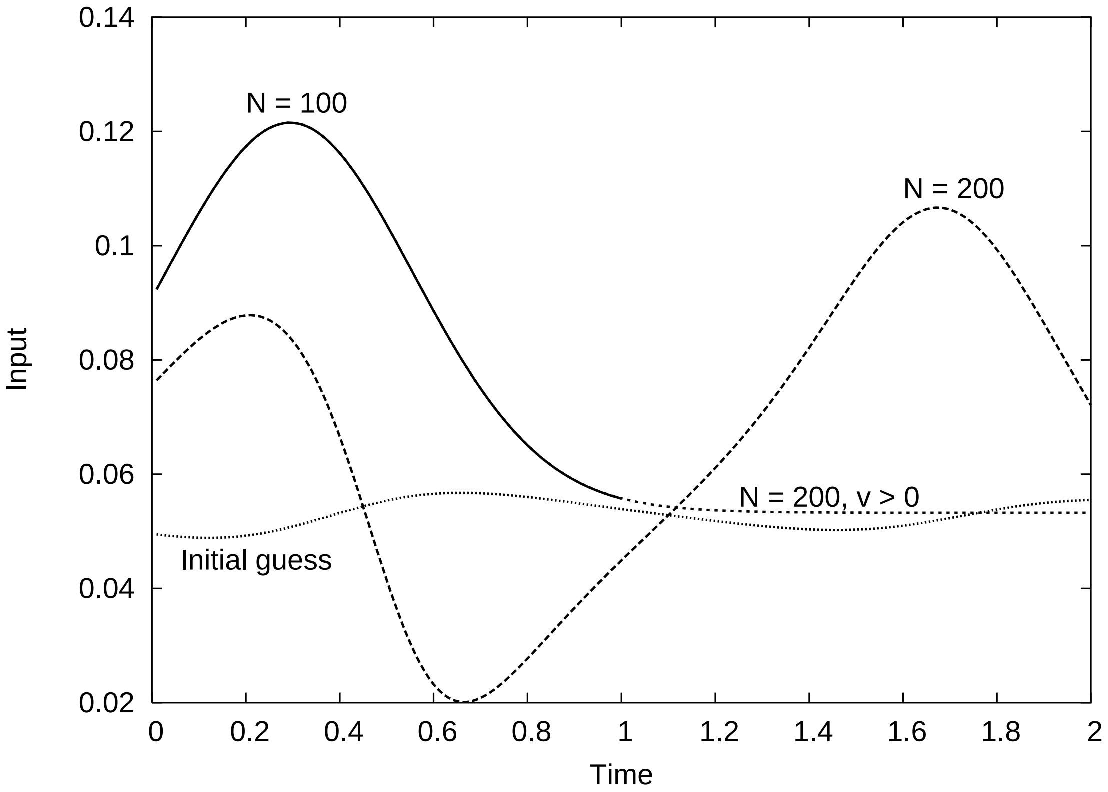

Figure 7.10: Optimal input profiles for spring system in Example 7.4.1

at the optimal solution. Even though the constraint is present in the formulation, the resulting solution is unconstrained. This behavior is a key difference between the use of linear and nonlinear models. In linear MPC, if the control trajectory changes when a constraint is added, then that constraint must be active at the solution. However, constraints in the nonlinear MPC problem may change the optimal solution, but need not be active at the solution if a new local minimum to the cost function is found. This behavior is one of the main differences between linear and nonlinear MPC.

The value of the cost function for the locally optimal solution with a horizon length of 200 time steps is 0.21985, while the cost function values for both the shorter horizon solution and the constrained result are 0.21463. Based solely on the value of the cost function, both these solutions are better than the solution found by the

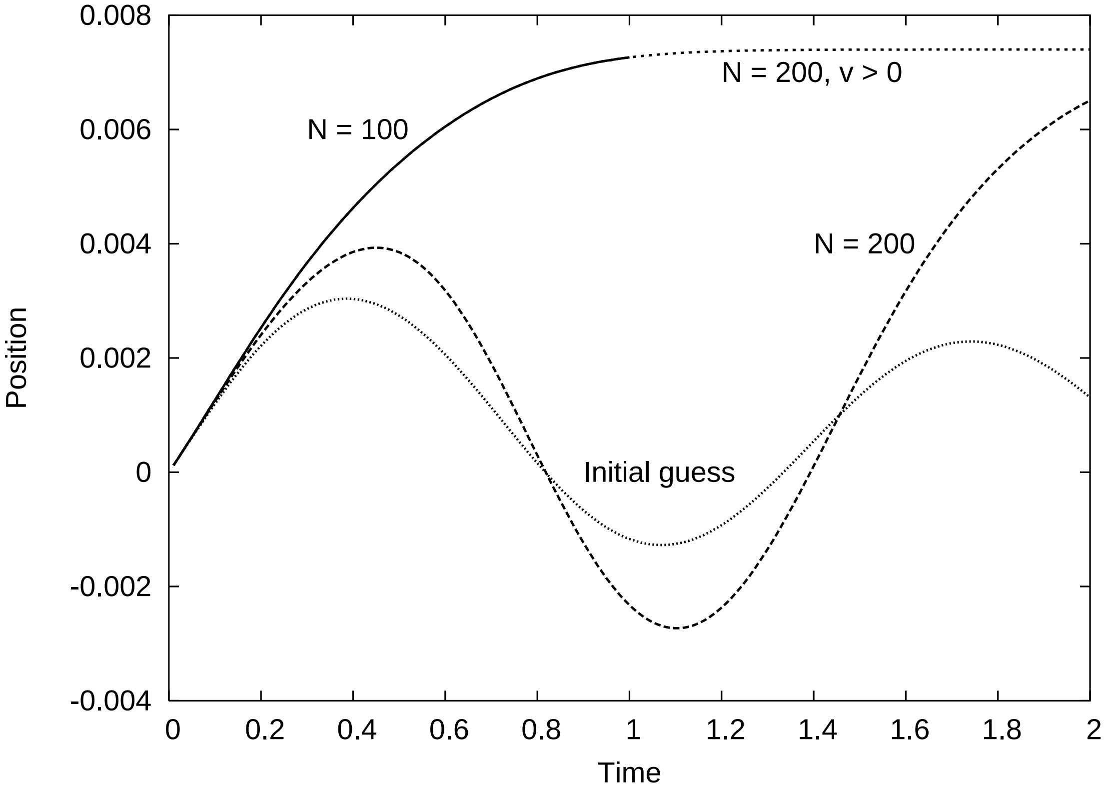

Figure 7.11: Optimal position profiles for spring system in Example 7.4.1

nominal formulation of N=200 without constraints. In the closed-loop sense, the locally optimal solutions are nearly equivalent since they both achieve the main goal of the controller – namely to stabilize the system at its set point. However, in cases in which the local optima take unwanted excursions, for instance via ignition, constraints should be placed on the regulator to prevent poor performance.

In practice, it may not be possible or practical to apply a constraint in the regulator. For instance, in the above example, the constraint is not added out of consideration for the true dynamics of the process, and if a disturbance were to occur, this constraint may prevent the controller from properly stabilizing the system. On the other hand, shortening the prediction horizon in the regulator is a generic solution that is applicable regardless of the process. For instance, the unwanted local minima in the CSTR system studied in Example 6.9.1 can be removed by sufficiently shortening the prediction horizon, while applying a constraint is not successful on all initial guesses. If a longer horizon is desired for performance reasons, then the result for the shorter horizon as described above can be extrapolated by a linear control law and used as an initial guess for the full horizon problem.

For the estimation problem, we revisit the batch gas reactor presented in Example 4.3.2.

Example 7.4.2 *An isothermal gas-phase reactor is charged with an initial amount of A and B, and the species are allowed to react according to the reversible reaction $2A \rightleftharpoons B$. The goal is to reconstruct the partial pressure of each species in the reactor during the reaction based on measurements of the total pressure of the vessel as the reaction proceeds. The system is modeled by*

$$\dot{x}_1 = -2k_1x_1^2 + 2k_2x_2, \qquad \dot{x}_2 = k_1x_1^2 - k_2x_2 \tag{7.18}$$

in which $k_1 = 0.16\ min^{-1}\ atm^{-1}$, $k_2 = 0.0064\ min^{-1}$. *The measured output is the total pressure described by*

$$y = x_1 + x_2. \tag{7.19}$$

We study the estimation problem presented in Example 7.4.2 with the initial conditions

$$x_0 = \begin{bmatrix} P_A \ P_B \end{bmatrix}^T = \begin{bmatrix} 3 \ 1 \end{bmatrix}^T, \qquad \bar{x}_0 = \begin{bmatrix} 0.1 \ 4.5 \end{bmatrix}^T.$$

The estimator is tuned as

$$Q_w = \begin{bmatrix} 1 \times 10^5 & 0 \\ 0 & 100 \end{bmatrix}, \qquad R_v = 100, \qquad \text{and} \qquad \Pi_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Note that we do not require tunings on the integrating disturbance – we are using a perfect model, so no integrators are added.

The sampling time of the system is six seconds and the estimation horizon is 20 sampling times. We examine the results of the state estimator after twenty output measurements are taken, i.e. at 114 seconds, in Figure 7.12. Without state constraints, the optimal estimate of the partial pressure of species B is negative. Applying the constraint that the partial pressures must each be positive yields a new optimal estimate for which the constraint is not active. This constrained estimate is clearly better, since the estimated states converge to the true states of the system. The cost function of the constrained estimator is 19.76, compared to the unconstrained value of 40.35. We present the output values associated with the constrained and unconstrained estimator in Figure 7.13. The constrained estimate fits the data well, while the unconstrained estimate fits the data in a way that does not adequately represent the true dynamics of the system. It is clear that enforcing physical constraints in the estimation problem plays a crucial role in yielding accurate state estimates.

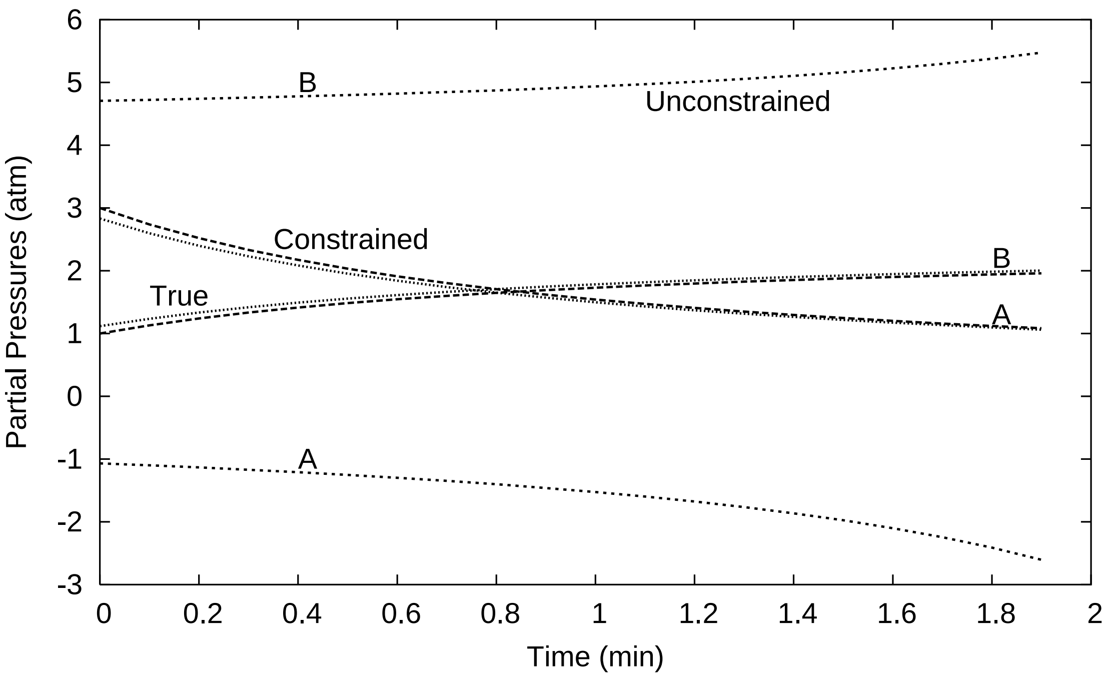

Figure 7.12: Optimal pressure estimates for Example 7.4.2

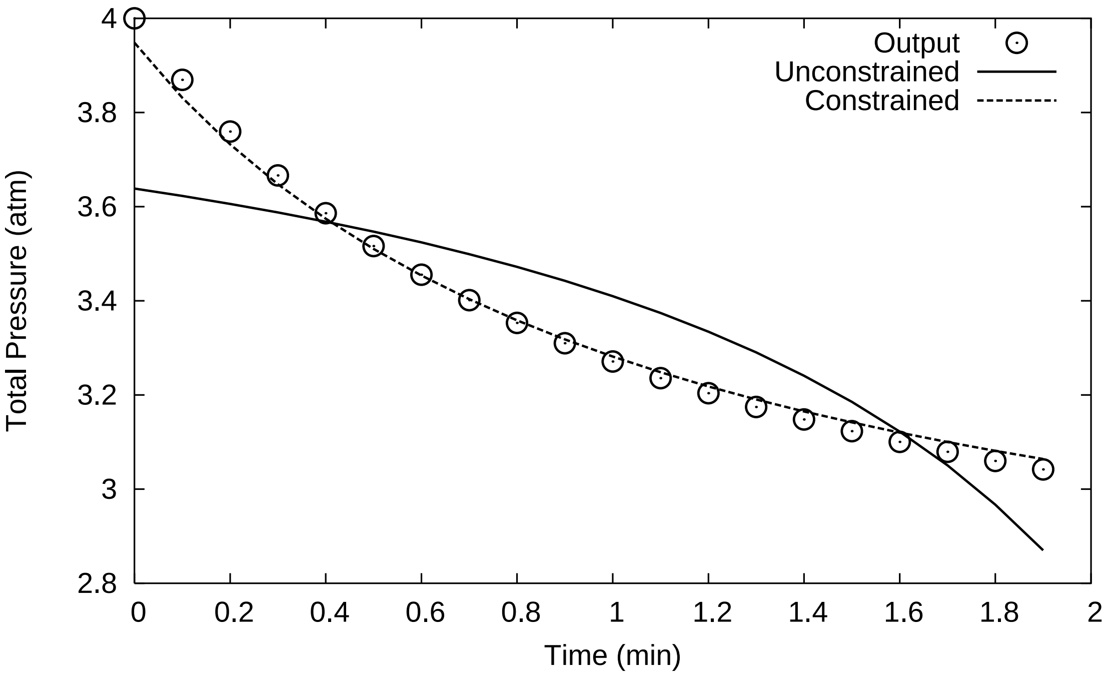

Figure 7.13: Fit of output estimates to measurements for Example 4.3.2

The effects of undesirable local minima in state estimation can be more severe to the closed-loop behavior of an MPC system [3] when compared to local minima in the regulator. One reason the estimator is more important is that both the target calculation and regulator rely on the results of the state estimator for good performance, while locally optimal behavior in the regulator does not affect the state estimates or steady-state targets. The stabilizing properties of local minima in the regulator were mentioned by Chen and Allgöwer [19]. These properties were proven in the context of stabilizing suboptimal control by Scokaert et al. [107]. We note that in the case of the regulator, the effect of local minima is to stabilize the system at set point, but to do so in an inefficient way. This behavior contrasts with the role of local minima in the estimation problem, since local minima do not necessarily converge to the true states of the system, as is evident in Example 7.4.2. It is possible that local minima in the estimator can lead to failure of the closed-loop system. We can conclude that local optima in the state estimation problem are an important concern for closed-loop operation when employing a nonlinear model. We show that through the application of physically motivated constraints, we can easily avoid these unwanted local minima for our examples.

7.5 Conclusions

Nonlinear models have significant advantages over linear models for control and estimation. At the same time, they also present new problems that must be addressed to guarantee their usefulness. This chapter demonstrates both the features and caveats of nonlinear models in MPC, and presents simple and effective methods for analyzing and avoiding these difficulties.

In this chapter, we show that the choice of disturbance models is an important issue for nonlinear plants. The choice of disturbance model can determine whether a system can be stabilized. In general, the input disturbance model is appropriate for cases of plant/model mismatch, although it is ineffective for a specific class of systems. We further demonstrate poor performance of linear MPC on this class of system. A future direction of research is to investigate a class of disturbance models in which the integrating term is added to model parameters. For instance, in the case of the solar collector, the disturbance is anticipated to come from the solar input, so adding the integrating disturbance to the solar input term is a natural choice.

We also investigate the occurence and role of local minima in the nonlinear regulator and estimator problems. While using linear models always results in convex optimization problems, regardless of horizon length, nonlinear models can introduce nonconvexities into these mathematical programs. In the case of the regulator, long prediction horizons can lead to locally optimal periodic solutions that can be removed by using a shorter prediction horizon. Also, constraints can be used to steer a system towards a more favorable local optimum. For the estimator, applying physical constraints to the system may avoid undesired local optima. Although in both cases, the constraints are not active at the solution, they aid the optimizer in finding worthwhile optimal solutions that may not be found otherwise. We recommend that even if a constraint is not expected to be encountered by the physical system, for instance negative absolute temperatures, it should be included in the controller and estimator formulations to avoid unwanted local minima.

Chapter 8

Conclusions

Skeptic does not mean him who doubts, but him who investigates or researches, as opposed to him who asserts and thinks that he has found.

— Miguel de Unamuno

Nonlinear model predictive control is a technology that is recieving increasing attention from academic researchers and industrial practitioners alike. Its strengths include its ability to handle constraints without integrator windup and its use of nonlinear process models that capture the behavior of the system in wider operating regions than linear models. While ad hoc methods for nonlinear control and estimation have been applied in industry for at least a decade, these methods are not reliable for general process models and typically do not have even nominal stability guarantees.

Many formulations for nonlinear MPC have been investigated by researchers, some complicated by terminal constraints, others with narrowing constraints to focus the state to set point. Most approaches rely on the industrial standard, the extended Kalman filter, for state estimation, or assume that the states can be measured. The formulations for the nonlinear MPC regulator and moving horizon estimation for nonlinear models presented in this study have been available for several years in similar forms, but due to computational limitations, true nonlinear MPC has not yet been widely practiced.

The main contributions of this dissertation are as follows. First, this study has presented a simplified nonlinear regulation problem without terminal constraints that combines ease of solution with excellent control properties. Next, a nonlinear moving horizon estimator with smoothing covariance was presented that was shown to be superior to the extended Kalman filter and the moving horizon estimator with a filtering covariance update. The regulator and estimation constrained minimization problems were solved reliably in real-time by a new algorithmic framework called feasibility-perturbed sequential quadratic programming. This optimization method possesses proven convergence properties, and has many advantages that are consequences of maintaining feasibility at each iteration. Finally, this computational approach was used to explore the fundamental differences between linear and nonlinear MPC performance properties. A more detailed description of the specific achievements of this study are summarized below.

8.1 Regulation

In this aspect of the study, we presented a regulator formulation that was advantageous for nonlinear model predictive control. The regulator objective function incorporated a terminal penalty that approximated the infinite-horizon tail cost of the open-loop control trajectory. Initially, the terminal state of the open-loop prediction was constrained to lie within a terminal region. We presented a global optimization method to calculate the size of this required terminal region. The region was constructed such that applying a linear feedback law inside the region resulted in a decrease in the cost function of at least one half the reduction of the cost function of the linearized system about the state and input targets. The region was further restricted so that applying the linear feedback law did not violate the state and input constraints. We determined that this terminal region is not usually small enough to warrant explicitly imposing an inequality constraint on the terminal state and presented these regions for a simple example. To guarantee that every regulator NLP had a solution, we augmented the regulator formulation to include softened state constraints. We demonstrated the effectiveness of this formulation on a variety of simulated examples. The resulting optimization problem, with terminal penalty, softened state constraints, and no terminal state constraints, was the regulator formulation considered for the remainder of this study.

8.2 Estimation

Since the regulator was formulated for state-space models, it was necessary to have an accurate estimate of the state at every sampling time. This portion of the study concentrated on estimating states from plant measurements. The challenges of state estimation are to eliminate the effects of noise in the measurements, infer states from a history of input and output data, and to determine integrated disturbances to remove offset in closed-loop operation. The contributions of this dissertation in the area of state estimation included deriving a formula for an approximate nonlinear smoothing covariance update. We showed that the smoothing update removed the undesirable periodic perturbations that resulted from the filtering update. We further demonstrated the benefits of moving horizon state estimation over the industrially popular extended Kalman filter. On simple examples, the extended Kalman filter did not converge to the true states of the system, but moving horizon estimation did converge using identical tuning parameters. We concluded that the moving horizon estimator with smoothing covariance update was a reliable state estimation method.

8.3 Efficient solution techniques

The optimization problems posed in the MPC regulator and moving horizon estimator share a similar structure. This part of the study introduced a new algorithm for solving nonlinear model predictive control regulator and moving horizon estimation problems by directly exploiting this structure. The method, called feasibility perturbed SQP, is similar to standard SQP methods since it approximates the optimization problem as a quadratic program and solves each quadratic program with a trust region constraint. However, the FP-SQP algorithm differs from typical SQP methods in that the solution to each QP is then perturbed onto the feasible set. A stabilizing perturbation matrix was developed that significantly reduced the computational burden compared to the naïve unstabilized perturbation. The resulting algorithm, by maintaining feasibility, is robust to on-line failures, is computationally efficient, and can generate feasible suboptimal iterates when required. The quadratic subproblems are always feasible, regardless of trust region size. Also, since all iterates satisfy the constraints, the objective is used as the merit function, avoiding the Maratos effect. The quadratic programs that are solved by this method possess a special overlapping block-diagonal structure and are decomposed and solved efficiently using a tailored primal-dual interior-point method. A scaled trust region method and structured Hessian approximations are introduced and studied. As shown in Table 6.3, the resulting methodology requires fewer iterations and far less computational time than a standard NLP solver on most examples. Further, the feasibility perturbed SQP algorithm is not as prone to failure as the standard nonlinear programming package.

8.4 Nonlinear MPC analysis

Due to nonlinearities in the model, the optimization problems in the regulator, estimator, and target calculation may all possess local minima. We demonstrated examples of local minima for both the regulation and estimation problems. We showed that shortening the prediction horizon in the regulator can avoid the appearance of unwanted local minima. We further showed that the application of constraints in both problems can remove local minima, especially in the state estimator. We also studied the closed-loop behavior of nonlinear model predictive control. We presented a solar collector example and concluded that for unmeasured disturbances or plant/model mismatch, the input disturbance model is usually more effective than the output disturbance model. We then presented a counter example in the form of a simple CSTR system in which the input disturbance model cannot stabilize the closed-loop system due to its inability to adjust the output target to a feasibile steady-state value of the plant. We developed a method of analyzing an aspect of closed-loop stability by examining the locus of steady states for the plant and model and comparing the effects of disturbance models on them. We compared nonlinear MPC to linear MPC and concluded that even in cases of mismodelled systems, a nonlinear model generally performed better than a linear model obtained from linearizing the original nonlinear model. We conclude that nonlinear MPC performs better than linear MPC for many examples relevant to the chemical process community.

8.5 Future Work

While this study has answered some open questions about practical implementation of nonlinear model predictive control, other questions remain open, and still others have been raised in this investigation. One of the biggest questions about nonlinear model predictive control concerns the ability to develop a reliable nonlinear model. Until nonlinear models are more easily identified, the impact of nonlinear solution methods will not be fully realized. Other research possibilities include efficient parametric sensitivity calculations for MPC applications, which is the computational bottleneck of our proposed approach. One potential solution is the use of parallel computing architectures such that derivatives with respect to each variable can be obtained simultaneously rather than in series. The improvement in performance time of such a technique is expected to be substantial. Another topic that merits future investigation is disturbance models. While input and output disturbance models often perform adequately, process knowledge could be used to model integrating disturbances more effectively. For instance, if disturbances are known to enter a process through a specific parameter, it seems reasonable to add the integrator to that parameter. To better understand the advantages and disadvantages of different formulations of NMPC, the control community needs to focus on a definitive set of benchmark problems. In the future, these benchmark examples will further define the new challenges of NMPC.

The software written for this study is freely available, and includes the models referenced in this investigation; see Appendix B for more information.

Appendix A

CSTR Case Study

To illustrate some of the features of a typical nonlinear process and to examine the difficulties involved in implementing a linear feedback control law on such a process, we consider a first order, exothermic, irreversible reaction $A \rightarrow B$ carried out in a continuous stirred tank reactor (CSTR) in the liquid phase with external coolingas described in Example 6.9.1. We desire to operate the reactor at an open-loop unstable operating point. For now, we assume that we can perfectly measure both the concentration of A and the temperature of the reactor. The cooling rate is the manipulated variable in this example.

A.1 Dynamics without control

As shown in Figures A.1 and A.2, multiple steady states exist for a range of values of the parameter T_c. The stability of these steady states are investigated and then a demonstration of a linear controller is performed on these systems.

The linearized matrix $\mathbf{A}$ in which $\dot{x} = \mathbf{A}x$ for this system is:

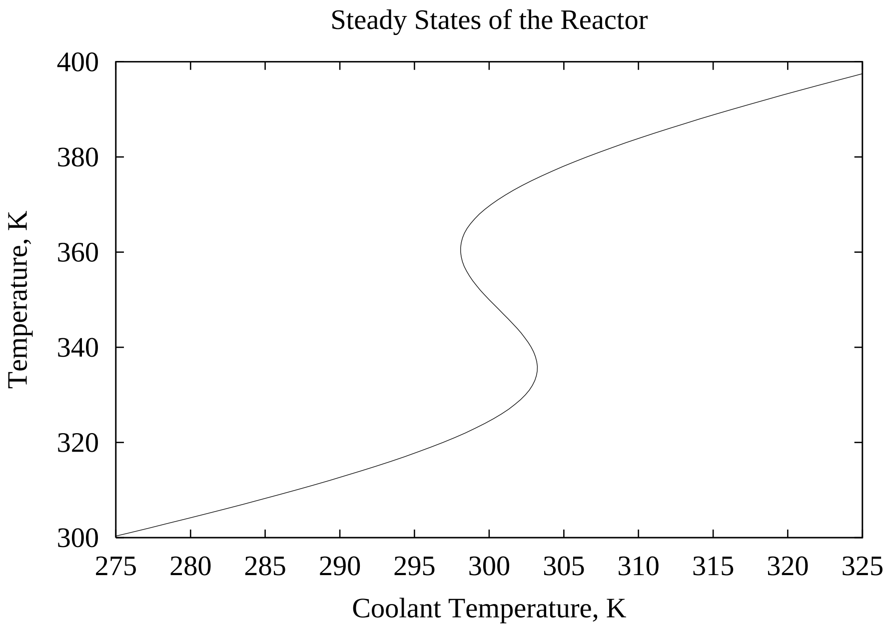

Figure A.1: Steady State Temperatures

$$A = \begin{bmatrix} -\frac{q}{V} - k_0 e^{-\frac{E}{RT_{ss}}} & -\frac{E}{RT_{ss}^2} k_0 C_{A_{ss}} e^{-\frac{E}{RT_{ss}}} \\ -\frac{\Delta H}{\rho C_p} k_0 e^{-\frac{E}{RT_{ss}}} & -\frac{q}{V} - \frac{\Delta H}{\rho C_p} \frac{E}{RT_{ss}^2} k_0 C_{A_{ss}} e^{-\frac{E}{RT_{ss}}} - \frac{UA}{V\rho C_p} \end{bmatrix} \tag{A.1}$$

The stability of the system at each steady-state point depends on the eigenvalues of **A** at that point. Table A.1 summarizes the stability characteristics of each steady state as they relate to Figure A.1.

Reactor Temperature Range	**Sign of Eigenvalues**	**Steady State**
T < 335.654	- , -	Sink
335.654 < T < 360.511	+ , -	Saddle
360.511 < T < 379.611	+ , +	Source
379.611 < T	- , -	Sink

Table A.1: Stability of Steady States

The parameter values at which the bifurcations occur are 298.08 K, 303.23 K, and 306.22 K for the coolant temperature. At 298.08 K, the bifurcation is a saddle-

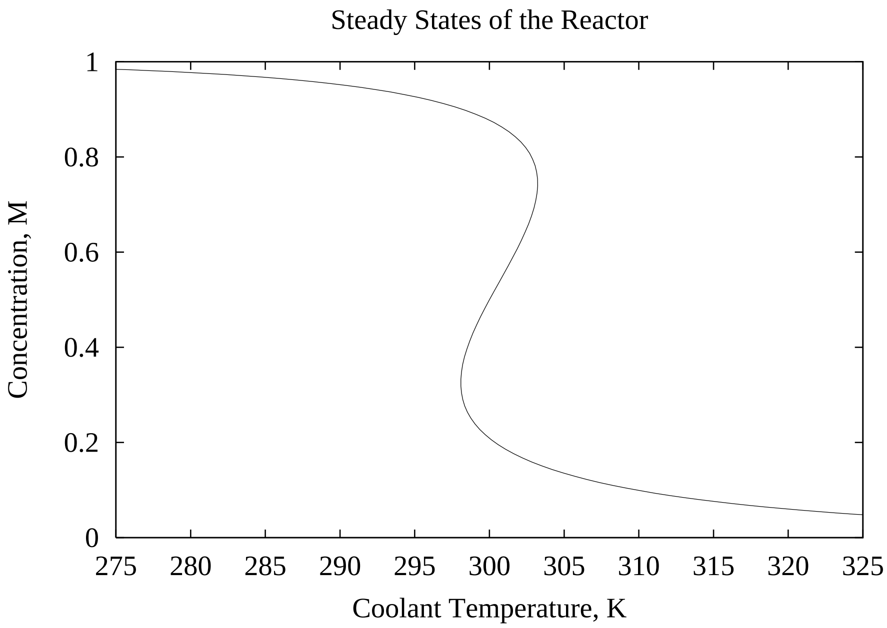

Figure A.2: Steady State Concentrations

node, which occurs on the upper branch of the temperature curve. A saddle-node bifurcation exists at 303.23 K on the lower branch of the temperature curve. A Hopf bifurcation occurs on the upper branch at a parameter value of 306.22 K. These dynamics are consistent with those of a CSTR in Region IIIb of Uppal, Ray, and Poore [121].

Figures A.3 to A.9 demonstrate the characteristics of the system as the parameter T_c is adjusted. First, only one steady state, which is stable (Figure A.3) is present.

Then, at the first bifurcation point, approximately 298.1 K (Figure A.4), a second steady state appears. The original steady state is still stable; however, the new steady state is unstable.

Figure A.5 shows the system that is studied in a later section of this document. Three steady states are in this figure; the one on the left is unstable, the one in the

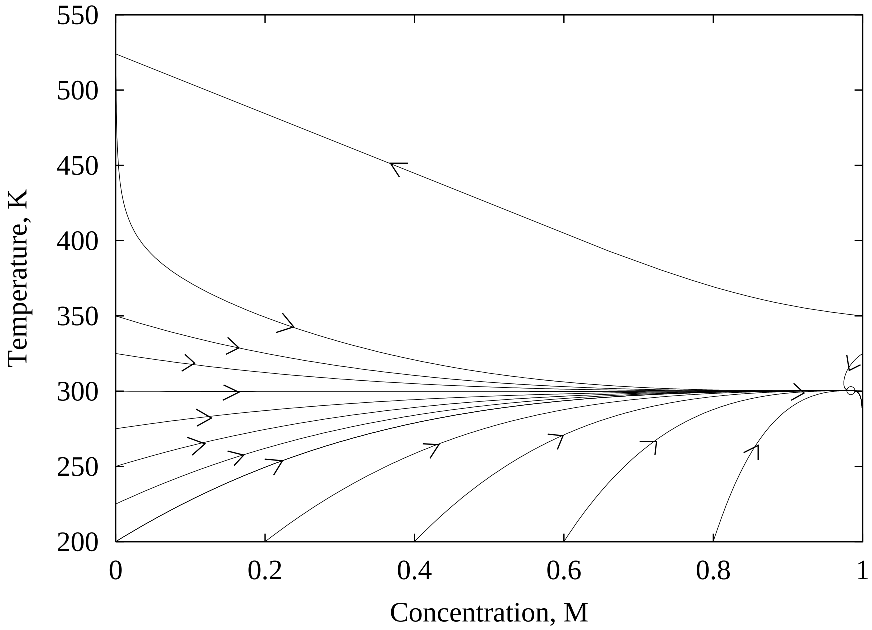

Figure A.3: Low Coolant Temperature Behavior ($T_c = 275$ K)

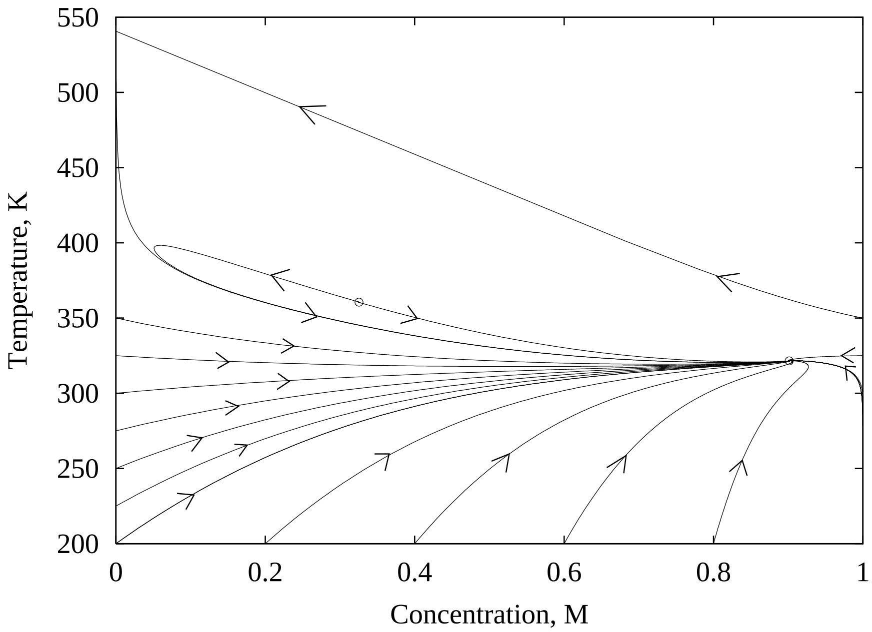

Figure A.4: Behavior at First Saddle-Node Point ($T_c = 298.08$ K)

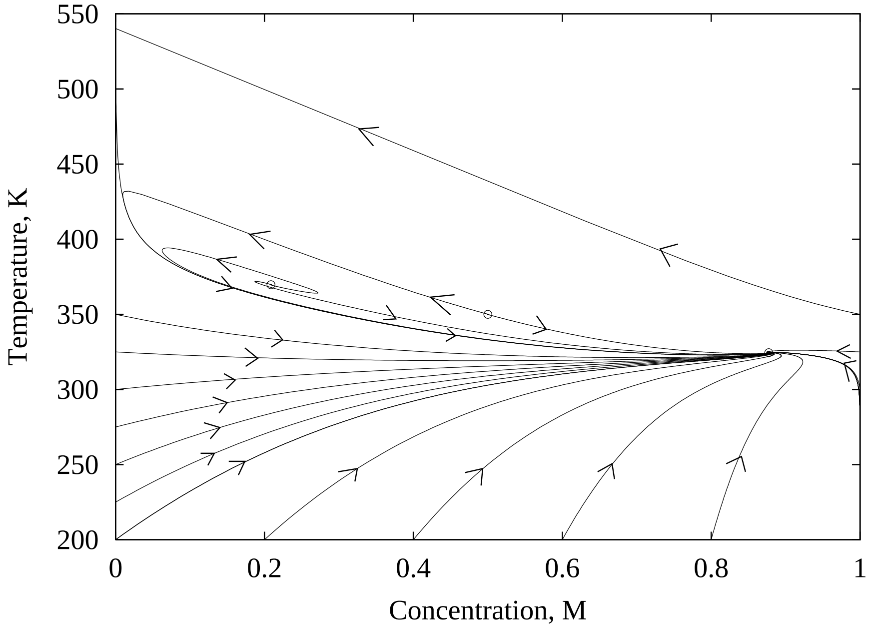

Figure A.5: Behavior with Three Steady States (T_c = 300 K)

middle is a saddle, and the one on the right is stable.

At the second bifurcation of about 303.2 K, shown in Figure A.6, only two steady states exist. The steady state on the right is stable, but the one on the left is an unstable steady state for which nearby states decay to a periodic orbit.

Between the second and third bifurcations, the only steady state is unstable. Figure A.7 shows the trajectories from the steady state as they approach a limit cycle.

Figure A.8 displays the Hopf bifurcation point. It can be seen that points near the steady state do not approach the steady state, but rather orbit it.

Finally, the steady state on the left becomes stable, as shown in Figure A.9.

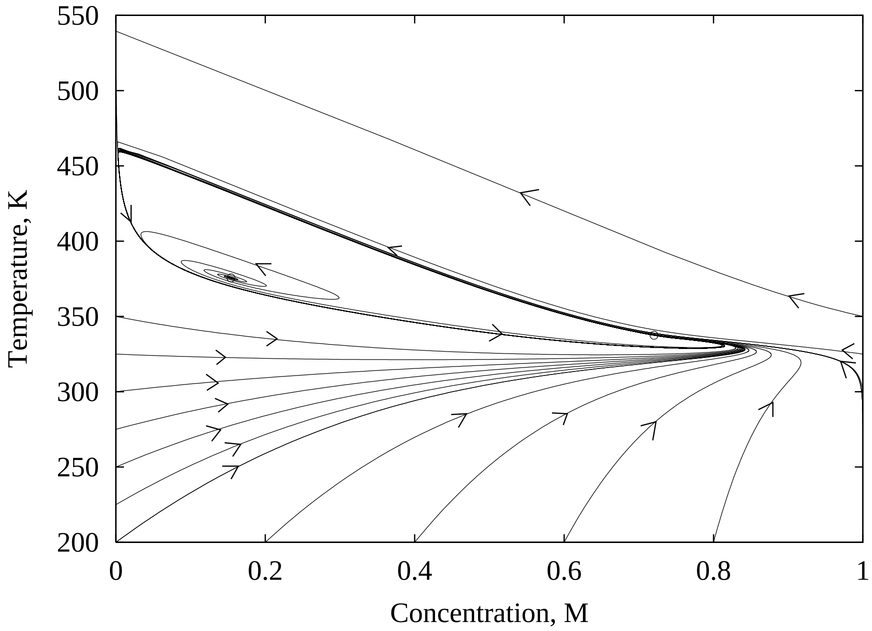

Figure A.6: Behavior at Second Saddle-Node Point ($T_c = 303.23$ K)

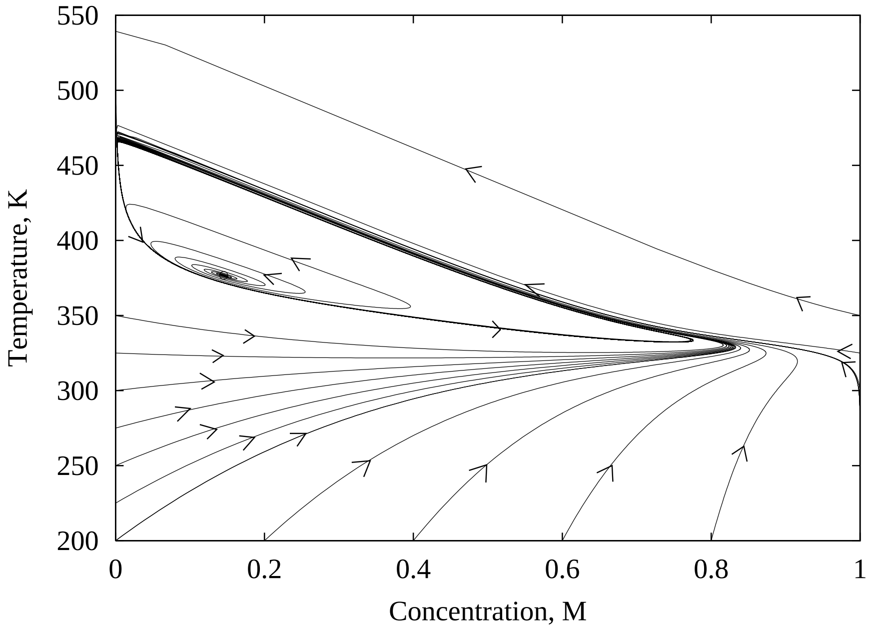

Figure A.7: Behavior in Unstable Range ($T_c = 304$ K)

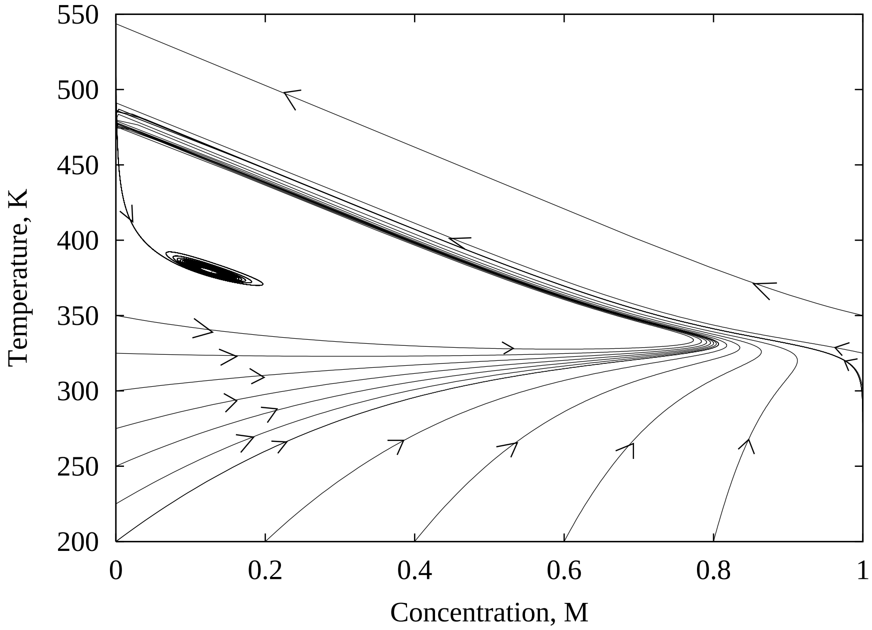

Figure A.8: Behavior at Hopf Bifurcation Point ($T_c = 306.22$ K)

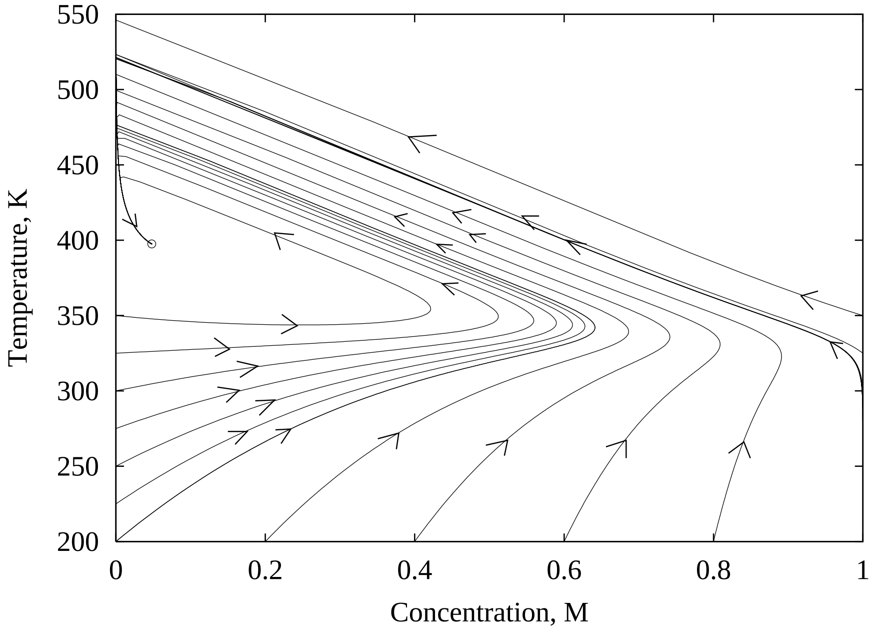

Figure A.9: Behavior at High Temperatures ($T_c = 325$ K)

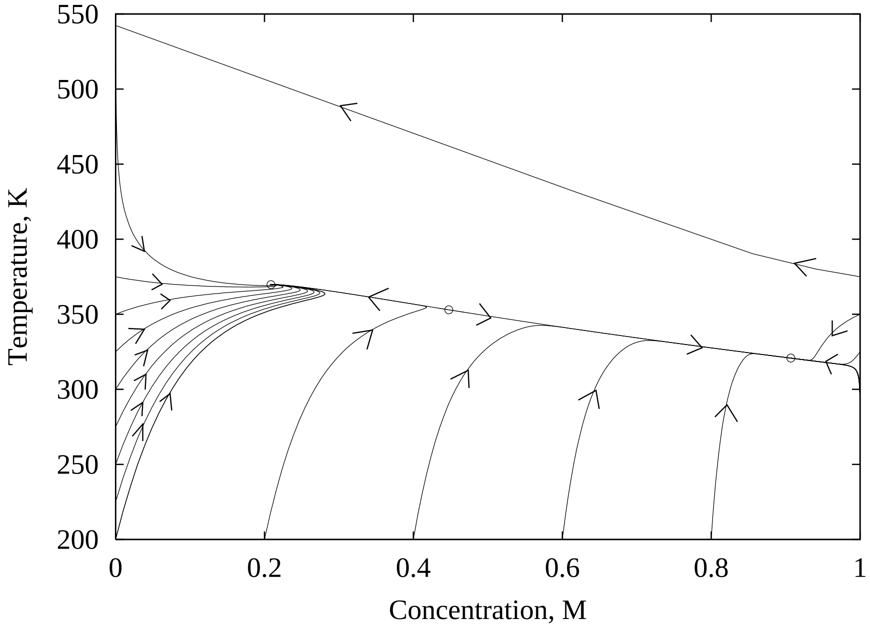

Figure A.10: Linear Controller on Unstable Point of Figure A.5

A.2 Dynamics under linear control

The need for some form of controller is evident, since it is not possible to maintain the system at an unstable steady state without control. To observe the effects of linear control on both the local and global stability of the system, linear feedback control is applied to the unstable steady states from the previous figures. The values of the feedback gain matrix K used were calculated based on the optimal linear quadratic regulator of the linearized system. The Figures A.10 and A.11 show properties that demonstrate why a linear controller fails for states far from the set-point for a nonlinear system.

In Figure A.10, we designate the steady state on the left of Figure A.5 as the set-point, and it is stabilized. However, two other steady states appear. The first, on the right, is stable. The result of this additional attractor is that states that are sufficiently distant from the desired steady state on the left are drawn instead

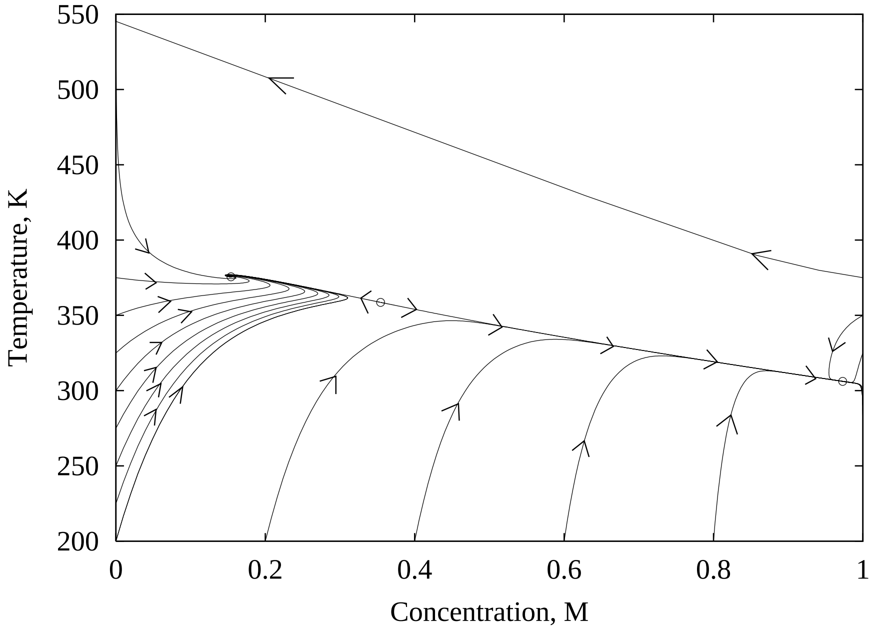

Figure A.11: Linear Controller on Second Saddle-Node Point

to the undesirable steady state on the right. This is an undesirable side effect of implementing a linear controller on a nonlinear system. The presence of the middle steady state, which is a saddle point, represents the boundary between states that are attracted to the set point and those that are attracted to the undesired steady state on the right.

Again, applying the linear controller to the Hopf point in Figure A.6, a similar phenomenon occurs. A saddle point (in the center) and an undesired stable steady state (on the right) appear. It is evident from Figures A.10 and A.11 that the linear feedback controller is only applicable in a limited region of the state space. In schemes such as target linearization, in which the linear control used is dependent upon the target to which the states and inputs are to be steered, the controller would reach the wrong steady state and the system would remain there. In normal linear MPC with disturbance models, the state would not stabilize at the additional steady-state, but rather would either reach the true set point or go unstable. For this reason, target linearization is not recommended for nonlinear systems unless the secondary stabilizing points are not expected.

Appendix B

NMPC Web Tool

One of the highlights of this study is the release of the algorithms presented in this dissertation on the World Wide Web. This prototype code is freely available software and is written for use with Octave versions 2.1.34 and higher. In this appendix, we include some of the documentation that accompanies the software. The NMPC tool is available for download at `http://www.che.wisc.edu/~tenny/nmpc/`. Accompanying the software is the suite of benchmark examples, many of which are presented in this dissertation.

B.1 Features of NMPC Tool

The following features are strengths of the NMPC tool. This list of features is included in the distribution as the file `nmpc_features`.

Argument checking:

The algorithm checks consistency in the size of all parameters. If any of the parameters are inconsistent, the code will exit with an error message describing the problem.

Automatic scaling to avoid ill-conditioning:

The scaling parameters can be entered by the user, or they will be determined by the algorithm automatically. The scaling factors are used to prevent extreme ill-conditioning in the QP solvers. The default scaling constants are determined by the initial conditions and the conditions at the set point. To override an automatic scaling constant, specify the scaling parameter in the parameters file.

Approximated infinite horizon solution:

When running the regulator, the approximated infinite horizon penalty is added to the objective, based on the linear feedback control law about the target. For the estimator, as the estimation horizon moves, the initial covariance is updated to account for previous model dynamics.

Nonlinear and/or linear plants, models, output equations, disturbances:

The algorithm has the capability of handling many types of plants and models, output equations, and disturbances. The code takes advantage of when the user specifies linear outputs and models instead of their nonlinear counterparts. Also, the capability is available to investigate not just step and white noise disturbances, but also many nonlinear disturbances added in both the plant and the output equation.

Discrete and continuous time support:

The model and the plant can be specified as either discrete time or continuous time; currently nonlinear discrete-time models are not supported, but they may be in the future. For estimation problems, this function is useful for analyzing plant data. For instance, if there is a plant data file, the output function could be specified, and the function could simply return the value of $y(t)$. In this way, state estimation can be performed from plant data. The plant can also be any linear or nonlinear function.

MHE (estimation with/without constraints):

The estimation problem is performed by moving horizon estimation. This approach has advantages over the extended Kalman filter in its ability to handle constraints in the estimate, as well as its capability of producing an optimal estimate.

Integrated disturbance support:

The algorithm is written to automatically augment the system for integrated disturbance models. When integrated disturbances are present, a target calculation will also be performed.

Estimation problem with/without regulator:

By not specifying the regulator parameters, the code will function as a state estimator (with no regulation). This capability is especially useful in analyzing historical data.

Efficient structured interior point method:

The regulator and estimator problems are solved with specially written interior-point QP solvers, which take advantage of the natural structure of the optimization problem for increased computational efficiency.

Feasible suboptimal method:

For solving nonlinear problems, the SQP method can be run suboptimally by specifying a maximum iteration limit on the SQP solver. The solution will be feasible with respect to the constraints and guaranteed to be no worse than the initial guess.

Warm restarts:

When there are only small disturbances to the control system, it is useful to take advantage of past information to speed convergence. In such cases, the user may choose to have the regulator implement warm restarts.

Reduced models (different size than plant):

In many practical control systems, the model only approximates the plant. For instance, in distillation columns, often the model is greatly reduced from the full first-principles model. To simulate such cases, the capability of having a different number of states in the plant than in the model is included.

Linear model on nonlinear plant:

This code is useful for controlling a nonlinear (which is typical) plant with linear MPC. This technique is useful for identifying the potential shortcomings of linear MPC.

General linear constraints:

This algorithm accepts any general linear constraints on the inputs, states, and disturbance estimates. All constraints except state disturbances and inputs are treated as soft constraints.

Plant/model mismatch:

Investigating controller robustness is made simpler by this algorithm, since it allows different code for the plant and the model.

Clipping/altering inputs in the plant:

The plant is capable of clipping the input in two ways. The first way is by not letting the estimator know the input was clipped. To perform this style of clipping, do NOT change the variable uk in the plant code. The second way of clipping is to let the estimator know the input was clipped. This can be performed by changing the value of uk in the plant to the clipped value. The algorithm will read uk after the plant is called and use that value as the input injected to the plant.

B.2 Available Parameters

The following is a list of input and output parameters that are used by the NMPC tool. This section appears as the file `nmpc_README` in the software package.

B.2.1 General Simulation Parameters

delt (REQUIRED):

The sampling time.

t (REQUIRED):

The simulation length, in number of sampling times.

basetime (default = 0):

The starting time of the simulation.

x0 (REQUIRED):

The initial state of the plant.

u0 (REQUIRED if using regulator with nonzero S or Greg):

The initial current input.

xbar0 (REQUIRED):

The initial state estimate of the model.

fplant OR Ap, Bp, Gp (One of the two sets REQUIRED):

This specifies the dynamics for the plant.

If fplant is specified, then the plant is treated as nonlinear. The variable fplant is a string which corresponds to the name of a function. The function takes the plant state and the current time as calling arguments. It also must declare the global variables uk and wk, which are the inputs and the input disturbances. It returns the time derivative of the plant states.

If fplant is not specified, then the plant is treated as linear.

discreteplant (default = 0):

If the plant is in discrete time, change discreteplant to 1.

fmodel OR Am, Bm, Gm (One of the two sets REQUIRED):

This specifies the dynamics for the model.

If fmodel is specified, then the model is treated as nonlinear. The variable fmodel is a string which corresponds to the name of a function. The function takes the state and the current time as calling arguments. It also must declare the global variables uk and wk, which are the inputs and the input disturbances. It returns the time derivative of the states.

If fmodel is not specified, then the model is treated as linear.

discretemodel (default = 0):

If the model is in discrete time, change discretemodel to 1. There is currently no support for discrete-time nonlinear models.

fpart (default = empty):

This specifies the Jacobian of the model. The string fpart corresponds to the name of a function that takes the state and the current time as calling arguments. It also must declare the global variables uk, which is the input, and wk, which is the state disturbance. It returns a matrix that has as many columns as the sum of states, inputs, and state disturbances and as many rows as the number of states. Each row corresponds to the derivative of each row of fmodel with respect to each state, then each input, and then each disturbance.

If fpart is not specified but fmodel is specified, the Jacobian is computed numerically.

fout OR Cm (One of the two REQUIRED):

This is the model output. If fout is specified, the output is treated as nonlinear. The variable fout is a string that corresponds to the name of a function that takes the state as a calling argument and returns the output.

If fout is not specified, then the output is treated as linear.

foutpart (default = empty):

This specifies the Jacobian of the model output. The string foutpart corresponds to the name of a function that takes the state as the calling argument. It returns a matrix that has as many columns as the number of states and as many rows as the number of outputs. Each row corresponds to the derivative of each row of fout with respect to each state.

If foutpart is not specified but fout is specified, the Jacobian is computed numerically.

foutplant OR Cp (One of the two REQUIRED):

This is the plant output. If foutplant is specified, the output is treated as nonlinear. The variable foutplant is a string that corresponds to the name of a function that takes the plant state as a calling argument and returns the output.

If foutplant is not specified, then the output is treated as linear.

foutdis (default empty):

This variable is a string which corresponds to the name of a function. The file takes time as a calling argument and returns the output disturbance vk, which is added to the output measurement from the plant.

finpdis (default empty):

This variable is a string which corresponds to the name of a function. The file takes time as a calling argument and returns the input disturbance wk, which is multiplied by Gp and added to the state equation for a linear plant or sent as a global variable to the fplant function.

B.2.2 Estimator Parameters

Nestimator (REQUIRED):

The horizon length of the moving horizon estimator.

Qest, Rest, Pest (REQUIRED):

Penalty matrices for the moving horizon estimator. Qest is the inverse covariance of the state noise. Rest is the inverse covariance of the output noise. Pest is the inverse covariance of the *a priori* initial state estimate.

Hest, hest, Sest, sest, Gest, gest (default empty):

Estimator constraints. The form of these matrices and vectors are:

$$\text{Hest x} \leq \text{hest, Sest w} \leq \text{sest, Gest v} \leq \text{gest.}$$

Xu, Xy (default zeros):

The integrated disturbance models. They appear as follows:

$$\begin{aligned} x_{k+1} &= F(x_k, u_k + \text{Xu } p_k, w_k, t) \\ p_{k+1} &= p_k + \xi_k \\ y_k &= g(x_k) + v_k + \text{Xy } p_k \end{aligned}$$

Where p is the integrated disturbance.

Qxi, Pz (REQUIRED if Xu and Xy are not both zeros):

Qxi is the inverse covariance of the integrator noise ξ_k. Pz is the inverse covariance of the *a priori* initial integrator state.

B.2.3 Regulator Parameters

N (REQUIRED if regulator present):

The length of the regulator prediction horizon.

isbatch (default = 0):

Set to 1 if running the process as a batch instead of continuous operation. The user must supply P and S must be nonzero. R will be ignored.

uset (REQUIRED if regulator present):

The input setpoint. If none is supplied, the code will function as an open-loop estimator.

xset, yset, fset (Only one REQUIRED):

The state or output setpoint, respectively, or a function that gives the output setpoint. If yset is given, xset is calculated. If xset is given, yset is calculated. Both may be supplied. The variable fset is a string that contains the name of a function that takes time as its argument and returns an output setpoint at each time.

Q (REQUIRED if regulator present):

The regulator penalty on deviation of states from setpoint. Must be positive semidefinite.

R (REQUIRED if regulator present):

The regulator penalty on deviation of inputs from setpoint. Must be positive semidefinite. Not required if S is nonzero.

S (REQUIRED if regulator present):

The regulator penalty on rate of change of inputs. Must be positive semidefinite. Not required if R is nonzero.

P (REQUIRED only if isbatch = 1)

The penalty on the state at the end of the prediction horizon for batch problems. Must be positive semidefinite.

Hreg, hreg, Ereg, ereg, Dreg, dreg, Greg, greg (default empty):

Regulator constraints. The form of these matrices and vectors are:

Hreg x $\leq$ hreg, Ereg y $\leq$ ereg, Dreg u $\leq$ dreg, Greg (u - uminus) $\leq$ greg.

coldrestart (default 0):

Determines whether the regulator makes use of the guess from the previous timestep. Otherwise, the regulator is initialized with a time-varying LQR solution.

B.2.4 Target Calculation Parameters

setinds (default 1:l):

This vector contains the indices of the outputs which have setpoints. The default corresponds to all outputs.

B.2.5 Scaling Parameters

xscale (calculated automatically unless specified):

The vector of the scales of the model states.

uscale (calculated automatically unless specified):

The vector of the scales of the inputs.

yscale (calculated automatically unless specified):

The vector of the scales of the outputs.

wscale (calculated automatically unless specified):

The vector of the scales of the input disturbances.

B.2.6 SQP Optimization Parameters

mhereltol (default 1e-4):

Relative tolerance of estimator SQP convergence.

mheabstol (default 1e-6):

Absolute tolerance of estimator SQP convergence.

regreltol (default 1e-4):

Relative tolerance of regulator SQP convergence.

regabstol (default 1e-8):

Absolute tolerance of regulator SQP convergence.

decrease_its (default 20000):

The total SQP iterations allowed. For suboptimal performace, lower this parameter.

B.2.7 QP Optimization Parameters

mheqpreltol (default 1e-4):

Relative tolerance of estimator QP convergence.

mheqpabstol (default 1e-8):

Absolute tolerance of estimator QP convergence.

regqpreltol (default 1e-8):

Relative tolerance of regulator QP convergence.

regqpabstol (default 1e-10):

Absolute tolerance of regulator QP convergence.

maxqpits (default 100):

The maximum number of interior-point QP iterations performed prior to giving up.

B.2.8 Output Variables

times:

The vector of sampling times.

ylong:

The matrix of measured outputs. There are as many rows as outputs and as many columns as sampling times.

xlong:

The matrix of states of the plant. There are as many rows as plant states and as many columns as sampling times.

ulong:

The matrix of injected inputs. There are as many rows as inputs and as many columns as sampling times.

xhatlong:

The matrix of estimated states of the model. There are as many rows as model states and as many columns as sampling times.

plong:

The matrix of estimated integrated disturbances for the model. There are as many rows as added integrators and as many columns as sampling times.

yset:

The output set-point vector.

ytlong:

The matrix of output targets from the target calculation. The output target may change if yset is not feasible. There are as many rows as outputs and as many columns as sampling times.

xsetlong:

The matrix of state targets of the model. There are as many rows as model states and as many columns as sampling times.

usetlong:

The matrix of input targets for the model. There are as many rows as inputs and as many columns as sampling times.

objlong:

The vector of regulator objective function values.

B.3 Examples Included

The NMPC software package includes nonlinear example systems, many of which are studied in this dissertation. The examples are:

copoly: The copolymerization reactor studied by Bindlish [6] and used in Example 4.3.3.

cstr: The exothermic tank reactor of Example 6.9.1.

cstr4: The exothermic tank reactor with jacket dynamics of Example 3.4.1.

gas_estim: The gas-phase batch reactor in Example 4.3.2 for estimation studies.

maxcstr: A tank reactor with unattainable maximum yield as described in Example 7.3.2.

pend: The inverted pendulum on a cart in Example 3.4.2.

react_estim: A high-order reaction with nonlinear measurement, described in Example 4.3.1 for estimation studies.

solar: The solar collector in Example 7.3.1.

spring: The magnetically actuated damped mass-spring system of Example 6.9.2.

Bibliography

[1] Joao Albuquerque, Vipin Gopal, George Staus, Lorenz T. Biegler, and Erik B. Ydstie. Interior point SQP strategies for large-scale, structured process optimization problems. *Computers & Chemical Engineering*, 23(4):543–554, 1999.

[2] R. A. Bartlett, A. Wächter, and L. T. Biegler. Active set vs. interior point strategies for model predictive control. In *Proceedings of the American Control Conference*, pages 4229–4233, Chicago, IL, June 2000.

[3] B. Wayne Bequette. Nonlinear predictive control using multi-rate sampling. *The Canadian Journal of Chemical Engineering*, 69:136–143, February 1991.

[4] Dimitri P. Bertsekas. *Dynamic Programming*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[5] L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. Technical report, Departmnet of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA, February 2001.

[6] Rahul Bindlish. *Modeling and Control of Polymerization Processes*. PhD thesis, University of Wisconsin–Madison, 1999.

[7] Robert R. Bless and Dewey H. Hodges. Finite element solution of optimal control problems with inequality constraints. In *Proceedings of the 1990 American Control Conference*, pages 242–247, May 1990.

[8] Hans Georg Bock, Moritz Diehl, Daniel Leineweber, and Johannes Schlöder. Efficient direct multiple shooting in nonlinear model predictive control. In F. Keil, W. Mackens, H. Voss, and J. Werther, editors, *Scientific Computing in Chemical Engineering II, Volume 2: Simulation, Image Processing, Optimization, and Control*. Springer, Berlin, Heidelberg, 1999.

[9] Hans Georg Bock and Karl J. Plitt. A multiple shooting algorithm for direct solution of optimal ontrol problems. In *Proceedings of the 9th IFAC World Conference*, pages 242–247, Budapest, 1984. Pergamon Press.

[10] A. E. Bryson and M. Frazier. Smoothing for linear and nonlinear dynamic systems. Proc. Optimum Systems Synthesis Conf., U.S. Air Force Tech. Rept. ASD-TDR-063-119, February 1963.

[11] Mark Cannon, Basil Kouvaritakis, and J. Anthony Rossiter. Efficient active set optimization in triple mode MPC. *IEEE Transactions on Automatic Control*, 46(8):1307–1312, August 2001.

[12] Malamas Caracotsios. *Model Parametric Sensitivity Analysis and Nonlinear Parameter Estimation. Theory and Applications.* PhD thesis, University of Wisconsin–Madison, 1986.

[13] A. M. Cervantes, A. Wächter, R. H. Tütüncü, and L. T. Biegler. A reduced space interior-point strategy for optimization of differential algebraic systems. *Computers & Chemical Engineering*, 24:983–990, 2000.

[14] Madalena Chaves and Eduardo D. Sontag. State-estimators for chemical reaction networks of Feinberg-Horn-Jackson zero deficiency type. In James B. Rawlings, Babatunde A. Ogunnaike, and John W. Eaton, editors, *Chemical Process Control– VI: Sixth International Conference on Chemical Process Control*, volume 97, Tucson, Arizona, January 2001. AIChE Symposium Series.

[15] Chieh-Li Chen, Sy-Hong Lin, and Cha'o-Kuang Chen. Application of Taylor transformation to nonlinear predictive control problem. *Applied Mathematical Modelling*, 20(9):699–710, 1996.

[16] H. Chen and F. Allgöwer. A computationally attractive nonlinear predictive control scheme with guaranteed stability for stable systems. *Journal of Process Control*, 8(5–6):475–485, 1998.

[17] H. Chen, A. Kremling, and F. Allgöwer. Nonlinear predictive control of a benchmark CSTR. In *Proceedings of the 3rd European Control Conference ECC'95*, pages 3247–3252, Rome, Italy, 1995.

[18] Hong Chen and Frank Allgöwer. Nonlinear model predictive control schemes with guaranteed stability. In R. Berber and C. Kravaris, editors, *NATO ASI on Nonlinear Model Based Process Control*, pages 465–494. Kluwer, 1998.

[19] Hong Chen and Frank Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

[20] Qi Chen and William A. Weigand. Dynamic optimization of nonlinear processes by combining neural net model with UDMC. *AIChE Journal*, 40(9):1488–1497, 1994.

[21] D. Chmielewski and V. Manousiouthakis. On constrained infinite-time linear quadratic optimal control. *Systems & Control Letters*, 29:121–129, 1996.

[22] C. R. Cutler and B. L. Ramaker. Dynamic matrix control—a computer control algorithm. AIChE National Meeting, Houston, TX, April 1979.

[23] G. De Nicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear time-varying systems. *IEEE Transactions on Automatic Control*, 43(7):1030–1036, 1998.

[24] J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[25] Moritz Diehl, H. Georg Bock, Johannes P. Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[26] John W. Eaton. Octave: Past, present and future. In Kurt Hornik and Fritz Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, Vienna, Austria, March 2001.

[27] B.C. Eaves and R.M. Freund. Optimal scaling of balls and polyhedra. *Mathematical Programming*, 23:138–147, 1982.

[28] R. Findeisen, M. Diehl, Z. Nagy, F. Allgöwer, H. G. Bock, and J. P. Schlöder. Computational feasibility and performance of nonlinear model predictive control schemes. In *Proceedings of European Control Conference, Seminário de Vilar, Porto, Portugal*, 2001.

[29] C. E. García. Quadratic dynamic matrix control (QDMC) of nonlinear processes: An application to a batch reaction process. AIChE National Meeting, San Francisco, CA, November 1984.

[30] Carlos E. García and A. M. Morshedi. Quadratic programming solution of dynamic matrix control (QDMC). *Chemical Engineering Communications*, 46:73–87, 1986.

[31] Hasmet Genceli and Michael Nikolaou. Design of robust constrained model-predictive controllers with Volterra series. *AIChE Journal*, 41(9):2098–2107, 1995.

[32] R. Gesthuisen and S. Engell. Determination of the mass transport in the polycondensation of polyethyleneterephthalate by nonlinear estimation techniques. In *Proceedings of the 1998 IFAC DYCOPS Symposium, Corfu, Greece*, 1998.

[33] Bruce P. Gibbs and David S. Weber. Parameter identification for nonlinear model predictive control of fossil power plants. In *Proceedings of the 35th Power Instrumentation Symposium*, pages 221–234, Kansas City, MO, 1992.

[34] Bruce P. Gibbs, David S. Weber, and David W. Porter. Application of nonlinear model-based predictive control to fossil power plants. In *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1850–1856, Brighton, England, 1991.

[35] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for SOL/NPSOL (Version 4.0): A Fortran package for nonlinear programming, technical report SOL 86-2. Technical report, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1986.

[36] A. Griewank and Ph. L. Toint. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, 39:119–137, 1982.

[37] John Hauser and Hinke Osinga. On the geometry of optimal control: the inverted pendulum example. In *Proceedings of the American Control Conference*, pages 1721–1726, Washington, DC, 2001.

[38] Michael A. Henson and Dale E. Seborg. *Nonlinear Process Control*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.

[39] Evelio Hernandez and Yaman Arkun. A nonlinear DMC controller: Some modeling and robustness considerations. In *Proceedings of the American Control Conference*, pages 2355–2360, Boston, MA, June 1991.

[40] Alan C. Hindmarsh. ODEPACK, a systematized collection of ODE solvers. In R. S. Stepleman, editor, *Scientific Computing*, pages 55–64, Amsterdam, 1983. North-Holland.

[41] T. Hong, J. Zhang, A. J. Morris, E. B. Martin, and M. N. Karim. Neural based predictive control of a multivariable microalgae fermentation. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 345–350, Beijing, China, 1996.

[42] Bei Hu and Arno Linnemann. Toward infinite-horizon optimality in nonlinear model predictive control. *IEEE Transactions on Automatic Control*, 47(4):679–682, 2002.

[43] Dexiean Huang and Yihui Jin. Application of neural networks to nonlinear predictive control. In *Proceedings of the 1997 IEEE International Conference on Neural Networks*, pages 724–727, Houston, TX, USA, 1997.

[44] Alberto Isidori. *Nonlinear Control Systems*. Springer-Verlag, Berlin, 2nd edition, 1989.

[45] Ali Jadbabaie, Jie Yu, and John Hauser. Unconstrained receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46(5):776–783, 2001.

[46] Shi-Shang Jang, Babu Joseph, and Hiro Mukai. Comparison of two approaches to on-line parameter and state estimation of nonlinear systems. *Industrial and Engineering Chemistry Process Design and Development*, 25:809–814, 1986.

[47] Shi Shang Jang and Leon Lih Jong Wang. Experimental study of rigorous nonlinear model predictive control for a packed distillation column. *Journal of the Chinese Institute of Chemical Engineers*, 28(3):151–162, 1997.

[48] Ch. Jänsh and M. Paus. Aircraft trajectory optimization with direct collocation using movable gridpoints. In *Proceedings of the 1990 American Control Conference*, pages 262–267, May 1990.

[49] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.

[50] Babu Joseph and Frieda Wang Hanratty. Predictive control of quality in a batch manufacturing process using artificial neural network models. *Industrial and Engineering Chemistry Research*, 32(9):1951–1961, 1993.

[51] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, J. Basic Engineering*, pages 35–45, March 1960.

[52] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Trans. ASME, J. Basic Engineering*, pages 95–108, March 1961.

[53] A.H.G.R. Kan and G.T. Timmer. A stochastic approach to global optimization. In *SIAM Conference on Numerical Optimization*, 1984.

[54] S. S. Keerthi and E. G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57(2):265–293, May 1988.

[55] Mayuresh V. Kothare, Vesna Nevistić, and Manfred Morari. Robust constrained model predictive control for nonlinear systems: A comparative study. In *Proceedings of the 1995 IEEE Conference on Decision and Control*, pages 2884–2885, New Orleans, LA, USA, 1995.

[56] W. H. Kwon, A. M. Bruckstein, and T. Kailath. Stabilizing state-feedback design via the moving horizon method. *International Journal of Control*, 37(3):631–643, 1983.

[57] T. A. G. Langrish, A. S. Brooke, C. L. Davis, H. E. Musch, and G. W. Barton. An improved drying schedule for Australian ironbark timber: optimisation and experimental validation. *Drying Technology*, 15(1):47–70, 1997.

[58] Jay H. Lee and Lawrence N. Ricker. Extended Kalman filter based nonlinear model predictive control. In *Proceedings of the American Control Conference*, pages 1895–1899, San Francisco, CA, June 1993.

[59] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Preprint 2001-23, Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, D-69120 Heidelberg, Germany, 2001.

[60] Markus Lendl, Udo H. Schwarz, Hans-Joachim Romeiser, Rolf Unbehauen, Michael Georgieff, and Götz F. Geldner. Nonlinear model-based predictive control of non-depolarizing muscle relaxants using neural networks. *Journal of Clinical Monitoring and Computing*, 15(5):271–278, 1999.

[61] Xingfei Li and Shaojun Xiao. Voluntary movement control of human elbow joint based on neural network predictive model. *Critical Reviews in Biomedical Engineering*, 26(5):418–419, 1998.

[62] Cheng Ling and Thomas F. Edgar. Experimental verification of model-based fuzzy gain scheduling technique. In *Proceedings of the American Control Conference*, pages 2475–2480, Baltimore, MD, June 1994.

[63] Cheng Ling and Thomas F. Edgar. Real-time control of a water-gas shift reactor by a model-based fuzzy gain scheduling technique. *Journal of Process Control*, 7(4):239–253, 1997.

[64] L. Magni, G. De Nicalao, L. Magnani, and R. Scattolini. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica*, 37(9):1351–1362, 2001.

[65] Bryon R. Maner and Francis J. Doyle III. Polymerization reactor control using autoregressive-plus Volterra-based MPC. *AIChE Journal*, 43(7):1763–1784, 1997.

[66] Byron R. Maner, Francis J. Doyle III, Babatunde A. Ogunnaike, and Ronald K. Pearson. Nonlinear model predictive control scheme using second order Volterra models. In *Proceedings of the American Control Conference*, pages 3253–3257, Baltimore, MD, June 1994.

[67] V. Manousiouthakis and D. Chmielewski. On constrained infinite-time nonlinear optimal control. Submitted for publication in Chem. Eng. Sci., 2000.

[68] David Q. Mayne and Hannah Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, July 1990.

[69] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optim.*, 2(1):575–601, 1992.

[70] Hannah Michalska and David Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, 1993.

[71] R. H. Miller, I. Kolmanovsky, E. G. Gilbert, and P. D. Washabaugh. Control of constrained nonlinear systems: a case study. *IEEE Control Systems Magazine*, 20(1):23–32, 2000.

[72] R. H. Miller, I. Kolmanovsky, E. G. Gilbert, and P. D. Washabaugh. Control of constrained nonlinear systems: a case study. *IEEE Control Systems Magazine*, 20(1):23–32, 2000.

[73] Manfred Morari and Jay H. Lee. Model predictive control: The good, the bad, and the ugly. In Y. Arkun and W. H. Ray, editors, *Chemical Process Control—CPC IV*, pages 419–444, Amsterdam, 1991. Fourth International Conference on Chemical Process Control, Elsevier.

[74] H. E. Musch, G. W. Barton, T. A. G. Langrish, and A. S. Brooke. Nonlinear model predictive control of timber drying. *Computers & Chemical Engineering*, 22(3):415–425, 1998.

[75] Kenneth R. Muske and Thomas A. Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12(5):617–632, 2002.

[76] Kenneth R. Muske and James B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.

[77] Kenneth R. Muske and James B. Rawlings. Nonlinear moving horizon state estimation. In Ridvan Berber, editor, *Methods of Model Based Process Control*, Nato Advanced Study Institute series: E Applied Sciences 293, pages 349–365, Dordrecht, The Netherlands, 1995. Kluwer.

[78] Rajendra K. Mutha, William R. Cluett, and Alexander Penlidis. Nonlinear model-based predictive control of control nonaffine systems. *Automatica*, 33(5):907–913, 1997.

[79] Zoltan Nagy, Rolf Findeisen, Moritz Diehl, Frank Allgöwer, H. Georg Bock, Serban Agachi, Johannes P. Schlöder, and Daniel Leineweber. Real-time feasibility of nonlinear predictive control for large scale processes – a case study. In *Proceedings of the American Control Conference*, pages 4249–4253, Chicago, IL, June 2000.

[80] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 1999.

[81] C. Onnen, R. Babuska, U. Kaymak, J. M. Sousa, H. B. Verbruggen, and R. Isermann. Genetic algorithms for optimization in predictive control. *Control Engineering Practice*, 5(10):1363–1372, 1997.

[82] Ashutosh A. Padwardhan and Thomas F. Edgar. Nonlinear model predictive control of a packed distillation column. *Industrial and Engineering Chemistry Research*, 32(10):2345–2356, 1993.

[83] Ashutosh A. Padwardhan, Glenn T. Wright, and Thomas E. Edgar. Nonlinear model-predictive control of distributed parameter systems. *Chemical Engineering Science*, 47(4):721–735, 1992.

[84] Gabriele Pannocchia and James B. Rawlings. Disturbance models for offset-free MPC control. Accepted for publication in *AIChE Journal*, 2002.

[85] T. Parisini and R. Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

[86] Robert S. Parker. Efficient nonlinear model predictive control: exploiting the Volterra-Laguerre model structure. In James B. Rawlings, Babatunde A. Ogunnaike, and John W. Eaton, editors, *Chemical Process Control– VI: Sixth International Conference on Chemical Process Control*, volume 97, Tucson, Arizona, January 2001. AIChE Symposium Series.

[87] Sachin C. Patwardhan and K. P. Madhavan. Nonlinear model predictive control using second-order model approximation. *Industrial and Engineering Chemistry Research*, 32(2):334–344, 1993.

[88] T. Peterson, E. Hernandez, Y. Arkun, and F. J. Schork. Nonlinear predictive control of a semibatch polymerization reactor by extended DMC. In *Proceedings of the 1989 American Control Conference*, pages 1534–1539, June 1989.

[89] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, editor, *Dundee Conference on Numerical Analysis*, volume 7, Dundee, June 1977. Springer-Verlag.

[90] Thomas Proll and Nazmul M. Karim. Model-predictive pH control using real-time NARX approach. *AIChE Journal*, 40(2):269–282, 1994.

[91] Mark L. Psiaki and Kihong Park. Trajectory optimization for real-time guidance: Part 1, time-varying LQR on a parallel processor. In *Proceedings of the 1990 American Control Conference*, pages 248–253, May 1990.

[92] M. N. Ramesh, M. A. Kumar, and P. N. Srinivasa Rao. Application of artificial neural networks to investigate the drying of cooked rice. *Journal of Food Process Engineering*, 19(3):321–329, 1996.

[93] Christopher V. Rao. *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems.* PhD thesis, University of Wisconsin–Madison, 2000.

[94] Christopher V. Rao and James B. Rawlings. Optimization strategies for linear model predictive control. In *Proceedings of the 1998 IFAC DYCOPS Symposium, Corfu, Greece*, 1998.

[95] Christopher V. Rao and James B. Rawlings. Steady states and constraints in model predictive control. *AIChE Journal*, 45(6):1266–1278, 1999.

[96] Christopher V. Rao, James B. Rawlings, and Jay H. Lee. Constrained linear state estimation – a moving horizon approach. *Automatica*, 37(10):1619–1628, 2001.

[97] Christopher V. Rao, James B. Rawlings, and David Q. Mayne. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. Submitted for publication in *IEEE Transactions on Automatic Control*, 2000.

[98] Christopher V. Rao, Stephen J. Wright, and James B. Rawlings. On the application of interior point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.

[99] Wolfgang Rauch and Poul Herremoës. Genetic algorithms in real time control applied to minimize transient pollution from urban wastewater systems. *Water Research*, 33(5):1265–1277, 1999.

[100] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, 1978.

[101] N. L. Ricker and J. H. Lee. Nonlinear model predictive control of the Tennessee Eastman challenge process. *Computers & Chemical Engineering*, 19(9):961–981, 1996.

[102] N. Lawrence Ricker. Decentralized control of the Tennessee Eastman challenge process. *Journal of Process Control*, 6(4):205–221, 1996.

[103] D. G. Robertson and J. H. Lee. A least squares formulation for state estimation. *Journal of Process Control*, 5(4):291–299, 1995.

[104] Douglas G. Robertson and Jay H. Lee. On the use of constraints in least squares estimation and control. *Automatica*, 38(7):1113–1124, 2002.

[105] Douglas G. Robertson, Jay H. Lee, and James B. Rawlings. A moving horizon-based approach for least-squares state estimation. *AIChE Journal*, 42(8):2209–2224, August 1996.

[106] Louis P. Russo and Robert E. Young. Moving horizon state estimation applied to an industrial polymerization process. In *Proceedings of 1999 American Control Conference, San Diego, California*, 1999.

[107] Pierre O. M. Scokaert, David Q. Mayne, and James B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, March 1999.

[108] F. G. Shinskey. *Feedback Controllers for the Process Industries*. McGraw-Hill, New York, 1994.

[109] Masoud Soroush and Costas Kravaris. Short horizon nonlinear model predictive control. In *Proceedings of the 1995 IEEE Conference on Control Applications*, pages 943–948, Albany, NY, USA, 1995.

[110] G. H. Staus, L. T. Biegler, and B. E. Ydstie. Adaptive control via non-convex optimization. In C. Floudas and P. Pardolas, editors, *Nonconvex optimization and its applications*, pages 1–19. Kluwer, 1996.

[111] Thorsten Stuetzle, Nathan Blair, William A. Beckman, and John W. Mitchell. Use of linear predictive control for a solar electric generating system. In *Proceedings of System Simulation in Buildings*, Liège, December 2002.

[112] Thorsten A. Stuetzle. Automatic control of the 30 MWe SEGS VI parabolic trough plant. Master's thesis, University of Wisconsin–Madison, 2002.

[113] Kwaku O. Temeng, Phillip D. Schnelle, and Thomas J. McAvoy. Model predictive control of an industrial packed bed reactor using neural networks. *Journal of Process Control*, 5(1):19–27, 1995.

[114] Matthew J. Tenny and James B. Rawlings. State estimation strategies for nonlinear model predictive control. In *AIChE Annual Meeting, Reno, Nevada*, 2001.

[115] Matthew J. Tenny and James B. Rawlings. Efficient moving horizon estimation and nonlinear model predictive control. In *Proceedings of the American Control Conference*, pages 4475–4480, Anchorage, Alaska, May 2002.

[116] Matthew J. Tenny, James B. Rawlings, and Rahul Bindlish. Feasible real-time nonlinear model predictive control. In James B. Rawlings, Babatunde A. Ogunnaike, and John W. Eaton, editors, *Chemical Process Control– VI: Sixth International Conference on Chemical Process Control*, volume 97, Tucson, Arizona, January 2001. AIChE Symposium Series.

[117] Matthew J. Tenny, James B. Rawlings, and Stephen J. Wright. Closed-loop behavior of nonlinear model predictive control. Submitted for publication in *AIChE J*, September 2002.

[118] Matthew J. Tenny, Stephen J. Wright, and James B. Rawlings. Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming. Submitted for publication in *Comp. Optim. Appl.*, August 2002.

[119] Y. A. Thomas. Linear quadratic optimal estimation and control with receding horizon. *Electronics Letters*, 11:19–21, January 1975.

[120] Zlatko Trajanoski, Werner Regittnig, and Paul Wach. Simulation studies on neural predictive control of glucose using the subcutaneous route. *Computer Methods and Programs in Biomedicine*, 56(2):133–139, 1998.

[121] A. Uppal, W. H. Ray, and A. B. Poore. On the dynamic behavior of continuous stirred tank reactors. *Chemical Engineering Science*, 29:967–985, 1974.

[122] J. von Neumann. Various techniques used in connection with random digits. *US Nat. Bur. stand. appl. Math.*, Ser. 12:36–38, 1951.

[123] Philip A. Wisnewski and Francis J. Doyle III. A reduced model approach to estimation and control of a Kamyr digester. *Computers & Chemical Engineering*, 20(Suppl. pt. B):S1053–S1058, 1996.

[124] Glenn T. Wright, Terence Breedijk, and Thomas F. Edgar. On-line parameter estimation and adaptation in nonlinear model-based control. In *Proceedings of the American Control Conference*, pages 2782–2787, Boston, MA, June 1991.

[125] S. J. Wright and M. J. Tenny. A feasible trust-region sequential quadratic programming algorithm. Submitted for publication in *SIAM Optim.*, August 2002.

[126] Stephen J. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, 1997.

[127] Jie Yu, Ali Jadbabaie, James Primbs, and Yun Huang. Comparison of nonlinear control design techniques on a model of the Caltech ducted fan. *Automatica*, 37(12):1971–1978, 2001.

[128] Jie Zhang and Julian A. Morris. Recurrent neuro-fuzzy networks for nonlinear process modeling. *IEEE Transactions on Neural Networks*, 10(2):313–326, 1999.

[129] Alex Zheng. Computationally efficient nonlinear MPC algorithm. In *Proceedings of the American Control Conference*, pages 1623–1627, Albuquerque, NM, June 1997.

Vita

Matthew Jeremy Tenny was born in Arlington Heights, Illinois on January 9, 1975. He graduated from Buffalo Grove High School in May 1993 and began studying at Rice University that fall. He graduated *magna cum laude* in 1997 with a Bachelor of Science degree in Chemical Engineering and a Bachelor of Arts degree in Computational and Applied Mathematics. He began his graduate studies in the fall of 1997 in the research group of James B. Rawlings in the Department of Chemical Engineering at the University of Wisconsin-Madison.

Permanent Address: 2919 Windsor Drive

Arlington Heights, IL 60004

This dissertation was prepared with LaTeX 2_ε[1] by the author.

[1]This particular University of Wisconsin compliant style was carved from The University of Texas at Austin styles as written by Dinesh Das (LaTeX 2_ε), Khe–Sing The (LaTeX), and John Eaton (LaTeX). Knives and chisels wielded by John Campbell and Rock Matthews.