RICE UNIVERSITY

Robust Model Predictive Control as a Class of Semi-Infinite Programming Problems

by

Dean E. Kassmann

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree
Doctor of Philosophy

Approved, Thesis Committee:

[illegible]

Thomas Badgwell, Chairman
Assistant Professor of Chemical Engineering

[illegible]

Robert Hawkins
Vice President, APC Technology
Aspen Technology, Inc.

[illegible]

Matthias Heinkenschloss
Associate Professor of Computational and Applied Mathematics

[illegible]

George Hirasaki
A. J. Hartsook Professor in Chemical Engineering

[illegible]

Richard Tapia
Noah Harding Professor of Computational and Applied Mathematics

Houston, Texas
April, 1999

Robust Model Predictive Control as a Class of Semi-Infinite Programming Problems

by

Dean E. Kassmann

Abstract

This thesis introduces a new interpretation of the problems arising in robust model predictive control (MPC). In practice, MPC algorithms are typically embedded within a multi-level hierarchy of control functions. The MPC algorithm itself is usually implemented in two pieces: a steady-state target calculation followed by a dynamic optimization. It is shown in this thesis that some of the most promising methods of imparting robustness to MPC algorithms result in semi-infinite programs. These programs arise from the addition of semi-infinite constraints to the nominal MPC algorithms which come from theoretical arguments that guarantee stability of the closed loop system or from requiring existing constraints to hold for an infinite set of plants. While the number of constrained variables is finite, the constraint must hold over an infinite set. This infinite set corresponds to a continuous uncertainty description for the model parameters.

In this dissertation it is also shown that the resulting optimization problems have a very unique structure. For some MPC algorithms the semi-infinite program (SIP) can be cast as an equivalent finite-dimensional nonlinear convex program. Primal-dual interior-point methods are used to efficiently solve the resulting optimization problem by exploiting its inherent convexity. Simulation examples illustrate the effects of uncertainty on nominal MPC algorithms and demonstrate the advantages of interior-point methods.

Acknowledgements

First, I would like to dedicate this dissertation to my father.

I could not have written this dissertation without the love and support of my family. Their continuous encouragement has made this dissertation possible.

My utmost gratitude goes to my advisor, Professor Tom Badgwell. Throughout my time at Rice, I could not have had a better mentor. Tom has shared his ideas and deep understanding of control theory and mathematics. He has had a great influence upon me. He is a compassionate and kind person and a great friend.

I would like to thank all the members of my thesis committee for their time and support. I am especially grateful to Professor Richard Tapia. I will never forget how much I learned in his courses and through our talks. I thank Robert Hawkins for many exciting and stimulating discussions. If it were not for him, this dissertation would likely contain quite different material. I thank Professor Heinkenschloss for his many helpful comments. Our talks helped frame a majority of this thesis. I thank Professor Hirasaki for his helpful suggestions and comments.

I would like to acknowledge Sameer Ralhan and Raymond Joe with whom I have spent my time here at Rice. Our many discussions and their insight into other aspects not presented in this dissertation is greatly appreciated.

Finally, I would like to acknowledge the financial support of The National Science Foundation, The Texas Higher Education Coordinating Board, and Aspen Technology, Inc. for making my work possible.

Contents

Illustrations

Tables

Chapter 1

Introduction

Model Predictive Control (MPC) represents the leading edge of control technology and is regarded by many as one of the most important developments in process control. MPC refers to a class of algorithms that compute a sequence of manipulated variable adjustments (control moves) in order to optimize the future behavior of a plant. As its name suggests, MPC uses an explicit mathematical model to predict how a process will evolve in time. The prediction is used to determine the optimal control moves that will bring the process to a desired state. The optimal control moves are the result of an online optimization. In this chapter we provide an introduction to MPC technology and outline the class of robust MPC problems to be studied.

1.1 The Class of Problems

In this dissertation we focus on *robust* model predictive control. The term 'robust' refers to the ability of an MPC algorithm to effectively deal with differences between the controller's internal model and the actual plant or system. Large differences can cause the closed loop system to go unstable or behave erratically. There are different ways to impart robustness or include model uncertainty in an MPC algorithm. We investigate one such method: to append or modify an existing constraint to the algorithm. This new constraint takes a very specific form—it is semi-infinite in nature:

$$h(x;\theta) \leq 0 \qquad \forall \theta \in \mathcal{U}.$$

While the number of constrained variables x is finite, the constraint must hold for all the parameters θ in an infinite set $\mathcal{U}$. While not generally recognized as such, the resulting online optimization takes the form of a semi-infinite program. These constraints arise from theoretical arguments that guarantee stability of the closed loop system by forcing the objective in the online optimization to form a nonincreasing sequence ([4], [3], [46], [79]) or from requiring the existing constraints to hold for an infinite set of plants [47].

Traditional implementations of linear model predictive control (which employ linear models) take the form of linear and quadratic programs. The controller executes approximately every minute, requiring the online solution of approximately 1,400 optimization problems per day. Implementation of model predictive control using nonlinear models is just beginning in industry; these applications take the form of more general nonlinear programs [78]. Their execution frequency depends upon the solution time of the optimization algorithm. The efficient solution of both types of optimization problems is extremely important in an online environment. Efficient solution methods for the linear *nominal* problem have been proposed by several authors [112], [80]. Less attention has been given, however, to developing efficient solution methods for robust MPC problems.

We propose to interpret robust model predictive control as a class of semi-infinite programming problems. We show that one of the most promising methods of imparting robustness to nominal MPC algorithms results in a semi-infinite program (SIP). For some MPC algorithms, the SIP can be cast as an equivalent finite-dimensional nonlinear convex program. To efficiently solve the resulting optimization problem we exploit the convexity through the use of primal-dual interior-point methods.

1.2 The MPC Hierarchy

In modern processing plants, MPC is implemented as part of a multi-level hierarchy of control functions. Figure 1.1 illustrates a representative control hierarchy. At the top level a plant-wide optimizer determines optimal steady-state settings for each unit in the plant. These may be sent to local optimizers at each unit which run more frequently or consider a more detailed unit model than is possible at the plant-wide level. The unit optimizer computes an optimal economic steady-state and passes this information to the MPC algorithm for implementation. The MPC algorithm must move the plant from one constrained steady-state to another while minimizing constraint violations along the way. Figure 1.1 shows that the MPC algorithm can be

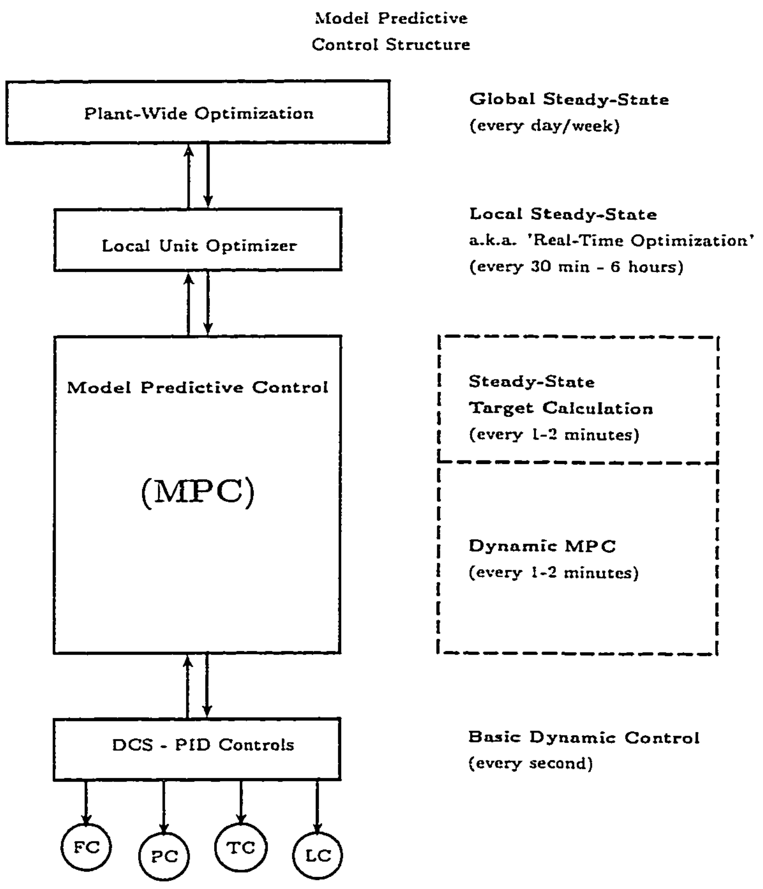
Model Predictive Control Structure
Plant-Wide Optimization
Global Steady-State (every day/week)
Local Unit Optimizer
Local Steady-State a.k.a. 'Real-Time Optimization' (every 30 min - 6 hours)
Model Predictive Control
(MPC)
Steady-State Target Calculation (every 1-2 minutes)
Dynamic MPC (every 1-2 minutes)
DCS - PID Controls
Basic Dynamic Control (every second)
FC
PC
TC
LC

Figure 1.1: MPC Control Hierarchy

further divided into a steady-state calculation and a dynamic calculation [20]. The dynamic MPC calculation has been studied extensively. Review papers by García et al. [36], Ricker [88], Morari and Lee [63], Muske and Rawlings [64], and Rawlings et. al [82] detail MPC theoretical issues. The papers by Mayne [57] and Lee [52] summarize the very latest technical developments in MPC control theory, and Froisy provides a vendor's perspective on industrial MPC technology, summarizing likely future developments [32]. Qin and Badgwell give a historical review of the industrial MPC algorithms and applications [77] [78].

The goal of the steady-state target algorithm is to recalculate the targets from the local optimizer every time the MPC controller executes. This must be done because disturbances entering the system or new input information from the operator may change the location of the optimal steady-state. This is a standard idea that dates back to the early days of optimal control theory (see e.g. [50]). The first reference to this idea in the context of MPC was by Cutler, Morshedi and Haydel [19]. Today, this separation is a common part of industrial MPC technology. Since the steady-state algorithm must run more frequently than the local optimizer, it uses a less detailed model. In practice, it uses a steady-state version of the dynamic model used for the dynamic optimization. The recalculated optimal steady-state is then passed to the dynamic MPC algorithm which determines how to best go from one constrained steady-state to the next. This steady-state target optimization is an alternative way to incorporate feedback into the MPC algorithm without explicitly including it in the dynamic MPC calculation.

Some MPC implementations combine the two calculations into a single optimization, solving for both the input profile and steady-state targets simultaneously. The advantage is that only one optimization problem must be solved at every controller execution. The disadvantage is that the steady-state objectives may conflict. The steady-state objective is often cast in terms of economics. For example, the goal may be to minimize the heat duty of a column or to maximize the throughput of a reactor. The dynamic objective is to move toward that endpoint, minimizing deviations and adhering to constraints along the way. These two objectives may not always be consistent. The controller may try to maintain one while giving up on the other. Additionally, when the two optimizations are combined, tuning becomes nontrivial as the control engineer must try to find tuning constants which give both satisfactory dynamic *and* steady-state performance.

In this work we assume that the MPC calculation is separated into dynamic and steady-state calculations. The steady-state portion is assumed to be driven by economics.

1.3 Robust MPC

In practice, the process model used by an MPC controller is never perfect. Modeling errors may be due to poor design or execution of test signals, invalid modeling assumptions or changes in plant operation. The controller, which relies on the model to predict the future plant behavior, can cause the closed loop system to go unstable for sufficiently large modeling errors.

Robust MPC algorithms are those which are modified in some way so as to better tolerate modeling errors. Two important concepts in robust MPC are: robust performance and robust stability. By robust performance we mean the ability to achieve acceptable control performance in the presence of model uncertainty. Robust stability is a weaker requirement which ensures stability in the presence of modeling errors. Robust stability is the goal of most robust algorithms presented in the literature; however, in an industrial setting control practitioners will actually require robust performance.

For the dynamic algorithm there are two general ways to impart robustness: detune the nominal controller or alter the MPC algorithm itself. These alterations result in additional constraints or a modified objective function for the optimization problem. Consider an objective function J that depends upon the current measured state x_k, the input sequence u_j, and model parameters $\theta \in \Omega$. Robust MPC formulations can be divided into four general classes ([58], [52], [57]):

- **Minimize the worst-case controller cost:** These algorithms, commonly known as *Min-Max* algorithms, minimize the worst case controller cost for a set of possible plants. This corresponds to replacing the objective of the nominal problem with

$$\min_{u_j} \left(\max_{\theta \in \Omega} J_i(x_k, u_j\, ; \theta) \right) .$$

The objective function corresponding to the worst case plant model is minimized at each sample time. An excellent overview of Min-Max methods can be found in the paper by Lee and Yu [54].

- **Detuning:** This method is motivated by the fact that one can always stabilize a stable plant by making the controller less aggressive. Detuning is accomplished by changing the tuning parameters of the nominal MPC algorithm so as to suppress input movement.

- **State constraints:** These algorithms penalize the states of the problem in some specific way. **State contraction constraints** [116] force all possible plant states to contract on a finite horizon. This is equivalent to appending the following constraint

$$||x_{k+N}(u_j, \theta)|| \leq \lambda \; ||x_k||, \qquad \forall \theta \in \Omega$$

for $0 < \lambda < 1$ to the nominal problem. **Terminal state constraints** [94] require the largest possible terminal state to lie within an invariant set, which is equivalent to appending

$$\max ||x_{k+N}(u_j, \theta)|| \in W_\alpha \qquad \forall \theta \in \Omega$$

to the nominal problem.

- **Cost function bounds:** These algorithms prevent an increase in the objective function or controller cost for all possible plants. Constraints of the form

$$J(x_k, u_j; \theta) \leq f \qquad \forall \theta \in \Omega$$

are appended to the nominal problem. Papers by Badgwell, [4] and [3], explain the use of cost function bounds for both linear and nonlinear stable plants.

These four classes are rather general. We focus on the last: cost function bounds. Of the methods, cost function bounds have the least trade-off in robustness vs. performance; i.e., they suffer the least performance penalty in the process of providing robust control. For continuous uncertainty descriptions, the constraint is not finite but rather infinite in nature:

$$J(x, u; x_k, \theta) \leq f, \qquad \forall \theta \in \mathcal{U}.$$

To date there has been very little study of the steady-state MPC algorithm. While there exist multi-stage dynamic calculations ([53], [54], [79]), there is no existing robust analysis of the steady-state target calculation.

1.4 Organization of the Thesis

Chapter 2 reviews the history of MPC and describes the dynamic and steady-state MPC algorithms in greater detail. Chapter 3 presents robust forms of the algorithms and shows how they can be cast as semi-infinite programs. In chapter 4 we present efficient solution methods for both problems. We show that the robust steady-state optimization problem can be cast as a second-order cone program, while efficient successive quadratic programming techniques can be used for the dynamic problem. Primal-dual interior-point methods are used to exploit the inherent convexity of both problems. Chapter 5 contains simulation examples, and Chapter 6 summarizes the most important conclusions and open questions.

Chapter 2

Nominal MPC

When the model used in MPC is assumed to describe the system dynamics *perfectly*, the controller is said to be a *nominal* MPC algorithm. In this chapter we describe both nominal dynamic and steady-state MPC algorithms and present a brief history of model predictive control.

2.1 The Evolution of MPC

While model predictive control can trace its origins back to the work of Kalman in the 1960s, it was really only widely introduced as an industrial control technology in the early 1980s. Over the years it has been fine-tuned to meet the needs of industrial practitioners. MPC technology was first developed to satisfy the control requirements of petrochemical plants. A typical plant consists of several continuous process units, each with the following characteristics:

- Multiple process inputs and outputs (multi-variable, interactive)
- Difficult dynamics (large dead time, inverse response, zero gain, nonlinear)
- Non-square, time-varying structure
- Stream compositions unknown and variable
- Highly constrained operating region

- Tight product specifications

Obstacles like these led researchers and engineers to develop MPC technology in the early 1970s. As of the end of 1995, there were over 2,200 reported applications using MPC [77]. These included refining, petrochemicals, pulp and paper, gas, mining/metallurgy, food processing, aerospace/defense, and automotive applications. In this section we present a brief chronological history of model predictive control technology. This presentation parallels that found in the paper by Qin and Badgwell [77].

2.1.1 Kalman

MPC can trace its origins back to the work of Kalman in 1960 [44]. Kalman was interested in determining when a linear control system can be considered to be optimal. He used a linear state-space model to describe plant dynamics:

$$x_{j+1} = Ax_j + Bu_j, \tag{2.1a}$$

$$y_j = Cx_j. \tag{2.1b}$$

The vector u refers to the process inputs; the vector x refers to the process states and the vector y refers to the process outputs. The model is discrete, deterministic, and linear. One of the most powerful aspects of this approach is that by changing the coefficient matrices A, B, and C, one can describe any linear process.

The controller Kalman designed minimizes a quadratic objective function that penalizes squared input and state deviations from the origin,

$$J = \sum_{j=0}^{\infty} \left(x_j^T Q\, x_j\right) + u_j^T R\, u_j\right) . \tag{2.2}$$

It is implicit in this formulation that all variables are written in terms of deviations from a desired steady-state. This problem is known as the steady-state infinite horizon Linear Quadratic Regulator (LQR). Kalman was able to find the *analytic* solution of the problem under suitable conditions. It is a proportional controller with a single gain matrix, K, determined by a the solution of a Ricatti matrix differential equation. The controller is given by

$$u_k = Kx_k. \tag{2.3}$$

It is important to note that the performance objective (2.2) has an infinite time horizon. This imparts strong stabilizing properties to the LQR algorithm [44]. One can show the LQR algorithm is stabilizing for any reasonable plant (stabilizable and detectable) as long as the objective function weight matrices, Q and R, are positive definite.

Even though the LQR provides an elegant and powerful solution to the problem of controlling an unconstrained linear plant, it was not received well in the chemical process industry. Industrialists were interested in the incorporation of process constraints and nonlinearities, neither of which are addressed by the Kalman LQR. There was also the question of what to do with uncertainty in the process model (robustness). Educational and cultural differences between industrial control practitioners and the academic community also prevented widespread use of the LQR controller. Technicians and engineers either had no exposure to LQR theory or regarded the concepts as impractical [77], [86], [36].

This led to an approach in industry of a new model-based control methodology in which a dynamic optimization problem would be solved repeatedly at each sample time. This idea was certainly not new; Lee and Markus [51], for example, anticipated modern MPC practice in their 1967 optimal control text:

> *One technique for obtaining a feedback controller synthesis from knowledge of open loop controllers is to measure the current control process state and then compute very rapidly for the open loop control function. The first portion of the function is then used during a short time interval, after which a new measurement of the function is computed for this new measurement. The procedure is then repeated.*

2.1.2 IDCOM & DMC

At a 1976 conference [87] and later in a 1978 *Automatica* paper [86], Richalet et al. first described what they called Model Predictive Heuristic Control (MPHC). They used a linear *impulse response model* to describe the plant. By making a step input or impulse input to a plant and recording the resulting change in output, it is simple, in principle, to extract the parameters for such a model. They used a quadratic performance objective function, similar to the LQR, but defined over a *finite* horizon. The desired future path of the plant output was specified by a reference trajectory instead of a set constant value. They included both input and output constraints in the formulation.

The solution software was named IDCOM, an acronym for Identification and Command. Optimal inputs were calculated using a heuristic iterative algorithm.

Independently, engineers at Shell Oil Corp. developed their own MPC technology in the early 1970s. Cutler and Ramaker presented their Dynamic Matrix Control (DMC) at the 1979 American Institute of Chemical Engineers (AIChE) conference [20]. This was an unconstrained multi-variable control algorithm that used a linear step response model for the plant and a quadratic objective function with a finite prediction horizon for the cost function. Using a linear step response model one can express the predicted future outputs of the plant as a linear combination of the future inputs. The matrix that ties the two together is what they called the *Dynamic Matrix*.

The DMC controller minimizes the differences in the predicted output and its set-point in a least-squares sense with penalties on manipulated variable movement (move suppression). Prett and Gillette [76] describe an early application of DMC to a fluid catalytic cracking unit. They implemented the control system in a multi-level hierarchy similar to that described previously in which constrained steady-state optimization and dynamic control are separated.

IDCOM and DMC represent the first generation of MPC algorithms. These focus mainly on decoupling multi-variable dynamics and treat constraints in a sub-optimal, heuristic manner.

2.1.3 QDMC

In a 1983 conference paper, Cutler et al. [19] described the Quadratic DMC algorithm QDMC. This formulation finds the optimal dynamic solution subject to input and output constraints. A more comprehensive description of the QDMC algorithm was given later by García and Morshedi [35]. QDMC retains the same key features of the DMC algorithm but provides a systematic way to find the optimal constrained solution. The resulting optimization takes the form of a convex Quadratic Program (QP). QDMC represents a second generation of MPC technology, which provides a systematic way to implement input and output constraints.

2.1.4 IDCOM-M & HEICON

As the second generation of MPC technology was implemented on larger and more complex problems, control engineers faced a new set of obstacles. While QDMC provided an excellent way to handle input and output constraints, it did not address problems created by large disturbances and changes in the degrees of freedom available to the controller. These problems can lead to an infeasible optimization problem.

One solution involves relaxing the output constraints. If an output constraint is *strictly enforced*, it is referred to as a *hard* constraint. If output constraint violations are allowed, the constraint is referred to as a *soft* constraint. It is common to minimize the amount of violation by adding a penalty term to the objective function. The flexibility provided by soft output constraints allows the QP to always be feasible. The disadvantage, however, is that output constraints may always be violated during normal operation.

An alternative solution is to allow constraints to be ranked in order of priority or importance. If the QP is infeasible constraints are dropped one by one until feasibility is attained. The order in which this *shedding* occurs is determined from a specified ranking priority.

These and other issues were addressed in a third generation of MPC technology. Setpoint, Inc. and Adersa developed a new version of the IDCOM algorithm, known as IDCOM-M, first described in a paper by Grosdidier et al. [40]. A second paper appeared in 1990 describing an application of the IDCOM-M algorithm to the Shell Fundamental Control Problem [33]. This paper provided additional details on constraint handling. The IDCOM-M algorithm employs a two-stage optimization, one for the outputs and then, if there are extra degrees of freedom, one for the inputs.

Other third generation MPC algorithms include the PCT algorithm sold by Profimatics, the RMPCT controller by Honeywell, and the PFC algorithm developed by Adersa. This generation distinguishes between several levels of constraints (hard, soft, ranked), provides some mechanism to recover from an infeasible solution, addresses the issues resulting from a control structure that changes in real time, and allows for a wider range of process dynamics and controller specifications.

2.1.5 Muske/Rawlings

Today, companies such as Aspen Technology and Honeywell provide MPC technology building on the foundations described above. Academic researchers have summarized limitations of this technology and have proposed several promising solutions. Key limitations include restrictive model forms and lack of strong stability guarantees. Muske and Rawlings [64] addressed these issues by building directly on the firm foundation provided by Kalman's LQR. By adding constraints to Kalman's LQR, they were able to develop an MPC algorithm that addresses industrial needs while retaining strong stability properties.

The key idea of Muske and Rawlings is to only consider a finite number of inputs, with the remaining inputs set to the origin. This allows the infinite sum in the cost function to be represented as a finite sum with a different weight on the final state. They showed the controller was stabilizing for all choices of linear plants, provided the model is perfect and the initial QP has a feasible solution. Two years later, in 1995, Meadows et al. extended this approach to include a large class of nonlinear plants and cost functions [58].

2.2 Model Forms

The models used in MPC algorithms can take many forms. We focus here on state-space and step/impulse response models. The former is favored by the academic community while the latter type is used extensively in MPC applications. State-space models are generally fundamental models arising from basic conservation laws (e.g. conservation of mass, momentum, and energy). Step or impulse response models are, as their name suggests, the result of step or impulse response tests in which the input to a unit is changed and the response is written down. This results in a straightforward, practical model for the behavior of a unit. Step and impulse response models are linear. By their very nature, they only capture stable processes. State-space models, because they are derived from differential equations, can be nonlinear and can capture both stable and unstable behavior.

2.2.1 State-Space Models

Discrete time state-space models generally take the form of a recurrence relation

$$\begin{aligned} x_j &= f(x_{j-1}, u_{j-1}), \\ y_j &= g(x_j). \end{aligned}$$

The recurrence relation usually results from some type of discretization of a set of differential equations describing the process. The models can be linearized about a steady-state or left nonlinear. Linear models are most often used in practice because they are more tractable in a real-time setting. In this case a general linear state-space model has the following form:

$$\begin{aligned} x_j &= Ax_{j-1} + Bu_{j-1}, \\ y_j &= Cx_j, \end{aligned}$$

where the matrices A, B, and C are obtained by linearizing about a steady-steady (x_s, u_s):

$$A = \left.\frac{\partial f}{\partial x}\right|_{x_s,u_s}, \quad B = \left.\frac{\partial f}{\partial u}\right|_{x_s,u_s}; \quad C = \left.\frac{\partial g}{\partial x}\right|_{x_s}.$$

The issue of how to best construct a discrete-time approximation to a continuous-time optimal-control problem is closely related to the numerical methods used in the solution of differential equations, i.e. numerical integration. This is a straightforward observation since most problems of interest require some numerical technique to solve. Initial approximations used simple finite difference or Euler methods to discretize the problem [12], [16]. Later Runga-Kutta techniques and collocation at Gauss points were used [17], [68], [83], [108]. Recent work done by Polak and others lays a framework for a general theory of consistent approximations for optimal control problems [71], [93].

Another important issue is the resulting structure of the discretized equations. The goal is to obtain discretizations with "local support" as these lead to block-banded linear systems. One way to achieve local support is to use higher order integration within a single time interval and enforce matching at interval boundaries. Santos et al. discuss this approach and show that it leads to a more numerically stable solution technique [89]. A higher order technique that offers local support and is well accepted by the chemical engineering community is collocation at Gauss points or collocation on finite elements [17]. The method is based on numerical integration using Gaussian quadrature; more information can be found in [9], [18], [107].

2.2.2 Convolution Models

Convolution models are used extensively in MPC applications. They are popular because:

- They can represent any stable linear plant without assuming a model order
- They can represent multivariable plants easily by superposition

- Model predictions are linear in the parameters
- In principle, the model coefficients can be determined graphically from a single step test

Convolution models most commonly appear in one of two forms: *impulse response models* and *step response models.*

Impulse Response Model

The impulse response model derives its name from considering the output behavior $y(t)$ for a pure impulse input. For a single-input, single-output, continuous time system, the Laplace convolution theorem yields

$$y(t) = \int_0^\infty g(\tau)u(t-\tau)d\tau.$$

where the function $g(t)$ is called the impulse response and $u(t)$ is the input profile. The discrete time approximation to the above equation is:

$$y_k = \sum_{i=1}^{\infty} h_i u_{k-i},$$

where the index k is the sample index. The coefficients h_i in the model are a function of $g(t)$ and the sample time T:

$$h_i = \int_{(i-1)T}^{iT} g(\tau)d\tau.$$

The infinite sum above is truncated for practical applications to yield the following finite impulse response model:

$$y_k = \sum_{i=1}^{N} h_i u_{k-i} \qquad h_N = 0.$$

For a multiple input, multiple output (MIMO) system, the model is given by a double summation. Let n be the number of inputs, m be the number of outputs, and let N be the number of model coefficients; then the model can be written as:

$$y^l_{k+j} = \sum_{j=1}^{n} \left(\sum_{i=1}^{N} h_i^{l,j} u^j_{k-i} \right) \qquad \forall\, l = 1, \ldots, m.$$

Here y_k^l is the lth output at time k. In impulse response models the current output depends upon a weighted sum of the past inputs.

Step Response Model

The DMC [20] and QDMC algorithms [35] make use of step response models, for which the output prediction is a function of the input changes δu_k, referred to as input moves:

$$\delta u_k = u_k - u_{k-1}.$$

The step response model is the integral of the impulse response model. Let N be the prediction horizon; then a MIMO discrete time step response model can be written as:

$$y_k^l = \sum_{j=1}^{n} \left(\sum_{i=1}^{N-1} a_i^{l,j} \delta u_{k-i}^j + a_N^{l,j} u_{k-N}^j \right), \qquad \forall\, l = 1, \ldots, m. \tag{2.4}$$

Equation (2.4) relates the lth output at time k to the past N inputs. It can be cast in matrix form as follows:

$$y = A_d \delta u + B_d \delta u_p + C_d u_p. \tag{2.5}$$

The matrix A_d is called the *Dynamic Matrix* because it relates the future predicted outputs to the future computed moves. The matrix B_d relates the future outputs to the past known moves δu_p. The matrix C_d relates the future outputs to the past known inputs u_p. The matrices A_d, B_d, and C_d are defined in Appendix A. Because the model tracks all future outputs and their relation to past inputs, the matrices can become very big. This results in a dynamic calculation which becomes quite large with increasing problem size.

2.3 Nominal MPC algorithms

MPC is an optimal-control based method in that it determines the optimal control adjustments by minimizing an objective function. The objective function typically penalizes deviations of the system states and inputs from desired targets. The states and the inputs are related through a process model. The theoretical framework which has developed around MPC does not depend on the particular model form and allows for many variations.

2.3.1 The Dynamic Algorithm

The classical MPC controller is as defined as follows: Given the current state of the process, x_k, find the future control moves, u_j, that will minimize the objective function:

$$J = \sum_{j=0}^{N-1} L_j(x_j, u_j) + L_N(x_N), \tag{2.6a}$$

where the predicted states and inputs are related through the process model

$$\begin{aligned} x_j &= Ax_{j-1} + Bu_{j-1}, \\ x_0 &= x_k. \end{aligned} \tag{2.6b}$$

At the same time we must respect input and output constraints

$$\begin{aligned} x &\in \mathcal{X}, \\ u &\in \mathcal{U}. \end{aligned} \tag{2.6c}$$

This description is similar to that of Meadows et al. [58] without a terminal state constraint. The input and state constraints come from process specifications. If another model form is used, equation (2.6b) is replaced by the corresponding model equations.

The idea behind MPC is shown graphically in Figure 2.1. The solid line in the past represents old control moves. The filled circles in the past are old states of the process. The goal is to calculate the predicted control moves (the dotted line) that will drive the predicted states (the open circles) to some predetermined set-point (the horizontal dashed line at the top). In the above equations, k is the current sample time, N is the prediction horizon.

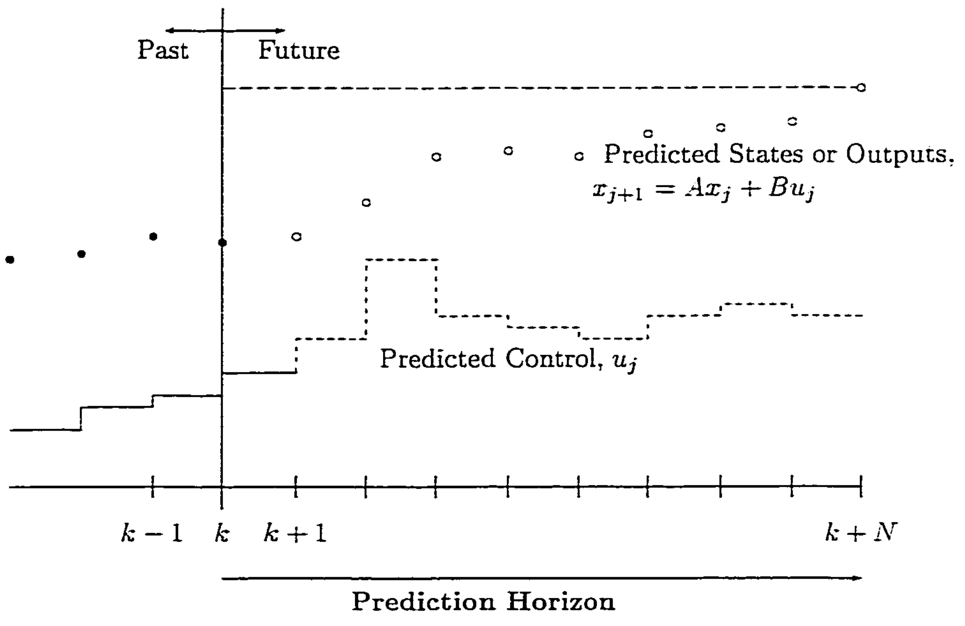

Figure 2.1: The Idea Behind MPC

The objective function (2.6a) is composed of a sum of *costs*, L_j, at each future sampling time that depend upon u_j, the future inputs, and x_j, the future states. In MPC terminology, J is referred to as the *cost function*, and L_j is referred to as the *stage cost function.* This formulation is useful in analyzing the theoretical properties of MPC algorithms; in [58], Meadows et al. outline the properties L_j must possess to guarantee existence of a solution for the MPC algorithm:

1. L is continuous.

2. $L(0, u) \to \infty$ as $||u|| \to \infty$.

3. The input constraint set $\mathcal{U}$, and the state constraint set $\mathcal{X}$ are closed.

4. A feasible u^* exists, which yields a bounded J for bounded x.

The state cost L_j can take any form that satisfies the above requirements, however in practice, it is common to use a quadratic functional,

$$L_j = (x_j - x_s)^T Q\,(x_j - x_s) + (u_j - u_s)^T R\,(u_j - u_s), \tag{2.7}$$

with Q and R positive-definite. Here, x_s and u_s refer to the desired set-point values for the system (generally calculated by the steady-state target calculation, see section 2.3.2). They are the values passed down from the unit optimizer or steady-state MPC optimizer in the control hierarchy mentioned previously. The quadratic form of (2.7) was used by Kalman [44] and is the most commonly used objective. The matrices Q and R are usually diagonal, positive-definite, weight factors that define the relative importance to each of the terms. On occasion a ℓ_1 norm formulation is used instead of the ℓ_2 norm above.

The form of (2.7) is usually simplified to make notation easier. The quantity $x_j - x_s$ is the deviation from the desired set-point. We could rewrite the difference as $\hat{x}_j$, with $\hat{x}_j$ now defined to be the *deviation variable*; however, it is more common to simply drop the circumflex with the understood assumption that the problem is always expressed in terms of deviation variables. So in the following,

$$L_j = x_j^T Q\,x_j + u_j^T R\,u_j, \tag{2.8}$$

it is implicit that all variables are written in terms of deviations from the desired steady state.

In addition to this notation it is not uncommon to represent the quadratic terms of (2.8) as weighted 2-norms of the variables:

$$\|x_j\|_Q^2 \overset{def}{=} x_j^T Q\,x_j. \tag{2.9}$$

When the system has reached the desired steady-state, the deviation variables and thus, the objective function, are identically zero. It is important to note that because Q and R are positive-definite, J is a strictly convex, non-negative quadratic objective function with a unique global minimizer at the origin.

The optimization problem that needs to be solved at each controller execution is:

$$\min_{u} \quad \sum_{j=0}^{N_c-1} \|x_j\|_Q^2 + \|u_j\|_R^2 + \sum_{j=N_c}^{N} + \|x_j\|_Q^2$$

subject to

$$\begin{aligned} x_j &= Ax_{j-1} + Bu_{j-1} \\ x_0 &= x_k, \\ x_j &\in \mathcal{X} \\ u_j &\in \mathcal{U}. \end{aligned} \tag{2.10}$$

The above minimization problem returns N_c optimal control inputs. Of the N_c optimal control moves that are calculated, only the first is implemented. At the next sample time the optimization is repeated with a new current state of the process, x_k.

The above optimization is a quadratic program (QP) when the sets $\mathcal{X}$ and $\mathcal{U}$ are simple bounds. If the model was derived from step response tests, (2.10) would be a variation of Dynamic Matrix Control (DMC) [20] or QDMC [35].

The N_c optimal control inputs u_j, $j = 0, \ldots, N_c$ that are calculated correspond to the length of the *control horizon*. The upper limit in the sum on the outputs is the *prediction horizon* N. The control horizon is the number of time steps into the future for which one wishes to control the process. The prediction horizon is the number of time steps into the future the model predicts (see Figure 2.2). The first formulation

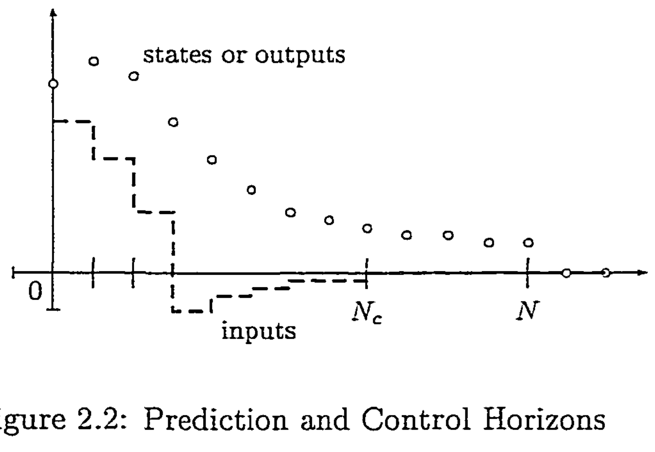

Figure 2.2: Prediction and Control Horizons

of MPC by Kalman [44] did not distinguish between control and prediction horizons and set

$$N_c = N = \infty.$$

Kalman's infinite horizon problem was tractable because there were no input or output constraints and the model was linear. Later, constraints were added to the formulation [20], [40]. To make the new problem tractable, the horizons were made finite, though allowed to be different. More recently, Rawlings and Muske showed how to formulate the problem with an infinite prediction horizon but finite control horizon for linear plants [64]. The showed that the infinite sum

$$\sum_{j=1}^{\infty} \|x_j\|_Q^2 + \|u_j\|_R^2$$

can be truncated by limiting the number of inputs the optimizer calculates. At the end of the control horizon N_c and for all time thereafter, the inputs and unstable system modes are set to the origin. The sum can then be split as

$$\sum_{j=1}^{N_c-1} \|x_j\|_Q^2 + \|u_j\|_R^2 + \sum_{j=N_c}^{\infty} \|x_j\|_Q^2$$

into a finite portion and infinite portion. For state-space models, the infinite portion can then be summed exactly. It is equivalent to a different weighting on the final state,

$$\sum_{j=0}^{N_c-1} \|x_j\|_Q^2 + \|u_j\|_R^2 + \|x_{N_c}\|_{Q_\infty}^2.$$

The problem of extending N_c to infinity for linear plants has been addressed by Rawlings and Scokaert [95] and Chmielewski and Manousiouthakis [14]. The key idea is that after some time in the future j^{min}, the constraints will no longer be active and the MPC algorithm will be equivalent to Kalman's unconstrained LQR. In this case, instead of setting the inputs to zero after some point in the future, the inputs are set to the solution of the LQR—a proportional controller with the Kalman gain. Rawlings and Scokaert show how to find a lower bound on j^{min}.

The infinite horizon Rawlings-Muske regulator possesses nice theoretical properties. Muske and Rawlings [64] showed how to guarantee closed loop stability of the controller for any choice of linear plants in the presence of input and output constraints, provided the model is perfect and the initial program is feasible.

The nominal algorithm (2.10) can be written as:

$$\begin{gathered}
\min_{u,x} \quad \|x\|_Q^2 + \|u\|_R^2 \\
\text{subject to} \\
h(x, u\,;\, x_k, \theta) = 0, \\
Dx + Fu \le f.
\end{gathered} \tag{2.11}$$

The vectors x and u are composed of x_j and u_j, respectively

$$\begin{aligned}
x &= \left[x_1^T\; x_2^T\; \cdots\; x_N\right]^T, \\
u &= \left[u_1^T\; u_2^T\; \cdots\; u_{N_c}\right]^T.
\end{aligned}$$

The function $h(x, u; x_k, \theta)$ represents the linear model with initial condition x_k and parameters θ. The inequality constraint represents the simple bounds on the inputs and states. For state-space models, the parameters θ are the matrices (A, B). For finite impulse response models, the states x can be directly replaced with notation for the outputs y. In this case, the parameter vector θ contains the impulse response coefficients.

We have chosen the form of (2.11) for simplicity. By varying the problem data and the functional form of h, we can represent a wide variety of nominal algorithms.

2.3.2 The Steady-State Algorithm

The steady-state target calculation for linear MPC commonly takes the form of a linear program (LP). Because most models in MPC applications are linear, with linear economics driving the controller, the result is a linear program.

Any linear model, whether it be state-space, step response, impulse response, or other, can be cast at steady-state in the following form:

$$\Delta y = G \Delta u.$$

Here $\Delta y \in \mathbb{R}^m$ represents the change between the current steady-state output and the last measured value, and $\Delta u \in \mathbb{R}^n$ represents the change between the current steady-state input and the last measured value:

$$\begin{aligned}\Delta u &= u_s - u_{k-1},\\ \Delta y &= y_s - y_{k-1}.\end{aligned}$$

Here y_s and u_s are vectors containing the future steady-state outputs and inputs, respectively, and y_{k-1} and u_{k-1} are the values of y and u at the previous time step. $G \in \mathbb{R}^{m\times n}$ is the *steady-state gain matrix* for the system. The LP finds optimal steady-state targets by minimizing an economic objective:

$$J_s = c^T u_s + d^T y_s, \tag{2.12a}$$

while maintaining the relationship between the steady-state inputs and outputs:

$$\Delta y = G \Delta u, \tag{2.12b}$$

respecting input and output constraints arising from process specifications

$$\underline{u} \leq u_s \leq \overline{u}, \tag{2.12c}$$

$$\underline{y} \leq y_s \leq \overline{y}, \tag{2.12d}$$

and ensuring the resulting solution does not force the dynamic MPC calculation to become infeasible:

$$N_c \Delta \underline{u} \leq \Delta u \leq N_c \Delta \overline{u}. \tag{2.12e}$$

In (2.12e) N_c refers to the control horizon of the dynamic MPC calculation, and $\Delta\underline{u}$ and $\Delta\overline{u}$ are the minimum and maximum bounds for the rate of change constraints in the dynamic MPC calculation. In (2.12c) and (2.12d) $\underline{u}$ and $\overline{u}$ are minimum and maximum bounds for u_s, respectively, with similar notation employed for the output bounds $\underline{y}$ and $\overline{y}$. Equation (2.12e) ensures the steady-state target is compatible with the rate of change or velocity constraints in the dynamic MPC calculation.

To avoid real-time infeasibilities in the LP, it is common to recast the *hard* output constraints $\underline{y} \leq y_s \leq \overline{y}$ in (2.12d) as *soft* constraints by adding slack variables $\overline{\epsilon} \geq 0 \in \mathbb{R}^m$, and $\underline{\epsilon} \geq 0 \in \mathbb{R}^m$, that allow for some amount of violation in the constraint:

$$-\underline{\epsilon} + \underline{y} \leq y_s \leq \overline{y} + \overline{\epsilon}. \tag{2.13}$$

The size of the violation is minimized by appending it to the objective function:

$$J_s = c^T u_s + d^T y_s + \overline{\epsilon} + \underline{\epsilon}. \tag{2.14}$$

Additionally, a bias is introduced into the model to incorporate feedback. It is assumed that the difference between the model prediction and the measured output at the current time is due to a constant step disturbance at the output. While other types of disturbance models can be incorporated into the MPC framework (see e.g. [64]), the constant output step disturbance assumption is standard in industrial applications. The model bias $b \in \mathbb{R}^m$ is based on a comparison between the current predicted output y and the current measured output $\hat{y} \in \mathbb{R}^m$:

$$b = \hat{y} - y. \tag{2.15}$$

The bias is added to the model:

$$\Delta y = G \Delta u + b. \tag{2.16}$$

The nominal steady-state target calculation is then:

$$\begin{aligned} \min_{u,y,\epsilon} \quad & c^T u_s + d^T y_s + \overline{\epsilon} + \underline{\epsilon} \\ \text{subject to} \quad & \\ & \Delta y_s = G \Delta u_s + b, \\ & \begin{array}{rcccl} \underline{u} & \leq & u_s & \leq & \overline{u}, \\ N_c \Delta \underline{u} & \leq & \Delta u & \leq & N_c \Delta \overline{u}, \\ -\underline{\epsilon} + \underline{y} & \leq & y_s & \leq & \overline{y} + \overline{\epsilon}, \\ 0 & \leq & \overline{\epsilon}, & & \\ 0 & \leq & \underline{\epsilon}. & & \end{array} \end{aligned} \tag{2.17}$$

It is common to calculate the moves Δu and Δy, instead of the inputs and outputs directly. The nominal steady-state target calculation can be expressed in velocity form as:

$$\begin{aligned} \min_{\Delta u, \Delta y, \epsilon} \quad & c^T \Delta u + d^T \Delta y + e^T \epsilon \\ \text{subject to} \quad & \\ & \Delta y = G \Delta u + b \\ & A_u \Delta u \leq b_u \\ & A_y \Delta y \leq b_y + \epsilon \\ & \epsilon \geq 0 \end{aligned} \tag{2.18}$$

where $e \in \mathbb{R}^{2m}$ is a vector which penalizes output deviations. For our discussion, we assume without loss of generality that

$$e = \begin{bmatrix} 1 \; 1 \; \cdots 1 \end{bmatrix}^T;$$

however, e may arbitrarily large in practice. The vector ϵ is comprised of stacking the upper and lower slack vectors:

$$\epsilon = \begin{bmatrix} \overline{\epsilon}^T \; \underline{\epsilon}^T \end{bmatrix}^T \tag{2.19a}$$

and $A_u \in \mathbb{R}^{4n x n}$ and $b_u \in \mathbb{R}^{4n}$ are given by

$$A_u = \begin{bmatrix} I \\ -I \\ I \\ -I \end{bmatrix} \qquad b_u = \begin{pmatrix} \overline{u} - u_{k-1} \\ -\underline{u} + u_{k-1} \\ N_c \Delta \overline{u} \\ -N_c \Delta \underline{u} \end{pmatrix}, \tag{2.19b}$$

while $A_y \in \mathbb{R}^{2m \times m}$ and $b_y \in \mathbb{R}^{2m}$ are given by

$$A_y = \begin{bmatrix} I \\ -I \end{bmatrix} \qquad b_y = \begin{pmatrix} \overline{y} - y_{k-1} \\ -\underline{y} + y_{k-1} \end{pmatrix}. \tag{2.19c}$$

Additionally, since Δy depends linearly upon Δu, the entire problem can be expressed in terms of Δu and ϵ, reducing the number of decision variables. The resulting LP can be cast in standard form and passed to an optimization algorithm such as the simplex method.

2.4 Summary of the Nominal Closed Loop Calculation

In this section we summarize the sequence of steps that make up a typical MPC calculation.

Estimator. The first step in the calculation is to estimate the model bias b_k:

$$b_k = y_{k-1} - \hat{y}_{k-1}.$$

Steady-state calculation. The steady-state target calculation takes b_k and computes the optimal steady-state targets y_k^s and u_k^s through the solution of:

$$\begin{aligned}(\Delta u^s, \Delta y^s, \epsilon^*)_k = \underset{\Delta u, \Delta y, \epsilon}{\operatorname{argmin}} \quad & c^T \Delta u + d^T \Delta y + e^T \epsilon \\ \text{subject to} \quad & \\ & \Delta y = G \Delta u + b \\ & A_u \Delta u \leq b_u \\ & A_y \Delta y \leq b_y + \epsilon \\ & \epsilon \geq 0,\end{aligned}$$

with

$$\begin{aligned} y_k^s &= \Delta y_k^s + y_{k-1}^s \\ u_k^s &= \Delta u_k^s + u_{k-1}. \end{aligned}$$

Dynamic calculation. Finally, the dynamic MPC controller takes y_k^s and u_k^s and produces the control input u_k that will be injected into the plant from the solution of

$$\begin{aligned} u_k = \underset{u,y}{\operatorname{argmin}} \quad & \|y - y_k^s\|_Q^2 + \|u - u_k^s\|_R^2 \\ \text{subject to} \quad & \\ & h(y, u; \theta) = 0 \\ & Dy + Fu \leq f. \end{aligned} \tag{2.20}$$

The first optimal input $(u_k)_1$ is then applied to the plant.

This set of calculations takes place at every controller execution. There may be auxiliary calculations that are done because of different model forms but the sequence of calculations remain the same. In the next chapter we formulate robust versions of the dynamic and steady-state algorithms.

Chapter 3

Robust MPC

In this chapter we investigate the formulation of the robust dynamic and steady-state problems. We will be concerned with casting those problems in a framework and terminology common to the field of optimization. As mentioned in the introduction, we use cost function bounds as the way to incorporate robustness in the dynamic optimization. For the steady-state target calculation, we consider the effect of uncertainty in the steady-state gain matrix and its effect on the LP.

3.1 Basics of Robust MPC

As opposed to nominal model predictive control where the model is assumed perfect, robust model predictive control allows for model uncertainty. To clarify our discussion we will need the following definitions which describe the closed loop behavior of a system.

Definition 3.1 (Convergence). A point is x_k is said to *converge* to the equilibrium point x^e if $||x_k - x^e|| \to 0$ as $k \to \infty$. □

Definition 3.2 (Stability). An equilibrium point x^e said to be *stable* if, for every $\rho > 0$, there exists an $r(\rho, x^e) > 0$ such that if $||x_0 - x^e|| < r$, then $||x_k - x^e|| < \rho$ for all $k \geq 0$. □

Definition 3.3 (Asymptotic Stability). An equilibrium point is said to be *asymptotically stable* if it is stable and if, in addition, there exists some $r > 0$ such that

$\|x_0 - x^e\| < r$ implies $\|x_k - x^e\| \to 0$ as $k \to \infty$. □

The equilibrium point in the above definitions refers to the targets calculated by the steady-state target optimization. Since it is almost universally assumed the targets do not change from one time-step to the next, the above definitions imply the equilibrium point does not change.

Definition 3.4 (Robust Stability). An MPC algorithm is said to be *robustly stabilizing* if it guarantees asymptotic stability in the presence of model uncertainty. □

Robust stability is the goal of most robust MPC algorithms. Most control practitioners, however, require robust performance. Robust performance is the ability to achieve acceptable control performance in the presence of model uncertainty. Both concepts require a mathematical description of the possible plants known as an uncertainty description.

3.2 Uncertainty Descriptions

Uncertainty in robust MPC is parameterized through an uncertainty description for the model parameters. No model can hope to perfectly describe a process. Modeling error may due to poor signal design or execution of test signals, invalid modeling assumptions or changes in plant operation. Additionally, a single model is often used to control a system over a wide operation region where the model may not be as accurate. Figure 3.1 shows both the nominal operating point $\bar{p}$ of a system and its common operation region. While the model may be accurate around $\bar{p}$, it may degrade as the system moves away to other points in the operating region.

3.2.1 Ellipsoidal Uncertainty

The uncertainty description defines the set of the most likely model parameters as well as the uncertainty in those parameters. It can be given parametrically or statistically.

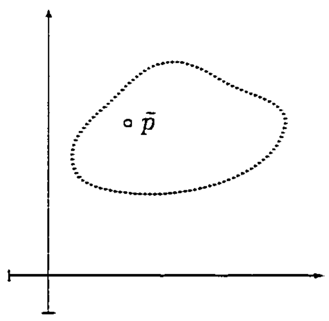

Figure 3.1: Conceptualization of the Operating Region for a System in the Model Parameter Space.

In model identification, if the process noise and model parameters are assumed to be normally distributed or Gaussian variables, the natural uncertainty description is an ellipsoidal bound on the parameters [5]. Illustrated in Figure 3.2, the parameter vector

$$\theta = [\theta_1 \cdots \theta_{n_\theta}]^T \in \mathbb{R}^{n_\theta}$$

is assumed to lie in the set:

$$\theta \in \Theta \overset{def}{=} \left\{\theta \ : \ (\theta - \theta_c)^T W^{-1} (\theta - \theta_c) \leq 1\right\}, \tag{3.1}$$

describing the joint confidence region. The center of the ellipse is located at θ_c and the symmetric positive-definite matrix W gives the size and orientation of the ellipsoid. In particular, the square roots of the reciprocals of the eigenvalues of W are the lengths of the semi-axes of the ellipsoid, and the eigenvectors of W define the directions of the semi-axes.

The true plant, defined by the set of parameters $\overline{\theta}$, is by assumption contained within the ellipse. The confidence region defined by the ellipsoid in (3.1) can equivalently be written as:

$$\theta \in \Theta \overset{def}{=} \left\{\theta_c + \beta V^{1/2} s \ : \ \|s\| \leq 1\right\}, \tag{3.2}$$

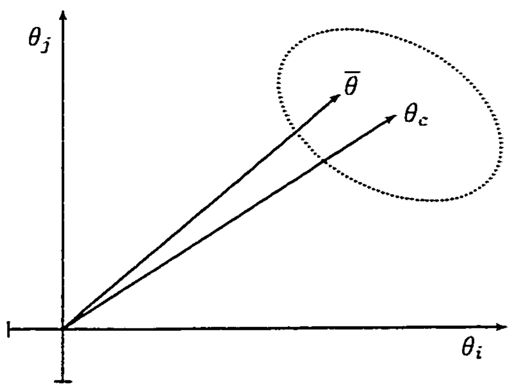

Figure 3.2: Ellipsoid Parameter Uncertainty

where we have replaced the matrix W with its statistical interpretation:

$$W = V^{-1}/\beta^2.$$

The matrix V is the covariance or estimate of the covariance of the estimated parameters. The term β is related to the radius of the ellipse and is given by

$$\beta = \Phi^{-1}(1-\alpha)$$

where $\Phi(x)$ is the univariate normal distribution function with zero mean and unit variance:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}s^2} ds$$

and $1-\alpha$ is the confidence or probability level. We can always guarantee a value of β to exist for any probability level $1-\alpha$, even for the case of when V is only an estimate of the true covariance [5]. For (3.2) to remain convex, $\alpha \leq 0.5$, or equivalently, $\beta \geq 0$.

The uncertainty description given by (3.2) is more general than (3.1) as it allows V to be rank deficient. When the covariance matrix V does not have full rank, the ellipsoid becomes degenerate, collapsing in specific directions corresponding to parameters which are known exactly. The directions in which the ellipse collapses correspond to the eigenvectors (or semi-axes) with associated eigenvectors that are zero (i.e. those directions for which a nonzero z produces $V^{1/2}z$ which is zero).

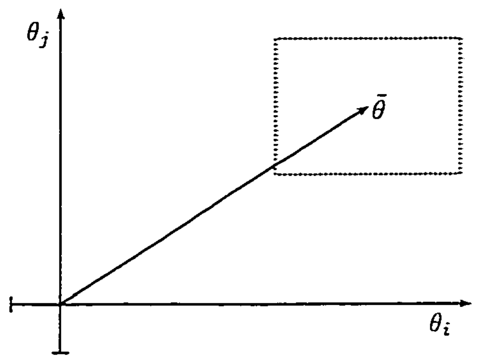

Figure 3.3: Box or Polytope Uncertainty

A similar discussion of ellipsoid uncertainty is also given by Ben-Tal and Nemirovski [7], who give a more formal treatment of the ellipsoid bounds in the general context of convex programming, and by Boyd et al. [10] who motivate the use of ellipsoid uncertainty descriptions for use in optimal control.

3.2.2 Box Uncertainty

If the ellipsoidal constraint can be interpreted in terms of a joint confidence region, then a box constraint can be interpreted in terms of joint confidence intervals (or as approximations to the ellipse). Figure 3.3 illustrates simple box constraints. One interpretation of box constraints is as minimum and maximum bounds of the ellipse. More complicated polytopes may also be used in an attempt to approximate the ellipse. A general polytope, including box constraints, can be expressed as bounds on some linear combination of the parameters θ. The uncertainty description $\mathcal{B}$ is then given by:

$$\theta \in \mathcal{B} \overset{def}{=} \left\{\theta \;:\; A_\theta \, \theta \leq b_\theta\right\}. \tag{3.3}$$

Both ellipsoidal and polytope uncertainty descriptions are common. Ellipsoidal uncertainty tends to result in a smaller final optimization problem at the cost of introducing a nonlinear inequality constraint into the formulation. Polytope uncertainty generally results in larger optimization problems but the added constraints are linear. For the robust dynamic algorithm, we will be concerned with only ellipsoidal uncertainty descriptions. For the robust LP, we will consider both types.

3.3 The Robust Dynamic Algorithm

Recall the four primary ways of imparting robustness to the dynamic MPC algorithm:

- Lee [52] describes the most well-known approach, the **Min-Max** algorithm. This algorithm minimizes the maximum cost function in the uncertainty region. This method tends to be overly conservative since it optimizes the worst-case performance at each time step.

- Another well-known way to incorporate robustness for stabilizable plants is to limit input movement by **detuning**. It is always possible to stabilize a stabilizable plant by suppressing the input movement, and in the limit of infinite move suppression, the system reverts to its open loop behavior. Vuthandam, Genceli, and Nikolaou give an example of this for a modified QDMC controller [109]. Large move suppression, while it stabilizes stabilizable plants, degrades performance and does not address the issue of unstable plants.

- The use of constraints is a much more promising way of introducing robustness to a nominally stabilizing MPC algorithm. Zheng [116] and Scokaert and Mayne [94] describe ways of including **terminal state constraints** that impart robustness. Zheng requires the largest possible terminal state to be smaller than the initial state at the end of a user specified horizon. This method does not directly generalize to unstable, nonlinear systems. Scokaert and Mayne were able to robustly stabilize nonlinear processes by using a **state contraction constraint** that ensured the process would eventually enter a region in input space that was equivalent to the unconstrained problem.

- Finally, Badgwell and coworkers show that it is possible to ensure robust stability by including a nonlinear constraint that forces the cost function for all plants in the uncertainty description to form a non-increasing sequence. This formulation has been given for linear state-space models with finite uncertainty [4], [46], stable and integrating plants for finite impulse response models [79] with continuous uncertainty as well, and nonlinear plants [3] with finite uncertainty. This type of constraint is referred to as a **cost function constraint.**

The approach by Badgwell is the most general and powerful formulation presented as yet. For a finite uncertainty description, it results in a quadratic constraint for each plant in the uncertainty description. For a continuous uncertainty description it results in a semi-infinite constraint over all the plant parameters in the uncertainty description.

Cost function bounds take the following form:

$$J(u; x_k, \theta) \leq J(\hat{u}_k; x_k, \theta), \qquad \forall\, \theta \in \mathcal{E}, \tag{3.4}$$

where J is the cost function for the problem; x_k is the current state vector of the system; u is the input vector to be computed; θ is the vector of model parameters; and $\hat{u}_k$ is a shifted version of the previous optimal input u^*_{k-1}. If the previous optimal input is given by

$$u^*_{k-1} = \left[u_1^{*T} \; u_2^{*T} \; \cdots \; u_{N_c}^{*\;T}\right]_{k-1}^T$$

then $\hat{u}_k$ is given by

$$\hat{u}_k = \left[u_2^{*T} \; \cdots \; u_{N_c}^{*\;T} \; 0^T\right]_{k-1}^T$$

and is known as the the *restriction of the input.* The above transformation shifts the elements of the previous optimal input to the left and appends a zero to the end of the vector. We give here a sketch of the argument for the robust stability of the closed loop system; the full proof is stated in [3].

Sketch of Proof. *The proofs vary for different model forms, but for the most part, have the same general structure. First it is shown that the input and the output of the true plant converge to the origin. Then it is shown that the origin is a stable* *equilibrium point for the closed loop system. The combination of convergence and stability yields asymptotic stability.*

By assumption, the true plant $\overline{\theta}$ is in the uncertainty description $\mathcal{E}$

$$\overline{\theta} \in \mathcal{E}.$$

From the cost function constraint and the definition of optimality, it can be shown that the restriction of the input $\hat{u}_{k+1}$ is a feasible solution (although not optimal) of the the problem at time $k+1$. It can then be shown that:

$$J(u^*_{k+1}; x_{k+1}, \overline{\theta}) \leq J(\hat{u}_{k+1}; x_{k+1}, \overline{\theta}).$$

Subtracting objectives at subsequent time-steps yields the equality:

$$J(\hat{u}_{k+1}; x_{k+1}, \overline{\theta}) - J(u^*_k; x_k, \overline{\theta}) = -\alpha(u_k, y_k).$$

where $\alpha(u_k, y_k)$ is a positive functional which smoothly approaches zero as u_k and y_k approach zero. For a quadratic objective, α is a positive-definite quadratic functional. Combining the two equations, we get

$$J(u^*_{k+1}; x_{k+1}, \overline{\theta}) - J(u^*_k; x_k, \overline{\theta}) \leq -\alpha(u_k, y_k).$$

This shows the sequence of optimal costs $\{J(u^*_k; x_k, \overline{\theta})\}$ for the actual plant is non-increasing. We also know the plant cost is bounded below by zero. Thus as the left side of the above inequality approaches zero, the input and output converge to the origin (due to the properties of J). This establishes convergence.

Stability of the origin is established by showing the state at time k is bounded for all $k > 0$

$$||x_k|| \leq \rho \qquad \forall k \ > 0$$

which can be shown to hold for objectives of the type commonly used in control. The combination of convergence and stability yields asymptotic stability. $\square$

In short, cost function bounds guarantee the sequence of objective costs $\{J(u^*_k; x_k, \overline{\theta})\}$ be non-increasing for the true plant. Although the objectives for other plants may increase, the true plant cost is guaranteed to not increase. This leads to robust stability.

Consider cost function bounds for a quadratic objective. The robust MPC calculation requires the solution of a nonlinear program of the form:

$$\begin{aligned} u_k = \underset{u_j}{\operatorname{argmin}} \quad & J_k(u; x_k, \theta) = \|x\|_Q^2 + \|u\|_R^2 \\ \text{subject to} \quad & \\ & h\,(x, u; x_k, \theta) = 0 \\ & Dx + Fu \leq f \\ & J(u; x_k, \theta) \leq J(\hat{u}_k; x_k, \theta), \qquad \forall \theta \in \mathcal{E}. \end{aligned} \tag{3.5}$$

The objective depends implicitly upon the model parameters θ through the model equations h. For the models presented in [4] and [79] the parameters appear linearly allowing us to rewrite the model h explicitly as:

$$x = P(u)\,\theta. \tag{3.6}$$

The matrix P is nontrivial function of u. The cost function constraint now becomes:

$$\theta^T S\,\theta \leq 0 \qquad \forall \theta \in \mathcal{E}, \tag{3.7}$$

where

$$S(u) = P^T(u)QP(u) - P^T(\hat{u})QP(\hat{u}). \tag{3.8}$$

The optimization problem now takes on the following form:

$$\begin{aligned} u_k = \underset{u}{\operatorname{argmin}} \quad & \|x\|_Q^2 + \|u\|_R^2 \\ \text{subject to} \quad & \\ & h(x, u; x_k, \theta) = 0 \\ & Dx + Fu \leq f \\ & \theta^T S(u)\,\theta \leq 0 \qquad \forall \theta \in \mathcal{E}. \end{aligned} \tag{3.9}$$

The method of solving (3.9) used in [4], [79] was to recognize that

$$\theta^T S(u)\,\theta \leq 0 \qquad \forall\,\theta \in \mathcal{E}$$

is equivalent to

$$\begin{aligned} &\max_{\theta} \quad \theta^T S(u)\,\theta \quad \leq 0 \\ &\text{subject to} \\ &\qquad \theta \in \mathcal{E} \end{aligned}$$

and solve the following nonlinear optimization, treating the semi-infinite inequality constraint as a nonlinear equality constraint:

$$\begin{aligned} u_k = &\operatorname*{argmin}_{u} \quad \|x\|_Q^2 + \|u\|_R^2 \\ &\text{subject to} \\ &\qquad h(x, u; x_k, \theta^*) = 0 \\ &\qquad Dx + Fu \leq f \\ &\qquad \theta^{*T} S(u)\,\theta^* \leq 0; \\ &\qquad \theta^* = \operatorname*{argmax}_{\theta} \quad \theta^T S(u)\,\theta \\ &\qquad\qquad\qquad \theta \in \mathcal{E}. \end{aligned}$$

The solution of the inner maximization was treated as a function evaluation. This turns out to be very expensive and very slow. A better approach is to treat (3.9) as a semi-infinite program, as we show in chapter 4. We next consider the effect of uncertainty in the steady-state LP.

3.4 The Robust Steady-State Algorithm

In the steady-state target calculation, the uncertain parameters θ are the elements of the steady-state gain matrix G. Let g_i be the column vector describing the i^{th} row of G:

$$G = [g_1 \;\; g_2 \;\; \cdots \;\; g_m]^T. \tag{3.10}$$

Assume for the moment that the outputs are independent random variables; no output is correlated to another. In this case we can construct a block diagonal covariance matrix for the process gains made up of the individual covariance matrices for each row of the gain matrix. In particular, if V_i is the covariance matrix corresponding to the ith row, then the covariance for the entire matrix is given by

$$V = \begin{bmatrix} V_1 & & & \\ & V_2 & & \\ & & \ddots & \\ & & & V_{n_y} \end{bmatrix}. \tag{3.11}$$

If the outputs are actually cross-correlated, then the off diagonal elements of V will be nonzero. The gains have been stacked into a single vector g:

$$g = [g_1^T \; g_2^T \; \cdots \; g_m^T]^T$$

In any case, we can assume without loss of generality that g is nominally $\bar{g}$ and has covariance V, yielding the following ellipsoidal uncertainty description:

$$g \in \mathcal{E} \overset{def}{=} \left\{ \bar{g} + \beta V^{1/2} s \; : \; \|s\| \leq 1 \right\}, \tag{3.12}$$

where $\mathcal{E}$ defines the ellipsoid for the entire matrix. This is the most natural way to pose the problem as it results in *constraint-wise* uncertainty. In the nominal LP, the gain matrix G is premultiplied by the (possibly dense) matrix A_y in the output constraint. This results in a linear combination of outputs. The ith component of the output constraint is given by

$$\sum_j a_{ij} g_j^T \Delta u \leq b_{y_i}.$$

In the general case, we will require ellipsoids of the type above to capture the uncertainty in the constraint. If, however, most of the elements of A_y are zero, as is the case for simple bounds on the outputs, it is possible to simplify the elliptic uncertainty description.

For simple bounds, all the a_{ij} are zero except one. There is a constant scalar factor γ premultiplying the ith constraint.

$$\gamma\, g_i^T \Delta u \leq b_{y_i}.$$

Each component of the constraint depends only upon a single row of the gain matrix. Thus we can define an ellipsoid for each row. Let the ith row of the gain matrix nominally be $\bar{g}_i^T$ with the covariance is V_i. The ellipsoid is given by:

$$g_i \in \mathcal{E}_i \overset{def}{=} \left\{ \bar{g}_i + \beta\, V_i^{1/2} s \ : \ \|s\| \leq 1 \right\}. \tag{3.13}$$

Unlike the dynamic algorithm there is no prior analysis of the steady-state algorithm. As a result, we will introduce the idea of the robust LP through an example. The solution of the LP always lies at the intersection of constraints. Consider the instance when the solution lies at the intersection of two output constraints. Figure 3.4 shows two output constraints represented by the solid lines. Due to uncertainty in the gain matrix, the bounds describing the constraints are blurred. While the nominal LP finds a solution at the vertex of the two constraints, it may in fact violate both.

Consider the following example. The steady-state model is given by

$$\Delta y = G \Delta u,$$

with $\Delta u \in \mathbb{R}^2$ and $\Delta y \in \mathbb{R}$. Let the nominal value of g be

$$\bar{g} = \begin{bmatrix} 1.00 \\ 0.75 \end{bmatrix}.$$

We will assume an ellipsoidal uncertainty description with the following covariance:

$$V = \begin{bmatrix} 0.03 & -0.05 \\ -0.05 & 0.40 \end{bmatrix}. \tag{3.14}$$

For the moment, we will ignore the bias and slack variables. We will require that the matrix elements be known with a probability level $1 - \alpha$ of 99%, which corresponds

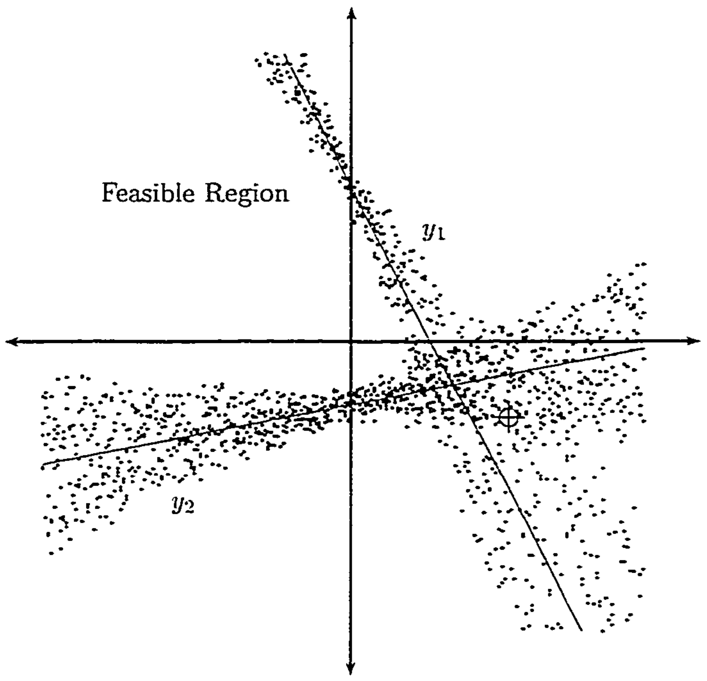

Figure 3.4: Fuzzy Output Constraints

to $\beta = 2.33$. The ellipsoid uncertainty can be approximated conservatively as a box constraint with the following data:

$$A_g = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \qquad b_g = \begin{pmatrix} 1.26 \\ 1.71 \\ -0.74 \\ 0.21 \end{pmatrix}.$$

The ellipsoid uncertainty description and its box approximation are illustrated in Figure 3.5. A problem arises when we consider output constraints in nominal problem (2.18). The constraints can correspond to critical variables—the temperature limit on a reactor vessel or the maximum allowable feed rate into a unit. Uncertainty or *fuzziness* in the constraint means that even though the nominal value may be binding at the upper or lower limit, the actual value may be outside the constraint region.

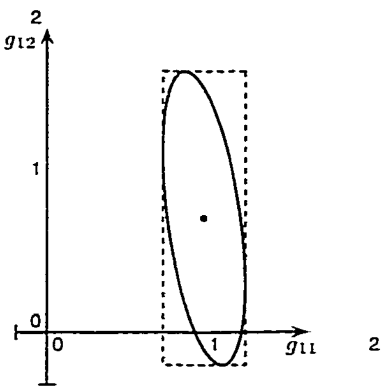

Figure 3.5: Ellipsoid and the Corresponding Box Uncertainty in the Example

To see this, we add the following constraints to the example:

$$-4 \le \Delta u_1 \le 4$$
$$-3 \le \Delta u_2 \le 1$$
$$-2 \le \Delta y_1 \le 4.$$

The feasible region defined by the constraints corresponds to the interior of the polygon in Figure 3.6. This region assumes nominal values for the gains.

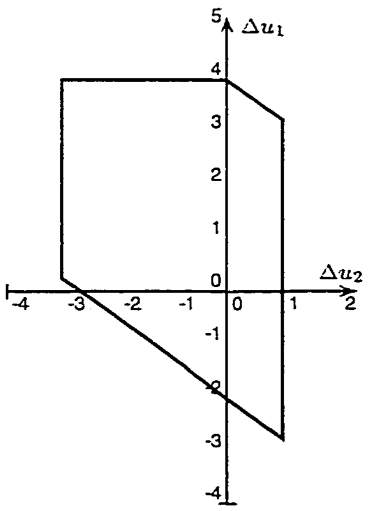

Figure 3.6: Feasible Region for the Nominal LP in the Example

The lower diagonal line describes the lower bound on the output $\Delta y_1 \geq -2$. The line is given by:

$$g_{11}\Delta u_1 + g_{12}\Delta u_2 + 2 = 0.$$

Uncertainty in g causes uncertainty in the constraint. Figure 3.7 shows the nominal constraint and its confidence bounds. Consider the case in which the constraint is binding. Uncertainty could force the true process to lie anywhere within the confidence limits, meaning it could easily lie outside of the feasible set. The equation describing the bounds is given by:

$$\tilde{g}^T \Delta u \pm \beta \|V^{1/2}\Delta u\| + 2 = 0,$$

where $\|\cdot\|$ is the standard Euclidean norm. Every output constraint will contain some

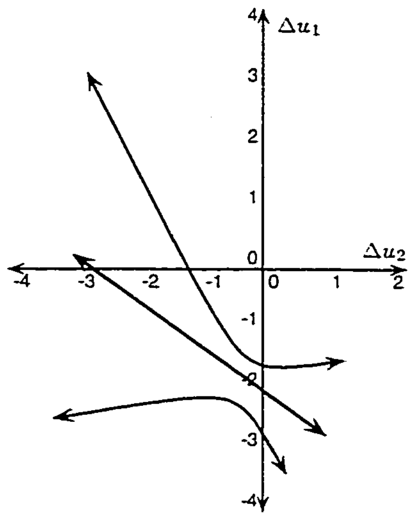

Figure 3.7: Confidence Limits on the Lower Bound $\Delta y_1 \geq -2$

degree of uncertainty. If we include the uncertainty from the upper constraint $\Delta y_1 \leq 4$ and plot the boundary of the new feasible set we obtain Figure (3.8). Uncertainty in the output constraints has transformed the original feasible region (dotted line) into the smaller feasible region shown by the solid line. Note that the feasible region remains convex.

As the controller is required to consider more and more possible plants, corresponding to an increase in the size of the uncertainty description, and consequently robustness, the size of the feasible region will decrease, corresponding to a more conservative controller. This is the familiar robustness/performance tradeoff.

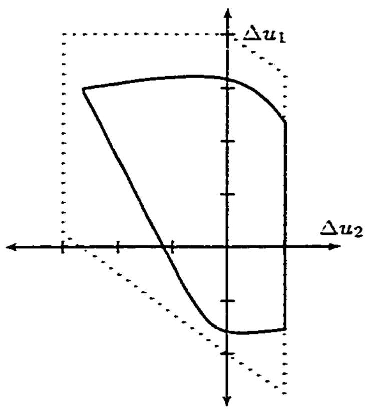

Figure 3.8: Feasible Regions for the Robust LP; Nominal Problem—Dotted Line; Elliptic Uncertainty—Solid Line

The feasible region can be defined as the set of inputs which produce feasible outputs for any possible value of the gains:

$$\Delta u \in \left\{ \Delta u \; : \; \begin{array}{l} A_y \Delta y = A_y G \Delta u \leq b_y, \; \forall G \in \mathcal{U} \\ A_u \Delta u \leq b_u \end{array} \right\},$$

where $\mathcal{U}$ is one of the uncertainty descriptions given earlier.

In addition to gain uncertainty affecting the constraints, it also has an effect on the objective function:

$$\begin{aligned} J_s &= c^T \Delta u + d^T \Delta y \\ &= c^T \Delta u + d^T G \Delta u. \end{aligned}$$

Let the gradient of the objective with respect to the inputs be f:

$$\nabla J_s = f = c + G^T d.$$

For our example, assume c and d are given by:

$$c = \begin{bmatrix}7.00 & 5.00\end{bmatrix}^T \qquad \text{and} \qquad d = \begin{bmatrix}-6.00\end{bmatrix}.$$

Uncertainty causes the value of f to change. f is nominally $\tilde{f}$:

$$\tilde{f} = \begin{bmatrix}1.00 & 0.50\end{bmatrix}^T,$$

and can take on values anywhere between $\underline{f}$ and $\overline{f}$:

$$\underline{f} = \begin{bmatrix}-.56 & 2.56\end{bmatrix}^T \qquad \overline{f} = \begin{bmatrix}-5.26 & 6.26\end{bmatrix}^T.$$

Figure 3.9 shows the the effect of this uncertainty in the objective function. As the objective changes, the solution of the LP changes. One possibility is to include the gains in the objective, minimizing them in some sense. From a control viewpoint, however, this may not make sense. A smarter control policy would be to minimize the best guess or nominal value of the gain. The goal is to drive the process toward the

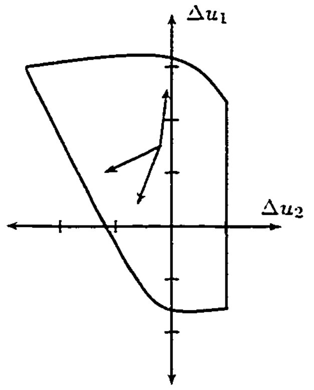

Figure 3.9: Uncertainty in the Objective Function

economic optimum using the best guess for the gain of the process but restrain the inputs to only those that ensure the process will remain feasible for any reasonable gains (those captured by the uncertainty description). If we use the nominal value of the gain $\tilde{G}$, the objective function for the robust LP becomes:

$$J_s = c^T \Delta u + d^T \tilde{G} \Delta u + e^T \epsilon.$$

Although this example is simple, it illustrates the nature of the problem. Model uncertainty forces the problem structure to change. The linear constraints so commonly used in control theory become nonlinear. In our example we were able to graphically determine the optimal solution. In practice, we need to solve the following optimization problem:

$$\min_{\Delta u, \epsilon} \quad c^T \Delta u + d^T \tilde{G} \Delta u + e^T \epsilon$$

subject to

$$\begin{aligned} & A_y G \Delta u \leq b_y + \epsilon - A_y b, \quad \forall\, G \in \mathcal{U} \\ & A_u \Delta u \leq b_u \\ & \epsilon \geq 0. \end{aligned} \tag{3.15}$$

Problem (3.15) is a convex, semi-infinite program that can be solved efficiently using the methods described in Chapter 4.

3.5 Summary of Robust MPC

Both the dynamic and steady-state robust algorithms take the form of semi-infinite programs. The semi-infinite constraint has the following specific form:

$$\begin{aligned} & h(x\,;\,\theta) = 0 \\ & g(x) \leq 0 \qquad \forall \theta \in \mathcal{E} \end{aligned}$$

where the number of variables x in the constraint is finite, but the constraint must hold (implicitly) for all the parameters θ in an infinite set $\mathcal{E}$. In the dynamic calculation, the constraint arises from theoretical arguments that guarantee stability of the closed loop system by forcing the objective in the online optimization to form a non-increasing sequence. In the steady-state calculation, the constraint arises from requiring the output constraints hold for an infinite set of plants. In the next chapter we consider the detailed structure of the resulting semi-infinite programs. We show that efficient solution algorithms can be constructed by exploiting convexity through the use of interior-point algorithms.

Chapter 4

Semi-Infinite Programming

Consider the following semi-infinite program (SIP):

$$\begin{aligned} \min_{x} \quad & f(x) \\ & h(x\,;\,\theta) = 0 \\ & g(x) \leq 0 \quad \text{for all} \quad \theta \in \mathcal{U}, \end{aligned} \tag{4.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^{N_e}$, and $g : \mathbb{R}^n \to \mathbb{R}^{N_i}$. θ is a set of parameters and, for most engineering problems $\mathcal{U}$ is closed and convex. The inequality constraint $g(x)$ depends implicitly upon the parameters in the model $h(x\,;\,\theta)$. Problem (4.1) occurs in many different fields ranging from robotics to structural design. Many robust optimization problems can be cast as semi-infinite optimization problems. While semi-infinite programming has been commonly used in the field of optimal control (see e.g. [85]), it is not a widely accepted tool for model predictive control.

4.1 Optimality Conditions

In this section we apply the first-order necessary conditions and the second-order necessary conditions to a general nonlinear program. Kuhn and Tucker presented these conditions in 1951 [49], and until the late 1980s they were referred to as the KT conditions. Earlier, however, Karush derived the same conditions in his 1939 master's thesis [45]. It is common practice now to credit Karush for his work and refer to these as the KKT (Karush-Kuhn-Tucker) conditions for nonlinear programming. Tapia and

Trosset give an excellent historical review of these events in the introduction of [101]. We are interested in the KKT conditions for the general nonlinear program because, as we show in the next section, we can use them to reduce a nonlinear programming problem with an infinite number of constraints to an equivalent (or approximate) problem with a finite number of constraints.

The KKT conditions can be seen as an extension of the Lagrange multiplier theory for problems with equality constraints to problems with both equality and inequality constraints. By the general nonlinear program with both equality and inequality constraints we mean:

$$\begin{aligned} &\min_x \quad f(x) \\ &\quad \text{subject to} \\ &\qquad h_i(x) = 0 \quad i = 1, \dots, n \\ &\qquad g_i(x) \geq 0 \quad i = 1, \dots, m. \end{aligned} \tag{4.2}$$

Definition 4.1. The set of indices given by:

$$A(x) = \Big\{ j \ : \ g_j(x) = 0 \Big\} \tag{4.3}$$

are known as the active indices at x and the corresponding inequality constraints $g_j(x)$ are known as the *active* or *binding* constraints for problem (4.2). □

Definition 4.2. A point x_* is regular for problem (4.2) if the set of vectors

$$\Big\{ \nabla h_1(x_*), \dots, \nabla h_{N_e}(x_*), \nabla g_i(x_*), i \in A(x) \Big\} \tag{4.4}$$

is linearly independent. □

The following propositions summarize the well-known first and second order necessary conditions for optimality; see [31], [37] for a detailed discussion.

Proposition 4.1.1. *(Karush-Kuhn-Tucker) If the regular point x_* is a local mini-*

mizer of problem (4.2), *then there exists* $\lambda_* \in \mathbb{R}^n$ *and* $\mu_* \in \mathbb{R}^m$ *such that:*

$$\begin{aligned}
&h_i(x_*) = 0 \quad i = 1, \dots, n \\
&g_j(x_*) \geq 0 \quad j = 1, \dots, m \\
&\nabla f(x_*) + \sum_{i=1}^{n} (\lambda_*)_i \nabla h_i(x_*) + \sum_{j=1}^{m} (\mu_*)_j \nabla g_j(x_*) = 0 \qquad (4.5) \\
&g_j(x_*)(\mu_*)_j = 0 \quad j = 1, \dots, m \\
&\mu* \geq 0.
\end{aligned}$$

These conditions are referred to as the Karush-Kuhn-Tucker or KKT conditions for optimality. □

Proposition 4.1.2. *(Karush-Kuhn-Tucker) If the regular point* x_* *is a minimizer for problem* (4.2), *then*

$$\nabla^2 f(x_*) + \sum_{i=1}^{N_e} (\lambda_*)_i \nabla^2 h_i(x_*) + \sum_{j=1}^{N_i} (\mu_*)_j g_j(x_*) \qquad (4.6)$$

is positive semi-definite on the null space of vectors in (4.4). □

The KKT conditions for the nonlinear program will be an integral part of our discussion of primal-dual interior-point methods in the next section. We will use these ideas when we discuss efficient solutions methods for the problems in robust MPC.

4.2 Basics of Primal-Dual Interior-Point Methods

In 1987, Kojima, Mizuno, and Yoshise [48] proposed the now celebrated primal-dual interior-point algorithm for linear programming. Since then, there has been considerable effort to extend interior-point methods to other optimization paradigms. The monotone linear complementarity problem (mLCP) is one such extension that directly generalizes the linear program to include quadratic objective functions. Wright showed the optimization resulting from nominal MPC falls into this category and that primal-dual interior-point methods can be used in their solution [112]. The text by Cottle, Pang, and Stone [15] provides an excellent discussion of many other linear complementarity problems.

This section introduces the ideas behind many of the interior-pont algorithms. We discuss the role of the central path and the perturbed KKT conditions. Wright's text [113] and the papers by Megiddo [59], and Kojima, Mizuno, Yoshise [48] are good references on the role of the central path in interior-point methods. Tapia [100] and Elbakry et al. [25] [24] provide an excellent discussion of the perturbed KKT conditions; we will rely heavily on their papers in the following discussion.

4.2.1 Perturbed KKT Conditions

Linear programming saw the first use of primal-dual interior-point methods. It is natural to introduce the idea in this context. Consider the standard form LP:

$$\begin{aligned} &\min_x \quad c^T x \\ &\text{subject to} \\ &\qquad Ax = b \\ &\qquad x \geq 0 \end{aligned} \tag{4.7a}$$

and its dual:

$$\begin{aligned} &\max_{y,z} \quad b^T y \\ &\text{subject to} \\ &\qquad A^T y + z = c \\ &\qquad z \geq 0, \end{aligned} \tag{4.7b}$$

where $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $m \leq n$. The other vectors and data are of appropriate dimension. The difference between the value of the objective function of the primal problem $c^T x$ and the value of the objective function of the dual problem $b^T y$ is known as the *duality gap* μ,

$$\mu = c^T x - b^T y. \tag{4.8}$$

For an LP, it can be shown that at optimality, the duality gap is zero. For any arbitrary x and any arbitrary y, the duality gap is some finite number. The goal of primal-dual interior-point methods is to drive the duality gap to zero.

As is common, we use X to denote the diagonal matrix with the elements of x on the diagonal and employ an analogous notation for other quantities. Also, the vector e is defined as a vector of all ones whose dimension will vary in context.

The KKT conditions for problem (4.7) are

$$F(x, y, z) = \begin{bmatrix} Ax - b \\ A^T y + z - c \\ XZe \end{bmatrix} = 0, \qquad (x, z) \geq 0 \tag{4.9}$$

corresponding to primal feasibility, dual feasibility, and complementarity. $F(x, y, z) = 0$ is a square nonlinear system of equations. The nonlinearity comes from the complementarity condition. In any Newton method solution of (4.9), we deal with linearized complementarity:

$$x_i \Delta z_i + z_i \Delta x_i = -x_i z_i$$

where x_i and z_i are elements of x and z respectively, and Δx_i and Δz_i are the corresponding Newton steps. Complementarity plays a crucial role in primal-dual interior-point algorithms. If $(x_i)_l$ is the i^{th} component of x at iteration l in some Newton method, and at iteration k, the component reaches zero $(x_i)_k = 0$, then for all time thereafter $l > k$, the only resulting choice is a zero step $(\Delta x_i)_l = 0$. This obviously prevents any chance of convergence to an optimal solution. This phenomenon is known–not so eloquently–as *sticking to the boundary.* The papers written by Tapia and colleagues and described at the beginning of this section give an excellent discussion of this problem.

The most natural alternative is to perturb the complementarity condition by some amount:

$$F(x, y, z) = \begin{bmatrix} Ax - b \\ A^T y + z - c \\ XZe \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \mu e \end{bmatrix}. \tag{4.10}$$

The perturbation keeps the iterates away from the boundary of the feasible set. If we were to solve (4.10) with some finite nonzero perturbation μ, we would converge to a triplet $(\hat{x}, \hat{y}, \hat{z})$. If we then changed μ and solved the equations again, we would converge to a different triplet. As μ continuously changes, we trace out a piecewise continuous curve known as the *central path.*

4.2.2 The Central Path

Adherence to a central path is what makes primal-dual interior-point methods polynomial algorithms. That is, the effort required to solve them can be bounded by a polynomial in the data.

The central path keeps iterates away from the boundary. If we were to graph the set of triplets described above, the result would be Figure 4.1. The axes are in what is called image-space. All the components of the complementarity condition are

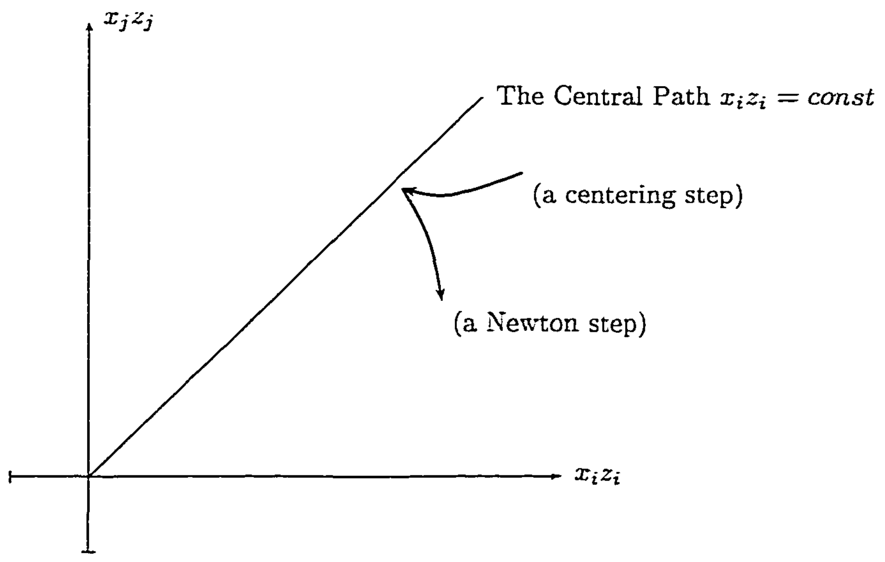

Figure 4.1: The Central Path

perturbed by the same amount. This is represented by the diagonal line in Figure 4.1, which is the central path. For a given μ, a solution of (4.10) converges to a point somewhere on the diagonal line. If however, instead of taking many Newton steps, which is (under certain conditions) equivalent to solving (4.10), we just took one Newton step at constant $\mu \neq 0$, we would be moving toward the central path and away from the boundary. We would not however, likely lie on the central path. If, on the other hand, we set $\mu = 0$ and took a Newton step, we would be taking a Newton step on our original problem. A Newton step at constant $\mu \neq 0$ is referred to as a *centering* step since it moves the iterates toward the central path.

One choice is to alternate centering steps and Newton steps as the iterations progress. This is exactly the famous predictor-corrector algorithm of Mizuno, Todd, and Ye [61]. At one iteration of the algorithm, the perturbation is set to zero. At the next iteration the perturbation is set to μ, the duality gap. In general, it is possible to take some combination of both. This is done through the introduction of a *centering parameter* σ:

$$\mu = \sigma(c^T x - b^T y).$$

When σ is zero, μ is zero and the algorithm takes a step on the original KKT conditions. When σ is one, μ is the duality gap and the algorithms take a step on the Perturbed KKT (PKKT) conditions. Intermediate values of σ can also be used.

During the course of a primal-dual interior-point algorithm the perturbation μ (the duality gap) is gradually reduced to zero. Thus at the solution, there is no perturbation (i.e. no duality gap) and the original unperturbed problem is solved.

The choice of μ (i.e. how to choose the resulting perturbed step), the choice of how to update the steps, and other factors lead to the different primal-dual interior-point algorithms in the literature. There are of course also differences that arise from applying these ideas to various optimization paradigms.

We end this section by mentioning that some authors have proposed using interior-point methods specifically for model predictive control. Wright [112] showed that the quadratic program associated with the dynamic portion of MPC can be interpreted as a monotone linear complementary problem (mLCP) and solved using interior-point methods. Rao, Wright, and Rawlings [80] showed that the problem structure of the mLCP can be exploited yielding a matrix factorization with a cost that grows linearly instead of cubicly with horizon length. Albuquerque et al. [1] have shown that these methods can outperform active-set methods for both control and simulation applications.

4.3 SIP Solution Methods

Different methods exist to solve problem (4.1), but they can be divided into the following five categories:

1. discretization methods (by grids and cutting planes);
2. local reduction methods;
3. exchange methods;
4. simplex-like methods;
5. descent methods.

The review papers by Hettich [42], Gustafson and Kortanek [41], Polak [70], Fiacco and Ishizuka [30], and Hettich and Kortanek [43] discuss these five areas in greater detail. We will focus on the first two.

Discretization and local reduction methods attempt to approximate problem (4.1) by imposing a finite number of constraints. The simplest way is by ordinary discretization.

4.3.1 Discretization Methods

Discretization methods attempt to discretize $\mathcal{U}$ and replace the problem (4.1) with an approximate problem. Let $\overline{\mathcal{U}} \subset \mathcal{U}$, $|\overline{\mathcal{U}}| < \infty$. The original problem is replaced by

$$\begin{aligned} \min_x \quad & f(x) \\ & h(x\,;\theta) = 0 \\ & g(x) \leq 0 \quad \text{for all} \quad \theta \in \overline{\mathcal{U}}. \end{aligned} \tag{4.11}$$

The set $\overline{\mathcal{U}}$ is typically called a grid. Unfortunately, for general problems, there does not exist a subset $\overline{\mathcal{U}}$ of $\mathcal{U}$ which yields identical solutions for (4.1) and (4.11). Also, if a finite sequence of finer and finer grids $\overline{\mathcal{U}}^{(n)}$ are used, it is not necessarily true that the accumulation points of (4.11) converge to the solution of (4.1). For these statements to be true, the grids must be chosen with care [43], [72].

Discretizing the set $\mathcal{U}$ has a parallel in optimal control. Set uncertainty is often used to parameterize a list of possible plants. The set can be thought of as a discretization of some continuous uncertainty description. For some linear problems, if the control algorithm has a given property for the members of the set, it also has the property for any plant within the set's convex hull. While this is true for some linear problems, cost-function bounds use quadratic constraints that destroy this property.

The papers by Grigorieff and Reemtsen [39] and Reemtsen [84] as well as the text by Polak [72] give conditions under which a solution of problem (4.11) is equivalent to (4.1) as sucessive grids are refined. The conditions are closely related to the theory of consistent approximations. While it can be shown that the convex hull ideas do not apply to cost-function bounds except for very special cases, the theory of consistent approximations provides a possible link between discretization theory and robust model predictive control.

Consider problem (3.9) with a linear state-space model and a set uncertainty description. The model parameters θ are the elements of the matrices (A, B) in the state-space model:

$$x_{k+1} = Ax_k + Bu_k.$$

The uncertainty description $\mathcal{U}$ contains an infinite number of possible plants (A, B). If, however, $\mathcal{U}$ was discretized as some grid $\overline{\mathcal{U}}$, then we could consider the discretized problem with only a finite number of (A, B). This is exactly the set uncertainty discussed by Badgwell and coworkers [4], [46]. They have shown that for robust linear MPC using cost function bounds and a set uncertainty description, the resulting optimization problem is a semi-definite program (SDP). The SDP can be solved directly or reformulated as a second-order cone program (SOCP). That means for any discretization, the resulting optimization problem is a SOCP. Cost-function bounds for a finite number of plants result in a quadratically constrained quadratic program:

$$\begin{aligned} &\min_x \quad \frac{1}{2}x^T Q_0\, x + q^T x \\ &\text{subject to} \\ &\qquad Ax = b \\ &\qquad \frac{1}{2}x^T Q_i\, x + q^T x \leq f_i, \qquad i = 1, \ldots, p. \end{aligned} \tag{4.12}$$

An extra quadratic constraint is appended to the nominal algorithm for each plant in the uncertainty description.

The current theory of cost-function bounds require that one of the members of the set be the true plant. However, Ralhan and Badgwell [79] have extended the theory to continuous uncertainty descriptions for certain model forms. The relationship between approximation theory for semi-infinite programming and set uncertainty descriptions may ultimately provide a way to choose a set uncertainty description which results in a solution equivalent to that for the problem with a continuous uncertainty description.

4.3.2 Local Reduction Methods

Local reduction methods rely on the fact that the semi-infinite program (4.1) can be reduced locally to a finite dimensional optimization problem. The term *local reduction* comes from noticing that

$$g(x\,;\,\theta) \leq 0 \quad \forall\, \theta \in \mathcal{U} \tag{4.13}$$

is equivalent to

$$\max_\theta (\, g(x\,;\,\theta),\ \theta \in \mathcal{U}\ ) \leq 0.$$

For problem (4.1) we let

$$\begin{aligned} G(x) \overset{def}{=} \max_\theta \ & g(x) \\ & h(x\,;\,\theta) = 0 \\ & \theta \in \mathcal{U}. \end{aligned} \tag{4.14}$$

The semi-infinite program then becomes

$$\begin{aligned} \min_x \quad & f(x) \\ & G(x) \leq 0. \end{aligned} \tag{4.15}$$

The function $G(x)$ may or may not be differentiable. The review by Polak [70] treats this general case. However, more progress can be made by treating $G(x)$ locally. The goal is to represent $G(x)$ locally near almost every $x_* \in \mathbb{R}^n$ by

$$G(x) = \max\left\{G^\ell(x) \ : \ \ell \in L\right\}$$

with smooth functions G^ℓ, $|L| < \infty$, defined on some neighborhood x_* [38]. This holds under mild regularity assumptions.

Denote all the local solutions of (4.14) by θ^ℓ, $\ell \in \overline{L}$. Problem (4.1) is equivalent to

$$\begin{aligned} \min_x \quad & f(x) \\ & h(x, \theta^\ell) = 0 \\ & G^\ell(x) \leq 0 \quad \text{for all} \quad \ell \in \overline{L} \end{aligned} \tag{4.16}$$

if the local solutions are nondegenerate regular points and the set $\overline{L}$ is finite. These assertions lead directly to the following algorithm:

```
Algorithm 1: Conceptual Reduction Method
(1)  Determine all the local solutions $\theta_*^\ell$, $\ell = 1, \ldots, n_\ell$ of
         $\theta_*^\ell = \operatorname*{argmax}_\theta \; g(x)$
                              $h(x\,;\,\theta) = 0$
                              $\theta \in \mathcal{U}$.
(2)  Starting with $x^{i,0} = x^i$, carry out $k_i$ steps of a nonlinear programming algorithm on the reduced problem
         $\min_x \; f(x)$
             $G^\ell(x) = g(x, \theta^\ell(x)) \le 0, \quad \ell = 1, \ldots, n_\ell$
     with the iterates denoted $x^{i,1}, \ldots, x^{i,k_i}$.
(3)  Set $x^{i+1} = x^{i,k_i}$.
```

The papers by Gramlich et al. [38] and Hettich and Kortanek [43] explain each of the steps and give guidelines on implementation. The most commonly used nonlinear programming algorithm used for the second step is successive quadratic programming.

4.4 Local Reduction and the Steady-State Target Calculation

In this section we consider the result of local reduction on the robust steady-state target calculation (3.15). The semi-infinite constraint

$$\begin{aligned} &h(x\,;\,\theta) = 0 \\ &g(x) \le 0 \qquad \forall\,\theta \in \mathcal{U} \end{aligned}$$

has the following specific form in the robust LP:

$$\begin{aligned} &\Delta y = G\Delta u \\ &A_y \Delta y \le b_y \qquad \forall\, G \in \mathcal{U}. \end{aligned} \tag{4.17}$$

Our discussion does not depend on the slack or bias terms in the constraint so we have ignored them for the sake of clarity. Because the constraint and model are linear we can directly substitute the model into the constraint to yield:

$$A_y G \Delta u \leq b_y \qquad \forall\, G \,\in \mathcal{U}. \tag{4.18}$$

There are two cases: ellipsoidal uncertainty and box uncertainty. We first consider ellipsoidal uncertainty $\mathcal{U} = \mathcal{E}$.

4.4.1 Ellipsoidal Uncertainty

Recall the ellipsoidal uncertainty description (3.12):

$$g \in \mathcal{E} \overset{def}{=} \left\{ \bar{g} + \beta\, V^{1/2} s \;:\; \|s\| \leq 1 \right\}.$$

When an uncertain linear constraint has data that is bound by an ellipsoid, the result is a second-order cone program (SOCP). When the semi-infinite problem (4.1) is convex with the constraints appearing as cones, Ben-Tal and Nemirovski [6, 7] have shown that the problem can be recast as a finite dimensional convex optimization. This is based in part on the work by Nesterov and Nemirovski [65] who introduced the idea of a second-order cone representable function (in other words, a function that can be cast as a second-order cone). They showed that a semi-infinite optimization with a second-order cone representable constraint can be better interpreted as an optimization over a second-order cone. Ben-Tal and Nemirovski [6] considered the following general linear program:

$$\begin{aligned} &\min_{x} \quad c^T x \\ &\text{subject to} \\ &\qquad\qquad a_i^T x \leq b_i \quad \forall\, a_i \in \mathcal{E}_i, \quad i = 1, \ldots, m, \end{aligned}$$

with uncertainty in the data a_i described by the ellipsoid:

$$a_i \in \mathcal{E}_i \overset{def}{=} \left\{ \tilde{a}_i + \beta\, V_i^{1/2} s \;:\; \|s\| \leq 1 \right\}.$$

Local reduction lets us recast the semi-infinite constraint

$$a_i^T x \leq b_i \quad \forall\, a_i \in \mathcal{E}_i, \quad i = 1, \ldots, m$$

as a maximization over the ellipsoid:

$$\max\{a_i^T x : a_i \in \mathcal{E}_i\}. \tag{4.19}$$

Maximization over a linear function over a convex and closed ellipsoid, however, has an unique analytic solution

$$\max\{a_i^T x : a_i \in \mathcal{E}_i\} = \bar{a}_i^T x + \beta \, \|V_i^{1/2} x\|. \tag{4.20}$$

Because there is only one local maximizer, which is in fact the global maximizer, of the reduced problem, we can rewrite the uncertain LP as follows:

$$\begin{aligned} &\min_x \quad c^T x \\ &\text{subject to} \\ &\qquad \bar{a}_i^T x + \beta \, \|V_i^{1/2} x\| \leq b_i \end{aligned} \tag{4.21}$$

for $i = 1, \ldots, m$. This is a second-order cone program because the constraint

$$\bar{a}_i^T x + \beta \, \|V_i^{1/2} x\| \leq b_i$$

is a second-order cone constraint. This allows us to interpret the semi-infinite constraint in the robust steady-state target calculation as a second-order cone constraint. In the robust LP, there is a matrix premultiplying the uncertain parameters.

$$\begin{aligned} &\min_x \quad c^T x \\ &\text{subject to} \\ &\qquad AGx \leq b, \quad \forall\, G \in \mathcal{U}. \end{aligned}$$

To see how this can be still cast as a second-order cone program, we rewrite the semi-infinite constraint component-wise:

$$\sum_j a_{ij} g_j^T x \leq b_i \quad G \in \mathcal{U}, \quad i = 1, \ldots, m, \tag{4.22}$$

which can be rewritten as

$$g^T(D_i x) \leq b_i \quad \forall g \in \mathcal{E}$$

where D_i is defined by:

$$D_i = [\text{diag}(a_{i1}\, e) \;\; \text{diag}(a_{i2}\, e) \;\; \cdots \;\; \text{diag}(a_{im}\, e)]^T,$$

and g is the vector of the rows of G stacked lengthwise with $e = \begin{bmatrix}1 \; 1 \; \cdots \; 1\end{bmatrix}^T$. From (4.19), the constraint becomes:

$$\bar{g}^T (D_i\, x) + \beta \, \|V^{1/2}\, D_i\, x\| \leq b_i \tag{4.23}$$

which is a second-order cone.

The result is the following problem:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subject to} \quad & \\ & \bar{g}^T (D_i\, x) + \beta \, \|V^{1/2}\, D_i\, x\| \leq b_i \end{aligned}$$

for $i = 1, \ldots, m$. This allows us to rewrite the semi-infinite program for the steady-state target calculation in model predictive control (3.15) as a SOCP:

$$\begin{aligned} \min_{\Delta u, \epsilon} \quad & c^T \Delta u + d^T \tilde{G} \Delta u + e^T \epsilon \\ \text{subject to} \quad & \\ & \bar{g}^T (D_i\, \Delta u) + \beta \, \|V^{1/2}\, D_i\, \Delta u\| + a_i^T b \leq b_{y_i} + \epsilon_i \\ & A_u \Delta u \leq b_u \\ & \epsilon \geq 0. \end{aligned} \tag{4.24}$$

for $i = 1, \ldots, m$. This optimization problem must be solved at every controller execution. Because the subproblem is convex, we have an analytic expression for the unique global maximizer. The conceptual reduction algorithm of the previous section reduces to simply finding, by some nonlinear programming technique, the solution of the second-order cone program. However, we can by our choice of approaches take advantage of the structure of the SOCP. There are two principal advantages to casting the problem in this form:

- First and foremost, the original semi-infinite optimization is recast in the form of a standard optimization problem with a finite dimensional constraint.
- Second, there has been tremendous activity in extending primal-dual interior-point methods to SOCP's—resulting in new efficient solution methods.

In the following section we describe second-order cone programming and how primal-dual interior-point methods can be used to exploit its inherent convexity.

4.4.2 Second-Order Cone Programming

A general second-order cone program (SOCP) has the following form:

$$\begin{aligned} \min_x \; & f^T x \\ & \|A_i x + b_i\| \leq c_i^T x + d_i, \quad i = 1, \ldots, N \\ & Gx = g, \end{aligned} \tag{4.25}$$

where $x \in \mathbb{R}^n$, $b_i \in \mathbb{R}^m$, and $g \in \mathbb{R}^p$. The operation $\|\cdot\|$ is the standard Euclidean norm; and A_i, f, c_i, d_i and G are of appropriate dimension. A wide variety of nonlinear convex optimization problems can be cast as SOCPs (see e.g. [56]), including linear and quadratic programs as special cases. An excellent reference for interior-point methods for second-order cone programming is the text by Nesterov and Nemirovski [65].

This is by no means an exhaustive summary or discussion; our goal is to provide a more complete picture of the robust LP and facilitate the use of interior-point methods as a tool in model predictive control. For a broader perspective, see the review by M. Wright [111].

The most promising algorithms for solving second-order cone programs are primal-dual interior-point methods. The goal in any interior-point method is to reduce the objective function while keeping the iterates strictly feasible with respect to the inequality constraints and, in the limit of a solution, satisfy the equality constraints as well. The application of these methods has been extended from their original use in solving LPs to numerous other optimization paradigms such as quadratic and semi-definite programming.

One of the reasons why researchers are focusing more attention on second-order cone programming is that any quadratically constrained quadratic program can be recast as a SOCP. In fact, linear programming, quadratic programming, second-order cone programming, and semi-definite programming are all special cases of optimizations over symmetric or self-scaled cones [65, 66, 67]. Many engineering and control applications can be cast as semi-definite programs [104, 114] or second-order cone programs. Boyd, Crusius and Hansson [10] show how SOCPs can be used in optimal control. They describe a robust optimal control problem in which the ℓ_∞ norm of the cost function (i.e. the peak tracking error) is minimized for uncertain impulse response coefficients in a finite impulse response model. They show that the problem can be solved efficiently as a SOCP. They also illustrate how SOCPs can be used in the optimal design of feedback controllers. This attention has led to the development of new optimization algorithms. There is some evidence [105] to suggest that while some SOCPs can be solved using tailored methods, a more general nonliner optimization code may be more efficient. This has yet to be proven in practice. The duality theory for these problems gives rise to a general complementarity condition. That condition combined with primal and dual feasibility form a square nonlinear system of equations which in principle determine the optimal solution.

In order to explain how the primal-dual interior-point methodology is used, consider the following SOCP:

$$\begin{aligned} &\min_{\hat{x}} \; f^T\hat{x} \\ &\text{subject to} \\ &\qquad \|A_i\hat{x} + b_i\| \le c_i^T\hat{x} + d_i, \quad i = 1, \ldots, n \\ &\qquad G\hat{x} = g. \end{aligned} \tag{4.26}$$

The first step is to recast (4.26) in standard form:

$$\begin{aligned} &\min_x \; c^T x \\ &\text{subject to} \\ &Ax = b \\ &x \geq_\kappa 0 \end{aligned} \tag{4.27}$$

where the optimization has been cast in terms of the new decision variable $x \in \mathbb{R}^\kappa$ defined as

$$x = \left(x_1^T \cdots x_n^T\right)^T$$

with

$$x_i = \begin{pmatrix} x_{i0} \\ x_{i1} \end{pmatrix} = \begin{pmatrix} c_i^T \\ A_i \end{pmatrix} \hat{x} + \begin{pmatrix} d_i \\ b_i \end{pmatrix}$$

and $x_{i0} \in \mathbb{R}$ and $x_{i1} \in \mathbb{R}^n$. The data of the problem are given in terms of the original data. Let

$$R = \begin{pmatrix} c_1^T \\ A_1 \\ c_2^T \\ A_2 \\ \vdots \\ A_n \\ c_n^T \end{pmatrix} \quad \text{and} \quad r = \begin{pmatrix} d_1 \\ b_1 \\ d_2 \\ b_2 \\ \vdots \\ d_n \\ b_n \end{pmatrix}.$$

Then

$$\begin{aligned} Rc &= f \\ AR &= G \\ Rb &= Rg - r. \end{aligned}$$

This, of course, assumes R^{-1} exists. When R^{-1} does not have full row rank, the problem can be reduced to an equivalent problem that does. The notation $x \geq_\kappa 0$

means $x \in \mathcal{K}$ where $\mathcal{K}$ is the Cartesian product of second-order cones:

$$\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_n.$$

The ith second-order symmetric cone is given by:

$$\mathcal{K}_i = \left\{ x_i = (x_{i0}, x_{i1}) \mid x_{i0}^2 - x_{i1}^T x_{i1} \geq 0, x_{i0} \geq 0 \right\}.$$

The dual of (4.27) is

$$\begin{aligned} &\min_{y,z} \quad b^T y \\ &\text{subject to} \\ &\qquad A^T y + z = c \\ &\qquad z \geq_{\mathcal{K}^*} 0 \end{aligned} \tag{4.28}$$

where $z \in \mathbb{R}^K$ and $y \in \mathbb{R}^K$ are the dual variables or Lagrange multipliers for the problem. The vector z possesses a structure similar to that of x.

$$z = \left(z_1^T \; \cdots \; z_n^T \right)^T$$

with

$$z_i = \begin{pmatrix} z_{i0} \\ z_{i1} \end{pmatrix}$$

and $z_{i0} \in \mathbb{R}$ and $z_{i1} \in \mathbb{R}^n$. It lies in the second-order cone defined by:

$$\mathcal{K}^* = \mathcal{K}_1^* \times \cdots \times \mathcal{K}_n^*$$

with

$$\mathcal{K}_i^* = \left\{ z_i = (z_{i0}, z_{i1}) \mid z_{i0}^2 - z_{i1}^T z_{i1} \geq 0, z_{i0} \geq 0 \right\}.$$

While this notation may at first seem burdensome, it casts the SOCP (and in fact any optimization over a self-scaled cone) in a framework analogous to linear programming as in section 4.2. This is more than just a convenient change of variables. It helps to extend the theory associated with linear programming to a much larger class of problems of engineering significance.

Recall that the duality gap μ is the difference between the primal and dual objective functions:

$$\mu = c^T x - b^T y = c^T x - (Ax)^T y = x^T z. \tag{4.29}$$

It can, in a sense, be regarded as a measure of the distance from optimality. The primal problem is said to be strictly feasible if there exists a point x for which the inequality constraints of the primal problem hold with strict inequality: $x >_{\kappa} 0$. Likewise, the dual problem is said to be strictly feasible if there exists a point z for which the inequality constraints of the dual problem hold with strict inequality: $z >_{\kappa^*} 0$.

In 1994, Nesterov and Nemirovski showed that the duality gap is zero at optimality if the primal and dual problems for the second-order cone program are strictly feasible. The proof for this is given in their text [65]. It is well known that this also holds true for any optimization over self-scaled cones. The primary goal of any interior-point method is to reach optimality by driving the duality gap to zero while satisfying primal and dual feasibility as well as complementarity (sometimes called the complementarity slackness condition).

The complementarity condition for the SOCP is given by:

$$x \circ z = 0,$$

where the binary operation "$\circ$" is defined as follows:

$$x \circ z \overset{def}{=} \begin{pmatrix} x^T z \\ x_{i0} z_{i1} + x_{i0} z_{i1} \end{pmatrix}$$

for $x \in \mathbb{R}^K$ and $z \in \mathbb{R}^{\kappa}$. The binary operation forms what is known as a Euclidean Jordan algebra [28, 29, 27], [26] or more generally an associative algebra [62]. It has very close connections to interior-point algorithms. The algebra can be viewed as the result of partitioning the vector x_i into a scalar component x_{i0} and a vector component x_{i1}. Associated with an element $x \in \mathbb{R}^K$ is the matrix $mat(x)$ defined to be $diag[X_1, \ldots, X_n]$ with

$$X_i = \begin{pmatrix} x_{i0} & x_{i1}^T \\ x_{i1} & x_{i0} I \end{pmatrix}.$$

We can now see that

$$x \circ z = mat(x)z = mat(z)x.$$

This lets us write Karush-Kuhn-Tucker (KKT) conditions for the primal and dual problems as follows:

$$F(x, y, z) = \begin{pmatrix} Ax - b \\ A^T y + z - c \\ Xz \end{pmatrix} = 0, \qquad (x, z) \geq_{\kappa} 0 \tag{4.30}$$

where we have denoted $X = mat(x)$. Applying any Newton type method on (4.30) yields the following linear system:

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} b - Ax \\ c - A^T y - z \\ Xz \end{pmatrix} \tag{4.31}$$

where $Z = mat(z)$.

The nonlinear complementarity condition of (4.31), $Xz = 0$, is generally relaxed in some specific way to keep the iterates away from the boundary and inside the second-order cone. The specific way in which complementarity is relaxed produces different search directions and thus different algorithms. A large class of these algorithms have been shown to be polynomial [90] [62], that is, the effort required to solve the problem to a given tolerance can be represented as a polynomial in the problem data.

Given some relaxed form of (4.31), there are two general classes of primal-dual interior-point methods: path-following algorithms and potential-reduction algorithms. We will consider the potential-reduction algorithm of Nesterov and Nemirovski [65]. This is the method used to solve the examples of the next chapter.

A full explanation of Nesterov and Nemirovski's potential-reduction algorithm can be found in the paper by Lobo et al. [56]. If x_{i1} is zero, the SOCP reduces to an LP and this algorithm reduces to Ye's potential-reduction algorithm [115] for linear programming. The potential function for the algorithm is given by

$$\psi(x, z) = (2N + \nu\sqrt{2N}) \ln \mu - \sum_{i=1}^{n} (\ln \gamma(x_i) + \ln \gamma(z_i)) - 2N \ln N, \tag{4.32}$$

where ν is an adjustable parameter and $\ln \gamma$ is the barrier function:

$$\gamma(x_i) = \gamma(x_{i0}, x_{i1}) = \begin{cases} (x_{i0}^2 - \|x_{i1}\|) & \text{for } x \geq_{\kappa} 0 \\ \infty & \text{otherwise.} \end{cases}$$

The potential function keeps the iterates away from the boundary by acting as a barrier for pair (x, z), balancing duality and feasibility. As ψ approaches $-\infty$, μ approaches 0, and (x, z) approaches optimality.

The actual steps of the algorithm are as follows:

Algorithm 2: Nesterov and Nemirovski's Potential-Reduction Algorithm

```
(1)   Generate a search direction by solving a linear system similar to
      (4.31) (see [65] or [56] for details).
(2)   a Perform a plane search on the potential function to determine
      the fraction of step to take.
(3)   Update the iterates.
```

One characteristic of potential-reduction algorithms is that under mild conditions, they guarantee a fixed amount of decrease in the potential function at each iteration. They also tend to perform well in practice.

Where potential-reduction algorithms use a potential function to keep the iterates strictly feasible, path-following algorithms employ the concept of a central path. There is a larger body of theory associated with path-following algorithms than there is with potential-reduction algorithms. However, potential-reduction algorithms can generally outperform path-following algorithms in practice. The search directions generated by path-following algorithms are generally the same as potential-reduction algorithms. However, the methods they use to stay away from the boundary differ.

The idea of the central path for second-order cone programming is similar to that for linear programming. It comes from relaxing or perturbing the complementarity condition of (4.30). The perturbed form of complementarity is given by:

$$Xz = \nu\, \mathrm{e}. \tag{4.33}$$

This results in a set of perturbed KKT (PKKT) conditions:

$$F_\nu(x,y,z) = \begin{pmatrix} Ax - b \\ A^T y + z - c \\ \nu e - Xz \end{pmatrix} = 0, \qquad (x,z) \geq_{\mathcal{K}} 0.$$

The central path is defined as the set of solutions $(x, y, z) \in \mathcal{K} \times \mathcal{K} \times \mathbb{R}^m$ for which $F_\nu(x, y, z) = 0$ for all $\nu > 0$. Just as in linear programming, the central path depends continuously and analytically on ν. Any accumulation point of $F_\nu = 0$ as ν tends to zero is a solution of (4.30).

Let us set $\nu = \sigma\mu$ where σ is a centering parameter and μ is the duality gap. We can then consider a Newton step on the PKKT conditions.

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} b - Ax \\ c - A^T y - z \\ \sigma\mu\, e - X z \end{pmatrix}. \tag{4.34}$$

Different search directions are generated by scaling the last equation, solving the system of equations, then unscaling. In fact the numerous search directions generated for semidefinite programming all have direct parallels in second-order cone programming [62].

Let us rewrite (4.34) in so-called normal form:

$$(AZ^{-1}XA^T)p_y = AZ^{-1}X(c - \sigma\mu X^{-1}e - A^T y) + b - Ax \tag{4.35a}$$

$$p_z = c - A^T y - z - A^T p_y \tag{4.35b}$$

$$p_x = -Z^{-1}X(p_z - \sigma\mu X^{-1}e - z) \tag{4.35c}$$

where we solve the smaller linear system (4.35a) for p_y and find p_z and p_x by substitution. This is the main computational effort of the algorithm and depending upon the nature of $AZ^{-1}XA^T$, various structures can be exploited. Once a search direction is computed, the fraction of the Newton step to take is computed using either a line search, or commonly, a fraction of the step to the boundary. The iterates are then updated.

We now give the predictor-corrector path following algorithm of Monteiro and Tsuchiya [62] for second-order cone programming. It can be considered a direct extension of Mizuno, Todd, and Ye's predictor-corrector algorithm [61] for linear-programming.

The predictor-corrector algorithm is composed of two steps. In the predictor step the centering parameter σ is set to zero, while in the corrector step, the centering parameter is set to one. The algorithm uses a value of

$$\mu = x^T z/n$$

and requires an initial point (x_0, y_0, z_0) in a neighborhood $\mathcal{N}_2$ of the central path. The algorithm is given as follows:

```
Algorithm 3: Monteiro and Tsuchiya's Path-Following Algorithm
Choose a constant $0 < \tau \leq 1/30$ and set $\mu_0 \overset{def}{=} \mu(x_0, z_0)$
Given a starting point $(x^0, y^0, z^0) \in \mathcal{N}_2(\tau)$, a tolerance $\epsilon \in (0, 1)$ for
$k = 1$ to $\infty$
  if $\mu_k \leq \epsilon\mu_0$ then stop
  Find $(p_x^k, p_y^k, p_z^k)$ from (4.35) with $\sigma = 0$
  Calculate the fraction of the step to take, $\alpha_k$, that if implemented
  would keep the iterates inside the enlarged neighborhood $\mathcal{N}_2(2\tau)$
  Update the iterates:
      $(\hat{x}^{k+1}, \hat{y}^{k+1}, \hat{z}^{k+1}) := (x^k, y^k, z^k) + \alpha_k (p_x^k, p_y^k, p_z^k)$
  Find $(\hat{p}_x^k, \hat{p}_y^k, \hat{p}_z^k)$ from (4.35) with $\sigma = 1$.
  Update the iterates:
      $(x^{k+1}, y^{k+1}, z^{k+1}) := (\hat{x}^{k+1}, \hat{y}^{k+1}, \hat{z}^{k+1}) + (\hat{p}_x^k, \hat{p}_y^k, \hat{p}_z^k)$
  Update $\mu$:
      $\mu_{k+1} := \mu(x^{k+1}, z^{k+1})$
  return
```

The definition of $\mathcal{N}_2$ can be found in [62]. As a final note, there are also extensions of other LP algorithms to second-order cone programming. For instance, the algorithm by Alizadeh and Schmieta [2] is a direct extension of Mehrota's LP predictor-corrector algorithm [60].

We conclude this section by mentioning that there are several direct extensions of path-following algorithms for linear programming to second-order cone programming. The algorithm of Monteiro and Tsuchiya [62] for second-order cone programming is a direct extension of Mizuno, Todd, and Ye's [61] predictor-corrector algorithm for linear-programming. Also, the algorithm by Alizadeh and Schmieta [2] is a direct extension of Mehrota's [60] LP predictor-corrector algorithm. We now explain how certain second-order cone programs can be interpreted stochastically with the semi-infinite constraint cast probabilistically.

4.4.3 SOCP and Stochastic Programming

For some classes of problems the semi-infinite constraint in (4.1) is interpreted in a stochastic framework. The goal is to satisfy the constraint in a probabilistic sense:

$$\text{prob}(\ g(x\,;\theta) \leq 0\ ) \ \geq\ 1-\alpha, \tag{4.36}$$

with $\theta \in \mathbb{R}^m$ a randomly distributed variable with a given mean and variance and $1-\alpha$ some generally high probability level. This classical way of dealing with uncertainty as a probabilistic constraint is known as *stochastic programming.* In fact, many robust optimization problems first arise as stochastic optimization problems. Stochastic programming is commonly used in the fields of operations research and managerial science, whereas semi-infinite programming is applied more often to engineering problems.

These ideas have recently been applied to model predictive control. Schwarm and Nikolaou have used stochastic programming as a way to address uncertainty in output constraint satisfaction for dynamic MPC. They consider an impulse response model with coefficients having a given mean and variance. In [92] they pose the standard output constraint as a probabilistic constraint which in turn is cast as an equivalent deterministic constraint. The optimization is then solved using nonlinear programming methods. The deterministic constraint they pose is actually a second-order cone. The techniques we discuss later for the robust LP can be used just as effectively on the dynamic portion of MPC with probabilistic output constraints. Instead of using nonlinear programming methods, it is possible to use recently developed interior-point methods to solve the optimization problem. Schwarm and Nikolaou extended their approach in [91]. Instead of simply asking that the constraints hold to a given probability, they minimize the expectation that the constraints will be violated. This is known as a stochastic program with recourse. These approaches for addressing output constraint satisfaction show great promise.

It has been known for some time in stochastic programming that a probabilistic random linear constraint (i.e. equation (4.36) for linear $g(x,\theta)$ and a normal distribution) can be cast as an equivalent nonlinear constraint (see e.g. §10.4 of [110] , §10.3

of [74]). However, the nonlinear equivalent was not recognized as a second-order cone until recently. Lobo et al. [56] make this connection explicit in their summary of SOCP applications.

Consider the linear probabilistic constraint:

$$\text{prob}(\, a_i^T x \leq b_i \,) \geq 1 - \alpha. \tag{4.37}$$

If a_i has mean $\bar{a}_i$ and covariance V_i, then $a_i^T x$ has mean $\bar{a}_i^T$ and variance $x^T V_i x$. The constraint can be written as an equivalent constraint with zero mean and unit variance:

$$\text{prob}\left(\frac{a_i^T x - \bar{a}_i^T x}{\sqrt{x^T V_i x}} \leq \frac{b_i - \bar{a}_i^T x}{\sqrt{x^T V_i x}} \right) \geq 1 - \alpha.$$

Thus the probability can be given by:

$$\Phi\left(\frac{b_i - \bar{a}_i^T x}{\sqrt{x^T V_i x}} \right) \geq 1 - \alpha,$$

or, equivalently,

$$\bar{a}_i^T x + \beta \, \|V^{1/2} x\| \leq b_i,$$

which is a second-order cone constraint. Here

$$\beta = \Phi^{-1}(1 - \alpha)$$

with

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}s^2} ds. \tag{4.38}$$

While this is a special case of the general probabilistic constraint, it is still very important. A general probabilistic constraint (4.36) is very computationally expensive since it involves a semi-infinite, multivariate probability integral to evaluate the associated probability distribution function. A second-order cone, however, can be evaluated quite efficiently.

This allows us to interpret the semi-infinite constraint in the robust steady-state target calculation probabilistically. It can be thought of as requiring the output

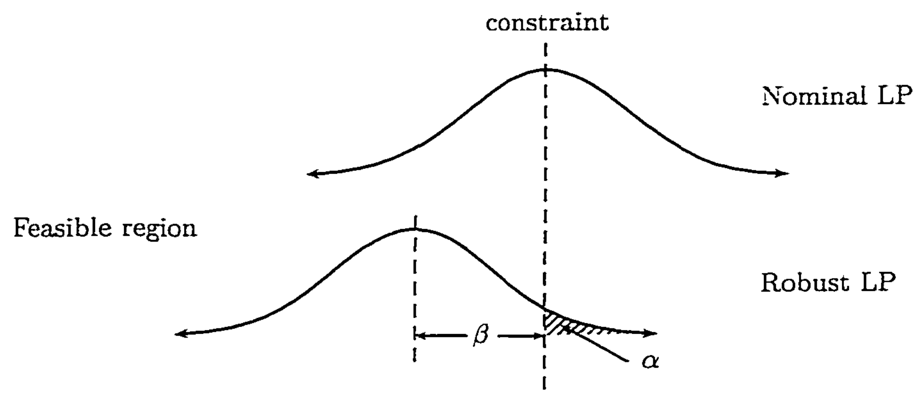

Figure 4.2: Probabilistic Interpretation of the Semi-Infinite Constraint in the Robust LP

constraints to hold a given probability level. Figure 4.2 illustrates this idea. The location of the constraint $\tilde{a}_i^T x \leq b_i$ given the nominal value of a_i is represented by the vertical lines. Possible realizable locations of the constraint are represented by the overlaid normal distribution. Equation (4.37) requires the probability that any realizable location of the constraint be less than or equal to $1 - \alpha$. This *shifts* the distribution. The new constraint is more restrictive and its location is at the mean of the shifted distribution.

In the next section we consider problem (4.17) when the uncertainty description is given by a polytope $\mathcal{U} = \mathcal{B}$.

4.4.4 Box Uncertainty

Let us consider the case when the uncertainty description is given by a polytope or box (3.3):

$$g \in \mathcal{B} \overset{def}{=} \left\{ g \ : \ A_g \, g \leq b_g \right\}.$$

If the subproblem in a semi-infinite optimization maximizes a linear function subject to a set of linear constraints when there is uncertainty in the problem data, then the semi-infinite LP can be cast as a standard LP with more constraints [21] [102]. Assume the uncertain parameter θ corresponds to the elements of the matrix A_g. The goal is to solve the following problem:

$$\begin{aligned} &\min_x c^T x \\ &\text{subject to} \\ &\qquad AGx \leq b \quad \forall g \in \mathcal{B}, \end{aligned}$$

where $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$. A straight-forward way to solve the problem is to simply enumerate all the possible values of A. The reason this is a valid approach relies on the fact that the solution of an LP will always lie on the boundary of the feasible set. So by enumerating all the possible values, the semi-infinite problem can be replaced by a finite problem with an increased number of inequality constraints. For every row $i = 1, \dots, m$ of A, there are 2^n possible permutations of the entries, created by replacing the n elements with their upper and lower bounds respectively. Doing so replaces the previous problem with the following problem with $m\,2^n$ constraints:

$$\begin{aligned} &\min_x \quad c^T x \\ &\text{subject to} \\ &\qquad A^* x \leq b, \end{aligned}$$

where A^* is the matrix created by enumerating all the possible combinations of A. For a 10-by-10 system with simple bounds on the outputs, this corresponds to over 20,000 constraints that would need to be added to the nominal LP. The problem size grows exponentially. For real-time applications such as control, this may be unacceptable since constraints need to be dropped or added as sensors fail or transmitters go off-line.

If the simplex method is used to solve the problem, it can be expected to slow down as it is forced to consider a larger and larger number of vertices. Other methods, such as interior-point methods will likely produce better solution times as the number of inequalities increases. There is a limit, however, to the usefulness of this approach because for large semi-infinite problems, this will produce enormous equivalent finite problems. In this case, ellipsoid uncertainty may result in shorter solution times.

4.5 Local Reduction and the Robust Dynamic Calculation

In this section we use the method of local reduction to solve the semi-infinite program associated with the robust dynamic MPC calculation. The semi-infinite constraint

$$\begin{aligned} h(x\,;\theta) &= 0 \\ g(x) &\leq 0 \qquad \forall \theta \in \mathcal{U} \end{aligned} \tag{4.39}$$

has the following specific form in the robust dynamic algorithm:

$$y = P(u)\theta \tag{4.40}$$

$$y(u)^T Q\, y(u) + u^T R\, u \leq y(\hat{u})^T Q\, y(\hat{u}) + \hat{u}^T R\, \hat{u} \qquad \forall \theta \in \mathcal{E}. \tag{4.41}$$

The output depends implicitly upon the model parameters θ. The vector $\hat{u}$ is a known constant. Because the constraint is quadratic and not linear, it is not obvious whether substituting the equality constraint into the inequality constraint will be beneficial. When (4.40) is substituted into (4.41) the approach is known as the *black-box* approach. If y and u are solved for simultaneously, the approach is referred to as the *all-at-once* approach.

4.5.1 All-At-Once vs. Black-Box

There are two general way to formulate optimization problems which arise in optimal-control applications: all-at-once and black-box. In this section we comment on the structure of each. This first part of this discussion dealing with the implicit function theorem follows the discussion given by Vicente [106], section 4.2.

While we cast the general optimization problem as (4.1) and consider x as the decision variable, in fact, we know that (4.1) has a very specific structure and x is partitioned into two types of variables: manipulated variables or inputs u, and controlled variables or outputs y. It is understood that in our discussion reference to outputs can just as well be interpreted as reference to states with any loss of generality.

$$x = \begin{pmatrix} u \\ y \end{pmatrix}.$$

The point we want to convey in this section does not depend on the fact that x is also composed of other variables, nor does it depend on the presence of inequalities. Therefore, we will consider only the simplified equality constrained problem

$$\begin{gathered} \min \quad f(u, y) \\ \text{subject to} \\ h(u, y\,;\,\theta) = 0. \end{gathered} \tag{4.42}$$

This problem can also be cast in what is known as reduced form through use of the implicit function theorem. Suppose there exists open sets $\mathcal{V}$ and Θ such that for all $u \in \mathcal{V}$ and $\theta \in \Theta$ there exists a solution y of $h(u, y\,;\,\theta) = 0$ such that the matrices $h_y(x)$ and $h_\theta(x)$ are invertible for all x with $u \in \mathcal{V}$, $\theta \in \Theta$, and $h(u, y\,;\,\theta) = 0$. Then the implicit function theorem guarantees the existence of a differentiable function

$$y(u\,;\,\theta) : \mathcal{V} \times \Theta \to \mathbb{R}^m$$

defined by

$$h(u, y(u\,;\,\theta)\,;\,\theta) = 0,$$

and problem (4.42) is equivalent to

$$\min \hat{f}(u\,;\,\theta) \triangleq f(u, y(u\,;\,\theta)). \tag{4.43}$$

This is the so-called *black-box* approach in which the nonlinear constraints are not visible to the optimizer. Its solution is part of the evaluation of the objective function $\hat{f}(u\,;\,\theta)$. The reduced problem can be solved by a Newton-like method. For optimal control problems, many algorithms follow this approach [11], [69]. This is the technique by which Badgwell [4] and Ralhan [79] solved the robust dynamic MPC calculation.

In the black-box method a set of initial inputs are chosen. The model equations are then solved outside of the optimizer and the state values passed back to the minimization problem. The optimizer finds a local minimizer u of the objective function and the process is repeated with the new solution in an attempt to move toward optimality. In this method the optimizer never sees the equality constraints; they are transparent. Strict feasibility is maintained with respect to the constraints since at each iteration the model equations are solved for the states. This is somewhat attractive for MPC applications, because if the optimizer should fail at some iteration, the inputs, while not optimal, are feasible and can be sent into the plant.

If there are inequality constraints, they are cast in terms of only u and are handled by the optimizer in the same fashion as the objective function evaluation. This is straightforward for the input constraints; however, more work must be done to handle the state constraints. In some cases the states that are returned from the nonlinear equation solver may not be feasible since the solver has no information about state constraints.

In the so-called *all-at-once* approach the inputs and states are solved for simultaneously inside of the optimizer. In this method the goal is to move toward optimality and feasibility at the same time. The nonlinear equation $h(u, y\,; \theta) = 0$ is appended to the optimization problem as a nonlinear equality constraint.

In [8], Bequette outlines the two approaches, as well as others, in the framework of nonlinear control. Both approaches have been known to the optimal control community for some time with the black-box approach being the classical formulation. With new and powerful optimization algorithms and the ability to handle large problems efficiently, the all-at-once approach is becoming more and more attractive.

While the black-box approach offers a smaller problem to solve, it forces the (possibly) costly task of independently handling the equality constraints. However, any iterate in the black-box approach is always strictly feasible with respect to the equality constraints. The black-box approach requires the evaluation of the function, its gradient, and Hessian upon request. This is costly since the Hessian is dense [112]. The sensitivity of the model to all the variables needs to be known. Because of the dense Hessian, this is especially expensive for stiff models and models derived from flow-sheet simulations [9]. Also, unless the gradient and Hessian are user-supplied, they will need to be approximated by numeric integration and this may yield poor results [37].

Experience has shown the all-at-once approach, while increasing the size of the problem significantly, offers better performance and is less likely to fail at finding a solution, especially for problems with more than a few states [81]. Also, state constraint handling is easy in this formulation. One of the major advantages of taking the all-at-once approach is freedom of the optimizer to move toward both optimality and feasibility at the same time.

When addressing semi-infinite constraints such as (4.39), it is very appealing to take the black-box approach because the subproblem requires maximization over θ. In this case, the subproblem is

$$G(x\,;\,\theta) = \max_{\theta} \quad g(u, y(u\,;\theta)).$$

This has the same drawbacks mentioned above. The all-at-once approach for the subproblem keeps the model equations appended as an equality constraint:

$$\begin{aligned} G(x\,;\,\theta) = \max_{\theta} \quad & g(u, y) \\ \text{subject to} \quad & \\ & h(u, y\,;\,\theta) = 0. \end{aligned}$$

This can be advantageous when g has a special structure as in the robust dynamic MPC calculation.

4.5.2 SQP Local Reduction Method

In this section we investigate the conceptual reduction algorithm of section 4.3.2 in greater detail. We show that the robust dynamic MPC algorithm of section 3.3 has a single unique maximizer. We then show how an SQP method with augmented Lagrangian can be used for step 2 of the conceptual reduction algorithm.

Step 1: Find all the local maxima

The first step of the conceptual reduction algorithm of section 4.3.2 is to calculate all the local maxima of the subproblem. For the robust dynamic MPC algorithm we need to solve the following optimization:

$$\begin{aligned}
&\max_{\theta} \quad y(u)^T Q\, y(u) + u^T R\, u - y(\hat{u})^T Q\, y(\hat{u}) - \hat{u}^T R\, \hat{u} \\
&\text{subject to} \\
&\qquad y(u\,;\,\theta) = P(u)\theta \\
&\qquad \theta = \tilde{\theta} + \beta\, V^{1/2} s \\
&\qquad \|s\| \le 1
\end{aligned}$$

Since the optimization is with respect to θ we can simplify the objective, dropping constant terms.

$$\begin{aligned}
&\max_{\theta} \quad y(u)^T Q\, y(u) - y(\hat{u})^T Q\, y(\hat{u}) \\
&\text{subject to} \\
&\qquad y(u\,;\,\theta) = H(u)\theta \\
&\qquad \theta = \tilde{\theta} + \beta\, V^{1/2} s \\
&\qquad \|s\| \le 1
\end{aligned} \tag{4.44}$$

Problem (4.44) is the maximization of a quadratic functional subject to a ball constraint (see c.f. [79]):

$$\begin{aligned}
&\max_{s} \quad (s-b)^T A\, (s-b) \\
&\text{subject to} \\
&\qquad s^T s \le 1
\end{aligned} \tag{4.45}$$

with $s \in \mathbb{R}^n$,

$$b = -\beta\, V^{1/2} \tilde{\theta}$$

and

$$A = \beta^2 V^{1/2^T} \big(H(u)^T Q H(u) - H(\hat{u})^T Q H(\hat{u})\big) V^{1/2}.$$

A unique global solution of (4.45) is guaranteed to exist [34] if the set $F = \{s : \|s\| = 1\}$ is not empty and $\text{rank}\left[A^{1/2}\ I\right]^T = n$. These conditions are both satisfied for any A and b for this problem. Thus, there exists only one unique local maxima of the subproblem, and it can be found by solving problem (4.44).

Step 2: Reduction with an SQP augmented Lagrangian solver

In this section we describe the algorithm given in [38] for use in step 2 of the conceptual method described in section 4.3.2. The particular SQP method uses a augmented Lagrangian and a BFGS update for the Hessian.

SQP algorithms have been used very successfully for the solution of nonlinear programming problems. They are generally quasi-Newton type algorithms. The augmented Lagrangian of the reduced-problem (4.16) is defined to be:

$$L(x, \lambda, \rho) \overset{def}{=} f(x) + \sum_{\ell=1}^{n_\ell(x_*)} \lambda_\ell G^\ell(x) + \frac{\rho}{2} \sum_{\ell=1}^{n_\ell(x_*)} \left(G^\ell(x)\right)^2. \tag{4.46}$$

Then step 2 of the conceptual reduction algorithm is given by algorithm 4.

```
**Algorithm 4:** Augmented Lagrangian SQP with BFGS solver
Given $x^{i,0} = x^i$, $B_{i,0} = B_i \in \mathbb{R}^{n \times n}$ negative definite.
Then **for** $j = 1$ **to** $k_i$
    Compute a solution $s_j$ and multipliers $\lambda^{i,j}$ of the quadratic program:
        $\max_s \ \frac{1}{2} s^T B_{i,j-1} s + F_x(x^{i,j-1})^T s$
        subject to
            $G_x^\ell(x^{i,j-1})^T s + G^\ell(x^{i,j-1}) \leq 0$
    for $\ell = 1, \ldots, n_\ell$
    Compute a step length $\alpha_j$.
    Update
        $x^{i,j} = x^{i,j-1} + \alpha_j s^j$
    and
        $B_{i,j} = B + \frac{y^j (y^j)^T}{(y^j)^T s} - \frac{B_{i,j-1} s^j (B_{i,j-1} s^j)^T}{(s^j)^T B_{i,j-1} s^j}$
    where
        $y^j = L_x^{x^i}(x^{i,j}, \lambda^{i,j}, c) - L_x^{x^i}(x^{i,j-1}, \lambda^{i,j}, c)$
    **return**
    Set
        $B_{i+1} = B_{i,k_i}, \qquad x^{i+1} = x^{i,k_i}.$
```

The algorithm uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) secant update. In unconstrained optimization, BFGS updates the Hessian by a rank two modification. More information on secant updates can be found in the classical references [22], [23].

There are several common strategies for determining the step length α_j (see c.f.

[37], [13]). It is important to choose a step length which ensures the algorithm does not converge to an accumulation point which is not a minimizer (the Maratos effect). Also, near a solution α_j must converge to one $\alpha_j \to 1$ to ensure convergence properties. The algorithm is superlinearly convergent [38]. Other variants of the SQP algorithm can be found in [73], [99], and [103].

One of the advantages of the SQP solver is that the QP subproblem can be solved using primal-dual interior-point methods. The standard form QP

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q\, x + c^T x \\ \text{subject to} & \\ & Ax = b \\ & x \geq 0 \end{aligned} \tag{4.47}$$

where Q is symmetric positive semidefinite $n \times n$ matrix can be recast to form a monotone linear complementary problem (mLCP). The dual of (4.47) is

$$\begin{aligned} \max_{y,z} \quad & -\frac{1}{2} x^T Q\, x + b^T y \\ \text{subject to} & \\ & -Q\, x + A^T y + z = c \\ & z \geq 0. \end{aligned} \tag{4.48}$$

The KKT conditions for the primal and dual problems are:

$$F(x, y, z) = \begin{bmatrix} Ax - b \\ -Qx + A^T y + z - c \\ XZe \end{bmatrix} = 0, \qquad (x, z) \geq 0 \tag{4.49}$$

which is a mLCP. This system can be solved using the ideas of Section 4.2.

4.6 Summary of Robust MPC and Semi-Infinite Programming

Both the dynamic and the steady-state target calculations take the form of a semi-infinite program. By using the method of local reduction, we are able to recast the semi-infinite problems into equivalent problems with a finite number of constraints.

The steady-state target calculation can be reduced to a second-order cone program (SOCP). We make use of recent advances in primal-dual interior-point methods to solve the resulting optimization problem.

While the dynamic calculation cannot be reduced to a single nonlinear convex optimization, we showed the subproblem in the conceptual reduction algorithm has a unique global maximizer. An SQP local reduction method which uses the augmented Lagrangian SQP technique with a BFGS Hessian update can be used as the NLP for the conceptual reduction algorithm. The QP subproblem is then solved using primal-dual interior-point methods.

Chapter 5

Simulation Examples

In this chapter we consider several examples that illustrate the closed loop behavior of the robust MPC algorithms. We simulate the robust steady-state target calculation coupled to a nominal dynamic controller. The dynamic controller is a variant of the QDMC algorithm with a sum of moves constraint and is described in Appendix A.

The examples show instances when, because of model mismatch or ill-conditioning, the targets calculated by the nominal LP result in poor control, while the targets calculated by the robust LP provide much improved control. The simulations were run on a desktop PC with an Intel Pentium II processor using MATLAB with a MEX interface to the C routine SOCP by Lobo et al. [55]. The SOCP routine makes use of Nesterov and Nemirovski's [65] potential reduction algorithm described in Section 4.4.2.

5.1 SISO Example

Consider the simple single-input, single-output (SISO) steady-state model given by:

$$(y_k - y_{k-1}) = g\,(u_k - u_{k-1}) + b,$$

where u_k and y_k are the input and output at time k, g is the model gain, and b is the model bias. Assume that the gain is nominally

$$\bar{g} = \frac{1}{2},$$

but the true plant gain is given by

$$g_{act} = 2,$$

which is assumed to lie in some ellipsoid. Also, assume unit constraints on the input and output:

$$-1 \leq y \leq 1,$$
$$-1 \leq u \leq 1.$$

Let the objective be given by:

$$J_s = u - 3y.$$

The coefficients of u and y are such that the objective wants to drive the process toward large inputs and outputs.

For the sake of simplicity, we focus initially on steady-state control only. This means that the MPC control consists of only a steady-state target calculation, with the sample time chosen long enough such that the process reaches steady-state between each calculation.

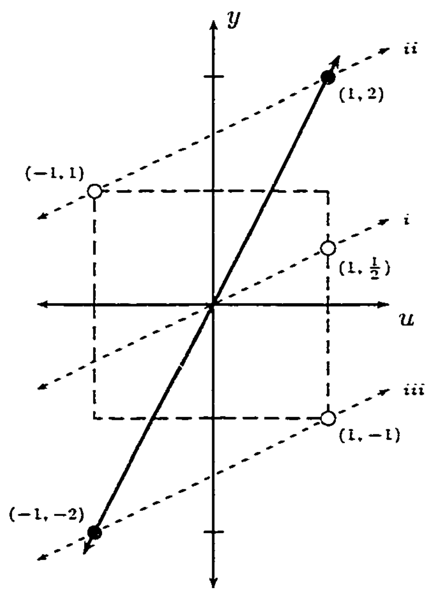

Figure 5.1: Model and Constraints in the Example: Solid Line—True Plant; Dashed Lines—Nominal Models

The simulation for the steady-state controller with the above data is shown by Figure 5.1. This is a phase portrait of the iterates as they evolve in time. The box in the figure represents the feasible region defined by the input constraints. The solid line represents true plant and has a slope $m = 2$. The dotted lines represent the nominal model at different time steps (including bias), each with a slope $m = \frac{1}{2}$.

We now trace the evolution of the closed loop system. In Figure 5.1, the open circles correspond to the predicted input and output. The filled circles correspond to the actual input and output. The numbers next to each circle indicate the value of the input output pair (u, y). If at time zero, the plant is at the origin with zero bias, then the model, indicted by line i, predicts the optimal steady-state should lie at $(1, \frac{1}{2})$. But because of mismatch, once the input $u = 1$ is injected, the true plant is at $(1, 2)$. Next bias is included in the model. This raises the nominal model line i to the new position ii. The new model has an intercept of $b = \frac{3}{2}$. This forces the model to agree with measured output. The optimal steady-state solution using the nominal model ii for this time-step is at $(-1, 1)$. Because of model error, however, the true plant is actually at $(-1, -2)$. We again update the bias, yielding line iii. The new

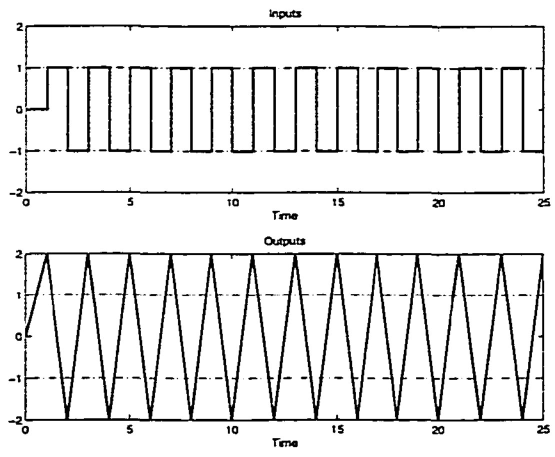

Figure 5.2: Steady-State Only Controller: Nominal LP

solution is at $(1, -1)$. Injecting $u = 1$ into the plant we see that the process is moved in the opposite direction and the closed loop system begins to cycle. This cycling is illustrated in Figure 5.2 where the inputs and outputs are plotted as functions of time. For this case the nominal LP drives the input from one side of the feasible space to the other at each time step. The output shows similar behavior except that it is *never feasible at the sample times.* Now consider what would happen if the nominal LP were replaced with the proposed robust LP.

Assume we know more than just the mean model gain. If we know to a 95 percent probability that the gain is between 2 and −1 (or equivalently that $\Delta g = 1\frac{1}{2}$), then we can control the process more efficiently with the robust LP. The solution of the

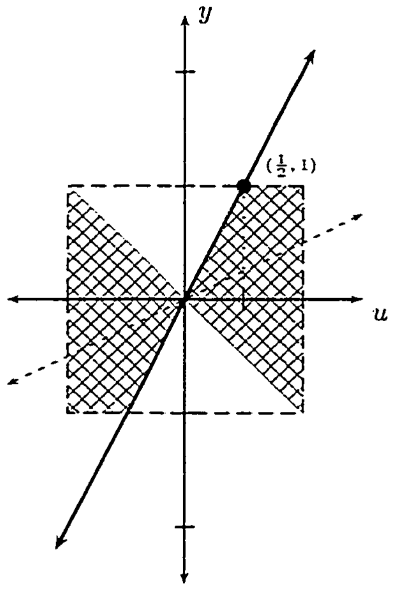

Figure 5.3: Model and Constraints in the Example: Solid Line—True Plant; Dashed Line—Nominal Model

robust LP can be found graphically (see Figure 5.3). The hashed region in the figure represents the gain uncertainty. The optimal steady-state solution is the input for which any realizable output remains feasible: $(u, y) = (\frac{1}{2}, 1)$. At the next iteration when bias is included in the calculation, the steady-state remains the same (see Figure 5.4). We can see the robust LP prevents oscillations in the input and output and prevents violation of the output constraint.

We now include dynamics in the problem. For simplicity, let the system be first

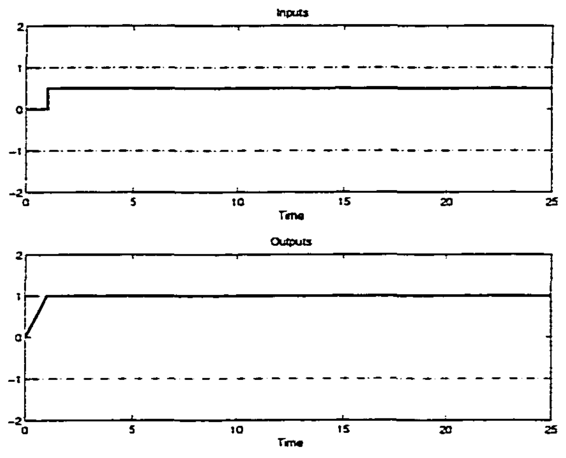

Figure 5.4: Steady-State Only Controller: Robust LP

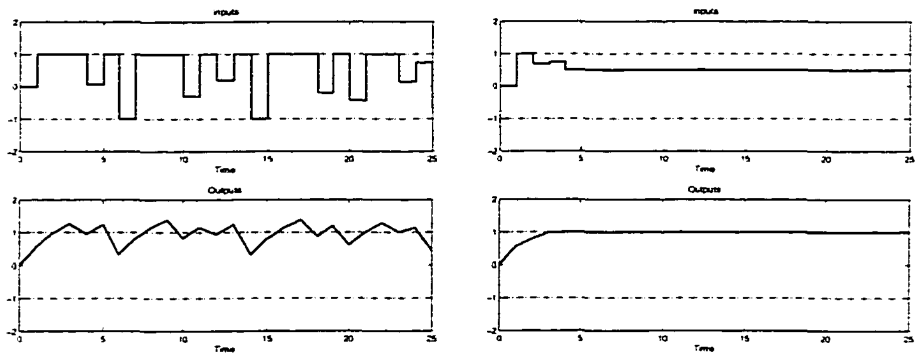

Figure 5.5: Conventional MPC Controller using the nominal LP (left) and robust LP (right)

order with a time-constant of 3 for both the nominal model and plant:

$$\tilde{g} = \frac{1}{3s+1}0.5, \quad g_{act} = \frac{1}{3s+1}2.0$$

We use a step-response model with 30 coefficients to describe both the nominal model and the plant. The dynamic calculation uses the modified version of the QDMC algorithm discussed in Section 4.3 with a control horizon of $c = 5$, a prediction horizon of $p = 35$, and simple tuning weights of $w_u = 0$ and $w_y = 1$.

As illustrated in Figure 5.5, the nominal LP cycles because of model mismatch, while the robust LP does not. While the cycling is not as strong as in the steady-state controller, the mechanism producing the cycling is the same. The steady-state algorithm is trying to find a target that agrees with the feedback it has received while satisfying the constraints. Because the model is it using is not correct, the steady-state target iterates end up 'bouncing' around their correct values. The input, in fact, is sometimes forced to cross from one side of its operating region to the other, and the upper output constraint is violated. The robust LP, on the other-hand, moves the system to the best operating point given the uncertainty of the problem.

While this example is somewhat pathological, it demonstrates the importance of addressing uncertainty in the steady-state target calculation. Next we consider a more complicated, multiple-input, multiple-output example.

5.2 Shell Fundamental Control Problem

Let us consider the performance of the robust LP on a modified subset of the Shell fundamental control problem [75] illustrated in Figure 5.6. The process is a heavy oil fractionator in which a gaseous feed stream is separated by removing heat. We will only consider the two-by-two subsystem:

$$y = \begin{bmatrix} \frac{4.05e^{-30s}}{50s+1} & \frac{1.77e^{-30s}}{60s+1} \\ \frac{5.39e^{-20s}}{50s+1} & \frac{5.72e^{-15s}}{60s+1} \end{bmatrix} u. \tag{5.1}$$

The outputs $y = \begin{bmatrix} y_1 \ y_2 \end{bmatrix}^T$ are the top end point y_1 and side end point y_2. The inputs $u = \begin{bmatrix} u_1 \ u_2 \end{bmatrix}^T$ are the top draw u_1 and side draw u_2. The time constants and dead-times are are assumed to be in units of minutes and the variables are all normalized. The problem has a known steady-state gain uncertainty

$$G = \tilde{G} \pm \Delta G \tag{5.2}$$

where

$$\tilde{G} = \begin{bmatrix} 5.04 & 1.77 \\ 5.39 & 5.72 \end{bmatrix} \quad \text{and} \quad \Delta G = \begin{bmatrix} 2.11 & 0.39 \\ 3.29 & 0.57 \end{bmatrix}. \tag{5.3}$$

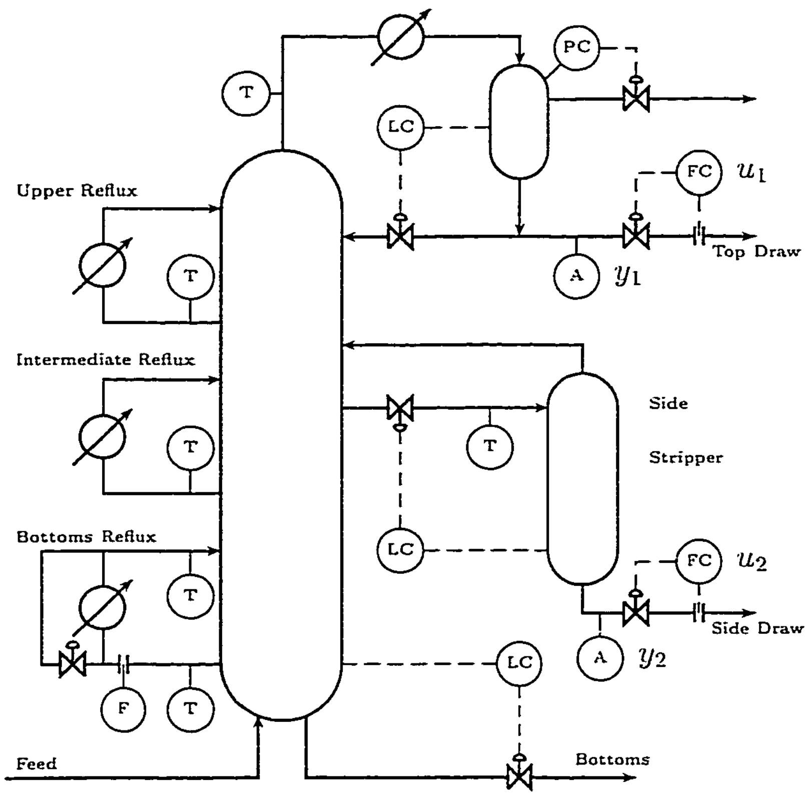

Figure 5.6: Heavy Oil Fractionator

The model is linear, first-order with dead-time. We assume the dead-times are known exactly and investigate the effect of gain uncertainty on the closed loop system. The system (5.1) will be represented using a 60 coefficient step-response model with a 5 minute sampling time. The constraints of the system are

$$\begin{aligned} -0.5 &\le u_1 \le 0.5 \\ -0.5 &\le u_2 \le 0.5 \\ -0.8 &\le y_1 \le 0.2 \\ -0.8 &\le y_2 \le 0.2 \end{aligned} \tag{5.4}$$

Because the system is linear, we have arbitrarily set the initial operating point to be the origin $(u, y) = (0, 0)$. The LP costs are such that the end points are maximized:

$$c = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \qquad d = -\begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}. \tag{5.5}$$

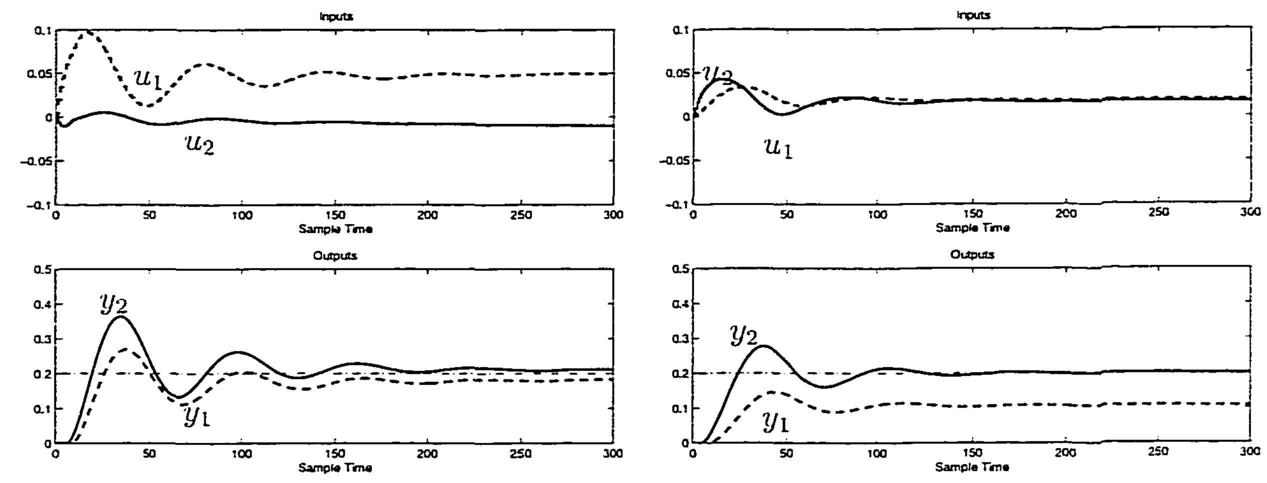

Figure 5.7: Closed loop response of the system with targets calculated via the nominal LP (left) and robust LP (right) for a perfect model and nonzero uncertainty. $\det G_m = \det G_{act} = 13.57$, $N = 60$, $c = 3$, $p = 70$, $w_u = 50$, $w_y = 1$.

In the simulations we consider the behavior of the closed loop system when the steady-state targets are calculated by the nominal LP, the robust LP, and the enumerated LP. Equation (5.3) is a box uncertainty description. The uncertainty for the robust LP is formed by inscribing an ellipse in the box defined by (5.3). The enumerated LP uses the box uncertainty directly and refers to the case in which the the output constraint is enumerated for all values of the gain. The targets are sent to the dynamic algorithm described in Appendix A. The dynamic algorithm implements input but no output constraints and is tuned to yield good performance for the nominal model. We only show plots for the nominal and robust LP, since the enumerated LP solution is similar to that for the robust LP.

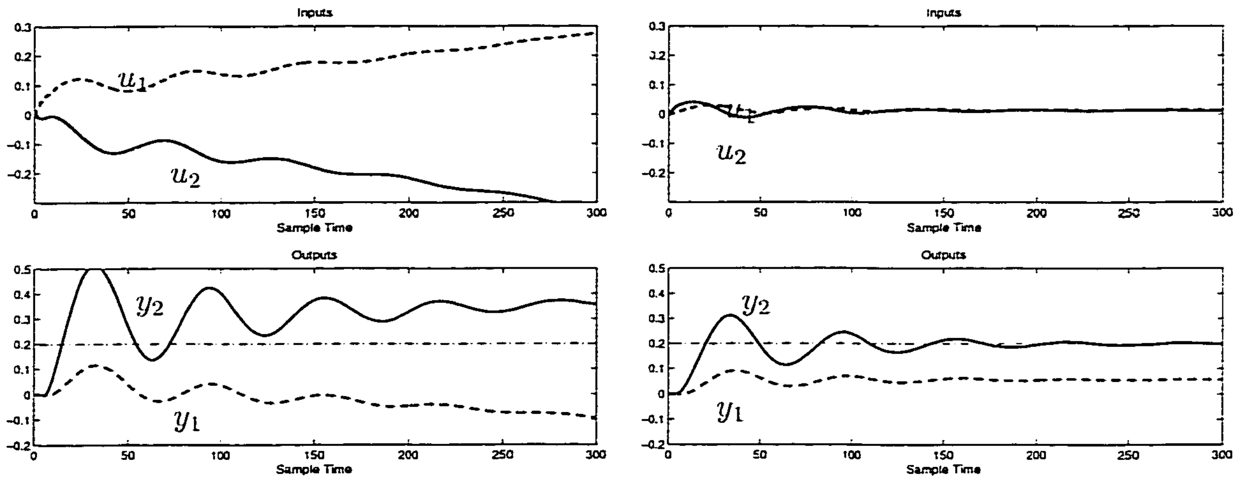

Figure 5.8: Closed loop response of the system with targets calculated via the nominal LP (left) and robust LP (right) in the presence of model uncertainty. $\det G_m = 13.57$, $\det G_{act} = -5.21$, $N = 60$, $c = 3$, $p = 70$, $w_u = 50$, $w_y = 1$.

Figure 5.7 shows the closed loop response when we assume the model given by (5.1) exactly describes the plant. The system starts at the origin, and the LP costs drive the system toward the upper output constraints. Both outputs in the nominal controller converge to their upper constraints. The outputs however, violate the constraint for a large amount of the time. Constraints are violated to a lesser degree when the robust controller is used. The robust controller pushes the side end point to its upper limit but finds a different optimal steady-state operating point for the top end point. The robust LP, unlike the nominal LP, does not always push the system to the intersection of constraints. In this case, the robust LP has pushed the system to the best operating point given the uncertainty in the problem.

Now consider the case when the plant and model differ. Assume the steady-state gain of the system is actually:

$$G_{act} = \begin{bmatrix} 2.05 & 2.12 \\ 8.39 & 6.12 \end{bmatrix}. \tag{5.6}$$

which lies within the uncertainty description (5.2). Figure 5.8 shows the closed loop response. The nominal controller goes unstable, but the robust controller is still stabilizing. This is in part due to the fact that the determinant of the model gain has the wrong sign.

$$\det G_{act} = -5.21, \qquad \det G_m = 13.57.$$

The plant moves in the opposite direction of the model prediction forcing the controller to go unstable. The robust algorithm successfully moves the top end point to its upper limit and again moves the side end point to a non-constrained steady-state. While the robust algorithm cannot guarantee closed loop stability when the determinant of the gain changes sign, this shows that it can be less sensitive to errors of this type.

Now assume that we were able to get a better estimate of the plant gain. In particular, assume that

$$G_m = \begin{bmatrix} 3.12 & 2.29 \\ 7.10 & 5.80 \end{bmatrix} \quad \text{with} \quad \Delta G = \begin{bmatrix} 1.50 & 0.30 \\ 1.90 & 0.40 \end{bmatrix}. \tag{5.7}$$

The actual gain (5.6) is still in our new uncertainty description. Using the same tuning for the dynamic controller, we obtain the simulation shown in Figure 5.9. The nominal controller is now stable but the side end point is minimized instead of maximized. If the simulation is carried out to long enough times, the side draw is forced to its lower bound with a net negative steady-state move in the side end point. The nominal controller, while predicting the end points should both be maximized, actually has moved the system in the wrong direction. This is because the model determinant still has wrong sign:

$$\det G_m = 1.80.$$

The discrepancy is not large enough to cause the controller to go unstable, but it is enough to move the system in the wrong direction. The robust controller is stable and maximizes both end points as the LP costs dictate.

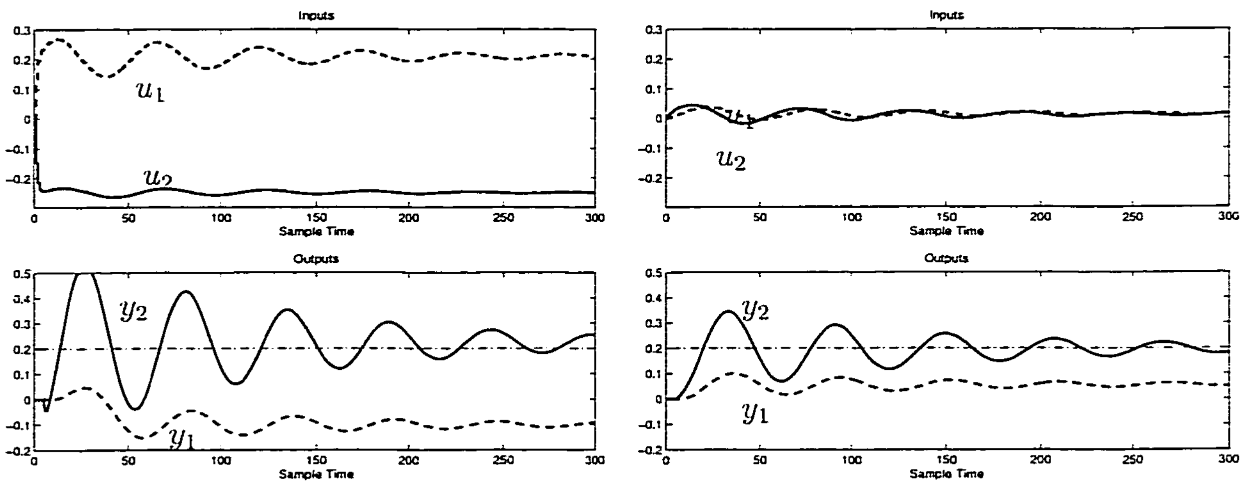

Figure 5.9: Closed loop response of the system with targets calculated via the nominal LP (left) and robust LP (right) in the presence of model uncertainty. $\det G_m = 1.80$, $\det G_{act} = -5.21$, $N = 60$, $c = 3$, $p = 70$, $w_u = 50$, $w_y = 1$.

Finally, assume that we can identify the actually plant perfectly, but $\Delta G \neq 0$. Assume

$$G_m = G_{act} \qquad \text{with} \qquad \Delta G = \begin{bmatrix} 0.80 & 0.25 \\ 1.10 & 0.30 \end{bmatrix}. \tag{5.8}$$

Figure 5.10 shows the resulting simulation. The nominal algorithm now correctly pushes the closed loop system in the correct direction. While the system does not reach the constraints in the time-frame of the plot, it eventually reaches both upper constraints. The robust controller continues to move the system to the best operating point given the uncertainty of the problem. If we were to decrease the uncertainty, the trajectories in the robust simulation would become closer and closer to those in the nominal simulation. In the limit of zero uncertainty $\Delta G \to 0$, the robust simulation is identical to the nominal simulation.

All these simulations were run using the enumerated LP as well. As mentioned previously, for this example, the control profile for the enumerated LP is very similar to the control profile for the robust LP. This is generally not the case and the two

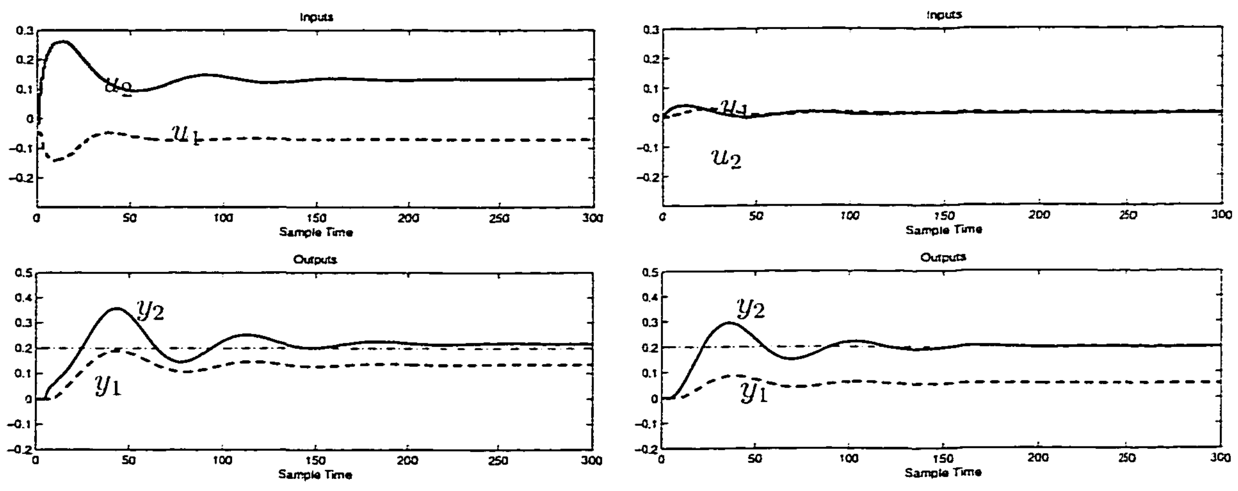

Figure 5.10: Closed loop response of the system with targets calculated via the nominal LP (left) and robust LP (right) for a perfect model and nonzero uncertainty. $G_m = G_{act}$, $\det G_m = -5.21$, $N = 60$, $c = 3$, $p = 70$, $w_u = 50$, $w_y = 1$.

controllers can produce different closed loop responses. Because the enumerated LP contains an increased number of constraints caused by enumerating all possible combinations of the gain, it is slower to solve. Table 1 shows the average solution time of all three algorithms for this example. The error shown in the number represents one standard deviation.

	Nominal LP	Robust LP	Enumerated LP
# of variables	6	6	66
# of constraints	12	12	132
Time (sec)	$(2.0 \pm 0.7) \times 10^{-2}$	$(4.4 \pm 0.7) \times 10^{-2}$	$(9.6 \pm 1.0) \times 10^{-1}$

Table 5.1: Comparison of Average Solution Times and Problem Size for the Nominal, Robust, and Enumerated LP for the Shell Fundamental Control Problem

The solution time for the robust LP is roughly twice the solution time of the nominal algorithm. This time can likely be improved by further exploiting the unique structure inherent in the robust LP. The particular SOCP routine used for this example takes no advantage of structure and requires strictly feasible initial primal and dual points. By exploiting structure and using a single stage method, it is likely that solution times can be reduced. The enumerated LP uses a simplex method. Because there are an increased number of constraints, it is slower by over an order of magnitude (even for this two-by-two example).

For these simulations the controller using targets calculated by the robust LP provided better control than the corresponding controller using targets calculated by the nominal LP. The robust algorithm was able to effectively handle model mismatch, even for the case when the determinant of the gain changed sign.

5.3 Nonlinear Distillation Column

Let us consider the performance of the robust LP on the high-purity, nonlinear, binary-component distillation column presented by Skogestad in [98] and [96]. The column, illustrated in Figure 5.11, is a 2-by-2 system in the LV configuration. The reflux L and boil-up V are used to control the mole fraction of light component in the distillate x_d and bottoms x_b. The distillate flow D and the bottoms flow B are used to control the liquid holdup in the condenser and reboiler through PID loops. They are part of the process model, and not included explicitly in the controller. The feed rate F is taken to be unity and assumed constant. All flows are expressed in molar units. The distillation column is described by a system of ordinary differential equations with 82 states, corresponding to the liquid holdup and composition (of the light component) on each of the columns' 41 stages. The model assumes: binary separation, constant pressure and negligible vapor holdup, total condenser, constant molar flows, equilibrium on all stages with constant relative volatility, and linearized liquid flow dynamics [96].

We have chosen to pose the problem in the LV configuration for illustrative purposes. Other configurations are possible and change the problem dynamics. Some of the difficulty and problems we will encounter in the control of the column can be

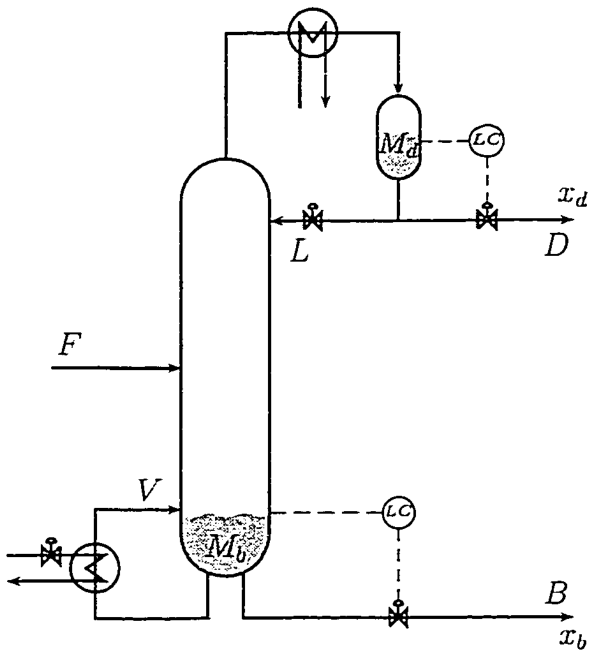

Figure 5.11: Typical Distillation Column Controlled in the LV-configuration

eliminated by choosing a different control configuration. The configuration used in a real-world process would depend upon the particular application and operating conditions. The paper by Skogestad et al. [97] discusses various control configurations, giving guidelines and rules of thumb for when to choose one over another.

For this example the outputs y and the inputs u are given by:

$$y = \begin{pmatrix} x_d \\ x_b \end{pmatrix}, \qquad u = \begin{pmatrix} L \\ V \end{pmatrix}.$$

At steady-state we have:

$$\Delta y = G\Delta u + b$$

where G is the system gain given by:

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}.$$

We can expect the gain to behave nonlinearly because of the underlying nonlinear model. In this example we would like to drive the separation to high purities,

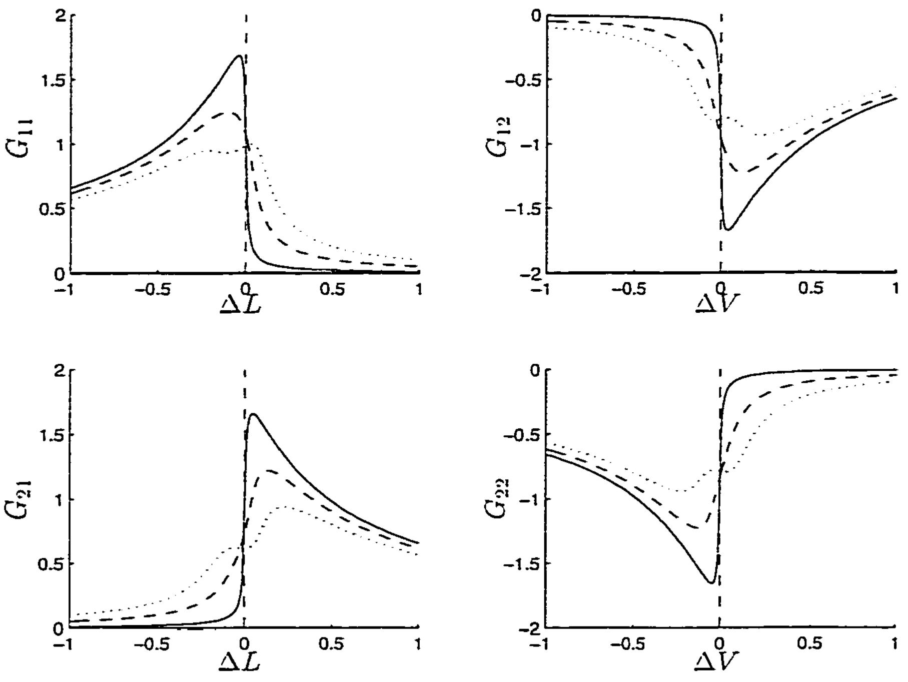

Figure 5.12: Nonlinear dependence of the steady-state gain on move size and operating point. Solid-line: high-purity. Dashed-line: mid-purity. Dotted-line: low-purity.

i.e. to the point where x_d is almost unity and x_b is almost zero. The column is strongly nonlinear for these conditions. It exhibits directional and operating point nonlinearities, illustrated in Figure 5.12. The figure shows each gain element's dependence upon move size for three different operating conditions: $y_{s1} = (0.90, 0.10)$, $y_{s2} = (0.95, 0.05)$, and $y_{s3} = (0.99, 0.01)$, with corresponding steady-state inputs: $u_{s1} = (1.5329, 2.0329)$, $u_{s2} = (1.8733, 2.3733)$, and $u_{s3} = (2.7063, 3.2063)$. These three operating conditions will be referred to as the low-, mid-, and high-purity steady-states, respectively. If the column were linear, the curves in Figure 5.12 would be horizontal lines with zero slope and an intercept equal to the steady-state gain. The directional dependence of the gain is shown in a lack of symmetry about $\Delta L = 0$ and $\Delta V = 0$. The operating point nonlinearity is shown in the fact that the curves do not all lie on top of one another. In fact, the curves do not all possess the same

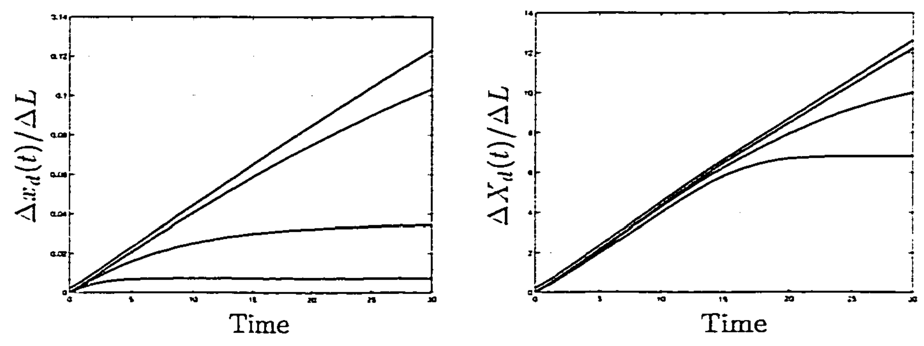

Figure 5.13: Scaled initial open loop nonlinear response about the high-purity steady-state for reflux move sizes of (from top to bottom) 0.1%, 1%, 10%, and 50% for standard (left plot) and logarithmic (right plot) distillate compositions.

intercept. The value of the gain for zero move size corresponds to the gain of the linearized system. This is a function of the operating point. The exact values of the gain at the three purities can be obtained by numerically differentiating the model. The mid- and high-purity gains are:

$$G_{mid} = \begin{pmatrix} 0.839 & -0.724 \\ 0.959 & -1.073 \end{pmatrix}, \qquad G_{high} = \begin{pmatrix} 0.875 & -0.862 \\ 1.085 & -1.098 \end{pmatrix}. \tag{5.9}$$

Finally, Figure 5.12 shows that the individual plots of the gain elements are almost mirror images of one another. This symmetry manifests itself as a highly ill-conditioned gain matrix. A quantitative measure of the ill-conditioning is the condition number r of the matrix. The condition numbers of the above two matrices are

$$r_{mid} = 15.97, \qquad r_{high} = 145.45.$$

This shows the system moves closer to singularity with increasing purity.

Let us consider the dynamic behavior of the column. If we plot the initial dynamic response of the distillate for various moves in the reflux we obtain the left-hand plot in Figure 5.13.

The initial dynamic response is a strong function of move size. We can observe that the time-constant of the system increases dramatically as we decrease the move size.

These nonlinearities cause problems for control systems. One way to reduce these problems is by introducing logarithmic compositions [96]. Let us define two logarithmic compositions:

$$X_d \overset{def}{=} \ln(x_d/(1-x_d)) \qquad \text{and} \qquad X_b \overset{def}{=} \ln(x_b/(1-x_b)). \tag{5.10}$$

A new output can then be defined as $Y_d = (X_d, X_b)$. The open loop response of the distillate logarithmic composition is shown in the right-hand plot in Figure 5.13. The initial response is independent of move size and is in fact also independent of operating point [96]. This is important for model predictive control applications in which a linear model is used. If we plot the gain of the system using logarithmic compositions as we did in Figure 5.12, we obtain Figure 5.14. The gain is now much less dependent upon move size, though it is still a function of operating point for very small moves. Also, observe that the gain is more symmetric about origin, indicating that the directional nonlinearity in the problem has been reduced. The value of the gain at the mid- and high-purity operating points determined by numerically differentiating the model is:

$$G_{mid} = \begin{pmatrix} 17.641 & -15.233 \\ 20.178 & -22.589 \end{pmatrix}, \qquad G_{high} = \begin{pmatrix} 88.378 & -86.999 \\ 109.564 & -110.942 \end{pmatrix}. \tag{5.11}$$

The condition numbers are the same as those for non-logarithmic compositions. We can expect the process to behave more linearly through the use of logarithmic compositions. Since we will be using a linear model to to approximate the column, we can expect better control.

We will employ logarithmic compositions in our simulations because they reduce the nonlinearity of the problem. It is the authors' experience that using standard compositions in the simulation demonstrates more the inability of the dynamic algorithm to handle the nonlinear system than the effect of steady-state uncertainty on the problem.

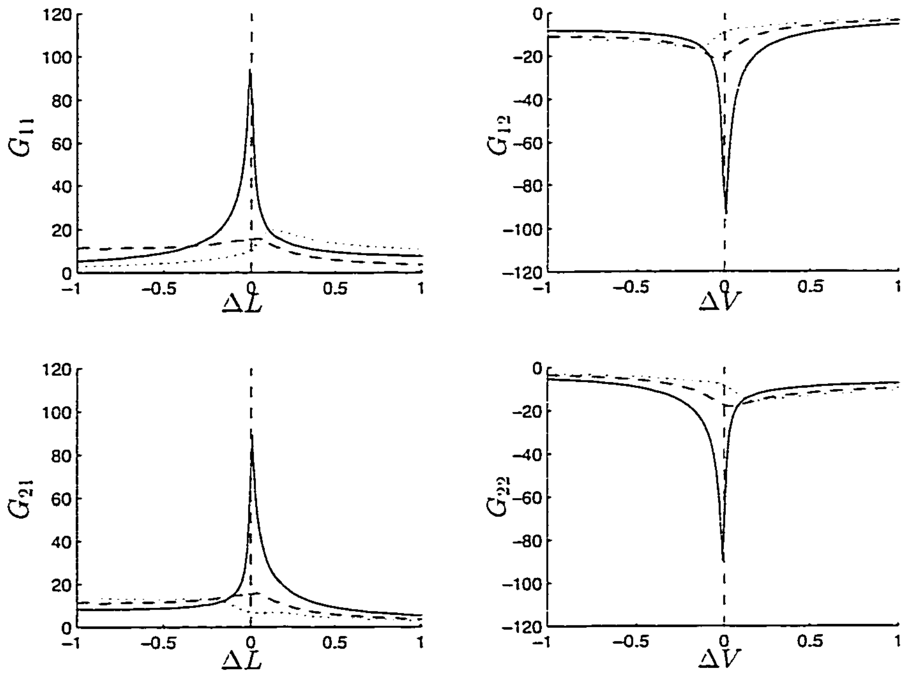

Figure 5.14: Dependence of the steady-state gain on move size and operating point for logarithmic compositions. Solid-line: high-purity. Dashed-line: mid-purity. Dotted-line: low-purity.

The simulations use a step-response model with a sample-time of 10 minutes. To obtain steady-state uncertainty information for use in the robust LP, we inject multiple steps into the nonlinear process allowing the system to reach steady-state after each step. The distribution of step sizes forms a random Gaussian signal. This provides a wealth of steady-state data which we identify using least-squares techniques to determine a the system gain and covariance. We obtain the dynamic step-response data by scaling a linearized model to the gain just determined. Ideally, we would determine the steady-state and dynamic information, including covariances as part of a single identification from step-response data.

The control problem has the following input and output constraints:

$$\begin{pmatrix} 1.5 \\ 2.0 \end{pmatrix} \le u \le \begin{pmatrix} 4.0 \\ 4.0 \end{pmatrix}, \tag{5.12}$$

$$\begin{pmatrix} 0.9 \\ 0.0 \end{pmatrix} \le y \le \begin{pmatrix} 1.0 \\ 0.1 \end{pmatrix},$$

where y is the original composition vector. The transformed compositions produce the following constraints: $2.3 \le X_d \le \infty$ and $-\infty \le X_b \le -2.3$. This, however, can pose problems in the linear program. The high-purity constraints become degenerate. Given the above constraints, we would want the LP to drive the top and bottom logarithmic compositions to $+\infty$ and $-\infty$, respectively. Because the inputs are bounded, these constraints are not 'seen' by the LP. To handle this, we replace the high-purity constraints as follows:

$$\begin{pmatrix} 0.9 \\ \epsilon \end{pmatrix} \le y \le \begin{pmatrix} 1.0-\epsilon \\ 0.1 \end{pmatrix}, \tag{5.13}$$

where ϵ is some estimate of the ultimate purity that the column can be expected to reach. In our case, $10^{-4} \le \epsilon \le 10^{-3}$. For the examples we take $\epsilon = 5.0 \times 10^{-4}$. This ensures the constraints do not become degenerate in the LP.

The LP objective data is given by:

$$c = -\begin{pmatrix} 1.0 \\ 50.0 \end{pmatrix}, \qquad d = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}, \tag{5.14}$$

which drives the inputs to large values (i.e. large reflux and boil-up). This in turn drives the column toward high purities.

Initially we identify the process about the mid-purity steady-state. This yields an estimate for the steady-state gain and covariance. For purposes of explanation, we will give an approximate value of ΔG instead of the actual covariance. The value of ΔG is determined from a confidence level of 95 percent. The actual algorithm,

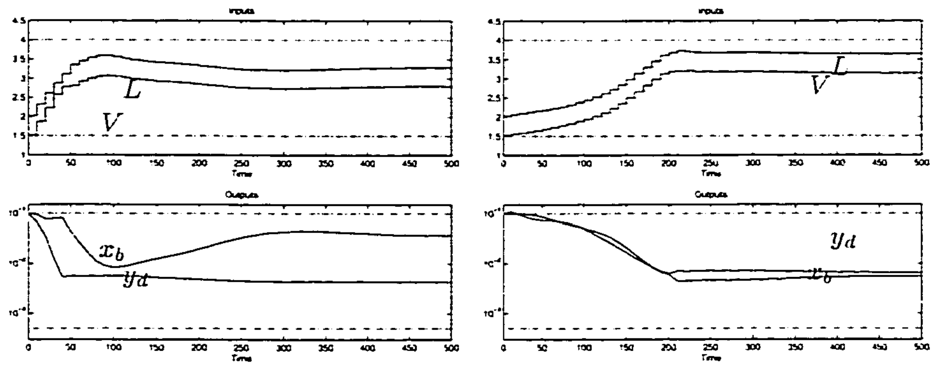

Figure 5.15: Closed loop response of the system with targets calculated via the nominal LP (left) and robust LP (right) in the presence of model uncertainty $||G - G_{est}||/||G|| = 0.24$. $N = 30$, $c = 8$, $p = 40$, $w_u = 20$, $w_y = 1$.

however, uses the covariance:

$$G_{est} = \begin{pmatrix} 14 & -17 \\ 14 & -20 \end{pmatrix}, \qquad \Delta G \sim \begin{pmatrix} 2.95 & 2.95 \\ 2.95 & 2.95 \end{pmatrix}. \tag{5.15}$$

A measure of the actual uncertainty in the problem is $||G - G_{est}||/||G||$ which quantifies the difference between the estimated and actual gain. It allows us to determine how accurately we identified the column. At mid-purity:

$$||G - G_{est}||/||G|| = 0.24.$$

This tells us the gain is accurate to roughly 25 percent.

The targets will be sent to the dynamic controller described in Appendix A. It has the following conventional tuning: $N = 30$, $c = 8$, $p = 40$, $w_u = 20$, $w_y = 1$. Here N is the number of coefficients in the step-response model; c is the control horizon; p is the prediction horizon; w_u is the input move suppression weight; and w_y is the output move suppression weight. This tuning ensures the dynamic controller is stable for the different conditions.

The dynamic controller uses the same input and output constraint set as the LP. The output constraints are cast as soft constraints and are not robust. While the targets sent to the dynamic controller are guaranteed not to violate the output constraints for the given uncertainty description at steady-state, the dynamic controller may move the inputs in such a way that the output constraints may not be met in the interim. A completely consistent approach would be to include a robust output constraint in the dynamic calculation as well. Another way to impart robustness to the dynamic controller is by increasing the move suppression w_u, which is the method we use.

We assume the system is brought to the low-purity operating point; at which time the controller is turned on. The controller should then drive the system to as high a purity as possible. The initial input and move history are assumed constant and initialized with the low-purity steady-state information. Figure 5.15 shows the result of the simulation. We have plotted the compositions on a logarithmic scale and have used $y_d = 1 - x_d$ to designate the overhead distillate composition.

The nominal controller (which uses the nominal LP) successfully drives the distillate composition to a high purity, but the bottom composition is driven to an incorrect operating point. The robust controller (which uses the robust LP) pushes the system to both high-purity limits without a problem. The plant at high purity is very singular and has a steady-state gain an order of magnitude greater than that of the identified model. The wrong constraint is approached by the nominal controller as a result of the targets sent by the steady-state controller. The feedback received from the dynamic controller combined with a poor value of the gain causes the nominal LP to produce targets that push the column in a direction where it believes it is approaching the high-purity constraint, when in fact it is approaching the lower constraint.

The robust controller, because it has information on the various values the gain may take, is more cautious and sends less aggressive targets to the dynamic algorithm. As a result the controller takes a little longer to reach its final operating point, but it can successfully move the system toward high purity. This is even though the actual system gain is outside its uncertainty description.

Let us identify the column again, this time obtaining a better estimate for the

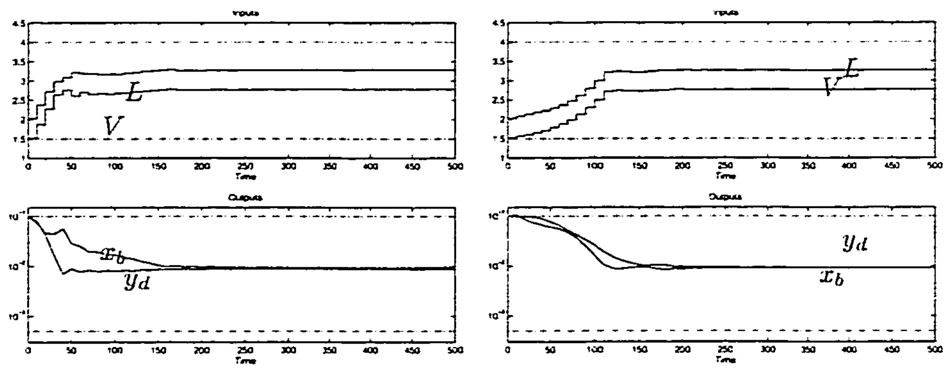

Figure 5.16: Closed loop response of the system with targets calculated via the nominal LP (left) and robust LP (right) in the presence of model uncertainty $||G - G_{est}||/||G|| = 0.0$. $N = 30$, $c = 8$, $p = 40$, $w_u = 20$, $w_y = 1$.

gain. In fact, assume we can identify the model in such a way that we obtain the exact value of the gain at the mid-purity operating point but the covariance is the same as in the previous simulation.

$$||G - G_{est}||/||G|| = 0.$$

The new simulation, shown in Figure 5.16, shows that both the nominal and robust controller can correctly push the system toward the high-purity limits. However, because of model mismatch and nonlinearity, neither controller can push the system to even higher purities. The robust controller takes slightly longer to reach the same steady-state limit as the nominal controller.

The linear program is pushing the system to a region of operating space where the model no longer represents the system. This is often the case in practice. Since the time and effort required to identify a model suitable for control of an industrial plant can be very costly, it is not done very often. In the interim between plant tests, the plant may be pushed in directions that increase model error. This can, as we have seen, result in poor control. Intelligent design of a control system can help to minimize this, but it cannot remove the problem.

Let us identify the system one more time. This time we will identify the system

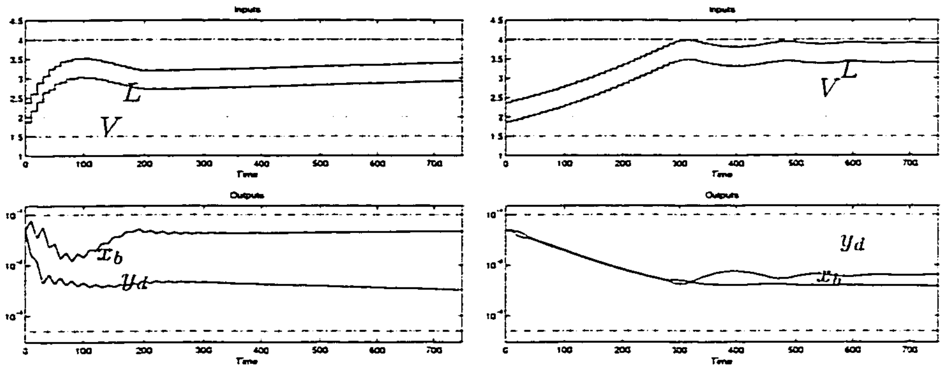

Figure 5.17: Closed loop response of the system with targets calculated via the nominal LP (left) and robust LP (right) in the presence of model uncertainty $||G - G_{est}||/||G|| = 0.0$, $(\det V)^{1/2} = 8.84$. $N = 30$, $c = 8$, $p = 40$, $w_u = 20$, $w_y = 1$.

at the high-purity steady-state. The simulations will use a sample time of 30 minutes and start the system off at the high-purity steady-state. At very high purities the time constants become much longer [96] thus requiring larger sample times. Again for simplicity, we will assume we can identify the model exactly at the high-purity steady-state, but with some nonzero covariance.

$$||G - Gest||/||G|| = 0 \quad \text{with} \quad \Delta G \sim \begin{pmatrix} 25.47 & 25.47 \\ 25.47 & 25.47 \end{pmatrix}. \tag{5.16}$$

This estimate for the gain, however, will still be inaccurate because the column is nonlinear and does not operate exactly at the high-purity limit. We expect that with the better estimate we will be able to move the column to a higher purity. Figure 5.17 shows the resulting simulation. The nominal algorithm displays the same behavior as before. While the distillate purity is pushed to a higher value, the bottom composition is poor. The robust algorithm successfully pushes both compositions to high purities.

Chapter 6

Conclusions and Open Questions

6.1 Conclusions

In this dissertation we have shown how some robust model predictive control problems can be interpreted more clearly as a class of semi-infinite programming problems. These problems arise from the addition of semi-infinite constraints to a nominal MPC algorithm. The constraints come from theoretical arguments which guarantee the robust stability of the closed loop system or from requiring existing constraints to hold for an infinite set of plants. In particular, we considered both the effect of using cost function bounds as the means to impart robustness to the dynamic MPC algorithm and the consequence of requiring the output constraints in the steady-state target calculation to hold for an infinite set of gains.

When investigating the theoretical properties of model predictive control, it is nearly universally assumed that the steady-state targets received by the dynamic controller do not change. In practice, however, the targets result from a steady-state optimization whose purpose is to reject disturbances and drive the system to some economic optimum. Mismatch between the model used in the target calculation and the true system can cause very poor control.

For the case of a steady-state target calculation based on a linear program, we have shown one way that model uncertainty can be incorporated. For a linear model with elliptic uncertainty on the model parameters, the result is a robust target calculation which takes the form of a second-order cone program (SOCP). This SOCP can greatly improve control by rigorously accounting for modeling error. The resulting SOCP structure can be exploited to develop efficient numerical solutions based on primal-dual interior-point methods.

For the case when model uncertainty is given instead by a polytope or box, the result is a robust steady-state target calculation which takes the form of an enumerated linear program (LP). The enumerated LP is generally more conservative than the corresponding SOCP and can be slower to solve. Primal-dual interior point methods can be used to reduce the solution time and provide an alternative, efficient solution strategy.

For the case of a dynamic MPC calculation based on a linear model with cost function bounds, we have shown that the nonlinear program is better interpreted as a semi-infinite program (SIP). The SIP can be solved using the method of local reduction with an SQP solver. The QP subproblem can be efficiently solved using primal-dual interior-point methods.

Finally, we showed several simulation examples where the robust LP, by incorporating the uncertainty in the steady-state gain, produces improved closed loop control. While the nominal LP is sensitive to gain uncertainty, the robust LP successfully moves the system to the best operating point for a given amount of uncertainty.

6.2 Open Questions

Many of the ideas presented in this dissertation can be extended directly to other problems in model predictive control. We showed how the robust LP associated with the steady-state target calculation can be interpreted as either a stochastic program or semi-infinite program. Both of these in turn are equivalent to the second-order cone program. The stochastic and semi-infinite programs approach the problem from different directions. The obvious question arises: which is the better approach when applied to MPC with nonlinear models? For linear systems they are identical, but when the underlying optimization problem is nonlinear, what is the difference in approaching the problem from these two viewpoints?

Consider the robust LP when the upper and lower bounds on an output collapse to a single value or a setpoint. This is equivalent to an equality constraint in the problem. What is the utility of the uncertainty description for a setpoint or equality constraint in the robust LP? Instead of setting the probability the constraint will hold, a better approach would be to maximize the likelihood that for a given input, the desired output (setpoint) will result. This is a different kind of problem than the one posed here, and it merits further study.

The theoretical properties of the closed loop system where robust steady-state targets are sent to a robust dynamic regulator have not been considered here. While the simulations indicate the system is stable under nominal tuning, it is not known under what conditions the robust controller is robustly stabilizing. Future work can consider the effect of combining robust target and robust dynamic calculations.

Finally, we conclude our comments on the solution methods presented in this dissertation. We proposed the use of primal-dual interior-point methods for the solution of the optimization problems. The simulations used a widely available code. If these ideas are to be used in an online environment, the specific type and structure of the interior-point method must be investigated and fine-tuned. Since speed is vitally important, the solution time must only grow modestly with increasing problem size.

Bibliography

[1] J. S. Albuquerque, V. Gopal, G. H. Staus, L. T. Biegler, and B. E. Ydstie, *Interior-Point SQP Strategies for Structured Process Optimization Problems*, Computers in Chemical Engineering **21** (1997), no. Suppl., S853–S859.

[2] F. Alizadeh and S. H. Schmieta, *Optimization with Semidefinite, Quadratic and Linear Constraints*, Tech. Report RRR 23–97, Rutgers Center for Operations Research, Rutgers University, November 1997.

[3] T. A. Badgwell, *A Robust Model Predictive Control Algorithm for Stable Nonlinear Plants*, International Symposium on Advanced Control of Chemical Processes (ADCHEM) (Banff, Canada), June 9–11 1997.

[4] ______, *Robust Model Predictive Control of Stable Linear Systems*, International Journal of Control **68** (1997), no. 4, 797–818.

[5] Y. Bard, *Nonlinear Parameter Esitmation*, Academic Press, Inc., New York, 1974.

[6] A. Ben-Tal and A. Nemirovski, *Robust Solutions of Uncertain Linear Programs via Convex Programming*, Tech. report, Technion Institute of Technology, Haifa, Israel, 1994.

[7] ______, *Robust Convex Optimization*, Tech. report, Technion Institute of Technology, Haifa, Israel, December 1996.

[8] B. W. Bequette, *Nonlinear Control of Chemical Processes: A Review*, Industrial and Engineering Chemistry Research **30** (1991), 1391–1413.

[9] L. T. Biegler, *Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation*, Computers & Chemical Engineering **8** (1984), no. 3/4, 243–247.

[10] S. Boyd, C. Crusius, and A. Hansson, *Control Applications of Nonlinear Convex Programming*, Journal of Process Control (1998), to appear.

[11] A. E. Bryson and Y. Ho, *Applied Optimal Control*, Hemisphere Publishing, New York, 1975.

[12] B. M. Budak, E. M. Berkovich, and E. N. Solov'e'va, *Difference Approximations in Optimal Control Problems*, SIAM Journal on Control **7** (1969), 18–31.

[13] R. M. Chamberlain, M. J. D. Powell, C. Lemarechal, and H. C. Pedersen, *The Watch-Dog Technique for Forcing Convergence in Algorithms for Constrained Optimization*, Mathematical Programming Study **16** (1982), 1–17.

[14] D. Chmielewski and V. Manousiouthakis, *On Constrained Infinite-Time Linear Quadratic Optimal Control*, Systems and Control Letters **29** (1996), 121–129.

[15] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complimentarity Problem*, Academic Press, San Diego, 1992.

[16] J. Cullum, *Discrete Approximations to Continious Optimal Control Problems*, SIAM Journal on Control **7** (1969), 32–49.

[17] J. E. Cuthrell and L. T. Beigler, *On the Optimization of Differential-Algebraic Process Systems*, AIChE Journal **33** (1987), no. 8, 1257–1270.

[18] ———, *Simultaneous Optimization and Solution Methods for Batch Reactor Control Profiles*, Computers & Chemical Engineering **13** (1989), no. 1/2, 49–62.

[19] C. Cutler, A. Morshedi, and J. Haydel, *An Industrial Perspective on Advanced Control*, AIChE National Meeting (Washington, D.C.), 1983.

[20] C. Cutler and B. L. Ramaker, *Dynamic Matrix Control–a Computer Control Algorithm*, AIChE National Meeting (Houston, TX), 1979.

[21] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, New Jersey, 1963.

[22] J. E. Dennis and J. J. Moré, *Quasi-Newton Method, Motivation and Theory*, SIAM Review **19** (1977), 46–89.

[23] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[24] A. S. El-Bakry, R. A. Tapia, T. Tsuchiya, and Y. Zhang, *On the Formulation and Theory of the Newton Interior-Point Method for Nonlinear Programming*, Journal of Optimization Theory and Applications **89** (1996), no. 3, 507–541.

[25] A. S. El-Bakry, R. A. Tapia, and Y. Zhang, *A Study of Indicators for Identifying Zero Variables in Interior-Point Methods*, SIAM Review **36** (1994), 45–72.

[26] J. Faraut and A. Korâyi, *Analysis on Symmetric Cones*, Oxford University Press, New York, 1994.

[27] L. Faybusovich, *Jordan Algebras, Symmetric Cones and Interior-Point Methods*, Manuscript, Department of Mathematics, Notre Dame, 1995.

[28] ______, *Euclidean Jordan Algebras and Interior-Point Algorithms*, Positivity I, Kluwer Academic Publishers, 1997, pp. 331–357.

[29] ______, *Linear Systems in Jordan Algebras and Primal-Dual Interior-Point Algorithms*, Journal of Computational and Applied Mathematics **86** (1997), 149–175.

[30] A. V. Fiacco and Y. Ishizuka, *Suggested Research Topics in Sensitivity and Stability Analysis for Semi-Infinite Programming*, Annals of Operation Research **27** (1990), 65–76.

[31] R. Fletcher, *Practical Methods of Optimization*, vol. Volume 2: Constrained Optimization, John Wiley & Sons, 1981.

[32] J. B. Froisy, *Model Predictive Control: Past, Present and Future*, ISA Transactions **33** (1994), 235–243.

[33] J. B. Froisy and T. Matsko, *IDCOM-M Application to the Shell Fundamental Control Problem*, AIChE Annual Meeting, 1990.

[34] W. Gander, *Least Squares with a Quadratic Constraint*, Numerische Mathematik **36** (1981), 291–307.

[35] C. E. García and A. M. Morshedi, *Quadratic Programming Solution of Dynamic Matrix Control (QDMC)*, Chemical Engineering Communications **21** (1986), 73–87.

[36] C. E. García, D. M. Prett, and A. M. Morari, *Model Predictive Control: Theory and Practice–a Survey*, Automatica **25** (1989), no. 3, 335–348.

[37] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, 1981.

[38] G. Gramlich, R. Hettich, and E.W. Sachs, *Local Convergence of SQP Methods in Semi-Infinite Programming*, SIAM Journal on Optimization **5** (1995), no. 3, 641–658.

[39] R. D. Grigorieff and R. M. Reemtsen, *Discrete Approximations of Minimization Problems, I. Theory, II*, Tech. Report 260 and 263, Technical University, Berlin, Germany, 1990.

[40] P. Grosdidier, B. Froisy, and M. Hammann, *The IDCOM-M controller*, Proceedings of the 1988 IFAC Workshop on Model Based Control (Oxford) (T. J. McAvoy, Y. Arkun, and E. Zafiriou, eds.), Pergamon Press, 1988, pp. 31–36.

[41] S. A. Gustafson and K. O. Kortanek, *Mathematical Programming: the State of the Art*, ch. Semi-Infinite Programming and Applications, pp. 132–157, Springer-Verlag, Berlin, 1983.

[42] R. Hettich, *Semi-Infinite Programming and Applications*, ch. A Review of Numerical Methods for Semi-Infinte Optimization, pp. 158–178, Springer-Verlag, 1983.

[43] R. Hettich and K. O. Kortanek, *Semi-Infinite Programming: Theory, Methods, and Applications*, Siam Review **35** (1993), no. 3, 380–429.

[44] R. E. Kalman, *Contributions to the Theory of Optimal Control*, Bull. Soc. Math. Mex. **5** (1960), 102–119.

[45] W. Karush, *Minima of Functions of Several Variables with Inequalities as Side Constraints*, Master's thesis, University of Chicago, Department of Mathematics, 1939.

[46] D. Kassmann and T. A. Badgwell, *Interior Point Methods in Robust Model Predictive Control*, AIChE Annual Meeting (Los Angeles, California), November 1997.

[47] ———, *The Robust LP: A Robust Steady-State Target Calculation for Model Predictive Control*, AIChE Annual Meeting (Miami Beach, Florida), November 1998.

[48] M. Kojima, S. Mizuno, and A. Yoshise, *A Primal-Dual Interior-Point method for Linear Programming*, Springer Verlag, New York, New York, 1989.

[49] H. W. Kuhn and A. W. Tucker, *Nonlinear Programming*, Proceedings of the Second Berkely Symposium on Mathematical Statistics and Probability (J. Neyman, ed.), University of California Press, 1951.

[50] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, John Wiley and Sons, New York, 1972.

[51] E. B. Lee and L. Markus, *Foundations of Optimal Control Theory*, John Wiley and Sons, 1967.

[52] J. H. Lee, *Recent Advances in Model Predictive Control and Other Related Areas*, Chemical Process Control—CPC V (Tahoe City, California), CACHE, January 1996.

[53] J. H. Lee and B. L. Cooley, *Stable Min-Max Control for State-Space Systems with Bounded Input Matrix*, submitted to IEEE Transactions in Automatic Control (1997).

[54] J. H. Lee and Zhenghong Yu, *Worst-case Formulations of Model Predictive Control for Systems with Bounded Parameters*, Automatica **33** (1997), no. 5, 763–781.

[55] M. S. Lobo, L. Vandenberghe, and S. Boyd, *SOCP: Software for Second-Order Cone Programming*, April 1997, Beta Version.

[56] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, *Applications of Second-Order Cone Programming*, Linear Algebra Appl. (1998), to appear.

[57] D. Q. Mayne, *Nonlinear Model Predictive Control: An Assessment*, Chemical Process Control—CPC V (Tahoe City, California), CACHE, January 1996.

[58] E. S. Meadows, M. A. Henson, J. W. Eaton, and J. B. Rawlings, *Receding Horizon Control and Discontinuous State Feedback Stabilization*, International Journal of Control **63** (1995), no. 5, 1217–1229.

[59] N. Megiddo, *Pathways to the Optimal Set in Linear Programming*, Progress in Mathematical Programming, Interior-Point and Related Methods (New York, New York) (N. Megiddo, ed.), Springer Verlag, 1989.

[60] S. Mehrotra, *On the Implementation of a Primal-Dual Interior-Point Method*, SIAM Journal on Control and Optimization **3** (1992), 575–601.

[61] S. Mizuno, M. J. Todd, and Y. Ye, *On Adaptive-Step Primal-Dual Interior-Point Algorithms for Linear Programming*, Mathematics of Operations Research **8** (1993), 964–981.

[62] R. D.C. Monteiro and T. Tsuchiya, *Polynomial Convergence of Primal-Dual Algorithms for the Second-Order Cone Program Based on the MZ-Family of Directions*, Tech. Report Memorandum No 617, The Institute of Statistical Mathematics, Tokyo, Japan, 1998.

[63] M. Morari and J. H. Lee, *Model Predictive Control: The Good, The Bad, And The Ugly*, Chemical Process Control–CPC IV (Elsevier, Amsterdam) (Y. Arkun and W.H. Ray, eds.), 1991, Fourth International Conference on Chemical Process Control.

[64] K. R. Muske and J. B. Rawlings, *Model Predictive Control with Linear Models*, AIChE Journal **39** (1993), no. 2, 262–287.

[65] Yu. Nesterov and A. Nemirovsky, *Interior-Point Polynomial Methods in Convex Programming*, Studies in Applied Mathematics, vol. 13, SIAM, Philadelphia, PA, 1994.

[66] Yu. E. Nesterov and M. J. Todd, *Self-Scaled Barriers and Interior-Point Methods for Convex Programming*, Math. Oper. Res. **22** (1997), 1–42.

[67] ———, *Primal-Dual Interior-Point Methods for Self-Scaled Cones*, SIAM Journal on Optimization **8** (1998), 324–364.

[68] C.P. Neuman and A. Sen, *A Suboptimal Control Algorithm for Constrained Problems Using Cubic Splines*, Automatica **9** (1973), 601–613.

[69] E. Polak, *Computational Methods in Optimization. A Unified Approach*, Academic Press, New York, 1971.

[70] ———, *On the Mathematical Foundations of Nondifferentiable Optimization in Engineering Design*, SIAM Review **29** (1987), no. 1, 21–89.

[71] ———, *On the Use of Consistent Approximations in the Solution of Semi-Infinite Optimization and Optimal Control Problems*, Mathematical Programming **62** (1993), 385–415.

[72] ———, *Optimization: Algorithms and Consistent Approximations*, Springer-Verlag, New York, 1997.

[73] E. Polak and A. L. Tits, *A Recursive Quadratic Programming Algorithm for Semi-Infinite Optimization Problems*, Journal of Applied Mathematics and Optimization 8 (1982), 325–349.

[74] A. Prékopa, *Stochastic Programming*, Kluwer Academic Publishers, Boston, 1995, Chapters 10 & 11.

[75] D. M. Prett and M. Morari, *The Shell Process Control Workshop*, Butterworths, Stoneham, MA, 1987.

[76] D.M. Prett and R.D. Gillette, *Optimization and Constrained Multivariable Control of a Cracking Unit*, Proceedings of the Joint Automatic Control Conference, 1980.

[77] S. J. Qin and T. A. Badgwell, *An Overview of Industrial Model Predictive Control Technology*, Fifth International Conference on Chemical Process control (Jeffry C. Kantor, Carlos E. García, and Brice Carnahan, eds.), AIChE Symposium Series 316, no. 93, 1997, pp. 232–256.

[78] ———, *An Overview of Nonlinear Model Predictive Control Applications*, Nonlinear Model Predictive Control Workshop - Assessment and Future Directions (Ascona, Switzerland), June 3-5 1998.

[79] S. Ralhan and T. A. Badgwell, *Robust Model Predictive Control of Linear Finite Impulse Response Plants*, Master's thesis, Rice Unversity, Houston, TX, 1998.

[80] C. V. Rao, S. J. Wright, and J. B. Rawlings, *Efficient Interior-Point Methods for Model Predictive Control*, Preprint ANL/MCS-P644-0597, Mathematics and Computer Science Division, Argonne National Laboratory, 1997.

[81] J. B. Rawlings and E. S. Meadows, *Model Predictive Control*, 1997, Michael A. Henson and Dale E. Seborg editors.

[82] J. B. Rawlings, E. S. Meadows, and K. R. Muske, *Nonlinear Model Predictive Control: A tutorial and survey*, ADCHEM 94 Proceedings (Kyoto, Japan), CACHE, 1994.

[83] G. W. Reddien, *Collocation at Gauss Points as a Discretization in Optimal Control*, SIAM Journal on Control and Optimization **17** (1979), no. 2, 298–306.

[84] R. Reemtsen, *Discretization Methods for the Solutions of Semi-Infinite Programming Problems*, Journal of Optimization Theory and Applications (1991), no. 71, 85–104.

[85] R. Reemtsen and Jan-J. Rückmann (eds.), *Semi-Infinite Programming*, Kluwer Academic Publishers, Boston, 1998.

[86] J. Richalet, A. Rault, J. L. Testud, and J. Papon, *Model Heuristic Control: Applications to Industrial Processes*, Automatica **14** (1978), 413–428.

[87] J. Richalet, A. Rault, J. L. Tesud, and J. Papon, *Algorithmic Control of Industrial Processes*, Proceedings of the 4th IFAC Symposium on Identification and System Parameter Estimation, 1976, pp. 1119–1167.

[88] N. Lawrence Ricker, *Model Predictive Control: State Of The Art*, Chemical Process Control - IV (New York, N.Y.), AIChE, 1991.

[89] L. O. Santos, Nuno M.C. de Oliveira, and L. T. Biegler, *Reliable and Efficient Optimization Strategies for Nonlinear Model Predictive Control*, IFAC Dynam-

ics and Control of Chemical Reactors (Copenhagen, Denmark), DYCORD+'95, 1995, pp. 33–38.

[90] S. H. Schmieta and F. Alizadeh, *Associative Algebras, Symmetric Cones and Polynomial Time Interior-Point Algorithms*, Tech. Report RRR 17–98, Rutgers Center for Operations Research, Rutgurs University, June 1998.

[91] A. T. Schwarm and M. Nikolaou, *Robust Output Constraint Satisfaction in MPC: Optimizing Under Uncertainty*, AIChE Fall National Meeting (Miami Beach, Florida), November 1997.

[92] ______, *Chance Constrained Model Predictive Control*, AIChE Fall Annual Meeting (Los Angeles, California), November 1998.

[93] A. Schwartz and E. Polak, *Consitent Approximations for Optimal Control Problems Based on Runga-Kutta Integration*, SIAM Journal on Control and Optimization **34** (1996), no. 4, 1235–1269.

[94] P. O. M. Scokaert and D. Q. Mayne, *Min-Max Feedback Model Predictive Control for Constrained Lienar Systems*, IEEE Transactions on Automatic Control **43** (1998), no. 8, 1136–1142.

[95] P. O.M. Scokaert and J. B. Rawlings, *Constrained Linear Quadratic Regulation*, Accepted for publication in IEEE Trans. Auto. Cont. (1997).

[96] S. Skogestad, *Dynamics and Control of Distillation Columns: A tutorial introduction*, Trans. IChemE **75** (1997), no. Part A.

[97] S. Skogestad, P. Lundstr om, and E.W. Jacobsen, *Selecting the Best Distillation Control Structure*, AIChE Journal **36** (1990), no. 5, 753–764.

[98] S. Skogestad, M. Morari, and J. Doyle, *Robust control of ill-conditioned plants: High purity distillation*, IEEE Transactions on Automatic Control **33** (1988), no. 12, 1092–1105.

[99] Y. Tanaka, M. Fukushima, and T. Ibaraki, *A Globally Convergent SQP Method for Semi-Infinite Nonlinear Optimization*, Journal of Computational and Applied Mathematics **23** (1988), 141–153.

[100] R. A. Tapia, *On the Role of Slack Variables in Quasi-Newton Methods of Constrained Optimization*, Numerical Optimization of Dynamic Systems (North Holland, Amsterdam, Holland) (Edited by L. C. Dixon and G. P. Szego, eds.), 1980.

[101] R. A. Tapia and M. Trosset, *An Extension of the Karush-Kuhn-Tucker Necessity Conditions to Infinite Programming*, SIAM Review **36** (1994), 1–17.

[102] R. Tichatschke, R. Hettich, and G. Still, *Connections Between Generalized, Inexact and Semi-Infinite Linear Programming*, Z. Für Operations Res. **33** (1989), 367–382.

[103] A. L. Tits, *Langrangian Based Superlinearly convergent Algorithms for Ordinary and Semi-Infinite Optimization Problems*, Ph.D. thesis, University of California, Berkeley, CA, 1980.

[104] L. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review **38** (1996), no. 1, 49–95.

[105] R. J. Vanderbei and H. Yuritan, *Using LOQO to Solve Second-Order Cone Programming Problems*, Tech. Report SOR-98-9, Statistics and Operations Research, Princeton University, 1998.

[106] L. N. Vicente, *Trust-Region Interior-Point Algorithms for a Class of Nonlinear Programming Problems*, Ph.D. thesis, Rice University, Houston, TX, 1996.

[107] J. V. Villadsen and W. E. Stewart, *Solution of boundary-value problems by orthogonal collocation*, Chemical Engineering Science **22** (1967), 1483–1501.

[108] J. Vlassenbroeck and R.V. Dooren, *A Chebyschev Technique for Solving Nonlinear Optimal Control Problems*, IEEE Transactions on Automatic Control **33** (1988), 333–340.

[109] P. Vuthandam, H. Genceli, and M. Nikolaou, *Performance Bounds for Robust Quadratic Dynamic Matrix Control with End Condition*, AIChE Journal **41** (1995), no. 9, 2083–2097.

[110] P. Whittle, *Optimization Under Constraints*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons Ltd., New York, 1971.

[111] M. H. Wright, *The Interior-Point Revolution in Constrained Optimizaiton*, Tech. Report 98-04-09, Computing Science Research Center, Bell Laboratories, Murry Hill, New Jersey, June 1998.

[112] S. J. Wright, *Applying New Optimization Algorithms to Model Predictive Control*, Fifth International Conference on Chemical Process control (Jeffry C. Kantor, Carlos E. García, and Brice Carnahan, eds.), AIChE Symposium Series 316, no. 93, 1997, pp. 147–155.

[113] ______, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, PA, 1997.

[114] S.P. Wu, Stephen Boyd, and L. Vandenberghe, *FIR Filter Design via Semidefinite Programming and Spectral Factorization*, Proc. of IEEE Conf. on Decision and Control, 1996.

[115] Y. Ye, *An $O(n^3L)$ Potential-Reduction Algorithm for Linear Programming*, Mathematical Programming **50** (1991), 239–258.

[116] Z. Q. Zheng, *Robust Control of Systems Subject to Constraints*, Ph.D. thesis, California Institute of Technology, 1995.

Appendix A

Nominal Dynamic MPC Algorithm for the Examples

In this appendix we describe the nominal MPC algorithm used in the examples. The algorithm is a variation of the quadratic dynamic matrix control (QDMC) [35] algorithm with an extra 'sum of moves' constraint and soft output constraints. Let n be the number of inputs, m be the number of outputs, c be the control horizon, p be the prediction horizon, and N be the length of the step response model. The quadratic program associated with QDMC is:

$$\begin{array}{rcl} \min\limits_{\delta u, \epsilon} & \delta u^T H \, \delta u - g^T \delta u & \\ \text{subject to} & & \\ \underline{u} - u_{k-1} \leq & L \, \delta u & \leq \overline{u} - u_{k-1} \\ \underline{y} - y^p \leq & A \, \delta u & \leq \overline{y} - y^p, \end{array} \tag{A.1}$$

where $\delta u \in \mathbb{R}^{nc}$ is the future input moves. The Hessian H and gradient g are given by:

$$H = A_f^T \Gamma^T \Gamma A_d + \Lambda^T \Lambda \quad \text{and} \quad g = A_d^T \Lambda^T \Lambda \, \hat{e}.$$

A_d is the *dynamic matrix*; $\Lambda = w_y I$ is the matrix of output error weights; and $\Gamma = w_u I$ is the matrix of input movement weights or move suppression weights. $\overline{u}$ and $\underline{u}$ are the upper and lower input bounds, respectively. Likewise, $\overline{y}$ and $\underline{y}$ are the upper and lower bounds on the outputs. The vector y^p is the future output prediction and u_{k-1} is the input at the previous sample time.

The dynamic matrix is given by

$$A_d = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix} \tag{A.2}$$

with

$$A_{ij} = \begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & 0 & \cdots & 0 \\ a_3 & a_3 & a_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_c & a_{c-1} & a_{c-2} & \cdots & a_1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_N & a_{N-1} & a_{N-2} & \cdots & a_{N-c+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_N & a_N & a_N & \cdots & a_{p-c+1} \end{bmatrix}_{ij} \tag{A.3}$$

and $(a_k)_{ij}$ is the kth step response coefficient relating the ith output to the jth input. The vector $\hat{e}$ is the error vector and is given by

$$\hat{e} = r - B_d \Delta u_p - C_d u_0$$

Here r is a vector representing the reference trajectory or setpoint of the system and B_d and C_d are matrices that relate the future output to the past known moves $\Delta u_p \in \mathbb{R}^{Nn}$ and past known inputs $u_0 \in \mathbb{R}^n$:

$$B_d = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1n} \\ B_{21} & B_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mn} \end{bmatrix} \tag{A.4}$$

with

$$B_{ij} = \begin{bmatrix} a_2 & a_3 & a_4 & \cdots & a_N \\ a_3 & a_4 & a_5 & \cdots & 0 \\ a_4 & a_5 & a_6 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{c+1} & a_c & a_{c-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{N-1} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{ij} . \tag{A.5}$$

Also

$$C_d = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ C_{m1} & C_{m2} & \cdots & C_{mn} \end{bmatrix} \tag{A.6}$$

with

$$C_{ij} = \begin{bmatrix} 0 & & & & a_N \\ & & & \cdots & \\ & & a_N & & \\ & a_N & & & \\ a_N & & & & \\ \vdots & & & & \\ a_N & & & & 0 \end{bmatrix}_{ij} . \tag{A.7}$$

In our formulation, we use soft output constraints and append a constraint that forces the sum of moves calculated by the algorithm to be equal to the total move target calculated by the steady-state algorithm Δu^s:

$$W\delta u = \Delta u^s$$

where W is a matrix that sums the individual moves.

$$W = \begin{bmatrix} E & & & \\ & E & & \\ & & \ddots & \\ & & & E \end{bmatrix}$$

with $E = [1\ 1\ \cdots\ 1]$. This, however, can cause problems. The addition of this end-point constraint causes the QP to become very ill-conditioned. It is analogous to the problems caused by end-point constraints in state-space MPC algorithms. Instead, we satisfy the constraint in a least squares sense and append it to the objective. The modified QP we solve is then:

$$\begin{aligned} &\min_{\delta u, \epsilon} \quad \delta u^T \hat{H}\, \delta u + \epsilon^T \Sigma\, \epsilon - \hat{g}^T \delta u \\ &\text{subject to} \\ &\quad \underline{u} - u_{k-1} \leq \quad L\, \delta u \quad \leq \overline{u} - u_{k-1} \\ &\quad \underline{y} - y^p - \underline{\epsilon} \leq \quad A_d\, \delta u \quad \leq \overline{y} - y^p + \overline{\epsilon} \end{aligned} \tag{A.8}$$

where $\epsilon = (\underline{\epsilon}, \overline{\epsilon})$ and the new Hessian $\hat{H}$ and gradient $\hat{g}$ are given by:

$$\hat{H} = A_d^T \Gamma^T \Gamma A + \Lambda^T \Lambda + W^T \Omega W \quad \text{and} \quad \hat{g} = A_d^T \Lambda^T \Lambda\, \hat{e} - W^T \Omega \Delta u^s.$$

IMAGE EVALUATION TEST TARGET (QA-3)