A Computational Approach to Nonlinear System Analysis

Thesis by

Jorge E. Tierno

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California
1996
(Submitted November 20, 1995)

To *Pepe*
Who told me about science,
And to *Mi Abuela Amelia*
Who showed me the way.

I miss you.

Acknowledgements

My dear reader, let me start with a warning and a plea. This section will be longer than the amount of work it precedes would grant. But although my thesis is short, the list of those that carried me through the years it took to write it is long. With your permission I will mention here a few of them, and I'll try to give you a flavor of what they did for me. I plea you to be patient and, by reading through these pages, to help me thank them.

In first place I have to thank my advisor, John Doyle. I learned many things from him, not the least of which were to eat while I eat, to sleep while I sleep, and to remember that the boat is always empty. Although our disagreements were many, from the use of outlines and archetypes to the meaning of sports and poetry, John shared with me the view of our field from his vantage point. I could not have asked for a more precious gift.

Much of the work in this thesis was guided by Richard Murray. Without his constant support, his unequaled ability to organize his ideas and mine, and his generosity, this thesis would have never been written. I hope to some day do for others what Richard did for me, and thus repay my debt. Thank you Richard, from the bottom of my latin heart.

I had good help in carrying out the work in this thesis. Pete Young, my first co-author, set the example I have ever since tried to follow. Irene Gregory added the aerospace flavor to my work, and taught me how to properly enjoy a baseball game. Adam Schwartz helped me carry out the comparison of my algorithm with other optimization methods.

What makes Caltech so special, is the highly talented and extremely interesting men and women that inhabit its labs and offices. Matt Newlin is the perfect example. Many hours I spent discussing with him the most diverse issues. I learned from Matt more than from any other person at Caltech, although we never agreed on whether can't and shouldn't are or not the same word, nor did we ever figure out why she brought that book, I did not want to be read to out of up for. My endless discussions over matters of taste with Bobby Bodenheimer made my bad days bearable and my good days great, and on the side he managed to teach me a little about UNIX and a lot about Greek mythology. A special word has to go to my officemates Mike Kantner, and Yun Huang who endured with me the ups and downs of a grad student's life. I know dear reader what you are thinking at this point. You probably know of the hours Sonja Glavaski dedicated to listening to my ramblings after a particularly bad day. You surely know of the many words of encouragement she gave me, of her always good advice, and of the endless supply of pens she kept on her desk for me to use. And knowing this you are probably mistaking my not mentioning her here for ingratitude. You should then also know that she explicitly asked not to be in this list. I will respect her wish and don't refer to her at all on this page.

Caltech also put me in contact with my friends at Club Latino and CLASES. They are a sampler of all the good our continent has to offer: Mario and Pili, true friendship from *la provincia*; Karina, the warmth and grace of the Caribbean; Ezzy and Flora, Mexican pride and Southern California debonair. Thank you all for the good times and the salsa steps, and for making me proud of my heritage.

Bear with me reader for two more paragraphs, because they are the most important ones. I do not believe that enough has ever been said of the value of a latin family, and I count with one of them as my biggest blessing. To my parents and siblings (Hi Mom, Hi Dad, Hi Rosana and José), goes here a big word of thanks for the last 30 years. As always, I'll count on you to carry me through the challenges the next 30 years are certain to bring along. And to my Southern California family, thanks for building a little piece of Uruguay in Los Angeles, and sharing it with me.

Finally, to the Sierras, the Pacific Ocean, and the view from the jetty in Balboa Point; to Whitney Houston's voice, and Gloria Estefan's rhythm; to the 405 South; to the outline of Catalina Island at sunset, thank you for being there when I needed you. I will always carry you with me, right next to the Southern Cross.

Abstract

Most practical control systems have significant nonlinear components. However, these systems are typically analyzed either through robustness analysis of their linearizations, or through extensive simulation of their nonlinear models. Other forms of analysis of nonlinear systems have not as yet led to computationally tractable solutions. The aim of this thesis is to extend the analysis methodology for linear systems given by the structured singular value framework to nonlinear systems. We study the question: Given an uncertain nonlinear system, driven by a nominal command signal over a finite time horizon, and subject to bounded noise, norm bounded feedback components, and uncertain parameters, how far from the nominal trajectory will the actual trajectory be? In order to inherit the properties of the structured singular value, we will use the 2-norm as measure for noise signals and undermodeled feedback components. As is the case for robustness analysis of linear systems, we can only find efficient computation algorithms for upper and lower bounds to the answer to this question.

To compute the lower bound we develop a power algorithm similar to the one developed for the structured singular value. Since, as was the case for linear systems, the algorithm is not guaranteed to converge in general, its analysis has to be done empirically. We test this algorithm by applying it to simulations of real systems and show that it performs better than other available optimization methods. To develop an upper bound, we study a class of rational nonlinear systems. We show that for problems in this class, an uncertain, constrained linear system can be constructed that achieves the same performance level. Upper bounds on the performance of these systems can be computed by solving linear matrix inequalities. Finally, we study extensions that can be obtained to these analysis methods when the system is linear but time varying.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Theoretical and computational tools for analysis and synthesis of robust controllers for linear systems are well developed in a variety of instances. Controllers generated with these tools can provide guaranteed performance in the presence of structured uncertainty, and the worst case disturbances for a given controller can be determined.

For linear time invariant (LTI) systems with complex, structured uncertainty, analysis of robust performance can be reduced to searching for the solution to a set of algebraic equations which give bounds on the achievable performance. One is thus able to find computationally efficient solutions, such as the power algorithm for the μ lower bound, without doing an explicit parameter search involving repeated simulation. This works because the system is linear and the performance and uncertainty descriptions are chosen so as to give computationally attractive solutions, even for large problems.

Current research in linear systems theory is devoted to extending the existing theory to incorporate more realistic uncertainty structures (such as real parameters) and to solving linear time varying and time invariant problems.

Analysis of nonlinear systems on the other hand has stayed mainly at the theoretical level. Lower bounds are usually computed through extensive simulation or local optimization techniques. However, these methods require large amounts of computation; standard optimization techniques fail even for small problems, and a search over the parameter space exhibits exponential growth with the number of parameters. Several upper bounds for the performance of nonlinear systems have been developed in the last few years. Most of these results are generalizations of Lyapunov theorems and depend on our being able to find a Lyapunov function for the system. However, there are no general, computationally tractable ways to accomplish this.

The aim of this thesis is to extend the linear systems analysis methodology, given by the structured singular value framework, to nonlinear systems. In order to achieve this, it is important to understand the characteristics of linear systems that are essential to the development of their analysis methods. Obviously, the relative simplicity of linear behavior, and of the underlying mathematics needed for their study, is a fundamental contributor.

There is, however, another characteristic of the problem without which practical, computer oriented solutions never would have arisen. Linear systems can be naturally described by finite amounts of data, and therefore, properties of the system can be expressed as functions on a finite dimensional space. If we want to replicate that success for nonlinear systems, we have to restrict our search to classes of nonlinear systems that can be described using a parameter space of finite dimension. For linear finite dimensional time invariant systems, the state space representation meets this criterion. The behavior of the systems can be deduced from the state space matrices. More importantly, the additional information needed to interpret these matrices is also finite.

What constitutes a valid finite description depends on whether we are analyzing the problem locally or globally. In what follows, we will assume that the performance of the system can be measured by one scalar index J. The system meets a given performance specification γ if $J \leq \gamma$.

When computing lower bounds on the achieved performance or necessary conditions for performance, we are concerned with local information. A lower bound on the performance index is established by exhibiting a set of signals that achieve it. Thus in order to develop computable lower bounds, the representation of the system has to be locally finite. In other words the local behavior of the system has to be computable in finite time. This is true in general for systems studied over a finite time horizon, since the behavior can be obtained by integrating the system's differential equations. (In linear time invariant systems, the whole infinite horizon behavior can be obtained in finite time working in the frequency domain. It is important to note that in this case infinite time horizon is used because it is convenient and not because it is the most justified approximation from an engineering point of view.)

Obtaining sufficient conditions for performance, or upper bounds on the performance index, implies establishing global properties of the system. The description of the system has to be such that these global conclusions can be derived in finite time. This cannot be done if all the information we have about the system is local: for a general system a finite number of local evaluations cannot predict global behavior. We have to restrict the set of allowable systems to a class for which global predictions can be made from finitely many local evaluations. (Linear systems for example form such a class.) In this case the finite description will be a set of coordinates identifying the particular instance within the allowable class that we are studying.

Starting from these considerations, we develop algorithms to determine upper and lower bounds on performance for nonlinear systems. The performance problem and class of systems considered have a solid engineering motivation and also meet the above conditions, which we believe are necessary to develop efficient computation algorithms. The next section outlines the remainder of the work.

1.1 Previous Work

The body of work done on robustness analysis of linear and nonlinear system is extensive and this section does not pretend to be a comprehensive review of it. The intention of this section is to place the work developed in this thesis in its context. The need to take into account uncertainty in modeling has been recognized from the early days of the development of a systematic control theory. In his seminal work on feedback amplifier design Bode introduced the concept of phase and gain margin, and with that the principal idea behind robustness analysis, namely that a condition of stability or performance based on a nominal model is not enough for practical purposes. In Bode's own words [3]:

> A theoretical characteristic that just met this requirement, however, would be unsatisfactory, since it is inevitable that the limiting phase would be exceeded in fact by minor deviations introduced either in the detailed design of the amplifier or in its construction.

The history of robust control of linear systems from this point on can be understood as an effort to systematize this statement. The small gain theorem introduced by Zames [37] in 1965 was the first significant milestone in this quest, and provided an exact robust stability test with respect to unstructured dynamic uncertainty. The flexibility and applicability of this tests increased significantly with the introduction of structured or block diagonal uncertainty (see, for example, [25]). A further step towards systematization of robustness analysis was taken in 1982 with the introduction of the structured singular value μ [7]. In [7] the need for computationally efficient upper and lower bound to the robust stability index was also discussed. This need was later confirmed when many of the robustness analysis problems of relevance to practical problems were proved to be NP-hard [4]. Most of the recent work has been dedicated to improving the computational methods for these bounds [20], and by extending their applicability to larger classes of uncertainty [34]. A large effort has been devoted also to understanding the theoretical nature of the upper and lower bounds and to establishing for which classes of uncertainty each of these conditions is necessary and sufficient [24].

Robustness analysis of nonlinear systems was developed as a generalization of the results obtained for linear systems. Robustness for nonlinear systems was proved to be equivalent to the existence of solution to Hamilton-Jacobi equations [33] or non-linear matrix inequalities [14]. However computational methods to establish the existence of these solutions have not been developed to a level comparable to their linear counterpart (i. e., existence of solutions of Riccati equations and linear matrix inequalities). These results still remain mainly of academic relevance, although research is actively being carried out in order to make them applicable.

The state of the art for nonlinear system analysis, as applied in industry today, still relies in repetitive simulation of the system under diverse conditions, and for a large number of randomly generated disturbances.

The work developed in this thesis significantly contributes to closing this gap existing between the theory and practice of robustness analysis for nonlinear systems. By studying performance over a finite time horizon, we can establish computationally tractable algorithms for upper and lower bounds on the robust performance index. Although the results presented here are not as sophisticated, from a mathematical point of view, as those in [33, 14], they are far more applicable to problems of engineering significance.

1.2 Contributions and Outline of This Work

The work developed in this thesis significantly contributes to closing this gap existing between the theory and practice of robustness analysis for nonlinear systems. By studying performance over a finite time horizon, we can establish computationally tractable algorithms for upper and lower bounds on the robust performance index. Although the results presented here are not as sophisticated, from a mathematical point of view, as those in [33, 14], they are far more applicable to problems of engineering significance.

We will organize these results as follows. In Chapter 2 we briefly review the main ideas in robustness analysis. We place special emphasis in the μ paradigm for robustness analysis, in order to introduce the notation and lexicon in use throughout this work.

We introduce the robust trajectory tracking problem in Chapter 3 and describe the different performance problems than can be reduced to it through adequate use of multiplicative and convolution weights. We also describe the noise and uncertainty models we will use. These are generalizations of the ones used in the μ framework for linear systems. We also point out in this chapter the similarities and differences between them.

Although we are not able to solve the robust trajectory tracking problem exactly, we can establish upper and lower bounds on the performance achieved. (Or correspondingly necessary conditions and sufficient conditions for a given performance specification.) The next three chapters of this thesis are devoted to the study of these necessary and sufficient conditions.

In Chapter 4, we develop a power algorithm to compute a lower bound on the nonlinear performance index. This algorithm is similar in nature to the one developed for the structured singular value, and has similar behavior. Since, as was the case for linear systems, the algorithm is not guaranteed to converge in general, its analysis is done empirically. We test this algorithm by applying it to simulations of real systems. We carry out several different performance tests on two different platforms: the Caltech ducted fan experiment and a simplified model of an F-16 jet fighter. The results of these tests are reported in Chapter 5.

The lower bound requires only local information and thus is developed for a large class of systems. In order to develop a global condition, we restrict the class of systems under study. In Chapter 6 we describe a class of rational nonlinear systems. For this class, an upper bound on the performance index for the robust trajectory tracking problem can be computed by solving a convex optimization problem in the form of a linear matrix inequality. We develop necessary and sufficient conditions for the solution of this problem by proving the equivalence between its performance and the performance of a constrained, uncertain linear system, for which μ type analysis techniques have been developed by other authors.

Finally, the last two sections of this thesis present some related work done on linear systems. Chapter 7 explores the different performance problems that can be posed and solved using μ type tests when we consider linear time varying systems over a finite time horizon. Chapter 8 discusses modifications done to the standard power algorithm for μ in order to improve its behavior in the mixed case.

The thesis is closed with a summary of the work and suggestions for future research directions.

Chapter 2
Overview of Robustness Analysis

The engineering motivation behind the theory of robust control is based on two unavoidable facts: first, all analysis and synthesis methods are based on the use of models. In most cases it is not practical to try the designs in prototypes at the early stages. Furthermore, critical control systems must have a reasonable expectation of good performance before they are implemented, since their failure can lead to significant losses. These factors force us to extensively analize systems using models before we proceed to the implementation stage. The second fact is that models are, by nature, incomplete and inaccurate. They are incomplete because they are deduced from a finite observation of the system. They are inaccurate because the observation is noisy and, what is even more important, because we are trying to project a physical system into a particular class of mathematical models we believe to be representative (e. g., the class of linear, finite dimensional model).

Robust control analysis and synthesis techniques were developed to deal with these inherent limitations of models. They make up for the incompleteness and inaccuracy in the model by associating with the system not one, but a set of models. The different models in the set will account for possible errors in the measurements and for the limitations of the measurements themselves. Furthermore, in an intuitive or ad hoc sense, limitations imposed by the model structure are more or less compensated for by the fact that several models are being considered at once.

The different branches in robust control have been developed around the choice of noise signal models and the nature of the class of systems under study. In this chapter we will review the main ideas behind the structured singular value or μ framework for robust control. Most of the work that we will develop in the rest of this thesis shares the philosophy of this framework, and therefore its terminology and notation. This chapter is intended for those readers who are not familiar with this framework. Most of the material here is standard. There are several good and complete references for the material presented here. This section is largely based on [22] and [34]. An extensive and in-depth analysis can also be found in [38].

2.1 Linear Fractional Transformations and Robust Control

Underlying every robust control paradigm there is a class or set of plants to be associated with the real system being studied. This set has to be rich enough to capture the behavior of the real system. It also has to admit a simple mathematical description. In what follows we will use extensively classes of systems described by feedback interconnections. For linear systems these interconnections are known as Linear Fractional Transformations or LFT's. Consider the system M with inputs u and v, and outputs y and z defined by the equations

$$\begin{aligned} y &= M_{11}u + M_{12}v \\ z &= M_{21}u + M_{22}v, \end{aligned}$$

with

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}.$$

Let $\boldsymbol{\Delta}$ be a set of systems Δ with dimensions compatible with z and u. Then for each $\Delta \in \boldsymbol{\Delta}$ we can define the following system

$$\begin{aligned} y &= M_{11}u + M_{12}v \\ z &= M_{21}u + M_{22}v \\ v &= \Delta z. \end{aligned} \tag{2.1}$$

We will denote the system that maps u into y, $\Delta * M$. This is the standard notation for the Redheffer star-product. Note that in the preceding development we made no mention to the nature of the system M. It can be a constant matrix or a linear operator between signal spaces. If Equations 2.1 have a unique solution for every u, we say that the system $\Delta * M$ is well posed. Finally, we define the set of systems

$$\mathbf{\Delta} * M = \{\Delta * M, \Delta \in \mathbf{\Delta}\}.$$

The sets of plants we will consider for robustness analysis will be of the form $\mathbf{B\Delta} * M$. The sets $\mathbf{B\Delta}$ will be in general of the form

$$\mathbf{B\Delta} = \{\text{blockdiag}(\Delta_1, \Delta_2, \ldots, \Delta_m), \|\Delta_i\| \leq 1\}.$$

Each Δ_i will have very simple structure, in general either diagonal or a full operator. We will refer to the structure of the elements of the set $\mathbf{\Delta}$ as the uncertainty block structure.

2.2 The μ Paradigm

The structured singular value μ was introduced in [7] as a tool for the robustness analysis of systems subject to (structured) uncertainty. The original definition of μ may be easily generalized to incorporate both real (e. g., parametric) and complex (e. g., dynamic) uncertainties (see [9] for example), and many robust stability/performance problems can be recast as one of computing μ with respect to an appropriate block structure. The notation used here is fairly standard and is essentially taken from [9] and [35]. For any square complex matrix M, we denote the complex conjugate transpose by M^*. The largest singular value and the structured singular value are denoted by $\overline{\sigma}(M)$ and $\mu(M)$ respectively. The spectral radius is denoted $\rho(M)$ and $\rho_R(M) = \max\{|\lambda| : \lambda \text{ is a real eigenvalue of } M\}$, with $\rho_R(M) = 0$ if M has no real eigenvalues. For any complex vector x, x^* denotes the complex conjugate transpose, and $|x|$ the Euclidean norm.

The definition of μ depends upon the underlying block structure of the uncertainties, which is defined as follows. Given a matrix $M \in \mathbf{C}^{n \times n}$ and three non-negative integers m_r, m_c, and m_C with $m := m_r + m_c + m_C \leq n$, the block structure $\mathcal{K}(m_r, m_c, m_C)$ is an m-tuple of positive integers

$$\mathcal{K} = (k_1, \ldots, k_{m_r}, k_{m_r+1}, \ldots, k_{m_r+m_c}, k_{m_r+m_c+1}, \ldots, k_m) \qquad (2.2)$$

where we require $\sum_{i=1}^{m} k_i = n$ in order that the dimensions are compatible. This now determines the set of allowable perturbations, namely define

$$\begin{aligned}\mathbf{\Delta}_{\mathcal{K}} = \{\Delta = \text{blockdiag}(&\delta_1^r I_{k_1}, \ldots, \delta_{m_r}^r I_{k_{m_r}}, \delta_1^c I_{k_{m_r+1}}, \ldots, \\ &\delta_{m_c}^c I_{k_{m_r+m_c}}, \Delta_1^C, \ldots, \Delta_{m_C}^C) : \delta_i^r \in \mathbf{R}, \delta_i^c \in \mathbf{C}, \\ &\Delta_i^C \in \mathbf{C}^{k_{m_r+m_c+i} \times k_{m_r+m_c+i}}\}, \end{aligned} \qquad (2.3)$$

where

$$\mathbf{B\Delta}_{\mathcal{K}} = \{\Delta : \Delta \in \mathbf{\Delta}_{\mathcal{K}}, \overline{\sigma}(\Delta) \leq 1\},$$

and I_k denotes the k-dimensional identity matrix. Note that $\mathbf{\Delta}_{\mathcal{K}} \in \mathbf{C}^{n \times n}$ and that this block structure is sufficiently general to allow for repeated real scalars, repeated complex scalars, and full complex blocks. The purely complex case corresponds to $m_r = 0$. Note also that the full complex blocks need not be square but we restrict them as such for notational convenience.

Definition 2.1 ([7]) *The structured singular value, $\mu_{\mathcal{K}}(M)$, of a matrix $M \in \mathbf{C}^{n \times n}$ with respect to a block structure $\mathcal{K}(m_r, m_c, m_C)$ is defined as*

$$\mu_{\mathcal{K}}(M) = \left(\min_{\Delta \in \mathbf{\Delta}_{\mathcal{K}}} \{\overline{\sigma}(\Delta) : \det(I - \Delta M) = 0\} \right)^{-1} \qquad (2.4)$$

with $\mu_{\mathcal{K}}(M) = 0$ if no $\Delta \in \mathbf{\Delta}_{\mathcal{K}}$ solves $\det(I - \Delta M) = 0$.

The following lemma will connect this definition with two commonly used properties of matrices.

Lemma 2.1 *The spectral radius $\rho(M)$ of the matrix M verifies*

$$\rho(M) = \mu_{\Delta_s}(M),$$

where

$$\mathbf{\Delta}_s = \{\delta I, \delta \in \mathbf{C}\},$$

and the maximum singular value $\overline{\sigma}(M)$ of M verifies

$$\overline{\sigma}(M) = \mu_{\Delta_p}(M),$$

where

$$\mathbf{\Delta_p} = \{\Delta_p, \Delta_p \in \mathbf{C^{n \times n}}\}.$$

Given two uncertainty structures $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$, define a third structure

$$\mathbf{\Delta} = \left\{ \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} : \Delta_1 \in \mathbf{\Delta}_1, \Delta_2 \in \mathbf{\Delta}_2 \right\}.$$

Denote μ_{Δ_1} the structured singular value computed with respect to $\mathbf{\Delta}_1$, μ_{Δ_2} when computed with respect to $\mathbf{\Delta}_2$ and μ when computed with respect to $\mathbf{\Delta}$. The following theorem will allow us to state the connection between the structured singular value and robust performance problems.

Theorem 2.2 (Main Loop Theorem [22])

$$\mu(M) < 1 \iff \left\{ \begin{array}{rcl} \mu_{\Delta_2}(M_{22}) & < & 1 \\ \max\limits_{\Delta_2 \in \mathbf{B\Delta}_2} \mu_{\Delta_1}(M * \Delta_2) & < & 1. \end{array} \right.$$

Theorem 2.2 can be interpreted as follows. Suppose that for a system M_p there is a property of the system (such as performance level achieved) that is achieved if and only if $\mu_{\Delta_p}(M_p) < 1$. Then all the plants in the set $B\mathbf{\Delta}_u * M$ are well posed and verify the property if and only if $\mu_{\Delta}(M) < 1$, where Δ is in the set

$$\mathbf{\Delta} = \left\{ \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} : \Delta_1 \in \mathbf{\Delta}_u, \Delta_2 \in \mathbf{\Delta}_p \right\}.$$

As an example consider the discrete time system given by the equations

$$\begin{aligned} x_{k+1} &= Ax_k + B_1 u_k + B_2 v_k \\ y_k &= C_1 x_k + D_{11} u_k + D_{12} v_k \\ z_k &= C_2 x_k + D_{21} u_k + D_{22} v_k, \end{aligned} \qquad (2.5)$$

and the associated matrices

$$M_1 = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \end{bmatrix},$$

and

$$M = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix}.$$

The system will be stable if and only if

$$\rho(A) < 1.$$

As a performance measure consider the L_2 into L_2 induced gain from u to y. This gain is less than 1 if and only if

$$\max_{\substack{\delta_1 \in \mathbf{C} \\ |\delta_1| \leq 1}} \overline{\sigma}(D + C\delta_1(I - A\delta_1)^{-1}B) < 1.$$

So the system is stable and the performance condition is met if and only if

$$\left\{ \begin{array}{rcl} \mu_{\mathbf{\Delta_s}}(A) & < & 1 \\ \max\limits_{\Delta_s \in \mathbf{B\Delta_s}} \mu_{\mathbf{\Delta_p}}(\Delta_s * M_1) & < & 1. \end{array} \right.$$

According to Theorem 2.2 these conditions are equivalent to

$$\mu_{\mathbf{\Delta_1}}(M_1) < 1,$$

where

$$\mathbf{\Delta_1} = \left\{ \begin{bmatrix} \Delta_s & 0 \\ 0 & \Delta_p \end{bmatrix} : \Delta_s \in \mathbf{\Delta_s},\ \Delta_p \in \mathbf{\Delta_p} \right\}.$$

Now add to the system the following equation, describing an uncertain component

$$v = \delta_u z \tag{2.6}$$

with $\delta_u \in \mathbf{R}$, and $|\delta_u| \leq 1$. The system is well posed for all δ_2 if and only if

$$\rho(D_{22}) < 1 \Leftrightarrow \mu_{\mathbf{\Delta_u}}(D_{22}) < 1,$$

where

$$\mathbf{\Delta_u} = \{\delta_u I : \delta_u \in \mathbf{C}\},$$

and the performance condition is met for all δ_u if and only if

$$\max_{\delta_u \in [-1,1]} \mu_{\mathbf{\Delta_1}}(\delta_u I * M) < 1.$$

Applying Theorem 2.2 again we see that the system given by Equations (2.5) and (2.6) is stable and well posed for all $\delta_2 \in [-1, 1]$ and has induced gain from u to y less than 1 if and only if

$$\mu_{\mathbf{\Delta_2}}(M) < 1$$

with

$$\boldsymbol{\Delta}_2 = \left\{ \begin{bmatrix} \Delta_s & 0 & 0 \\ 0 & \Delta_p & 0 \\ 0 & 0 & \delta_u I \end{bmatrix} : \Delta_s \in \boldsymbol{\Delta}_{\mathbf{s}}, \Delta_p \in \boldsymbol{\Delta}_{\mathbf{p}}, \delta_u \in \mathbf{C} \right\}.$$

Several other robust performance problems for LTI systems can be written as μ computations. The reader is referred to [22] or [34] for more examples.

2.3 Upper and Lower Bounds

The quantity introduced in the previous section — the structured singular value μ — can be used to determine whether or not a family of system meets a performance specification. The structured singular value is a function that in general has to be computed numerically. However, for many interesting uncertainty structures, the problem of computing μ has been proven to be NP-hard. (See for example [32] or [4].) In practical terms these complexity results mean that in general we will only be able to compute upper and lower bounds on μ. A substantial part of the research effort in this area has been devoted to the development of practical algorithms for the computation of these bounds. The nature of these algorithms is of particular interest to us, since the work in this thesis extends them to a certain class of nonlinear problems. In this section we will review these algorithms.

Lower bound

In order to obtain a lower bound for μ, we define the following sets of block diagonal matrices (which are also dependent on the underlying block structure).

$$\mathcal{Q}_{\mathcal{K}} = \{\Delta \in \boldsymbol{\Delta}_{\mathcal{K}} : \delta_i^r \in [-1\ 1], \delta_i^{c*}\delta_i^c = 1, \Delta_i^{C*}\Delta_i^C = I_{k_{m_r+m_c+i}}\} \qquad (2.7)$$

$$\begin{aligned} \mathcal{D}_{\mathcal{K}} = \{ & block\ diag(e^{j\theta_1}D_1, \ldots, e^{j\theta_{m_r}}D_{m_r}, D_{m_r+1}, \ldots, D_{m_r+m_c}, \\ & d_1 I_{k_{m_r+m_c+1}}, \ldots, d_{m_C} I_{k_m}) : \theta_i \in [-\frac{\pi}{2}\ \frac{\pi}{2}], \\ & 0 < D_i = D_i^* \in \mathbf{C}^{k_i \times k_i}, 0 < d_i \in \mathbf{R}\} \end{aligned} \qquad (2.8)$$

The key to obtaining a lower bound lies in the fact that the μ problem may be reformulated as a real eigenvalue maximization with respect to the scaling set defined in (2.7). The following theorem is taken from [35].

Theorem 2.3 ([35]) *For any matrix $M \in \mathbb{C}^{n\times n}$, and any compatible block structure $\mathcal{K}$,*

$$\max_{Q\in\mathcal{Q}_{\mathcal{K}}} \rho_R(QM) = \mu_{\mathcal{K}}(M). \tag{2.9}$$

This immediately gives us a theoretical lower bound since we have that for any $Q \in \mathcal{Q}_{\mathcal{K}}$, $\rho_R(QM) \le \mu_{\mathcal{K}}(M)$. The idea then is to find an efficient way to compute a local maximum of the function $\rho_R(QM)$ over $Q \in \mathcal{Q}_{\mathcal{K}}$. Note that since this function is non-convex, we cannot guarantee to find the global maximum and hence we only obtain a lower bound for μ.

It was shown in [35] that the maximization in (2.9) can be tackled via a power iteration. We will not go into any of the details of the theoretical development here, but merely present the final result. This algorithm will form the starting point for the development of an adaptive power iteration to find a μ lower bound.

In order to avoid the notation becoming excessive, we consider a simple block structure with $m_r = m_c = m_C = 1$ for the remainder of this section. This is purely for notational convenience, and that the general formulae for an arbitrary block structure, as defined earlier, are simply obtained by duplicating the appropriate formulae for each block. So given $\mathcal{K} = (k_1, k_2, k_3)$, the appropriate scaling set becomes

$$\mathcal{Q}_{sub} = \{\text{blockdiag}(q^r I_{k_1}, q^c I_{k_2}, Q^C) : q^r \in [-1\ 1], q^{c*}q^c = 1,$$
$$Q^{C*}Q^C = I_{k_3}\}. \tag{2.10}$$

Now consider four vectors $b, a, z, w \in \mathbb{C}^n$ and partition them compatibly with this block structure as

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \quad z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}, \quad w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \tag{2.11}$$

where $b_i, a_i, z_i, w_i \in \mathbb{C}^{k_i}$. Now allow these vectors to evolve via the following power iteration:

$$\tilde{\beta}_{k+1} a_{k+1} = M b_k \tag{2.12}$$

$$z_{1_{k+1}} = \tilde{q}_{k+1} w_{1_k} \qquad z_{2_{k+1}} = \frac{w_{2_k}^* a_{2_{k+1}}}{|w_{2_k}^* a_{2_{k+1}}|} w_{2_k} \tag{2.13}$$

$$z_{3_{k+1}} = \frac{|w_{3_k}|}{|a_{3_{k+1}}|} a_{3_{k+1}}$$

$$\hat{\beta}_{k+1} w_{k+1} = M^* z_{k+1} \tag{2.14}$$

$$b_{1_{k+1}} = \hat{q}_{k+1} a_{1_{k+1}} \qquad b_{2_{k+1}} = \frac{a_{2_{k+1}}^* w_{2_{k+1}}}{|a_{2_{k+1}}^* w_{2_{k+1}}|} a_{2_{k+1}} \tag{2.15}$$

$$b_{3_{k+1}} = \frac{|a_{3_{k+1}}|}{|w_{3_{k+1}}|} w_{3_{k+1}}$$

where $\tilde{q}_{k+1}$ and $\hat{q}_{k+1}$ evolve as

$$\tilde{\alpha}_{k+1} = sgn(\hat{q}_k) \frac{|b_{1_k}|}{|a_{1_{k+1}}|} + Re(a_{1_{k+1}}^* w_{1_k}) \tag{2.16}$$

$$\text{If } |\tilde{\alpha}_{k+1}| \geq 1 \quad \text{Then } \tilde{q}_{k+1} = \frac{\tilde{\alpha}_{k+1}}{|\tilde{\alpha}_{k+1}|} \quad \text{Else } \tilde{q}_{k+1} = \tilde{\alpha}_{k+1}$$

$$\hat{\alpha}_{k+1} = sgn(\tilde{q}_{k+1}) \frac{|b_{1_k}|}{|a_{1_{k+1}}|} + Re(a_{1_{k+1}}^* w_{1_{k+1}})$$

$$\text{If } |\hat{\alpha}_{k+1}| \geq 1 \quad \text{Then } \hat{q}_{k+1} = \frac{\hat{\alpha}_{k+1}}{|\hat{\alpha}_{k+1}|} \quad \text{Else } \hat{q}_{k+1} = \hat{\alpha}_{k+1}$$

and $\tilde{\beta}_{k+1}, \hat{\beta}_{k+1}$ are chosen positive real so that $|a_{k+1}| = |w_{k+1}| = 1$.

Then it was shown in [35] that if the above iteration converges to an equilibrium point, we have a matrix $Q \in \mathcal{Q}_{sub}$ such that $QMb = \tilde{\beta} b$ and $w^* QM = \hat{\beta} w^*$, so that $\max(\tilde{\beta}, \hat{\beta})$ gives us a lower bound for $\mu_{\mathcal{K}}(M)$. Furthermore, if $\tilde{\beta} = \hat{\beta}$ then this bound corresponds to a local maximum of (2.9).

Substantial numerical experience with the algorithm has revealed that it usually converges fairly rapidly, and that we do obtain $\tilde{\beta} = \hat{\beta}$ in practice so that the resulting bound is usually a good one (i. e., close to μ.) However, in some cases the iteration does not converge, and the resulting guess for $Q \in \mathcal{Q}_{\mathcal{K}}$ yields a poor lower bound for μ. For a more detailed discussion of the power iteration performance, see [36].

Upper bound

For any $Q \in Q_{\mathcal{K}}$, for any pair of invertible matrices D_l and D_r such that $D_l Q = Q D_r$, the following series of inequalities hold

$$\begin{aligned}
\rho(QM) &\leq \overline{\sigma}(QM) \\
&= \overline{\sigma}(D_l^{-1} D_l Q M) \\
&= \overline{\sigma}(D_l^{-1} Q D_r M) \\
&= \overline{\sigma}(Q D_r M D_l^{-1}) \\
&\leq \overline{\sigma}(D_r M D_l^{-1}).
\end{aligned}$$

From these inequalities we can derive the following

Theorem 2.4

$$\mu(M) = \max_{Q \in Q_{\mathcal{K}}} \rho(QM) \leq \min_{D_l, D_r} \overline{\sigma}(D_r M D_l^{-1}),$$

where D_l, and D_r are invertible and verify

$$D_l Q = Q D_r \quad \forall Q \in Q_{\mathcal{K}}.$$

Theorem 2.4 provides us with an upper bound to the structured singular value in the form of a minimization problem. This problem can be shown to be convex [22]. The last inequality in the theorem can be rewritten to take the form of a Linear Matrix Inequality as shown by the following series of inequalities

$$\begin{array}{rrcll}
& \overline{\sigma}(D_r M D_l^{-1}) & < & 1 & \\
\Leftrightarrow & \overline{\sigma}(D_r M D_l^{-1}) - 1 & < & 0 & \\
\Leftrightarrow & u^*((D_l^{-1})^* M^* D_r^* D_r M D_l^{-1} - I)u & < & 0 & \forall u \\
\Leftrightarrow & u^*(D_l^{-1})^*(M^* D_r^* D_r M - D_l^* D_l) D_l^{-1} u & < & 0 & \forall u \\
\Leftrightarrow & u^*(M^* D_r^* D_r M - D_l^* D_l) u & < & 0 & \forall u \\
\Leftrightarrow & M^* D_r^* D_r M - D_l^* D_l & < & 0. &
\end{array}$$

When parts of the uncertainty structure are real, this upper bound can be further refined. For a complete development of this issue the reader is referred to [34]. Recently, there has been significant progress in the development of numerical solutions for linear matrix inequalities. Software packages for solving LMI's are now commercially available and perform reasonably well.

Another significant feature of the upper bound presented here is that it also has an interpretation as an exact test (i. e., necessary and sufficient condition) for robust performance. In general terms the upper bound given in Theorem 2.4 is a test for robust performance, when all the uncertainty components are norm bounded operators from L_2 into L_2. For a complete treatment of this subject see [24].

2.4 Extensions to μ

Recent work has shown that in many cases, considering a slightly more general version of the system described in the previous section allows us to set additional robust performance questions in the μ framework (or a variation of it). These modifications consist of adding linear constraints that involve all the variables in the system (states, inputs and outputs) equally. These systems are called implicit systems, and a full treatment of them can be found in [24]. It is shown there that robust performance can be determined by computing a function of the system and constraints matrices similar to μ. It is also shown that upper and lower bounds similar to the ones for the standard structured singular value can be determined. (Although in the case of the lower bound, computation is a lot harder.) In this section we will present some results obtained for this class of systems. We will use these results in Chapters 6 and 7.

Consider the following system of equations:

$$\left\{ \begin{array}{rcl} y & = & Mu \\ u & = & \Delta y \\ 0 & = & Cu, \end{array} \right. \tag{2.17}$$

where M and C are constant matrices, and Δ is a block diagonal matrix verifying $\overline{\sigma}(\Delta) \leq 1$. The above system can also be written

$$0 = \begin{bmatrix} I - \Delta M \\ C \end{bmatrix} u.$$

With this representation we can define an extension to the structured singular value:

Definition 2.2 ([24]) *The structured singular value of the system defined by equations (2.17) is defined as: If*

$$\ker \begin{bmatrix} I - \Delta M \\ C \end{bmatrix} = \{0\} \quad \text{for all } \Delta \in \boldsymbol{\Delta},$$

define the structured singular value of the system in 2.17 to be

$$\mu_{\Delta,C} = 0;$$

otherwise define it as

$$\mu_{\Delta,C} = \left(\min \left\{ \overline{\sigma}(\Delta) : \Delta \in \boldsymbol{\Delta}, \ker \begin{bmatrix} I - \Delta M \\ C \end{bmatrix} \neq \{0\} \right\} \right)^{-1}.$$

The connection between this definition and robust stability/robust performance of the system is given by the following:

Theorem 2.5 *If M and C are finite dimensional, discrete time LTI systems and Δ is LTI, then the system described by equations (2.17) is stable if and only if $\mu_{\Delta,C(e^{j\omega})}(M(e^{j\omega}) < 1$ for all $\omega \in [-\pi, \pi]$.*

Finally, we present a result that allows us to compute an upper bound on the extended structured singular value. We will use this upper bound later in this thesis.

Theorem 2.6 ([24]) *If there exist invertible matrices D_l and D_r such that*

$$D_l \Delta = \Delta D_r \quad \forall \Delta$$

and

$$(C^{\perp})^* (M^* D_r^* D_r M - D_l^* D_l) C^{\perp} < 0, \tag{2.18}$$

the system of equations (2.17) has only the trivial solution.

The linear matrix inequality (2.18) gives us a sufficient condition for robust performance in this class of systems. As was the case with the standard μ upper bound, this condition can also be proven necessary and sufficient for robust stability for an appropriate class of uncertain operators Δ.

Chapter 3
Robustness Analysis for Nonlinear Systems

Robustness analysis for linear systems has traditionally been approached as a disturbance rejection or gain minimization problem. This approach relies on the fact that rejecting noise around the null trajectory is equivalent to rejecting noise around any trajectory. When extending this concept to nonlinear systems, we will have to account for the fact that the superposition principle will not hold. A successful paradigm for robustness analysis of nonlinear systems must be based on the study of the behavior of the system around prespecified trajectories.

Also, linear system analysis is generally done over an infinite time horizon. (Although final time horizon versions of some of the linear analysis results have been developed.) When studying linear time invariant systems, an infinite time horizon actually simplifies both the setup of the problem and the algorithms used to solve it. An important reason for this simplicity is that the behavior of a linear system over an infinite horizon can be completely characterized by a finite set of numbers. (For example, the entries in the system matrices A, B, C, and D in the usual state space representation.) This would only be possible for limited and highly structured subsets of the class of nonlinear systems. However, the behavior of a fairly large class of nonlinear systems can be specified, at least locally, with finite information if the system is studied over a finite interval.

In this thesis we develop a paradigm for robustness analysis of nonlinear systems which extends the ideas from linear theory. In doing this we incorporate the differences noted above. Far from being a restriction, by taking into account these issues — analysis around fixed trajectories and study of a finite interval — we develop an analysis procedure than can easily be applied to a large class of practical problems.

Many nonlinear analysis problems of engineering interest can be reduced to a problem of tracking a nominal trajectory. We mention here a few examples:

- A car changing lanes on an automated highway. By communicating with neighboring vehicles the car first determines that there is sufficient clearance in the adjoining lane to change position. Then it performs the maneuver with a nominal steering command, corrected with an inner loop controller.
- An airplane performing an automatic change of altitude and heading. The pilot enters the new heading and altitude and the flight computer determines nominal commands to perform it. A second control loop maintains the airplane around the planned trajectory.
- A power plant going through a change in load. The plant is designed to respond to the load change in a particular manner. However, varying conditions (temperature, component wear) and external disturbances can cause deviations from the nominal.

In all these cases, the designer has in mind an appropriate path to be completed in a finite predetermined time, and builds his control system accordingly. Since the real system is not exactly the one used for the design, and since it is also subject to noise, the system will not follow the intended trajectory. The question of interest becomes: will the real trajectory, under the worst conditions possible, remain close to the nominal one in an appropriate norm? We call this question the robust trajectory tracking problem. (We give it a more formal definition in the next section.)

The robust trajectory problem is naturally set up as an optimization problem: given an uncertain nonlinear system subject to noise, maximize the distance to the nominal trajectory for allowable noise signals, parameter values, and uncertain dynamical components. However, in order for this statement to be meaningful, we have to find a class of problems large enough to have engineering relevance, for which we can efficiently find useful bounds to this optimization problem. In this selection there is necessarily a tradeoff between the generality and applicability of the class of problems under study, and the efficiency of the algorithms developed to compute them. The μ framework achieves such a tradeoff. The uncertainty models include real parameters varying in intervals and norm bounded dynamical components. This gives the framework flexibility at a low cost: it is reasonably straightforward to reliably obtain the necessary model-uncertainty pair from standard identification techniques. It is also straightforward to include engineering knowledge about the process (such as noise or under-modeled components bandwidth) into the noise description. Furthermore, together with recent developments, the μ framework can be used to tackle disturbance rejection with respect to two of the most common sources of noise: white noise and harmonic noise. Finally, the computation tools for upper and lower bounds of μ are efficient: robustness analysis for a system as sophisticated as the space shuttle are routinely performed.

We would like to achieve this same balance between generality, ease of use, and efficiency in computation, in the development of an analysis paradigm for nonlinear systems. We thus the μ-framework use as a starting point in the development of robustness analysis tools for nonlinear systems. In the first section of this chapter we describe the particular robust trajectory tracking problem with which this thesis is concerned. In the second section we will show that extending the μ framework to nonlinear systems makes it more flexible, and that it allows us to ask more natural performance questions.

3.1 The Robust Trajectory Tracking Problem

We are concerned with the study of perturbations of systems around a prespecified nominal trajectory over a finite time horizon with initial time t_i and final time t_f. The performance specifications and all characterizations of the noise signals, under-modeled components, and uncertain parameters affecting the system are given over this time interval.

We restrict our study to nonlinear systems whose dynamics can be represented by a smooth function F of the states x, a set of signals u corresponding to external disturbances, a set of signals v corresponding to the effect of under-modeled components, a set of parameters δ, and a set of nominal commands U responsible for steering the nominal system along the nominal trajectory. The evolution of the system will thus be the solution to

$$\dot{x} = F(x, u, v, \delta, U, t),$$

with initial conditions $x = x_o$. The trajectory will be described by a set of coordinates Y, given by a smooth function G of the same variables

$$Y = G(x, u, v, \delta, U, t).$$

The nominal trajectory Y_n, will thus be given by

$$Y_n = G(x, 0, 0, \delta_n, U, t),$$

where δ_n are the nominal values for the parameters.

The signals v are not independent of the other signals, but the exact dependence is not known (hence the name undermodeled dynamics). We assume, however, that v is the image under a structured, norm bounded operator of a set of variables z, given as a smooth function of the other variables in the problem

$$z = H(x, u, v, \delta, U, t).$$

We will denote Δ the mapping from z to u, $\boldsymbol{\Delta}$ the class of operators with same block diagonal structure. Since we will analyze the system for a fixed nominal trajectory, we can incorporate the nominal commands into the functions F, G, and H, and define

$$\begin{aligned} f_U(x, u, v, \delta, t) &= F(x, u, v, \delta, U, t) \\ h_U(x, u, v, \delta, t) &= H(x, u, v, \delta, U, t). \end{aligned}$$

Since we are only interested in the error between the nominal trajectory and the actual one, we will also define

$$g_U(x, u, v, \delta, t) = G(x, u, v, \delta, U, t) - Y_n.$$

Whenever no confusion is possible we will drop the subscript U. The structure of the operator Δ we will consider is block diagonal, and will thus induce a partition in the sets of signals v and z

$$\begin{aligned} v &= [v_1, v_2, \ldots, v_p] \\ z &= [z_1, z_2, \ldots, z_p]. \end{aligned}$$

Note that v and z are divided in the same number of blocks of signals, but correspondent blocks do not have to have the same number of signals. The size of the noise signals entering the problem will also require the introduction of a partition of the signal set u

$$u = [u_1, u_2, \ldots, u_m].$$

Figure 3.1 shows a diagram of the system interconnection. There are several similarities with the linear μ framework reviewed in the previous chapter, and some differences. The undermodeled dynamical components are feedback loops as was the case in linear systems. However, the effect of the parameters on the system can be more general. The dynamics are not only nonlinear, but they are also non-autonomous: the system includes explicit functions of time.

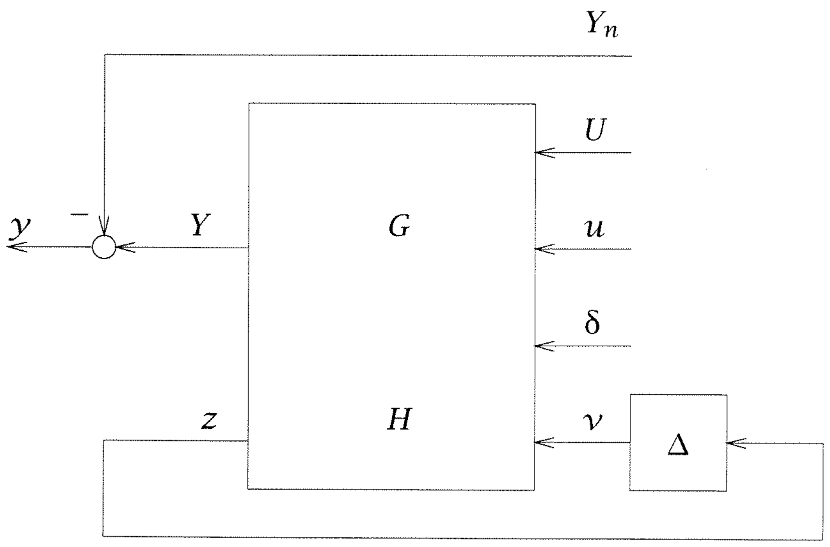

Figure 3.1: Uncertain system interconnection.

We will measure the sizes of all signals in the 2-norm defined as usual over a finite time horizon

$$\|a\|_2 = \left[\frac{1}{t_f - t_i} \int_{t_i}^{t_f} a^t a \, dt \right]^{\frac{1}{2}}.$$

The performance index is given by the 2-norm of the error signal

$$J = \|y\|_2.$$

The size of the noise signals will be specified for each block of signals in the partition given

$$\|u_i\|_2 = N_i \qquad i = 1, 2, \dots, m.$$

All the information we have on the undermodeled dynamical components is that Δ is in $\mathbf{B\Delta}$. We will use the induced 2-norm as a measure of the size of Δ. This restriction is then equivalent to imposing the following relations between the signals v and z

$$\|v_i\|_2 = \|z_i\|_2 \qquad i = 1, 2, \dots, p.$$

We will allow the parameters δ to vary in closed intervals

$$d_i \leq \delta_i \leq D_i \qquad i = 1, 2, \dots, r.$$

Finally, we will also allow some or all of the initial conditions to vary in given closed intervals

$$c_i \leq x_i(t_i) \leq C_i \qquad i = 1, 2, \dots, n.$$

The preceding performance, noise, and uncertainty descriptions given in this section can be summarized as a constrained optimization problem. Since we only use one norm, we will drop the subscript 2 from now on to keep the notation unencumbered. We now state the following:

Problem 3.1 (Robust Trajectory Tracking) *Given an uncertain nonlinear system, driven by the equation*

$$\dot{x} = f_U(x, u, v, \delta, t)$$

and the nominal command signal U, with initial conditions satisfying

$$c_i \leq x_i(t_i) \leq C_i \qquad i = 1, 2, \ldots, n,$$

what is the maximum value of the norm of the error signal

$$\|y\| = \|g_U(x, u, v, \delta, t)\|$$

subject to the constraints

$$\begin{array}{rcll} \|u_i\|_2 & = & N_i & i = 1, 2, \ldots, m \\ \|v_i\|_2 & = & \|z_i\|_2 & i = 1, 2, \ldots, p \\ d_i \leq & \delta_i & \leq D_i & i = 1, 2, \ldots, r \ ? \end{array}$$

This problem may seem too specific to be of any consequence in real applications. When analyzing linear systems, we use different kinds of multiplicative weighting functions (weights) to impose a frequency content to noise signals, to determine the bandwidth of an uncertain operator, or to determine the region in the frequency domain over which we would like to impose the performance specification. The use of these weights adds flexibility to the linear analysis techniques, and is fundamental in establishing the validity of the method. With similar techniques we can reduce many interesting problems to a particular instance of Problem 3.1. In the following section we present several examples showing how common performance, noise, and uncertainty specifications can be set in or approximated by the framework proposed.

3.2 Use of Weights and Multipliers

Given that the combined system (plant plus weights) is nonlinear, we are not restricted in our choice of weights to linear time invariant filters. Of particular importance is the fact that we can use time domain multipliers (e. g., smooth saturation functions) to modify the original problem. A combination of these techniques will allow us, for example, to impose a desired distribution of signal energy both over frequency and over the time interval studied. Two particularly useful time domain multipliers are smooth steps and square pulses in time. We denote $S_\tau(t)$ the step function defined as

$$S_\tau(t) = \begin{cases} 1 & t \leq \tau \\ 0 & t > \tau + \epsilon \\ \text{smooth} & \text{everywhere else,} \end{cases}$$

and denote $P_{\tau,\sigma}(t)$ the pulse function defined as

$$P_{\tau,\sigma}(t), = \begin{cases} 1 & \tau \leq t \leq \sigma \\ 0 & t > \sigma + \epsilon \\ 0 & t < \tau - \epsilon \\ \text{smooth} & \text{everywhere else,} \end{cases}$$

where ϵ is small compared to the time constants for the problem at hand. In the rest of this section we show examples of how these and other weights, combined with 2-norm bounds on signals and the addition of parameter and under-modeled components, can be used to represent the performance measures, noise signals and uncertain dynamics arising in different robust performance problems.

Performance measures

The general problem stated in the last section uses the 2-norm, computed over the entire time horizon over which the system is allowed to evolve as the measure of the performance attained. However, this is not necessarily the objective of interest in engineering applications.

In many applications we are primarily interested in hitting a certain target at the end of the trajectory. This is the case, for example, in auto-landers, where the objective is to touch down within a certain rectangle in the runway. It is possible, however, to approximate this performance objective by defining a new performance variable

$$y' = \frac{t_f - t_i}{t_f - \tau}(1 - S_\tau(t))y$$

where τ is chosen so that the interval of interest for the performance measure is $[\tau, t_f]$. There is a tradeoff between the correlation between $\|y'\|_2$ and $y(t_f)$, and the ease of computation of the solution. When $\tau \rightarrow$

t_f, (and $\epsilon \to 0$), $\|y'\|_2 \to |y(t_f)|$. However, in general, the closer τ is to t_f, the harder the computation becomes.

Another common performance measure is the maximum over time of the error signal. Again this can be approximated by the use of a weighting function and by introducing a new parameter to the system. Define the new error variable

$$y' = \frac{t_f - t_i}{\delta_t} P_{\delta_o, \delta_o + \delta_t}(t) y$$

and allow the parameter δ_o to vary in the interval

$$t_i \le \delta_o \le t_f - \delta_t.$$

We will then have that

$$\lim_{\delta_t \to 0} \lim_{\epsilon \to 0} \max_{\delta_o}(\|y'\|) = \max_{t_i \le t \le t_f}(|y(t)|).$$

As in the previous example there is a tradeoff between the ease of computation and the accuracy with which the $\|y'\|_2$ approximates the desired quantity. In Chapter 5 we will present examples of the use of these two setups.

Remarks: We can use different weights on the different components of the performance variable y. Ramps, or higher order functions, can be used instead of steps to weight differently the error signal across the time horizon. Different implementations can be used for the step and pulse functions used in this section. Gaussian pulses can be used instead of square ones, for example. An important issue to keep in mind when choosing the weighting function is the numerical conditioning of the system with respect to numerical differentiation. The weighting functions have to be tailored to the particular dynamics under study and to the integration algorithm used to solve the equations. Finally, besides the time domain weighting functions described, we can use frequency domain filters as we do in the linear time invariant case.

Noise signals

The general problem is limited to disturbance signals with fixed 2-norm over the time interval under study. As was the case with the performance specification, with the use of adequate time and frequency domain weighting functions, we can make this specification more realistic with respect to applications of practical interest.

Although intuitively a larger disturbance should produce a larger error, this is not necessarily the case in a nonlinear system. We may in principle be interested in a noise specification of the form

$$\|u_i\| \leq N_i. \tag{3.1}$$

By adding one parameter to the description of the system δ_u, we can represent the constraint (3.1) within the framework of the general problem

$$\begin{aligned} u_i &= \delta_u u_i' \\ \|u_i'\| &= N_i \\ |\delta_u| &\leq 1. \end{aligned}$$

The general problem does not consider any particular distribution of the "energy" in the disturbance signals across the interval under consideration. The signals in the allowable set can have an average value over a wide support or a peak over a narrow support. Depending on the application we may want to study the effects on the system of the former, the latter or a combination of the two.

If we want to study the effect of duration δ_t pulses on the error signals, we can use the following setup. Add a new parameter to the description of the system δ_o and a new signal u' verifying the equation

$$u = \frac{t_f - t_i}{\delta_t} P_{\delta_o, \delta_o + \delta_t} u'$$

and maximize the performance index with the additional constraints

$$\|u'\| \leq N_i$$

and

$$t_i \leq \delta_o \leq t_f - \delta_t.$$

It is harder to study the effects of signals that have a more or less uniform distribution over the time interval $[t_i, t_f]$, using the general setup of Problem 3.1. A first approach is to use a low pass filter, to make sharp noise signals — using a sizable part of the allotted 2-norm in high frequency components that are subsequently filtered and discarded — less attractive to the optimization agent. Another possibility is to section the interval $[t_i, t_f]$ into smaller intervals and to introduce a set of signals, each of which is allowed a fraction of the total norm, and use time domain weights to make each of these signals act on only one of the subintervals. If we want to divide the interval in two, for example, we introduce two new noise signals u' and u'' such that

$$u_i = S_{t_m} u' + (1 - S_{t_m}) u''$$

where t_m is the center point of the interval $[t_i, t_f]$, and add the constraints

$$\begin{aligned} \|u'\| &= \frac{\sqrt{2}}{2} N_i \\ \|u''\| &= \frac{\sqrt{2}}{2} N_i. \end{aligned}$$

This arrangement is schematically described in Figure 3.2. The two methods described can be combined. As in previous cases, we will have a tradeoff between how accurately we want to impose the uniformity of the signal and the size of the problem we have to solve.

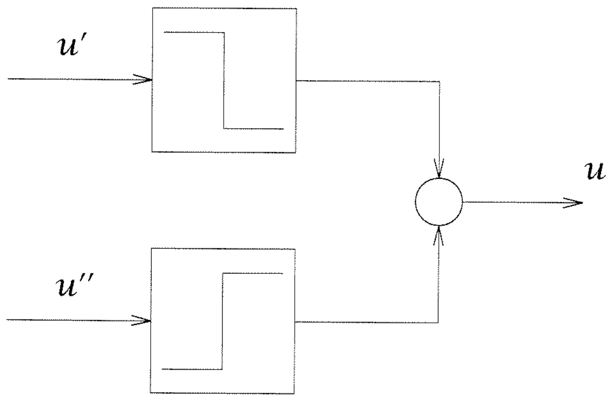

Figure 3.2: Input energy distribution.

Under-modeled dynamics

The only constraint put on the under-modeled components in the robust trajectory tracking problem is on their norm. This constraint will allow operators that are non-causal in the sense that it is possible to have

$$\int_{t_i}^{t} v_i^t v_i \, dt > \int_{t_i}^{t} z_i^t z_i \, dt. \quad t_i < t < t_f \tag{3.2}$$

The techniques described in this section do not allow us to impose constraints that will make inequality (3.2) impossible. We can, however, impose boundaries in the time interval $[t_i, t_f]$ across which the under-modeled operator Δ is not allowed to move signal energy. In many situations this can be sufficient. (For example, a step in one of the nominal command signals can produce a pulse in the signal z at a certain instant t_o. Since this is known beforehand, we can prevent the operator Δ from mapping the energy contained in that pulse to the signal v at instants previous to t_o.) If we want to impose one such barrier, for example, we introduce two new uncertain operators Δ' and Δ'', and two new signals v' and v'', verifying

$$\begin{aligned} v' &= \Delta' S_{t_o} z \\ v'' &= \Delta''(1 - S_{t_o}) z \\ v &= S_{t_o} v' + (1 - S_{t_o}) v''. \end{aligned}$$

This arrangement is shown schematically in Figure 3.3.

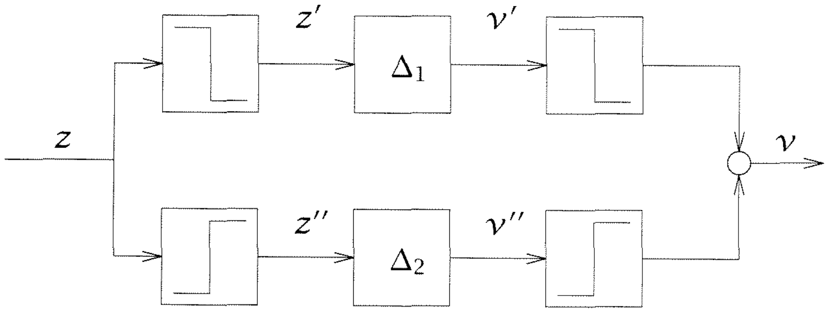

Figure 3.3: Undermodeled dynamics energy distribution.

The list of examples presented in this section is not exhaustive. It shows that even though the robust trajectory tracking problem as presented in Problem 3.1 is fairly restrictive, by using adequate filters and multipliers, it captures a large class of problems of engineering interest.

However, merely reducing an engineering question to an optimization problem does not necessarily get us any closer to solving it. The optimization problem has to accept a reasonable computational solution. Reasonable means accurate, well behaved numerically, and with slow growth in computation time with the size of the problem. In the following chapters of this thesis, we will develop algorithms for computing upper and lower bounds for the answer to Problem 3.1, and study their behavior on several examples.

3.3 Example

In this section we will briefly introduce an example to illustrate the concepts developed previously. An in-depth study of this and other examples is reported in Chapter 5.

The vehicle under study is a flying fan attached to a rotating stand. The flow can be vectored by means of controlled flaps. (For a complete description of the vehicle, see Chapter 5.) The nominal trajectory is a step in altitude, starting and ending at hover. The maneuver is to be completed within 9 seconds. Our principal performance objective is the accuracy in vertical position at the end of the interval. In this case Y will have only one component, namely the altitude of the vehicle. We will penalize the error between the actual trajectory and the nominal one on the whole 9 second interval, but we will penalize more the later instants. We choose a ramp as weighting function. The performance index used is

$$J = \left\| \frac{t}{9}(Y - Y_n) \right\|$$

where t is time in seconds, and Y_n is the nominal trajectory. The nominal model does not account for aerodynamic forces on the vehicle. We will model their effect through three disturbances entering as torques along the three axes of rotation. The 2-norm of each of these noise signals is set to be .006 Nm. The uncertain block tries to capture the undermodeled dynamics of the electrical motor that is driving the fan. The input to the uncertain block is the command signal given to the motor filtered through a high pass filter. The uncertain parameter represents the application point of the reaction force on the vehicle, since this point changes with, among other factors, the position of the flaps. This distance is considered to be .25m + δ with $|\delta| < .1$m. A schematic diagram of the necessary system setup is given in Figure 3.4. The original performance analysis problem is thus rewritten as an instance of Problem 3.1 asked of the system enclosed in the dashed rectangle.

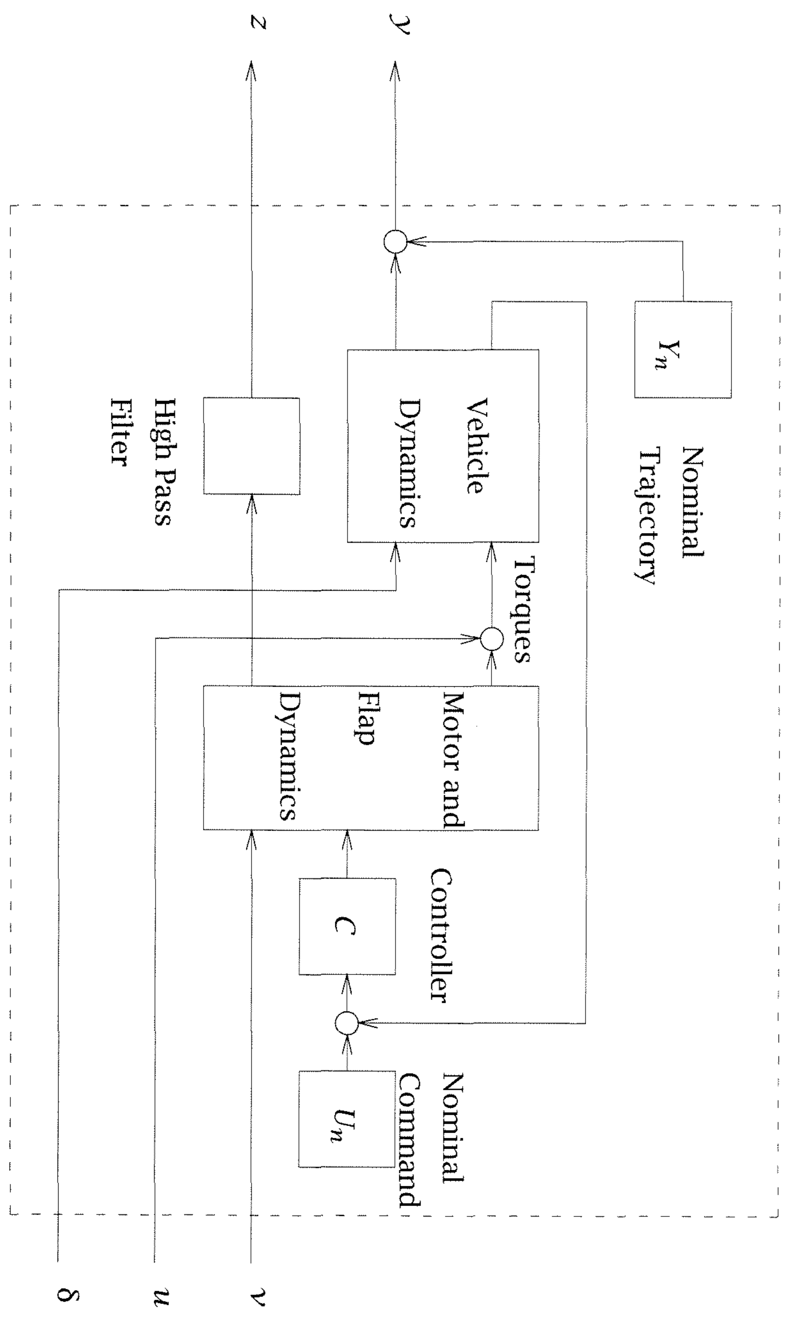

Figure 3.4: Ducted fan example setup.

Chapter 4
Necessary Conditions

Evaluating worst case performance for an uncertain system, either linear or nonlinear, is in general a non convex optimization problem. Obtaining an exact answer is thus in general going to be extremely difficult if not impossible. For linear systems it has been established that for many problems of interest, the associated decision problem (i. e., is the performance level better than a given value γ ?) is NP-hard, further suggesting that exact solutions cannot be obtained using algorithms with realistic computation growth.

In general, we will thus have to settle for upper and lower bounds on these measures. The nature of these bounds is going to be very different. We are looking for upper and lower bounds on a constrained maximization problem. These problems take the general form

$$\max_{\sigma\in\Sigma} \mathcal{F}(\sigma). \tag{4.1}$$

An upper bound is derived by constructing a set $\Sigma' \supset \Sigma$, such that the optimization problem,

$$\max_{\sigma\in\Sigma'} \mathcal{F}(\sigma)$$

is convex and admits an exact solution or at least a practical numerical one. Since $\Sigma' \supset \Sigma$ we will have

$$\max_{\sigma\in\Sigma} \mathcal{F}(\sigma) \le \max_{\sigma\in\Sigma'} \mathcal{F}(\sigma).$$

Lower bounds are in principle much easier to obtain. Any evaluation of the function $\mathcal{F}$ will give us a lower bound. For every $\sigma \in \Sigma$,

$$\mathcal{F}(\sigma) \le \max_{\sigma\in\Sigma} \mathcal{F}(\sigma).$$

A lower bound for Equation (4.1) can then be obtained through repeated evaluation of $\mathcal{F}$. For many applications, this is actually the current state of the art. This approach is not as naïve as it may seem. It provides substantially more information about the system than just the value of the lower bound. And when applied to dynamical system problems, it provides, without much extra computation, lower bounds for several optimization indices at once. This follows from the fact that the most expensive component of evaluating $\mathcal{F}$ in this case is the integration of the equations of motion. Once these equations have been integrated, the evaluation of different performance measures is usually very simple. A better lower bound for the specific performance objective under consideration can be obtained by local optimization. Although in principle this optimization could be tackled using any of the standard maximization algorithms available, our experience with linear systems tells us that they are usually inefficient in this kind of problem. Power-like algorithms usually obtain the same or better answers in much shorter time. (See [20] for a comparison in the linear case.)

In order to develop a power-like algorithm, we first have to establish conditions that are met at a local maximum of the function (necessary conditions) and then write an iteration whose equilibrium points verify the same conditions.

In this chapter we will show how the robust trajectory tracking problem presented in Chapter 3 can be recast in the standard optimization framework of the Euler-Lagrange theorem, and how this theorem can be used to derive necessary conditions for robust performance. Finally, we will derive from these necessary conditions an efficient power-like algorithm to compute a lower bound for the robust performance measure. In Chapter 6 we will develop an upper bound for this problem when the nonlinear systems are restricted to be in a particular class.

4.1 The Euler-Lagrange Optimization Framework

The first step in deriving a power algorithm is establishing conditions characterizing local maxima. We will use first order conditions for local extrema.

First order conditions for extrema of dynamical systems have been developed for different optimization indices and signal constraints. (See for example [15] or [5].) We will derive the necessary conditions for a local maximum of Problem 3.1 from the Euler-Lagrange optimization setup. In this section we will review the standard Euler-Lagrange optimization framework for dynamical systems; in the following section we will show how the robust trajectory tracking problem reduces to an instance of the general Euler-Lagrange problem.

The following theorem summarizes the Euler-Lagrange framework:

Theorem 4.1 [5] *For a dynamical system described by the equations:*

$$\dot{x} = f(x,u,t) \quad x(0) \text{ given}, t_i \le t \le t_f$$

a performance index of the form

$$J = \int_{t_i}^{t_f} L(x,u,t)\, dt$$

and restrictions on the final state

$$G(x(t_f)) = c$$

if the signal u_o achieves an extremum of J, then there exists a vector of constants ζ and a solution to the two point boundary value problem

$$\begin{aligned} \dot{x} &= f(x,u,t) \\ \dot{\lambda} &= -\left(\frac{\partial f}{\partial x}\right)^t \lambda - \left(\frac{\partial L}{\partial x}\right)^t && (4.2) \\ 0 &= \left(\frac{\partial L}{\partial u}\right) + \left(\frac{\partial f}{\partial u}\right)^t \lambda && (4.3) \end{aligned}$$

with boundary conditions:

$$\begin{aligned} x(0) & \quad \text{given} \\ \lambda(t_f) &= \left(\frac{\partial G}{\partial x(t_f)}\right)^t \zeta. \end{aligned}$$

Furthermore, if these conditions are met we will have

$$\lambda(t_i) = \frac{\partial J}{\partial x(0)}. \quad (4.4)$$

This theorem states necessary conditions for a set of signals being a first order extremum of the performance index, in terms of the existence of a solution of an associated adjoint system (Equation 4.2. Furthermore, the solutions of the original and adjoint system verify the constraint equation 4.3. We will see that this constraint can be interpreted as an alignment condition between the inputs of the original nonlinear system and a set of outputs of the adjoint one. This structure will be exploited when we develop the power algorithm.

4.2 Necessary Conditions

To use the Euler-Lagrange necessary conditions in the robust trajectory tracking problem, we have to write the performance index, noise signals, uncertain parameters, and undermodeled dynamical components of the system as constraints compatible with the hypothesis of Theorem 4.1.

Performance index

The performance index is naturally written in the form required by Theorem 4.1. Letting

$$L = \frac{1}{2}y^t y,$$

then optimizing $\|y\|$ is equivalent to optimizing

$$J = \int_{t_i}^{t_f} L\,dt =: \frac{1}{2}\|y\|^2.$$

Noise signals

The only constraints allowed by Theorem 4.1 are in the final values of states. Thus the norm restrictions for the noise signals and the uncertain operator have then to be imposed through final conditions on additional states created for that purpose. We will describe the case for one noise signal only. However, the generalization to several signals is obtained simply by repeating the single signal case. To impose the 2-norm condition on the noise signals, we add to the system a new state named x_u governed by the differential equation

$$\dot{x}_u = \frac{1}{2}u^t u \qquad x_u(t_i) = 0.$$

Then $\|u\| = N$ if and only if $x_u(t_f) = \frac{t_f - t_i}{2}N^2$.

Remark: As was stated in Section 3.2 the equality condition can be changed into an inequality with the introduction of an additional parameter.

Under-modeled dynamical component

The under-modeled dynamical block is characterized by a constraint on the norms of its input signals v and output signals z

$$\|v\| = \|z\|.$$

To impose this equality we add to the system a state x_Δ, governed by the differential equation

$$\dot{x}_\Delta = \frac{1}{2}(z^t z - v^t v) \qquad x_\Delta(t_f) = 0.$$

Then $\|v\| = \|z\|$ if and only if $x_\Delta(t_f) = 0$.

Uncertain parameters and uncertain initial conditions

Optimality with respect to uncertain initial conditions or uncertain parameters will be established through the gradient condition given by Equation (4.4). To treat parameters as initial conditions we create a state that tracks the parameter δ. Let x_δ follow the equation,

$$\dot{x}_\delta = 0 \qquad x_\delta(t_i) = \delta.$$

At a local maximum, either a) the derivative of the performance index with respect to the value of the parameter or the state initial condition is zero, or b) it is negative and the parameter is at the lower end of the interval, or c) it is positive and the parameter is at the higher end of the interval.

Summary

Summarizing, the robust trajectory tracking problem is equivalent to optimizing the performance index

$$J = \int_{t_i}^{t_f} L\,dt = \frac{1}{2}\|y\|^2$$

for the system verifying the differential equation

$$\begin{aligned}
\dot{x} &= f(x,u,v,x_\delta) && \text{(Dynamics)}\\
\dot{x}_u &= \tfrac{1}{2}u^t u && \text{(Noise constraint)}\\
\dot{x}_\Delta &= \tfrac{1}{2}(z^t z - v^t v) && \text{(Uncertainty constraint)}\\
\dot{x}_\delta &= 0, && \text{(Uncertain parameter)}
\end{aligned}$$

where

$$\begin{aligned}
y &= g(x,u,v,x_\delta) && \text{(Performance output)}\\
z &= h(x,u,v,x_\delta) && \text{(Uncertainty output)}
\end{aligned}$$

with given initial conditions

$$x(t_i) = x_0,\ x_u(t_i) = 0,\ x_\Delta(t_i) = 0,\ x_\delta(t_i) = \delta$$

and final conditions

$$x_u(t_f) = \frac{t_f - t_i}{2}N^2,\ x_\Delta(t_f) = 0.$$

This problem is in the form required by Theorem 4.1. So a set of signals u, v, and a parameter δ achieve the worst case value of the performance index J only if there exists λ_x, λ_u, λ_Δ, λ_δ, verifying

$$\begin{aligned}
\dot{\lambda} &= -\left(\tfrac{\partial f}{\partial x}\right)^t \lambda - \left(\tfrac{\partial h}{\partial x}\right)^t z\lambda_\Delta - \left(\tfrac{\partial g}{\partial x}\right)^t y\\
\dot{\lambda}_\delta &= -\left(\tfrac{\partial f}{\partial x_\delta}\right)^t \lambda - \left(\tfrac{\partial h}{\partial x_\delta}\right)^t z\lambda_\Delta - \left(\tfrac{\partial g}{\partial x_\delta}\right)^t y\\
\dot{\lambda}_u &= 0\\
\dot{\lambda}_\Delta &= 0,
\end{aligned} \tag{4.5}$$

with final state conditions

$$\begin{aligned}
\lambda(t_f) &= 0\\
\lambda_\delta(t_f) &= 0,
\end{aligned}$$

verifying the following alignment conditions

$$\begin{aligned}
\left(\tfrac{\partial f}{\partial u}\right)^t \lambda + u\lambda_u + \left(\tfrac{\partial h}{\partial u}\right)^t z\lambda_\Delta + \left(\tfrac{\partial g}{\partial u}\right)^t y &= 0\\
\left(\tfrac{\partial f}{\partial v}\right)^t \lambda + \left(\left(\tfrac{\partial h}{\partial v}\right)^t z - v\right)\lambda_\Delta + \left(\tfrac{\partial g}{\partial v}\right)^t y &= 0,
\end{aligned} \tag{4.6}$$

and such that the initial state verifies

$$\lambda_\delta(t_i) = 0, \text{ or } \begin{cases} \delta = -1 \\ \text{and} \\ \lambda_\delta(t_i) < 0 \end{cases} \text{ or } \begin{cases} \delta = 1 \\ \text{and} \\ \lambda_\delta(t_i) > 0. \end{cases} \tag{4.7}$$

The nature of the performance index chosen, and the fact that none of the functions f, g, or h depend explicitly on the additional states x_u and x_Δ, give Equations (4.5) and (4.6) a particular structure. The states λ_u and λ_Δ will be constants. And the choice of the 2-norm as a performance measure is reflected in the fact that the adjoint system dynamics are driven by the performance output of the system. In the following sections we will make this structure clearer by introducing convenient notation, and we will show how a natural power-like iteration can be derived from it.

Dynamical systems interpretation

For a given trajectory of the original nonlinear system, let $\Lambda = (\lambda, \lambda_\delta)^t$ and define the time varying matrices

$$A(t) = \begin{bmatrix} \left(\frac{\partial f}{\partial x}\right)^t & 0 \\ \left(\frac{\partial f}{\partial x_\delta}\right)^t & 0 \end{bmatrix} \qquad B(t) = \begin{bmatrix} \left(\frac{\partial g}{\partial x}\right)^t & \left(\frac{\partial h}{\partial x}\right)^t \\ \left(\frac{\partial g}{\partial x_\delta}\right)^t & \left(\frac{\partial h}{\partial x_\delta}\right)^t \end{bmatrix}$$

$$C(t) = \begin{bmatrix} \left(\frac{\partial f}{\partial u}\right)^t & 0 \\ \left(\frac{\partial f}{\partial v}\right)^t & 0 \end{bmatrix} \qquad D(t) = \begin{bmatrix} \left(\frac{\partial g}{\partial u}\right)^t & \left(\frac{\partial h}{\partial u}\right)^t \\ \left(\frac{\partial g}{\partial v}\right)^t & \left(\frac{\partial h}{\partial v}\right)^t \end{bmatrix},$$

and consider the dynamical system driven by the equations

$$\begin{aligned} -\dot{\Lambda} &= A\Lambda + B \begin{bmatrix} \nu \\ \xi \end{bmatrix} \\ \begin{bmatrix} -\gamma \\ \kappa \end{bmatrix} &= C\Lambda + D \begin{bmatrix} \nu \\ \xi \end{bmatrix}, \end{aligned} \tag{4.8}$$

with zero final conditions. Then if the signals u and v and the value of the parameter δ achieve a local maximum, there exist constants λ_u and λ_Δ, such that the following alignment conditions are verified between inputs and outputs of the direct and adjoint dynamical systems

$$\begin{aligned} \nu &= y \\ \xi &= \lambda_\Delta z \end{aligned} \tag{4.9}$$

and

$$\begin{aligned} \lambda_u u &= \gamma \\ \lambda_\Delta v &= \kappa. \end{aligned} \tag{4.10}$$

Equation (4.7) states that at an optimum, either the derivative of the performance index with respect to the value of the parameter is zero, or it is negative and the parameter is at the lower end of the interval or it is positive and the parameter is at the higher end of the interval.

This interconnection structure is represented in Figure 4.1. Note that although the two point boundary value problem has both initial and final state conditions, in this case they separate into two sets. The initial conditions are imposed on the states of the original nonlinear system and the final conditions are imposed on the states of the adjoint system. It is this particular structure of the two point boundary value problem that will allow us to develop a power algorithm to solve it.

4.3 A Power Algorithm

Power algorithms are probably the simplest method to find a fixed point of a given function ϕ. Suppose x_o is a fixed point of ϕ, i. e.,

$$x_o = \phi(x_o).$$

A power algorithm to find x_o starts with an initial guess x_1 and updates is according to the rule:

$$x_{k+1} = \phi(x_k)$$

If there exists a neighborhood of x_o, in which x_o is the only fixed point, and where the following condition

$$\|\phi(x) - x_o\| < \|x - x_o\|$$

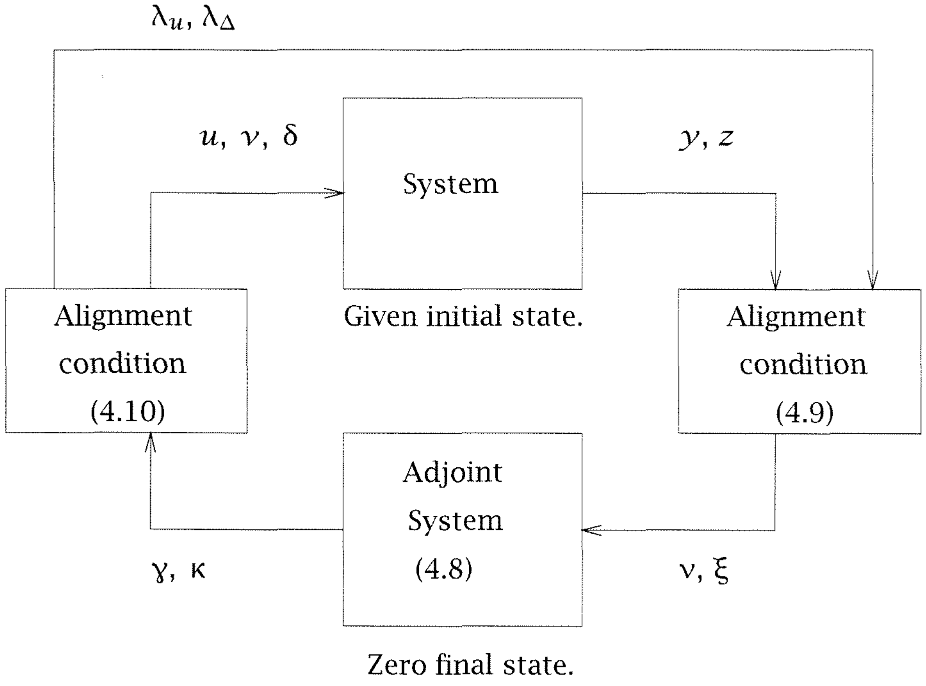

Figure 4.1: Dynamical systems interpretation of the equilibrium conditions.

is verified, then it can be proven that the power algorithm converges to x_o

Power algorithms have been successfully used to compute, among other things, local maxima of nonlinear functions and eigenvalues and singular values of matrices. The main advantage of power algorithms, when compared to other iterative methods, is the simplicity of each iteration. Iterations usually consist of little more than a function evaluation. In the case of eigenvalue computations, iterations usually require simply computing a matrix vector product. Another important characteristic of power methods is their tendency to explore a large region of the search space in their early stages. When applied to computing lower bounds for the linear performance index μ, power algorithms have proven to be fast and reliable. (Some of the implementation details of the power algorithm for the lower bound of μ will be presented in Chapter 8. Other modifications can be found in [20].) The robust trajectory tracking problem is very similar in nature to the structured singular value problem and when applied to linear systems, the robust trajectory tracking problem actually reduces to a special case of μ (see Chapter 7). This motivates the development of a power-like algorithm to compute a lower bound for Problem 3.1. In this section we first give a qualitative description of power algorithms when applied to matrix problems; we then briefly describe the application of the standard μ power algorithm to linear systems over a finite time horizon; finally, we show how these ideas can be extended in an elegant way to the general nonlinear problem.

Heuristics behind power algorithms for linear matrices

Power algorithms are normally used to compute eigenvalues and singular values of matrices. Suppose that the matrix M is diagonalizable, and that its largest eigenvalue is unique. Starting from a random point ν_0, the power algorithm produces a sequence as follows [12]:

$$\begin{aligned} z^{(k)} &:= M\nu^{(k-1)} \\ \lambda^{(k)} &:= \|z^{(k)}\|_2 \\ \nu^{(k)} &:= z^{(k)}/\lambda^{(k)}. \end{aligned}$$

Let x_i be the eigenvectors of the matrix M. If ν_0 has a component along the maximum eigenvalue direction

$$\nu^{(0)} = a_1 x_1 + \sum_{i=2}^{n} a_i x_i$$

then it follows that

$$M^k \nu^{(0)} = a_1 \lambda_1^k \left[x_1 + \sum_{i=2}^{n} \frac{a_i}{a_1} \left(\frac{\lambda_i}{\lambda_1} \right)^k x_j \right].$$

If the terms

$$\frac{\lambda_i}{\lambda_1}$$

are sufficiently small, then the component of $\nu^{(0)}$ along the maximum eigenvector direction is amplified, while the others are contracted, and the procedure converges to the maximum eigenvalue. Since singular values of M are eigenvectors of M^*M, the same procedure can be used to compute the maximum singular value. In this case we will have to perform alternating power steps by M and M^*.

Power type algorithms are naturally suited for searching for directions of maximum gain. This fact makes them appealing for the computation of performance measures in disturbance rejection settings. Computing the structured singular value, for example, implies computing the largest of the spectral radii of the matrices in a set. There are two search directions: one within the set of matrices to find the one that has the largest spectral radius, and one for a given matrix to find its eigen-direction of maximum gain. The power algorithm for μ does both searches simultaneously. From a qualitative point of view, the power steps, alternating multiplications by M and M^*, help look for directions of maximum gain of the particular matrix under study. The alignment conditions forced between power steps perform the search in the matrix set. A discrete time linear system, considered over a finite time horizon, can be represented by the matrix of the map between inputs and outputs. Many robust performance questions asked of this type of systems can be solved by computing the structured singular value of the map matrix with respect to an adequate structure (see Chapter 7). It is interesting to note that in this case the power step with respect to M is taken by integrating the difference equations of the system forwards in time, and the power step by M^* is computed by simulating an adjoint dynamical system backwards in time. The succession of simulations of linear systems amplifies the component the chosen input has along the direction of maximum gain, while the operations carried out in between power steps look for the system in the class with maximum gain.

Conceptually, all the operations necessary to carry out this procedure are still well defined when the system is nonlinear. The substitutions needed to convert one case into the other are natural. Multiplication by the system matrix is equivalent to integration of the equations of motion. Multiplication by the adjoint matrix is equivalent to integration of the transposed linearized system. Although numerically harder, the operations are not conceptually different than in the linear case. And in principle, the heuristics explaining why this type of algorithm is efficient for finding directions of maximum gain still hold even when the system is nonlinear. In the next section we will prove that these substitutions are the appropriate ones, and from them we will develop a power algorithm to solve the nonlinear robust trajectory tracking problem. In Chapter 5 we will present numerical evidence that this heuristic justification is valid.

Power algorithm for nonlinear systems

Earlier in this section we showed how the first order necessary conditions for robust performance could be interpreted as an interconnection of dynamical systems.

Four maps can be identified in the diagram in Figure 4.1. The first map integrates the equations of motion of the original system, with the given initial conditions and for a set of input signals u, v. The constants λ_u and λ_Δ are not changed. (These are included in the definition of the map to simplify the final expression.)

$$\Phi : (u(t), v(t), \delta, \lambda_u, \lambda_\Delta) \mapsto (y(t), z(t), \delta, \lambda_u, \lambda_\Delta).$$

The position of the map Φ in the system interconnection corresponding to the necessary conditions is shown in Figure 4.2. The second map generates the input signals for the adjoint system by forcing the alignment conditions in Equation (4.9).

$$\Pi : (y(t), z(t), \delta, \lambda_u, \lambda_\Delta) \mapsto (y(t), \lambda_\Delta z(t), \lambda_u, \lambda_\Delta).$$

Figure 4.3 indicates the position of this map in the general interconnection. The third map in the sequence integrates the equations for the adjoint system along the current trajectory with the given inputs.

$$\Psi : (\nu, \xi, \lambda_u, \lambda_\Delta) \mapsto (\gamma, \kappa, \lambda_u, \lambda_\Delta).$$

The final map we define computes new inputs for the original system by using the alignment condition in Equation (4.10), and also computes new values for δ, λ_u, λ_Δ. There is more than one way of carrying out this evaluation. We propose here one possibility

$$\begin{aligned}
\Theta &: (\gamma, \kappa, \lambda_u, \lambda_\Delta) \mapsto (u(t), v(t), \delta, \lambda_u, \lambda_\Delta) \\
\lambda_u &:= \frac{\|\gamma\|}{U_e} \frac{\gamma \cdot u}{\|\gamma\| U_e} \\
u &:= \frac{\gamma}{\lambda_u} \\
\lambda_\Delta &:= \frac{\|\kappa\|}{\|z\|} \frac{\kappa \cdot v}{\|\kappa\| \|v\|} \\
v &:= \frac{\kappa}{\lambda_\Delta}.
\end{aligned}$$

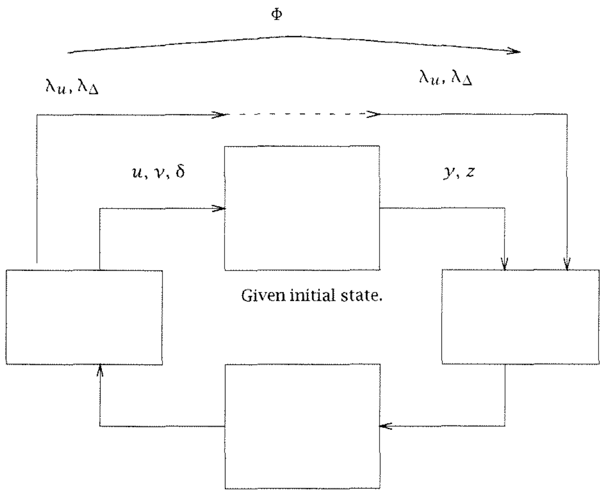

Figure 4.2: Φ operator: Integration of system equations.

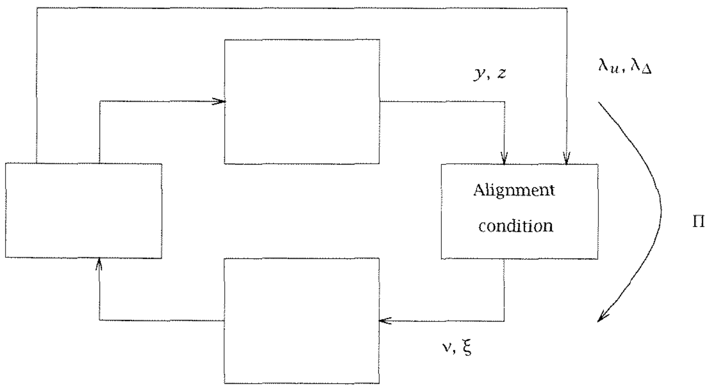

Figure 4.3: Π operator: Forcing the alignment conditions.

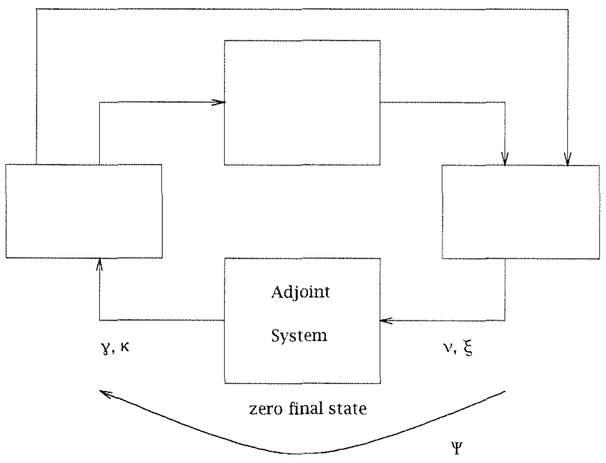

Figure 4.4: Ψ operator: Integration of adjoint system equations.

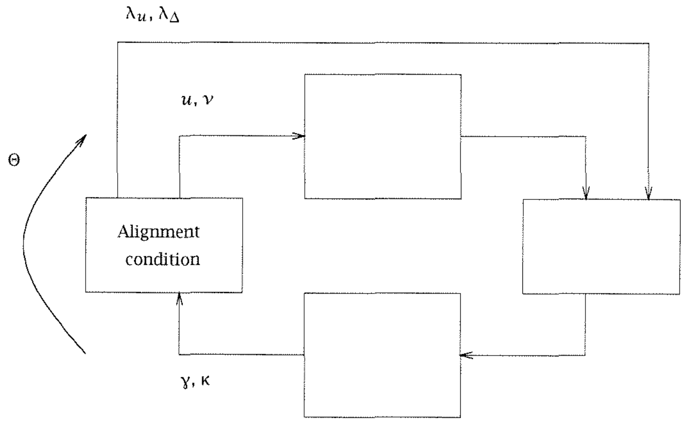

Figure 4.5: Θ operator: Forcing alignment conditions.

and δ is computed as follows:

$$\begin{aligned} \chi &:= \delta + \lambda_\delta(t_i) \\ \delta &:= \begin{cases} -1 & \chi < -1 \\ \chi & -1 \le \chi \le 1 \\ 1 & \chi > 1. \end{cases} \end{aligned}$$

Figures 4.4 and 4.5 show these maps relative to the necessary conditions interconnection. The mappings Φ and Ψ are, as described in the previous section, the nonlinear equivalent of power steps or matrix vector multiplications. The operations carried out in the mappings Π and Θ impose the alignment conditions for optimality. The diagram in Figure 4.1 commutes when the signals u and v and the constants δ, λ_u, and λ_Δ are the ones corresponding to an extremum point. In other words, by going once around the diagram following the maps Φ, Π, Ψ, and Θ, we should return to the starting point. This fact is formalized in the following

Lemma 4.2 *The signals u and v, and the parameter δ, verify the optimality conditions if and only if there exists λ_u, λ_Δ such that*

$$(u, v, \delta, \lambda_u, \lambda_\Delta)$$

is a fixed point of the composition $\Theta \circ \Psi \circ \Pi \circ \Phi$.

We can now use an iterative algorithm to search for the fixed points of this composition. The standard form of such an algorithm is given by the pseudo-code in Table 4.1.

```
$x^{(1)}$:=random;
repeat
        $x^{(i+1)} := \Theta \circ \Psi \circ \Pi \circ \Phi(x^{(i)})$;
until ($|x^{(i+1)} - x^{(i)}| < \epsilon |x^{(i)}|$)
```

Table 4.1: Power algorithm pseudo-code.

If the algorithm converges, it converges to a fixed point of the composition $\Theta \circ \Psi \circ \Pi \circ \Phi$ and thus, according to Lemma 4.2, to a set of signals that meet the necessary conditions for a critical point.

Remarks: In order to prove convergence, we would have to prove that the composition $\Theta \circ \Psi \circ \Pi \circ \Phi$ is a contraction around local maxima. We can only prove this under very limiting conditions. The standard power algorithm for the lower bound on μ has not been proven to be stable for a general uncertainty structure, and is in fact known to be unstable in some cases (see Chapter 8 for a discussion). Hence the evaluation of the algorithm will have to be done empirically.

The described algorithm has many of the characteristics of the power algorithm for linear systems. In particular, if the system is linear, the adjoint system is linear time invariant, and in this case the algorithm reduces to the standard power iteration alternating multiplication by M and M^*.

4.4 Numerical Implementation

There is more than one set of maps Φ, Π, Ψ, and Θ that can be constructed from the equilibrium equations and from which we can derive a power algorithm. Furthermore, there are several ways in which those mappings can be implemented numerically. We will discuss in this section some of the issues arising in this implementation, and some of the possible variations.

Time axis discretization

The maps Φ, Π, Ψ, and Θ operate on infinite dimensional spaces. A digital computer implementation of the given algorithm will thus have to restrict these to finite dimensional spaces by discretizing the time axis.

There are several approaches to obtaining a discretization of the time axis. It is important to allow for a non uniform time scale, since the numerical sensitivity of the nonlinear system to discretization error can vary drastically along a trajectory. Most numerical integrators of differential equations will, however, create a partition of the time scale that minimizes the integration error. We will use then the set of time stops provided by the integration routine as our discretization. The discretization can be determined at the first iteration, or it can be updated every iteration (after every system simulation). The first approach will reduce the number of operations, since in this case it is not necessary to interpolate the old signals at the new time instants to verify convergence. However, if the disturbances are significant, the trajectories may deviate from the nominal making a new time axis necessary.

By discretizing the time axis we lose freedom on the intersample behavior. We will be carrying out the maximization of the performance index over a reduced signal set. As before, knowledge of the behavior of the system is needed to ensure that this reduction is not significant. We have two ways of controlling the intersample behavior. Most integration algorithms accept as parameters maximum and minimum time steps. A good understanding of the system will tell us what is the fastest rate of variation in the input signals that can possibly affect its behavior. By specifying the maximum and minimum allowable time steps, we can guarantee that such signals fall within the search set. We also have a certain degree of control over the intersample behavior. We can use a hold of any given order to interpolate the value of the signals in between time samples. The choice of hold order will be given by the tradeoff between accuracy and computation effort.

Alternative formulations

The iteration presented is not the only one possible. In particular it is not necessary to update the adjoint system every iteration. The adjoint can be computed once every n iterations for example. The effect of this modification depends on the characteristics of the nonlinear dynamics, the size of the disturbances and the particular trajectory under consideration. It could in principle be harmful or beneficial. Since the behavior of nonlinear system is a lot more diverse than in the linear case, it is important to tailor the analysis tools to the particular problem at hand. An advantage of the power algorithm is that due to its simplicity, it easily accepts modifications that adapt it to the application being considered.

Computation time growth

An important characteristic of the iteration in the power algorithm is that the number of operations necessary grows slowly with the size of the problem. There are two dimensions in the size of a problem. The length of the time horizon, and the order of the nonlinear system (where order measures number of states, number of inputs and outputs, and number of parameters).

The number of operations necessary to evaluate the alignment conditions in the maps Θ and Π grows linearly with the number of subdivisions in the time axis. The growth of simulation time with the length of the horizon will depend on the type of system, and on the integration algorithm, but will in general also be linear (although longer simulation periods may reduce their accuracy). Numerical evaluation of the adjoint system also requires a constant number of operations per time sample, so the growth will be linear.

Computation time growth with respect to the system order is perhaps more important. Most sophisticated optimization algorithms such as NPSOL need very expensive computations when measured with respect to system size. Although they may be able to guarantee faster convergence rates, computation time can very quickly become prohibitive. The most expensive computation in the power algorithm is the evaluation of the adjoint system. Jacobian computation in general requires a number of function evaluations that grows with the square of the order of the problem. However, in the case of the power algorithm, we only need to compute the Jacobians along known directions. In Equation (4.8) we only need to compute the matrix vector product and not the matrices themselves. The number of function evaluations remains only linear. (The number of operations needed to do these evaluations depends on the nature on the nonlinearities. If the original system were actually linear, evaluation of the Jacobian would require the dimension squared operations.) The growth of the computation effort needed to integrate the equation depends once again on the nature of the system, but will in general grow quadratically with the size of the problem. Finally, computation of the alignment conditions grows only linearly with the number of constraints.

Another aspect of the measure in computation time is the number of iterations needed to converge to a solution. We have no systematic way of establishing this, although it is an important characteristic of the algorithm. The numerical results we have obtained, however, suggest that the number of iterations needed by the power algorithm compares very favorably with other more classical approaches. This was also observed in the linear case.

Stabilization

As we stated earlier, we have not proven that the composition of maps is a contraction, and thus that the algorithm is stable at local maximums. It is possible, however, to augment the stability of the algorithm by averaging the signals over several iterations. For example, for lag 2 averaging, the iteration step will be

$$x^{(i+1)} := .5(\Theta \circ \Psi \circ \Pi \circ \Phi(x^{(i)}) + x^{(i)}).$$

There are several possible ways of deriving a set of maps whose composition has as fixed points signals verifying the necessary conditions. For example, when deriving the map Θ, we could update v and λ_Δ simultaneously. Since we know the norm of v and the value of $\left(\frac{\partial h}{\partial v}\right)^t z$, we can determine v and λ_Δ by intersecting the line passing through the origin and with direction $\left(\left(\frac{\partial h}{\partial v}\right)^t z - v\right)$ with the circle centered at $\left(\frac{\partial h}{\partial v}\right)^t z$ and with radius $\|v\|$. This approach requires more operations (although the number of operations still grows linearly with the number of time samples). It is possible that different approaches have better convergence properties in some cases and worse in others. Since it is simple to implement all the versions, the choice of algorithm can be made an option of the function call.

Chapter 5
Numerical Examples

Given the wide diversity in the behavior of nonlinear systems, it is fair to say that no one optimization algorithm will always work. This fact makes comparison between different algorithms hard. It also makes it difficult to prove that any given algorithm is suitable for a general purpose. Evaluation of the performance of the algorithms has to be done on a case by case basis. Nonlinear systems, however, can still be classified in families of problems. All aircraft models, for example, are very similar, and variations in the possible behaviors are more of a quantitative than a qualitative nature. Evaluation of the algorithm can be done on test cases for classes of problems. These examples have to be chosen so as to contain all the important characteristics of their class.

In this chapter we present the results obtained from trying the power algorithm developed in Chapter 4 on two different systems for several different disturbance and performance specifications. The systems are nonlinear, parts of the models are obtained from first principles (equations of motion of rigid bodies), and parts of the models are obtained through measurements and implemented with look-up tables. The models include command and rate saturations. These systems, and the performance problems we set up for them, have many of the characteristics of typical aircraft (or other types of vehicles) applications.

The results obtained are compared with different alternative approaches to solving the problems. These results indicate the viability of the algorithm. This chapter will be divided in two sections, corresponding to each of the systems analyzed. A brief description of the dynamics of the system is included at the beginning of each section, followed by the different problems solved and the results obtained.

5.1 The Caltech Ducted Fan

The first example we will treat tests the noise rejection capabilities of a full state feedback LQR design for an experimental ducted fan platform (Figure 5.1).

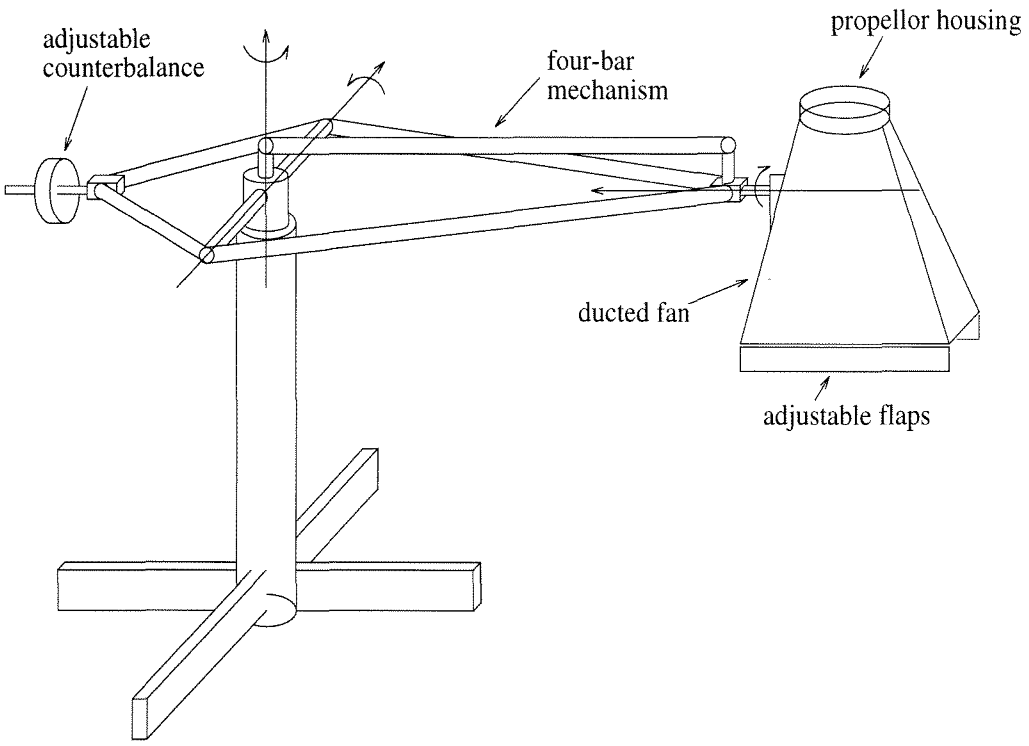

Figure 5.1: Overview of the experimental setup.

The experimental vehicle consists of a relatively simple ducted fan aircraft that can provide two dimensional vectored and reverse thrust. The aircraft is bolted to a rotating arm, which limits its motion to three degrees of freedom: one rotational and two translational, approximately on the surface of a sphere defined by the arm. With this geometry, the ducted fan is completely controllable with just the vectored thrust. The ducted fan itself consists of a wooden duct powered by a variable speed electric motor driving a propeller. A detachable flap assembly, mounted at the end of the duct gives the fan vectored and reverse thrust. The setup and its nonlinear model are described in more detail in [6].

Step in height, simple performance specification

We consider a controller whose objective is to make the vehicle track a step in vertical position, i. e., we want to move it from hover at zero altitude to hover at another pre-specified altitude. The performance measure for the disturbance rejection problem will be the 2-norm of the distance from the nominal to the actual trajectory in the y position of the vehicle from beginning to end of the maneuver. A time domain weight multiplies the error signal to capture the fact that we consider the error at the final time to be more important than at the beginning. We will have three disturbances entering as torques along the three axes of rotation. The 2-norm of each of this noise signals is set to be 0.006 Nm over the 9 second time horizon (i. e., the rms value of the torque is 0.006 Nm). The uncertain block tries to capture the unmodeled dynamics of the electrical motor that is driving the fan. The input of the uncertain block is the command signal given to the motor filter through a high pass filter. The uncertain parameter represents the application point of the reaction force on the vehicle, since this point changes with, among other factors the position of the flaps. This distance is considered to be $0.25\text{m} + \delta$ with $|\delta| < 0.1\text{m}$. A Simulink schematic diagram of the simulation model is given in Figure 5.2.

In Figure 5.3 we show the evolution of the error between the nominal and the actual trajectory starting from a random noise signal. Figure 5.4 shows the last two iterations of the same error signal.

In Figures 5.5 and 5.6 we show the disturbance signal and the square of the input and output to the uncertain block respectively.

For this problem we also study the stabilizing effect of averaging the signals from two consecutive iterations as discussed in Section 4.4. We ran the algorithm 30 times starting at different random points, both with and without averaging. Figures 5.7 and 5.8 show the corresponding frequency distribution of results. We can see that the filtering skewed the distribution towards the largest value. We can also see from these figures that a problem can have more than one stable point and it will be necessary in general to run the algorithm more than once. This is, however, common to all algorithms based on local search.

To verify the results we used the classic optimization package NPSOL

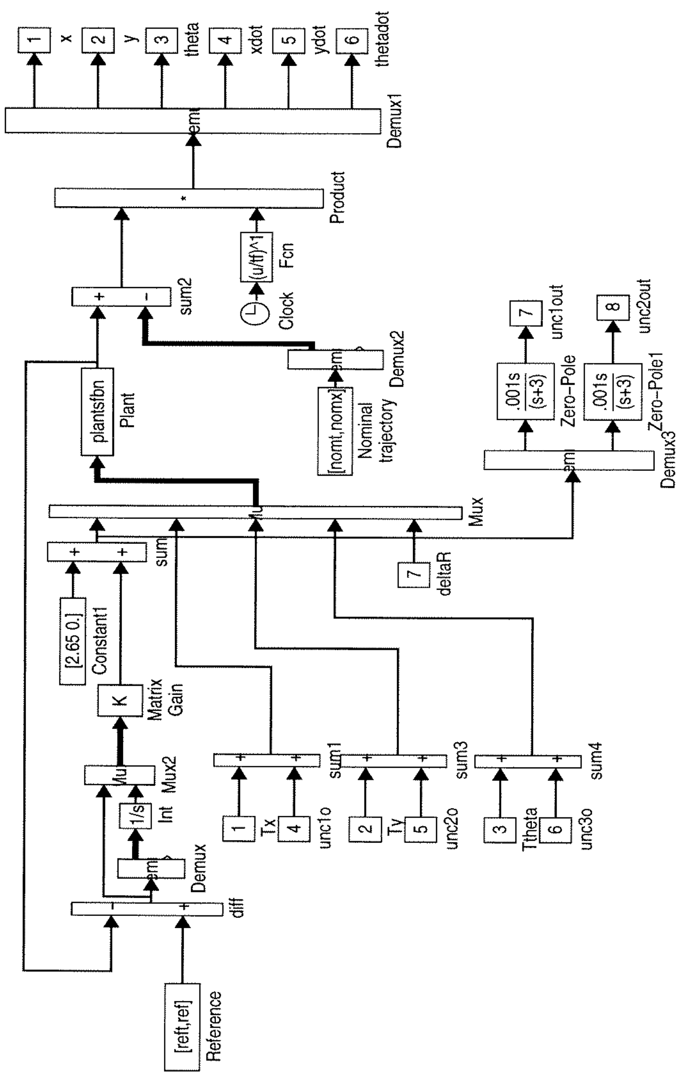

Figure 5.2: Simulink diagram for the ducted fan experiments.

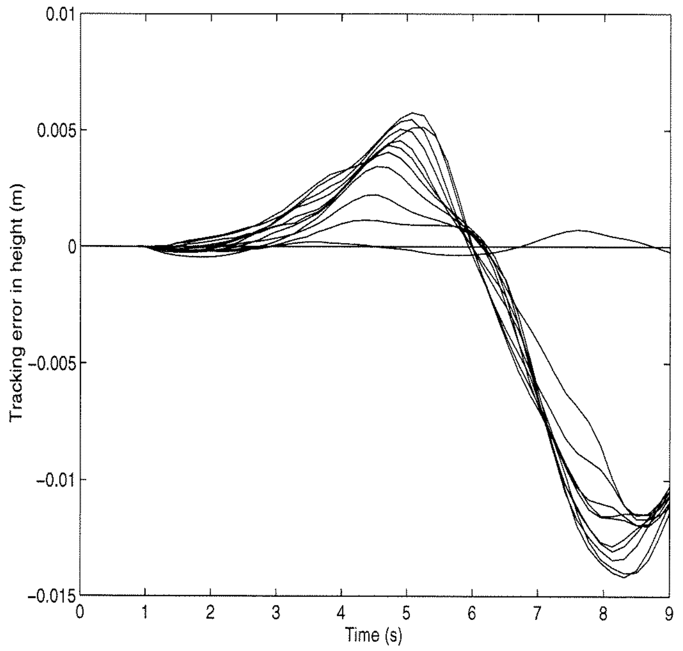

Figure 5.3: Evolution of the worst case trajectory.

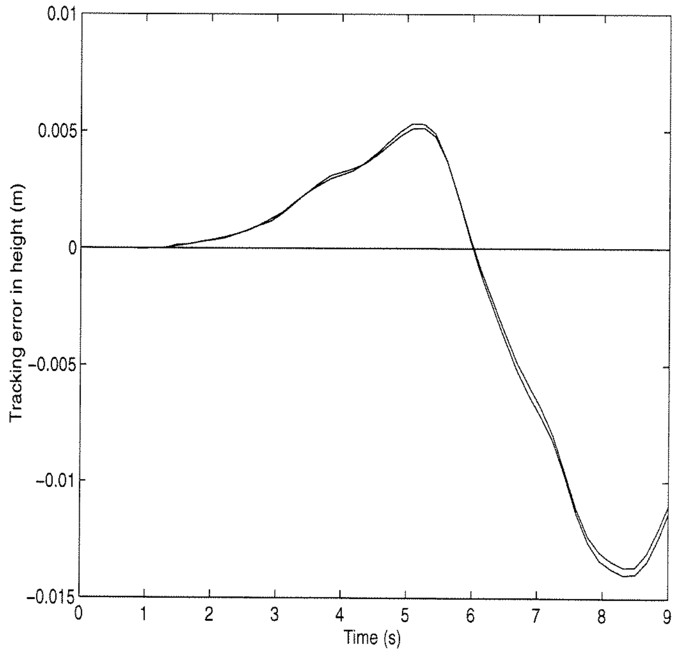

Figure 5.4: Trajectory for the last two iterations of the algorithm.

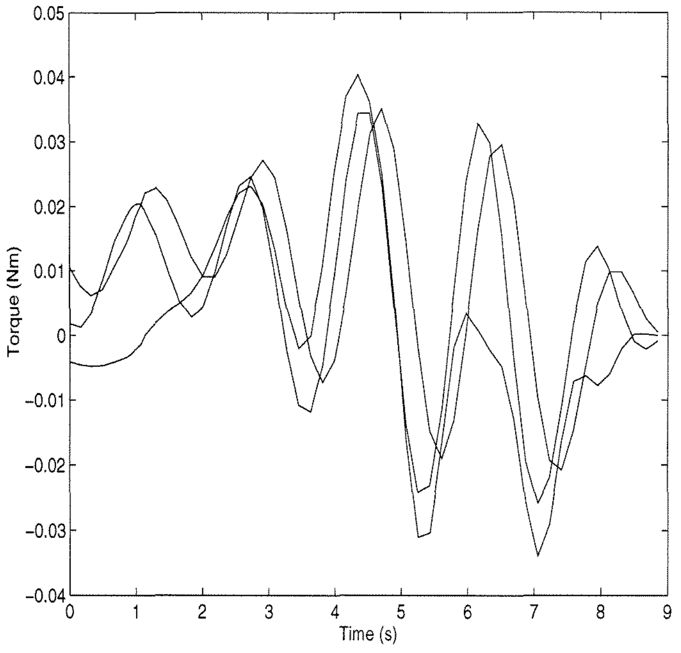

Figure 5.5: Final noise signals.

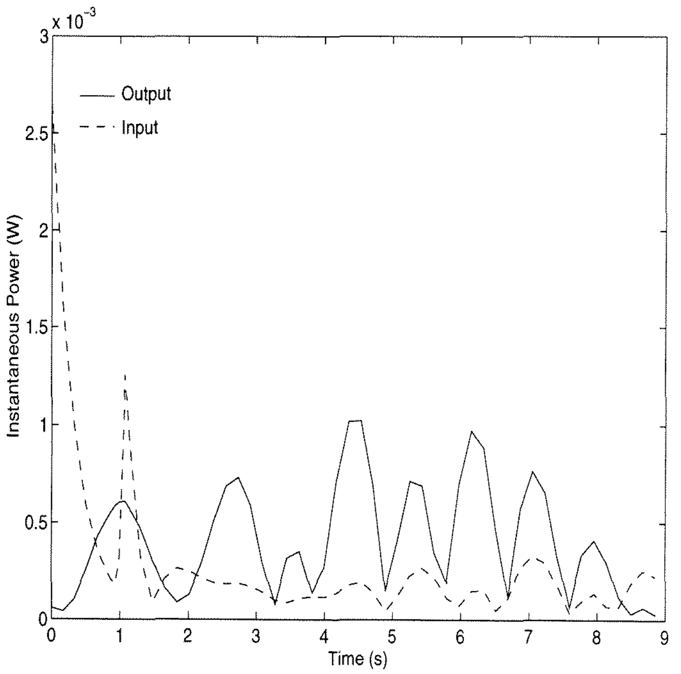

Figure 5.6: Power of final inputs and outputs to the uncertain block.

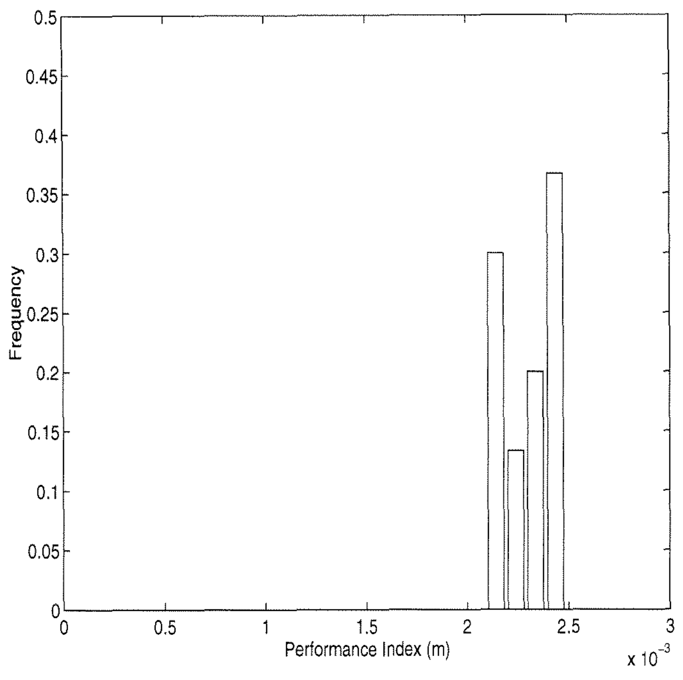

Figure 5.7: Distribution of values with averaging.

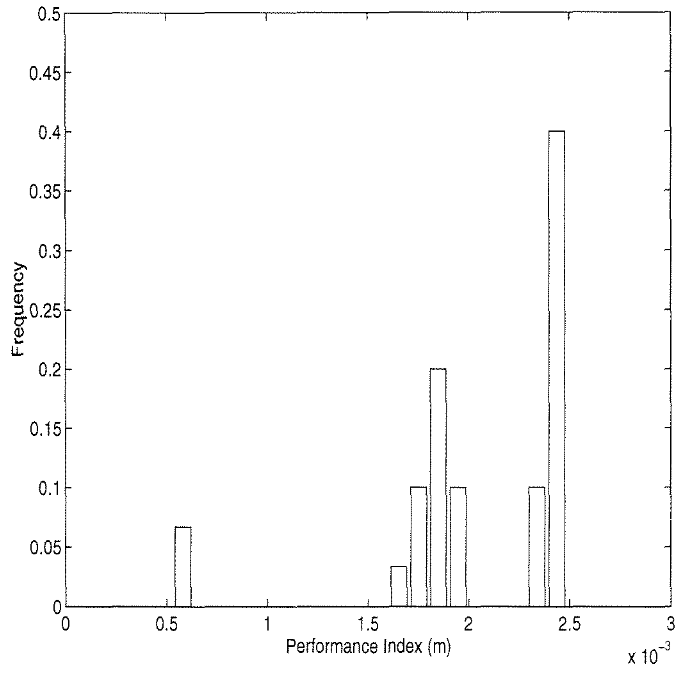

Figure 5.8: Distribution of values without averaging.

[11]. NPSOL is a very general optimization package for nonlinear programming. It can tackle any kind of constrained and unconstrained nonlinear program. In this particular problem, however, NPSOL proved to be very sensitive to the initial guess. This is due to the feedback nature of the undermodeled dynamical component. A small violation of the norm constraint corresponding to this component can have a significant effect on the overall result. The behavior of NPSOL on this particular problem was in general poor. In most runs it failed to find a feasible solution. When it found one, the running time was at least three times longer, even though most of the components in NPSOL were compiled, when our code was mostly in Matlab scripts. The next example was compared to another optimization algorithm. As we will see in the following section, the comparison also favors the power algorithm.

Step in height, full performance specification

The maneuver specifies that we start at hover, i. e., with zero velocity in all directions, and end at hover. Furthermore, it is not only important to reach the desired height; we also want to keep the x position constant. In order to impose these restrictions we will use as performance signals the error in all the six state variables for the vehicle. (These are the mechanical states. We are not considering the states added by the controller and the different filters.) All other specifications remained the same.

After 30 iterations the algorithm only converged to within 2 percent. We believe that this is due to the fact that there exists a subspace of signals for which the value of the optimization index is fairly constant. The algorithm then rotates in this subspace. The signals change, but the optimization index does not. A similar situation can occur when using power algorithms to compute the maximum eigenvalue of a matrix. If the corresponding eigenspace is not of dimension 1, the algorithm will converge to the value of the spectral radius, but not necessarily to one eigenvector.

It is very important to point out that for most of the iterations, the system is simulated with signals that meet the constraints. (This is due to the way signals are updated.) If the signals don't converge, we can choose among the simulations done with signals that meet the constraints, the one that yielded the largest value of the optimization index. In this case the value reported was $J = 0.411$.

We compared our algorithm in this case with the one developed in [26]. Each iteration of this algorithm is more expensive than the one carried out by the power algorithm. After 500 iterations the algorithm in [26] reported a value for the local maximum of $J = 0.431$. After 30 iterations it reported a value of $J = 0.165$. The comparison between power type and gradient based algorithms shown here is very similar to the one observed in the linear case [20]. The latter kind of algorithms requires a much larger number of more expensive iterations to give a similar answer than the former one. Figures 5.9 and 5.10 compare the disturbance signals obtained by the gradient and power algorithms respectively.

5.2 F-16 Maneuvers

We want to determine whether the algorithm is suitable for aerospace applications. As a first step, the algorithm's ability to handle a model that includes a number of nonlinear equations and tabular data with a relatively high number of parameters, all characteristic of a typical aircraft, must be ascertained. In this section we study different performance problems for a set of maneuvers using a model for an F-16 fighter. We will first briefly review the basic ideas and the notation of aircraft dynamics. Then we will describe the different maneuvers analyzed and the results obtained.

Aircraft dynamics

In order to make the maneuver descriptions in the next section more understandable, we will include here a brief description of aircraft dynamics. There are several classical references on this subject. The material in this section is largely based on the treatment in [16]. All vector quantities, forces, momentums, velocities, and accelerations will be referred to a vehicle fixed axis of reference as shown in Figure 5.11. To each axis corresponds a linear velocity, an angular velocity, a force, and a momentum. Table 5.1 summarizes the notation we will be using for each of these terms.

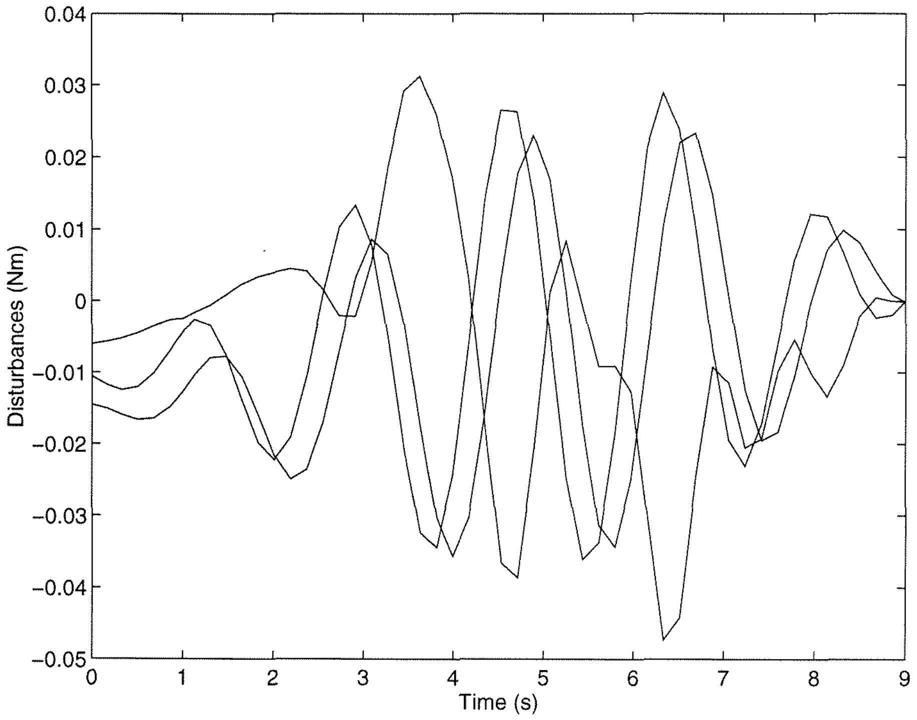

Figure 5.9: Worst case signals: gradient algorithm.

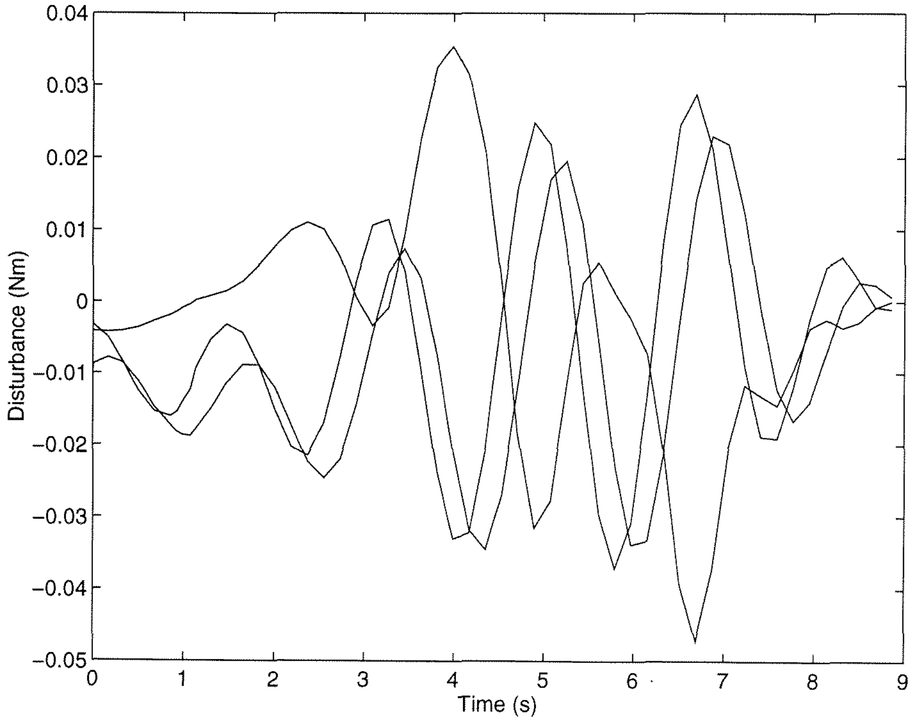

Figure 5.10: Worst case signals: power algorithm.

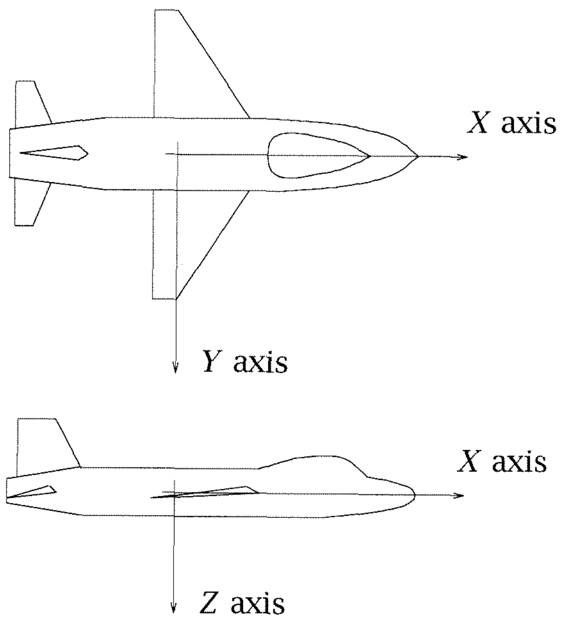

Figure 5.11: Aircraft-fixed axis system.

Axis	Linear Velocity	Angular Velocity	Force	Momentum
X-axis	U	P	F_X	L
Y-axis	V	Q	F_Y	M
Z-axis	W	R	F_Z	N

Table 5.1: Velocities, forces and moments.

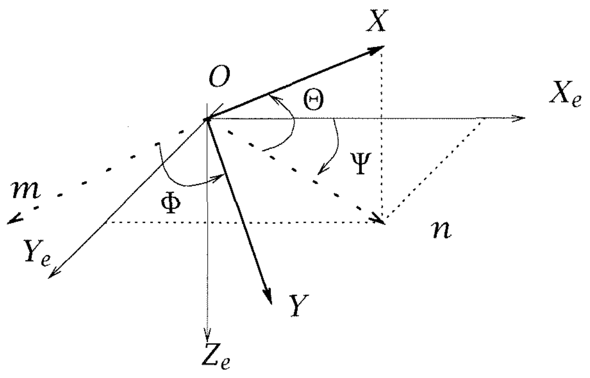

Figure 5.12: Euler angle transformation from earth fixed to body fixed systems. To transform $OX_eY_eZ_e$ to $OXYZ$, first rotate by Ψ around OZ_e(OX_e becomes On and OY_e becomes Om), then rotate by Θ around Om, and finally rotate by Φ around OX.

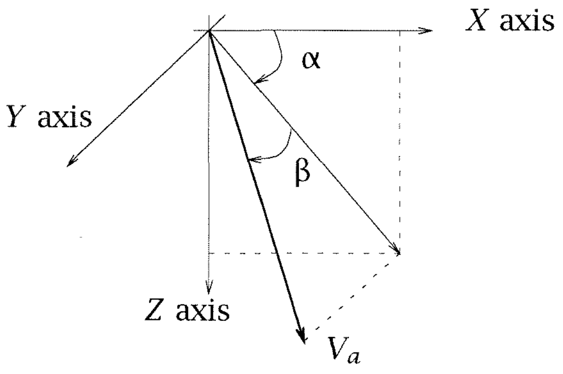

Figure 5.13: Orientation of relative wind in body fixed axis.

We will also need to refer the vehicle fixed reference frame to a ground based one. For the treatment here we will assume that a ground-fixed reference is inertial. If needed, proper modifications can be made to account for the Earth's rotation. We will define the Euler angles Θ, Φ, and Ψ as shown in Figure 5.12. To compute the aerodynamic forces and moments on the airplane, we need to know the direction of the wind with respect to the vehicle fixed reference frame. We will denote by $-V_a$ the velocity of wind with respect to the vehicle. The angle β formed by the vector V_a and the plane XoZ is called sideslip. The angle α formed by the projection of V_a on the plane XoZ and the axis oX is called angle of attack. (See Figure 5.13.)

It can be shown that the aerodynamic forces and moments will be approximately of the form

$$F = \frac{1}{2}\rho S V_a^2 C_F,$$

where C_F is a dimensionless coefficient, ρ, the density of air, V_a, the velocity of the vehicle with respect to air, and S a characteristic surface. We will then have

$$\begin{array}{lcll}
F_{Xa} & = & \frac{1}{2}\rho S V_a^2 C_x & \text{(Aero. force along X-axis)} \\
F_{Ya} & = & \frac{1}{2}\rho S V_a^2 C_y & \text{(Aero. force along Y-axis)} \\
F_{Za} & = & \frac{1}{2}\rho S V_a^2 C_z & \text{(Aero. force along Z-axis)} \\
La & = & \frac{1}{2}\rho S V_a^2 C_l & \text{(Rolling moment)} \\
Ma & = & \frac{1}{2}\rho S V_a^2 C_m & \text{(Pitching moment)} \\
Na & = & \frac{1}{2}\rho S V_a^2 C_n. & \text{(Yawing moment)}
\end{array} \tag{5.1}$$

The coefficients C_F in these formulas depend on the angle of attack, sideslip, position of the different control surfaces of the airplane and, to a lesser scale, on the linear and angular velocities of the airplane. We will consider only three control surfaces: aileron, rudder, and elevator. The other actuator we will use, throttle control of the engine, is not of an aerodynamic nature. Although the positions of the control surfaces affect all of the coefficients C_F defined above, the airplane is designed so that the rudder influences mainly C_n, aileron position affects mainly C_l, and elevator position affects mainly C_m.

Two other forces will appear in the dynamic equations for the airplane: gravity and the thrust provided by the engine. Thrust will be a function of altitude, air density, air speed, and throttle position. How the different factors, affect thrust depends on the nature of the engine (propeller, turboprop, or jet engine). Denoting by

$$(x_{Cg}, y_{Cg}, z_{Cg})$$

the position of the center of mass of the airplane in the $oXYZ$ frame, the equations of motion become

$$\begin{array}{lclcl}
ma_{x_{Cg}} & = & m[\dot{U} + QW - RV] & = & F_{Xa} - mg\sin\Theta + T \\
ma_{y_{Cg}} & = & m[\dot{V} + RU - PW] & = & F_{Y_a} + mg\cos\Theta\sin\Phi \\
ma_{z_{Cg}} & = & m[\dot{W} + PV - QU] & = & F_{Z_a} + mg\cos\Theta\cos\Phi,
\end{array} \tag{5.2}$$

and

$$\begin{aligned}
\dot{P}I_x - \dot{R}I_{xz} + QR(I_x - I_y) - PQIxz &= L_a \\
\dot{Q}I_y + PR(I_x - I_z) - R^2 I_{xz} + P^2 I_{xz} &= M_a \\
\dot{R}I_z - \dot{P}I_{xz} + PQ(I_y - I_x) + QRI_{xz} &= N_a,
\end{aligned} \tag{5.3}$$

where T is the thrust provided by the engine and I_{ij} are the inertia coefficients, and g is the gravitational constant.. Note that the rates P, Q, and R and the derivatives $\dot{\Phi}$, $\dot{\Psi}$, and $\dot{\Theta}$ are not independent.

Although Equations (5.2) and (5.3) are complicated, it is possible to obtain simplifications by making assumptions about the trajectory being followed. It can be seen that, in general, lateral and longitudinal motions are fairly independent when considering perturbations around constant altitude and constant heading motion, for example.

Steady maneuvers

Several steady state maneuvers can be considered, e. g., constant altitude, constant forward speed flight. In order to sustain this steady state, we have to determine the position of the control surfaces that will provide the necessary forces and moments. In the example mentioned, the steady state is characterized by

$$\begin{aligned}
F_Y &= 0 && \text{(Lateral acceleration)} \\
\Phi &= 0 && \text{(No roll)} \\
\Psi &= 0 && \text{(No yaw)} \\
F_Z \cos\Theta + F_X \sin\Theta &= 0 && \text{(Constant altitude)} \\
\dot{\Theta} &= 0 && \text{(Constant pitch)} \\
-F_Z \sin\Theta + F_X \cos\Theta &= 0. && \text{(Constant forward speed)}
\end{aligned} \tag{5.4}$$

Combining Equations (5.2), (5.3), and (5.4) plus the equations describing the aerodynamic forces and momentums, we can derive the position of the different control surfaces. We will refer to these values as the trim condition necessary for steady flight. In general the set of equations obtained is nonlinear, and has to be solved numerically.

While planning maneuvers for aircraft it is often desirable to maintain the sideslip angle β equal to zero. This is particularly important while turning. (It guarantees that all accelerations are going to stay in the oXZ

plane of the airplane.) A turn with zero sideslip is called a coordinated turn. Trim conditions for a coordinated turn are obtained following the same procedure described above.

In the sections that follow, we will analyze the behavior of the airplane when trimmed for different steady maneuvers and perturbed by atmospheric disturbances and uncertainty in different parameters of operation.

Model used

The aircraft used in this example application is an F16. The aerodynamic model is a reduced version of the full model obtained in wind tunnel tests at NASA Langley in 1979 [28]. It consists of tabular data with typical interpolation routines and nonlinear equations of motion. The engine model is that of an after-burning turbofan. The airplane model utilized in this application is defined for speeds ranging of up to Mach 0.6 and angle of attack interval between -10 and 45 degrees. The model includes 4 traditional controls (elevator, aileron, rudder, and throttle) and 13 states (velocity vector, attitude angles, angular velocities, navigational position, altitude, and engine power). Furthermore, the aerodynamic coefficients are built up in a traditional way and the equations of motion are full nonlinear flat Earth equations.

Constant height coordinated turn

The first trajectory we will analyze for this system is a constant altitude, constant rate of turn (or "constant g") coordinated turn. The x-coordinate of the nominal effective center of gravity (Cg) location is set at $x_{Cg} = 0.2\overline{c}$, where $\overline{c}$ is the maximum aerodynamic chord length of the wing. This choice makes the aircraft statically stable. The aircraft initiates the maneuver at 10,000 ft flying at 500 ft/s, and performing a $4.5g$ turn. The aircraft will be trimmed for this maneuver. Note, however, that there will be no feedback loop correcting these values when the nominal system is perturbed.

During the maneuver the aircraft is subjected to atmospheric turbulence in vertical, horizontal, and lateral directions modeled by Von Karman spectra and implemented by Dryden filters [1]. The 2-norm of the noise used is such that a constant input with the given norm will produce at the output of the filters a wind of 44 ft/s in the X direction and .4 ft/s in the Y and Z directions. The position of the center of gravity Cg will be an uncertain parameter of the system. The actual location of the center of gravity is going to be $0.1995\bar{c} \leq x_{Cg} \leq 0.2005\bar{c}$. The performance variable will be turn radius of the trajectory, which we would like to remain constant. The values of the disturbances and uncertain parameters are relatively mild, except for the wind disturbance in the X direction.

The algorithm converges after 5 iterations to within 0.1 percent (i. e., the total relative variation of all the signals in the system between the last two iterations was less than 0.1 percent). It converges to a value of $Cg = .2005\bar{c}$ and the 2-norm of the error signal is 76 ft. Figure 5.14 shows the nominal and perturbed trajectories.

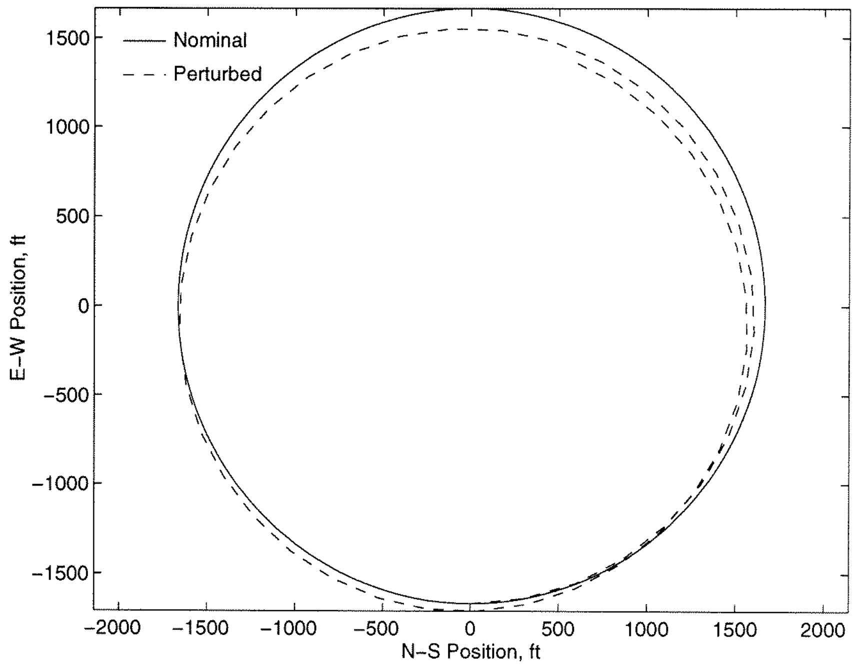

Figure 5.14: Nominal and perturbed trajectories.

In order to validate the results obtained, we will compare with the standard Monte Carlo analysis technique. For each of five possible values of the parameter, we run 100 simulations with randomly chosen noise signals. In Figure 5.15 we present the results of all 500 iterations, and we compare the error obtained at each of them with the worst case value provided by our algorithm. In Figure 5.16 we present a histogram of the values obtained from the 500 iterations, and again compare those results with the worst case value.

Constant rate of climb coordinated turn

The trajectory is a constant climb, constant g coordinated turn. The aircraft initiates the maneuver at 10,000 ft flying at 500 ft/s. The F16 is then trimmed to climb at 50 ft/s while maintaining a 4.5 g coordinated turn. We study the maneuver over a 30s interval. Figure 5.18 illustrates the nominal trajectory. (All other nominal conditions are identical to the ones in the previous example.)

During the maneuver the aircraft is subjected to atmospheric turbulence in vertical, horizontal, and lateral directions modeled by Von Karman spectra and implemented by Dryden filters as in the previous example. In addition, seven parameters in the model are allowed to vary individually on a closed interval. These parameters include variation in Cg position as well as uncertainty in the aerodynamic forces and moments along each axis. For the example presented here the numerical values for the variations are shown in Table 5.2. The bounds given for the center of mass are relative to the mean aerodynamic chord. The bounds given for the aerodynamic forces and moments are as a fraction of the value given by the lookup tables.

The algorithm is asked to find the combination of parameters and wind gusts that produce the largest norm of the performance variable vector, i. e., turning radius and altitude error. The worst case combination produced by the algorithm gives the value of each of the parameters at the end point of the allowable interval of variation. Table 5.3 summarizes the values found for the parameters.

The resulting 2-norm of the performance variables is 230 ft. The model simulation used by the algorithm was built in a Simulink diagram shown in Figure 5.17. The behavior of the airplane under the worst case parameter variation selected by the algorithm is illustrated in Figures 5.18 through

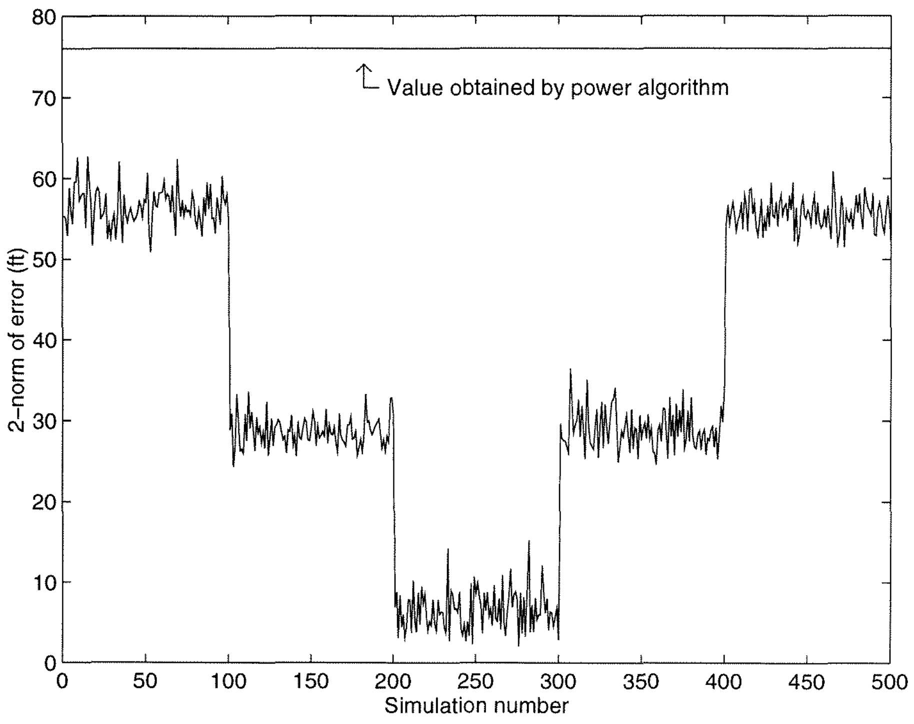

Figure 5.15: Monte Carlo analysis of the perturbations.

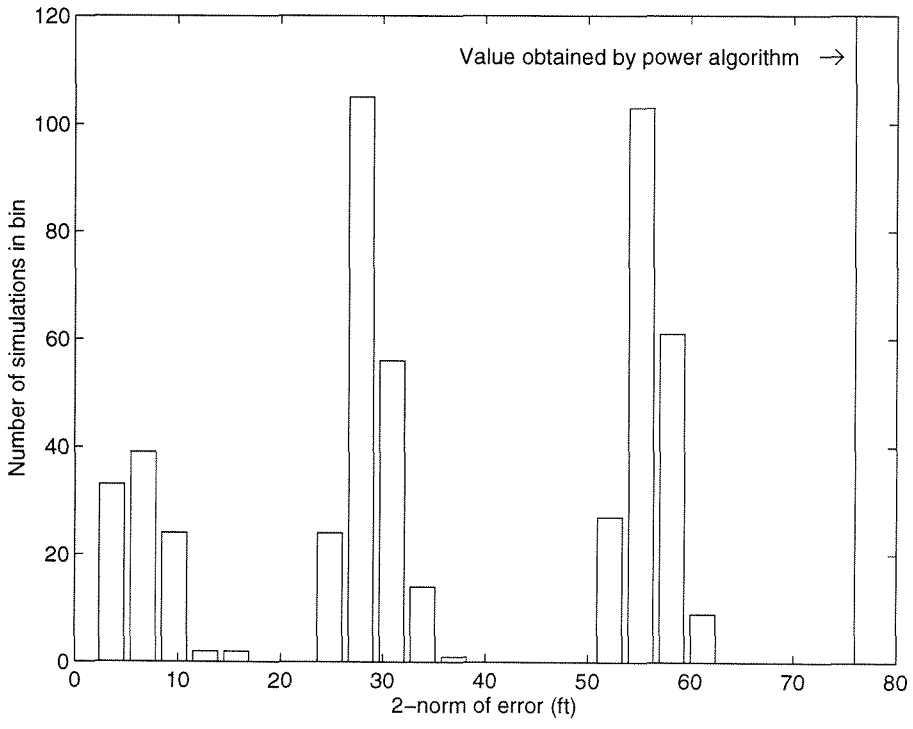

Figure 5.16: Monte Carlo analysis of the perturbations.

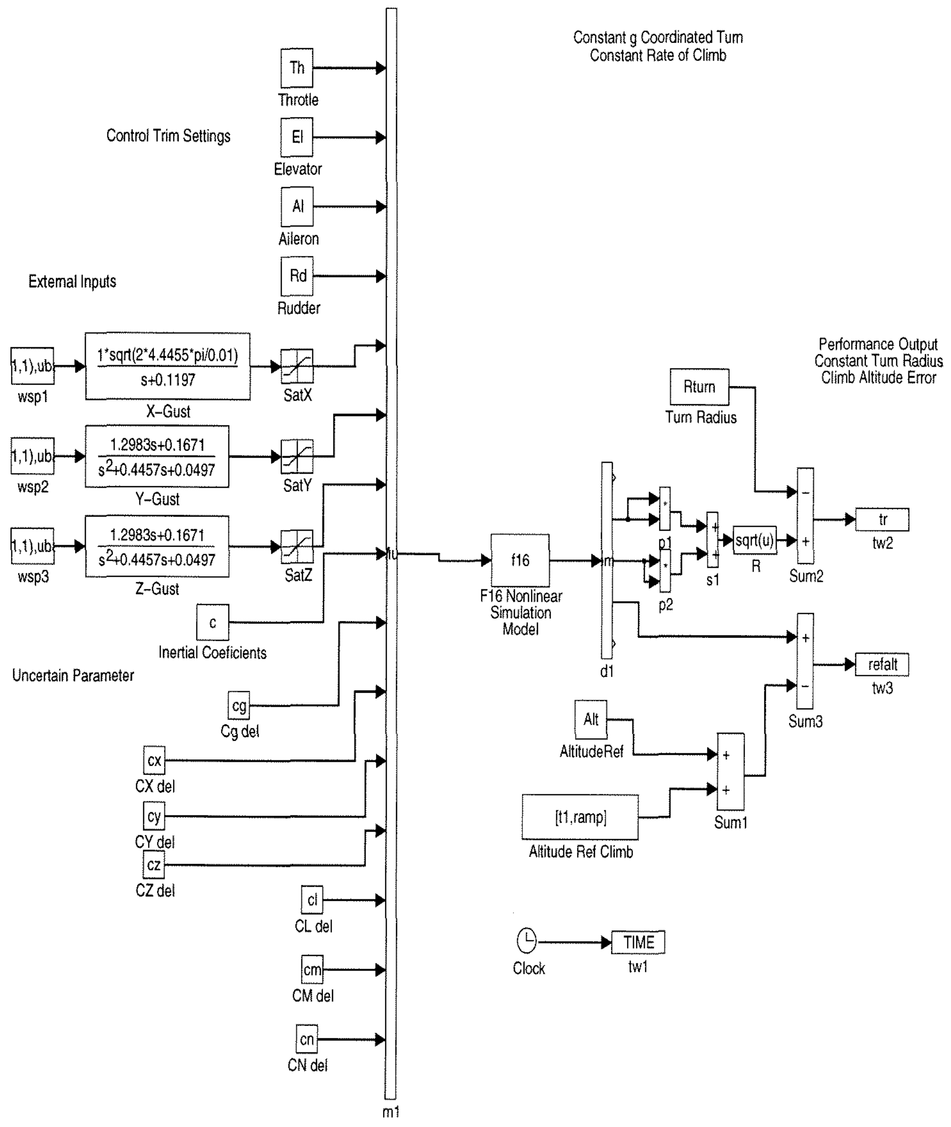

Figure 5.17: Simulink diagram for the F16 robust performance analysis.

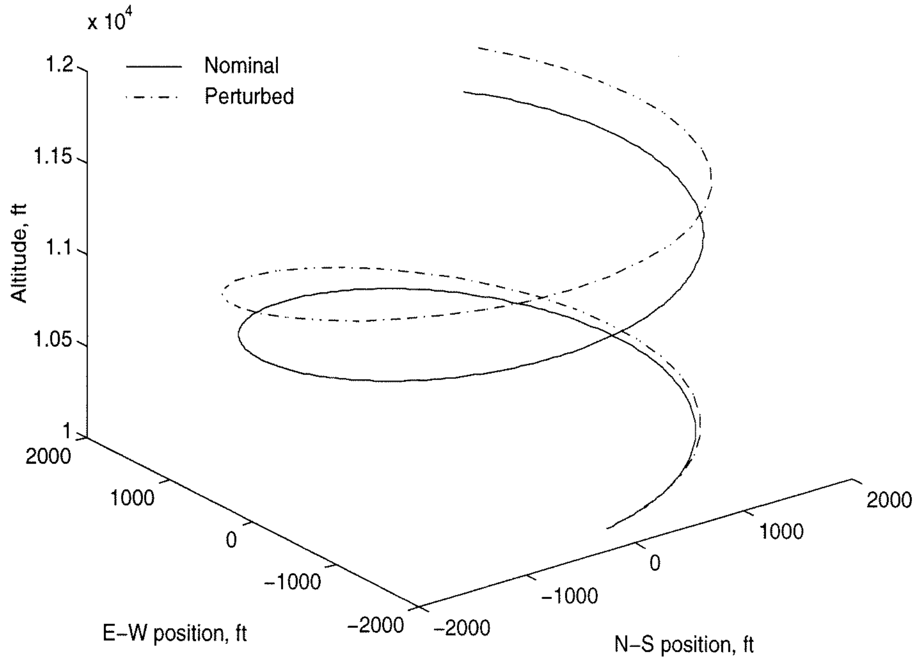

Figure 5.18: Spatial view of the trajectory.

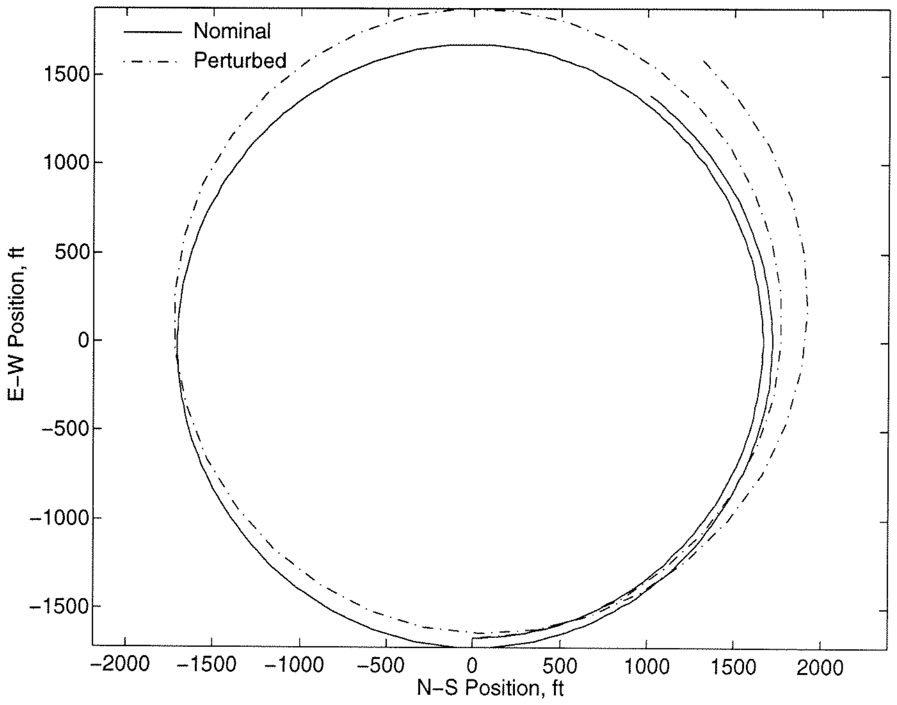

Figure 5.19: Ground track of the trajectory.

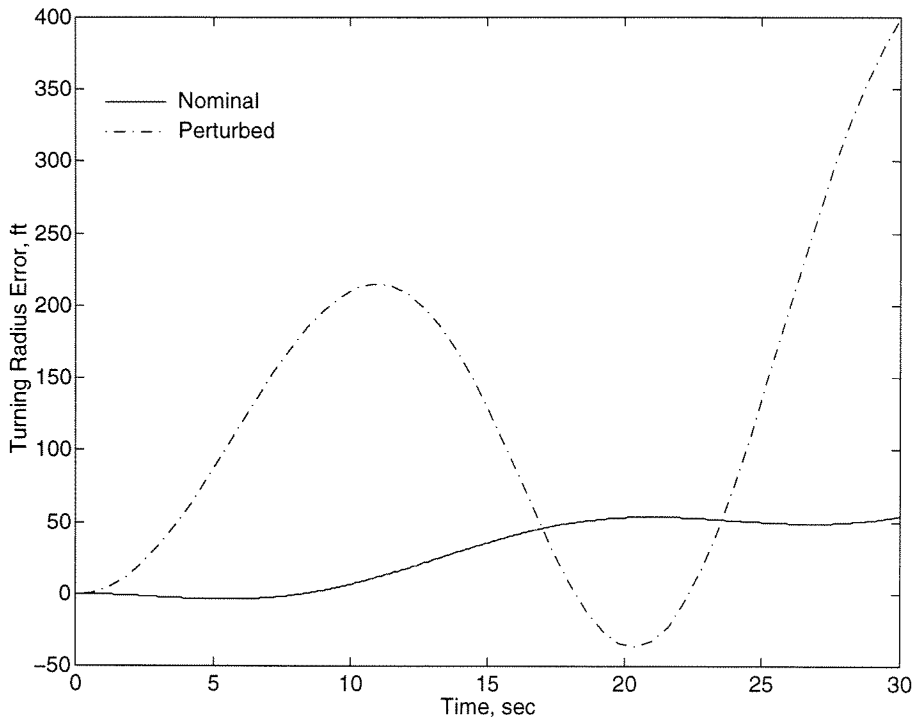

Figure 5.20: Turning radius error.

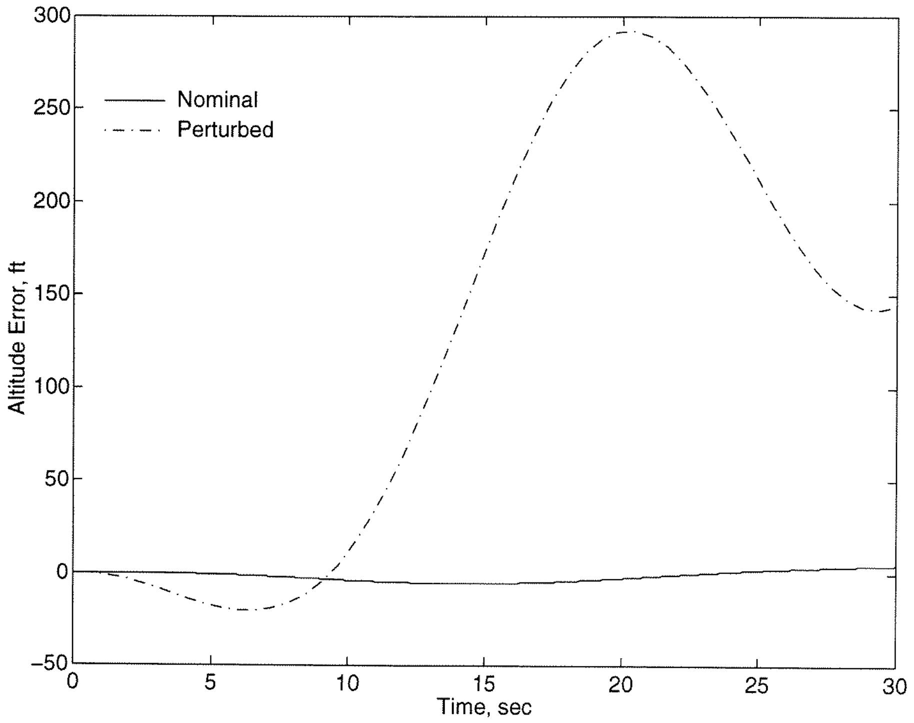

Figure 5.21: Altitude error.

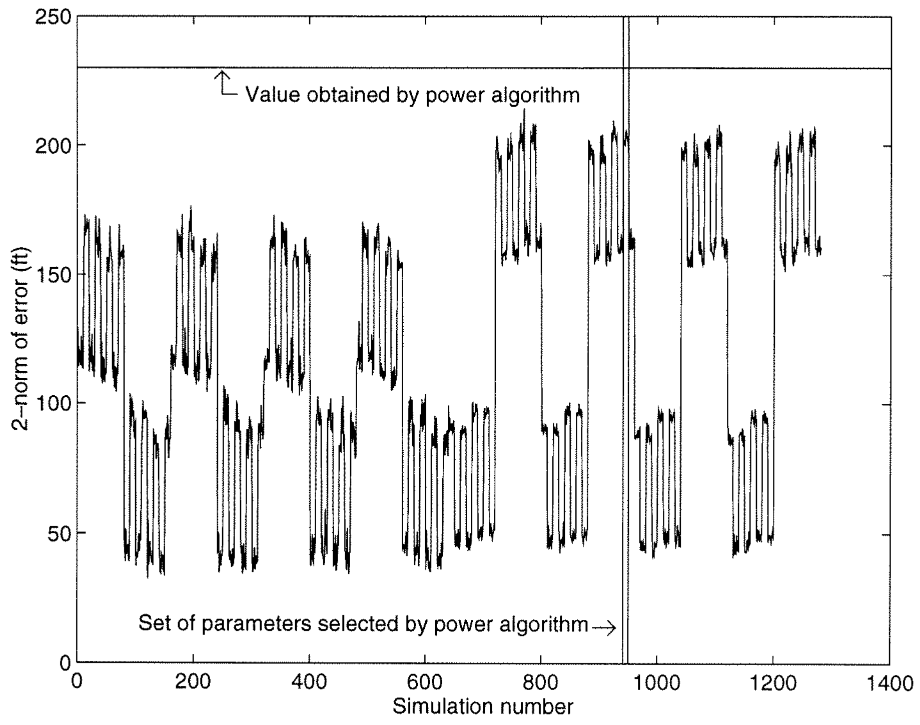

Figure 5.22: Comparison of stochastic and worst case analysis.

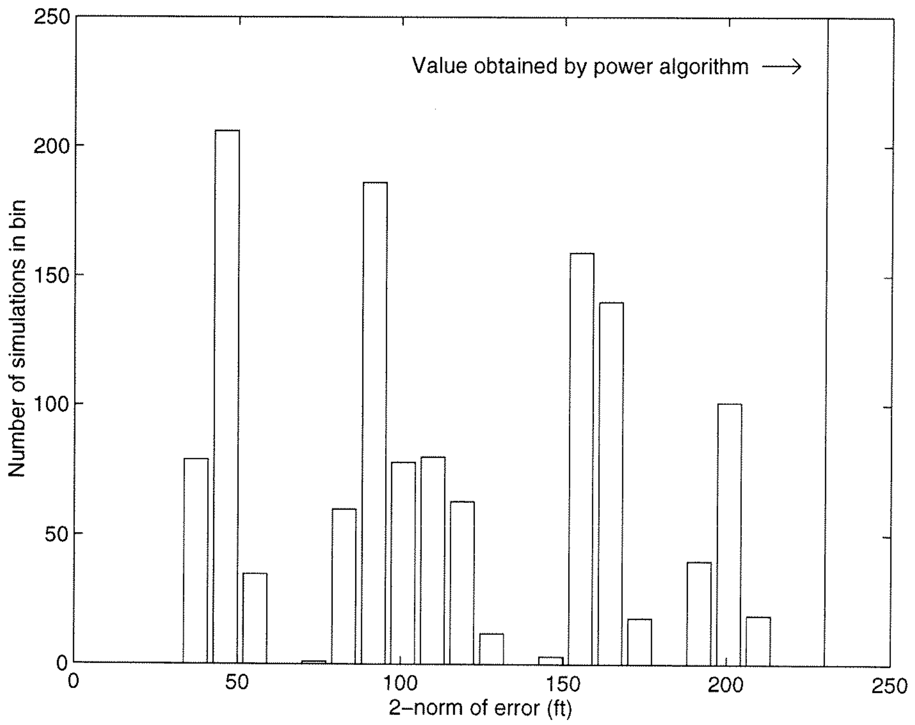

Figure 5.23: Comparison of stochastic and worst case analysis.

		Lower	Upper
Parameter	Name	Bound	Bound
Center of mass	Cg	$0.195\bar{c}$	$0.205\bar{c}$
oX Force	Cx	.975	1.025
oY Force	Cy	.985	1.015
oZ Force	Cz	.97	1.03
Rolling Moment	Cl	.95	1.05
Pitching Moment	Cm	.95	1.05
Yawing Moment	Cn	.95	1.05

Table 5.2: Allowed variations for the uncertain parameters.

Parameter	Value
Cg	$0.195\bar{c}$
Cx	1.025
Cy	.985
Cz	.97
Cl	.95
Cm	.95
Cn	1.05

Table 5.3: Final values for the uncertain parameters.

5.21. The solid line in all of the figures represent nominal trajectory while the dashed lines represent the perturbed trajectory.

To compare with more traditional ways of evaluating nonlinear system behavior, Monte Carlo simulations were run. For each parameter the endpoints of the interval of variation were selected as allowable values. A system simulation with random turbulence subjected to the same restrictions is run for each possible combination of parameter values, 128 in this case. For each of these parameter combinations, ten simulations are performed. The resulting 2-norm of each simulation is plotted in Figure 5.22. The figure shows the 2-norm of the performance vector for each of the simulations as well as the worst case 2-norm. A total of 1280 simulations were performed. The total running time for the power algorithm was 250 minutes. The 1280 Monte Carlo simulation took 290 minutes. (It is also important to mention that the code we use is not optimized for running time. All the computations are done within MATLAB script files. In particular the Jacobian computations as done by the MATLAB function `linmod` is particularly inefficient for our needs. A substantial increase in speed could be achieved simply by writing the code in a compiled language.)

As can be seen from Figure 5.22, the 2-norm of the worst case parameter combination with atmospheric winds shaped by the algorithm is indeed larger than any combination of parameters with random atmospheric winds. The two vertical lines demarcate the interval that corresponds to the same combination of parameters as that selected by the algorithm for the worst case. While this combination is not unique, as is evident from the figure, it does provide us with a better lower bound on the worst case behavior of the airplane for the allowable set of parameter variations than the Monte Carlo method. In terms of computational efficiency, the worst case algorithm is at least four times faster than the Monte Carlo simulations in this particular case. Figure 5.23 presents a frequency distribution of the error 2-norms obtained from the Monte Carlo simulations. The vertical line indicates the worst case value.

Performance specified in the "∞-norm"

The last performance problem analyzed for this model is also for a constant rate of climb constant rate of turn trajectory, but with a different performance objective. Instead of finding the worst case 2-norm of the error signal, we want to find the time instant at which the difference between the nominal and actual trajectories is largest. We will proceed in a manner similar to the one described in Section 3.2. We will multiply each error signal by a pulse of the form

$$P_{t_o}(t) = \exp(-(t - t_o)^2)$$

where t_o is a parameter to the problem. (We use separate parameters for each of the error signals.) Figure 5.24 shows the function P_0. The effective width of this pulse is around 2s, and so this will be the precision with which we are going to find the location of the maximum. The use of a thinner pulse increases the precision but also affects the stability of the algorithm.

For this run we consider the same perturbation as in the last example, except the center of mass remains fixed at the nominal position. The center of the pulses will be allowed to vary between 2 and 28 seconds. (This is to ensure that the pulse does not go out of the interval under consideration.) The algorithm converges in 4 iterations to within .005 percent. The values of the parameters selected are shown in Table 5.4.

Parameter	Value
Cx	.875
Cy	.985
Cz	.97
Cl	1.05
Cm	1.05
Cn	.95
t_o(alt.)	28
t_o(turn)	28

Table 5.4: Final values for the uncertain parameters.

The worst case for the error occurs at the end of the interval. For the worst case signals and parameters, Figure 5.25 shows the error as a function of the position of the pulses. It confirms that the maximum is achieved when both are at the end of the interval.

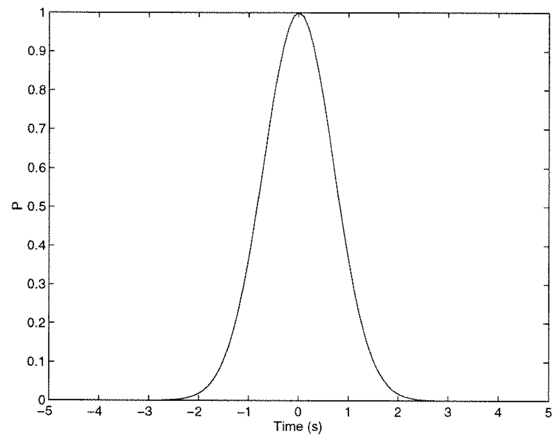

Figure 5.24: Pulse signal.

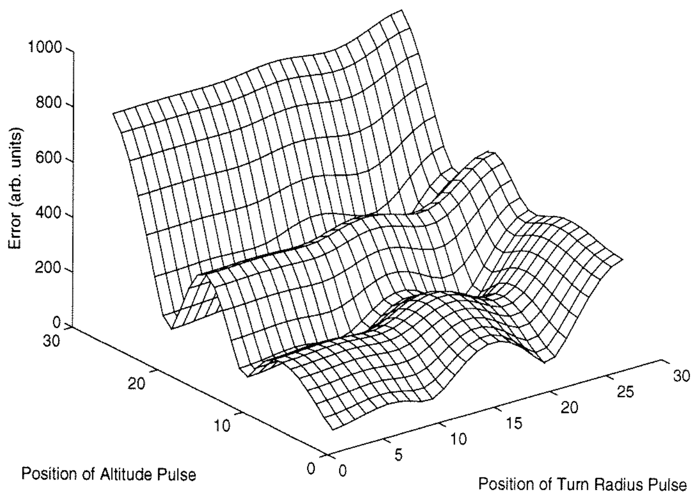

Figure 5.25: Error as a function of pulse position.

Appendix: Matlab Function for the Lower Bound

Introduction

`nllb.m` is a Matlab script that computes a lower bound on the worst case performance of a nonlinear system subject to noise, unmodeled dynamics, and uncertain parameters. It is based on an iterative, power-like algorithm that converges to points in signal and parameter space verifying necessary conditions for a maximum of the tracking error. The theoretical details of the algorithm and its implementation can be found in [31].

Problem Setup

The problem will be described to the algorithm via a Simulink file containing a simulation model of the nonlinear vehicle and a set of variables passed as inputs to `nllb` describing the uncertainty configuration and sizes, the performance variables and other operating parameters.

Simulink file describing the System

The user provides a Simulink file meeting the following characteristics:

- Inputs, in the following order: disturbance signals, inputs corresponding to outputs of the uncertain dynamics, value of the uncertain parameters.
- Outputs, in the following order: performance outputs, outputs that are connected to the inputs of the unmodeled dynamics.

Note that the performance outputs have to be the difference between the actual and nominal trajectory. This nominal trajectory, together with any nominal command signals that guide the system through it have to be part of the simulation file.

Input parameters to `nllb`

The calling syntax is:

```
[g,nt,ubad,parbad,ybad,xbad,conv]=
nllb(sys,perfout,perfin,unc,par,ti,tf,simulator,
                              options,outfile)
```

The following input parameters have to be specified to describe the problem:

`sys`: A string variable that contains the name of the Simulink file describing the problem.

`perfout`: A scalar variable describing the number of output signals that make up the performance measure.

`perfin`: An m by 2 array describing the noise signals, where m is the number of noise signal groups, `perfout(i,1)` is the number of inputs in the ith group, and `perfout(i,2)` is the 2-norm of the ith group.

`unc`: A p by 2 array describing the uncertain dynamics, where p is the number of uncertain dynamics blocks, `unc(i,1)` is the number of output signals from the system connected to the input of the ith block, and `unc(i,2)` is the number of input signals to the system connected to the output of the ith block. All blocks have induced norm 1.

`par`: A q by 2 array describing the uncertain parameters, where q is the number of uncertain parameters, `par(i,1)` is the lower bound for the ith parameter, and `par(i,2)` is the upper bound.

`ti, tf`: Scalar variables describing the beginning and end of the time horizon considered.

`simulator`: A string variable containing the name of the differential equation integrator to be used. It can be one of `euler, rk23, rk45, gear, adams, linsim`. See the Simulink manual for a description of their characteristics.

The function `nllb` also accepts some optional parameters that control the behavior of the algorithm.

`options`: A vector of 3 elements. The first element is the tolerance of convergence `tol`. The algorithm stops when signals change less than `tol` between iterations. The second element is the maximum number of iterations allowed. The last element is a flag set to 1 if you want to plot intermediate results or to zero otherwise.

`outfile`: The name of a file where intermediate results are to be stored. Default is the screen.

The function `nllb` gives the following output variables.

`g`: The maximum tracking error achieved.

`nt`: Time axis for the worst case signals.

`ubad`: A T by s matrix, where T is the number of elements in the time axis and s is the number of noise and unmodeled dynamics inputs containing the signals that achieve the performance g.

`parbad`: A vector containing the values of the parameters achieving the performance g.

`xbad`: A T by n matrix containing the state trajectories that achieve the performance g.

`conv`: A flag set to 1 if the algorithm stopped because the tolerance requirement was met or to 0 if it stopped because it reached the maximum number of iterations specified.

Chapter 6
Sufficient Conditions

Several upper bounds for performance of nonlinear systems have been developed in the last few years. Most of these results are generalizations of Lyapunov theorems and depend on us being able to find a Lyapunov function for the system, or to solve a Hamilton Jacobi equation (see for example [33, 14, 13]). However, there are no systematic ways to accomplish this.

The success of practical methods for analysis of linear systems can be explained to some extent by the fact that linear systems can be naturally described by finite amounts of data, and that therefore properties of the system can be expressed as functions on a finite dimensional space. If we want to replicate that success for nonlinear systems we have to restrict our search to classes of nonlinear systems that can be described finitely.

In the preceding chapters we showed how by considering systems over a finite time horizon, an efficient algorithm for a lower bound on the robust performance level could be developed. Since lower bounds require only local information, the finite description of a system is in this case simply a simulation process: a computer program that returns the outputs corresponding to a given input. Finite time horizon is required to guarantee finite simulation time.

For the upper bound since we are looking for global instead of local results we need to be able to describe the system globally with finite data. We present in this chapter a large class of problems that accept such a representation. We will also show that for problems in that class an important measure of performance can be evaluated by analyzing an auxiliary linear system constructed from the original nonlinear one.

The linear problem belongs to a class for which analysis methods have already been developed. In particular an upper bound on the performance of the linear problem will also be an upper bound for the performance of the original nonlinear one. Together with the lower bound developed in Chapter 4, these two results extend the robustness analysis tools available for linear systems to the nonlinear case.

The organization of this chapter is as follows: in Section 6.1 we describe the class of nonlinear systems under study; in Section 6.2 we will build the auxiliary linear system and prove technical lemmas that connect the behavior of the two; finally in Section 6.3 we prove the main result of this chapter establishing the equivalence in the behavior of the nonlinear and auxiliary system. From there we derive sufficient conditions for performance. A preliminary version of these results has appeared in [30].

6.1 Rational Nonlinear Systems

Our goal is to develop an algorithm, suitable for implementation on a digital computer, to compute a global upper bound for the performance of a nonlinear system. A fundamental precondition for the development of such an algorithm is a finite description of the nonlinear system. Note that a finite description implies more than specifying the problem with finitely many characters; the decoding of such a description must be achievable by a finite procedure as well. Given the large variety of behaviors in nonlinear systems, it is doubtful that such an encoding exists for a general class of problems. However, we will exhibit a large class of nonlinear problems, that we believe to be of engineering relevance, for which we can develop a finite representation and derive from it efficient analysis algorithms.

The class of problems we will study consists of discrete time nonlinear systems, with a finite time horizon performance specification (i. e., we are concerned with the behavior of the system over T time steps). The evolution of the state x and output y of the system will be governed by the equations

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, \delta_p, k) \\ y_k &= g(x_k, u_k, \delta_p, k), \end{aligned} \qquad k = 1, 2, \ldots, T \tag{6.1}$$

where u is the vector of input signals, δ_p is a vector of uncertain parameters, and x_1 is the initial state. The functions f and g are rational expressions of the state, input, and parameter variables. The system will be well posed in the following sense: there exist vectors of positive constants K_k^x, K_k^u, and K_k^δ such that $f(x_k, u_k, \delta_p, k)$ and $g(x_k, u_k, \delta_p, k)$ are well defined for all x, u, and δ_p satisfying

$$|x_k| < K_k^x, \quad |u_k| < K_k^u, \quad |\delta_k^p| < K_k^\delta \quad k = 1, 2, \dots, T, \tag{6.2}$$

where absolute values and inequalities are meant in the element by element sense. Furthermore, if the input signals and parameters remain within those given bounds for all $k = 1, 2, \dots, T$, and the initial state x_1 verifies $|x_1| < K_1^x$, then the state remains within its bounds for all $k = 1, 2, \dots, T$. Finally, we will assume that 0 is an equilibrium point, i. e., $f(0, 0, \delta_p, k) = g(0, 0, \delta_p, k) = 0$.

This class of systems admits a constant matrix representation similar in nature to the Linear Fractional Transformation (LFT) commonly used for linear systems. To simplify the notation we will consider a system with one state, one scalar input, and one uncertain parameter. The extension to a general problem in the class is straightforward.

As f and g are rational functions of x_k and u_k that do not blow up at zero, they can be expressed as linear fractional transformations on those same variables. This means that there exists a matrix M_k whose entries only depend on k and natural numbers m_x and m_u such that

$$\begin{bmatrix} f(x_k, u_k, \delta_k^p, k) \\ g(x_k, u_k, \delta_k^p, k) \end{bmatrix} = M_k * D_k, \tag{6.3}$$

where $*$ denotes the Redheffer star-product, and D is the diagonal matrix

$$D_k = \text{blockdiag}(\delta_k^p, x_k I_{m_x}, u_k I_{m_u}).$$

Remarks: The LFT is just a convenient way of representing the functions f and g. It is not meant to be interpreted as a multiplicative transfer function, the way frequency domain LFTs for linear systems are. By writing the LFT we can separate the coefficient part of the functions f and g from the variable names. (In the same way we can represent a polynomial by giving the vector of coefficients.) This LFT representation gives us the finite information description of the nonlinear system. To specify the system all we need to give is the matrix M and the structure of the matrix D. In the sections that follow we will build a linear system that can be represented, in the conventional way, with a similar LFT on the matrix derived from M. We will also show that the performances of these two systems are related.

The entries in the matrix M can be functions of the time index k. Whether they are or not does not affect the development that follows. For simplicity we will drop the index k. The modifications necessary to account for the dependence of M on k are immediate.

The condition that all states remain bounded when the initial conditions and inputs are bounded is restrictive. In general it may be hard to determine whether this will be the case. In general, computing these constants may be harder. It will be necessary to rely on knowledge of the system to derive bounds on the states. For an aircraft, for example, we can have a very good idea of how fast it can possibly accelerate, and thus we can derive bounds on all the velocities and positions over a finite horizon. The bounds don't need to be very tight. Tighter bounds, however, will result in an optimization problem with better numerical conditioning. There will be a tradeoff between the accuracy of the solution and the amount of work devoted to obtaining the preliminary bounds.

Example

Consider the system described by the equations

$$\begin{aligned}
x_{k+1}^1 &= (x_k^2)^2 + x_k^1 \\
x_{k+1}^2 &= x_k^1 u_k + x_k^2 \\
y_k &= x_k^1.
\end{aligned}$$

The matrix D_k in this case will be

$$\begin{bmatrix} x_k^1 & & & \\ & x_k^2 & & \\ & & x_k^2 & \\ & & & u_k \end{bmatrix},$$

and the matrix M_k will be

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

This system is defined for all values of x_k^1, x_k^2, and u_k. Furthermore, since

$$\begin{aligned} |x_{k+1}^1| &\leq |(x_k^2)^2| + |x_k^1| \\ |x_{k+1}^2| &\leq |x_k^1||u_k| + |x_k^2|, \end{aligned}$$

the constants K_k^x can be calculated for any k starting from bounds on the initial conditions, and bounds on the inputs.

6.2 Auxiliary Linear System

In this section we will construct a linear system starting from the matrix M of the finite representation of the nonlinear system discussed in the previous section. We will do this construction in two stages. First for a one time step system, and then for a T time steps one. Intuitively, what we are going to do can be explained as follows: first construct an uncertain linear system depending on a series of real parameters, entering in the same way as the states and inputs enter in the original nonlinear system. Then use linear constraints to force those parameters to track the corresponding states and inputs. The notation necessary will be cumbersome. To make the development easier to follow, we will first construct the auxiliary system for one time step, and then for two. The generalization to T time steps follows. For the same reason we will also work with a one state, one uncertain parameter, one input system, and we will assume that M does not depend on k. Lifting these assumptions is immediate.

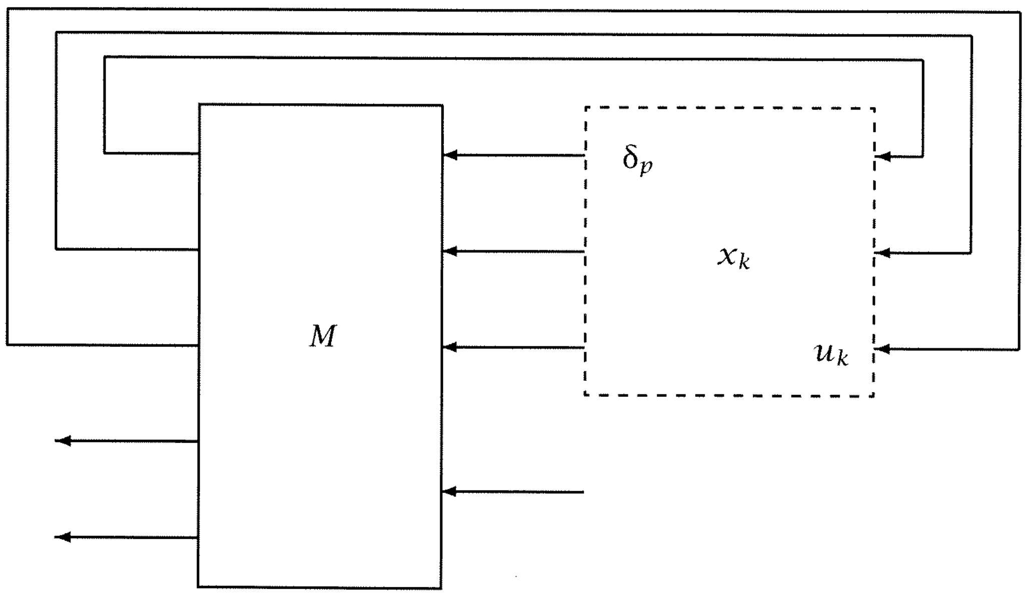

Figure 6.1: Nonlinear system as an LFT.

Auxiliary system for a length 1 time horizon

In Section 6.1 we showed how rational nonlinear systems can be expressed as linear fractional transformations on the states, parameters and inputs. The system is represented by a linear fractional transformation of the form

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = (M * D). \tag{6.4}$$

A block diagram for this system is shown in Figure 6.1. Partition the matrix M compatibly with the signals in Figure 6.1 and define the matrices

$$M_e = \begin{bmatrix} M_{11} & M_{12} & 0 & M_{13} & 0 & 0 & 0 & M_{14} \\ M_{21} & M_{22} & 0 & M_{23} & 0 & 0 & 0 & M_{24} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_{31} & M_{32} & 0 & M_{33} & 0 & 0 & 0 & M_{34} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_{41} & M_{42} & 0 & M_{43} & 0 & 0 & 0 & M_{44} \\ M_{51} & M_{52} & 0 & M_{53} & 0 & 0 & 0 & M_{54} \end{bmatrix}$$

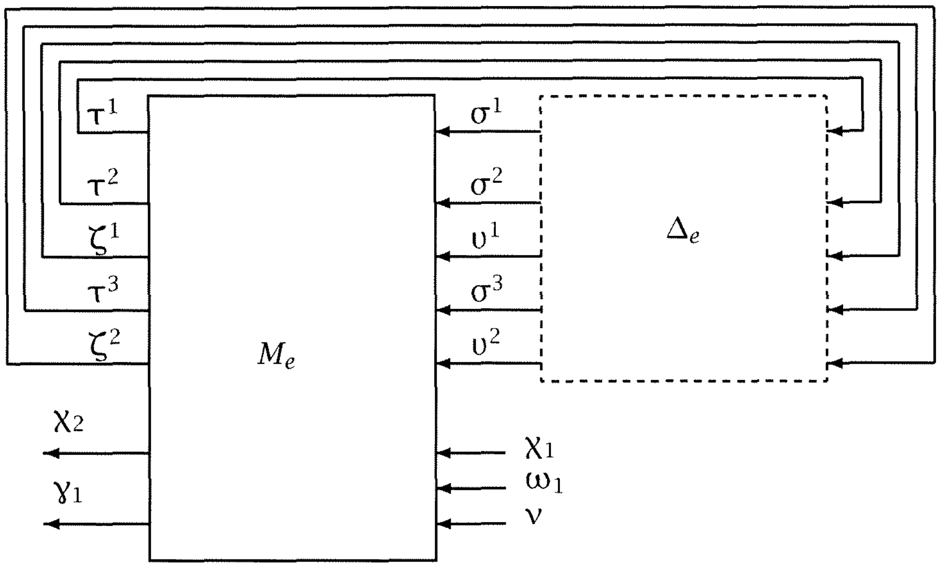

Figure 6.2: Second step in construction of auxiliary system.

and

$$\Delta_e = \begin{bmatrix} \delta^p & & & & \\ & \delta^x I & & & \\ & & \delta^x & & \\ & & & \delta^u I & \\ & & & & \delta^u \end{bmatrix}.$$

Consider the linear, uncertain, discrete time system shown in Figure 6.2. This system has one state, two inputs and one output. The final step in the development of the auxiliary system is adding linear implicit constraints on the inputs signals. Add to the system in Figure 6.2 the two linear constraints

$$\begin{aligned} \upsilon^1 - \chi_1 &= 0 \\ \upsilon^2 - \omega_1 &= 0. \end{aligned} \tag{6.5}$$

The resulting set of equations characterizes an implicit uncertain linear system as those described in Section 6.3, and can be represented by the block diagram in Figure 6.3, where C_e is the matrix

$$C_e = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{bmatrix}. \tag{6.6}$$

The following lemma establishes the connection between the nonlinear and the auxiliary system.

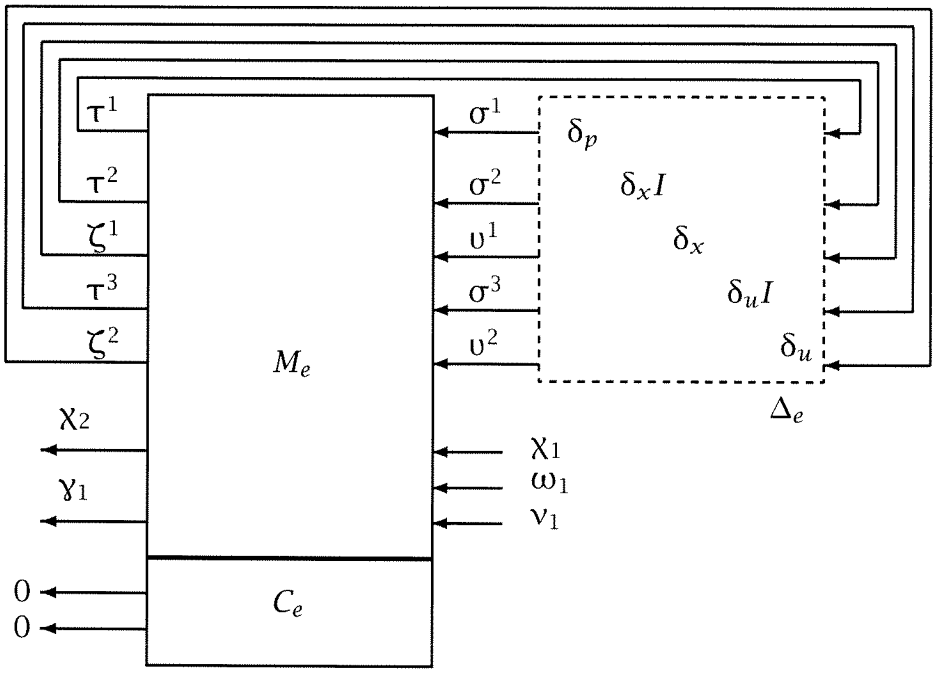

Figure 6.3: Third step in construction of auxiliary system.

Lemma 6.1 *If* $\nu = 1$, $\chi_1 = x_1$, *and* $\omega_1 = u_1$ *then*

$$\begin{aligned} \chi_2 &= f(x_1, u_1, \delta_1^p) = x_2 \\ \gamma_1 &= g(x_1, u_1, \delta_1^p) = y_1. \end{aligned} \tag{6.7}$$

Proof: Note that the extension of the system was done so that

$$\begin{aligned} \upsilon^1 &= \delta^x \nu_1 \\ \upsilon^2 &= \delta^u \nu_1. \end{aligned} \tag{6.8}$$

So by setting $\nu_1 = 1$ in Equations (6.8), we have $\upsilon^1 = \delta^x$ and $\upsilon^2 = \delta^u$. The constraints in (6.5) further impose

$$\begin{aligned} \delta^x &= \chi_1 \\ \delta^u &= \omega_1, \end{aligned} \tag{6.9}$$

and thus according to the hypothesis of this lemma

$$\begin{aligned} \delta^x &= x_1 \\ \delta^u &= u_1. \end{aligned}$$

Since

$$\begin{bmatrix} \chi_2 \\ \gamma_1 \end{bmatrix} = (M * \Delta)\nu_1, \qquad (6.10)$$

and

$$\begin{bmatrix} x_2 \\ y_1 \end{bmatrix} = M * D,$$

substitution of (6.9) in (6.10) and (6.2) gives the desired result. ■

We finish this section with a technical lemma we will need for the main theorem of this chapter.

Lemma 6.2 *If* $\nu = \chi_1 = \omega_1 = 0$, $\delta^p < K^p$, $|\delta^x| < K^x$, *and* $|\delta^u| < K^u$, *then all signals in Figure 6.3 are* 0.

Proof: The norm bound conditions in δ^p, δ^x and δ^u imply that $M_e * \Delta_e$ is a well posed linear fractional transformation. This means that the system of linear equations it defines has a unique solution. Since for a linear system $\vec{0}$ is a solution when the inputs are set to 0 the lemma follows. ■

Auxiliary system for a length T time horizon

For one time step, Lemma 6.1 shows that the auxiliary linear system states and outputs track those of the nonlinear one. This result can be extended to any time horizon of finite length T by concatenating as many instances of the auxiliary system in a simple fashion. In this section we show how to carry out that concatenation. For simplicity in notation we will show how to do this concatenation for two time steps; the generalization to any finite number of steps is straightforward. Consider the matrices M_e^1, M_e^2, Δ_e^1, and Δ_e^2 that define the auxiliary system for the first two time steps. First, connect them as shown in Figure 6.4, to form another uncertain system denoted (M_c, Δ_c). The input vector a to the matrix M_c and the corresponding output vector will inherit from the auxiliary systems at time steps 1 and 2 the following partition:

$$\begin{aligned} a &= [\sigma_1^1, \sigma_1^2, \upsilon_1^1, \sigma_1^3, \upsilon_1^2, \sigma_2^1, \sigma_2^2, \upsilon_2^1, \sigma_2^3, \upsilon_2^2, \chi_1, \omega_1, \omega_2, \nu]^t \\ b &= [\tau_1^1, \tau_1^2, \zeta_1^1, \tau_1^3, \zeta_1^2, \tau_2^1, \tau_2^2, \zeta_2^1, \tau_2^3, \zeta_2^2, \chi_3, \gamma_1, \gamma_2]^t. \end{aligned} \qquad (6.11)$$

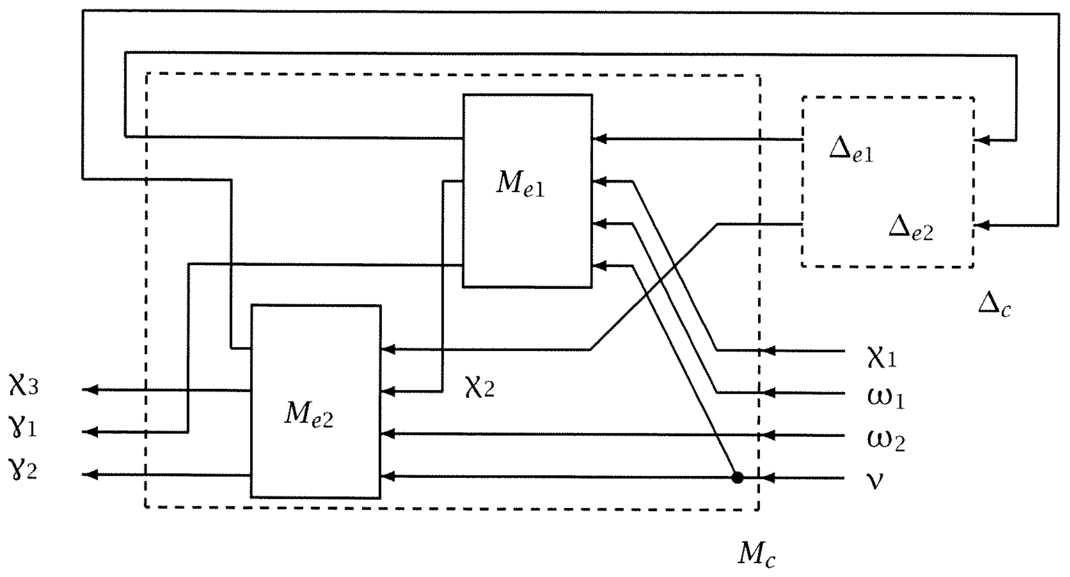

Figure 6.4: Two time-steps auxiliary system.

From Figure 6.4 note that the input χ_2 of the second auxiliary system has been connected to the signal

$$\chi_2 = M_{41}\sigma_1^1 + M_{42}\sigma_1^2 + M_{43}\sigma_1^3 + M_{44}\nu$$

where M_{ij} corresponds to the i, j element of the partition of the matrix for the first auxiliary system used in the previous section. Since χ_2 is a linear combination of the entries in a, we can write it as

$$\omega_2 = C_1 a.$$

Add then to the system (M_c, Δ) the following constraints in a

$$\begin{aligned} \upsilon_1^1 - \chi_1 &= 0 \\ \upsilon_1^2 - \omega_1 &= 0 \\ \upsilon_2^1 - C_1 a &= 0 \\ \upsilon_2^2 - \omega_2 &= 0. \end{aligned} \tag{6.12}$$

We can write all these constraints as one vectorial equation in a:

$$Ca = 0.$$

We have thus defined a new implicit uncertain system (M_c, C, Δ_e). The following is a direct consequence of Lemma 6.1.

Lemma 6.3 *If* $\nu = 1$, $\chi_1 = x_1$, $\omega_1 = u_1$, *and* $\omega_2 = u_2$, *then*

$$\begin{aligned} \chi_3 &= x_3 \\ \gamma_k &= y_k \quad k = 1, 2. \end{aligned} \tag{6.13}$$

Proof: It follows from applying Lemma 6.1 to the first component of M_c, concluding that the χ_2 input is connected to a signal of value x_2, and applying Lemma 6.1 again to this second half. ∎

We can repeat this procedure for all time steps. Note that in the final step we can include or omit the final state as part of the output. Whether we do so or not will depend on the nature of our problem and the performance specification being considered.

The second lemma of the preceding section also generalizes to the length T horizon.

Lemma 6.4 *If* $\nu = \chi_1 = \omega_1 = \omega_2 = 0$ *and* $|\delta_i^p| < K_i^p$, $|\delta_i^x| < K_i^x$, *and* $|\delta_i^u| < K_i^u$ *for* $i = k,\ k+1$, *then all signals in Figure 6.4 are* 0.

Proof: Apply Lemma 6.2 to the first block in the interconnection. Note that all signals that are propagated to the next block are zero, and apply Lemma 6.2 again. ∎

Once again the generalization of this lemma to T time steps is immediate.

6.3 Performance Analysis

We will now proceed to show how the auxiliary system just defined can be used to analyze performance for the class of nonlinear systems discussed in Section 6.1. We will show in this section that the performance specification given for the nonlinear system is met if and only if a similar performance specification holds for the auxiliary linear system. We can then test the performance of the auxiliary linear system using the standard analysis techniques.

Nonlinear and auxiliary performance

The performance problem that will result in the linear system is slightly different than the most common one. In general, as was shown in Chapter 2, performance problems for linear systems have the form

$$\|y\| \leq \|u\|,$$

where y is the performance output and u is the noise input. In our case we will have a performance requirement of the form:

$$\|u\| \leq 1 \quad \Rightarrow \|y\| \leq 1.$$

However, a slight modification of the linear system can convert the second type of specification to the first. The idea for this modification is taken from [19]. Partition the matrix M_c compatibly with the signals,

$$[\sigma_1^1\, \sigma_1^2\, \upsilon_1^1\, \sigma_1^3\, \upsilon_1^2\, \sigma_2^1\, \sigma_2^2\, \upsilon_2^1\, \sigma_2^3\, \upsilon_2^2] \text{ and } [\chi_1\, \omega_1\, \omega_2\, \nu],$$

and denote the two parts M_{c_1} and M_{c_2}. We now define the matrix

$$M_a = \begin{bmatrix} & M_{c_1} & & \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ & M_{c_2} & & \end{bmatrix},$$

and build the uncertain constrained linear system as shown in Figure 6.5, denoted (M_a, C, Δ_a). Define the norm bounded uncertainty structures:

$$\begin{aligned} \mathbf{B\Delta_e} = \{&\text{blockdiag}(\delta_1^p, \delta_1^x I_{m_x+1}, \delta_1^u I_{m_u+1}, \delta_2^p, \delta_2^x I_{m_x+1}, \\ &\delta_2^u I_{m_u+1}, \ldots, \delta_T^p, \delta_T^x I_{m_x+1}, \delta_T^u I_{m_u+1}), \\ &|\delta_i^p| \leq K_i^p, |\delta_i^x| \leq K_i^u i = 1, 2, \ldots, T\} \end{aligned}$$

and

$$\begin{aligned} \mathbf{B\Delta_a} \quad = \quad &\{\text{blockdiag}(\Delta_e, \Delta_p, \Delta_{x_1}, \Delta_\omega), \Delta_e \in \mathbf{B\Delta}, \\ &\|\Delta_p\| \leq 1, \|\Delta_{x_1}\| \leq 1, \|\Delta_\omega\| \leq 1\} \end{aligned}$$

where Δ_p, Δ_{x_1}, and Δ_ω are full matrices with dimensions defined by Figure 6.5. (See [29] for a more detailed study of this analysis setup.)

We are now ready to state the main theorem of this chapter that will allow us to derive a sufficient condition for performance of the original nonlinear system.

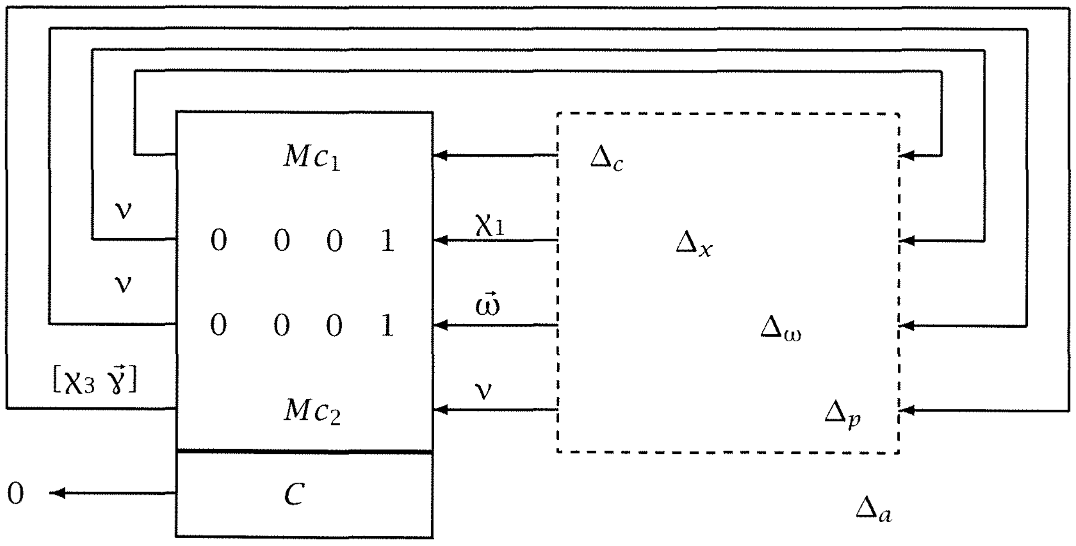

Figure 6.5: Analysis setup.

Theorem 6.5 *For all signals u with $\|u\|_2 \leq 1$ and $|u_k| < K_k^u$ and all initial states $x_1 < K_k^x$, we will have $\|y\|_2 \leq 1$ if and only if the performance auxiliary system is well posed for all $\Delta_a \in \mathbf{B\Delta_a}$.*

Proof: First assume the system has a nonzero solution. Then $\nu \neq 0$ because otherwise all signals are 0. This follows from $\vec{\omega} = \Delta_\omega \nu$, $x_1 = \Delta_x \nu$ and Lemma 6.4. Since the equations are linear, then there is also a solution with $\nu = 1$. The condition $\|\Delta_\omega\| \leq 1$ implies that $\|\vec{\omega}\| \leq 1$, and the condition $\|\Delta_p\| \leq 1$ implies then that $\|\vec{\gamma}\| \geq 1$. Since $\nu = 1$ from Lemma 6.3, we conclude that there is an input and initial condition for the nonlinear system within the specified bounds such that $\|y\| > 1$, and so the performance requirement is not met.

Conversely assume that an allowable initial state x_1 and an allowable input signal u exist such that $\|y\| \geq 1$. Then setting $\nu = 1$, $\vec{\omega} = u$, and $\omega_1 = x_1$ in the set of equations represented by Figure 6.5, we have $\|\gamma\| \geq 1$, and consequently there exists a $\Delta_a \in \mathbf{B\Delta_a}$ such that the given system of equations has a nonzero solution.

■

This theorem establishes the equivalence between analysis of a nonlinear system and analysis of a linear one. The solution of the given linear problem is NP-Complete. Theorem 6.5 thus tells us that solving the performance analysis question in the class of nonlinear systems being considered is not worse than an NP-Complete problem whose size grows with the length of the time horizon.

Implicit uncertain linear systems

In Chapter 2 we summarized some recent results in implicit systems. We reproduce here some of the results mentioned there for continuity. An implicit uncertain system, denoted (A, C, Δ) with input u and output y, is defined by the system of equations

$$\begin{aligned} \begin{bmatrix} z \\ y \end{bmatrix} &= A \begin{bmatrix} v \\ u \end{bmatrix} \\ 0 &= C \begin{bmatrix} v \\ u \end{bmatrix} \\ v &= \Delta_u z, \end{aligned} \tag{6.14}$$

where $\|\Delta_u\|_2 \leq 1$ Without loss of generality we will assume $\|\Delta_u\| \leq 1$. We can add to this system the following performance specification

$$\|y\| \leq \|u\|, \tag{6.15}$$

or equivalently

$$y = \Delta_p u \tag{6.16}$$

with $\|\Delta_p\| \leq 1$. As usual define $\Delta = \text{blockdiag}(\Delta_u, \Delta_p)$. Then the following theorem provides us with a sufficient condition for the given performance specification to be met.

Theorem 6.6 *[24] If there is a solution to the linear matrix inequality:*

$$C^{\perp}(A^t D_l^* D_l A - D_r^* D_r) C^{\perp *} < 0 \tag{6.17}$$

where D_r and D_l are positive definite matrices verifying $D_l \Delta = \Delta D_r$ for all Δ with $\|\Delta\| \leq 1$, then the system of Equations (6.14) and (6.16) admits only the trivial solution (and therefore the performance specification given in (6.15) is met for all allowable Δ_u).

Sufficient conditions for performance

Once the problem of performance of the nonlinear system has been reduced to a performance question over a linear one, we can use the results of the linear theory to analyze problems in our class.

Without loss of generality we can assume that the performance bounds, the norm of the input and the bounds K^x and K^u are all 1. Putting Theorems 6.5 and 6.6 together we can prove the following

Theorem 6.7 *The nonlinear system in (6.1) verifies the given performance specification, if there are positive definite matrices D_l and D_r verifying $\Delta_a D_l = D_r \Delta_a$ for all $\Delta_a \in \mathbf{\Delta}_\mathrm{a}$ and such that*

$$C^{\perp}(M_a^* D_l^* D_l M_a - D_r^* D_r) C^{\perp *} < 0.$$

A sufficient condition for performance of a nonlinear system can thus be reduced to solving a convex optimization problem over a finite dimensional space. The number of parameters in the optimization problem grows linearly with the length of the horizon T. The optimization problem takes the form of a Linear Matrix Inequality (LMI). Recent developments in systems theory have shown that several important problems can be reduced to solving LMI's and consequently a significant effort has been put in developing practical algorithms for them [18]. Commercial packages are available that implement some of these methods [10].

Remarks: Current LMI solving algorithms usually have computation time growth proportional to the cube of the size of the matrix M_a. However, these algorithms do not exploit the specific structure of our matrices and it is conceivable that cubic growth with T can be avoided by tailoring the standard algorithms to our specific case.

6.4 Numerical Example

We will test the analysis technique on simplified dynamics for the Ducted Fan experimental setup described in [6]. The nonlinear dynamics used are

$$\left\{\begin{array}{rcl} \dot{x}_1 & = & x_2 \\ \dot{x}_2 & = & u_1 + u_2 x_5 \\ \dot{x}_3 & = & x_4 \\ \dot{x}_4 & = & -u_1 x_5 + u_2 \\ \dot{x}_5 & = & x_6 \\ \dot{x}_6 & = & r u_1 + k x_2 x_5, \end{array}\right. \tag{6.18}$$

with the outputs defined as

$$\left\{\begin{array}{rcl} y_1 & = & x_1 \\ y_2 & = & x_3 \\ y_3 & = & x_5. \end{array}\right. \tag{6.19}$$

Since x_5 appears in all the nonlinear terms, we will rewrite these equations as

$$\left\{\begin{array}{rcl} \dot{x}_1 & = & x_2 \\ \dot{x}_2 & = & u_1 + u_2 \delta \\ \dot{x}_3 & = & x_4 \\ \dot{x}_4 & = & -u_1 \delta + u_2 \\ \dot{x}_5 & = & x_6 \\ \dot{x}_6 & = & r u_1 + k x_2 \delta, \end{array}\right. \tag{6.20}$$

with the constraint

$$\delta - x_5 = 0. \tag{6.21}$$

We will consider a 1 second time horizon. The 2-norm of the combined inputs will be set at .1 and our performance measure will be the 2-norm of the combined outputs. As a first step we compute a lower bound for the performance using the procedure described in [31]. To compute an upper bound we convert the system into a discrete time one, with a sample time of 0.05s. The auxiliary linear system corresponding to the 20 time samples is represented by a 147 by 127 system matrix and a 20 by 127 constraints matrix. The procedure described above answers the question, can the norm of the output be bigger than γ? Since we have a lower bound for the maximum γ, we will ask the question for a succession of values. The results obtained are shown in Table 6.1. The computations were made with the Matlab toolbox LMI-Lab. The last column in the table shows the spectral radius of the right hand side of LMI, at the final point.

γ/γ_l	Feasible	Iter.	ρ(RHS) &
1	no		.96
3	yes	34	-496
2	yes	117	-0.8868
1.5	no		0.96
1.75	no		0.96
1.85	no		0.96
1.95	yes	160	-0.452

Table 6.1: Upper bound feasibility.

Remarks: The LMI solver stops if after a certain number of iterations it cannot improve the performance index. Thus reports of non feasibility could in principle not be true. This is consistent with the nature of the test. As an upper bound it only guarantees that the error will not be worse than a certain amount. There is a tradeoff between computation time, and the precision with which we would like to have the answer. It is therefore of fundamental importance to have both an upper and lower bound. In our case computation time varied between 4 and 8 hours depending on the number of iterations needed. After 300 iterations the LMI were assumed not feasible.

6.5 Conclusion

The main result of this chapter (Theorem 6.5) proves the equivalence of the questions, “Does an uncertain rational nonlinear system always meet a given performance specification?” and the apparently simpler one, “Does a linear uncertain system always meet a given performance specification?”

From this equivalence we derived a convex upper bound for a large class of nonlinear performance problems that takes the form of a finite Linear Matrix Inequality. We believe that this result, together with the algorithm for computing a lower bound for the same class of problems presented in Chapter 3, provides a very good first step in the direction of extending the tools for analysis of linear systems to nonlinear ones.

The upper bound derived from Theorem 6.5 is difficult to compute even for a small problem with current technology, since the size of the resulting LMI can easily exceed the capability of LMI solvers like LMI-Lab. For example, evaluating the performance condition over a 9 second time horizon for a complete model of the Caltech Ducted Fan experimental setup ([6]) will give an LMI with approximately 3000 parameters. However, a large effort is being put into the development of algorithms to solve LMI's, and we believe that the technology to compute the nonlinear upper bounds derived in this paper will soon be available. Since the number of parameter grows linearly with the time horizon, a tenfold increase in computer speed will allow us to solve a problem ten times bigger. On the other hand, solving the problem in the traditional way by gridding the initial state, noise and parameter spaces in order to obtain a global answer results in problems with exponential growth; consequently these methods will benefit far less from increased computation speed.

Another consequence of Theorem 6.5 is that robustness analysis of performance for the class of nonlinear systems presented in this paper is proved to be not intrinsically harder than analysis of linear systems, from a computational complexity point of view.

Chapter 7

Linear Time Varying Systems

In previous chapters we showed how, when considering a nonlinear system over a finite time horizon, we could develop analysis algorithms similar to those used for linear time invariant systems studied over an infinite time horizon. If the system under study is linear but time varying, additional results can be obtained by analyzing the behavior of the system over a finite time horizon.

In this chapter we will explore different possible setups for uncertainty and performance requirements in the time domain based on quadratic constraints for discrete time systems, and we will show how they can be reduced to the computation of the structured singular value of a constant matrix.

An important issue in time domain based tests is computational complexity. It is to be expected that the complexity will grow with the length of the time horizon. It is important to exploit the structure of the matrices associated with the time domain tests to avoid this growth becoming prohibitive. We will discuss how the standard algorithm for computing the lower bounds for μ can be modified to take advantage of the structure of this particular problem.

We also investigate the behavior of the time domain tests in the limit when the time horizon tends to infinity. We establish connections between these limits and the frequency domain robustness tests. We expect these connections to shed light on the nature of the frequency domain tests for systems with uncertainty described by integral quadratic constraints.

The notation used here is fairly standard and is essentially taken from [9] and [35]. For any square complex matrix M we denote the complex conjugate transpose by M^*. The largest singular value and the structured singular value are denoted by $\overline{\sigma}(M)$ and $\mu(M)$ respectively. The spectral radius is denoted $\rho(M)$. For any complex vector x, x^* denotes the complex conjugate transpose, x^t, the transpose and $|x|$, the Euclidean norm. Finally, $\{a_i\}_{i=0}^n$ denotes the sequence $a_0 \cdots a_n$.

7.1 Uncertain Finite Time Horizon Systems

LTI systems are normally described as transfer functions. However, in order to derive computable tests we have to describe them in terms of constant matrices. This is achieved in state space by incorporating the delay operator into the uncertainty, and in the frequency domain case by doing a search over frequency, the analysis at each frequency point reducing to a constant matrix problem. None of these approaches can be applied to systems considered over a finite time horizon. However, for these systems a natural finite matrix representation can be achieved by mapping the temporal axis into the spatial one. To illustrate this concept consider the system that obeys the following equations

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k, \end{aligned} \tag{7.1}$$

over a time horizon of length 3. These equations can be rewritten as:

$$\begin{pmatrix} x_{k+1} \\ y_k \end{pmatrix} = M_k \begin{pmatrix} x_k \\ u_k \end{pmatrix}. \tag{7.2}$$

We can now define a mapping from the initial state and the time history of the inputs from $k = 0$ to 2, to the final state and the time history of the outputs from $k = 0$ to 2. Denote that mapping with the symbol $M_{[3]}$ (see Figure 7.1). The system is now represented by a single constant matrix $M_{[3]}$, over which we can write our performance bounds. However, the size of the matrix $M_{[k]}$ grows with the length of the time horizon. This means that a strong emphasis has to be put on the development of efficient algorithms for the computation of the stability tests.

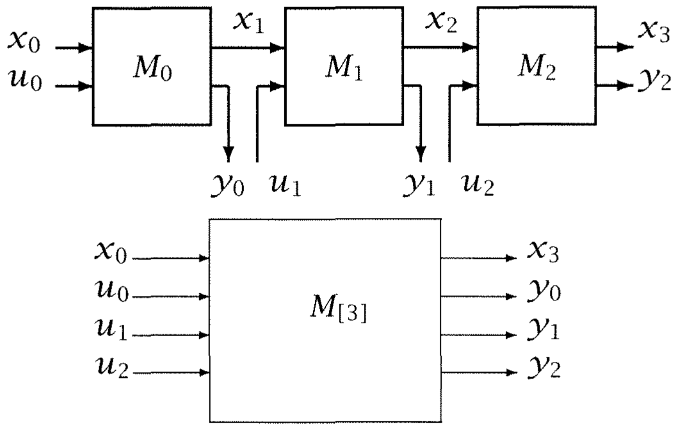

Figure 7.1: Conversion of the problem to a constant matrix.

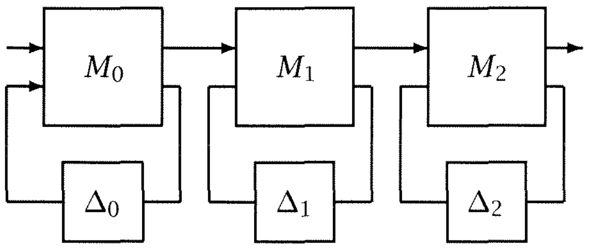

Figure 7.2: Adding uncertainty as an LFT.

We can add uncertainty to this system to model time varying or time invariant parameters or norm bounded unmodeled dynamics. As is the case with infinite horizon systems, we will describe the uncertain model as a linear fractional transformation of a nominal plant, and a structured uncertainty operator. As an example, Figure 7.2 shows how time varying parametric uncertainty can be added to the system given by Equations (7.2).

In the sections that follow we will show how to form the uncertain system as an LFT for the other classes of problems.

7.2 Robust Performance Problems

A wide class of system analysis problems can be characterized as noise rejection problems. In this case, given a bound on a particular norm of the inputs, it is desired to find the worst case norm of the output. The quotient between those two bounds is called the performance of the system. In infinite horizon systems, stability is a precondition for the norms of the outputs, and thus performance, to be defined. This is not the case in discrete time finite horizon systems. However, for systems described as LFT's we will still require as a precondition for performance that when the driving input (the one associated with the performance condition) and the initial state are zero, then all internal signals are zero.

In what follows we will describe several robust performance problems, and we will recast them as a μ computation for an adequate constant matrix and block structure. For simplicity we will write the equations for a first order, 2 input, 2 output system considered over a 3 time steps horizon. However, they can all be applied to an arbitrary system, and the generalization of the formulas is straightforward. For signals that are functions of time, we will denote by a_k their value at time instant k and by $a = (a_1, a_2, \dots, a_T)$ the vector corresponding to their time history.

$l_2 \rightarrow l_2$ performance under parametric uncertainty

We will start with the discrete time version of the standard robust performance question that is answered with a μ test in the LTI, infinite time horizon case. Given a system as in Equations (7.2), and a partition of the inputs and outputs

$$\begin{aligned} u_k &= (u_k^0, u_k^1)^t \\ y_k &= (y_k^0, y_k^1)^t, \end{aligned}$$

we would like to answer the following

Question 7.1 *If*

$$\begin{aligned} u_k^1 &= \delta_k^1 y_k^1 \quad k = 0,1,2 \\ |\delta_k^1| &\leq 1, \end{aligned}$$

is it true that the following two conditions hold:

$$\left\{ \begin{aligned} x_0 &= 0 \\ u_k^0 &= 0 \quad k = 0,1,2 \end{aligned} \right. \Longrightarrow \left\{ \begin{aligned} x_3 &= 0 \\ u_k^1 &= 0 \quad k = 0,1,2 \\ y_k^0 &= 0 \\ y_k^1 &= 0, \end{aligned} \right.$$

and

$$|y^0|^2 + |x_3|^2 < |u^0|^2 + |x_0|^2 \ ?$$

Remark: The first condition is similar to the requirement of well posedness in infinite time horizon. We require that if the system is not driven, then all internal signals should remain zero. The second condition is the counterpart of the performance condition in the standard setup.

When the answer to Question 7.1 is yes, we say that the system meets the robust performance requirement. The answer to Question 7.1 is yes if and only if for every $\Delta_p \in \mathbf{R}^{4\times 4}$ with $\|\Delta_p\|_2 \le 1$, for all $\delta_i \in [-1,1]$, the following set of equations has only the trivial solution

$$\begin{aligned}(x_3, y^0, y^1)^t &= M_{[3]}(x_0, u^0, u^1)^t \\ (x_0, u^0)^t &= \Delta_p(x_3, y^0)^t \\ u_k^1 &= \delta_1 y_k^1 \quad k = 1,2,3\end{aligned}$$

where $M_{[3]}$ is constructed as it was shown in the preceding section. This is equivalent to $I - \Delta_1 M_{[3]}$ being invertible for every

$$\Delta_1 \in \boldsymbol{\Delta}_1 = \left\{\text{blockdiag}(\Delta_p, \delta_1^1, \delta_2^1, \delta_3^1) : \Delta_p \in \mathbf{R}^{4\times 4},\ \delta_k^1 \in \mathbf{R}\right\},$$

and thus by the definition of the structured singular value the answer to Question 7.1 is yes if and only if

$$\mu_{\Delta_1}(M_{[3]}) < 1.$$

Remark: It is not necessary for the parameters to vary with time. The temporal nature of the parameters is reflected by the uncertainty structure in $\boldsymbol{\Delta}_1$. This shows an important difference between the finite and infinite horizon cases. In the latter, the temporal nature of the uncertainty determines the test to perform (μ, frequency domain upper bound, state space upper bound). In the former the temporal nature of the uncertain operators is reflected in the block structure.

$l_\infty \to l_2$ performance under parametric uncertainty

Another possible performance requirement is a bound on the total energy in the output, given that the magnitude of the input, at each time instant, is bounded (the bound being either constant or a function of time). We will show how to set this performance requirement as a μ problem for a system with parametric uncertainty.

For the system in the preceding section we would like to answer now the following:

Question 7.2 *If*

$$\begin{aligned} u_k^1 &= \delta_k^1 y_k^i \\ |\delta_k^1| &\leq 1 \\ |u_k^0| &\leq 1 \quad k = 0,1,2 \\ |x_0| &\leq 1, \end{aligned}$$

is it true that the following two conditions hold:

$$\left\{ \begin{aligned} x_0 &= 0 \\ u^0 &= 0 \end{aligned} \right. \Longrightarrow \left\{ \begin{aligned} x_3 &= 0 \\ u_k^1 &= 0 \quad k = 0,1,2 \\ y_k^0 &= 0 \\ y_k^1 &= 0, \end{aligned} \right.$$

and

$$|y^0|^2 + |x_3|^2 < 1 \,?$$

This question can be answered by determining whether or not the following set of equations has nontrivial solutions

$$\begin{aligned} (x_3, y^0, y^1)^t &= M_{[3]}(x_0, u^0, u^1)^t \\ (x_0, u^0)^t &= (\delta^x, \delta_0^0, \delta_1^0, \delta_2^0)^t a \\ a &= \Delta(x_3, y_0^0, y_1^0, y_2^0)^t \\ u_k^1 &= \delta_k^1 y_k^1 \;\; k = 1,2,3 \end{aligned}$$

for all $\Delta \in \mathbf{R}^{1\times 4}$ with $\|\Delta\|_2 \leq 1$, for all $\delta_i^j \in [-1,1]$, $\delta^x \in [-1,1]$. Partition the matrix $R = M_{[3]}$ according to the partition in the input and output vectors

$$\left[x_0, u_0^0, u_1^0, u_2^0\right], \; \left[u_0^1, u_1^1, u_2^1\right], \; \left[x_3, y_0^0, y_1^0, y_2^0\right], \; \left[y_0^1, y_1^1, y_2^1\right],$$

and build the matrix

$$M_e = \begin{bmatrix} 0 & R_{11} & R_{12} \\ I_4 & 0 & 0 \\ 0 & R_{21} & R_{22} \end{bmatrix}.$$

Again from the definition of μ, the system will only have trivial solutions, and therefore the answer to Question 7.2 is yes if and only if

$$\mu_{\Delta_2}(M_e) < 1,$$

where

$$\Delta_2 = \left\{\text{blockdiag}(\Delta^y, \delta^x, \delta_i^0, \delta_i^1); \Delta^y \in \mathbf{R}^{1\times 4},\ \delta_k^j \in [-1,1]\right\}.$$

Affine systems

In some applications we need to set an affine performance specification. For example suppose the desired response to the input u^o is y^o. Given that there is bounded noise added to the command signal u^o, we would like to know whether or not the maximum possible distance from the desired trajectory is smaller than a preset amount d. If y is the output corresponding to a given input u, our performance requirement is met if and only if

$$\max_{|u-u^o|\le n} |y - y^o| < d.$$

In what follows we show how this question when asked of an uncertain system can be recast as μ problem.

For the system in the preceding sections, and a given set of signals u^o, x^i, y^o, x^f, we would like to answer the following

Question 7.3 *If*

$$\begin{aligned} |u^0 + u^o|^2 + |x_0 + x^i|^2 &< 1 \\ u_k^1 &= \delta_k^1 y_k^1 \quad k = 0,1,2 \\ |\delta_k^1| &\le 1, \end{aligned}$$

is it true that the following two conditions hold:

$$\begin{cases} x_0 &= 0 \\ u_k^0 &= 0 \quad k = 0,1,2 \end{cases} \Longrightarrow \begin{cases} x_3 &= 0 \\ u_k^1 &= 0 \quad k = 0,1,2 \\ y_k^0 &= 0 \\ y_k^1 &= 0, \end{cases}$$

and

$$|y^0 + y^o|^2 + |x_3 + x^f|^2 < 1\ ?$$

We proceed in the same fashion as in the previous cases. Define the signals

$$(\tilde{x}_3, \tilde{y}^0, \tilde{y}^1)^t = M_{[3]}(x^i, u^o, 0)^t.$$

The answer to Question 7.3 is yes if and only if the following system of linear equations

$$\begin{aligned} (x_3, y^0, y^1)^t &= M_{[3]}(\tilde{x}_0, \tilde{u}^0, u^1)^t + (x^f - \tilde{x}_3, y^o - \tilde{y}^0, -\tilde{y}^1)^t a \\ a &= \Delta_p (x_3, y^0)^t \\ (\tilde{x}_0, \tilde{u}^0)^t &= \Delta_i a \\ u_k^1 &= \delta_k^1 y_k^1 \quad k = 1,2,3 \end{aligned} \tag{7.3}$$

has only the trivial solution for all $\Delta_p \in \mathbf{R}^{1\times 4}$, $\Delta_i \in \mathbf{R}^{4\times 1}$ with $\|\Delta_p\|_2 \le 1$, $\|\Delta_i\|_2 \le 1$, and for all $\delta_i^1 \in [-1, 1]$. To prove this claim, note that if there exists a solution to these equations with $a \neq 0$, then there is one with $a = 1$. In this case the equations can be rewritten as

$$\begin{aligned} (x_3, y^0, y^1)^t &= M_{[3]}(x_0, u^0, u^1)^t + (x^f, y^o, 0)^t \\ \tilde{x}_0 &= x_0 + x^i \\ \tilde{u}^0 &= u^0 + u^o \\ a &= \Delta_p (x_3, y^0)^t \\ (\tilde{x}_0, \tilde{u}^0)^t &= \Delta_i a \\ u_k^1 &= \delta_k^1 y_k^1 \quad k = 1,2,3 \end{aligned} \tag{7.4}$$

and thus the second condition of Question 7.3 is violated. Similarly, the reader can verify that if a nontrivial solution with $a = 0$ exists, the first condition of the question is violated. Now build the matrix

$$M_e = \begin{bmatrix} \begin{matrix} x^f - \tilde{x}_3 \\ y^o - \tilde{y}^0 \end{matrix} & R_{11} & R_{12} \\ 1 & 0 & 0 \\ -\tilde{y}^1 & R_{21} & R_{22} \end{bmatrix},$$

and the set

$$\Delta_3 = \left\{ \text{blockdiag}(\Delta_p, \Delta_1, \{\delta_i^1\}_{i=0}^{i=2}) : \Delta_p \in \mathbf{R}^{1\times 4},\ \Delta_i \in \mathbf{R}^{1\times 4},\ \delta_i \in \mathbf{R} \right\},$$

then the system given by Equations (7.3) has only trivial solutions and therefore the answer to Question 7.3 is yes if and only if

$$\mu_{\Delta_3}(M_e) < 1.$$

Performance under uncertainty with memory

In the preceding sections, we described how different performance requirements could be evaluated with respect to parametric uncertainty. We will now show how we can evaluate robustness with respect to dynamic uncertainty. The dynamic operators we will consider can be either time invariant (that is their matrix has a Toeplitz structure), or time varying (the matrix is lower triangular), and they are bounded in the $l_2 \to l_2$ induced norm. For simplicity we will describe the time invariant case with an l_2 into l_2 performance specification, but as in the preceding sections the procedure is general.

Question 7.4 *If*

$$u_k^1 = \sum_{j=0}^{k} \delta_{k-j}^1 y_j^1 \quad k = 0, 1, 2$$

and

$$\sum_{j=0}^{2} |\delta_j^1|^2 \le 1,$$

is it true that

$$\begin{cases} x_0 &= 0 \\ u_k^0 &= 0 \quad k = 0,1,2 \end{cases} \Longrightarrow \begin{cases} x_3 &= 0 \\ u_k^1 &= 0 \quad k = 0,1,2 \\ y_k^0 &= 0 \\ y_k^1 &= 0 \end{cases}$$

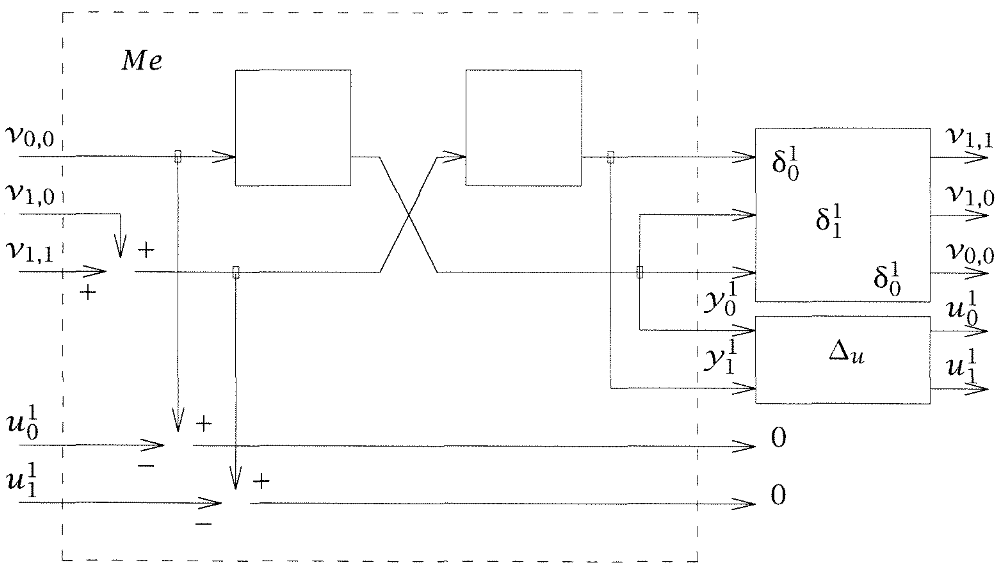

Figure 7.3: Dynamic uncertainty as an LFT with implicit equations.

and

$$|y^0|^2 + |x_3|^2 < |u^0|^2 + |x_3|^2?$$

Again, as in the preceding sections, it can be shown that the answer to Question 7.4 is yes if and only if the following system of equations

$$\begin{aligned}
(x_3, y^0, y^1)^t &= M_{[3]}(x_0, u^0, u^1)^t \\
(x_0, u^0)^t &= \Delta_p(x_3, y^0)^t \\
v_{k,j} &= \delta_{k-j} y_j^1 \\
u_k^1 &= \sum_{j=0}^{k} v_{k,j} \quad k = 0, 1, 2 \\
(u^1)^t &= \Delta_u (y^1)^t
\end{aligned} \tag{7.5}$$

has only trivial solutions for all $\Delta_p \in \mathbf{R}^{4\times4}$ with $\|\Delta_p\|_2 \le 1$, for all $\Delta_u \in \mathbf{R}^{3\times3}$ with $\|\Delta_u\|_2 \le 1$, for all $\delta_i \in [-1, 1]$. This system of equations can be represented by the diagram in Figure 7.3, where for simplicity we only represent two time steps. This diagram is different than the one in the preceding sections since it includes implicit equations.

To build the matrix corresponding to Figure 7.3, partition the matrix R as in the preceding section, and introduce the variables

$$w = [x_0, u^0, v, u^1]^t$$

$$z = [x_3, y_0^1, y_1^1, y_1^1]^t.$$

Define also the following two matrices

$$C = \begin{bmatrix} 0_{1\times 4} & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0_{1\times 4} & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0_{1\times 4} & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & -1 \end{bmatrix},$$

and M_e as shown in Figure 7.3. (Note in Figure 7.3 the two first set of inputs x_0 and u^0, and the two first set of outputs x_3 and y^0 have been omitted to simplify the diagram.) It can be verified that the system in Equations (7.5) have only trivial solutions if and only if

$$\ker\left(\begin{bmatrix} I - \Delta_4 M_e \\ C \end{bmatrix}\right) = \left\{\vec{0}\right\}$$

for all $\Delta_4 \in \mathbf{\Delta}_4$ where

$$\begin{aligned} \mathbf{\Delta}_4 &= \Big\{\text{blockdiag}(\Delta_p, \{\delta_{k,j}\}_{k=0,j=0}^{k=1,j=k}, \Delta_u), \\ &\quad \Delta_p \in \mathbf{R}^{4\times 4},\ \delta_{k-j}^1 \in \mathbf{R}, \Delta_u \in \mathbf{R}^{3\times 3}\Big\}. \end{aligned}$$

Following the definition in [24], the answer to Question 7.4 is yes if and only if

$$\mu_{C,\Delta_4}(M_e) < 1.$$

Remark: For each performance requirement, we can add different kinds of uncertainty. Thus for all the performance questions described, we can compute robustness with respect to any mix of dynamic and parametric uncertainty both time varying and time invariant over the horizon considered.

7.3 Computational Issues

Lower bound

It was shown in [35] that the computation of a lower bound for $\mu(M)$ can be tackled via a power iteration. Although in theory this algorithm can be used directly to establish a sufficient condition for the time domain performance specifications, it needs to be modified for practical considerations.

Extensive experimentation done with the power algorithm shows that the complexity of the algorithm is dominated by the multiplication of M and M^* with corresponding vectors. The cost of these operations is proportional to the square of the size of M. In time domain tests, the size of M grows linearly with the number of time steps considered. However, due to the special nature of the matrices involved, quadratic growth of the computation time with the number of time steps can be avoided.

In order to see how the structure of $M_{[k]}$ can be exploited to reduce the time complexity of the calculation, the multiplication

$$M_{[k]} \begin{pmatrix} x_0 \\ u_0 \\ \vdots \\ u_{k-1} \end{pmatrix}$$

is equivalent to computing the output of the system, when the initial condition is x_0 and the input is given by $u_0 \dots u_{k-1}$. Thus multiplication by $M_{[k]}$ can be done using the following sequence of operations

$$\begin{aligned} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} &= M_1 \begin{pmatrix} x_0 \\ u_0 \end{pmatrix} \\ &\vdots \\ \begin{pmatrix} x_k \\ y_k \end{pmatrix} &= M_k \begin{pmatrix} x_{k-1} \\ u_{k-1} \end{pmatrix}. \end{aligned}$$

The computation time for this set of operations grows linearly with the number of time steps considered. The multiplication by $M_{[k]}^*$ is similar and involves simulating backwards in time the transpose of the original system.

Upper bound

The same computational complexity problems arising in the lower bound arise in the computation of the upper bound. However, it is not as straightforward to use the structure in the matrix $M_{[k]}$ to reduce the growth of the computation time, when using current state of the art optimization algorithms for LMI's.

Using gradient search, we can develop an algorithm that is slower on small problems when compared to the LMI methods, but whose computation time does not degrade as much with the number of time samples. This is due to the fact that the complexity in computing the gradient depends on the number of repeated singular values, and this apparently is more strongly related to the order of the system than to the number of time samples taken. However, our tests of this algorithm are still preliminary and more extensive experimentation and further research is needed in this area.

7.4 Connections to the Frequency Domain Tests

If the system under consideration is LTI, the uncertainty description is repeated at each time instant and the performance condition is given as a full block mapping the final state to the initial state, we can develop some connections between the time domain tests and the corresponding frequency domain ones. From these connections we expect to derive a better understanding of the nature of the frequency domain tests for LTI systems.

Consider an n-dimensional linear time invariant system, defined by the equations

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k. \end{aligned}$$

Let

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

and

$$G(z) = zI_n * M,$$

where $*$ denotes the Redheffer star-product and let $\mathbf{\Delta}$ be an uncertainty structure. Let $M_{[k]}$ be the time domain mapping associated to G, with the following uncertainty structure

$$\mathbf{\Delta}_{[\mathbf{k}]} = \left\{ \text{blockdiag}(\underbrace{\Delta, \Delta, \cdots, \Delta}_{k}), \Delta \in \mathbf{\Delta} \right\}.$$

With these definitions we will have

$$M_{[k]} * \Delta_{[k]} = (M * \Delta)^k. \tag{7.6}$$

Denote

$$\begin{aligned} td(M,k) &= \mu_{\Delta_1}(M_{[k]}) \\ tdub(M,k) &= ub_{\Delta_1}(M_{[k]}), \end{aligned}$$

where

$$\boldsymbol{\Delta}_1 = \left\{\text{blockdiag}(\Delta_p, \Delta_u), \Delta_p \in \mathbf{C}^{nxn}, \Delta_u \in \boldsymbol{\Delta}_{[\mathbf{k}]}\right\},$$

and $ub_{\boldsymbol{\Delta}}$ denotes the $\mu_{\boldsymbol{\Delta}}$ upper-bound as defined in [22]. Finally, for an uncertainty set $\boldsymbol{\Delta}$ we will define the set $\mathbf{O}_{\mathbf{N}}\boldsymbol{\Delta}$ of operators with same block diagonal structure than the elements of $\boldsymbol{\Delta}$ but with each entry in the set $\mathcal{O}(\mathbf{C}^N)$ of operators defined from $\mathbf{C}^N$ into $\mathbf{C}^N$.

The following theorem is adapted from [22]. It connects the robust stability question posed for an LTI system in infinite horizon, to the performance of a finite time horizon system.

Theorem 7.1 *[22] Given a system M and an uncertainty structure Δ_u, then there exists $K > 0$ such that for all $k > K$, $td(M_{[k]}) < 1$, if and only if $\mu_{\boldsymbol{\Delta}}(M) < 1$.*

By using results on the lossless-ness of the S-procedure when the system is a constant matrix ([17], [24], [2]), we can generalize the preceding result to the upper bounds in the frequency and time domain respectively. The following theorem is essentially from [2]:

Theorem 7.2 *[2] Given a constant matrix M and an uncertainty structure $\boldsymbol{\Delta}$ then $ub_{\boldsymbol{\Delta}}(M) < 1$ if and only if $M * \Delta$ is well posed for all $\Delta \in \mathcal{O}_N\boldsymbol{\Delta}$, for all $N \in \mathbf{N}$.*

We will then have the following

Theorem 7.3 *There exists $K > 0$ such that*

$$tdub(M(\cdot), k) < 1 \ \text{ for all } k > K,$$

only if

$$fdub_{\Delta_u}(G(z)) < 1.$$

Proof: Assume $tdub(M(\cdot), k) < 1$.
Then according to the small gain theorem for all N for all $\Delta_{[k]} \in \mathcal{O}_N\Delta_{[k]}$

$$\|M_{[k]} * \Delta_u\| < 1.$$

From Equation (7.6) it follows that for all Δ in $\mathcal{O}_N\boldsymbol{\Delta}$, for all $k > K$

$$\|(M * \Delta)^k\| < 1. \tag{7.7}$$

Since for any operator A,

$$\rho(A)^k = \rho(A^k) \leq \|A^k\|,$$

Equation (7.7) implies that for all Δ in $\mathcal{O}_N\boldsymbol{\Delta}$

$$\rho(M * \Delta) < 1,$$

and thus for all $\delta \in \mathbf{C}, |\delta| = 1$, for all $\Delta \in \mathcal{O}_N\boldsymbol{\Delta}$

$$(\delta I_n * M) * \Delta_u$$

is well posed. Thus according to Theorem (7.2):

$$ub_{\Delta}(\delta I * M) < 1 \;\forall\; \delta \in \mathbf{C},\; |\delta| = 1$$

and thus

$$fdub_{\Delta}(M) < 1.$$

■

7.5 Conclusions

We present a setup in which performance of a time varying, finite time horizon linear system, under uncertainty described by quadratic constraints, can be tested. The performance conditions take the form of standard μ tests on constant matrices. However, it is not desirable to use directly the usual computation schemes for the μ lower and upper bounds, since those do not exploit the special structure of the matrix derived from the finite time horizon problem. We discussed a modification to the lower bound power algorithm that achieves linear growth in the computation time with the number of time steps considered. We also discussed some possible modification to the upper bound algorithms. Our results in this area are, however, preliminary. By using lossless-ness results for the S-procedure applied to constant matrices, we were also able to establish connections between the frequency domain tests and the limits of time domain tests.

Chapter 8

Improvements to the Power Algorithm for μ

The definition of μ involves an associated maximization problem which is not convex, so that it is difficult to compute μ exactly, and research has focused on the development and refinement of upper and lower bounds (see [8, 23] for example). The μ problem can be shown to be equivalent to a real spectral radius maximization problem, and from this one can develop a power iteration scheme to compute a lower bound. This was shown for the purely complex case in [23], and generalized to the mixed case in [35]. This power iteration affords an attractive method for computing a μ lower bound. The iteration scheme appears typically to have good convergence properties, and each iteration of the scheme is very cheap, requiring only such operations as matrix-vector multiplications and vector inner products, so that the resulting lower bound algorithm is very fast (see [35] for details).

Unfortunately, the lower bound power iteration is not always guaranteed to converge. Although one can still obtain a lower bound from the scheme in this case, it may no longer correspond to a local maximum of the real spectral radius, and so the bound may be poor. Results in [21] strongly suggest that the quality of the upper and lower bounds for μ are critical to the performance of any Branch and Bound scheme to refine them. This means that any improvement in the lower (or upper) bound performance is highly desirable, whether one wishes to use the bounds directly, or as part of a Branch and Bound scheme. This chapter presents some new approaches to computing an improved μ lower bound.

We would like an algorithm to compute a lower bound for μ that is fast, accurate, and reliable. For all iterative algorithms, however, there is always a tradeoff between speed, accuracy, and reliability. The power iteration algorithm for the lower bound is no exception to this rule, and there the penalty for having an algorithm that is in general fast and accurate is that there exist cases where it fails to converge to a solution. In this section we will describe one set of cases where this happens and how the algorithm can be modified to correct the problem at the cost of increasing the growth rate. We will also describe a connection between the equilibrium conditions at a maximum of $\rho(MQ)$ and the solution of μ for rank 1 matrices. We will derive from this connection a different way of updating the real part of Q that generalizes the method used for the complex part of Q. Finally, we will discuss how the use of an adaptive algorithm can be used to keep the performance and the growth rate of the standard power algorithm (SPA) for most cases, and achieve the performance of the modified algorithm when the SPA fails to converge.

8.1 Updating the Real Perturbation

In this section we present an alternative way to update the real part of the perturbation at each iteration of the power algorithm. This procedure involves the solution of μ for a rank one matrix and directly generalizes the complex case.

Throughout this section we will assume that the reader is acquainted with the results in Sections 5 and 6 of [35]. Every effort has been made to make the notation here consistent with the one in [35].

The complex power algorithm and the rank one solution

The power algorithm can be better understood by considering the solution to the maximization of $\rho(M_{r1}\Delta)$ for a certain rank 1 matrix M_{r1}. For the complex case it is particularly easy to prove this connection and it is formalized in the following

Theorem 8.1 *For a matrix M and a given block structure $\mathcal{K}(0, m_c, m_C)$, let $Q \in \mathcal{Q}_{\mathcal{K}}$, $D \in \mathcal{D}_{\mathcal{K}}$ and vectors a, b, c, d verify*

$$\begin{array}{rclcrcl} Mb & = & \beta a & & M^*z & = & \beta w \\ b & = & Qa & & b & = & D^{-1}w \\ z & = & Q^*QDa & & z & = & Q^*w, \end{array} \tag{8.1}$$

where the components of a, b, c, d partitioned according to the perturbation structure verify the following non-degeneracy condition $b_i, a_i, z_i, w_i \neq 0$ and $a_i^* w_i \neq 0$.

Then Q achieves the maximum of $\rho(M_{r1}Q)$ over $Q \in \mathcal{Q}_{\mathcal{K}}$, where $M_{r1} = aw^*$. Furthermore, any Q_1 solving this problem will also verify Equations (8.1).

Proof: From Lemmas 10 and 11 in [35] the components of the vectors a, b, z, w partitioned according to the perturbation structure verify

$$\begin{array}{rclr} z_i^c & = & \dfrac{w_i^{c*}a_i^c}{|w_i^{c*}a_i^c|}w_i^c & 1 \leq i \leq m_c \\ b_i^c & = & \dfrac{a_i^{c*}w_i^c}{|a_i^{c*}w_i^c|}a_i^c & \\ z_i^C & = & \dfrac{|w_i^c|}{|a_i^C|}a_i^C & 1 \leq i \leq m_C \\ b_i^C & = & \dfrac{|a_i^c|}{|w_i^C|}w_i^C. & \end{array} \tag{8.2}$$

Partitioning Q in the same way, and according to Equations (8.1), the parts of Q verify

$$\begin{array}{rclr} q_i^c & = & \dfrac{a_i^{c*}w_i^c}{|a_i^{c*}w_i^c|} & 1 \leq i \leq m_c \\ Q_i^C a_i^C & = & \dfrac{|a_i^c|}{|w_i^C|}w_i^C & 1 \leq i \leq m_C. \end{array} \tag{8.3}$$

These conditions imply

$$\begin{array}{rcl} w_i^{c*}q_i^c a_i^c & = & |w_i^{c*}a_i^c| \\ w_i^{C*}Q_i^C a_i^C & = & \|w_i^C\|\|a_i^C\|. \end{array} \tag{8.4}$$

It is easy to see that Equations (8.4) imply that Q maximizes

$$\rho(w^*Qa) = \rho(aw^*Q) = \rho(M_{r1}Q).$$

Conversely, any Q_1 that maximizes $\rho(M_{r1})$ has to verify Equations (8.4). This implies that

$$\begin{aligned} q_{1i}^c &= q_i^c \\ Q_i^C a_i^c &= Q_{1i}^C a_i^c \\ w_i^{c*} Q_i^C &= w_i^{c*} Q_{1i}^C. \end{aligned} \tag{8.5}$$

And Equations (8.5) can be rewritten as:

$$\begin{aligned} b &= Q_1 a \\ z &= Q_1^* w \end{aligned}$$

So Q_1 verifies also Equations 8.1. ■

The power algorithm consists thus conceptually of the following steps:

- Start with some given values for b, w.
- Update a with a power step of the form $\hat{\beta} a = Mb$.
- Compute the Q that maximizes $\rho(aw^*Q)$.
- Update z as $z = Q^*w$.
- Update w with a power step of the form $\tilde{\beta} w = M^*z$.
- Compute The Q that maximizes $\rho(aw^*Q)$.
- Update b as $z = Q^*a$.

Repeat the cycle.

The mixed-μ case

A similar though more restrictive theorem holds for the mixed-μ case.

Theorem 8.2 *For a matrix M and a given block structure $\mathcal{K}(m_r, m_c, m_C)$, let $Q \in \mathcal{Q}_{\mathcal{K}}$, $D \in \mathcal{D}_{\mathcal{K}}$ and vectors a, b, c, d verify*

$$\begin{aligned} Mb &= \beta a & \qquad M^*z &= \beta w \\ b &= Qa & b &= D^{-1}w \\ z &= Q^*QDa & z &= Q^*w, \end{aligned}$$

where the components of a, b, c, d partitioned according to the perturbation structure verify the following non-degeneracy condition $b_i, a_i, z_i, w_i \neq 0$ and $a_i^* w_i \neq 0$.

Then Q achieves the maximum of $\rho(M_{r1}Q)$ over $Q \in \mathcal{Q}_{\mathcal{K}}$, where $M_{r1} = aw^*$.

The proof of this theorem, though slightly more complex technically, follows the same lines as that of Theorem 8.1 and can be found in [34]. Finding the Q that maximizes $\rho(M_{r1}Q)$ in the real case is slightly harder than in the complex case. However, a simple algorithm exists that can solve this problem in linear time. This method is described in detail in [34].

The similarity of these two results suggests that the solution to the rank one problem can be used to update the real part of the perturbation as well. However, there are cases when not all solutions of the rank 1 mixed-problem are going to be equilibrium points of the original one. Namely, when two or more of the products $w_i^{r*} a_i^r$ are in phase (or in opposition of phase), several solutions exist, not all of which verify Equation (8.1). When this happens we need to have a procedure that chooses the correct one. We have not as yet looked in detail into that problem, but evidence suggests that such a procedure should exist. For the purpose of the numerical experiments in this chapter we use a very simple scheme that falls back into the standard way of updating the real perturbation (e. g., taking steps in the direction of the gradient) when the rank one solution scheme fails to converge.

8.2 Inverse Iteration

There is a set of matrices for which the convergence properties of the standard power algorithm are particularly poor. Consider the set $\mathcal{R}_{\mathcal{B}}$ of matrices that for a given uncertainty structure verify $\rho_R(MQ) \ll \rho(MQ)$ where Q achieves the maximum of $\rho_R(MQ)$. In this case the power step taken to update the eigenvectors will magnify the component along the spectral radius more than the component along the real spectral radius. A small error in the vectors a, b, w, z will thus tend to be amplified, and the algorithm, even if started close to the global maximum, will drift away.

Instead of updating a and w in the standard way, we can use a step of the form

$$\begin{aligned} x &:= (Q_k M - \beta_k I)^{-1} b_k \\ \beta_{k+1} a_{k+1} &:= M \frac{x}{\|x\|}. \end{aligned}$$

If Q is close to the equilibrium point, then QM has a real eigenvalue close to β. In this case the inverse of $Q_k M - \beta_k$ will have a very large gain along the direction of the corresponding eigenvector. This fact can be used to address the problem the power algorithm has with the mentioned set of matrices. However, in order for this way of updating the vectors to have a real effect on the algorithm, the vectors and value of β have to be close to the equilibrium values. A way to address this difficulty is treated in the following section.

Mixing inverse and power iteration

The power algorithm for the lower bound of μ is derived by writing a set of equations characterizing a local maximum of $\rho(MQ)$ and deriving from them a set of recursive formulas whose equilibrium point verifies those equations. There is not a unique way of deriving recursive formulas with such a property. Different formulations will lead to algorithms with different convergence properties. We will present in this section a possible formulation, that although it requires the use of higher computational complexity operations, has proven in practice to have better convergence behavior.

In order to do this we will introduce the following notation. We separate the perturbation Q in its real Q_1 and complex (both scalars and full blocks) Q_2 components

$$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}.$$

Analogously we partition the vectors a, b, c, d:

$$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \qquad z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \qquad w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

and the matrix M

$$M = \begin{pmatrix} M_{11} & M_{21} \\ M_{12} & M_{22} \end{pmatrix}.$$

We will then have the following

Theorem 8.3 *For a given matrix M and a given uncertainty structure, let $Q \in \mathcal{Q}_{\mathcal{K}}$, non-zero vectors a, b, c, d, and a positive scalar β such that $\beta I - M_{11}Q_1$ is non-singular and $b_2 \neq 0$, $z_2 \neq 0$. Assume also that all the diagonal elements of Q_1 are nonzero. Then the conditions:*

$$\begin{array}{rcl} Mb = \beta a & M^*z = \beta w & (8.6) \\ b = Qa & b = D^{-1}w & (8.7) \\ z = Q^*QDa & z = Q^*w & \end{array}$$

are equivalent to the conditions

$$\begin{array}{rcl} M_C b_2 = \beta a_2 & M_C^* z_2 = \beta w_2 & (8.8) \\ b_2 = Q_2 a_2 & b_2 = D_2^{-1} w_2 & (8.9) \\ z_2 = Q_2^* Q_2 D_2 a_2 & z_2 = Q_2^* w_2 & \\ Q_1(\beta I - M_{11}Q_1)^{-1} M_{12} b_2 & = \quad D^{-1}(\beta I - M_{11}^* Q_1)^{-1} M_{21}^* z_2 & (8.10) \end{array}$$

where $M_C = (M_{22} + M_{21}Q_1(\beta I - M_{11}Q_1)^{-1}M_{12})$

Proof: $(\Rightarrow)$

Rewrite Equation (8.6) in terms of the partitioned vectors

$$\begin{array}{rcll} \beta a_1 & = & M_{11}Q_1 a_1 + M_{12}Q_2 a_2 & (8.11) \\ \beta a_2 & = & M_{21}Q_1 a_1 + M_{22}Q_2 a_2 & (8.12) \\ \beta z_1 & = & Q_1 M_{11}^* z_1 + Q_1 M_{21}^* z_2 & (8.13) \\ \beta z_2 & = & Q_2^* M_{12}^* z_1 + Q_2^* M_{22}^* z_2. & (8.14) \end{array}$$

Solving for a_1 and z_1 in Equations (8.11) and (8.13) and substituting in Equations (8.12) and (8.14), we obtain Equation (8.8). Equations (8.9) and (8.10) derive directly from Equations (8.7), by partitioning the vectors and substituting a_1 and a_2 as before.

($\Leftarrow$)

Define $a_1 = (\beta I - M_{11}Q_1)^{-1}M_{12}b_2$ and $w_1 = (\beta I - M_{11}^*Q_1)^{-1}M_{21}^*z_2$. Define also $b_1 = Q_1a_1$ and $z_1 = Q_1w_1$. Then Equations (8.8) imply Equations (8.11) and (8.13). Finally Equations (8.9) and (8.10) imply Equation (8.7). ■

In the same way that the standard power algorithm is derived from conditions (8.6) and (8.7), another iterative algorithm can be derived from Equations (8.8) through (8.10). Namely, assuming the usual non-degeneracy conditions and using Lemmas 10 and 11 in [35], these conditions are equivalent to

$$\begin{aligned}
\beta a_2 &= (M_{22} + M_{21}Q_1(\beta I - M_{11}Q_1)^{-1}M_{12})b_2 \\
z_i^c &= \frac{w_i^{c*}a_i^c}{|w_i^{c*}a_i^c|}w_i^c \\
b_i^c &= \frac{a_i^{c*}w_i^c}{|a_i^{c*}w_i^c|}a_i^c \\
\beta w_2 &= (M_{22}^* + M_{12}^*(\beta I - Q_1M_{11}^*)^{-1}Q_1M_{21}^*)z_2 \\
z_j^C &= \frac{|w_j^C|}{|a_j^C|}a_j^C \\
b_j^C &= \frac{|a_j^C|}{|w_j^C|}w_j^C,
\end{aligned} \tag{8.15}$$

where $1 \le i \le m_c$ and $1 \le j \le m_C$ plus the conditions

$$\begin{aligned}
&Re(a_i^{r*}w_i^r) \ge 0 && \text{for} \quad q_i^r = 1 \\
&Re(a_i^{r*}w_i^r) \le 0 && \text{for} \quad q_i^r = -1 \qquad 1 \le i \le m_r \\
&Re(a_i^{r*}w_i^r) = 0 && \text{for} \quad |q_i^r| < 1,
\end{aligned} \tag{8.16}$$

where a_i^r and w_i^r are formed by partitioning the vectors

$$\begin{aligned}
a_1 &= (\beta I - M_{11}Q_1)^{-1}M_{12}b_2 \\
w_1 &= (\beta I - M_{11}^*Q_1)^{-1}M_{21}^*z_2
\end{aligned} \tag{8.17}$$

according to the perturbation structure, and the real components of b and z are computed as

$$\begin{aligned}
b_1 &= Q_1a_1 \\
z1 &= Q_1w_1.
\end{aligned}$$

An iterative algorithm that will have Equations (8.15) and (8.16) as equilibrium conditions is:

- Start with some given values for b, w and M_C.
- Update a_2 with the power step $\hat{\beta} a_2 = M_C b_2$.
- Compute the Q_2 that maximizes $\rho(a_2 w_2^* Q_2)$.
- Update z_2 as $z_2 = Q_2^* w_2$.
- Update w_2 with the power step $\tilde{\beta} w_2 = M_C^* z_2$.
- Compute The Q_2 that maximizes $\rho(a_2 w_2^* Q_2)$.
- Update b_2 as $z_2 = Q_2^* a_2$.
- Compute a_1 ,w_1 from Equation (8.17).
- Compute the Q that maximizes $\rho_R(aw^*Q)$ and update Q_1.
- Update β as $.5(\hat{\beta} + \tilde{\beta})$.
- Update $M_C = (M_{22} + M_{21}Q_1(\beta I - M_{11}Q_1)^{-1}M_{12})$.

Repeat the cycle.
Remarks: Although Theorem 8.3 represents a simple algebraic rearrangement of conditions (8.6) and (8.7), it has a great impact on the behavior of the power algorithm derived from them. In the standard power algorithm, the equilibrium vector a is not necessarily in the direction of the largest eigenvalue. If $\rho_R(MQ) < \rho(MQ)$ at the equilibrium point, SPA can not converge. As in the modified algorithm, the equilibrium vector a_2 corresponds always to the largest eigenvalue of $M_C Q_2$ and the mentioned obstacle to convergence has been removed.

If $\beta I - M_{11}Q_1$ is not invertible for the Q that achieves μ, then there is a perturbation Q' with $Q_1' = Q_1$ and $Q_2' = 0$ that achieves μ. In this case the μ problem is ill posed, not being at that point continuous on the elements of M. We will in general only be interested in problems where this does not happen, i. e., we will only consider problems where $\mu(M_{11}) < \mu(M)$.

The matrices M_C and Q_2 don't have to be formed as we are only interested in the products of these matrices with vectors. By optimizing the way we actually implement these operations, we can reduce the computation time.

Though the convergence properties of this scheme are better than for the previous one, this scheme is computationally more expensive. It is thus desirable to mix the procedures and only use the slower one in the cases were the other fails to converge. The scheduling of the different procedures can be done in more than one way. In the next section we present the results obtained with a very simple one. However, more numerical experimentation with more sophisticated scheduling schemes is desirable.

Updating Q_1 using the solution to the rank one problem does not have to converge always, since in some cases not all of its solutions will give equilibrium points for the original problem. So if this algorithm fails to converge, it is necessary to continue iterating using a different way of updating Q_1. We hope to solve this problem by getting a better understanding of the connections between the two problems. It is reasonable to expect that progress can be easily made in this area.

The step where β is updated can be used to ameliorate the convergence properties of the algorithm. For example, the value of β can be updated through a low pass filter (e. g., average the value of the last 2 cycles).

8.3 Experimental Results

Our aim was to find an algorithm that would preserve the numerical properties (convergence, accuracy, growth rate) of the power algorithm in those cases where the power algorithm behaves correctly and improve its convergence in the other cases without excessively degrading its speed and accuracy. A reasonable way to achieve this purpose is to have different algorithms of increasing accuracy and complexity, and a scheduling rule that will start with the fastest one and shift to the others progressively as they fail to converge. The results obtained using a simple scheduling rule are presented in this section.

Algorithm scheduling used

The scheduling rule used to obtain the results in the following section is rather simple. Our emphasis was put more into testing the principle of the adaptive algorithm rather than into finding an optimal scheduling scheme. The results obtained are nevertheless very encouraging.

The scheduling rule used is:

- Run the power algorithm for 50 iterations.
- If this fails to converge, run the mixed power algorithm for another 50 iterations, using the rank one solution to update the real perturbation.
- If this fails to converge, run the mixed power algorithm for another 50 iterations, using a gradient step to update the real perturbation.
- If this fails to converge, find a destabilizing perturbation around the latest point in the iteration.

Remarks: The limit number of iterations chosen reflects our numerical experience with the power algorithm. To make the algorithm more general, this number has to be made a function of the size of the problem.

It is possible to add further steps to this algorithm. Using gradient search has proven in our experience to be slow but reliable.

Results on $\mathcal{R}_{\mathcal{B}}$ matrices.

When used on random system matrices, the performance of the standard power iteration algorithm is more than satisfactory (see [35]). There is not a noticeable change in that performance by using the new algorithm discussed in this chapter. However, as it has been mentioned earlier, there is a set of matrices for which the convergence properties of the power algorithm are poor.

We tested the two algorithms on series of matrices in the set $\mathcal{R}_{\mathcal{B}}$. We generated matrices in this set that for a given uncertainty structure have a value of $\mu = 1$. We present here the results obtained for two different uncertainty structures. In order to compare the behavior of the two systems,

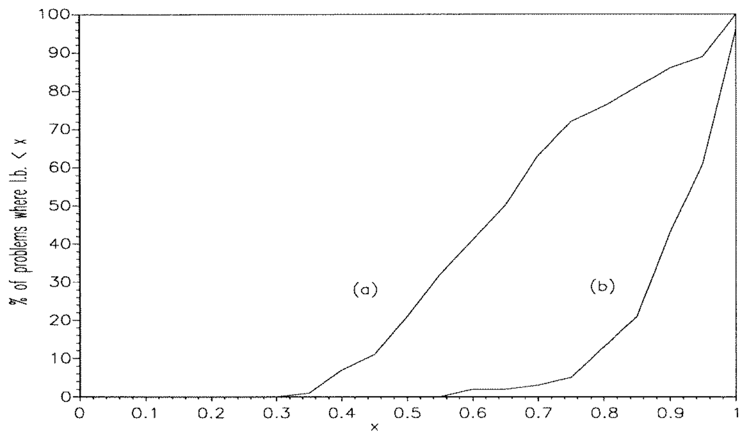

Figure 8.1: Comparison of the behavior of power iteration and mixed power iteration on matrices where $\rho_R(MQ) \ll \rho(MQ)$. The structure is 4 real parameters, 2 complex parameters and a 2 by 2 full block. (a) Standard power algorithm, (b) Modified power algorithm.

we will present the cumulative distribution of their answers for series of 100 matrices. Figure 8.1 shows that distribution for matrices of size 8 with an uncertainty structure of 4 real parameters, 2 complexes and a 2 by 2 full block. In this case over 50 percent of the answers given by the algorithm are larger than .9.

The performance degrades somehow when the number of real parameters increases. In Figure 2 we show the results obtained with matrices of size 12 with an uncertainty structure of 8 reals, 4 complex scalars and a 2 by 2 full block. However, in both cases the algorithm converged to an equilibrium point 95 percent of the time. (In the cases where the algorithm does not converge, we compute a perturbation that will give MQ a real eigenvalue and output that eigenvalue as a lower bound). And also in both cases the performance is increased significantly with respect to the original algorithm.

8.4 Conclusions

The algorithm in this chapter improves our capacity to efficiently compute a good lower bound for the structured singular value. However, it still is not guaranteed to converge to a local maximum in every case without

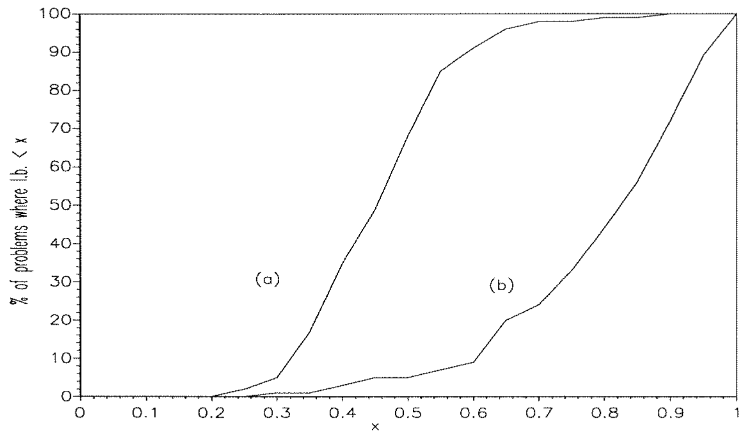

Figure 8.2: Comparison of the behavior of power iteration and mixed power iteration on matrices where $\rho_R(MQ) \ll \rho(MQ)$. The structure is 8 real parameters, 2 complex parameters and a 2 by 2 full block. (a) Standard power algorithm, (b) Modified power algorithm.

having to fall back into gradient search methods.

It can be shown that if instead of updating Q at every iteration in the algorithm as suggested in Section 8.1 we take a relaxed step of the form

$$Q_{k+1} = (1-t)Q_k + tQ_{r1},$$

where Q_{r1} is the solution to the rank one problem mentioned in the same section, and if the vectors a, b, z, and w and the scalars $\hat{\beta}$ and $\tilde{\beta}$ are sufficiently close to the equilibrium condition, then for t small enough, Q is updated in the direction of the gradient. This fact, together with a suitable scheduling algorithm for t could be used to guarantee the convergence of the algorithm without degrading its speed excessively.

Current research in system identification leads to μ problems on high order, highly structured matrices [27]. Future research will adapt this algorithm to the computation of lower bounds for these types of problems. By exploiting the structure of the problem we will attempt to reduce the growth rate of computation time with problem size, and in this way develop an algorithm capable of handling μ problems generated from practical System Identification applications (which may contain thousands of uncertain parameters).

Chapter 9
Concluding Remarks

In this last chapter we conclude this thesis by summarizing the results presented and by suggesting lines of future work.

9.1 Summary

The development of numerical tools for robustness analysis of linear systems has been successful. In the last 15 years these tools have been adopted by control engineers, to a large extent replacing the traditional graphical tools developed in the 1950's. Central to the success of these techniques has been the fact that robustness properties of linear systems can be established by computing functions of finite matrices.

The main motivation behind this thesis is to extend these methods to nonlinear problems. Due to the wide variety of behavior in nonlinear systems, in order to develop practical computation algorithms, it is necessary to work with a restricted class of problems and systems.

The performance problem we deal with in this work is robust trajectory tracking: a system is designed to complete a prespecified path in a known finite time. Since the real system is not exactly the one used for the design, and since it is also subject to noise, the system will not follow the intended trajectory. The question of interest becomes: will the real trajectory, under the worst conditions possible, remain close to the nominal one in an appropriate norm? In order achieve the same tradeoff between generality, applicability and computational efficiency that the structured singular value framework achieves for linear systems, we use the 2-norm as the measure for the noise signals, the under-modeled components gain, and the performance objectives. Since the underlying optimization problem is not convex — as is also the case in linear system analysis — we are not able to compute the worst case error. We have to settle for upper and lower bounds.

In Chapter 4 we develop an algorithm to compute a lower bound on the performance index. This algorithm is similar in nature to the power algorithm for μ and shares with it many of its numerical properties. Although the algorithm is not proven to converge, a numerical study of its behavior, carried out in Chapter 5, shows that it is well behaved when applied to some practical examples, and that it outperform alternative methods on those same problem.

To obtain an upper bound for the performance index, we need to have a finite global description of the system. Such a description can only be obtained for restricted classes of systems. In Chapter 6 we introduce a class of rational nonlinear systems that can be represented by a linear fractional transformation on a constant matrix. For systems in this class we prove the equivalence of the questions "Does an uncertain rational nonlinear system always meet a given performance specification?" and "Does a constrained linear uncertain system always meet a given performance specification?" From this equivalence we derived a convex upper bound for a large class of nonlinear performance problems that takes the form of a finite linear matrix inequality.

These two results constitute a first step in the direction of extending the tools for analysis of linear systems to nonlinear ones.

As a particular case, when the system is linear but time varying, we show that many interesting performance questions could be answered with μ tests if the system is considered over a finite time horizon. Although these systems could be tackled using the general nonlinear machinery presented before, linearity can be exploited to simplify computation, especially for the lower bound algorithms.

Finally, we present modifications of the standard power algorithms for the mixed μ problem that improve its behavior in harder to compute problems. These modifications are particularly relevant to our work, since all the problems derived from time domain analysis have uncertainty structures dominated by real components.

9.2 Future Research

As a first step in the direction of extending the μ analysis computations to nonlinear systems, the work in this thesis poses a set of new questions to be answered.

More experience is needed with the behavior of the lower bound power algorithm. It has to be tried in a larger variety of nonlinear systems, and with a larger array of uncertainty descriptions. It is important to understand the bounds on its performance in order to develop improvements for it; this is the way the power algorithm for linear systems was perfected. It is also important to gain experience with the setup for the robust trajectory tracking problem: how to choose uncertainty descriptions, time and frequency domain weights, noise bounds, and performance criteria.

The LMI resulting from the upper bound in Chapter 6, although large, is highly structured. The current LMI solvers do not exploit this structure. It is conceivable that by doing so, computation time can be reduced significantly. Even if these computation gains are achieved, the upper bound presented in this paper is still restricted to a limited class of systems and needs a global bound on the states. The search for better upper bounds is going to constitute the main thrust in this area of research.

Recent research has shown that for linear systems, model validation can be stated as a modified robust performance problem. The results in this thesis suggest that similar modifications can be made to the nonlinear analysis procedures to obtain nonlinear model validation along trajectories.

A combination of linear and nonlinear techniques can be used to systematize describing function analysis. The results of Chapter 6 can be applied to the harmonic analysis of systems whose nonlinearities have rational describing functions. These techniques will allow us to extend describing function analysis to multi-input, multi-output systems with several nonlinear components in a systematic and practical way.

Vale.

Bibliography

[1] Federal Aviation Administration. Advisory circular: Automatic landing systems, January 1971. AC No 20-57.

[2] H. Bercovici, C. Foias, and A. Tannenbaum. Structured interpolation theory. *Operator Theory: Advances and Applications*, 47:195–220, 1990.

[3] H. W. Bode. *Network Analysis and Feedback Amplifier Design.* D. Van Nostrand Company, Inc., 1945.

[4] R. D. Braatz, P. M. Young, J. C. Doyle, and M. Morari. Computational complexity of μ calculation. *IEEE Transactions on Automatic Control*, 39(5):1000–1002, 1994.

[5] A. E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control.* Halsted Press, 1975.

[6] H. Choi, P. Sturdza, and R. M. Murray. Design and construction of a small ducted fan engine for nonlinear control experiments. In *Proceedings of the American Control Conference*, pages 2618–2622, 1994.

[7] J. C. Doyle. Analysis of feedback systems with structured uncertainty. In *IEE Proceedings, Part D*, volume 129, pages 242–250, Nov. 1982.

[8] M. K. H. Fan and A. L. Tits. Characterization and efficient computation of the structured singular value. *IEEE Transactions on Automatic Control*, 31:734–743, 1986.

[9] M. K. H. Fan, A. L. Tits, and J. C. Doyle. Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics. *IEEE Transactions on Automatic Control*, 36:25–38, 1991.

[10] P. Gahinet, A. Nemiroskii, A. Laub, and M. Chilali. *The LMI Control Toolbox.* The MathWorks Inc., 1994.

[11] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for NPSOL.* Dept. of Operations Research, Stanford University, January 1986.

[12] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The John Hopkins University Press, 1983.

[13] W. M. Lu. *Control of Uncertain Systems: Convex Characterizations.* Ph.D. thesis, California Institute of Technology, 1994.

[14] W. M. Lu and J. C. Doyle. Robustness analysis and synthesis of nonlinear uncertain systems. Technical Report CIT-CDS-94-010, California Institute of Technology, 1994.

[15] D. G. Luenberger. *Optimization by Vector Space Methods.* John Wiley & Sons, 1968.

[16] D. McRuer, I. Ashkenas, and D. Graham. *Aircraft Dynamics and Automatic Control.* Princeton University Press, 1973.

[17] A. Megretski and S. Treil. Power distribution inequalities in optimization and robustness of uncertain systems. *Journal of Mathematical Systems, Estimation and Control,* 3(3):301–319, 1993.

[18] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming,* volume 13 of *Studies in Applied Mathematics.* SIAM, 1994.

[19] M. P. Newlin. *Model Validation, Control, and Computation.* Ph.D. thesis, California Institute of Technology, 1995.

[20] M. P. Newlin and S. Glavaski. Advances in the computation of the μ lower bound. In *Proceedings of the American Control Conference,* pages 442–446, 1995.

[21] M. P. Newlin and P. M. Young. Mixed μ problems and Branch and Bound techniques. In *Proceedings of the 29th Conference on Decision and Control*, pages 1230–1235. IEEE, 1990.

[22] A. Packard and J. C. Doyle. The complex structured singular value. *Automatica*, 29(1):71–109, 1993.

[23] A. Packard, M. K. H. Fan, and J. C. Doyle. A power method for the structured singular value. In *Proceedings of the 27th Conference on Decision and Control*, pages 2132–2137. IEEE, 1988.

[24] F. Paganini and J. C. Doyle. Analysis of implicitly defined systems. In *Proceedings of the 33rd Conference on Decision and Control*, pages 3673–3678. IEEE, 1994.

[25] M. G. Safonov. Stability margins for diagonally perturbed multivariable feedback systems. *IEE Proceedings, Part D*, 129:251–256, 1982.

[26] A. L. Schwartz. Ph.D. Dissertation, U. C. Berkeley. Document in preparation, 1996.

[27] R. S. Smith and J. C. Doyle. Towards a methodology for robust parameter identification. In *Proceedings of the American Control Conference*, pages 2394–2399, 1990.

[28] B. L. Stevens and F. L. Lewis. *Aircraft Control and Simulation*. John Wiley & Sons, 1992.

[29] J. E. Tierno and J. C. Doyle. Finite time horizon robust performance analysis. In *Proceedings of the 33rd Conference on Decision and Control*, pages 3080–3085. IEEE, 1994.

[30] J. E. Tierno and R. M. Murray. Robust performance analysis for a class of uncertain nonlinear systems. In *Submitted to the* $34^t h$ *CDC*. IEEE, 1995.

[31] J. E. Tierno, R. M. Murray, and J. C. Doyle. An efficient algorithm for performance analysis of nonlinear control systems. In *Proceedings of the 1995 American Control Conference*, pages 2717–2721, 1995.

[32] O. Toker and H. Özbay. On the NP-hardness of the purely complex μ computation, analysis/synthesis, and some related problems in multidimensional systems. In *Proceedings of the 1995 American Control Conference*, pages 447–451, 1995.

[33] A. J. van der Schaft. l_2-gain analysis of nonlinear systems and nonlinear state feedback h_∞ control. *IEEE Transactions on Automatic Control*, 37(6):770–784, 1992.

[34] P. M. Young. *Robustness with Parametric and Dynamic Uncertainty.* Ph.D. thesis, California Institute of Technology, 1993.

[35] P. M. Young and J. C. Doyle. Computation of μ with real and complex uncertainties. In *Proceedings of the 29th Conference on Decision and Control*, pages 1230–1235. IEEE, 1990.

[36] P. M. Young, M. P. Newlin, and J. C. Doyle. Practical computation of the mixed μ problem. In *Proceedings of the American Controls Conference*, pages 2190–2194, 1992.

[37] G. Zames. On the input-output stability of nonlinear time-varying feedback systems, parts I and II. *IEEE Transcations on Automatic Control*, 11:228–238 and 465–476, 1966.

[38] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice Hall, 1996.