Model Validation, Control, and Computation

Thesis by

Matthew Philip Newlin

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

California Institute of Technology
Pasadena, California
1996
Submitted September 22, 1995

Acknowledgments

First, I thank my advisor, Professor John Doyle, who contributed to my education more than I ever would have guessed. I would also like to thank my committee members, Professors Joel Burdick, Richard Murray, and Roy Smith. I greatly appreciate all their time and efforts.

Most of Chapter 2 is a summary of the work of others; it is presented because it provides the context for much of the remaining chapters. The rank one solution in Section 2.5, however, is new. The work in Chapter 3 was completed with Sonja Glavaski, while the work in Chapter 4 was shared with Peter Young. Chapters 5 and 6 were influenced by Peter Young and Roy Smith, respectively. A good deal of Chapter 7 is the work of Fernando Paganini and Raffaello D'Andrea.

Most importantly, I thank my friends and family, who have meant everything to me. None of this would have been possible without them.

Abstract

Engineering in general is concerned with controlling and predicting future behavior with some certainty despite having only imperfect information. Although feedback can be an exceptionally effective engineering tool and is often easy to apply, the behavior of a system under feedback can be extremely sensitive to model mismatch, which is always present. The potential for unpredictable behavior is a major drawback to the engineering application of feedback. Robust control theory addresses this difficulty by parametrizing a family of feedback controllers that are less sensitive to model mismatch.

Despite encouraging early applications, robust control theory has so far been deficient in analysis of systems, synthesis of controllers, connection to real problems, and applicability to nonlinear problems. Further, results on the computational complexity of robust control problems that necessitate either bounds computation or a restricted class of problems have cast doubts about the potential utility of the area.

Initial work in robust control focused on complex uncertainty in the frequency domain. A perceived deficiency is that such model sets are unrealistic: uncertainty in mass, stiffness, aero-coefficients, and the like are naturally modeled as real variations. This thesis includes initial work on practical upper bound computation and substantially improved lower bound computation for moderately large robust control analysis problems that include such real parametric uncertainty, despite the computational complexity of the problems. Although better upper bound computation than that described here is now available for small problems, such is not the case for large problems. The improved lower bound computation chronicled here is desirable because the initial lower bound computation for problems with real parametric uncertainty is not as reliable as in the complex case. Additionally, this thesis shows that branch and bound is a limited but critical tool for better computation, a fact that previously has gone unrecognized.

Together, these contributions allow for the practical computation of robust control problems of engineering interest and provide the basis not only for applications that may ultimately determine the utility of the robust control paradigm but also for the computation of various outgrowths of the μ

framework, which is the basis for computational robust control.

One such outgrowth is the model validation problem. Model validation tests whether a robust control model in the μ framework is consistent with experimentally determined time histories—quite a different problem than standard system identification. This thesis shows that the model validation problem is indeed closely related to the standard μ problem and its computation.

The practical computation of the model validation problem, which should follow naturally from the work presented here, provides the basis for the connection between robust control theory and practical applications. Future work along these lines should elevate the application of robust control theory from chance and intuition to a standard engineering tool.

Further, the techniques that render the model validation problem similar to the standard μ problem are applicable to a great variety of systems analysis and design problems. This newly perceived generality of the μ paradigm may ultimately provide a unifying framework for the many seemingly disparate aspects of systems and control design.

Table of Contents

List of Figures

Chapter 1

Introduction

Although our knowledge of real engineering problems remains incomplete, experience with real problems has led to an understanding of the underlying physics that describe real engineering problems and has led to engineering judgment about the limitations of typical applications of physics to real engineering problems. Because mathematical descriptions do not exactly represent real problems, we cannot eliminate the role of engineering judgment. Research in the field of control should lead to better engineering judgment; we do not merely want more tools and techniques, we want also to understand their use. Of particular concern in real engineering problems, perhaps the entire reason for engineering, is the prediction of future behavior. We want to do more than characterize the past. This concern is the principal motivation for model-based control rather than black box control based on some "intelligent" scheme with no explicit model, such as neural nets. Further, the engineer should be able to understand the model, in order to have some confidence in its predictive value.

The prediction of the future behavior of real engineering problems is difficult principally because mathematical models and problem statements cannot exactly represent a real engineering problem. In many instances this difference is minor and can be overlooked without concern. With feedback, however, even minor differences often must not be overlooked.

Since it is difficult both to develop theory for complicated performance objectives and to capture every concern of the engineer in a mathematical performance objective, the control theory applied to a real engineering problem is typically some easily specified optimization problem for a simple mathematical model. The optimal solution of such a problem compromises unspecified performance objectives and is frequently not suitable for use on the real problem. An extreme example of this deficiency occurs in optimal feedback control designed with an inexact model. Often, the "optimal" solution is worse than no control whatsoever and frequently drives a stable system unstable. In less extreme examples, the predicted behavior may still be far from reality. This is not a condemnation of feedback but of the haphazard application of feedback

to real problems; the careful application of feedback can provide dramatic improvements that are not easily unobtained any other way. Feedback can, however, be sensitive to modeling error.

Robust control theory addresses the issue of modeling error by requiring performance for every member in a family of models rather than for a single model. We might think of this as an attempt to include some "true" model that exactly describes the physical system in the set description. Since we do not want to be excessively conservative, we do not want extra elements in the set, and, by this reasoning, we do not want a set at all. Rather, we have a set because of our limited knowledge of the physical system and because we are willing to accept additional conservatism in order to have a simpler model. (A set description can easily be simpler than a description of some particular element in the set.) Further, heroic efforts to include the "true" model in the set might be pointless when we cannot exactly specify our true performance objectives.

Alternatively, we might think of the specification of the set of models as a way to parametrize a family of controllers. The idea is that if the family is rich enough, a controller that does well for every member of the family will not compromise too far in favor of performing particularly well for a single member of the family. Robust control theory mimics the effect of modeling error by requiring performance for each member in a family of models. Of course, accurately characterizing modeling error is difficult in the same way that accurately characterizing the model is difficult. From this point of view, the idea is not to cover the modeling error but rather to parametrize controllers that are typically insensitive to modeling error. Thus by specifying additional (possibly meaningless) performance objectives, we hope to satisfy meaningful but unspecified and perhaps mathematically unspecifiable requirements. Robust control theory concerns itself with the formulation and solution of such problems.

A particularly fruitful approach to robust control theory is the μ framework, described later, and its generalizations. The μ framework is appealing because it is both sufficiently general to include many other approaches and problems as special cases, and sufficiently structured to allow for practical computation of the general problem (rather than having a different method for each problem), while providing a relatively natural way to formulate engineering problems.

The application of the μ framework to real engineering problems has had two significant limitations: a restricted family of allowable uncertainty set descriptions; and little connection to the data beyond first principles modeling. The work presented here provides two principal contributions. The first is providing practical computation of both upper and lower bounds for the mixed

μ problem despite NP completeness results, thus providing a more useful family of allowable uncertainty descriptions. The second contribution is showing that the μ framework is easily extended to be more powerful, incorporating many more problems of engineering interest, particularly the inclusion of measured data in the problem formulation, allowing modeling and model validation to be part of the same framework and thus providing a practical way of connecting robust control models to real problems. While each of these two contributions is interesting in its own right, together they provide the foundation for the practical application of the μ framework to real problems.

An outline of the thesis, which presents these two contributions along with some introductory material and a look at what might lie ahead, follows.

The usefulness of the structured singular value μ, introduced by Doyle in [13], lies in the fact that many robustness problems can be recast as problems of computing μ with respect to some block structure. In recent years a great deal of interest has arisen with regard to robustness problems involving uncertain parameters that are not only norm-bounded but also constrained to be real. This type of problem falls within the μ framework by extending the original definition of μ to allow both real and complex uncertainties in the block structure (see Young et al., [43], for example). This mixed μ problem has fundamentally different properties than the complex μ problem, with important implications for computation. In the remainder of this chapter we give a brief introduction to this type of analysis and the engineering motivation behind it. The bulk of this work, however, is concerned with the theoretical and computational issues that subsequently arise. For more engineering motivation for the use of these approaches, see [27, 2, 14] and the references therein.

Chapter 2 outlines previous work on the computational nature of the problems considered subsequently and provides the motivation and context for the solutions presented here. It is now well known that real μ problems can be discontinuous in the problem data and that computation for the general mixed μ problem is NP hard. While the discontinuities have little relevance to problems motivated by engineering applications, the NP hardness results are extremely important, having adverse implications for certain research directions.

Roughly speaking, the fact that mixed μ is NP hard means that it cannot be computed exactly in the worst case without entirely unacceptable growth in computation cost with problem size. To obtain acceptable computation one is forced either to consider special cases or to relax the requirement for exact computation on worst-case problems. In Sections 2.3 and 2.4 we consider several special cases, including those of Kharitonov's theorem ([20]) and its extensions. These special cases are problems where μ is equal to a particular upper bound, which is relatively easy to compute. Unfortunately, these special

cases are of limited practical value because they fit few engineering problems, and the NP hardness results strongly suggest that they cannot be usefully extended.

Since the general mixed μ problem is NP hard, we do not attempt to solve it exactly but rather to obtain good bounds. Furthermore, recent results suggest that even approximate methods are also NP hard, so we do not expect good worst-case behavior but rather aim for good typical behavior.

Fortunately, engineering problems do not require exact mixed μ computation: mathematical models are only approximations of real physical systems, and the uncertainties cover the deficiencies in our knowledge of those systems. Thus it is somewhat naive to think that we can have precise knowledge of the uncertainty levels in real engineering problems. For engineering purposes, 10 to 20 percent accuracy is usually adequate, and it is more important to get answers in a reasonable amount of time than it is to achieve higher accuracy. It also matters little to the engineer that there are intractable problems so long as we can obtain reasonable solutions to problems of engineering interest. These problems may have many dozens of parameters, so algorithms with polynomial growth rate are highly desirable.

A power iteration to compute a lower bound for complex μ was generalized to the mixed case by Young and Doyle in [41]. Even though each iteration of the scheme is inexpensive, involving only matrix-vector multiplications and vector inner products, the algorithm's convergence properties are typically good. Unfortunately, the lower bound power iteration does not converge on a significant number of problems. Although one can still obtain a lower bound from the scheme in such cases, the bound may be poor. In Chapter 3 we present new approaches to computing an improved μ lower bound. Young's standard power algorithm (SPA) from [41] is our starting point. We develop several new algorithms after examining the convergence properties of the SPA. These algorithms are then combined to yield a substantially improved power algorithm. A comparison among the new algorithms presented here and those described in previous work is shown in Section 3.7.

The practical computation of an upper bound to mixed μ is presented in Chapter 4, involving the minimization of the eigenvalues of a Hermitian matrix. Algorithms for the computation of these bounds are described in Section 4.3. The quality of these bounds, along with their computational requirements as a function of problem size, is explored in Section 4.4.

Since the upper bound can be formulated as a linear matrix inequality (LMI) optimization problem, the continuing research in this area should ultimately render the algorithms of Chapter 4 obsolete. Unfortunately current LMI computation is inadequate for large problems with thousands of optimization parameters because the methods generally form pseudo-Hessians and have

numerical conditioning difficulties as well, so the algorithms in Chapter 4 are still relevant.

Chapter 5 investigates the practical application of branch and bound techniques to the computation of the mixed μ problem. A selection of results from a fairly extensive numerical study is presented. These numerical experiments suggest that if one is interested in solving fairly large problems, then one cannot expect the branch and bound scheme to achieve a greater degree of accuracy than the bounds usually produce alone, which in this case is approximately 20 percent. Thus the branch and bound scheme is being used not as a general computation scheme per se, but only to correct the occasional problems for which the bounds are poor, and for these problems, to achieve the degree of accuracy that the bounds typically get. This investigation emphasizes the necessity for good bounds. Fortunately, computing μ to within 20 percent accuracy is generally quite adequate for engineering purposes. The results in Chapter 5 suggest a clearly defined role for branch and bound techniques in mixed μ computation.

The model validation problem, originally formulated in the robust control context by Smith and Doyle in [35], provides a connection between control theory and reality: the model validation problem is to determine whether there is an element of the robust control model set that accounts for the experimental observation.

In Chapter 6 we present a generalization of μ, denoted by μ_g, that solves the model validation problem. Because the similarity to μ is pronounced, the approach taken for the calculation of μ_g involves the development of more readily calculated upper and lower bounds that are also quite similar to the upper and lower bounds for μ. Further, many of the techniques described in Chapters 3 and 4 are readily applicable to the computation of μ_g.

Chapter 7 puts forth the argument that the μ framework and its generalizations are applicable to a broader range of problems and consequently provide a unified framework for the application of modeling and control. Whether the unified framework is ultimately useful depends on future research, particularly in the computation of the implicit formulation presented there. While the implicit formulation is more general, the computation presented in preceding chapters should prove useful in the implicit case.

1.1 Notation

If a and b are real numbers with $a < b$, then (a, b) is an open interval and $[a, b]$ is a closed interval, while $\{a, b\}$ is the set containing the two elements a and b. The set of real numbers is denoted by $\mathbb{R}$, and the set of complex numbers is denoted by $\mathbb{C}$. $\mathbb{R}^+ = \{\, a \mid 0 < a \in \mathbb{R} \,\}$, and the closure is denoted

with an overbar: $\overline{\mathbb{R}^+} = \{ a \mid 0 \leq a \in \mathbb{R} \}$.

The transpose and conjugate transpose of a matrix, A, are denoted by A^T and A^* respectively. The spectral radius is denoted by $\rho(A)$, while the largest real eigenvalue and the absolute value of the largest magnitude real eigenvalue are denoted by $\overline{\lambda_r}(A)$ and $\rho_r(A)$. If A is Hermitian the subscript r may be dropped. The maximum and minimum singular values of A are denoted by $\overline{\sigma}(A)$ and $\underline{\sigma}(A)$. The dimension of A is $\dim(A)$, and the kernel of A is $\ker(A)$.

The norm of a vector, $\|w\|$, is the Euclidean norm, and $<x, y>$ is the inner product of the vectors x and y. The induced vector norm is used in the matrix case:

$$\|A\| = \sup_{\|w\|=1} \|Aw\| = \overline{\sigma}(A).$$

The norm of a signal, w, is taken to be the 2-norm,

$$\|w\| = \left[\int_{-\infty}^{\infty} \|w(t)\|^2 \, dt \right]^{1/2},$$

and the set of bounded energy signals is denoted by $\mathcal{L}_2$:

$$\mathcal{L}_2 = \{ w \mid \|w\| < \infty \}.$$

The unit ball of $\mathcal{L}_2$ is denoted by $\mathbf{B}_{\mathcal{L}_2}$. Similarly, if $\mathbf{\Delta}$ is a normed set, then $\mathbf{B}_{\mathbf{\Delta}}$ is the subset of elements with norm less than or equal to 1.

Frequently, our vectors are in product spaces, e.g., $\mathbb{X} = \mathbb{X}_1 \times \ldots \times \mathbb{X}_n$, naturally defining a partitioning of vectors in $\mathbb{X}$. If $\mathbb{X}$ and $\mathbb{Z}$ both have such a structure, then an operator $M : \mathbb{X} \longrightarrow \mathbb{Z}$ also has a partitioning, and $M_{ij} : \mathbb{X}_j \longrightarrow \mathbb{Z}_i$ denotes the resulting "blocks" of M. We similarly partition the spaces that the operators live in. Conversely, a collection of x_j or M_{ij} defines the relevant product spaces.

Often we group some of these subspaces together and denote this with a new subscript. For example, if $J = \{1, 2\}$, then $\mathbb{X}_J = \mathbb{X}_1 \times \mathbb{X}_2$, and $M_{iJ} : \mathbb{X}_J \longrightarrow \mathbb{Z}_i$.

A block diagonal operator is one where $i \neq j \implies M_{ij} = 0$. In this case we define $M_i = M_{ii}$. For a block diagonal operator, the input and output spaces $\mathbb{X}$ and $\mathbb{Z}$ must be partitioned into the same number of blocks.

We also have occasion to use subscripts to denote different elements of a set, with no particular product space implied.

Block Diagrams and LFTs

Block diagrams represent sets of equations and interconnections of systems. The block diagram for the generic robust control model shown in Figure 1.1

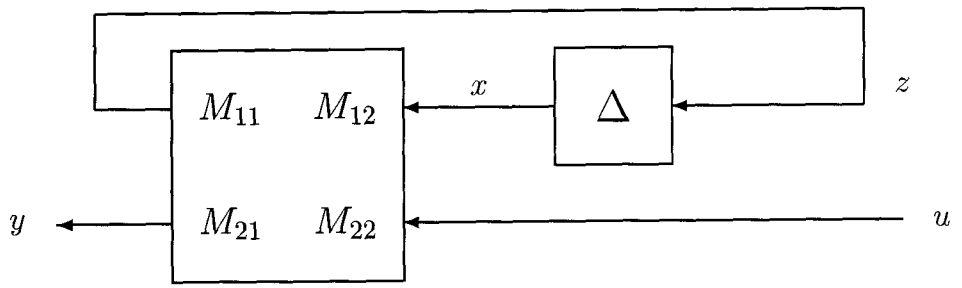


Figure 1.1: The generic robust control model structure.

represents the equations

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

We can eliminate z and x from these equations and solve for y in terms of M, Δ, and u, resulting in

$$y = (M_{22} + M_{21}\Delta(I - M_{11}\Delta)^{-1}M_{12})u. \tag{1.1}$$

This operator is a linear fractional transformation (LFT) on M and is referred to as the star product of Δ and M. Equation (1.1) is abbreviated to

$$y = (\Delta \star M)u.$$

Similarly, $M \star \Delta = M_{11} + M_{12}\Delta(I - M_{22}\Delta)^{-1}M_{21}$.

The LFT $\Delta \star M$ is said to be well posed if and only if there is a unique solution to the loop equations shown in Figure 1.1. It is easy to see that the LFT $\Delta \star M$ is well posed if and only if $(I - M_{11}\Delta)$ is invertible.

If $\Delta \star M$ is well posed, then the only solution to the equations $(I - M_{11}\Delta)z = 0$ and $(I - \Delta M_{11})x = 0$ is $z = x = 0$. If $I - M_{11}\Delta$ is not invertible, then there are infinitely many solutions, with $\|z\|$ and $\|x\|$ arbitrarily large. The equations $(I - M_{11}\Delta)z = 0$ and $(I - \Delta M_{11})x = 0$ are equivalent to the equations

$$\begin{aligned} z &= M_{11}x \\ x &= \Delta z, \end{aligned}$$

which are in turn equivalent to Figure 1.2. This figure is a special case of Figure 1.1 in that the equations representable by Figure 1.2 are a subset of the equations representable by Figure 1.1 with $z = 0$.

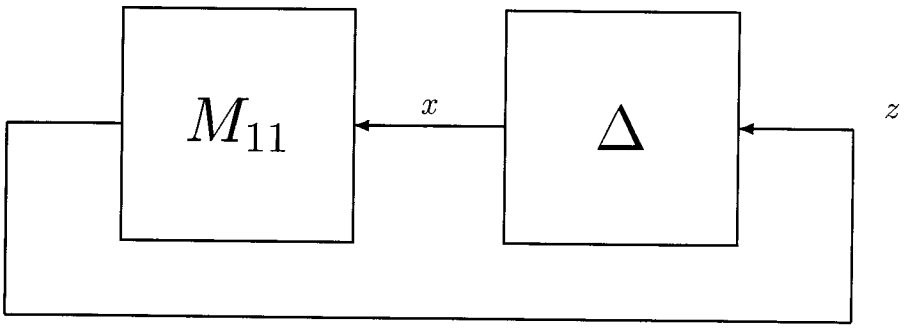


Figure 1.2: The robust stability μ analysis framework.

If the LFT $\Delta \star M$ is not well posed, then the nonzero solutions to the equations in Figure 1.2 are solutions to the equations in Figure 1.1 with $u = 0$ and y typically nonzero. Thus, when the LFT $\Delta \star M$ is not well posed, the gain from the input u to the other signals in Figure 1.1 is infinite, and the feedback loop is in some sense unstable.

Although $\Delta \star M_{11}$ is not an operator and cannot be defined the way that $\Delta \star M$ is, we may still refer to the well posedness of $\Delta \star M$ as the stability and well posedness of $\Delta \star M_{11}$.

$$y + \frac{c}{m}\dot{y} + \frac{k}{m}y = \frac{f}{m}$$

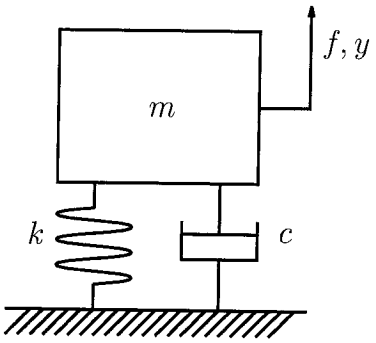


Figure 1.3: A mass-spring-damper system.

Transfer functions are an important example of LFTs. Consider the state space realization of a discrete time system

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} = M \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \tag{1.2}$$

This system's transfer function,

$$G(z) = D + C(zI - A)^{-1}B = \frac{I}{z} \star M,$$

is an LFT on $\frac{1}{z}I$, a repeated complex scalar. If λ is an eigenvalue of A, then the LFT is not well posed at $z = \frac{1}{\lambda}$, just as $\frac{1}{z-a}$ is not well posed at $z = a$.

Systems having uncertainty also can be represented easily using LFTs. A natural type of uncertainty is unknown coefficients in a state space model. The following simple example is taken from Packard and Doyle ([27]). Begin with a familiar idealized mass-spring-damper system as shown in Figure 1.3. Suppose m, c, and k are fixed but uncertain, with

$$\begin{aligned} m &= m(1 + w_m\delta_m) \\ c &= c(1 + w_c\delta_c) \\ k &= k(1 + w_k\delta_k), \end{aligned}$$

where δ_m, δ_c, and δ_k are all uncertain real scalars that are known to lie in the interval $[-1, 1]$ but are otherwise unknown. The known parameters m, c, and k may be thought of as the nominal values of m, c, and k respectively, and the known weights w_m, w_c, and w_k serve to normalize the uncertainty range to the unit ball.

Then, by defining $x_1 = y$ and $x_2 = \dot{y}$, we can write the differential equation in state-space form with $\Delta = \text{diag}(\delta_m, \delta_c, \delta_k)$ as

$$\begin{bmatrix} \dot{x} \\ y \end{bmatrix} = (\Delta \star M) \begin{bmatrix} x \\ f \end{bmatrix}$$

$$M = \begin{bmatrix} -w_m & \frac{-w_c}{m} & \frac{-w_k}{m} & \frac{-k}{m} & \frac{-c}{m} & \frac{1}{m} \\ 0 & 0 & 0 & 0 & c & 0 \\ 0 & 0 & 0 & k & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -w_m & \frac{-w_c}{m} & \frac{-w_k}{m} & \frac{-k}{m} & \frac{-c}{m} & \frac{1}{m} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

More generally, the perturbed state-space system

$$\begin{aligned} x_{k+1} &= A(\delta)x_k + B(\delta)u_k \\ y_k &= C(\delta)x_k + D(\delta)u_k, \end{aligned}$$

where δ is a vector of parameters that enter rationally, can be written as an LFT on a diagonal matrix Δ made up of the (possibly repeated) elements of δ. The one exception is that $\frac{1}{\delta}$ cannot be represented as an LFT because LFTs are well posed at $\delta = 0$.

A fundamental property of LFTs that contributes to their importance in linear systems theory is that interconnections of LFTs are again LFTs. For example, suppose we have three components, each with an LFT uncertainty model, interconnected as in Figure 1.4. By simply reorganizing the diagram, collecting all of the known systems together, and collecting all of the perturbations (the various Δ_i) together, we get the diagram shown in Figure 1.5, where M depends on G_1, G_2, G_3 and the diagram layout in Figure 1.4, but not on the Δ_i or the d_i, y_i, or u. Note that the feedback operator in Figure 1.5 is block diagonal.

It is simple to find the matrix M in Figure 1.5: the arrows in Figure 1.5 represent the same vectors represented by the arrows in Figure 1.4, and the M_{ij} block of M is the operator from the j^{th} vector on the input side of M to the i^{th} vector on the output side, with all other vectors on the input side of M and all other operators set to zero. Figure 1.5 and Figure 1.4 represent the exact same equations.

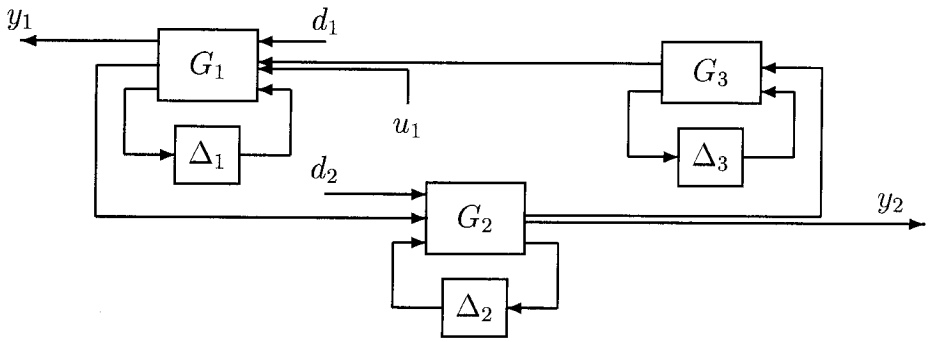


Figure 1.4: An example interconnection of LFTs.

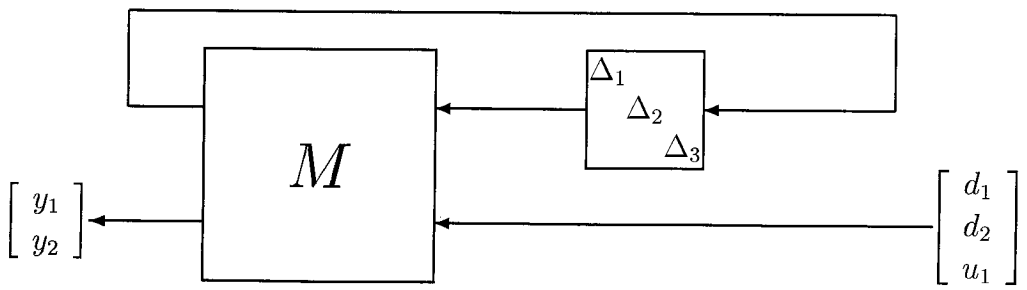


Figure 1.5: The macroscopic representation of Figure 1.4.

Because additive and multiplicative uncertainty, $G + \Delta$ and $G(I + \Delta)$ respectively, are special cases of linear fractional uncertainty descriptions, we can convert problems with additive and/or multiplicative uncertainty, or indeed any interconnection of systems with uncertainty entering in a linear fractional manner, into the standard framework of Figure 1.2.

1.2 A Review of μ: Robust Stability and Robust Performance

The definition of μ is dependent upon $\mathbf{\Delta}$, the underlying block structure of the uncertainties. Typically $\mathbf{\Delta}$ is a set of block diagonal operators defined in terms of simpler sets corresponding to the partioning of the input and output spaces. Typically, this partitioning is the same for all $\Delta \in \mathbf{\Delta}$. We partition the set of subscripts into three disjoint subsets $\mathbf{r}$, $\mathbf{c}$, and $\mathbf{C}$, corresponding to three types of operators. The sets $\mathbf{\Delta}_r$ have the form

$$\{ \delta_r I \mid \delta_r \in \mathbb{R} \},$$

the sets $\mathbf{\Delta}_c$ have the form

$$\{ \delta_c I \mid \delta_c \in \mathbb{C} \},$$

and the sets $\mathbf{\Delta}_C$ are sets of all complex matrices of the correct dimension. These blocks of the block structure are referred to as repeated real scalar blocks or real blocks, repeated complex scalar blocks, and full complex blocks or full blocks, respectively. When the set $\mathbf{r}$ is empty, we have a complex μ problem; when $\mathbf{c}$ and $\mathbf{C}$ are empty, we have a real μ problem; otherwise, we have a mixed μ problem.

Definition 1.1 ([13]) *The structured singular value, $\mu(M)$, of a constant complex matrix M with respect to a block structure $\mathbf{\Delta}$ is defined as*

$$\mu(M) \triangleq \max \left\{ 0 \bigcup \{ \overline{\sigma}(\Delta)^{-1} \mid \Delta \in \mathbf{\Delta}, \det(I - \Delta M) = 0 \} \right\}. \quad (1.3)$$

It is easy to see from the definition that the LFT $\Delta \star M$ in Figure 1.1 is well posed for all $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$ if and only if $\mu(M_{11})$ is less than 1. Thus μ is defined to be the answer to a large family of robust stability problems.

Sometimes each Δ_j is a full complex block, and is not necessarily square. An equivalent definition of μ in this case, introduced by Fan and Tits in [15], is now given because it is the basis for the definition of μ_g in Chapter 6. Let $z = Mx$ and let J be the set of indices of the partitioning of M.

$$\begin{aligned} \mu(M) &\triangleq \max_{\|x\|=1} \{ \gamma \mid \|x_j\| \gamma \leq \|z_j\|, \forall j \in J \} \\ &= \max_{\|x\|=1} \{ \gamma \mid \|x_j\| \gamma = \|z_j\|, \forall j \in J \}. \end{aligned} \tag{1.4}$$

The equivalence to the second maximization problem, where the inequality is replaced by equality, is due to Doyle ([13]).

While μ does provide an exact test for robust stability, it is not obvious from Definition (1.3) how the value of μ may be computed. However, it is easy to obtain the following bounds:

$$\rho_r(M) \leq \mu(M) \leq \overline{\sigma}(M). \tag{1.5}$$

These bounds by themselves are usually too crude for our purposes, but they can be refined as shown in Chapters 3, 4, and 5.

We define the following sets of block diagonal matrices (which depend on the block structure of $\mathbf{\Delta}$) to refine the bounds for μ in Equation (1.5) and to apply branch and bound to the μ problem. These sets are used in subsequent chapters; they are collected here for easy reference.

$$\begin{aligned}
\mathbf{B}_{\mathbf{\Delta}} &= \{ \Delta \mid \Delta \in \mathbf{\Delta}, \delta_r \in [-1,1], \|\delta_c\| \leq 1, \overline{\sigma}(\Delta_C) \leq 1 \} \\
\mathbf{B}_{\mathbf{\Delta}_i} &= \{ \Delta \mid \Delta \in \mathbf{\Delta}, \delta_r \in [c_r - r_r, c_r + r_r], \|\delta_c\| \leq 1, \overline{\sigma}(\Delta_C) \leq 1 \} \\
\mathbf{B}_{\mathbf{\Delta}_{\mathbb{C}}} &= \{ \Delta \mid \Delta \in \mathbf{\Delta}, \delta_r \in \mathbb{C}, \|\delta_r\| \leq 1, \|\delta_c\| \leq 1, \overline{\sigma}(\Delta_C) \leq 1 \} \\
\mathbf{B}_{\mathbf{\Delta}_1} &= \{ \Delta \mid \overline{\sigma}(\Delta) \leq 1 \} \\
\partial\mathbf{B}_{\mathbf{\Delta}} &= \{ \Delta \mid \Delta \in \mathbf{\Delta}, \delta_r \in [-1,1], \delta_c^* \delta_c = 1, \Delta_C^* \Delta_C = I \} \\
\partial\mathbf{B}_{\mathbf{\Delta}_i} &= \{ \Delta \mid \Delta \in \mathbf{\Delta}, \delta_r \in [c_r - r_r, c_r + r_r], \delta_c^* \delta_c = 1, \Delta_C^* \Delta_C = I \} \\
\mathbf{Q} &= \{ \Delta \mid \Delta \in \mathbf{\Delta}, \delta_r \in \{-1,1\}, \delta_c^* \delta_c = 1, \Delta_C^* \Delta_C = I \} \\
\widehat{\mathbf{D}} &= \left\{ \operatorname{diag}(e^{j\theta_r} D_r, D_c, d_C I) \;\middle|\; 0 < D_i, i \in \mathbf{r} \bigcup \mathbf{c}, d_C \in \mathbb{R}^+ \right\} \\
\mathbf{D} &= \left\{ D \;\middle|\; D = D^* \in \widehat{\mathbf{D}} \right\} \\
\mathbf{G} &= \{ \operatorname{diag}(G_r, 0) \mid G_r = G_r^*, r \in \mathbf{r} \} \\
\widehat{\mathbf{G}} &= \{ G \mid G \in \mathbf{G} \text{ is diagonal} \}.
\end{aligned}$$

The following two theorems, which address robust stability and performance questions for LFT feedback interconnections, were proved for complex uncertainty descriptions by Packard and Doyle in [27]. Let $\mathbf{\Delta}$ be partitioned into $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$, defining μ_1 and μ_2 respectively.

Theorem 1.2 (Well Posedness) *Let $\beta > 0$. The LFT $\Delta_1 \star M$ is well posed for $\left\{ \Delta_1 \;\middle|\; \Delta_1 \in \mathbf{\Delta}_1, \overline{\sigma}(\Delta_1) \leq \frac{1}{\beta} \right\}$ if and only if $\mu_1(M_{11}) < \beta$.*

Theorem 1.3 (Main Loop Theorem) *Let $\beta > 0$. Then $\mu(M) < \beta$ if and only if both $\mu_1(M_{11}) < \beta$ and $\Delta_1 \in \left\{ \Delta_1 \;\middle|\; \Delta_1 \in \mathbf{\Delta}_1, \overline{\sigma}(\Delta_1) \leq \frac{1}{\beta} \right\}$ implies $\mu_2(\Delta_1 \star M) < \beta$.*

The proofs are identical to those given in [27] for the complex case (a number of minor variations to these theorems similarly extend to the mixed case). For a more complete feedback interpretation of these results, see [27] and the references therein.

The following theorems address the problems of stability and performance, for all elements of the model set, in the frequency domain. See [27], for example, for more details.

Theorem 1.4 *Consider Figure 1.1. Let $M(s)$ and $\Delta(s)$ be linear time-invariant and stable. Then*

$$(\Delta(s) \star M(s)) \text{ is stable } \forall \Delta(s) \in \mathbf{B}_{\mathbf{\Delta}} \iff \sup_{\omega} \mu(M_{11}(j\omega)) < 1,$$

where the block structure for the μ calculation is defined by $\mathbf{B}_{\mathbf{\Delta}}$.

Theorem 1.5 *Consider Figure 1.1. Let $M(s)$ and $\Delta(s)$ be linear time-invariant and stable. Then*

$$\|\Delta(s) \star M(s)\|_\infty < 1 \; \forall \; \Delta(s) \in \mathbf{B}_{\mathbf{\Delta}} \iff \sup_{\omega} \mu(M(j\omega)) < 1,$$

where the block structure, $\widehat{\mathbf{\Delta}}$, for the μ calculation is

$$\widehat{\mathbf{\Delta}} = \left\{ \text{diag}\left(\mathbf{\Delta}, \mathbb{C}^{\dim(u) \times \dim(y)} \right) \right\}.$$

These two theorems are sufficient to motivate the (complex) constant matrix μ problem, where the operators are constant matrices with elements in $\mathbb{C}$ and the signals are constant vectors, also with elements in $\mathbb{C}$. The fact that these two theorems are still true in the mixed case means that mixed μ still provides an exact test for robust stability and robust performance problems, but now with real uncertainties allowed. The LFT machinery for rearranging various robustness problems into μ problems (such as the conversion of robust performance problems into robust stability problems, as above) is also applicable to the mixed case, so that mixed μ retains its versatility and applicability to a large class of problems.

1.3 LFT Examples

This section contains a few simple examples of the conversion of problems to a standard LFT form. The reformulations are both illustrative and will also be useful later.

Quadratic Programming

The indefinite quadratic programming problem given by

$$\text{Is} \max_{b_l \le x \le b_u} \|x^* A x + p^* x + c\| \ge 1? \tag{1.6}$$

for $A \in \mathbb{R}^{n \times n}$, x, p, b_l, and $b_u \in \mathbb{R}^n$, and $c \in \mathbb{R}$ is easily reformulated to replace the set $b_l \le x \le b_u$ with the set $-1 \le y_i \le 1$ so the following problem is equivalent.

$$\text{Is} \max_{-1 \le y_i \le 1} \|y^* M_{21} y + M_{31}^* y + M_{33}\| \ge 1? \tag{1.7}$$

for $M_{21} \in \mathbb{R}^{n \times n}$, $M_{31} \in \mathbb{R}^n$, and M_{33} and $y_i \in \mathbb{R}$. Then, defining $M_{32} = [1 \ldots 1] \in \mathbb{R}^n$, $M_{13} = M_{32}^T$, and $\Delta = \text{diag}(y_i)$, the real μ problem Is $\mu(M) \ge 1$? depicted in Figure 1.6 is also equivalent.

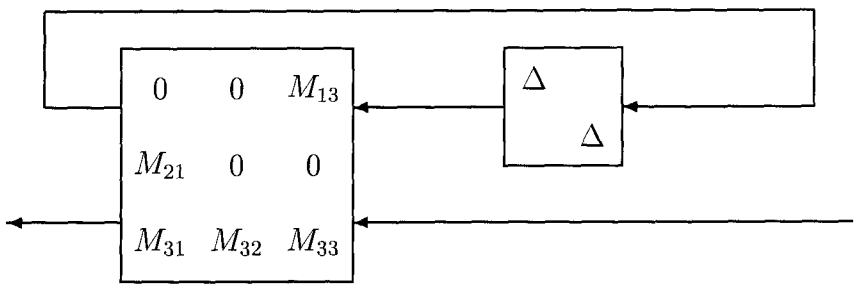


Figure 1.6: The quadratic programming μ problem.

Interval Polynomials

An interval polynomial is a set of polynomials whose coefficients are real and lie in some specified interval. Consider the interval polynomial

$$P(s,k) = s^n + [s^{n-1} + \ldots + s^0] \begin{bmatrix} a_1(k) \\ \vdots \\ a_n(k) \end{bmatrix}$$

and assume that $P(s,0) \neq 0$. Suppose further that the parameter variation is affine in k: $a = Fk + g$. We want to know if there is a root of the interval polynomial at s. Define

$$\begin{aligned} \Delta &= \mathrm{diag}(k_i) \\ \text{and } u &= \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \end{aligned}$$

so that $a = F\Delta u + g$. Then

$$\begin{aligned} P(s,k) &= s^n + [s^{n-1} + \ldots + s^0](F\Delta u + g) \\ &= P(s,0) + [s^{n-1} + \ldots + s^0]F\Delta u. \end{aligned}$$

Now define

$$\begin{aligned} v^* &= \frac{[s^{n-1} + \ldots + s^0]F}{P(s,0)} \\ \text{and } M &= -uv^* \end{aligned}$$

so that

$$\begin{aligned} 0 &= P(s,k) \\ \iff 0 &= 1 + v^*\Delta u = det(I - M\Delta). \end{aligned}$$

Since the roots of a polynomial are continuous in its coefficients, checking stability of the interval polynomial (no roots in $\mathbb{C}^+$) requires checking $P(s,0)$ and $P(j\omega,k)$ $\forall\omega \in \mathbb{R}$. Thus problems of robust stability of an interval polynomial (or a ratio of such polynomials for that matter) are examples of linear fractional uncertainty, and hence can be recast as mixed μ problems.

Mapping Disks in the Complex Plane

There is always a scalar LFT

$$w = \frac{az + b}{cz + d}$$

that maps any three given distinct points in the complex plane to any other three given distinct points in the complex plane. Each set of three points defines a generalized disc in the complex plane: if the points are colinear, the generalized disc is a halfplane, and the order of the points defines which of the two subsets of the plane is the interior. Thus any halfplane or disk in $\mathbb{C}$ can be mapped to the unit disc in $\mathbb{C}$ with a scalar LFT. Such mappings make sense for repeated complex scalar blocks as well. Consequently, the theorems

for $\mathbf{B}_{\mathbf{\Delta}}$ are quite general. The set $\mathbf{B}_{\mathbf{\Delta}}$ could just as well be any generalized disk, $\mathbb{D}$. An easy way of handling such sets is to map the unit disk to them: $\mathbb{D} = \mathbf{B}_{\mathbf{\Delta}} * M$. (It is routine to show that the scalar LFT can be put in the form of Figure 1.1 if d is not zero.)

Similarly, covering a real interval with a disk need not be done with a disk that is symmetric about the real axis. Also, the boundary of disks and halfplanes map to the boundary of the unit disk with an LFT, and rays, arcs, and line segments map to portions of the boundary of the unit disc with an LFT.

Chapter 2

Fundamental Properties of μ Computation

Since μ is defined to be the answer to a great many control problems, it is not surprising that μ is, in general, difficult to compute. For the same reason, the effective computation of μ is desirable. In later chapters we shall see that effective computation is possible. We shall also see that it is important to keep in mind the theoretical nature of the problem when developing effective computation. This chapter addresses the theoretical nature of the computation and its implications. Although the motivation for effective μ computation is quite general, the problem of analyzing robustness with respect to real parameter variation has been the motivation for many of the advences presented in later chapters.

The problem of analyzing robustness with respect to real parameter variations has received a great deal of attention in recent years. Although many different approaches to the problem have been attempted, each approach can be classified in one of two major research programs. One program is typified by the approach taken here and may be thought of as extending the complex μ theory to real perturbations. The other program, referred to here as the polynomial approach, has focused on extending Kharitonov's celebrated result on interval polynomials ([20]) to more general uncertainty structures. In this chapter the polynomial approach is placed in the μ/LFT framework in order to consider the implications of several recent results that are relevant to both research programs.

We first consider the implications of the fact that the purely real μ problem is discontinuous in the problem data. We argue that although discontinuities do not actually occur in problems of engineering interest, these results suggest that mixed μ computation may sometimes be poorly conditioned.

In contrast, the result that mixed μ is, in general, an NP hard problem has important and direct implications for practical application of any computational scheme. Indeed, this result suggests that entire classes of algorithms that attempt to compute mixed μ will be prohibitively expensive, even on

problems of moderate size.

Two strategies are available to deal with this apparent intractability of mixed μ computation. One possibility is to consider only a subset of problems where μ is easy to calculate explicitly. Unfortunately, such problems are relevant to few problems of engineering interest; so another strategy, such as the one proposed here, is needed.

A consequence of the computational complexity of the mixed μ problem is the need to evaluate computational schemes by evaluating their typical performance on a large class of typical problems. This chapter concludes with a discussion of classes of randomly generated problems used for evaluating μ computation.

2.1 Continuity

It is easy to see that $\mu(M)$ is continuous in M when all blocks of Δ are complex: $\det(\mu - M\Delta)$ is a polynomial in μ, and the roots of the polynomial, $\{\ \mu \mid \det(\mu - M\Delta) = 0\}$, are continuous in its coefficients, which are in turn continuous in M and Δ. Each complex root corresponds to a real root with Δ rotated appropriately, since $\Delta \in \mathbf{\Delta} \implies e^{j\theta}\Delta \in \mathbf{\Delta}$ whenever $\mathbf{\Delta}$ is a complex block structure.

In contrast, real μ and mixed μ are not continuous in their argument. For example, the real μ problem $\mu(x + iy)$, $x, y \in \mathbb{R}$, at $x = 1$ is 1 if $y = 0$ and is 0 otherwise. This discontinuity clearly adds computational difficulties to the problem, because any method involving some type of search over a family of M, such as a frequency response plot, must address the possibility of missing a point of discontinuity. More important, however, serious doubt is shed on the usefulness of real μ as a robustness measure in such cases: the system model is always a mathematical abstraction from the real world and is computed only to finite precision, so it is desirable that any type of robustness measure we use be continuous in the problem data.

Packard and Pandey address the problem of "regularizing" discontinuous μ problems in [29]. They show a variety of ways to construct a sequence of continuous problems that converges to the original discontinuous problem. The continuous problems may be constructed for example by adding a small amount of complex uncertainty to each real uncertainty. This regularization is well motivated for conventional engineering applications, where unmodeled dynamics always produce some phase uncertainty.

Packard and Pandey also show that mixed μ problems containing some complex uncertainty are typically continuous, even without the regularization procedure outlined above. This result is reassuring, since one is usually interested either in robust performance problems, which contain at least one

complex block, or in robust stability problems with some unmodeled dynamics, which are naturally represented by complex uncertainty. Thus, in problems of engineering interest, the potential discontinuity of μ should not arise. A further consequence, relevant to engineering problems, is that the numerical conditioning of mixed μ computation could cause difficulties and thus deserves more study.

2.2 NP Completeness and Computational Complexity

The decision problem Is $\mu \geq 1$? with real block structure is shown by Rohn and Poljak in [32] to be NP hard. Furthermore, Demmel ([12]) shows that even approximate solutions are also NP hard. Since a set of problems is NP hard whenever a subset is, the general mixed μ problem is NP hard as well.

A presentation of these results by Braatz et al. ([7]) is based on the reformulation, shown in Chapter 1, of the indefinite quadratic programming problem

$$\text{Is} \max_{b_l \leq x \leq b_u} \|x^* A x + p^* x + c\| \geq 1? \tag{2.1}$$

with $A \in \mathbb{R}^{n \times n}$, x, p, b_l, and $b_u \in \mathbb{R}^n$, and $c \in \mathbb{R}$ as a real or mixed μ problem. It is well known that the indefinite quadratic programming problem is NP complete. The idea is simple: the number of corners in the constraint grows exponentially with problem size, and none of the corners can be ruled out without some computation. This is in contrast to linear programming, where interior point methods allow for polynomial time computation despite an exponential growth in number of corners. Since any quadratic programming problem can be solved by solving a μ problem, it follows that the mixed μ problem is NP hard.

Given that the decision problem Is $\mu \geq 1$? is NP hard, we conclude that it is also NP complete since it as easily seen that the decision problem is also polynomial time verifiable: whenever the answer to the decision problem is Yes, there is a Δ that, if known, allows us to verify in polynomial time that $\mu \geq 1$. Clearly, calculating μ is at least as difficult. Although it is not a precise use of the terminology, we will say that μ computation is NP hard.

It remains a fundamental open question in the theory of computational complexity to determine the exact consequences of NP completeness in a problem, and we refer the reader to Garey and Johnson ([18]) for a treatment of the subject. It is generally accepted, however, that if a problem is NP complete that it cannot be computed in polynomial time in the worst case. Typically, worst-case computation cost grows exponentially with problem size.

For the reader not familiar with these concepts, we offer the following illustration. Figure 2.1 shows two different growth rates of computation time versus problem size n. For each of two growth rates there are two algorithms, one that can solve a problem of size $n = 10$ in 10 seconds, and another that can solve the same problem in 0.01 seconds. The first growth rate is n^3, a polynomial time growth rate typical of algorithms for eigenvalues, singular values, multiplication of two matrices, and so forth. The second growth rate is 2^n, an exponential time growth rate typical of algorithms that check all the edges or vertices of some polytope, for example.

Growth Rate	Problem Size (n)				
	10	20	30	40	50
n^3	0.01 seconds	0.08 seconds	0.27 seconds	0.64 seconds	1.25 seconds
	10 seconds	1.33 minutes	4.50 minutes	10.67 minutes	20.83 minutes
2^n	0.01 seconds	10.24 seconds	2.91 hours	124.3 days	348.7 years
	10 seconds	2.84 hours	121.4 days	340.5 years	3.49×10^5 years

Figure 2.1: A comparison of computation time growth with problem size for polynomial and exponential time growth rates.

It is readily seen that given an algorithm with a polynomial time growth rate we can apply the algorithm to larger and larger problems with a reasonable increase in the computational requirements. In contrast, an algorithm with an exponential time growth rate implies a dramatic increase in computational requirements with problem size, and for even moderate sizes the problem rapidly becomes intractable. Even when the exponential time algorithm is much faster on small problems, it still rapidly becomes impractical as the problem size increases—compare the middle two rows of Figure 2.1.

It is important to realize that looking for polynomial time exact computation for μ is the *same* as for many other problems. Such research is rightly in the field of computational complexity, not control. These problems are the same because any of them can be converted into any other in polynomial time.

With this in mind, we see that proofs of guaranteed convergence to μ are irrelevant. Any such proof is meaningless for any computation other than those that are trivially small and consequently easy. Large problems demand polynomial time algorithms, regardless of the speed on small problems. The fact that the mixed μ problem is NP hard means that we cannot expect to find such algorithms if we attempt to solve the general problem exactly for

all cases. Any practical effort to compute the answer to an NP hard problem must be measured by its performance on a large number of typical problems.

A good deal of research in robust control computation has ignored this issue entirely.

NP completeness is a property of a problem, not a property of an algorithm; it sets a limit on how fast an algorithm that computes exact answers can be. If mixed μ problems were shown to have polynomial time growth, it would be a dramatic result in the theory of computational complexity. Certainly, some algorithms can be much worse than others; for example, an algorithm may exhibit exponential growth for a problem that is computable in polynomial time. Such an algorithm would be intolerable for large problems, as shown in Figure 2.1.

Since the mixed μ problem is NP hard, we expect that given any algorithm to compute μ there will be problems for which the algorithm cannot find the answer in polynomial time. Thus, for all practical purposes, even moderately large examples of such problems are computationally intractable; practical computation requires giving up exact computation in the worst case.

An approach to practical computation is to consider special cases of the general problem that may be easier to solve. The difficulty with this approach is that we would like the resulting algorithm to be widely applicable to a large number of engineering problems, and the special cases that are easily solvable may be too restrictive. For this reason, subsequent chapters do not adopt this approach but rather concentrate on the general problem. Nevertheless, since special cases have been the focus of so much research, the next few sections consider those special cases for which computation of μ is relatively easy.

2.3 Problems with Special Structure; Restrictions on Δ

In light of the NP completeness results presented in the preceding section, it is natural to ask if there are special cases of the mixed μ problem that are relatively easy to compute. There are, and in practically all such cases it can be verified a priori that μ is equal to its upper bound which can be computed as a convex optimization problem. (The upper bound is the subject of Chapter 4.) Unfortunately, these special cases are relevant to few problems of engineering interest.

Although somewhat artificial, it is useful to separately consider restrictions on the nominal system and on the uncertainty structure (respectively M and Δ in Figure 2.2), because restrictions on each one result in easily computable special cases. Computation is easier when M is highly structured, while less structure on Δ makes computation easier. However, these easier problems are

not particularly useful: problems motivated by real engineering applications typically have unstructured nominal systems combined with highly structured uncertainty, and they consequently are difficult to compute.

For simplicity, consider the standard problem of robust stability for the system in Figure 2.2 where Δ is norm-bounded by 1. The least structured Δ is a single block that is allowed to be an arbitrary nonlinear, time-varying operator. In that case the small gain condition is necessary and sufficient, and the test is simply $\|M\|_\infty < 1$. This test remains necessary and sufficient when Δ is restricted to be causal, and also when it is further restricted to be either linear time-varying (LTV) or linear time-invariant (LTI).

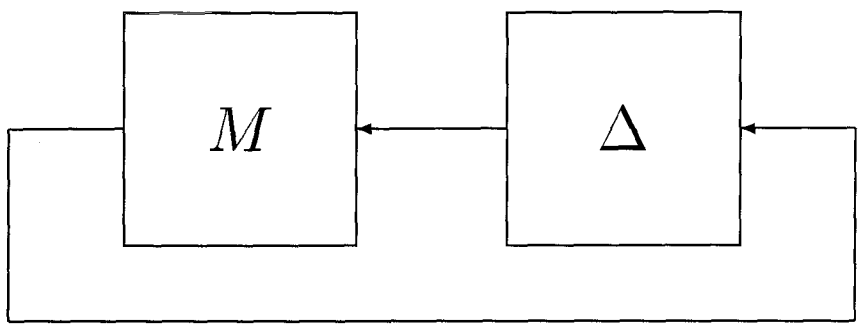


Figure 2.2: The standard robust stability problem.

Additional structure on Δ leads to μ tests of increased complexity, but some special cases exist when μ is equal to its upper bound. If Δ is block diagonal with any number of LTV perturbations, then recent results, obtained independently by Shamma ([33]) and Megretsky ([23]), show that the exact test for this case is equivalent to the upper bound to a complex constant matrix μ problem (the upper bound may be found in Theorem 4.3). Frequently however, the smallest destabilizing perturbation for such a problem has a memory much longer than might seem reasonable in a practically motivated problem, so we might wonder whether a set described by such a Δ is too large.

Also, in the constant matrix case, if Δ consists of three or fewer full complex blocks, or if Δ consists of one full and one repeated scalar block, then μ is equal to its upper bound. In general, however, μ is not equal to its upper bound for more elaborate uncertainty structures unless additional structure is imposed on M. The case of structural restrictions on M is considered in the next section.

2.4 Restrictions on M and Kharitonov-Type Results

A recently popular research program has focused on extending Kharitonov's celebrated result on interval polynomials (see [20]) to more general uncertainty structures. An interval polynomial is a set of polynomials whose coefficients are real and lie in some specified interval. Kharitonov showed that one need check only four polynomials to determine whether every polynomial in the interval polynomial is stable. Several additional results have since been proved for similar problems, such as polynomials whose coefficients are affine in some real parameters (see Bartlett et al., [5], for example), and the solutions typically involve checking the edges or vertices of some polytope in the parameter space. The family of interval polynomial stability problems where the coefficient perturbation dependence is affine is readily converted into a family of real μ problems on matrices that are rank one, as in Section 1.3.

The rank one mixed μ problem is studied in detail by Chen et al. in [8], where an easily computed expression for the solution to this problem is developed. They then solve several problems from the literature, that are special cases of rank one μ problems, hence relatively easy. The need to check a combinatoric number of edges, often a feature of Kharitonov-type results, is unnecessary. Checking edges is an example of an expensive algorithm for an easy problem.

The next section presents a simple and intuitive algorithm for solving the rank one μ problem. This solution is also useful as a step in the computation of a lower bound to the general problem, where the matrix need not be rank one.

2.5 The Rank One μ Problem

The rank one mixed μ problem is simply the special case of computing $\mu(M)$ where M is a dyad. The simplicity of the rank one mixed μ problem is apparent in the following graphical interpretation in the complex plane.

Let $M = uv^*$, $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$, and $\alpha \in \mathbb{R}^+$. Then

$$\begin{aligned} 0 &= \det(I - \frac{\Delta}{\alpha}M) = \det(1 - v^*\frac{\Delta}{\alpha}u) \\ &\iff v^*\Delta u = \alpha \\ &\iff \sum_{r\in\mathbf{r}} \delta_r v_r^* u_r + \sum_{c\in\mathbf{c}} \delta_c v_c^* u_c + \sum_{C\in\mathbf{C}} v_C^* \Delta_C u_C = \alpha. \end{aligned}$$

Recalling the definition of μ, Equation (1.3), we see that

$$\mu = \max_{\Delta\in\mathbf{B}_{\mathbf{\Delta}}} \left\{ \alpha \;\middle|\; \det(I - \frac{\Delta}{\alpha}M) = 0 \right\}$$

$$= \max_{\Delta \in \mathbf{B}_\Delta} \left(\sum_{r \in \mathbf{r}} \delta_r v_r^* u_r + \sum_{c \in \mathbf{c}} \delta_c v_c^* u_c + \sum_{C \in \mathbf{C}} v_C^* \Delta_C u_C \right).$$

This equation is the basis for the graphical interpretation of the rank one mixed μ problem. The problem is solved by choosing δ_r, δ_c, and Δ_C so that the complex vectors $\delta_r v_r^* u_r$, $\delta_c v_c^* u_c$, and $v_C^* \Delta_C u_C$ add up to the largest possible positive real number. The contributions of the complex blocks cannot be optimal unless they are colinear; they must all have the same phase. Thus

$$\sum_{c \in \mathbf{c}} \delta_c v_c^* u_c + \sum_{C \in \mathbf{C}} v_C^* \Delta_C u_C = e^{j\Psi} L_C \tag{2.2}$$

and the rank one mixed μ problem reduces to choosing real numbers $\delta_r \in [-1, 1]$ and $\Psi \in [-\pi, \pi]$ to maximize α where

$$\sum_{r \in \mathbf{r}} \delta_r v_r^* u_r + e^{j\Psi} L_c = \alpha. \tag{2.3}$$

By thinking of Equation (2.3) as a vector sum in the complex plane, it is apparent that the solution to this problem easily can be obtained geometrically.

The following algorithm is guaranteed to compute mixed μ exactly for the rank one matrix $M = uv^*$. It also allows us to find an optimal perturbation $Q \in \partial \mathbf{B}_\Delta$.

- Start with $\delta_r = \text{sgn}(\text{Re}(v_r^* u_r))$.
- Compute $S = \text{sgn}(\text{Im}(\sum_{r \in \mathbf{r}} \delta_r v_r^* u_r))$.
- Rank all the components $\delta_r v_r^* u_r$ by argument, $\angle(S \delta_r v_r^* u_r)$, in descending order.
- ▶ For the highest rank component not yet considered, assign the optimal value of this δ_r unrestricted in sign or magnitude to $\widehat{\delta_r}$ (the optimal value depends on L_C and $\angle(S \delta_r v_r^* u_r)$).
- If $\text{sgn}(\widehat{\delta_r}) = -\text{sgn}(\delta_r)$ and $\widehat{\delta_r} \notin (-1, 1)$, and this is not the last possible component, then go back to the previous step ▶, else proceed. In either case, reassign $\delta_r = \max(-1, \min(\widehat{\delta_r}, 1))$.
- We now have an optimal δ_r and can easily compute the optimal value of Ψ and then the perturbation Q.

The algorithm requires at most a search over real parameters, the cost of which grows linearly with the number of real blocks. Thus we have a closed form solution for both $\mu(M)$ and the associated $Q \in \partial \mathbf{B}_\Delta$, with trivial computational requirements.

A version of this algorithm tailored to Matlab[1] implementation is efficient and also has proved useful in the computation of lower bound to the more general μ problem, as explained in Chapter 3.

2.6 Convex Problems

The rank one mixed μ problem also can be addressed within the upper bound framework described in Chapter 4.

Theorem 2.1 ([40]) *Suppose we have a rank one matrix $M \in \mathbb{C}^{n\times n}$, then $\mu(M)$ equals its upper bound from Theorem 4.4.*

Thus rank one problems as well as the problems of Section 2.3 are equivalent to LMI optimization problems. Such problems are convex and are comparatively easy to compute.

Although solving the upper bound LMI optimization problem is not the best way to solve a rank one μ problem, all easy μ problems are ones where μ is equal to its upper bound. Indeed, if μ is not equal to its upper bound in Theorem 4.4, it is generally difficult even to verify the value of μ. Typically, we can verify the value of μ only by showing that the upper and lower bounds are equal.

This theorem reinforces the results of Chen et al. ([8]) and offers some insight into why the problem becomes so much more difficult when we move away from the affine parameter variation case to the multilinear or polynomial cases ([34]). These cases correspond to μ problems where M is not necessarily rank one and hence may no longer be equal to the upper bound, and so may no longer be equivalent to a convex optimization problem (there are rank two matrices for which μ does not equal its upper bound). There are no practical general algorithms based on edge-type theorems, because the results are relevant only to a small class of problems. Furthermore, even in the affine parameter case, there is a combinatoric number of edges to check.

2.7 Generating Test Matrices

Although the material in this section relies on results from Chapters 3 and 4, it is presented here because the need for test matrices is a direct consequence of the computational complexity of the mixed μ problem. The results in this section are due to Young ([39]).

We have seen that the mixed μ problem is NP hard, which implies that the worst-case performance of any algorithm is poor, in terms of either the

[1]Matlab is a convenient language for writing such algorithms and is available from The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760

accuracy of the bounds or the growth rate in computation. In fact, we can construct examples for which the bounds in Theorems 3.2 and 4.4 are arbitrarily far apart. For our purposes the real issue is whether we can develop a practical algorithm whose typical performance is acceptable. In order to examine the typical performance in Section 4.4, we run the algorithm repeatedly on a large number of test matrices, randomly generated from within certain classes, then collect statistical data. In this section we describe the three specific types of random matrices used.

The most straightforward way to generate random complex matrices in Matlab is with the μ-Tools command `crand`. This command generates matrices whose elements are random variables, and by setting `rand('normal')` in Matlab we can choose these elements to be normally distributed with zero mean. We refer to this type of random matrix as a `crand` matrix.

Unfortunately `crand` matrices are not at all representative of matrices arising from control problems. A more natural class is formed by randomly generating State Space A, B, C, and D matrices using the μ-Tools command `sysrand` and evaluating the transfer function at some frequency, usually placed roughly in the middle of the modes. We refer to this type of random matrix as a `sysrand` matrix.

For the purposes of testing algorithms it is desirable to be able to generate problems for which we know the answers a priori. The following algorithm provides us with the means to generate such problems:

- Randomly generate matrices $\widehat{D} \in \widehat{\mathbf{D}}$, $\widehat{G} \in \widehat{\mathbf{G}}$, and $Q \in \mathbf{Q}$. In addition randomly generate a unitary matrix $Y \in \mathbb{C}^{n\times n}$ and a real nonnegative diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1 \ldots \sigma_n)$ with

$$\begin{aligned} \sigma_i &= 1 \quad \text{for} \quad i \in \{1 \ldots r\} \text{ and} \\ \sigma_i &< 1 \quad \text{for} \quad i \in \{r+1 \ldots n\}, \end{aligned} \tag{2.4}$$

  where r is some integer satisfying $1 \leq r < n$. Finally generate a random unit norm vector $\eta \in \mathbb{C}^n$ with the restriction that

$$\eta_i = 0 \qquad \text{for} \qquad i \in \{r+1 \ldots n\}. \tag{2.5}$$

- Compute $X \in \mathbb{C}^{n\times n}$ as any unitary matrix that satisfies the equation

$$X\eta = (Q^{-1} - j\widehat{G})(I + \widehat{G}^2)^{-\frac{1}{2}} Y\eta. \tag{2.6}$$

  The matrix $(Q^{-1} - j\widehat{G})(I + \widehat{G}^2)^{-\frac{1}{2}}Y$ is unitary, so that we can always find such an X.

- Compute $M \in \mathbb{C}^{n\times n}$ as

$$M = \widehat{D}^{-1}\left((I + \widehat{G}^2)^{\frac{1}{4}} X\Sigma Y^* (I + \widehat{G}^2)^{\frac{1}{4}} + j\widehat{G}\right)\widehat{D}. \tag{2.7}$$

Before proving that this algorithm does indeed generate problems for which we know the answers a priori, we need a preliminary result.

Theorem 2.2 *Suppose we have $M \in \mathbb{C}^{n\times n}$ and a compatible block structure. If the infimization in the μ upper bound, Equation (4.4), is achieved and is equal to $\mu(M)$, then*

$$\max_{Q\in\mathbf{Q}} \rho_r(QM) = \mu(M). \tag{2.8}$$

This theorem says that if μ is equal to its upper bound, Equation (4.4), and the infimum in the upper bound is achieved, then the worst-case perturbation may be taken to be on a vertex.

Proof: From Equation (3.2) we immediately conclude that

$$\max_{Q\in\mathbf{Q}} \rho_r(QM) \leq \mu(M).$$

Hence the result is trivial for $\mu(M) = 0$, so consider the case where $\mu(M) > 0$. Then by a simple scaling argument we may without loss of generality assume $\mu(M) = 1$. Suppose we have the perturbation $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ achieving the maximum in Equation (3.2), so there is an $x \in \mathbb{C}^n$ with $x \neq 0$ such that

$$QMx = x.$$

This implies that the block components of the vectors x and $z = Mx$ satisfy

$$\begin{aligned} q_r z_r &= x_r \quad \text{for} \quad r \in \mathbf{r} \\ q_c z_c &= x_c \quad \text{for} \quad c \in \mathbf{c} \\ Q_C z_C &= x_C \quad \text{for} \quad C \in \mathbf{C}. \end{aligned} \tag{2.9}$$

Now by assumption we have $D \in \mathbf{D}$ and $G \in \mathbf{G}$ such that

$$(M^*DM + j(GM - M^*G) - D) \leq 0$$

so that in particular

$$x^*(M^*DM + j(GM - M^*G) - D)x \leq 0.$$

Expanding this expression, and substituting Equation (2.9), we see that

$$\sum_{r\in\mathbf{r}} \left\| D_r^{\frac{1}{2}} z_r \right\|^2 \leq \sum_{r\in\mathbf{r}} q_r^2 \left\| D_r^{\frac{1}{2}} z_r \right\|^2 \tag{2.10}$$

(note that all the complex blocks q_c and Q_C are unitary). Since $\|q_r\| \leq 1$ for all $r \in \mathbf{r}$

$$\|q_r\| < 1 \implies \left\| D_r^{\frac{1}{2}} z_r \right\| = 0. \tag{2.11}$$

Since $D_r > 0$, Equation (2.11) implies $z_r = 0$ and hence, by Equation (2.9), $x_r = 0$. Thus we may take $q_r = 1$ for all such blocks and satisfy $QMx = x$ with $Q \in \mathbf{Q}$. ■

Theorem 2.3 *Suppose we have a matrix $M \in \mathbb{C}^{n\times n}$ and a compatible block structure. Denote the upper bound from Theorem 4.6 by $\widehat{\mu}$. Then the following two conditions are equivalent.*

a: *There exist matrices $\widehat{D} \in \widehat{\mathbf{D}}$ and $\widehat{G} \in \widehat{\mathbf{G}}$ achieving the infimum in Theorem 4.6, and $\widehat{\mu}(M) = \mu(M) = 1$.*

b: *M can be generated by the above algorithm.*

Proof: (**b** $\implies$ **a**) Equation (2.7) implies

$$(I + \widehat{G}^2)^{-\frac{1}{4}}(\widehat{D}M\widehat{D}^{-1} - j\widehat{G})(I + \widehat{G}^2)^{-\frac{1}{4}} = X\Sigma Y^*$$

and hence

$$\overline{\sigma}\left((I + \widehat{G}^2)^{-\frac{1}{4}}(\widehat{D}M\widehat{D}^{-1} - j\widehat{G})(I + \widehat{G}^2)^{-\frac{1}{4}}\right) = 1. \tag{2.12}$$

Then Theorem 4.6 implies that $\widehat{\mu}(M) \leq 1$, with $\widehat{D} \in \widehat{\mathbf{D}}$ and $\widehat{G} \in \widehat{\mathbf{G}}$ achieving this bound. By construction $\Sigma Y^* Y\eta = \eta$, so Equation (2.6) implies

$$X\Sigma Y^* Y\eta = (Q^{-1} - j\widehat{G})(I + \widehat{G}^2)^{-\frac{1}{2}} Y\eta.$$

Rearranging this equation and noting that any $\widehat{D} \in \widehat{\mathbf{D}}$ and $Q \in \mathbf{Q}$ commute, we get

$$Q\widehat{D}^{-1}\left((I + \widehat{G}^2)^{\frac{1}{4}} X\Sigma Y^* (I + \widehat{G}^2)^{\frac{1}{4}} + j\widehat{G}\right)\widehat{D}\widehat{D}^{-1}(I + \widehat{G}^2)^{-\frac{1}{4}} Y\eta = \widehat{D}^{-1}(I + \widehat{G}^2)^{-\frac{1}{4}} Y\eta.$$

Substituting M from Equation (2.7) and defining

$$x = \widehat{D}^{-1}(I + \widehat{G}^2)^{-\frac{1}{4}} Y\eta,$$

where x is not zero, we have

$$QMx = x,$$

hence $\mu(M) \geq 1$. Together with Equation (2.12) this implies **a**.

($\mathbf{a} \Longrightarrow \mathbf{b}$) Theorems 3.2 and 4.6, together with **a** imply that there are $\widehat{D} \in \widehat{\mathbf{D}}$, $\widehat{G} \in \widehat{\mathbf{G}}$, and $Q \in \partial \mathbf{B}_{\mathbf{\Delta}}$ such that

$$\overline{\sigma}\left((I+\widehat{G}^2)^{-\frac{1}{4}}(\widehat{D}M\widehat{D}^{-1} - j\widehat{G})(I+\widehat{G}^2)^{-\frac{1}{4}}\right) \leq 1 \tag{2.13}$$

$$QMx = x \tag{2.14}$$

with $x \neq 0$. By Theorem 2.2 we may assume $Q \in \mathbf{Q}$ without loss of generality. Furthermore, by continuity of singular values and by Theorem 4.6, we have equality in Equation (2.13). Let $X\Sigma Y^*$ be a singular value decomposition of Equation (2.13):

$$X\Sigma Y^* = (I+\widehat{G}^2)^{-\frac{1}{4}}(\widehat{D}M\widehat{D}^{-1} - j\widehat{G})(I+\widehat{G}^2)^{-\frac{1}{4}}. \tag{2.15}$$

The algorithm could choose these $\widehat{D}$, $\widehat{G}$, Q, Y, and Σ. From Equation (2.15) we see that M satisfies Equation (2.7), so it remains to show that our algorithm could choose this X.

Substituting for M in Equation (2.14) we obtain

$$X\Sigma Y^*(I+\widehat{G}^2)^{\frac{1}{4}}\widehat{D}x = (Q^{-1} - j\widehat{G})(I+\widehat{G}^2)^{-\frac{1}{2}}(I+\widehat{G}^2)^{\frac{1}{4}}\widehat{D}x.$$

Define $y = (I+\widehat{G}^2)^{\frac{1}{4}}\widehat{D}x \neq 0$ and $\widehat{y} = \frac{y}{\|y\|}$ so that

$$X\Sigma Y^*\widehat{y} = (Q^{-1} - j\widehat{G})(I+\widehat{G}^2)^{-\frac{1}{2}}\widehat{y}. \tag{2.16}$$

Since $(Q^{-1} - j\widehat{G})(I+\widehat{G}^2)^{-\frac{1}{2}}$ and X are unitary, we see that $\|\Sigma Y^*\widehat{y}\| = 1$. Defining $\eta = Y^*\widehat{y}$ and noting that $\|\eta\| = 1$, we see that the structure of Σ implies that η satisfies Equation (2.5) and thus could have been chosen by the algorithm. Then Equation (2.16) becomes

$$X\eta = (Q^{-1} - j\widehat{G})(I+\widehat{G}^2)^{-\frac{1}{2}}Y\eta$$

so our algorithm could choose this X and hence generate M. ∎

Remarks: The above algorithm was first developed for the purely complex case by Fan et al. ([15, 28]). Note that we can select the number of singular values coalesced at the minimum of the upper bound function in Theorem 4.6, and a simple extension to the algorithm allows us also to select the number of eigenvalues coalesced at the maximum of the lower bound function in Theorem 3.2.

This algorithm can randomly generate any problem with the upper bound achieved and equal to $\mu = 1$, together with optimal scaling matrices achieving the upper and lower bounds. For these problems there is no gap between the bounds from Theorems 3.2 and 4.4, although the lower bound is still a nonconvex maximization problem. We refer to a random matrix generated by the above algorithm as a `nogap` matrix.

Chapter 3

The Lower Bound

In Chapter 2 we saw that the computation of μ is inherently difficult and effective computation requires approximation. Although upper and lower bounds to μ each have their own system interpretation in the original problem, neither is terribly interesting in isolation: a bound is typically of little use without some measure of its quality. Such a measure is provided by upper and lower bounds that are close together.

While Chapter 4 addresses the computation of an upper bound to μ, the present chapter addresses the computation of a lower bound to mixed μ. A power iteration to compute a lower bound for the complex μ problem (see Packard et al., [28]) was generalized to the mixed case by Young as follows. First, mixed μ is reformulated as the real eigenvalue maximization problem $\max_Q \rho_r(QM)$. Chapter 2 explains why the goal is to find a local maximum rather than the global maximum. Second, a local maximum of $\rho_r(QM)$ implies an alignment between the right and left eigenvectors of QM. This alignment condition is in turn associated with the solution to a certain set of matrix-vector equations. Third, a power iteration is developed whos equilibrium points satisfy the alignment conditions and thus provide a local maximum. The theoretical development of the power iteration, together with some aspects of its implementation, is fully described by Young and Doyle in [41].

Even though each iteration of the scheme is inexpensive, involving only matrix-vector multiplications and vector inner products, it often converges quickly to a large local maximum. Unfortunately, the lower bound power iteration is not guaranteed to converge to even a local maximum, and does not converge on a significant number of problems. Although we can still obtain a lower bound from the scheme in such cases by holding the real blocks of the perturbation constant, the resulting lower bound is generally not a local maximum and may be poor. The first effort to enhance the performance of the standard power algorithm (SPA) is presented by Tierno and Young in [37], with encouraging results.

In this chapter we present the SPA and several new approaches to computing an improved μ lower bound. We begin with the reformulation of μ as an

eigenvalue maximization problem and the characterization of local maximums in Section 3.1. An alternative, more general, derivation is in Chapter 6, but for now we follow a more traditional development. We then present the SPA and an analysis of its convergence properties. Subsequently, we describe a variety of modifications to the SPA which result in algorithms with better convergence properties. These algorithms are then combined to yield a substantially improved power algorithm, and the resulting performance is presented in Section 3.7.

3.1 A Maximization Problem

First consider the computation of a lower bound to the mixed μ problem. We cannot simply replace the real perturbations with complex perturbations and then use the complex μ lower bound since that would include perturbations from outside the permissible set $\mathbf{\Delta}$, and so would not yield a valid lower bound.

If Δ_a and Δ_b are both in both in $\mathbf{B_\Delta}$ then $\Delta_a\Delta_b$ is too, so we may replace the lower bound $\rho_r(M) \leq \mu(M)$ in Equation (1.5) with

$$\max_{\Delta \in \mathbf{B_\Delta}} \rho_r(\Delta M) \leq \mu(M).$$

From the definition of μ it is easy to see that the inequality may be replaced by equality.

Lemma 3.1 *For any matrix $M \in \mathbb{C}^{n\times n}$ and any compatible block structure*

$$\mu(M) = \max_{\Delta \in \mathbf{B_\Delta}} \rho_r(\Delta M). \tag{3.1}$$

Furthermore, since $\partial\mathbf{B_\Delta} \subset \mathbf{B_\Delta}$ it is easy to see that

$$\max_{Q \in \partial\mathbf{B_\Delta}} \rho_r(QM) \leq \mu(M).$$

Again, the inequality may be replaced by equality.

Theorem 3.2 ([41]) *For any matrix $M \in \mathbb{C}^{n\times n}$ and any compatible block structure*

$$\max_{Q \in \partial\mathbf{B_\Delta}} \rho_r(QM) = \mu(M). \tag{3.2}$$

Note that $\rho_r(QM) = \rho_r(MQ)$.

Since this maximization problem is not convex in Q and there are examples with strictly local maxima, we are in general only able to find local maxima. We would like this lower bound to be close to μ, therefore our goal is to find an efficient way to compute a (large) local maximum of the function $\rho_r(QM)$ over $Q \in \partial\mathbf{B_\Delta}$.

When Algorithms Fail to Converge

The algorithms described in Sections 3.2 through 3.7 are iterative and try to converge to a local maximum of $\rho_r(QM)$. When these algorithms fail to converge, as they occasionally do, we can still find a lower bound. The idea is simple: if we cannot find a local maximum of $\rho_r(QM)$, then a lower bound is found by evaluating $\rho_r(QM)$ at a Q that does not correspond to a local maximum. Some care is required because the Q must yield a real eigenvalue.

With a candidate mixed perturbation from the iteration scheme (i.e. a scaling matrix $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$) and a guess for the lower bound one can construct a valid perturbation. Suppose we have a matrix M partitioned as

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{3.3}$$

and block structures $\mathbf{\Delta}_1$ and $\mathbf{\Delta}_2$ compatible with M_{11} and M_{22} respectively. Then the block structure $\mathbf{\Delta}$ defined as

$$\mathbf{\Delta} = \{ \operatorname{diag}(\Delta_1, \Delta_2) \mid \Delta_1 \in \mathbf{\Delta}_1, \Delta_2 \in \mathbf{\Delta}_2 \} \tag{3.4}$$

is compatible with M. Further assume we have arranged the problem so that $\mathbf{\Delta}_1$ consists of purely real uncertainties, and $\mathbf{\Delta}_2$ consists of purely complex uncertainties. Then we have the following lemma.

Lemma 3.3 *Suppose we have a matrix $M \in \mathbb{C}^{n\times n}$ as above, a perturbation $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$ as above, and a real scalar $\alpha > 0$. If $\det(I - \frac{M_{11}\Delta_1}{\alpha}) = 0$ define $\beta = \alpha$ else define β as*

$$\beta = \rho\left(\Delta_2\left(\frac{\Delta_1}{\alpha} \star M\right)\right) \tag{3.5}$$

then $\min(\alpha, \beta) \leq \mu(M)$.

Proof: If $\det(I - \frac{M_{11}\Delta_1}{\alpha}) = 0$ then $\Delta = \operatorname{diag}(\frac{\Delta_1}{\alpha}, 0)$ sets $\det(I - \Delta M) = 0$ so that $\min(\alpha, \beta) = \alpha \leq \mu(M)$. Otherwise, either β in Equation (3.5) $= 0$ and $\min(\alpha, \beta) \leq \mu(M)$, or $\Delta = \operatorname{diag}(\frac{\Delta_1}{\alpha}, \frac{\Delta_2}{\beta}e^{-j\theta})$ sets $\det(I - \Delta M) = 0$ for some $\theta \in [0,\ 2\pi]$ so that $\min(\alpha, \beta) \leq \mu(M)$. ■

This lemma gives us the means to compute a lower bound to μ given the candidate guesses for the perturbation and the lower bound Q and α provided we are not in the pure real case. (In the pure real case our bound is α if $\det(I - \frac{M_{11}\Delta_1}{\alpha}) = 0$ and zero otherwise.) This scheme is included in all lower bound algorithms so they always return a valid lower bound, regardless of convergence.

The remainder of the chapter addresses the problem of finding local maximums, rather than the problem of what to do when we fail.

Characterization of Local Maximums

Efforts to compute a lower bound to μ have focused on finding a local maximum of Equation (3.2) rather than the NP hard problem of finding the global maximum. This section presents conditions that must be satisfied at every local maximum. The characterization of a maximum point of $\rho_r(QM)$ at $Q = I$ is in terms of an alignment of the right and left eigenvectors of M.

For the rest of this chapter we make a non-degeneracy assumption $w_i^* x_i \neq 0, i \in \mathbf{r} \bigcup \mathbf{c} \bigcup \mathbf{C}$. For any matrix $Q \in \partial \mathbf{B}_{\mathbf{\Delta}}$ define the index set

$$\mathcal{J} \triangleq \{ i \in \mathbf{r} \mid \|\Delta_i\| < 1 \} \tag{3.6}$$

and define the allowable perturbation set

$$\mathbf{B}_{\mathbf{\Delta}_\epsilon} \triangleq \{ \Delta \mid \Delta_i \in \mathbf{B}_{\mathbf{\Delta}_i} \text{ for } i \notin \mathcal{J}, \|\Delta_i\| < 1 + \epsilon \text{ for } i \in \mathcal{J}, \Delta \in \mathbf{\Delta} \}. \tag{3.7}$$

If $Q \in \partial \mathbf{B}_{\mathbf{\Delta}}$ we see that for sufficiently small $\epsilon > 0$, $\Delta \in \mathbf{B}_{\mathbf{\Delta}_\epsilon}$ implies $Q\Delta \in \mathbf{B}_{\mathbf{\Delta}}$ and $\Delta Q \in \mathbf{B}_{\mathbf{\Delta}}$.

Theorem 3.4 ([41]) *Suppose the matrix $M \in \mathbb{C}^{n \times n}$ has a distinct real eigenvalue $\lambda_0 > 0$ with right and left eigenvectors, x and w respectively, satisfying the non-degeneracy assumption. Further suppose that $\rho_r(M) = \lambda_0$. If, for some $\epsilon > 0$, the function $\rho_r(QM)$ attains a local maximum over the set $Q \in \mathbf{B}_{\mathbf{\Delta}_\epsilon}$ at $Q = I$, then there exists a matrix $D \in \mathbf{D}$ with $\theta_i = \pm\frac{\pi}{2}$ for every $i \in \mathcal{J}$, and a real scalar $\psi \in (-\frac{\pi}{2}, \frac{\pi}{2})$, such that $w = e^{j\psi} Dx$.*

Remarks: The condition $w = e^{j\psi} Dx$ is referred to as alignment of the eigenvectors w and x.

A partial converse to Theorem 3.4 was shown in [41]: if $w = e^{j\psi} Dx$ under the above assumptions, then no directional derivative of the eigenvalue achieving $\rho_r(QM)$ over the set $Q \in \mathbf{B}_{\mathbf{\Delta}_\epsilon}$ is real and positive at $Q = I$.

3.2 A Lower Bound Power Algorithm

In the first part of this section we review the SPA as developed by Young and Doyle in [41], following the tradition established by Packard et al. in [28] for the complex only case. In the second part we show the results of a comparison between the SPA and standard optimization code on the same problems. These results support the assertion that the SPA is a good starting point for the development of better algorithms for μ lower bound computation.

The SPA

Satisfying the alignment conditions for $\mu(M)$ can be reduced to finding matrices $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ and $D \in \mathbf{D}$ with $\theta_i = \pm\frac{\pi}{2}$ for $i \in \mathcal{J}(Q)$ and non-zero vectors v, w, x, and z such that the following set of equations holds.

$$\begin{aligned} \beta z &= Mx & \beta w &= M^* v \\ x &= Qz & x &= D^{-1} w \\ v &= Q^* Q D z & v &= Q^* w. \end{aligned} \tag{3.8}$$

Finding such solutions may be attempted via the power iteration below. We do not go into any of the details of the theoretical development here, but merely present the final result.

We explicitly write the formulae only for the simple block structure with one repeated real scalar block, one repeated complex scalar block, and one full complex block. This is for notational simplicity only. The formulae for an arbitrary block structure are obtained simply by repeating the formulae for each type of block appropriately. Except for the power steps Equations (3.11) and (3.13), the blocks are updated independently.

With this block structure,

$$\partial\mathbf{B}_{\mathbf{\Delta}} = \{\ \mathrm{diag}(q_r I, q_c, Q_C)\ \mid q_r \in [-1,1], q_c^* q_c = 1, Q_C^* Q_C = I\ \}. \tag{3.9}$$

Partition the four vectors v, w, x, and $z \in \mathbb{C}^n$ compatibly as

$$v = \begin{bmatrix} v_r \\ v_c \\ v_C \end{bmatrix} \quad w = \begin{bmatrix} w_r \\ w_c \\ w_C \end{bmatrix} \quad x = \begin{bmatrix} x_r \\ x_c \\ x_C \end{bmatrix} \quad z = \begin{bmatrix} z_r \\ z_c \\ z_C \end{bmatrix} \tag{3.10}$$

The SPA iteration is as follows.

$$\widetilde{\beta}_{k+1} z_{k+1} = M x_k \ :\ \widetilde{\beta}_{k+1} \in \mathbb{R}^+,\ \|z_{k+1}\| = 1 \tag{3.11}$$

$$\begin{aligned} v_{r_{k+1}} &= \widetilde{q}_{k+1} w_{r_k} \\ v_{c_{k+1}} &= \frac{w_{c_k}^* z_{c_{k+1}}}{\left\| w_{c_k}^* z_{c_{k+1}} \right\|} w_{c_k} \\ v_{C_{k+1}} &= \frac{\|w_{C_k}\|}{\left\| z_{C_{k+1}} \right\|} z_{C_{k+1}} \end{aligned} \tag{3.12}$$

$$\widehat{\beta}_{k+1} w_{k+1} = M^* z_{k+1} \ :\ \widehat{\beta}_{k+1} \in \mathbb{R}^+,\ \|w_{k+1}\| = 1 \tag{3.13}$$

$$\begin{aligned} x_{r_{k+1}} &= \widehat{q}_{k+1} z_{r_{k+1}} \\ x_{c_{k+1}} &= \frac{z_{c_{k+1}}^* w_{c_{k+1}}}{\left\| z_{c_{k+1}}^* w_{c_{k+1}} \right\|} z_{c_{k+1}} \end{aligned} \tag{3.14}$$

$$x_{C_{k+1}} = \frac{\|z_{C_{k+1}}\|}{\|w_{C_{k+1}}\|} w_{C_{k+1}}$$

where $\widetilde{q}_{k+1}$ and $\widehat{q}_{k+1}$ evolve as

$$\begin{aligned} \widetilde{\alpha}_{k+1} &= \operatorname{sgn}(\widehat{q}_k)\frac{\|x_{r_k}\|}{\|z_{r_{k+1}}\|} + \operatorname{Re}(z^*_{r_{k+1}} w_{r_k}) \\ \widetilde{q}_{k+1} &= \min(\max(-1, \widetilde{\alpha}_{k+1}), 1) \\ \widehat{\alpha}_{k+1} &= \operatorname{sgn}(\widetilde{q}_{k+1})\frac{\|x_{r_k}\|}{\|z_{r_{k+1}}\|} + \operatorname{Re}(z^*_{r_{k+1}} w_{r_{k+1}}) \\ \widehat{q}_{k+1} &= \min(\max(-1, \widehat{\alpha}_{k+1}), 1). \end{aligned} \tag{3.15}$$

An important feature of these relationships is that they do not involve the matrices Q and D; Q and D are defined implicitly by the vectors at equilibrium.

If the above iteration converges to an equilibrium point then we have a matrix $Q \in \partial \mathbf{B}_{\mathbf{\Delta}}$ such that $QMx = \widetilde{\beta}x$ and $w^*QM = \widehat{\beta}w^*$, so that $\max\{\widetilde{\beta}, \widehat{\beta}\}$ gives us a lower bound to $\mu(M)$. Furthermore if $\widetilde{\beta} = \widehat{\beta}$ then this bound corresponds to a local maximum of Equation (3.2). In a significant number of cases the iteration does not converge, and the application of Lemma 3.3 yields a poor lower bound to μ. These are precisely the cases that need improvement, motivating the work in Sections 3.3 through 3.6.

The SPA vs. Standard Optimization

In order to examine the average computational requirements both for the SPA implemented in Matlab (with no .mex files) and for directly solving Equation (3.2) via the standard optimization techniques of NPSOL[1], also implemented in Matlab (with .mex files), we ran both algorithms 100 times on random complex matrices with independent normally distributed elements, and collected statistical data. This was repeated for problems of various sizes.

The ratio of computed lower bounds and computational requirement versus matrix size are shown in Figure 3.1 and Figure 3.2 respectively for block a structure consisting of complex scalar uncertainties. The flop count for the SPA includes an upper bound computation, while the flop count for the standard optimization does not include the .mex file flops, where most of the computation occurs, so the difference between the algorithms is even more pronounced than that shown in Figure 3.2.

It can be seen that the SPA is much faster, even though the lower bound obtained is better. This advantage becomes more significant as the size of the problem grows. Note that NPSOL is finding local maximums for these problems; the SPA computes better bounds because it tends to find larger

[1]Information about NPSOL is available from the Stanford Office of Technology Licensing, 350 Cambridge Ave, Suite 250, Palo Alto, CA 94306

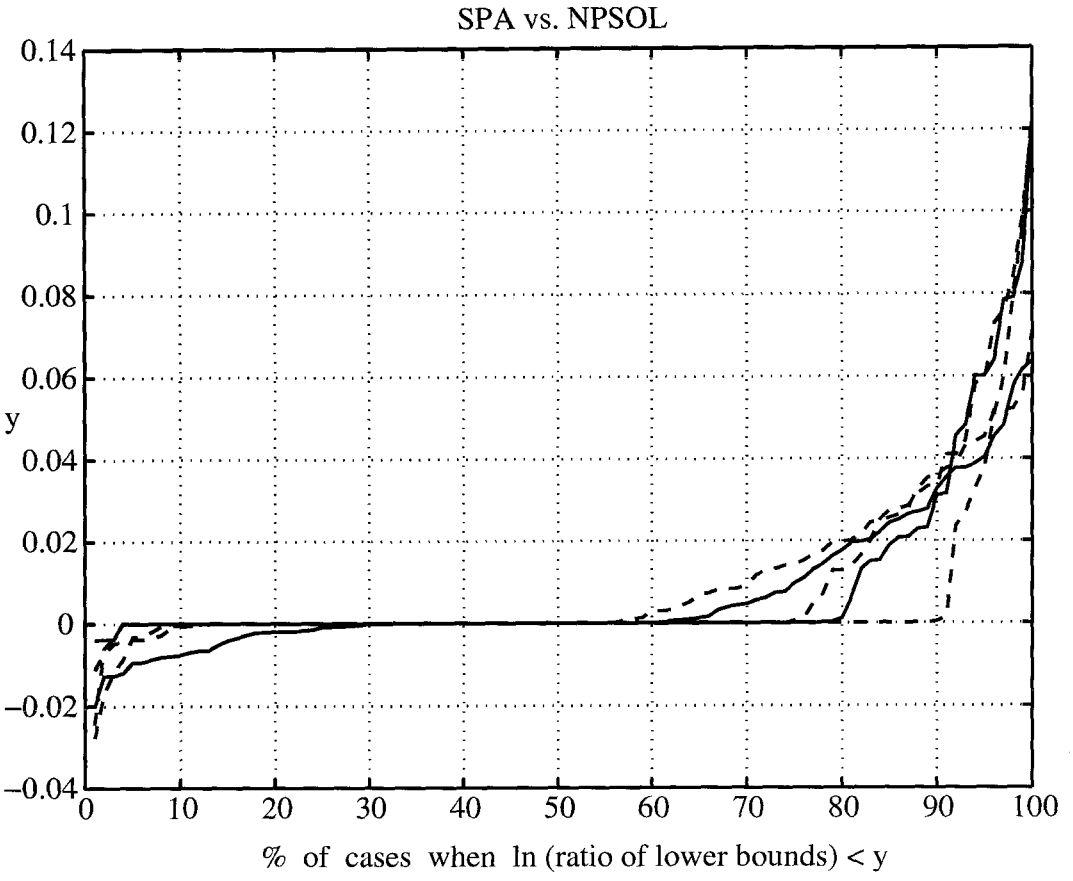


Figure 3.1: ln((SPA lower bound)/(NPSOL lower bound)) for problems of size 4 through size 30—the curves are not labeled as they are qualitatively the same. The SPA computes better bounds: its bounds are often significantly better, and seldom significantly worse than NPSOL's bounds.

local maximums. Larger local maximums have relatively larger regions of attraction in the SPA than they do in the gradient search.

For these reasons, our quest for better computation is based on improvements to the SPA.

3.3 The SPA Using the Rank One Solution

In this section we explain a connection between local maximums of $\rho_r(QM)$ and the solution of a rank one μ problem. The power iteration attempts to force the right and left eigenvectors of QM to satisfy the alignment condition of Theorem 3.4 associated with a local maximum of $\rho_r(QM)$. It turns out that this alignment condition is also associated with the solution to a certain rank one μ problem formed from the eigenvectors.

The resulting rank one μ problem leads to a new way of updating Q in

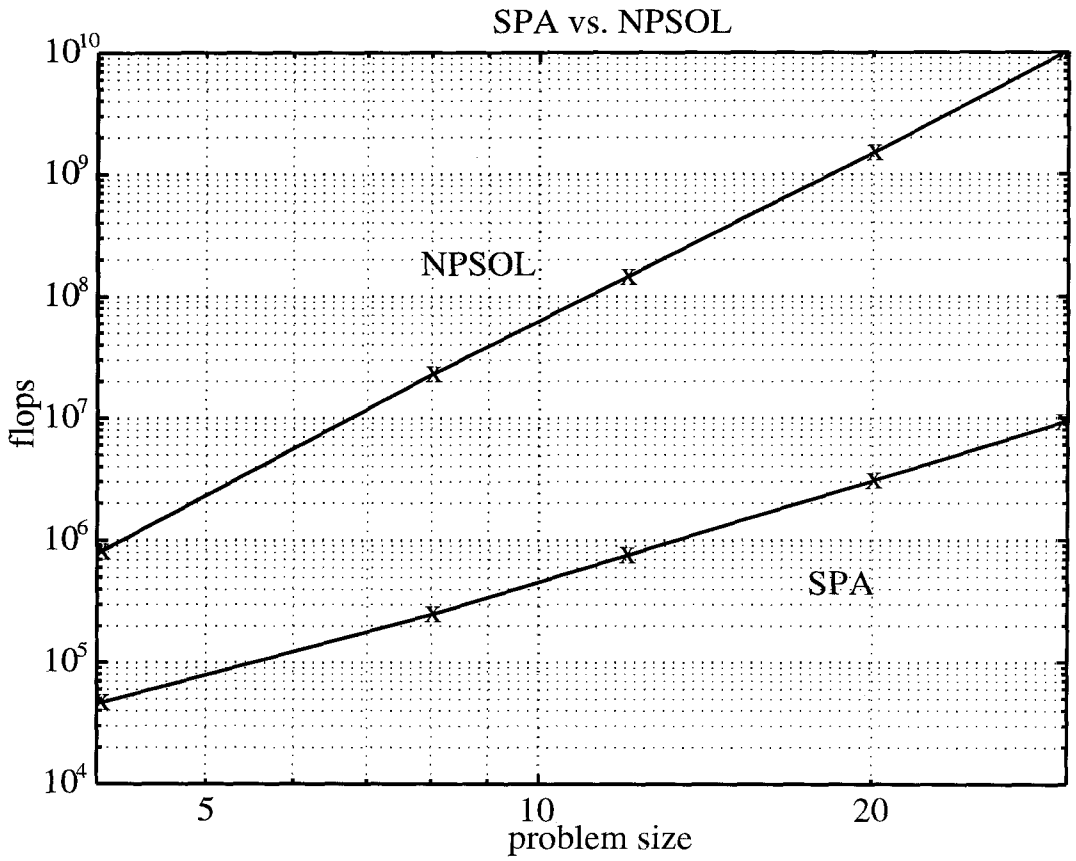


Figure 3.2: SPA flop count and a partial NPSOL flop count for problems of sizes 4, 8, 12, 20, and 30. A line connecting the data for each method shows the trend. Clearly, the SPA can solve much larger problems than NPSOL can.

the SPA. The new algorithm is nearly as fast as the SPA and provides better bounds. This is the simplest improvement over the SPA in this chapter.

The Rank One Algorithm (ROA)

The relationship between each μ problem and a corresponding rank one μ problem is shown by the following two theorems that connect the alignment conditions of Theorem 3.4 with a particular rank one μ problem.

Theorem 3.5 ([40]) *Suppose we have $M = zw^*$ and a compatible block structure, with z and $w \in \mathbb{C}^n$ satisfying the non-degeneracy assumptions. Further suppose we have $Q \in \partial \mathbf{B}_{\mathbf{\Delta}}$ with $q_r \neq 0$ for all real blocks $r \in \mathbf{r}$ such that $\rho_r(QM) = \beta > 0$. Then we have $\beta = \mu(M)$ iff there exists $D \in \mathbf{D}$ with $\theta_i = \pm\frac{\pi}{2}$ for $\|q_r\| < 1$ and $\psi \in (-\frac{\pi}{2}, \frac{\pi}{2})$ such that $w = e^{j\psi}DQz$.*

Remarks: The proof of Theorem 3.5 relies on Theorem 3.4, the fact that QM has only one nonzero eigenvalue, and the fact that a local maximum for the rank one problem is global.

Theorem 3.6 ([40]) *Suppose we have the matrix $M \in \mathbb{C}^{n\times n}$ and $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ such that QM has a real positive eigenvalue. Further suppose that $q_r \neq 0$ for all real blocks $r \in \mathbf{r}$, and that the corresponding right and left eigenvectors of QM, denoted by x and w respectively, satisfy the non-degeneracy assumption. Then there exists $D \in \mathbf{D}$ with $\theta_i = \pm\frac{\pi}{2}$ for $\|q_r\| < 1$ and $\psi \in (-\frac{\pi}{2}, \frac{\pi}{2})$ such that $w = e^{j\psi}Dx$ iff the matrix $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ solves the rank one μ problem $\max_{\widehat{Q}\in\partial\mathbf{B}_{\mathbf{\Delta}}} \rho_r(\widehat{Q}M_1)$ where $M_1 = zw^*$ and $z = Q^{-1}x$.*

Proof: By assumption we have $w^*x > 0$ so that $w^*Q\widehat{x} > 0$ and hence $\rho_r(QM_1) > 0$. The result follows from Theorem 3.5. ∎

Theorems 3.5 and 3.6 follow the notation in Theorem 3.4, and thus obscure the simplicity of the equivalent rank one solution. Since the local maximum at QM is characterized by conditions on the eigenvectors of QM corresponding to some eigenvalue β, we define an equivalence class of matrices M that have an eigenvalue β and the same corresponding eigenvectors at QM. The set of problems $\mu(M)$, where M is in the equivalence class, are equivalent in the sense that all satisfy the alignment conditions at Q. Furthermore, the problem of computing μ of rank one member of the equivalent class has no other local maximums. This rank one problem is easily solved, as in Section 2.4, and allows us to find $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ that is consistent with the alignment condition. This is used to modify the standard power iteration as follows.

- Start with initial guesses for x and $w \in \mathbb{C}^n$.
- ▶ Update z with the power step $\widetilde{\beta}z = Mx$.
- Compute the $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ that maximizes $\rho_r(Qzw^*)$.
- Update v with $v = Q^*w$.
- Update w with the power step $\widehat{\beta}w = M^*v$.
- Compute the $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ that maximizes $\rho_r(Qzw^*)$.
- Update x with $x = Qz$.
- If converged, then stop, else go to ▶.

Sometimes the rank one problem constructed from the eigenvectors of a local maximum has multiple solutions. Some of these solutions of the rank one mixed μ problem might not correspond to equilibrium points of the original problem. This can happen when two or more products $w_r^* z_r$ have the same phase or opposite phase. The ROA should be modified to deal with these cases better.

3.4 The Wrap in Reals Algorithm (WRA)

This section presents a more substantial modification to the SPA. The principal difficulty with the SPA is that if $\rho_r(QM) \ll \rho(QM)$ then the algorithm often does not converge. One way of thinking of this is that the "power" steps Equations (3.11) and (3.13) are increasing the component corresponding to $\rho(QM)$ faster than the rest of the steps can move towards the solution corresponding to $\rho_r(QM)$. The idea in this section is to utilize the good convergence properties of the SPA on complex blocks, while proceeding more cautiously on the real blocks.

For the remainder of this section we denote all complex blocks with the subscript c and separate the perturbation Q into its real and complex components, Q_r and Q_c, as follows.

$$Q = \begin{bmatrix} Q_r & 0 \\ 0 & Q_c \end{bmatrix}.$$

We partition the vectors v, w, x, and z and the matrix M compatibly:

$$v = \begin{bmatrix} v_r \\ v_c \end{bmatrix} \quad w = \begin{bmatrix} w_r \\ w_c \end{bmatrix} \quad x = \begin{bmatrix} x_r \\ x_c \end{bmatrix} \quad z = \begin{bmatrix} z_r \\ z_c \end{bmatrix}$$

$$M = \begin{bmatrix} M_{rr} & M_{rc} \\ M_{cr} & M_{cc} \end{bmatrix}.$$

The following theorem gives alignment conditions in terms of the real and complex block components in a way that allows for an algorithm that updates the real blocks independently of the complex blocks.

Theorem 3.7 ([37]) *For a given matrix $M \in \mathbb{C}^{n\times n}$ and a given uncertainty structure $\mathbf{\Delta}$, suppose we have $Q \in \mathbf{Q}$, non-zero vectors v, w, x, and z, and a positive scalar β such that $\beta I - M_{rr}Q_r$ is nonsingular. Further assume that $b_c \neq 0$, $z_c \neq 0$ and all the diagonal elements of Q_r are nonzero.*

Then the following conditions hold

$$\beta z = Mx \qquad \beta w = M^* v \tag{3.16}$$

$$x = Qz \qquad x = D^{-1} w \tag{3.17}$$

$$v = Q^* Q D z \qquad v = Q^* w$$

iff there exist non-zero vectors v_c, w_c, x_c, and z_c satisfying the following conditions

$$\beta z_c = (\frac{Q_r}{\beta} \star M)x_c \qquad \beta w_c = (\frac{Q_r}{\beta} \star M)^* v_c \tag{3.18}$$

$$x_c = Q_c z_c \qquad x_c = D_c^{-1} w_c \tag{3.19}$$

$$v_c = Q_c^* Q_c D_c z_c \qquad v_c = Q_c^* w_c$$

$$Q_r(\beta I - M_{rr}Q_r)^{-1}M_{rc}x_c = D_r^{-1}(\beta I - M_{rr}^* Q_r)^{-1}M_{cr}^* v_c. \tag{3.20}$$

Remarks: The proof involves nothing more than writing Equations (3.16) and (3.17) in the new notation and eliminating v_r, w_r, x_r, z_r from the equations. The resulting equations allow us to characterize a local maximum over Q_c for a fixed Q_r and further, to characterize when this is a local maximum over Q_r too.

An iterative algorithm that separates the real and complex blocks in this way follows.

- Start with some given values for x, w, β, Q_r, and M.
- ▶ Update z_c and $\widetilde{\beta}$ with the power step $\widetilde{\beta} z_c = (\frac{Q_r}{\beta} \star M)x_c$.
- Update v_c as in the SPA.
- Update w_c and $\widehat{\beta}$ with the power step $\widehat{\beta} w_c = (\frac{Q_r}{\beta} \star M)^* v_c$.
- Update x_c as in the SPA.
- If converged, then go to the next step, else go to ▶.
- Compute $z_r = (\beta I - M_{rr}Q_r)^{-1}M_{rc}x_c$ and $w_r = (\beta I - M_{rr}^* Q_r)^{-1}M_{cr}^* v_c$.
- Update Q_r.
- Update β.
- If converged, then stop, else go to ▶.

This specifies a class of algorithms that may update Q_r in a variety of ways. In the implementation of this algorithm, we run one rank one iteration first in order to get the starting values of w, x, β, and Q_r needed for the WRA.

Compared to the ROA, the WRA is computationally more expensive, but the convergence properties are much better, even with a simple Q_r update. It is thus reasonable to mix the two procedures, and only use the slower and more reliable one in the cases where the faster scheme fails to converge.

In the SPA at a local maximum of $\rho_r(MQ)$, the vector z is not necessarily in the direction of the largest eigenvalue of MQ, and the algorithm might not converge if $\rho_r(MQ) \ll \rho(MQ)$. In the WRA, however, at a local maximum of $\rho_r(MQ)$ the vector z_c always corresponds to the largest eigenvalue of $(\frac{Q_r}{\beta} \star M)Q_c$. The WRA may be thought of as first implementing the SPA on the complex part of the problem with the real part fixed (the SPA has very good convergence properties on complex problems), then improving the real part of the problem with the complex part fixed. While Tierno and Young ([37]) first used this idea of separating the real and complex parts of the problem, they did not exploit the convergence properties of the SPA on complex problems. Thus it is not surprising that the WRA has significantly better convergence properties than both the SPA and the algorithm in [37].

3.5 Shift and Inverse Algorithm (SIA)

Another way to find an eigensolution that does not correspond to $\rho(MQ)$ is based on the observation that if $\lambda z = MQz$ and β^s is not an eigenvalue then

$$(\lambda - \beta^s)^{-1} z = (MQ - \beta^s I)^{-1} z.$$

When λ and β^s are close to each other, $(\lambda - \beta^s)^{-1}$ is a large eigenvalue of $(MQ - \beta^s I)^{-1}$, and a power iteration based on this equation converges to the eigensolution $\lambda z = MQz$ of MQ. Such an iteration is called a shift and inverse iteration.

We use this idea to modify the SPA. The Q updates are as in the SPA, except that now Q must be formed explicitly. This algorithm, called the SIA, proceeds as follows.

- Start with a Q, a β^s close to an eigenvalue of MQ, a z close to a right eigenvector of MQ, and a w close to a left eigenvector of QM.

- ▶ Update z_k and $\widetilde{\beta}$ with the inverse power step
  $(\widetilde{\beta} - \beta^s)^{-1} z_k = (MQ - \beta^s I)^{-1} z_{k-1}$ with $\widetilde{\beta} \in \mathbb{R}^+$ and $\|z_k\| = 1$.

- Update $Q \in \partial \mathbf{B}_{\mathbf{\Delta}}$ and β^s.

- Update w_k and $\widehat{\beta}$ with the inverse power step
  $(\widehat{\beta} - \beta^s)^{-1} w_k = (M^* Q^* - \beta^s I)^{-1} w_{k-1}$ with $\widehat{\beta} \in \mathbb{R}^+$ and $\|w_k\| = 1$.

- Update $Q \in \partial \mathbf{B}_{\mathbf{\Delta}}$ and β^s.

- If converged, then stop, else go to ▶.

We may also perform the inverse power step multiple times before updating Q and β.

This algorithm converges quickly when we start close to an eigensolution. When we do not start close to an eigensolution, however, it performs poorly. Thus it is appropriate to use it only in conjunction with some other algorithm that gets near a solution first.

3.6 Combining the Algorithms

In previous sections we introduced three new algorithms for mixed μ lower bound computation. Here, we combine them in a way that utilizes their respective good qualities and avoids their bad qualities. All the algorithms have enhanced convergence properties compared to the SPA, but are also more computationally costly.

Since the ROA is computationally the least expensive—nearly as inexpensive as the SPA—we want to preserve the efficiency of the ROA for those problems where the ROA converges, while improving convergence and therefore accuracy for those problems where the ROA fails to converge.

In the cases where the ROA fails to converge we need to continue looking for a local maximum with one of our other algorithms. Note that all our algorithms start with some guess for the perturbation or the vectors as input. If these guesses are close to a local maximum then the algorithms perform particularly well. Consequently, when we decide to switch to another algorithm in the middle of the computation, we can take advantage of the computation already performed. Knowing that the SIA has particularly good properties only when we are quite close to the equilibrium point, we choose to continue with the WRA because it works better than the SIA when we are not so close to an equilibrium point. In the case that the WRA also fails to converge—typically it just slows down, often near the optimum (this might be remedied with a better Q_r update)—we continue with the SIA.

The following combination of the algorithms defines a new algorithm, denoted as CPA, for combined power algorithm. This is the CPA algorithm used in the next section, (i.e. this is the scheduling used).

- Run the ROA for up to 50 iterations.
- If not yet converged, then run the WRA for up to 50 iterations.
- If not yet converged, then run the SIA for up to 50 iterations.
- If not converged, then construct a lower bound from the current perturbation.

This scheduling reflects experience from testing not presented here. Certainly, there is room for improvement. In particular, the scheduling should depend on a priori knowledge of the problem.

3.7 Numerical Experience

The nature of the mixed μ problem is such that the only meaningful way to evaluate an algorithm is by testing it on a large number of representative problems. This section presents a comparison of several algorithms, each run on the same type of problems. It also shows how the performance of the best algorithm depends on problem size.

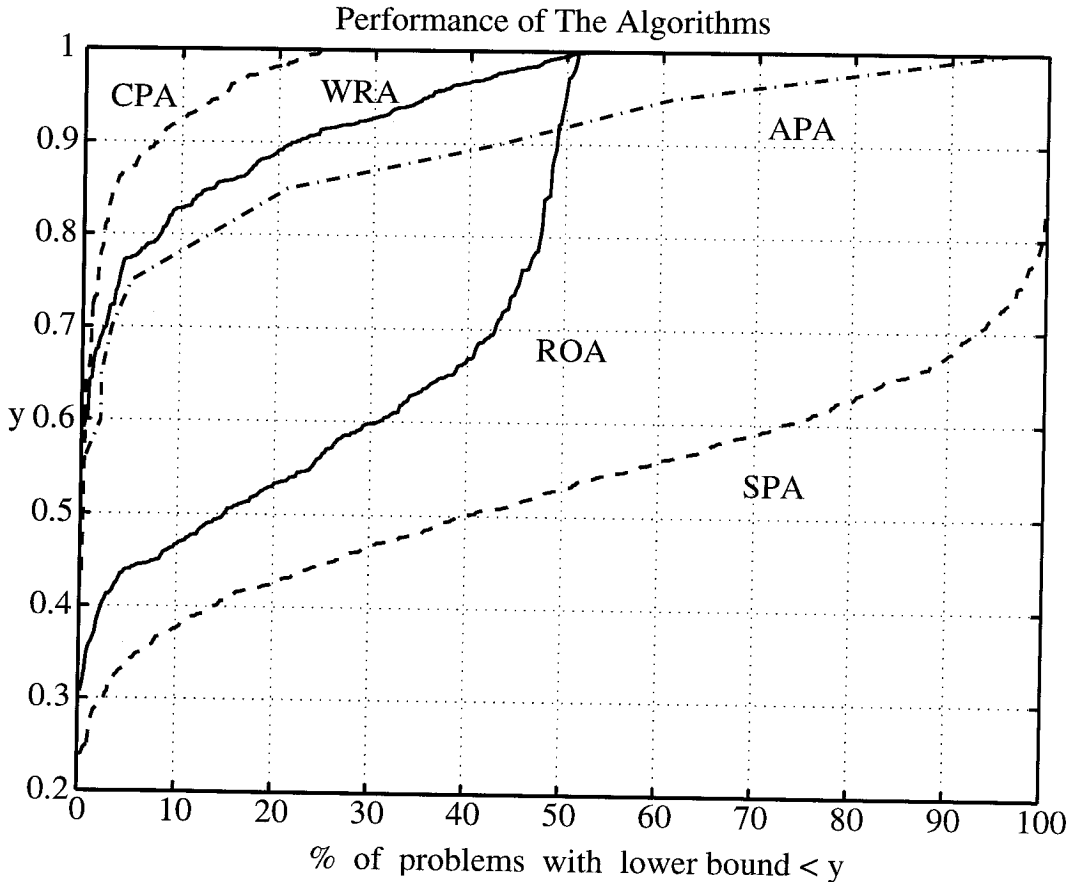


Figure 3.3: Algorithm performance on 500 hard problems.

As discussed in Chapter 2, it is desirable to be able to generate nontrivial problems for which we know μ in order to test algorithms. We use the procedure in Section 2.7 to generate a class of `nogap` matrices that also typically satisfy $\rho_r(QM) \ll \rho(QM)$. We denote these matrices as the set $\mathcal{R}_{\mathcal{B}}$. We emphasize that these are particularly difficult problems for mixed μ lower bound

computation. This is desirable because existing algorithms work well on most problems; the point of this research is to do better in the cases where existing code is inadequate.

Figure 3.3 shows a comparison between the algorithms described here and also the algorithm of Tierno and Young in [37], denoted as APA. The SIA is not included because it performs poorly unless it is given a good starting point; it is only meaningful in conjunction with another algorithm. We tested the algorithms on 500 matrices in the set $\mathcal{R}_{\mathcal{B}}$ each with 4 real parameters, 2 scalar complex blocks and one 2×2 complex block in the block structure, and with $\mu = 1$. The CPA is a dramatic improvement over the SPA.

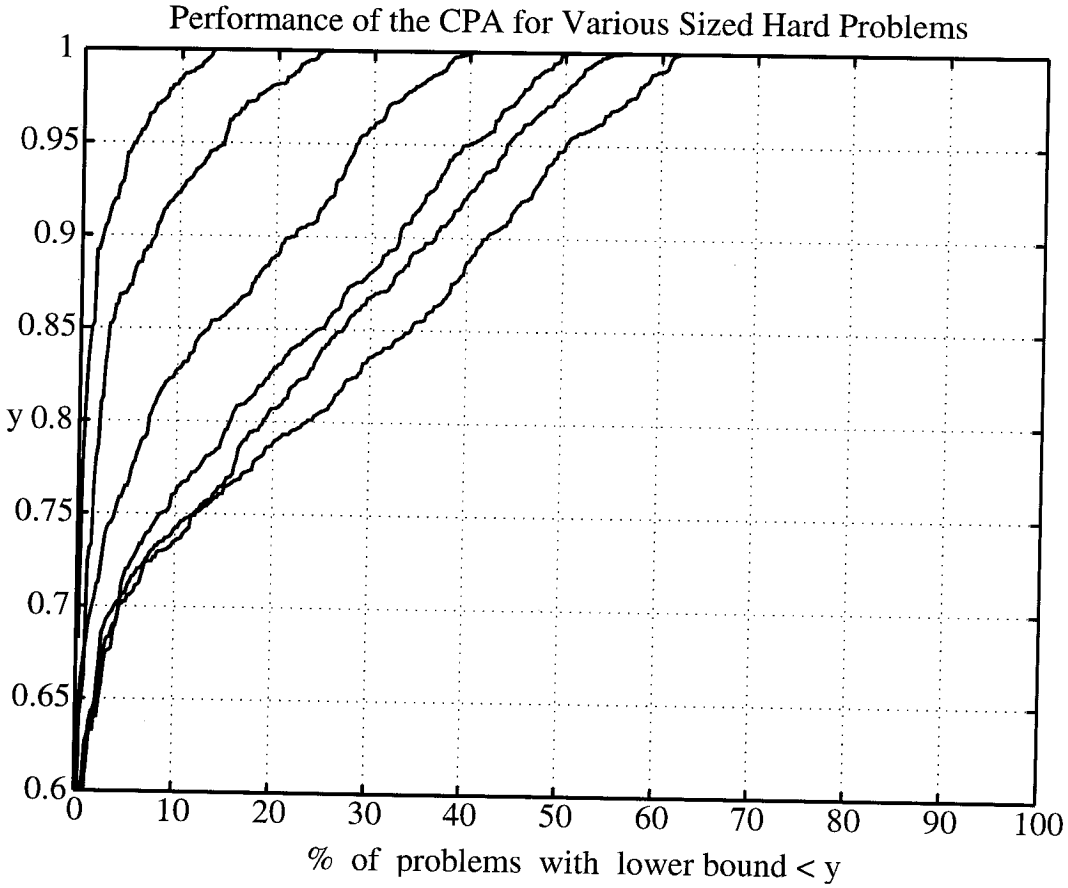


Figure 3.4: Computation becomes more difficult as problem size increases from 2 to 12 real parameters. A line is plotted for each set of problems. Each set contains problems of a single size, either 2, 4, 6, 8, 10, or 12 real parameters. The computation is the CPA on $\mathcal{R}_{\mathcal{B}}$ matrices.

Figure 3.4 shows how the bound computation using the best algorithm, the CPA, becomes more difficult as the problem size increases. The best performance shown is with 2 real parameters, and the worst is with 12. All problems were particularly hard problems from the set $\mathcal{R}_{\mathcal{B}}$ with $\mu = 1$, and

with 2 scalar complex blocks and one 2×2 complex block in the block structure. These results further motivate research in better bounds computation.

Chapter 4

The Upper Bound

Although the lower bound computation described in the preceding chapter typically calculates μ quite accurately, the lower bound by itself is not satisfactory. Without a high quality upper bound, we have no way of knowing if the lower bound is reasonably accurate for a particular problem.

Since the μ upper bound from Theorem 4.4 is a convex optimization problem, we could try to solve it using a variety of convex programming techniques. For instance we know that gradient search methods lead us to the minimum eventually, although they may be slow. (Even though the upper bound is not in general differentiable if the maximum eigenvalue is repeated, it is possible to compute a generalized gradient that gives a descent direction.) Since such approaches to the problem have not been entirely satisfactory, we would like to exploit the specific structure of the problem in order to speed up the computation.

4.1 An Infimization Problem

We now consider an upper bound to μ. First note that for any $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$ and any $D \in \mathbf{D}$, $D\Delta = \Delta D$. From the simple equivalence $\det(I - \Delta M) = \det(I - \Delta D^{-1} D M) = \det(I - D^{-1}\Delta D M) = \det(I - \Delta D M D^{-1})$ we immediately have the following invariance property.

Lemma 4.1 *For any matrix $M \in \mathbb{C}^{n\times n}$, any compatible block structure, and any $D \in \mathbf{D}$,*

$$\mu(M) = \mu(DMD^{-1}). \tag{4.1}$$

In light of Equation (4.1), we refine the upper bound in Equation (1.5) to obtain the following.

Theorem 4.2 *For any matrix $M \in \mathbb{C}^{n\times n}$ and any compatible block structure,*

$$\mu(M) \leq \inf_{D\in\mathbf{D}} \overline{\sigma}(DMD^{-1}). \tag{4.2}$$

This is the standard upper bound from complex μ theory.

The upper bound in Theorem 4.2 is equivalent to the LMI upper bound in Theorem 4.3 in that the two infima in Equations (4.2) and (4.3) are the same.

Theorem 4.3 *For any matrix $M \in \mathbb{C}^{n\times n}$ and any compatible block structure,*

$$\mu(M) \leq \inf_{D\in\mathbf{D}} \left[\min_{\alpha\in\overline{\mathbb{R}^+}} \{ \alpha \mid (M^*DM - \alpha^2 D) \leq 0 \} \right]. \qquad (4.3)$$

The set $\mathbf{D}$ is the same for mixed μ problems as for complex μ problems. If we use Equation (4.2) or Equation (4.3) for an upper bound for a mixed μ problem, we are doing nothing more than replacing the real parameters with complex parameters (thus covering the admissible perturbation set $\mathbf{B}_{\mathbf{\Delta}}$ with a larger set) and using the complex upper bound. Thus this bound does not exploit the phase constraint of the real perturbations. For this reason one might expect that this is often a poor bound to the mixed μ problem, and indeed this is the case.

The upper bound presented in [16] by Fan et al. does exploit this phase constraint and gives a bound that is never worse than the standard upper bound from complex μ theory, and is frequently much better.

Theorem 4.4 ([16]) *For any matrix $M \in \mathbb{C}^{n\times n}$ and any compatible block structure,*

$$\mu(M) \leq \inf_{\substack{D\in\mathbf{D}\\ G\in\mathbf{G}}} \left[\min_{\alpha\in\overline{\mathbb{R}^+}} \{ \alpha \mid (M^*DM + j(GM - M^*G) - \alpha^2 D) \leq 0 \} \right]. \qquad (4.4)$$

Remarks: Both Theorem 4.4 and Theorem 4.3 are special cases of Theorem 5.9, which is proved in Chapter 5.

The difference between Equation (4.3) and Equation (4.4) is fairly intuitive. If $x \in \ker(I - \Delta M)$, then $x^* j(GM - M^*G)x = 0$. Theorem 4.4 uses the scaled small gain of Theorems 4.2 and 4.3 on the subspace $\bigcup_{\Delta\in\mathbf{B}_{\mathbf{\Delta}}} \ker(I - \Delta M)$.

It is clear that if we enforce the choice $G = 0$ in Equation (4.4), then we recover the standard complex μ upper bound in Equation (4.3). Thus the phase constraint of the real parameters is being exploited to give us extra degrees of freedom in the G scaling matrix and to obtain a better bound.

Since the above minimization involves an LMI it is convex, so that all local minima are global, and hence is computationally attractive.

The algorithm implementation described in Section 4.3 relies on the fact that the upper bound may be reformulated several different ways, as stated in the following lemma.

Lemma 4.5 *Suppose we have a matrix $M \in \mathbb{C}^{n\times n}$ and a real scalar $\alpha > 0$. Then the following statements are equivalent.*

I *There exist matrices $D_I \in \mathbf{D}$ and $G_I \in \mathbf{G}$ such that*

$$\overline{\lambda}\left(M^*D_IM + j(G_IM - M^*G_I) - \alpha^2 D_I\right) \leq 0. \tag{4.5}$$

II *There exist matrices $D_{II} \in \widehat{\mathbf{D}}$ and $G_{II} \in \widehat{\mathbf{G}}$ (or $D_{II} \in \mathbf{D}$ and $G_{II} \in \mathbf{G}$) such that*

$$\overline{\lambda}\left(M^*_{D_{II}}M_{D_{II}} + j(G_{II}M_{D_{II}} - M^*_{D_{II}}G_{II})\right) \leq \alpha^2 \tag{4.6}$$

where we denote $M_D \triangleq DMD^{-1}$.

III *There exist matrices $D_{III} \in \widehat{\mathbf{D}}$ and $G_{III} \in \widehat{\mathbf{G}}$ (or $D_{III} \in \mathbf{D}$ and $G_{III} \in \mathbf{G}$) such that*

$$\overline{\sigma}\left(\left(\frac{M_{D_{III}}}{\alpha} - jG_{III}\right)(I + G^2_{III})^{-\frac{1}{2}}\right) \leq 1. \tag{4.7}$$

IV *There exist matrices $D_{IV} \in \widehat{\mathbf{D}}$ and $G_{IV} \in \widehat{\mathbf{G}}$ (or $D_{IV} \in \mathbf{D}$ and $G_{IV} \in \mathbf{G}$) such that*

$$\overline{\sigma}\left((I + G^2_{IV})^{-\frac{1}{4}}\left(\frac{M_{D_{IV}}}{\alpha} - jG_{IV}\right)(I + G^2_{IV})^{-\frac{1}{4}}\right) \leq 1. \tag{4.8}$$

Proof: With the singular value decompositions for D and G and the definitions

$$\begin{aligned}
D = U_D\Sigma_D U_D^* \qquad & D_a = D^2 \\
D_c = (1+G^2)^{-1/4}D \qquad & D_c = (1+\Sigma_G^2)^{-1/4}U_G^*D \\
M_b = DMD^{-1} \qquad & M_{b_1} = U_G^*DMD^{-1}U_G \\
G = U_G\Sigma_G U_G^* \qquad & G_a = \alpha DGD \\
G_b = \alpha G \qquad & G_{b_1} = \alpha\Sigma_G \\
G_2 = (I+G^2)^{-1/2} \qquad & G_{21} = (I+\Sigma_G^2)^{-1/2} \\
G_4 = (I+G^2)^{-1/4} \qquad & G_{41} = (I+\Sigma_G^2)^{-1/4},
\end{aligned}$$

it is easy to see that the following are equivalent.

$$\begin{aligned}
&M^*D_aM + j(G_aM - M^*G_a) - \alpha^2 D_a \leq 0 \\
&\overline{\lambda}(M^*D_aM + j(G_aM - M^*G_a) - \alpha^2 D_a) \leq 0
\end{aligned}$$

$$M^*D_aM + j(G_aM - M^*G_a) \leq \alpha^2 D_a$$

$$M_b^* M_b + j(G_b M_b - M_b^* G_b) \leq \alpha^2 I$$
$$\overline{\lambda}(M_b^* M_b + j(G_b M_b - M_b^* G_b)) \leq \alpha^2$$
$$\overline{\lambda}(M_{b_1}^* M_{b_1} + j(G_{b_1} M_{b_1} - M_{b_1}^* G_{b_1})) \leq \alpha^2$$
$$M_{b_1}^* M_{b_1} + j(G_{b_1} M_{b_1} - M_{b_1}^* G_{b_1}) \leq \alpha^2 I$$

$$\tfrac{M_b^* M_b}{\alpha^2} + j\left(G_b \tfrac{M_b}{\alpha} - \tfrac{M_b^*}{\alpha} G_b\right) \leq I$$

$$\left(\tfrac{M_b}{\alpha} - jG_b\right)^* \left(\tfrac{M_b}{\alpha} - jG_b\right) - G^2 \leq I$$

$$\left(\tfrac{M_b}{\alpha} - jG_b\right)^* \left(\tfrac{M_b}{\alpha} - jG_b\right) \leq I + G^2$$

$$\left(\left(\tfrac{M_b}{\alpha} - jG_b\right) G_2\right)^* \left(\left(\tfrac{M_b}{\alpha} - jG_b\right) G_2\right) \leq I$$

$$\overline{\sigma}\left(\left(\tfrac{M_b}{\alpha} - jG_b\right) G_2\right) \leq 1$$
$$\overline{\sigma}\left(\left(\tfrac{M_{b_1}}{\alpha} - j\Sigma_G\right) G_{21}\right) \leq 1$$

$$\overline{\sigma}\left(G_4 \left(\tfrac{D_c M D_c^{-1}}{\alpha} - jG\right) G_4\right) \leq 1$$
$$\overline{\sigma}\left(G_{41} \left(\tfrac{D_{c_1} M D_{c_1}^{-1}}{\alpha} - j\Sigma_g\right) G_{41}\right) \leq 1.$$

The equivalences in the Lemma (and more) follow easily. ∎

Remarks: The equivalence between **I**, **II**, **III** for D_I, D_{II}, and D_{III} and $\in \mathbf{D}$ and G_I, G_{II}, and G_{III} and $\in \mathbf{G}$ was shown by Fan et al. ([16]). From the proof we can easily obtain the formulae to convert between the various forms.

A useful consequence of the lemma is the following theorem.

Theorem 4.6 *For any matrix $M \in \mathbb{C}^{n\times n}$ and any compatible block structure let*

$$B(M, \widehat{D}, \widehat{G}, \alpha) = \overline{\sigma}\left((I + \widehat{G}^2)^{-\frac{1}{4}} \left(\frac{\widehat{D} M \widehat{D}^{-1}}{\alpha} - j\widehat{G}\right) (I + \widehat{G}^2)^{-\frac{1}{4}}\right).$$

Then

$$\mu(M) \leq \inf_{\substack{\widehat{D}\in\widehat{\mathbf{D}} \\ \widehat{G}\in\widehat{\mathbf{G}}}} \left[\min_{\alpha\in\overline{\mathbb{R}^+}} \left\{ \alpha \;\middle|\; B(M, \widehat{D}, \widehat{G}, \alpha) \leq 1 \right\}\right], \tag{4.9}$$

and this infimum is the same as the infimum in Theorem 4.4.

Each of the two different formulations of the upper bound, in Theorems 4.4 and 4.6, has its advantages. The problem statement from Equation (4.4) has the advantages that it is linear in the matrices D and G and is convex, and hence does not have difficulties associated with local but not global minima. The problem statement from Equation (4.9) has the advantages that it is minimizing the norm of a matrix, offering some numerical advantages, that $\widehat{D}$ enters the problem exactly as in the standard complex μ upper bound, that $\widehat{G}$ enters the problem in a balanced symmetric fashion, and that $\widehat{G}$ is now a real diagonal matrix.

Since the upper bound is convex there is a wide variety of numerical techniques we could apply to this minimization. However, even for medium size problems ($n < 100$), the optimization over the D and G scaling matrices could involve several thousand parameters, depending on the block structure $\mathbf{\Delta}$. Therefore, in order to handle problems of this size with reasonable computation times, a straight forward application of standard optimization techniques will not suffice. Instead, we must exploit the specific structure of the problem to develop an efficient algorithm that can handle problems of this size.

4.2 A Theoretical Framework for the Problem

This section provides a further connection between μ and the upper bound of Theorem 4.4. The material in this section is largely due to Young ([39]), and is included for completeness only; we do not use the results elsewhere.

To examine the equivalence between μ and its upper bound, we first consider under what conditions a given pair of scaling matrices D_0 and G_0 yields a value of α that is equal to the minimum of the upper bound infimization, and under what conditions α is equal to μ. The remainder of this subsection presents some of the ideas and states without proof four theorems that address these issues.

Suppose we have matrices $M \in \mathbb{C}^{n\times n}$, $D_0 \in \mathbf{D}$ and $G_0 \in \mathbf{G}$ and a real scalar $\alpha > 0$ such that $\overline{\lambda}(M^*D_0M + j(G_0M - M^*G_0) - \alpha^2 D_0) = 0$ with r eigenvalues coalesced at the maximum. Further suppose that the eigenvectors are given by $U_0 \in \mathbb{C}^{n\times r}$ where $(M^*D_0M + j(G_0M - M^*G_0) - \alpha^2 D_0)U_0 = 0$ and $U_0^*D_0U_0 = I_r$. Then the question of whether or not we can find a pair of scaling matrices $D \in \mathbf{D}$ and $G \in \mathbf{G}$ to improve upon D_0 and G_0 can be related to whether or not there exists a vector $\widehat{\eta} \in \mathbb{C}^r$, $\|\widehat{\eta}\| = 1$ such that the quantity $\phi_{D,G}(\eta) \triangleq \eta^*U_0^*(M^*DM + j(GM - M^*G) - \alpha^2 D)U_0\eta$ satisfies $\phi_{D,G}(\widehat{\eta}) \leq 0$. The function $\phi_{D,G}(\eta)$ can in turn be written as $\phi_{D,G}(\eta) =< (D,G),(D(\eta),G(\eta))>=<D,D(\eta)> + <G,G(\eta)>$, an inner product between the pair (D,G) and the pair $(D(\eta),G(\eta))$ parametrized by the vector η. This can be used to define a set $\nabla_{\mathcal{Y}}$ as the set of all such pairs $(D(\eta),G(\eta))$ for $\|\eta\| =$

1, together with an extended set $\widehat{\nabla}_y \supset \nabla_y$. The details of these constructions are given by Young and Doyle in [42], and the analogous constructions for the complex μ case are given by Packard and Doyle in [26]. The relationship between the minima of the upper bound function and μ is intimately related to the nature of these two sets. This is stated explicitly in the following two theorems.

Theorem 4.7 *Suppose we have matrices $M \in \mathbb{C}^{n\times n}$, $D_0 \in \mathbf{D}$ and $G_0 \in \mathbf{G}$ and a real scalar $\alpha > 0$ such that $\overline{\lambda}(M^*D_0M + j(G_0M - M^*G_0) - \alpha^2 D_0) = 0$ with r eigenvalues coalesced at the maximum. Further suppose that the eigenvectors are given by $U_0 \in \mathbb{C}^{n\times r}$ where $(M^*D_0M + j(G_0M - M^*G_0) - \alpha^2 D_0)U_0 = 0$ and $U_0^*D_0U_0 = I_r$. Then D_0 and G_0 are minimizing arguments of the upper bound problem*

$$\alpha = \inf_{\substack{D\in\mathbf{D}\\ G\in\mathbf{G}}} \left[\min_{\widehat{\alpha}\in\overline{\mathbb{R}^+}} \left\{ \widehat{\alpha} \;\middle|\; (M^*DM + j(GM - M^*G) - \widehat{\alpha}^2 D) \leq 0 \right\} \right] \tag{4.10}$$

if and only if $0 \in \mathrm{co}(\nabla_y)$.

Theorem 4.8 *Suppose we have $M \in \mathbb{C}^{n\times n}$ and $U_0 \in \mathbb{C}^{n\times r}$ and $\alpha > 0$ defined as in Theorem 4.7. Then $\alpha = \mu(M)$ if and only if $0 \in \widehat{\nabla}_y$.*

This type of theoretical framework has been quite successful in analyzing the complex μ problem, and should be similarly useful for the mixed μ problem. The following theorem, due to Fan et al. is also reminiscent of an earlier result for complex μ.

Theorem 4.9 ([16]) *Suppose we have $M \in \mathbb{C}^{n\times n}$, then provided the infimum in Equation (4.4) is achieved and the corresponding largest eigenvalue of $(M^*DM + j(GM - M^*G) - \alpha^2 D)$ is distinct, then $\mu(M)$ is equal to its upper bound from Theorem 4.4.*

It is not always possible to improve upon the complex μ upper bound via the G scaling matrix as is illustrated in the following theorem, stated without proof.

Theorem 4.10 *Suppose we have a real matrix $M \in \mathbb{R}^{n\times n}$ and a block structure with none of the real scalars repeated. Then an optimal choice for G in Equation (4.4) is $G = 0$.*

This is an important class of problems: we encounter μ problems where M is real at both low and high frequencies when M is constructed from State Space A, B, C, and D matrices. This theorem does not apply if any of the real parameters are repeated.

4.3 The Upper Bound Algorithm

The upper bound algorithm described here uses a mixture of the formulations in Lemma 4.5. We begin with the problem in the form of Equation (4.9), so we can use some methods from the complex μ computation, together with various other techniques, to obtain a fairly good estimates of $\widehat{D}$, $\widehat{G}$ and α. These are then converted into an initial guess for the problem in the form of Equation (4.4) and the algorithm improves on these. More specifically, the algorithm is as follows:

1. In order to balance the matrix M we first compute the $\widehat{D}$ that solves $\inf_{\widehat{D}\in\widehat{\mathbf{D}}} \left\|\widehat{D}M\widehat{D}^{-1}\right\|_F$ using a generalization of Osborne's method ([24]), as in the standard complex μ upper bound. The matrix $\widehat{M} \triangleq \widehat{D}M\widehat{D}^{-1}$ is balanced and satisfies $\mu(M) = \mu(\widehat{D}M\widehat{D}^{-1})$, This procedure provides our initial guess for $\widehat{D} \in \widehat{\mathbf{D}}$. $\overline{\sigma}(\widehat{M})$ is an upper bound to the complex version of the problem.

2. The lower bound is now computed using the SPA algorithm from Chapter 3, applied to the balanced matrix $\widehat{M}$.

3. For any fixed level of α form each (nonzero) block of G as $G_r = \frac{1}{2j\alpha}(\widehat{M}_r - \widehat{M}_r^*)$ where $\widehat{M}_r$ is the corresponding sub-matrix of $\widehat{M}$ (so that jG_r cancels the skew-Hermitian part of $\frac{\widehat{M}_r}{\alpha}$). Then bisect between the lower and current upper bound to find the smallest α such that

$$\overline{\sigma}\left((I+G^2)^{-\frac{1}{4}}\left(\frac{\widehat{M}}{\alpha} - jG\right)(I+G^2)^{-\frac{1}{4}}\right) \leq 1.$$

   Convert to $\widehat{G} \in \widehat{\mathbf{G}}$ by performing the singular value decomposition $G = U\Lambda U^*$, redefining $\widehat{G}$ as Λ, and absorbing the U matrix into $\widehat{D} \in \widehat{\mathbf{D}}$ and $\widehat{M}$.

4. Now that we have initial guesses for $\widehat{D} \in \widehat{\mathbf{D}}$ and $\widehat{G} \in \widehat{\mathbf{G}}$, we compute a descent direction for $\widehat{G} \in \widehat{\mathbf{G}}$ together with an appropriate step length. This step and a second are taken, resulting in a new $\widehat{G}$.

5. The matrix $\widehat{D} \in \widehat{\mathbf{D}}$ is updated by computing a diagonal matrix $\widehat{D}_d \in \widehat{\mathbf{D}}$ (so that it commutes with $\widehat{G}$) which minimizes

$$\left\|\widehat{D}_d(I+G^2)^{-\frac{1}{4}}\left(\frac{\widehat{M}}{\alpha} - jG\right)(I+G^2)^{-\frac{1}{4}}\widehat{D}_d^{-1}\right\|_F$$

   again using a generalization of Osborne's method. We then absorb $\widehat{D}_d$ into $\widehat{D} \in \widehat{\mathbf{D}}$.

6. Step 4 is repeated.

7. We now have guesses for $\widehat{D} \in \widehat{\mathbf{D}}$, $\widehat{G} \in \widehat{\mathbf{G}}$ and α for the upper bound problem in Equation (4.9). These are converted into $D \in \mathbf{D}$ and $G \in \mathbf{G}$ that form guesses for the upper bound problem in Equation (4.4). We now improve these guesses using a descent algorithm which iteratively computes a descent direction and an appropriate step length, for both $D \in \mathbf{D}$ and $G \in \mathbf{G}$ simultaneously. At each step we compute a new upper bound by solving the associated eigenvalue problem, and quit when the bound stops decreasing (within tolerance).

The balancing in step 1 of the algorithm serves several purposes. First we obtain a $\widehat{D} \in \widehat{\mathbf{D}}$ that approximately solves $\inf_{\widehat{D} \in \widehat{\mathbf{D}}} \overline{\sigma}(\widehat{D} M \widehat{D}^{-1})$, or in other words the standard upper bound to the associated complex μ problem. Since we have reformulated the problem in Equation (4.9) so that the $\widehat{D}$ matrix enters exactly as in the complex μ upper bound, and the $\widehat{G}$ matrix enters in a balanced symmetric fashion, this $\widehat{D}$ matrix also serves as a good first guess for the mixed μ upper bound. A good deal of numerical experience with the generalized Osborne's method for computing the complex μ upper bound has shown that it is fast and usually works well. Thus by reformulating the problem in this fashion we can exploit these properties in the mixed μ upper bound too. This balancing also numerically preconditions the problem, and can greatly improve the performance of the subsequent steps.

In step 3 of the algorithm we generate our initial guess for $\widehat{G}$. The approach is somewhat intuitive, and although there are no guarantees, it appears to work quite well. Thus our $\widehat{D}$ and $\widehat{G}$ estimates, which require little computation time, are usually fairly good before we enter the descent portion of the algorithm, and hence we can restrict ourselves to a small number of descent steps. This is crucial in obtaining a fast implementation, since the descent steps are quite computationally expensive, and may require many steps for even modest improvement.

In step 7 we compute a descent direction for $D \in \mathbf{D}$ and $G \in \mathbf{G}$, together with an appropriate step length. We compute matrix descent directions for D and G simultaneously by computing a generalized gradient of the upper bound function. In this way we avoid separate computation for the individual elements of the D and G matrices. This is important not only for speed of computation, but also because in the case of repeated eigenvalues there may not be a descent direction with respect to any individual elements of D and G, when there is a descent direction if all the elements are allowed to move simultaneously. In the case that the maximum eigenvalue is distinct, this descent direction coincides with the usual gradient direction. The step length computation is somewhat ad hoc, but ensures that the maximum eigenvalue of the upper bound function decreases, and that we satisfy the constraint $D > 0$.

Similar comments with regard to the computation of descent directions and step lengths apply to steps 4 and 6.

This implementation of the upper bound results in an algorithm that is quite efficient and can handle medium size problems ($n < 100$) with reasonable computational requirements. Results regarding both the quality of the bounds and their computational requirements (as a function of problem size) are presented in Section 4.4. This algorithm has been implemented as as Matlab function (.m file), and is currently available in conjunction with the μ-Tools toolbox ([2]). The algorithm returns upper and lower bounds for $\mu(M)$, together with appropriate scaling matrices $\widehat{D} \in \widehat{\mathbf{D}}$, $\widehat{G} \in \widehat{\mathbf{G}}$ for the upper bound problem in Equation (4.9), and $Q \in \partial\mathbf{B}_{\mathbf{\Delta}}$ for the lower bound problem Equation (3.2).

The mixed μ upper bound in the form of Equation (4.4) is a class of LMI problems. The solution of LMIs is a subject of much research interest right now, since they appear in many control problems (see Doyle et al., [14]). This algorithm represents a first attempt at solving one particular LMI. As more refined algorithms for the solution of LMIs appear, then they can be used to improve the μ upper bound computation (see Beck and Doyle, [6]).

4.4 Algorithm Performance

The main issues we are interested in with regard to the algorithm performance are the computational requirements of the algorithm and the accuracy of the resulting bounds. Recalling Chapter 2, we are interested in the typical performance of the algorithm, rather than the worst-case performance, so we run the algorithm repeatedly on a class of random problems and collect statistical data.

One test is to examine the average computational requirements for the algorithm versus matrix size. The results are shown in Figure 4.1. The test problems were `crand` matrices with block structures consisting of all scalar uncertainties, 90 percent of them chosen as real and the rest complex. The results are typical of other block structures. The same data for the appropriate complex μ problem is shown for comparison. The results were obtained by running Matlab on a Sparc 1 workstation, and it can be seen that we can reasonably expect to handle problems of size 10 in about 10 seconds, and problems of size 50 in about 2-3 minutes. Modern computers are substantially faster.

It also can be seen that the experimental growth rate in computation time for the existing implementation is approximately n^2. This is probably an artifice of the implementation in Matlab, an interpretive language. A more realistic measure of the computational growth rate is in terms of total floating

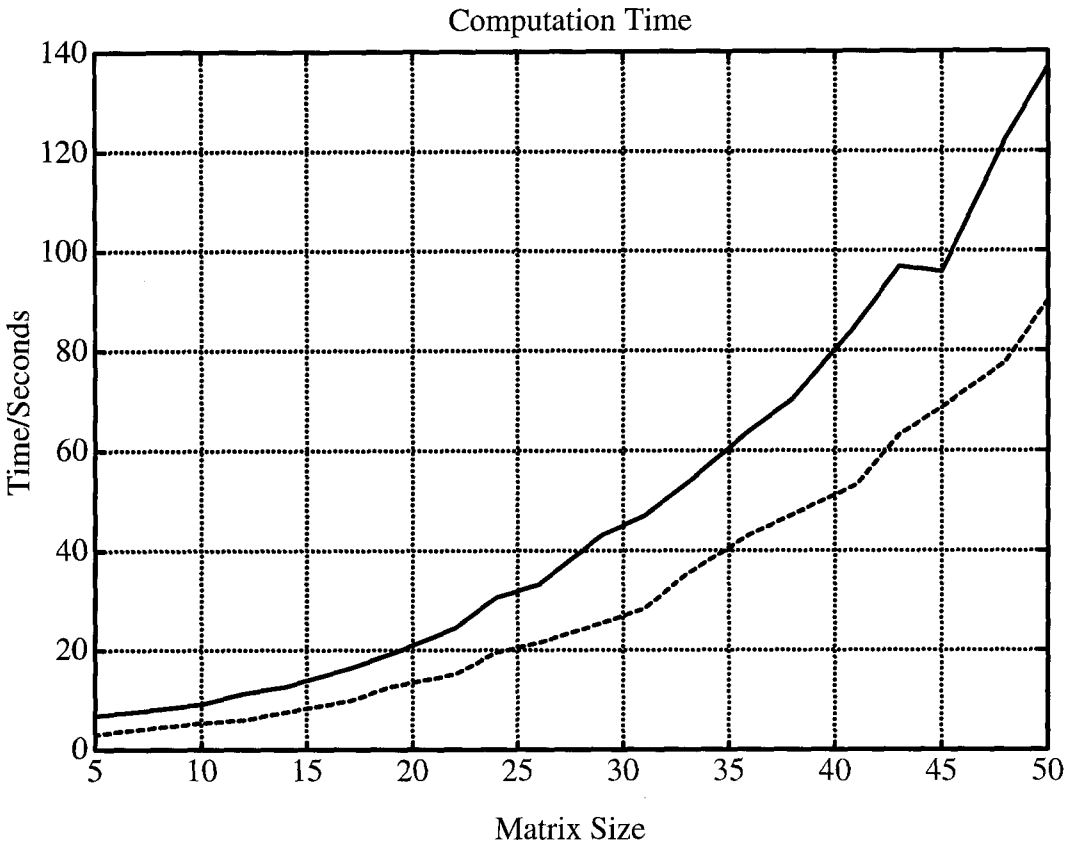


Figure 4.1: Typical computation time requirements versus matrix size for mixed μ problem (solid) and complex-μ problem (dashed).

point operations (flops). If this measure is adopted, as in Figure 4.2, then it is seen that the experimental growth rate in flops is approximately n^3. In any case the algorithm growth rate appears reasonable whether measured in terms of time or flops required, and in fact it is easy to show that for a fixed maximum number of iterations, which is enforced in the code whether or not the algorithm has converged, the computational cost is not more than of order n^3 flops.

The next set of tests evaluates the accuracy of the bounds. Again we used `crand` matrices, and the same class of block structures except with 80 percent of the uncertainties chosen to be real. This time we compared the upper and lower mixed μ bounds, and also the mixed μ and complex μ upper bounds. The complex μ bounds were obtained by simply replacing all the real perturbations with complex ones, but without changing the matrix. Thus the complex upper bound is strictly larger than the mixed upper bound. The results are shown in Figure 4.3, and indicate that for these problems we are obtaining fairly tight bounds, even for large problems.

It is also apparent that for these problems there is typically not much of a

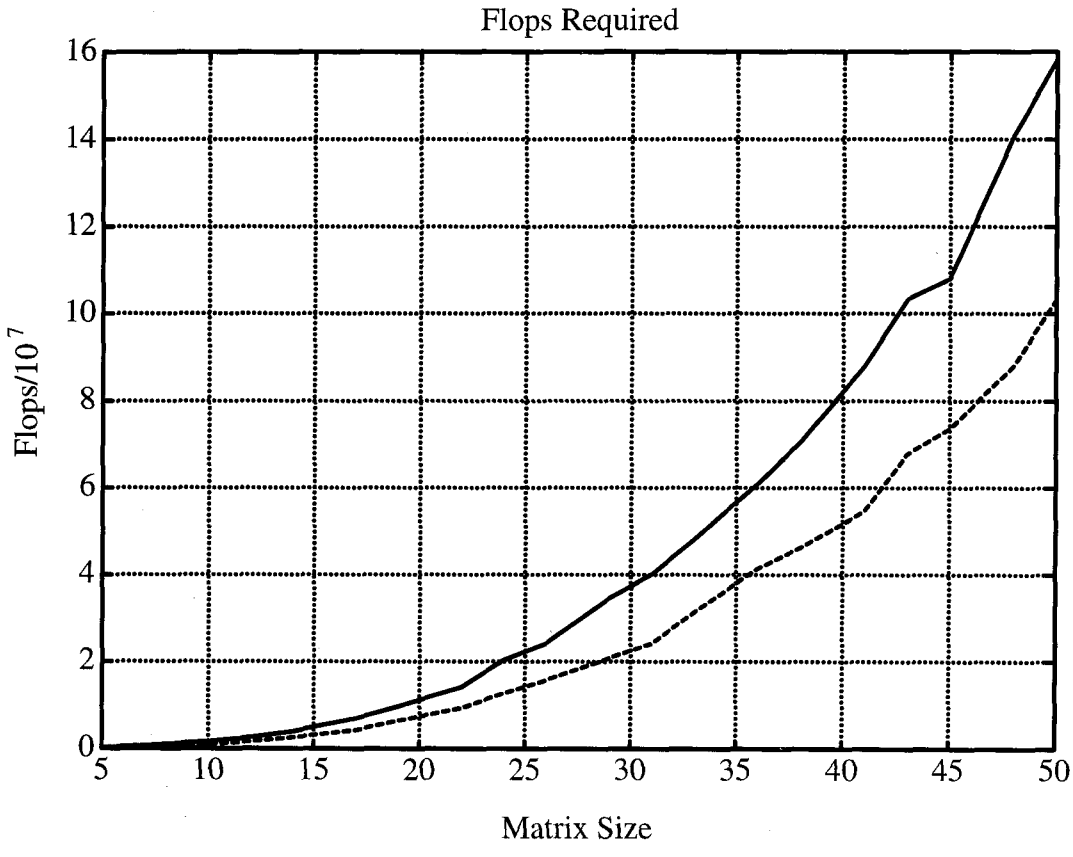


Figure 4.2: Typical computation requirements in flops versus matrix size for mixed μ problem (solid) and complex-μ problem (dashed).

gap between mixed μ and complex μ. This class of matrices is interesting from the point of view of the lower bound performance, since the lower bound with a mixed perturbation is close to the size of the lower bound with a complex perturbation. On the other hand, it is not too interesting from the point of view of the upper bound performance, since the G scaling matrix cannot greatly reduce the upper bound.

A similar set of tests was performed with `sysrand` matrices rather than `crand` matrices. Some results from these tests are shown in Figure 4.4. It can be seen that the bounds are usually reasonably tight, even for the largest ($n = 50$) problems, and the gap between the mixed and complex upper bounds is often reasonably large.

A number of tests used `nogap` matrices, and it was found that the upper bound computation was typically within 1 to 2 percent of the optimum for these matrices. The lower bound performance was not as good, and in fact the lower bound computation—the SPA—often fails to converge on this type of matrix, and provides a poor bound, motivating the work in Chapter 3.

The algorithm was also tested on a variety of other block structures and

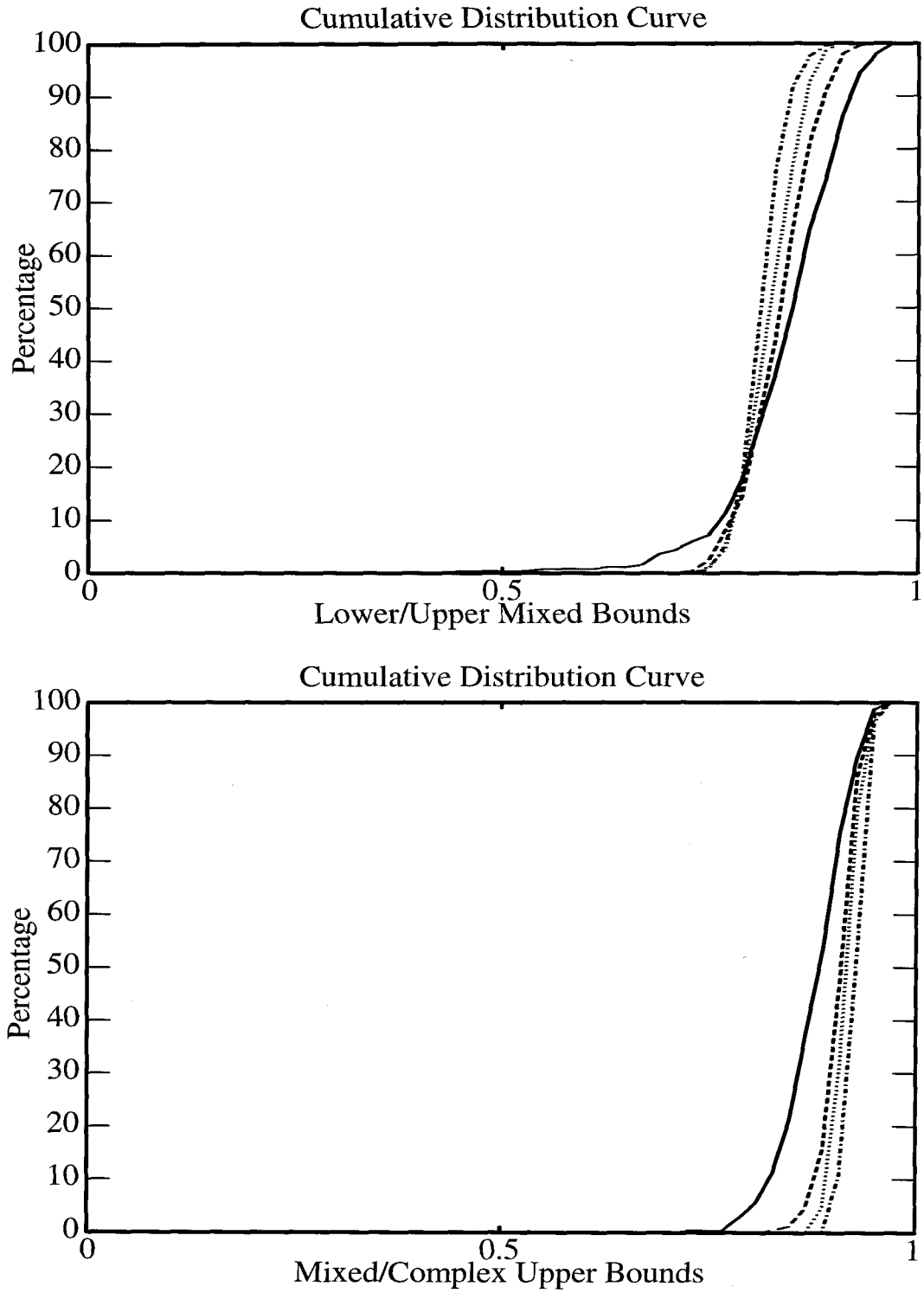


Figure 4.3: Ratios of mixed-μ lower to upper bounds, and mixed-μ to complex-μ upper bounds, for a sample of `crand` matrices. Matrices of sizes 10 (solid), 20 (dashed), 30 (dotted), and 50 (dashdot).

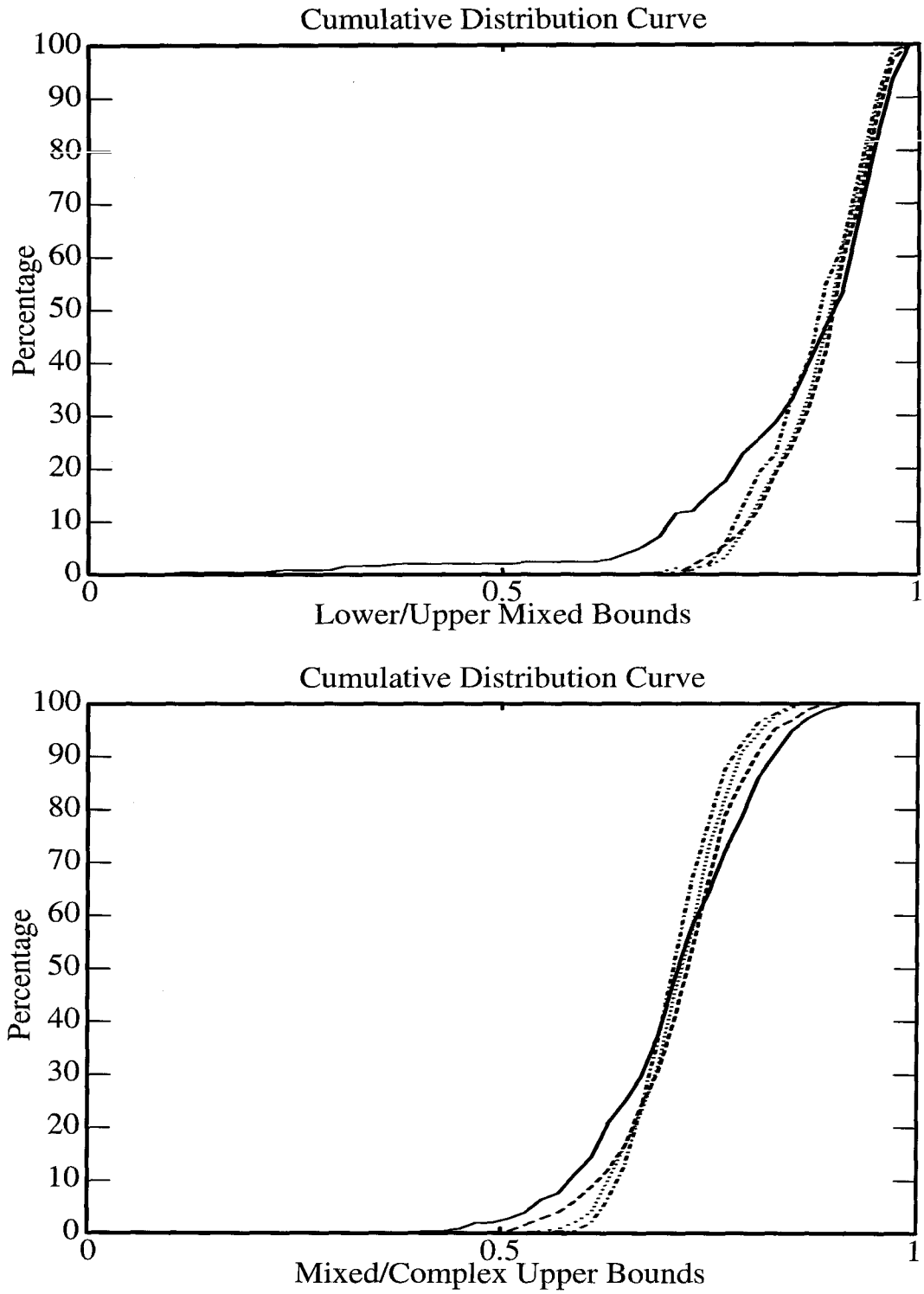


Figure 4.4: Typical ratios of mixed-μ lower to upper bounds, and mixed-μ to complex-μ upper bounds, for `sysrand` matrices of sizes 10 (solid), 20 (dashed), 30 (dotted), and 50 (dashdot).

the properties appear similar to those described above. An exception to this is the pure real case, which appears to have significantly poorer properties than any other.

4.5 Practical Examples

While the results from the previous section are encouraging, the real issue is the algorithm's performance on actual engineering examples. A number of interesting applications of the software to problems arising from real physical systems have already been undertaken. The control design of a missile autopilot is considered by Balas and Packard in [3]. The software is used to examine the robust performance of the control design subject to perturbations in Mach number (real), angle of attack (real), and unmodeled dynamics (complex). This results in a mixed μ problem with two repeated real scalar blocks and three full complex blocks. The robust performance μ plots for this problem, and the associated complex μ problem (simply "covering" the real uncertainties with complex ones), are shown in Figure 4.5. The mixed μ lower bound (the SPA) is performing poorly where the problem has degenerated to a nearly pure real problem.

It can be seen that the mixed μ bounds are quite different from the complex μ bounds. In particular the complex μ approximation to the problem indicates that the controller robustness properties are poor around 40 rad/s, whereas the mixed μ analysis indicates that there is no difficulty at this frequency. The performance predictions for different controllers are also different, and the closed-loop performance predictions from the mixed μ bounds are consistent with the simulations (see [3] for details).

Control of a flexible structure is considered by Balas et al. in [4], and the robustness of the design is evaluated with respect to variations in the natural frequencies of the structural modes (real perturbations), as well as unmodeled dynamics (complex perturbations). This results in a mixed μ problem with five real scalar blocks and three full complex blocks. The robust performance μ plots for this problem, and the associated complex μ problem are shown in Figure 4.6.

In this case the mixed and complex bounds are quite close (see [4] for a physical interpretation of these results). The control design predictions were verified in simulation and experiment. For these and several other examples the software worked well, providing tight bounds for the associated mixed μ problems.

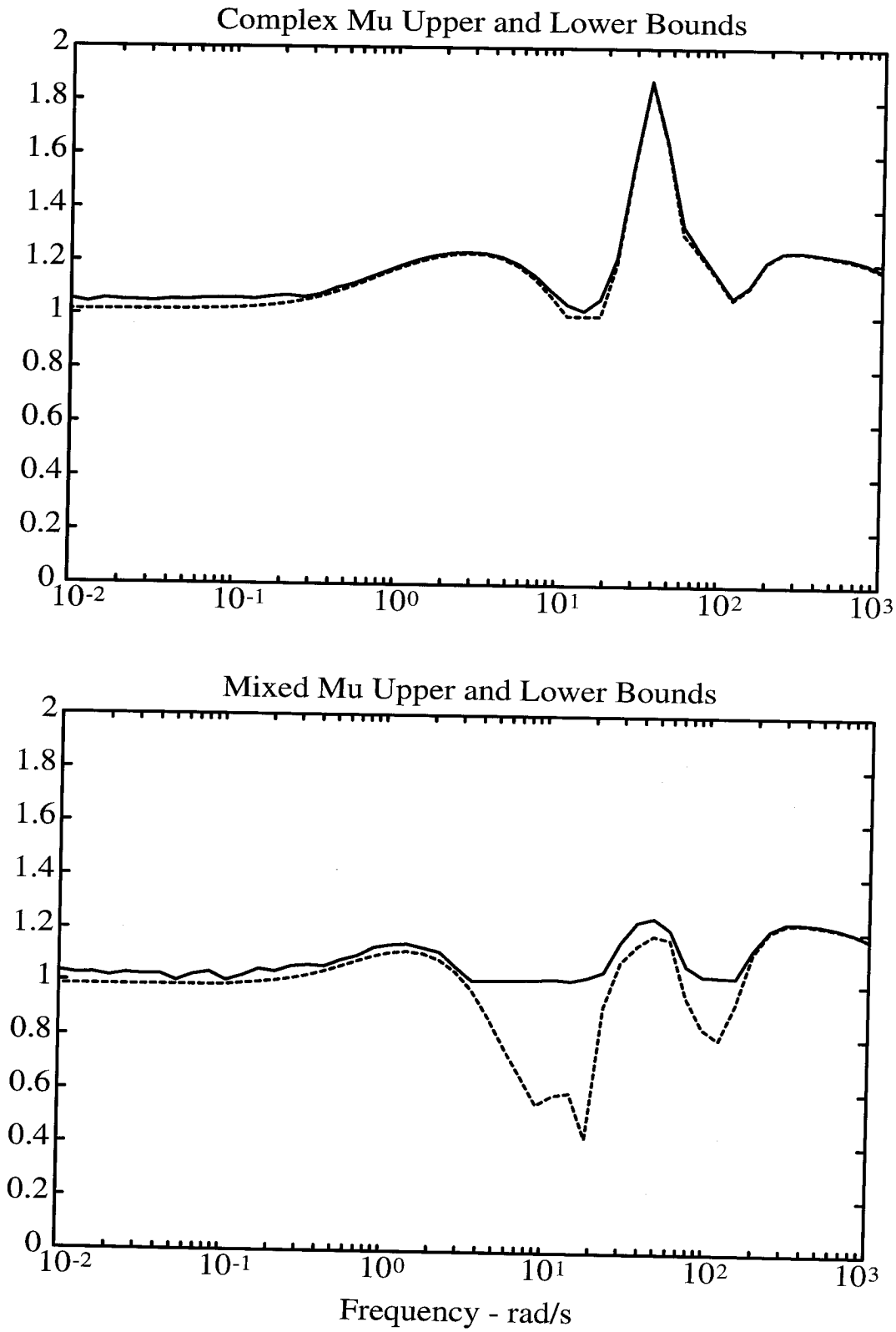


Figure 4.5: Robust performance μ plots for the missile autopilot problem.

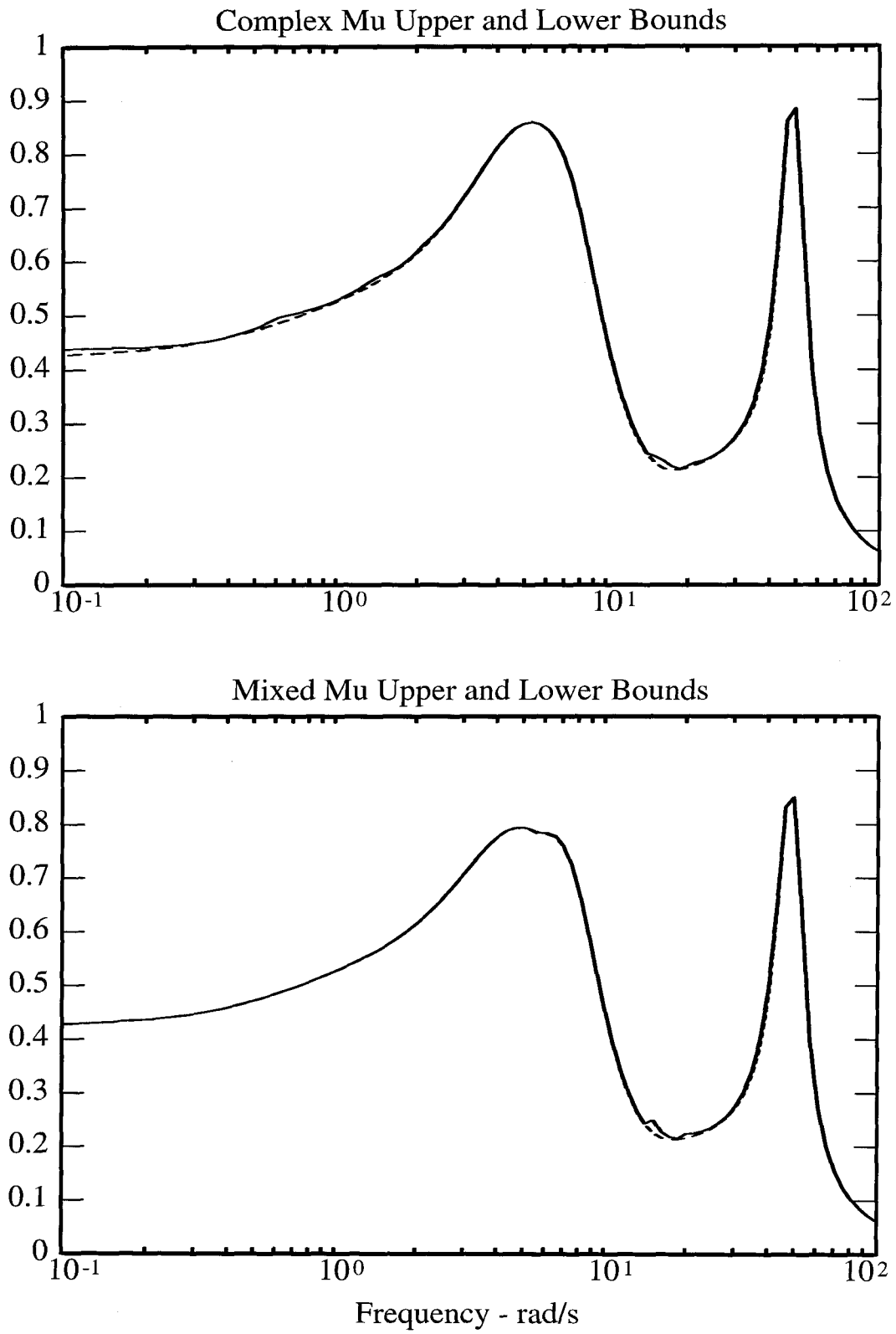


Figure 4.6: Robust performance μ plots for the flexible structure problem.

Concluding Remarks

In the previous sections we encountered problems, both randomly generated and practically motivated, where the values of mixed μ and complex μ could be far apart or close together. Since it is hard to know a priori which case one will encounter it is important to have good performance from both the upper and lower bound algorithms.

All the previous tests were aimed at evaluating the typical performance of the algorithm, and it appears that the algorithm is performing well for most problems. This does not mean however that one cannot encounter mixed μ problems where the gap between the upper and lower bounds is large, and it can be seen from Figure 4.4 that a few such cases were found. Furthermore it is possible to construct matrices for which the gap between mixed μ and the optimal upper bound from Theorem 4.4 is arbitrarily large, regardless of the computation method. For these cases one must consider improving the bounds themselves. A promising approach is to use the existing bounds as part of a branch and bound scheme which iteratively refines them.

Chapter 5

Branch and Bound

The previous two chapters addressed the issue of computing bounds for the mixed μ problem. The lower bound in Chapter 3 is an optimization problem whose maximum achieves μ. Unfortunately, we can find only local maxima efficiently. In contrast, the upper bound in Chapter 4 is a convex optimization problem, and can be computed efficiently and reliably. However, the optimum of the upper bound problem does not, in general, achieve μ, and there can be large gaps between the optimal bounds when the block structure of the mixed μ problem includes real blocks or repeated complex scalar blocks. This chapter shows that the careful application of branch and bound techniques results in substantially improved computation.

Branch and bound is a general technique that is applicable to optimization problems where upper and lower bounds which depend on the domain of optimization are computed. Branch and bound partitions the domain to get better bounds. The finer the domain of optimization is partitioned, the smaller the gap between the upper and lower bounds.

Branch and bound is inherently iterative and convergence proofs come easily. Nevertheless, it is important to remember the algorithm limitations imposed by the nature of the problem as discussed in Chapter 2; if we have an algorithm that is guaranteed to converge to μ, then we can expect exponential growth in computational cost with problem size, which is unacceptable. Thus, even with the application of new techniques such as branch and bound, we expect computation that is qualitatively similar to that of Chapters 3 and 4.

While other authors have proposed branch and bound algorithms for computing stability margins with respect to real perturbations (see [1, 11, 34] and the references therein), they have not specifically addressed the issue of avoiding obviously exponential growth in computational expense with problem size. Balakrishnan et al. ([1]) use inexpensive bounds that result in an exponential growth in the number of required branches, while DeGaston and Safonov ([11]) use bounds that are themselves exponentially expensive to compute.

In contrast, we look to branch and bound to see if we can get better bounds computation with reasonable cost. In particular, we want to see if branch

and bound helps on problems where the previously described computation is inadequate—problems with excessive gaps between the upper and lower bounds. We consider the application of branch and bound techniques with the goal of improving approximate solutions quickly even for problems with many dozens of parameters; the work that has gone before in the control community has not.

Branch and bound provides strong motivation for improved bounds computation, since mildly better bounds computation branch and bound is much more effective. Better bounds computation will not obviate the need for branch and bound: in any lower bound computation, we need an upper bound that is close to ensure its quality; in any upper bound computation, there may be a large gap between the infimum in Equation (4.4) and μ.

In order to efficiently apply branch and bound techniques to the mixed μ problem, we develop an extension to μ in Section 5.2. This new function allows us to evaluate μ on subsets of $\mathbf{B}_{\mathbf{\Delta}}$ constructed by partitioning the real interval corresponding to a real perturbation. It is easy and important to do this partitioning along repeated complex variables also, but for clarity we do not do so here. (Any connected strict subset of $\partial\mathbf{B}_{\mathbf{\Delta}}$ can be invertibly mapped to $[-1, 1]$ with a simple LFT.)

In Section 5.5 we investigate the performance of a variety of branch and bound algorithms on mixed μ problems. The tests show that, when used with the best bounds available, branch and bound is quite effective at achieving good bounds for most difficult problems. However, it is also clear that if the convergence tolerance is set too tight then the difficult problems require an excessive number of iterations. This suggests that branch and bound is useful only when it does not need to be used very much or very often. Thus the branch and bound scheme should not be used as a general computation scheme per se, but only to as a way to fix the occasional problems for which the bounds are poor, and for these problems to achieve the degree of accuracy that the bounds typically get.

5.1 Branch and Bound Algorithms

Branch and bound algorithms may be considered to be an advanced form of gridding the parameter space, so we first consider gridding. We need a few definitions.

Definition 5.1 *Let Δ be a compact subset of a metric space. A* finite δ cover *of Δ is a finite collection of sets $\{\Delta_i\}$ such that each Δ_i is contained in a δ ball and $\Delta \subset \bigcup_i \Delta_i$.*

Remarks: We refer to a finite δ cover of Δ as a gridding of Δ.

Throughout this chapter the subscripts on the Δ do not denote the components of a product space; Δ_i is a subset of the same metric space that contains Δ.

Definition 5.2 *Let Δ be a compact subset of a metric space and $UB(\Delta)$ and $LB(\Delta)$ be upper and lower bounds to the real valued function $f(\Delta)$. The bounds UB and LB are* continuous *on Δ if, given any point $\Delta_0 \in \Delta$ and any $\epsilon > 0$, there is a δ such that any Δ_1 that contains Δ_0 and is in a δ ball satisfies $UB(\Delta_1) - LB(\Delta_1) < \epsilon$.*

The following theorem follows easily from the definitions and the assumption that Δ is compact.

Theorem 5.3 *Let $M \in \mathbb{C}^{n\times n}$ be a matrix, Δ be a compact subset of a metric space, and $UB(M, \Delta)$ and $LB(M, \Delta)$ be continuous upper and lower bounds of $f(M, \Delta) = \max_{Q\in\Delta} g(M, Q)$, an optimization problem. Then $\forall \epsilon > 0$ there is a δ such that any δ cover $\{\Delta_i\}$ of Δ satisfies $\max_i UB(M, \Delta_i) - \max_i LB(M, \Delta_i) < \epsilon$.*

Remarks: Since $\max_i UB(M, \Delta_i) \geq f(M, \Delta) \geq \max_i LB(M, \Delta_i)$, gridding "solves" the optimization problem. The mixed μ problem is a special case of the problems addressed by the theorem. An easy way to find a δ cover of Δ is to repeatedly bisect each of the parameters independently. This results in an exponential growth of the number of sets in the δ cover with the number of parameters in Δ, consistent with the analysis of Chapter 2.

Branch and bound can be thought of as a smarter way of gridding the parameter space. As a branch and bound algorithm proceeds, the parameter space Δ is partitioned into more and more subsets. At each iteration, the collection of subsets forms a cover of Δ and the bounds at that step are $\max_i UB(M, \Delta_i) \geq f(M, \Delta) \geq \max_i LB(M, \Delta_i)$. The partitioning, however, is often substantially more efficient than gridding. We start with the branch and bound algorithm from [1]. Here the parameter space is the subset of $\mathbb{R}^n$ where each element is constrained to lie in a finite interval.

Initialize $\{\Delta_i\} = \Delta_1$
Let $UB = \max_i UB(M, \Delta_i)$
$LB = \max_i LB(M, \Delta_i)$
while $UB - LB > \epsilon$
Let Δ be any element of $\{\Delta_i\}$ with $UB(M, \Delta) = UB$.
Partition Δ into Δ_a and Δ_b by bisecting Δ along one of its longest edges.
Add Δ_a and Δ_b to $\{\Delta_i\}$.
Remove Δ from $\{\Delta_i\}$.
endwhile

Theorem 5.4 *This branch and bound algorithm converges. (Eventually $UB - LB \leq \epsilon$.)*

Proof: By contradiction suppose the algorithm does not converge. Then for all i such that $UB(M, \Delta_i) = UB$, $UB - LB > \epsilon$ implies $UB(M, \Delta_i) - LB(M, \Delta_i) > \epsilon$, which in turn implies Δ_i is not in a δ ball. So the longest edge of each of the Δ_i is too long and therefore must not be getting bisected. This is a contradiction. ■

Remarks: Any element of $\{\Delta_i\}$ for which $UB(M, \Delta_i) < LB$ will never again be partitioned and need not be considered further. In a computer implementation of this algorithm, all such elements can be deleted from memory. This is called pruning. Also, it is clear that this algorithm can exhibit exponential growth in the number of branching parameters.

We have two primary reasons for looking at more general branch and bound algorithms. The first is that we would like to apply branch and bound techniques to a wider class of problems, using bounds that do not meet the conditions necessary to guarantee convergence. These problems correspond to μ problems where we choose the branching space to be a subset of the optimization space $\mathbf{B}_{\mathbf{\Delta}}$. In this case, we could guarantee convergence to within ϵ of the worst-case gap (this is stated more precisely in the theorem below). For the mixed μ problem, this alternative is particularly attractive: it is important to include repeated scalar blocks in the branching space because the gaps between the bounds can be too large otherwise, and these blocks require only one branching parameter each; it is important not to include full blocks in the branching space since the number of required branching parameters is large, and the gaps between our bounds are reasonable when we do not branch on these parameters.

The other motivation for more general branch and bound schemes is the desire for more freedom to choose how to partition Δ, the domain of optimization. This is motivated by the fact that we can construct problems that exhibit any of the following characteristics.

- bisecting one particular variable will bring the bounds within tolerance while bisecting any other will not
- a single off-center cut will bring the bounds within tolerance, while many bisections would be necessary to bring the bounds within tolerance
- bisecting a single variable along a short side will bring the bounds within tolerance, while bisection along any of the longer sides will not

Potential remedies to these difficulties include

- careful selection of which parameter to cut
- allowing off-center cuts
- allowing a cut of a shorter side

Consequently we allow a larger class of bounds in our general branch and bound algorithms.

Definition 5.5 *Let Δ be a compact subset of a metric space and $UB(\Delta)$ and $LB(\Delta)$ be upper and lower bounds to the real valued function $f(\Delta)$. The bounds UB and LB are* γ continuous *in Δ if, given any point $\Delta_0 \in \Delta$ and any $\epsilon > 0$, there is a δ such that any Δ_1 that contains Δ_0 and is in a δ ball satisfies $UB(\Delta_1) - LB(\Delta_1) < \gamma + \epsilon$.*

Now consider the effect of gridding the parameter space with γ continuous bounds. Given any $\epsilon > 0$, there is a δ such that any δ cover of Δ_1 satisfies $\max_i UB(M, \Delta_i) - \max_i LB(M, \Delta_i) < \gamma + \epsilon$ so gridding "solves" the mixed μ problem to within $\gamma + \epsilon$.

In order to allow a wide class of partitioning schemes we need to define a notion of cutting convergence.

Definition 5.6 *A partitioning scheme is* convergent *if, given $\delta > 0$, there is an N such that every subdomain resulting from at least N partitioning steps is contained in a δ ball.*

We now define a general class of branch and bound algorithms:

Initialize	$\{\Delta_i\} = \Delta_1$
Choose	Our partitioning scheme to be any convergent partitioning scheme
Let	$UB = \max_i UB(M, \Delta_i)$
	$LB = \max_i LB(M, \Delta_i)$
while	$UB - LB > \gamma + \epsilon$
	choose Δ to be any element of $\{\Delta_i\}$ with $UB(M, \Delta) = UB$.
	Partition Δ into Δ_a and Δ_b according to our partitioning scheme.
	Add Δ_a and Δ_b to $\{\Delta_i\}$.
	Remove Δ from $\{\Delta_i\}$.
endwhile	

Theorem 5.7 *This branch and bound algorithm converges to within γ. (Eventually $UB - LB < \gamma + \epsilon$.)*

Since the bounds for μ on a box in Section 5.2 fit this framework, we can guarantee convergence for a mixed μ branch and bound scheme. However, as we are looking for an algorithm that efficiently finds approximate solutions to most of the problems we are interested in, our convergence guarantees are of little help because they are unacceptably slow. Furthermore, any convergence guarantees for the general problem will also be unacceptably slow. In fact, if any single problem requires branch and bound, then we can construct a sequence of problems with exponential growth in the number of iterations required with problem size.

The real issue is whether or not we can produce a practical scheme whose typical computation time is polynomial despite the fact that the worst-case computation time is exponential. An important feature of branch and bound algorithms is that we always have the bounds UB and LB; we do not have to wait until the algorithm converges in order to get bounds. Consequently we can investigate the performance of branch and bound algorithms in the context of efficient computation with nonexponential growth despite the exponential nature of convergent branch and bound algorithms.

Once we give up our convergence guarantees and strive for improved bounds with moderate computational expense, we may alter our algorithm to branch only on the parameters that are likely to result in significant improvement. Thus we branch only on real blocks and repeated complex scalar blocks, but not on full blocks. In order to implement such a branch and bound scheme we need to extend the bounds for mixed μ on $\mathbf{B}_{\mathbf{\Delta}}$ to be able to compute bounds for $\mu(M)$ on Δ_i.

5.2 μ on a Box

It is not immediately apparent from the definition of μ in Definition 1.1 that branch and bound can even be applied to this problem. We need the reformulation of the problem in Lemma 3.1

$$\mu(M) = \max_{\Delta \in \mathbf{B}_\Delta} \rho_r(\Delta M)$$

so that there is an obvious parameter space that can be partitioned in a branch and bound scheme: the maximization is of a function defined on a matrix M over a compact set $\mathbf{B}_\Delta$ in a metric space.

Clearly if we have sets $\mathbf{B}_{\Delta_i} \subset \mathbf{B}_\Delta$ with $\bigcup \mathbf{B}_{\Delta_i} = \mathbf{B}_\Delta$ then

$$\mu(M) = \max_{\Delta \in \bigcup \mathbf{B}_{\Delta_i}} \rho_r(\Delta M) = \max_i \max_{\Delta \in \mathbf{B}_{\Delta_i}} \rho_r(\Delta M).$$

We define μ on a box to be

$$\mu_i(M) \triangleq \max_{\Delta \in \mathbf{B}_{\Delta_i}} \rho_r(\Delta M). \tag{5.1}$$

Then it follows that

$$\mu(M) = \max_i \mu_i(M). \tag{5.2}$$

Denote upper and lower bounds for μ_i as ub_i and lb_i respectively, so that $lb_i \leq \mu_i \leq ub_i$, and define $L \triangleq \max_i lb_i$ and $U \triangleq \max_i ub_i$. Then

$$\max_i lb_i \leq \max_i \mu_i \leq \max_i ub_i$$

so that

$$L \leq \mu \leq U.$$

Thus L and U are upper and lower bounds for $\mu(M)$ that depend on the local bounds lb_i and ub_i, and on the partitioning $\bigcup \mathbf{B}_{\Delta_i} = \mathbf{B}_\Delta$.

Recalling the definition of $\mathbf{B}_\Delta$, we see that Equation (5.1) is useful for branch and bound algorithms which branch on the real parameters only. Note that it is easy, using Theorem 5.8 and simple manipulations, to transform the problem so that we can conveniently branch on repeated complex scalar blocks. For the remainder of this chapter we consider only $\mathbf{B}_{\Delta_i}$ that result from partitioning the intervals that contain the real variables in $\delta_r I$ blocks of $\mathbf{B}_\Delta$.

This definition is obviously quite similar to the original definition for μ on the unit box $\mathbf{B}_\Delta$. The following sections show that the computation of bounds for μ on a box is quite similar to the computation of bounds for μ on the unit box, as developed in Chapters 3 and 4.

5.3 The Lower Bound

In this section we show how the lower bound computation for μ can be generalized to calculate a lower bound to μ on a box. As in the standard case, the maximum occurs on the boundary:

Theorem 5.8 *For any matrix $M \in \mathbb{C}^{n\times n}$, any compatible block structure $\partial\mathbf{B}_{\mathbf{\Delta}}$, and any $\mathbf{B}_{\mathbf{\Delta}_i}$ that results from partitioning the real variables in $\delta_r I$ blocks of $\mathbf{B}_{\mathbf{\Delta}}$,*

$$\max_{Q\in\partial\mathbf{B}_{\mathbf{\Delta}_i}} \rho_r(QM) = \mu_i(M).$$

Remarks: This is a trivial extension of Theorem 3.2.

To explain the generalization of the lower bound computation, we must explain a bit more about the SPA than we did in Chapter 3. The SPA is based on characterizations of local maxima of $\rho_r(QM)$. We parametrize all allowable perturbations to Q as $E(t) = (I + Gt)(I - Gt)^{-1}$ for $t \geq 0$ (and small enough), where $G \in \mathbb{G}$ and $\mathbb{G}$ is the set of all allowed perturbation directions. As before, there are some non-degeneracy assumptions. With the above perturbation β, x and w are functions of t satisfying

$$\begin{aligned} \beta x &= E(t)QMx \\ \beta w^* &= w^*E(t)QM. \end{aligned}$$

If β is at a local maximum at $t = 0$ for all $G \in \mathbb{G}$ then

$$\mathrm{Re}(w^*Gx) \leq 0 \quad \forall G \in \mathbb{G} \quad \text{at } t = 0. \tag{5.3}$$

Defining $z = Q^*x$ and partitioning z and w to conform to the blocks of $\mathbf{\Delta}$, we get the following conditions equivalent to Equation (5.3).

$$\begin{array}{rcl} w = dQz \\ d \geq 0 \end{array} \Longrightarrow \begin{array}{rcl} Q &=& \dfrac{wz^*}{\|w\|\,\|z\|} \\ d &=& \dfrac{\|w\|}{\|z\|} \end{array}$$

$$w^*qz \geq 0 \implies q = \frac{z^*w}{\|z^*w\|}$$

$$\begin{array}{rcll} \mathrm{Re}(w^*qz) \geq 0 \\ (0 \text{ for internal } q) \end{array} \Longrightarrow \begin{array}{rcll} q &=& \mathrm{sgn}(\mathrm{Re}(w^*z)) & (\text{if } \mathrm{Re}(w^*z) \neq 0) \\ q &\in& [-1,1] & (\text{if } \mathrm{Re}(w^*z) = 0) \end{array}$$

for full blocks, repeated complex scalar blocks, and real blocks respectively. For the full blocks, Q can be different than the above so long as its action on z and w is the same as the Q above. In fact the full blocks of Q must be

different for Q to be in $\partial \mathbf{B}_{\mathbf{\Delta}}$—all the singular values of the full blocks must be 1.

For μ on a box, the first two equivalent conditions stay the same and the last becomes

$$\begin{aligned} \mathrm{Re}(w^*z) > 0 &\implies q \text{ is at its upper limit} \\ \mathrm{Re}(w^*z) < 0 &\implies q \text{ is at its lower limit} \\ \mathrm{Re}(w^*z) = 0 &\implies q \text{ could be internal.} \end{aligned}$$

The SPA is easily modified to agree with the new version of these equations. This modified power algorithm seems to work quite well. Similarly, the more advanced lower bound algorithms of Chapter 3 can be generalized to calculate a lower bound to μ on a box, though we will not go into that here.

5.4 The Upper Bound

In this section we develop an upper bound to μ on a box. This bound contains the conventional bound as a special case, and is suitable for computation in the same way the old bound is. Current upper bound algorithms have been generalized to calculate an upper bound to μ on a box.

We first need to define the matrix S_i. Let $r > 0$ and c be the diagonal matrices that satisfy $\mathbf{B}_{\mathbf{\Delta}_i} = c + r\mathbf{B}_{\mathbf{\Delta}}$, and let

$$S_i = \begin{bmatrix} 0 & \sqrt{r} \\ \sqrt{r} & c \end{bmatrix}$$

so that $\mathbf{B}_{\mathbf{\Delta}_i} = \mathbf{B}_{\mathbf{\Delta}} \star S_i$. In this capacity, S_i maps the real interval $[-1, 1]$ to the real interval $[c - r, c + r]$.

Theorem 5.9 *For any matrix $M \in \mathbb{C}^{n\times n}$, any compatible block structure, and any $\mathbf{B}_{\mathbf{\Delta}_i}$, suppose that $\det(I - c\frac{M}{\alpha}) \neq 0$. Let $\widehat{M} = (S_i \star \frac{M}{\alpha})$. Then*

$$\mu_i(M) \leq \inf_{\substack{D\in\mathbf{D} \\ G\in\mathbf{G}}} \left[\min_{\alpha\in\overline{\mathbb{R}^+}} \left\{ \alpha \;\middle|\; (\widehat{M}^*D\widehat{M} + j(G\widehat{M} - \widehat{M}^*G) - D) \leq 0 \right\} \right]. \tag{5.4}$$

Note that if $\det(I - c\frac{M}{\alpha}) = 0$ then α is a lower bound to μ_i.

In order to prove this theorem, we first present some convenient machinery.

Lemma 5.10 *Let ΔL_{11} and ML_{22} be square. If $\det(I - \Delta L_{11}) \neq 0$ and $\det(I - ML_{22}) \neq 0$, define*

$$\begin{aligned} \Delta \star L &= L_{22} + L_{21}(I - \Delta L_{11})^{-1}\Delta L_{12} \\ L \star M &= L_{11} + L_{12}(I - ML_{22})^{-1}ML_{21}. \end{aligned}$$

Then (i)

$$\begin{aligned}\det\left(I-\begin{bmatrix}\Delta & \\ & M\end{bmatrix}L\right)& \\ &= \det(I-\Delta L_{11})\det(I-(\Delta\star L)M)\\ &= \det(I-ML_{22})\det(I-\Delta(L\star M))\end{aligned}$$

and (ii)

$$\begin{aligned}\det\left(I-\begin{bmatrix}\Delta & \\ & M\end{bmatrix}L\right)=0& \\ \iff & \det(I-(\Delta\star L)M)=0\\ \iff & \det(I-\Delta(L\star M))=0.\end{aligned}$$

Proof: (i)—recalling that $\det(I-AB)=\det(I-BA)$, we see that (i) is merely Schur's complement for

$$\det\begin{bmatrix}I-\Delta L_{11} & -\Delta L_{12}\\ -ML_{21} & I-ML_{22}\end{bmatrix}.$$

(ii)—follows trivially from (i). ■

We also need to define the matrix L: Let $\mathbf{B}_{\mathbf{\Delta}_1}=\{\,\Delta\mid\overline{\sigma}(\Delta)<1\,\}$ and, for any $\widehat{G}\in\widehat{\mathbf{G}}$, define

$$L=\begin{bmatrix}j\widehat{G}(I+\widehat{G}^2)^{-\frac{1}{2}} & (I+\widehat{G}^2)^{-\frac{1}{4}}\\ (I+\widehat{G}^2)^{-\frac{1}{4}} & 0\end{bmatrix}$$

so that $\mathbf{B}_{\mathbf{\Delta}}\subset\mathbf{B}_{\mathbf{\Delta}_{\mathbb{C}}}\star L\subset\mathbf{B}_{\mathbf{\Delta}_1}\star L$, where $\mathbf{B}_{\mathbf{\Delta}_{\mathbb{C}}}$ is the unit ball for the block structure with the repeated real scalar blocks replaced with repeated complex scalar blocks. Each real interval $[-1,1]$ in $\mathbf{B}_{\mathbf{\Delta}}$ is contained in an offset disk in the complex plane, and the real scalar g_i (a component of $\widehat{G}$) determines the center and radius of the complex disk.

The following theorem is a simple version of Lemma 4.5, and is stated here for convenience.

Theorem 5.11 *Suppose we have a matrix* $M\in\mathbb{C}^{n\times n}$ *and a real scalar* $\alpha>0$*, then there exist matrices* $D\in\mathbf{D}$, $G\in\mathbf{G}$ *such that*

$$\overline{\lambda}\left(M^*DM+j(GM-M^*G)-\alpha^2D\right)\le 0 \tag{5.5}$$

if and only if there exist matrices $\widehat{D}\in\widehat{\mathbf{D}}$, $\widehat{G}\in\widehat{\mathbf{G}}$ *such that*

$$\overline{\sigma}\left((I+\widehat{G}^2)^{-\frac{1}{4}}\left(\frac{\widehat{D}M\widehat{D}^{-1}}{\alpha}-j\widehat{G}\right)(I+\widehat{G}^2)^{-\frac{1}{4}}\right)\le 1. \tag{5.6}$$

We are now ready to prove the upper bound theorem. The proof uses Theorem 5.11. Theorems 4.3 and 4.4 are special cases of the following theorem. We also can easily see where the conservativeness of these theorems arises.

Proof (Theorem 5.9): Recall the definition for μ on a box:

$$\mu_i(M) = \max_{\Delta \in \mathbf{B}_{\mathbf{\Delta}_i}} \rho_r(\Delta M).$$

$$\begin{aligned}
\widehat{\alpha} > \mu_i \quad &\Longleftrightarrow \quad 0 \notin \det(\alpha - \mathbf{B}_{\mathbf{\Delta}_i} M) && \forall \alpha \in [\widehat{\alpha}, \infty) \\
&\Longleftrightarrow \quad 0 \notin \det(I - \mathbf{B}_{\mathbf{\Delta}_i} \tfrac{M}{\alpha}) && \forall \alpha \in [\widehat{\alpha}, \infty) \\
&\Longleftrightarrow \quad 0 \notin \det(I - (\mathbf{B}_{\mathbf{\Delta}} \star S_i) \tfrac{M}{\alpha}) && \forall \alpha \in [\widehat{\alpha}, \infty) \\
&\Longleftrightarrow \quad 0 \notin \det(I - \mathbf{B}_{\mathbf{\Delta}} (S_i \star \tfrac{M}{\alpha})) && \forall \alpha \in [\widehat{\alpha}, \infty) \\
&\Longleftarrow \quad 0 \notin \det(I - (\mathbf{B}_{\mathbf{\Delta}_1} \star L)(S_i \star \tfrac{M}{\alpha})) && \forall \alpha \in [\widehat{\alpha}, \infty) \\
&\Longleftrightarrow \quad 0 \notin \det(I - \mathbf{B}_{\mathbf{\Delta}_1} (L \star (S_i \star \tfrac{M}{\alpha}))) && \forall \alpha \in [\widehat{\alpha}, \infty) \\
&\Longleftrightarrow \quad \overline{\sigma}(L \star (S_i \star \tfrac{M}{\alpha})) \leq 1 && \forall \alpha \in [\widehat{\alpha}, \infty).
\end{aligned}$$

(It is easy to check that the conditions of Lemma 5.10 hold.) We can replace M by $\widehat{D} M \widehat{D}^{-1}$ because $\mu_i(M) = \mu_i(\widehat{D} M \widehat{D}^{-1}) \ \ \forall \widehat{D} \in \widehat{\mathbf{D}}$. Note also that $\widehat{D}(S_i \star \frac{M}{\alpha})\widehat{D}^{-1} = S_i \star (\widehat{D} \frac{M}{\alpha} \widehat{D}^{-1})$. Thus the last inequality can be expanded as

$$\overline{\sigma}\left((I + \widehat{G}^2)^{-\frac{1}{4}} \left(\widehat{D}\sqrt{r} \frac{M}{\alpha} (I - c\frac{M}{\alpha})^{-1} \sqrt{r} \widehat{D}^{-1} - j\widehat{G} \right) (I + \widehat{G}^2)^{-\frac{1}{4}} \right) \leq 1. \tag{5.7}$$

Now apply Theorem 5.11. ∎

The condition in Equation (5.6) is used in computation in Chapter 4 as follows: choose $\widehat{G}$ and $\widehat{D}$ to reduce $\overline{\sigma}$, search for an α (smaller is better) that satisfies Equation (5.6) and repeat. The condition Equation (5.7) can be used in exactly the same way.

The condition in Equation (5.5) is used similarly, except that α is solved for explicitly—with everything else held constant, finding the smallest α satisfying the inequality in Equation (5.5) is a generalized eigenvalue problem. This also can be done for Equation (5.4) with the help of the following theorem.

Theorem 5.12 *The smallest α satisfying the inequality in Equation (5.4) is given by*

$$\alpha = \overline{\lambda_r} \begin{bmatrix} cM^* - j\sqrt{r}GD^{-1}\sqrt{r}M^* & \sqrt{r}D\sqrt{r}M + \sqrt{r}GD^{-1}G\sqrt{r}M \\ \sqrt{r}D^{-1}\sqrt{r}M^* & cM + j\sqrt{r}D^{-1}G\sqrt{r}M \end{bmatrix}.$$

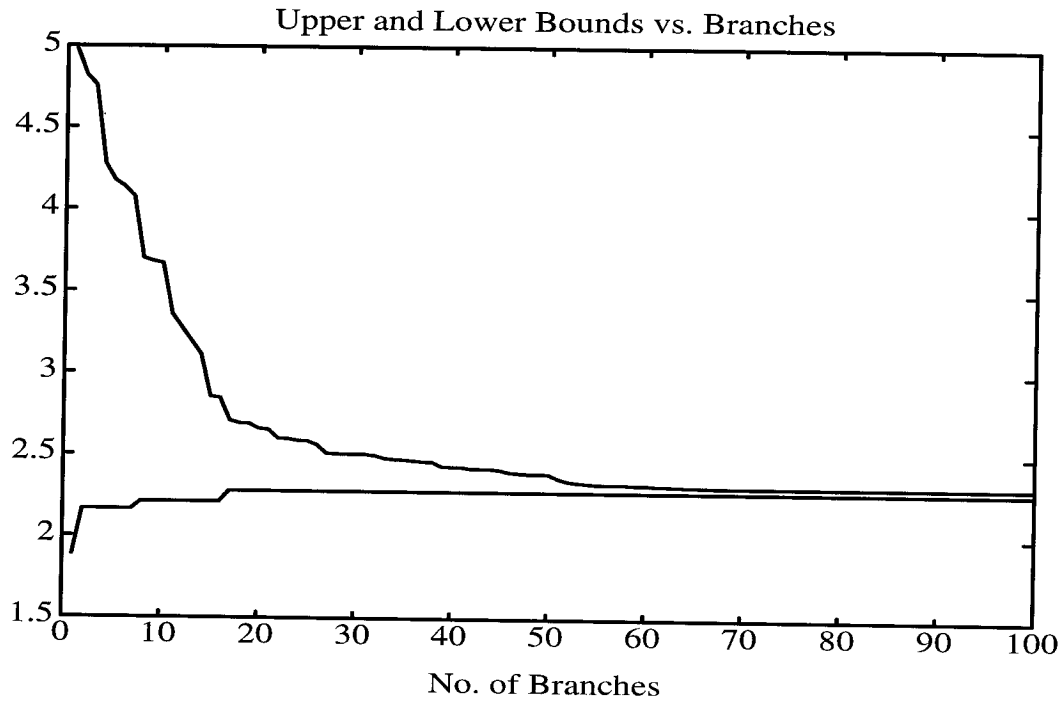


Figure 5.1: Progress on the most difficult of 10,000 problems with 4 scalar reals and 1 scalar complex.

5.5 Computation

In this section we investigate the performance of branch and bound algorithms on mixed μ problems. Since we would like the problems to be representative of control problems, we use `sysrand` matrices.

A preliminary investigation of the effect off-center cuts and careful choice of cutting parameter (not always among those with the longest sides) for these random problems showed little indication that these choices were worth the added expense and complication. Consequently, the tests and examples presented here are for algorithms that bisect along a longest side. Nevertheless it is quite possible that there are other classes of problems with engineering motivation where more careful choices are important.

First we show the progress of our branch and bound algorithm on four specific problems. The problem in Figure 5.1 is the result of a search for a particularly difficult problem from a set of 10,000 problems with a block structure of 4 real scalars and 1 complex scalar. (We did not run branch and bound on all these problems, instead we ran branch and bound on the 100 with the worst relative gaps before any branch and bound computation.) The branch and bound algorithm we used on this problem utilizes the bounds computation code by Beck and Doyle in [6] and by Tierno and Young in [37], modified for μ on a box.

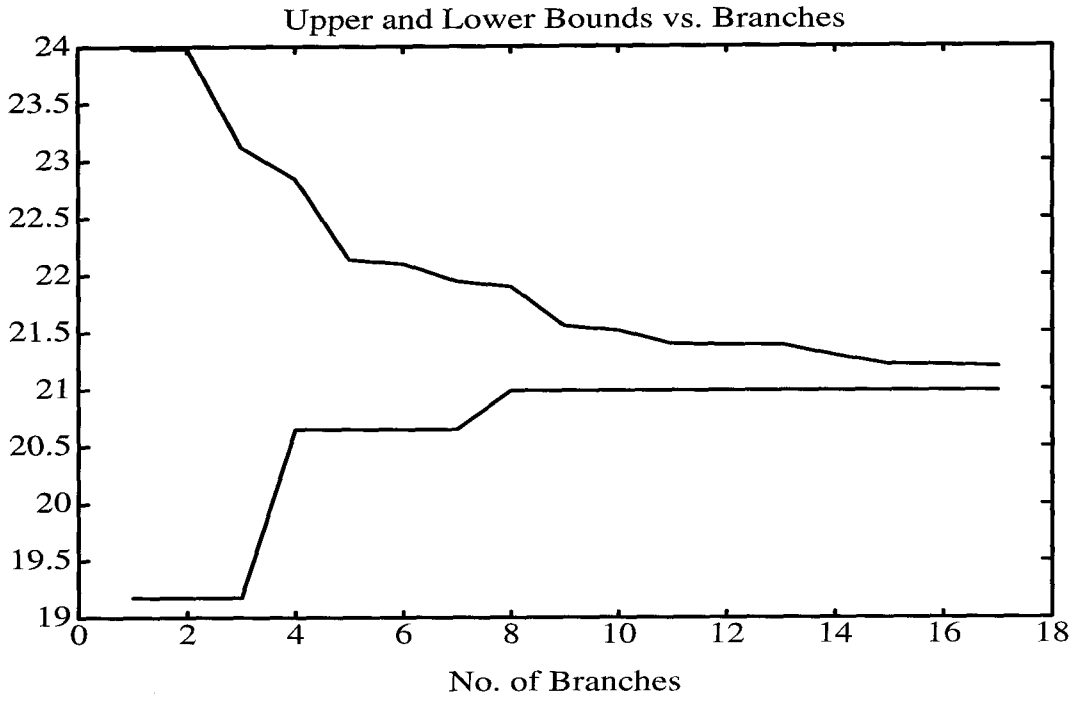


Figure 5.2: Progress with 1 size 12 repeated real, 4 scalar reals, and 1 complex full block of size 3.

Figure 5.2 shows a typical problem with a block structure of 1 size 12 repeated real block, 4 scalar real blocks, and 1 size 3 complex full block. The repeated real block seems to make the μ computation harder, but it only adds one branching parameter. It seems that branch and bound is particularly useful in these cases.

In Figure 5.3 we see the progress on a problem with a block structure of 4 scalar reals and 4 scalar complex blocks. Here the bounds are only γ continuous, and the algorithm converges quickly to the theoretical gap.

Finally Figure 5.4 shows the progress on a problem with 20 parameters, where the progress is slower than on the smaller problems. It is not clear how much of the gap is due to slow progress and how much is due to bounds that are only γ continuous.

Next we consider the effect of the quality of the bounds. Clearly we should get faster convergence with better bounds, but there remains the question of whether the increased computational effort the better bounds entail is worth the benefits. Figure 5.5 compares two branch and bound schemes on a problem closely related to the mixed μ problem. The labeling is best explained by example. The label "B50" means that this is the most difficult problem encountered by algorithm B, from among the easiest (for algorithm B) 50 percent of the problems, that is, 50 percent of the problems are easier than this problem.

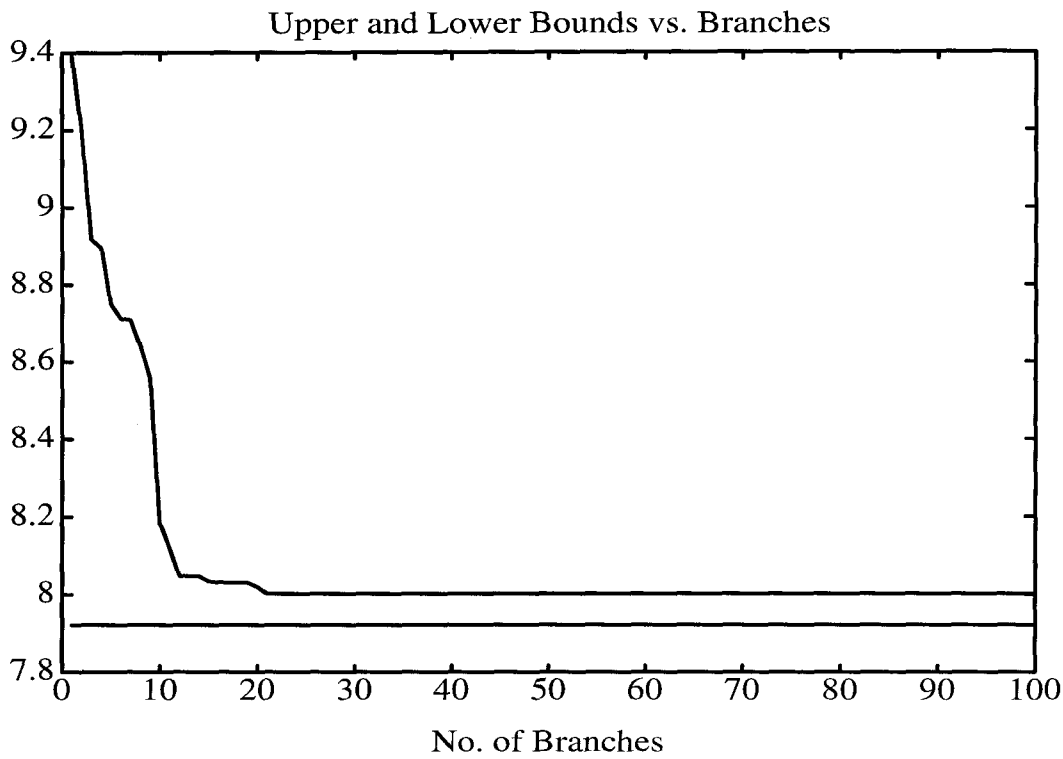


Figure 5.3: Progress on a problem with only γ continuous bounds.

Figure 5.5 shows that the growth rate for algorithm B, which uses inexpensive bounds (the induced norm for the upper bound and evaluation in the center of the box for the lower bound), is unacceptable for any of the levels "B100"-"B10." On the other hand, algorithm A, which uses expensive bounds, is quite efficient.

In contrast to Figure 5.5, which compares the progress of a branch and bound scheme with good but expensive bounds to one with crude but cheap bounds, Figure 5.6 compares the progress of the branch and bound scheme with good but expensive bounds to one with bounds that are nearly as good and nearly as expensive. The upper plot was generated using a branch and bound scheme using the mixed μ upper bound in Theorem 5.9 (scheme A), while the lower plot came from a scheme (scheme C) employing the same lower bound and an upper bound obtained by covering the real parameters with complex ones and evaluating the complex μ upper bound (equivalent to enforcing the choice $G = 0$ in Equation (5.4)). This bound is a little cheaper to compute but not quite as good as Equation (5.4). We have plotted the relative gap between the upper and lower bounds versus the number of branches on a log-log scale. It is clear that even this level of reduction in the quality of the bounds markedly affects the performance of the overall scheme. Thus we are led to conclude once more that the performance of the bounds is crucial to the

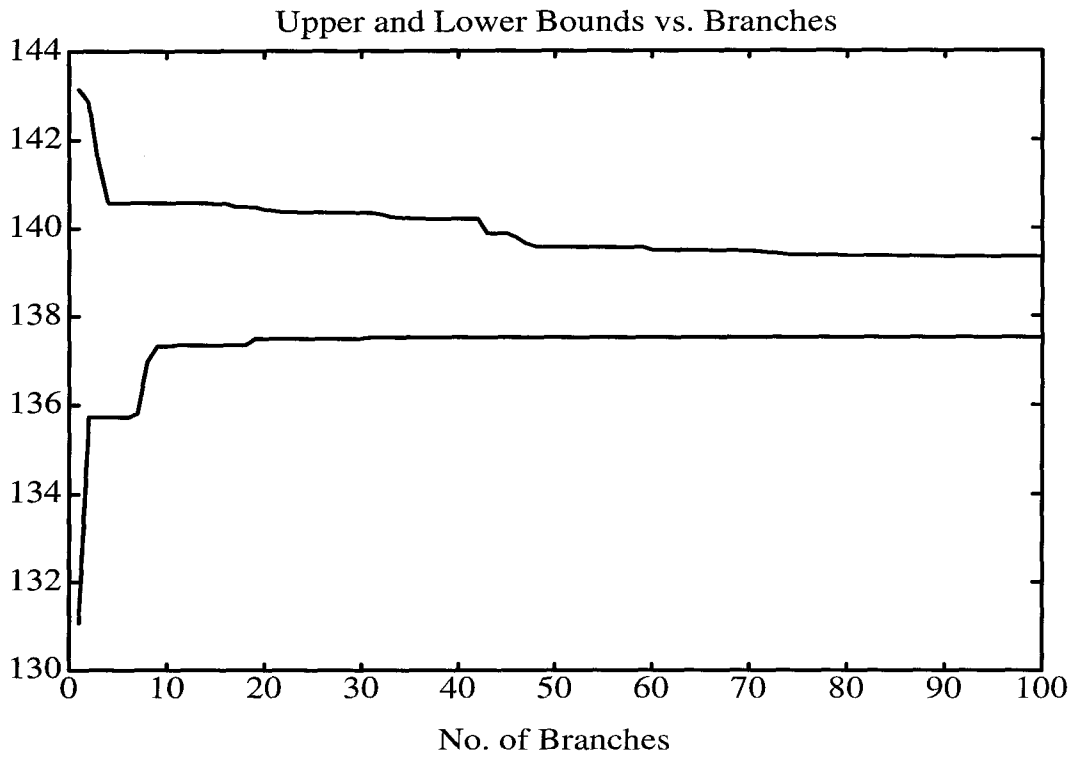


Figure 5.4: Progress on a problem with 20 parameters.

performance of the overall scheme, and that for a high performance branch and bound scheme it is important to use the best bounds available.

In Figure 5.7 we have plotted the required number of branches versus number of real parameters for a series of branch and bound tests of differently specified convergence criteria. The uncertainties consisted of 2 to 64 real scalars, depending on the problem, and approximately one fourth as many complex scalars. Each curve is a plot of the most difficult problem encountered from a pre-set number of runs, where for each problem the requirement for convergence was to reach a pre-specified tolerance between the upper and lower bounds, as labeled on the curve. Tolerances of 1 percent, 5 percent, 10 percent and 20 percent were considered, and for any problem the run was terminated if it failed to converge to the required tolerance within 100 branches. (Some of the curves terminate prematurely if the next problem size did not converge in time.) The graph is plotted on a log-linear scale, so that any straight line with non-zero slope represents an exponential growth rate.

It is clear from Figure 5.7 that if the tolerance is set tight enough then the typical growth rate is unacceptable (see the 1 percent curve for example). Thus as the problem size increases the required computation quickly becomes impractical, so we cannot expect to be able to achieve these tolerances. For the 20 percent curve, however, the computational requirements remain modest

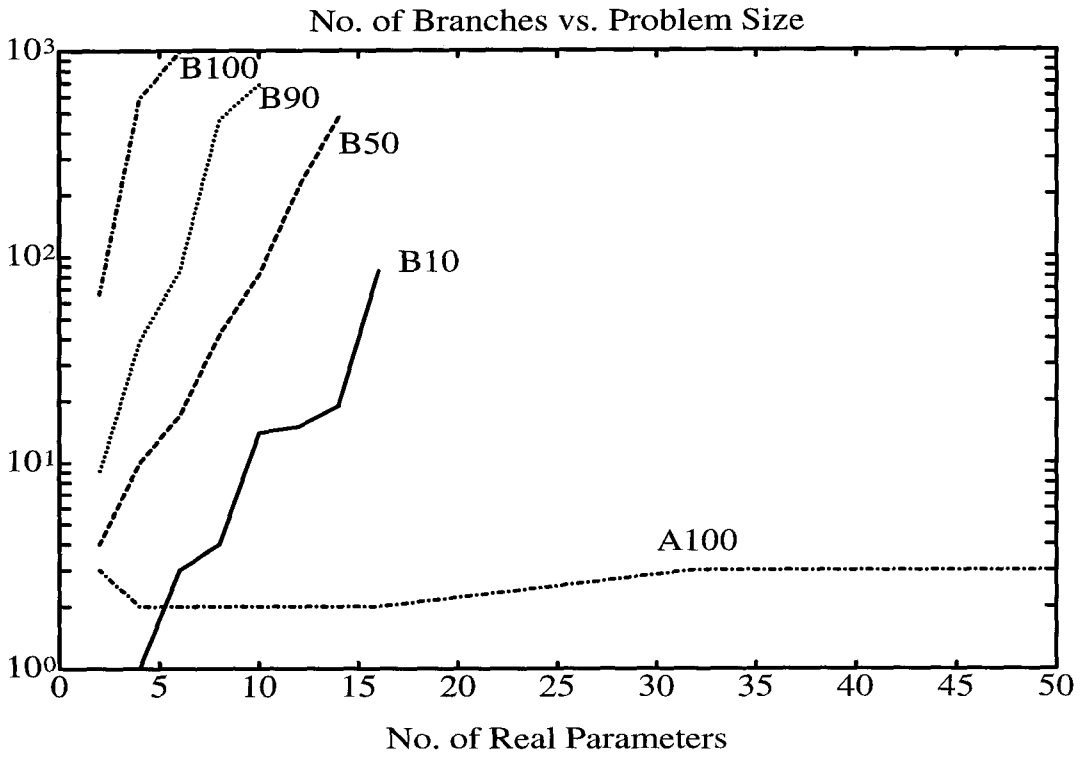


Figure 5.5: The growth rate of branch and bound computation steps versus number of real parameters, for scheme A and scheme B.

even for the largest problems tested. Fortunately this degree of accuracy is usually quite sufficient for engineering purposes. This suggests that branch and bound should be applied only in a carefully limited way.

Thus the branch and bound scheme should not be used as a general computation scheme per se, but only to fix the occasional problems for which the bounds are poor, and for these problems to achieve the degree of accuracy of the normal bounds on more typical problems. This emphasizes the necessity of good bounds.

To further illustrate this point consider the plot in Figure 5.8. This plot shows a mixed μ computation for a problem with 1 complex and 4 real scalar uncertainties, where the initial bounds were quite poor (85 percent relative gap as opposed to a typical level of less than 20 percent). We have plotted the current upper and lower bounds for the problem versus the number of branches, so that the progress of the branch and bound scheme on the problem can be seen. Initially quite rapid progress is made so that in only 29 steps the new bounds are within 20 percent. It is also apparent that the progress of the scheme slows quite dramatically after this point, so that achieving greater levels of accuracy requires substantially more computational effort and rapidly becomes impractical.

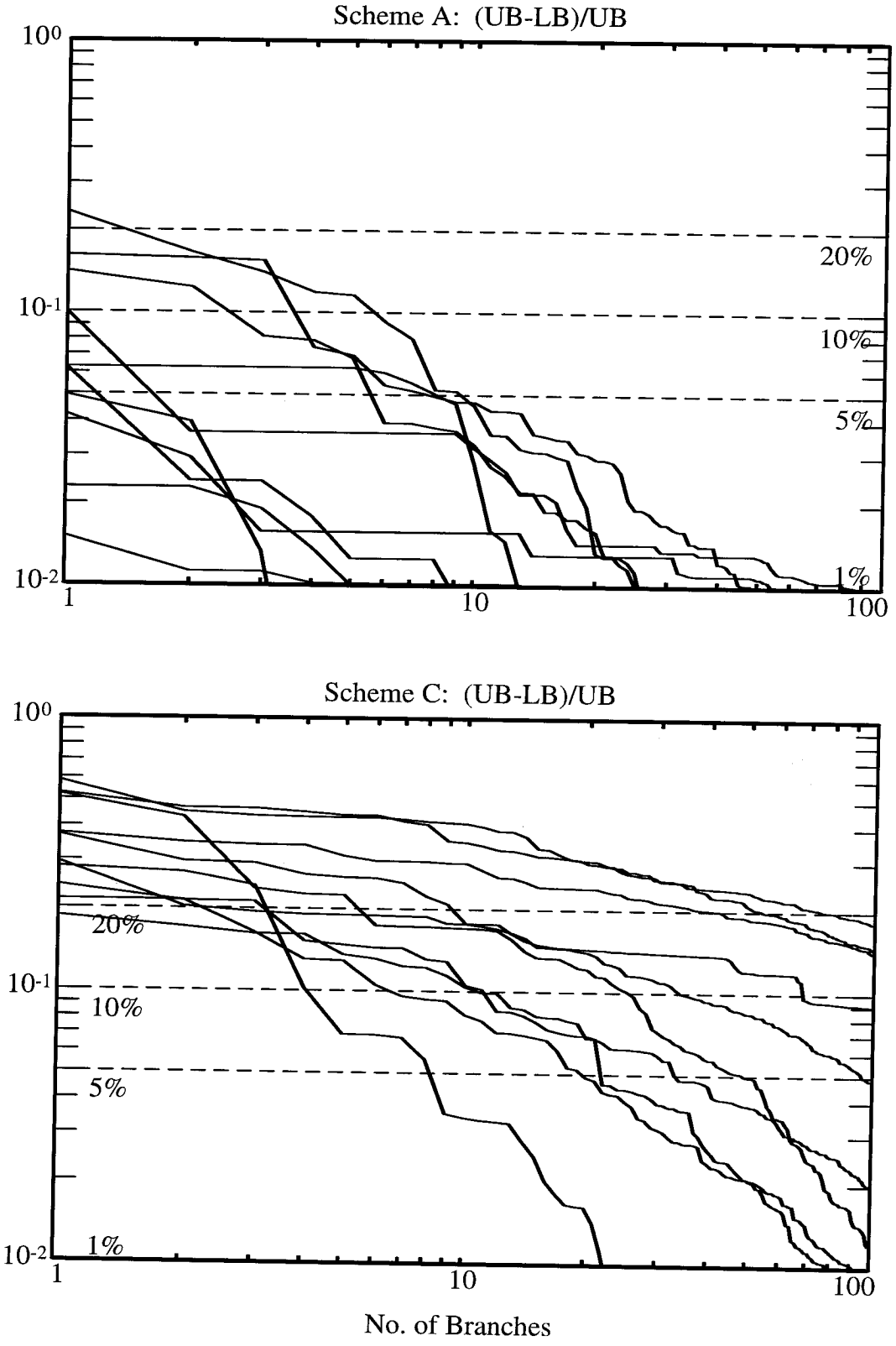


Figure 5.6: A comparison of branch and bound schemes.

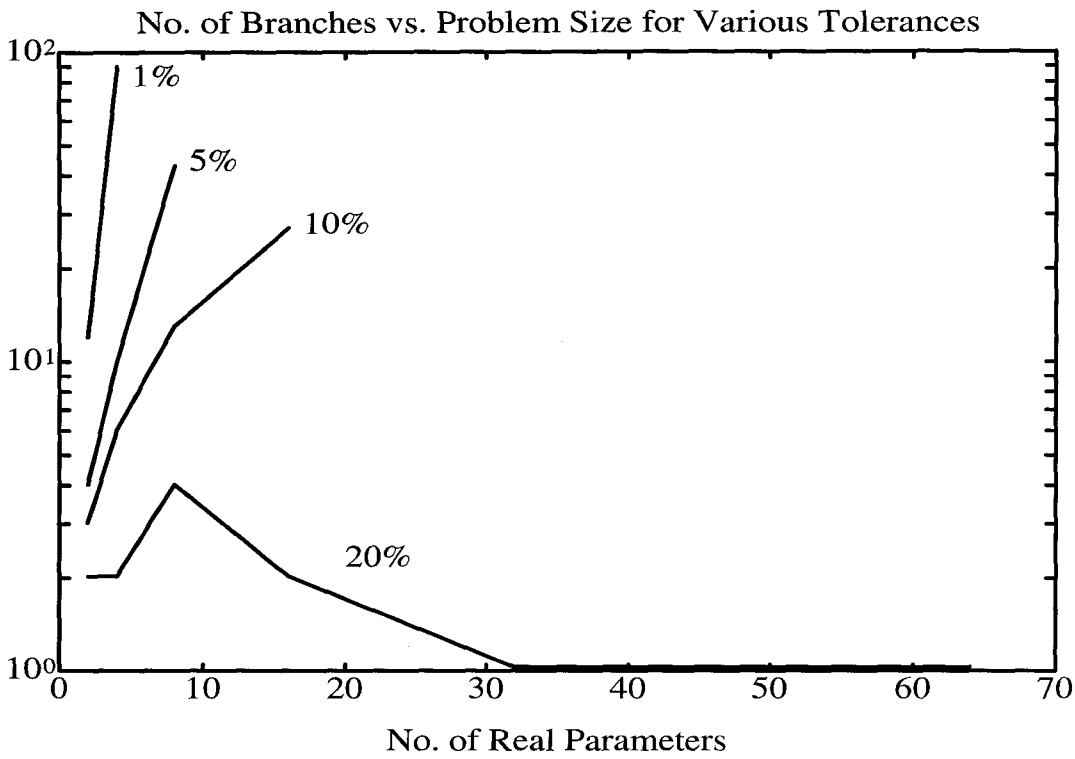


Figure 5.7: Branch and bound computational requirements for varying degrees of required accuracy.

We conclude that the sensible application of branch and bound techniques to the mixed μ problem requires some restraint: we must use branch and bound only on the worst problems (those with the biggest gaps between the upper and lower bounds) and use it only until the gaps are tolerably small. Any more aggressive use of branch and bound can easily result in unacceptably large computation requirements. Even with the proposed application of branch and bound there will occasionally be problems which we must give up on, because the require too much computation to bring the bounds within tolerance. Figure 5.9 shows the results of the proposed application of branch and bound to 49 problems with 16 real blocks. The branch and bound scheme requires three times the computation to achieve adequate bounds for all of the problems where the standard computation, without branch and bound, is inadequate.

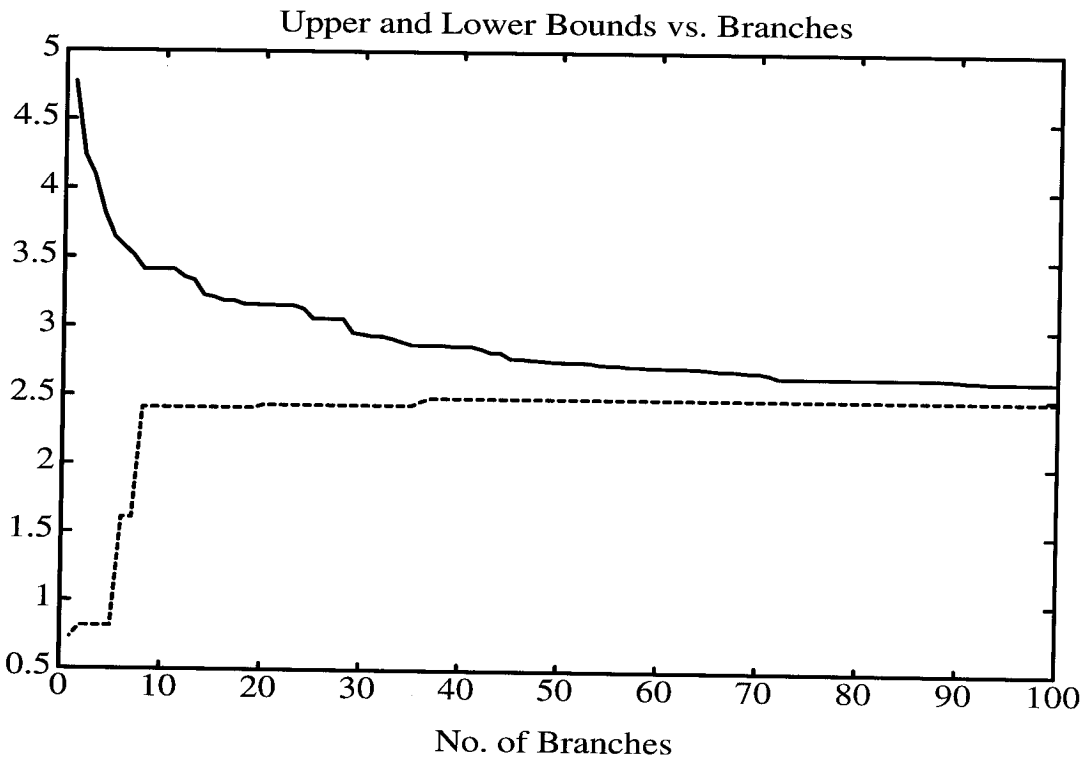


Figure 5.8: Progress of branch and bound for a hard problem.

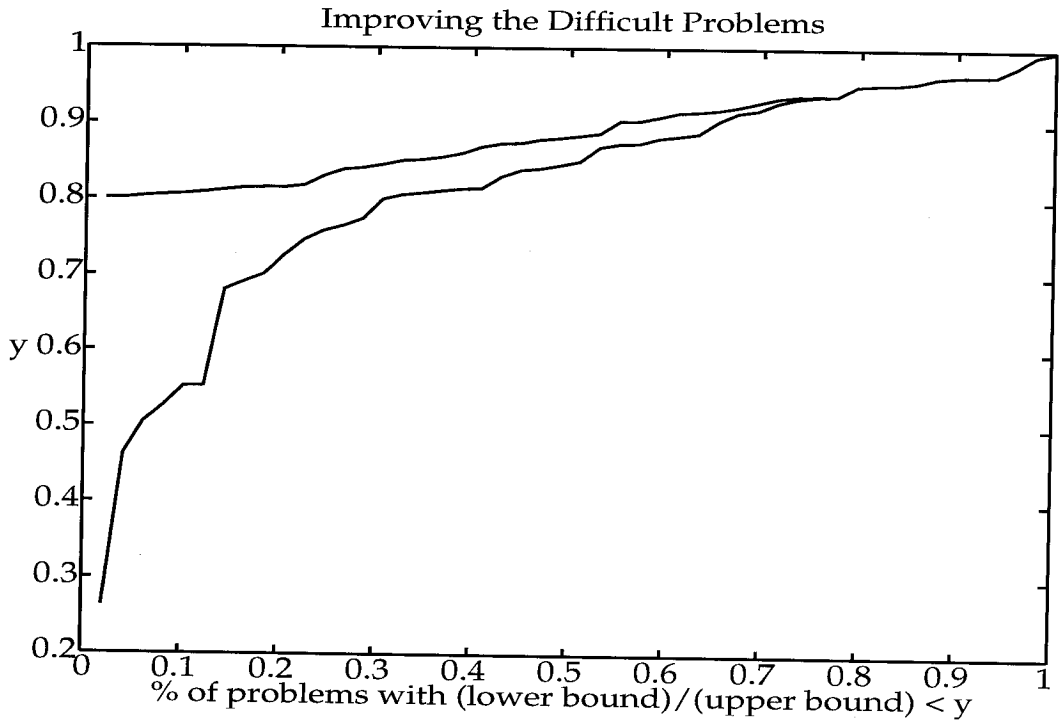


Figure 5.9: The proposed application of branch and bound to the mixed μ problem vs. no branch and bound. The branch and bound scheme required three times as much computation in order to ensure that each of the 49 problems satisfied (lb/ub)$>$ 0.8.

Chapter 6

Model Validation

In previous chapters we have seen that robust control theory gives us the power to represent physical systems with a model that includes a particular type of uncertainty: block structured norm-bounded uncertainty, representing perturbations to the nominal model, entering the model in a linear fractional manner. We have also seen that computation for evaluating worst-case performance of the (model of the) closed loop system is practical. This is robust performance analysis. Before the robust control methods can be applied, we are faced with the difficulty of selecting good models for the analysis of a system and for the synthesis of controllers. The uncertainty, both perturbations and noise, must, in some sense, be identified.

Current identification methods are well developed for models where all of the residuals are attributed to additive noise, but these are not adequate for robustness analysis in that they cannot account for a commonly encountered circumstance: the loss of stability under feedback not predicted by the nominal model. Robust control models can predict such situations, since they include norm-bounded unmodeled dynamics. This in not an issue in the SISO case because good margins imply robust performance. The natural MIMO extension of SISO margins is norm-bounded uncertainty.

Presently, no identification techniques exist for models with additive noise and norm-bounded perturbations. It is in fact a poorly posed problem: the physical system can only be observed by input output measurements and, for reasonably posed problems, the residuals can be attributed either entirely to additive noise or entirely to norm-bounded perturbations. In practice, an we would run many experiments attempting to isolate the effects of noise from those of perturbations.

The model validation problem, originally formulated in the robust control context by Smith and Doyle in [35], provides a connection between control theory and reality. The model validation procedure evaluates the applicability of a specified robust control model with respect to an input output experiment. It determines whether or not there is an element of the robust control model set which accounts for the experimental observation. The model validation test

therefore provides a necessary condition for a model to describe a physical system. The work presented in this chapter begins to address the difficult issue of obtaining suitable robust control models for physical systems.

In this chapter we present a generalization of μ, denoted by μ_g, because it solves the model validation problem. The similarity to μ is pronounced. The principal difference is that in μ_g each perturbation, Δ, in the set of perturbations, $\mathbf{\Delta}$, is now divided into two parts, Δ_J and Δ_K, and the system is stable when $\|\Delta_J\| < 1/\mu_g$ and $\|\Delta_K\| > \mu_g$. A similar result for a special case in which a particular part of the system is square is due to Packard ([25]). Here, we treat the general case.

Although the generalization of the μ framework and computation is important beyond the model validation problem, we reserve that topic for Chapter 7. Here, we study the model validation problem and its computation in detail.

The approach taken for the calculation of μ involves the development of more readily calculated upper and lower bounds and we take a similar approach for μ_g. A high quality μ_g upper bound can be formulated as an LMI, and thus poses no more difficulties than the convex LMI calculation of the μ upper bound. The lower bound computation is also similar to the μ lower bound computation, and the techniques of Chapter 3 are generally applicable.

The model validation problem is introduced in Section 6.1 and the constant matrix case considered in this chapter is formulated. The generalization of μ is introduced in Section 6.2. Necessary and sufficient conditions for the well posedness of μ_g problems are also presented.

Section 6.3 formulates the model validation problem as a well posed μ_g problem. The connection is made through a robust minimum gain interpretation of the μ_g problem, also presented in Section 6.3. In some special cases the μ_g problem can be reformulated as a standard robust performance problem for an inverted system. The conditions under which a μ_g problem can be easily recast as a μ problem in this way are given.

Critical to the application of μ_g to the model validation problem is the calculation of an upper bound for μ_g, which is studied in Section 6.4. It is shown that an upper bound can be calculated as a convex LMI optimization problem. Under certain conditions, also presented in this section, this upper bound is exactly equal to μ_g. The lower bound is presented in Section 6.5, along with an algorithm for its calculation, and the theory behind the algorithm. The theoretical development presented here contains the standard μ results as an easy special case.

6.1 The Model Validation Problem

As with much of robust control theory, the model validation problem is naturally stated in the continuous time domain. In practical applications however, data is taken by sampling, and LTI problems can be transformed to the frequency domain as matrix problems at a finite number of frequencies. This section introduces the model validation problem somewhat abstractly, then states precisely the constant matrix problem addressed subsequently.

An assumption, inherent in the use of any model, is that the model can describe any observed input output behavior of the physical system, past, present, or future. Model validation addresses a necessary condition: that the model be able to describe all previously observed input output behavior of the system. Consistency with all experimental data provides little hard information about the applicability of the model. There may be experiments, as yet unperformed, which will invalidate the model. Model validation is a misleading term: it is not possible to validate a model of any descriptive value, it is possible only to invalidate a model. Model validation is the process of collecting data and invalidating models.

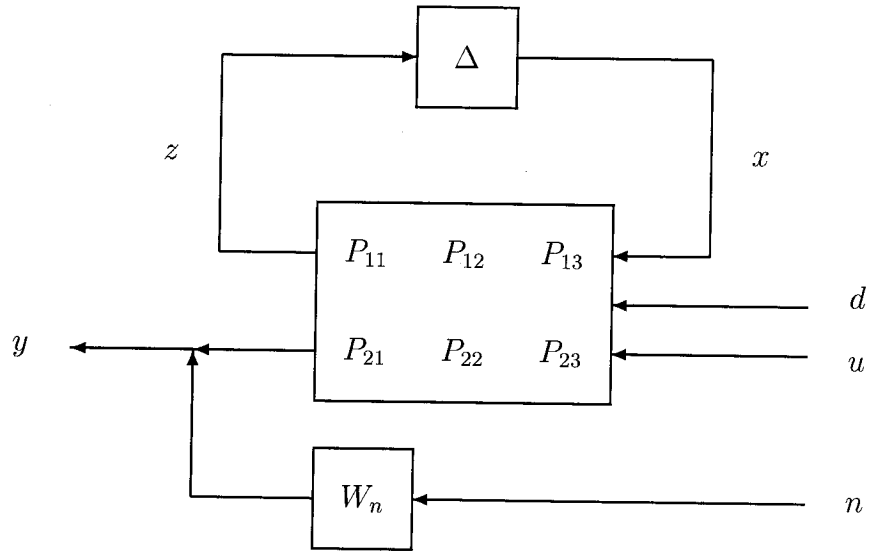


Figure 6.1: The block diagram of the model validation problem.

A generic model validation structure is shown in Figure 6.1. The unknown inputs acting directly on the output y, corresponding to measurement error and sensor noise, are denoted by n. The unknown inputs acting on the plant model itself are denoted by d. Note that Figure 6.1 is a special case of Figure 6.5, with the input partitioned into three distinct parts, two unknown and one known, and with the output known.

The formulation here differs slightly from that considered in previous work in that the measurement noise, n, is explicitly distinguished from the other disturbances, d. This distinction, along with the condition that W_n be invertible, is a convenient way to insure that the problem meets the requirement that every measured output, y, is modeled as being corrupted by additive noise. This requirement is reasonable in any physically motivated problem, and can be met with more general formulations, but the resulting notation is awkward.

The perturbation, Δ, has a prescribed block structure, $\Delta \in \mathbf{\Delta}$, and norm bound, $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$. We assume that the models under consideration are stable for all $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$ to insure that $(I - P_{11}\Delta)^{-1}$ is always well defined. Again, this is a reasonable practical assumption since a model set which includes both stable and unstable models is unlikely to be a good description of a given physical system.

Note that if $y - P_{23}u = 0$ then the model validation problem is solved trivially with n, d and Δ all zero. In the following, we assume that this is not the case.

A formal statement of the model validation problem is now given.

Problem 6.1 (Model Validation) *Let P be a robustly stable plant model with block structure $\mathbf{\Delta}$ as in Figure 6.1. Given measurements (u, y), do there exist a $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$ and signals d and n satisfying $\|d\| \leq 1$ and $\|n\| \leq 1$, such that*

$$y = W_n n + (\Delta \star P) \begin{bmatrix} d \\ u \end{bmatrix} ? \tag{6.1}$$

Any (Δ, d, n) satisfying these conditions are referred to as admissible. The existence of and admissible (Δ, d, n) is a necessary condition for the model to be able to describe the system. On the other hand, if no such Δ, d, and n exist the model cannot account for the observation and we say that y and u invalidate the model. Model validation gives conclusive information only when there is no model in the set that is consistent with y and u; there is no way of proving that a model is valid simply because there is no way of testing every experimental condition.

If the system is in the continuous time domain the model validation test should be performed for continuous time measurements y and u. In practice, however, data is taken by sampling, and LTI problems can be reformulated as matrix tests at a finite number of frequencies. Then, at each frequency, P and Δ are complex valued matrices and n and d are complex valued vectors. In this case, the statement of the model validation problem remains the same, except that now all signals and operators are vectors and matrices. In the work presented here we consider this formulation of the problem.

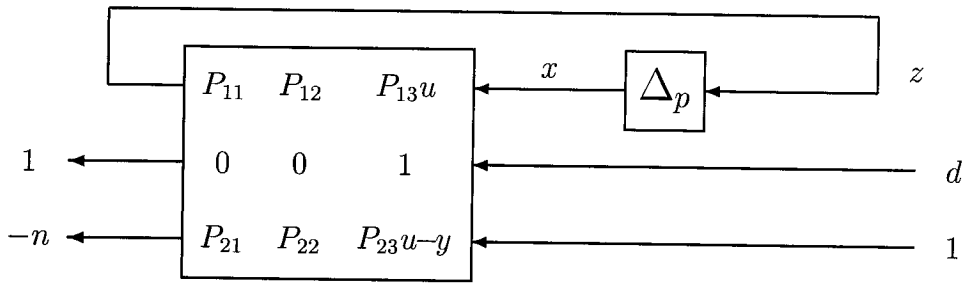
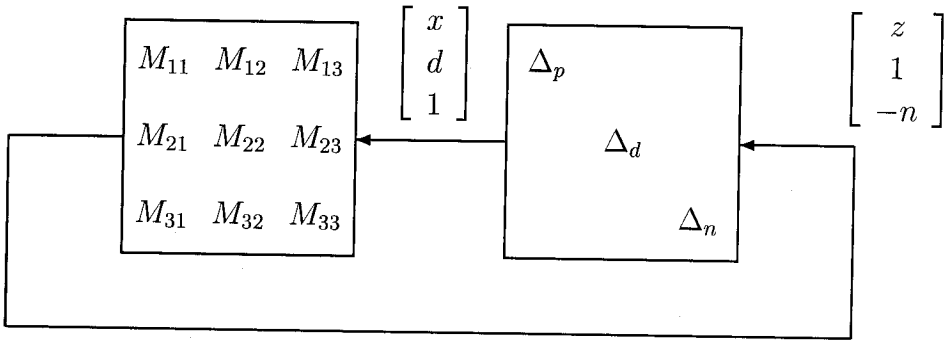


Figure 6.2: The equivalent system with no signal constraints.



with

$$\begin{array}{lll} M_{11} = P_{11} & M_{12} = P_{12} & M_{13} = P_{13}u \\ M_{21} = 0 & M_{22} = 0 & M_{23} = 1 \\ M_{31} = P_{21} & M_{32} = P_{22} & M_{33} = P_{23}u - y \end{array}$$

Figure 6.3: The equivalent system as a generalized μ problem.

Solving for n in terms of x, d, y and u then absorbing the known signals y and u into P results in the relationships represented by Figure 6.2. Solutions to Figure 6.2 are equivalent to solutions to Figure 6.1. The advantage of this manipulation is that there are no signal constraints among the solutions we seek.

The difference between the model validation problem as represented in Figure 6.2 and the robust performance analysis problem (which has a similar block diagram) is that we are not looking for solutions with the largest gain from input to output. Rather, we want the norm of the output, $-n$, to be small. Figure 6.3 represents the same equations as those represented by Figure 6.2, with Δ_n and Δ_d defined so that $1 = \Delta_n(-n)$ and $d = \Delta_d 1$, so a small n corresponds to a large Δ_n. Thus the model validation problem motivates

a generalization of μ where we look for a Δ that satisfies $\det(I - M\Delta) = 0$ with some of the delta blocks as small as possible and some of the delta blocks as large as possible. Solutions of this generalization of μ are solutions to Figure 6.1 with the smallest (Δ_p, n, d) possible. These solutions answer questions such as

- What are the smallest (Δ_p, n, d) that are consistent with the data (y, u)?
- Given a norm bound on Δ_p, what is the smallest noise (n, d) consistent with the data?
- Given a norm bound on (n, d), what is the smallest perturbation Δ_p consistent with the data?
- Is a model with specified norm bounds on n, d, and Δ_p consistent with the observed data?

Answering this last question solves the model validation problem.

6.2 A Generalization of the Structured Singular Value

The μ framework considers perturbation blocks, Δ, satisfying a maximum norm constraint. Here we introduce a second class of perturbations which satisfy a minimum gain constraint. To formalize this class, consider a block structure, $\mathbf{\Delta}$, partitioned in two with the index sets J and K as follows.

$$\mathbf{\Delta} = \begin{bmatrix} \mathbf{\Delta}_J & 0 \\ 0 & \mathbf{\Delta}_K \end{bmatrix}, \tag{6.2}$$

where

$$\mathbf{\Delta}_J = \{ \operatorname{diag}(\ldots \Delta_j \ldots) \mid j \in J \} \tag{6.3}$$

$$\mathbf{\Delta}_K = \{ \operatorname{diag}(\ldots \Delta_k \ldots) \mid k \in K \}. \tag{6.4}$$

Consider a matrix, M, partitioned in accordance with $\mathbf{\Delta}$,

$$M = \begin{bmatrix} M_{JJ} & M_{JK} \\ M_{KJ} & M_{KK} \end{bmatrix}.$$

Let $z = Mx$ as in Figure 6.4. The positive function $\mu_g(M)$ is defined for $M \in \operatorname{dom}(\mu_g)$ by

$$\mu_g(M) \triangleq \max_{\|x\|=1} \left\{ \gamma \;\middle|\; \begin{array}{rcl} \|x_j\|\,\gamma & \leq & \|z_j\|\,, \forall j \in J \\ \|z_k\|\,\gamma & \leq & \|x_k\|\,, \forall k \in K \end{array} \right\}. \tag{6.5}$$

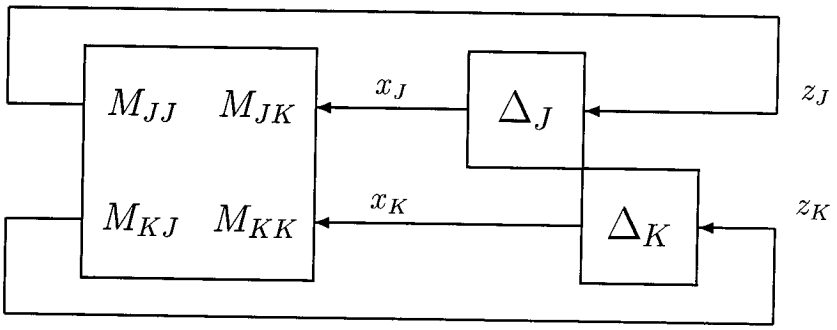


Figure 6.4: The interconnection structure for the generalization of μ.

The domain of definition of μ_g, $\mathrm{dom}(\mu_g)$, is discussed in the next section.

The form of this definition (and that of Equation (1.4)) is similar to the more general Integral Quadratic Constraints (IQCs, see Megretski and Treil [22]).

Well Posedness and a Simple Bound

A well posed μ_g problem is one where the maximization problem in Equation (6.5) has a well defined finite solution. The above definition of μ_g is well posed if and only if M_{KK} does not have a nontrivial kernel. The collection of such problems is denoted $\mathrm{dom}(\mu_g)$. More formally,

$$\mathrm{dom}(\mu_g) = \{\, M \mid M_{KK}x_K = 0 \implies x_K = 0 \}$$

Clearly, if M_{KK} has a nontrivial kernel then choosing $x_J = 0$, $x_K \in \ker\{M_{KK}\}$, $\|x_K\| = 1$, allows all γ to satisfy the inequalities of Equation (6.5), thus rendering the definition of μ_g vacuous.

The well posedness requirement implies that $\dim(z_K) \geq \dim(x_K)$. (See Figure 6.4.) In the case where $\dim(z_K) = \dim(x_K)$, well posedness is equivalent to the invertibility of M_{KK}. This case leads to a reformulation of $\mu_g(M)$ as a μ problem on another matrix,as described shortly. All other results are stated for the more general and, as we shall see, more applicable nonsquare case.

The following lemma shows that the answer to every $\mu_g(M)$ problem in which M_{KK} does not have a nontrivial kernel is finite by giving an easily calculated upper bound to $\mu_g(M)$. The bound also can be used to initialize the computation of the more precise upper bound discussed in Section 6.4.

Lemma 6.2 *If* $M \in \mathrm{dom}(\mu_g)$ *then*

$$\mu_g(M) \leq \|M_{JJ}\| + \frac{\|M_{JK}\| \, \|M_{KJ}\| + 1}{\underline{\sigma}(M_{KK})}.$$

Proof: Let $M \in \mathrm{dom}(\mu_g)$, $x \neq 0$, and

$$\begin{aligned} \|x_J\| \gamma &\leq \|M_{JJ}x_J + M_{JK}x_K\| \\ \|M_{KJ}x_J + M_{KK}x_K\| \gamma &\leq \|x_K\| . \end{aligned}$$

Then we have one of the following four cases.

1. $x_J = 0$. Then
   $x_K \neq 0$ so $\gamma \leq \frac{\|x_K\|}{\|M_{KK}x_K\|} \leq \frac{1}{\underline{\sigma}(M_{KK})}$.
2. $x_J \neq 0$, $M_{JK} = 0$. Then
   $\gamma \leq \frac{\|M_{JJ}x_J\|}{\|x_J\|} \leq \|M_{JJ}\|$.
3. $x_J \neq 0$, $M_{JK} \neq 0$, $\frac{\|x_K\|}{\|x_J\|} \leq \frac{\|M_{KJ}\|+\|M_{JK}\|^{-1}}{\underline{\sigma}(M_{KK})}$. Then
   $$\begin{aligned} \gamma &\leq \frac{\|M_{JJ}x_J+M_{JK}x_K\|}{\|x_J\|} \\ &\leq \|M_{JJ}\| + \|M_{JK}\| \frac{\|M_{KJ}\|+\|M_{JK}\|^{-1}}{\underline{\sigma}(M_{KK})} \\ &= \|M_{JJ}\| + \frac{\|M_{JK}\|\|M_{KJ}\|+1}{\underline{\sigma}(M_{KK})}. \end{aligned}$$
4. $x_J \neq 0$, $M_{JK} \neq 0$, $\frac{\|x_K\|}{\|x_J\|} > \frac{\|M_{KJ}\|+\|M_{JK}\|^{-1}}{\underline{\sigma}(M_{KK})}$. Then
   $$\begin{aligned} \gamma &\leq \frac{\|x_K\|}{\|M_{KJ}x_J+M_{KK}x_K\|} < \frac{\|x_K\|\|x_J\|^{-1}}{\underline{\sigma}(M_{KK})\|x_K\|\|x_J\|^{-1}-\|M_{KJ}\|} \\ &\leq \frac{(\|M_{KJ}\|+\|M_{JK}\|^{-1})\underline{\sigma}(M_{KK})^{-1}}{\|M_{KJ}\|+\|M_{JK}\|^{-1}-\|M_{KJ}\|} = \frac{\|M_{JK}\|\|M_{KJ}\|+1}{\underline{\sigma}(M_{KK})}. \end{aligned}$$

And the lemma is proved. ∎

Similar arguments lead to the tighter but messier upper bound

$$\gamma \leq \|M_{JJ}\| + \frac{N + [N^2 - 4 \, \|M_{JJ}\| \, \|M_{JK}\| \, \|M_{KJ}\| \, \underline{\sigma}(M_{KK})]^{1/2}}{2\underline{\sigma}(M_{KK})}$$

where $N = 1 + \|M_{JK}\| \, \|M_{KJ}\| - \|M_{JJ}\| \, \underline{\sigma}(M_{KK})$.

Thus we see that $M \in \mathrm{dom}(\mu_g)$ if and only if $\mu_g(M)$ is well posed, so that the definition of $\mathrm{dom}(\mu_g)$ above is justified.

A Special Case: Formulation as a μ Problem

In this section we present the first of two distinct cases of μ_g computation. When M_{KK} is square the μ_g problem can be recast as a μ problem and μ calculation approaches can be used. In the general nonsquare case the bounds

computation of previous chapters must be generalized. These generalizations are addressed in subsequent sections.

When M_{KK} is square and $\mu_g(M)$ is well posed, the μ_g problem is equivalent to a μ problem. This is stated precisely in the following theorem.

Theorem 6.3 *If $M \in \mathrm{dom}(\mu_g)$ and M_{KK} is square then*

$$\mu_g(M) = \mu(\widehat{M}),$$

where

$$\widehat{M} \triangleq \begin{bmatrix} M_{JJ} - M_{JK}{M_{KK}}^{-1}M_{KJ} & M_{JK}{M_{KK}}^{-1} \\ -{M_{KK}}^{-1}M_{KJ} & {M_{KK}}^{-1} \end{bmatrix},$$

and the block structure for the $\mu(\widehat{M})$ problem is

$$\mathbf{\Delta} = \mathrm{diag}(\mathbf{\Delta}_J, \mathbf{\Delta}_K^T).$$

Proof: $M \in \mathrm{dom}(\mu_g)$ and M_{KK} is square implies that ${M_{KK}}^{-1}$ exists. Noting that

$$\begin{array}{rl} & \left\{ \begin{array}{rcl} y &=& M_{KJ}x + M_{KK}u \\ z &=& M_{JJ}x + M_{JK}u \end{array} \right. \\ \iff & \left\{ \begin{array}{rcl} u &=& -{M_{KK}}^{-1}M_{KJ}x + {M_{KK}}^{-1}y \\ z &=& M_{JJ}x + M_{JK}(-{M_{KK}}^{-1}M_{KJ}x + {M_{KK}}^{-1}y) \end{array} \right. \end{array}$$

we see that ♣ and ♠ below are equivalent.

$$\clubsuit \qquad \begin{bmatrix} z \\ y \end{bmatrix} = M \begin{bmatrix} x \\ u \end{bmatrix} \text{ and } \left\{ \begin{array}{rcl} \|x\|\gamma &\leq& \|z\| \\ \|y\|\gamma &\leq& \|u\| \end{array} \right\}$$

$$\spadesuit \qquad \begin{bmatrix} z \\ u \end{bmatrix} = \widehat{M} \begin{bmatrix} x \\ y \end{bmatrix} \text{ and } \left\{ \begin{array}{rcl} \|x\|\gamma &\leq& \|z\| \\ \|y\|\gamma &\leq& \|u\| \end{array} \right\}.$$

Now,

$$\begin{array}{rl} & \mu_g(M) \geq \gamma \\ \iff & \exists\, (u,x,y,z)\colon \clubsuit \\ \iff & \exists\, (u,x,y,z)\colon \spadesuit \\ \iff & \mu(\widehat{M}) \geq \gamma, \end{array}$$

so

$$\begin{array}{rcl} \mu_g(M) = \gamma & \Longrightarrow & \mu(\widehat{M}) \geq \gamma, \text{ and} \\ \mu_g(M) \geq \gamma & \Longleftarrow & \mu(\widehat{M}) = \gamma. \end{array}$$

Thus $\mu_g(M) = \mu(\widehat{M})$. ■

Theorem 6.3 is applicable to model validation only in the case where the output, y, is a scalar. Section 6.4 presents an upper bound for the general case, where M_{KK} is not square.

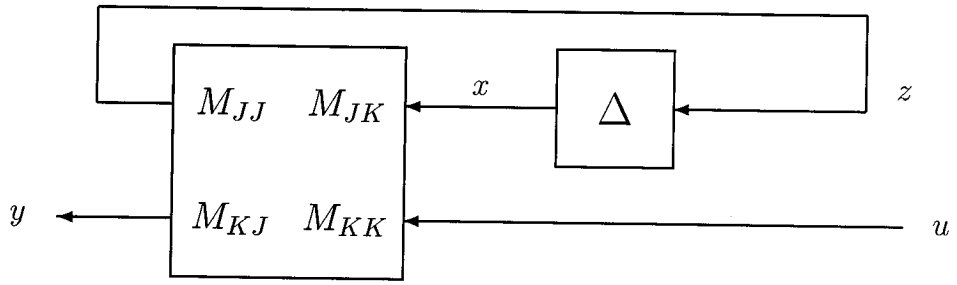


Figure 6.5: The generic robust control model structure.

6.3 μ_g Solves the Model Validation Problem

This section shows that the model validation problem is a special case of the generalized μ problem. It is convenient to first show that the μ_g problem solves a robust minimum gain problem, where the minimum $\mathcal{L}_2$ to $\mathcal{L}_2$ induced gain of an LFT system is bounded below for all perturbations in a ball. In contrast, the standard robust performance problem of Section 1.2 states performance as a maximum gain problem.

To pose the robust minimum gain problem more precisely, consider the generic system illustrated in Figure 6.5. The nominal ($\Delta = 0$) transfer function is $y = M_{KK}(s)u$, and the new performance requirement is $\|y\| \geq \|u\|$ for all $u \in \mathcal{L}_2$. The nominal performance requirement is equivalent to

$$\inf_{\omega} \underline{\sigma}(M_{KK}(j\omega)) \geq 1.$$

Now consider the performance requirement for the perturbed system,

$$y = (\Delta(s) \star M(s))u, \quad \Delta(s) \in \mathbf{B}_{\mathbf{\Delta}}.$$

In this context, robust performance is equivalent to

$$\inf_{\omega} \underline{\sigma}(\Delta \star M(j\omega)) \ \geq 1, \quad \forall \Delta \in \mathbf{B}_{\mathbf{\Delta}}.$$

The appropriate constant matrix test, i.e. the test for each fixed ω, is formulated as a μ_g problem in Theorem 6.4 below. We assume that $(\Delta \star M)$ is robustly stable, which is equivalent to $\mu(M_{JJ}) < 1$ (recall Theorem 1.4), and that the nominal system satisfies the performance criteria so that $\ker(M_{KK}) = \emptyset$.

Theorem 6.4 *Let $\mu(M_{JJ}) < 1$, and $M \in \mathrm{dom}(\mu_g)$. Then*

$$\min_{\Delta \in \mathbf{B}_{\mathbf{\Delta}}} \underline{\sigma}(\Delta \star M) > 1 \quad \Longleftrightarrow \quad \mu_g(M) < 1,$$

where the block structure, $\widetilde{\mathbf{\Delta}}$, for the μ_g problem is $\mathbf{\Delta}_J = \mathbf{\Delta}$, $\mathbf{\Delta}_K = \mathbb{C}^{\dim(u)\times\dim(y)}$, and $\widetilde{\mathbf{\Delta}} = \mathrm{diag}(\mathbf{\Delta}_J, \mathbf{\Delta}_K)$.

Proof: $\mu(M_{JJ}) < 1$ implies that $(\Delta \star M)$ is well defined, and $M \in \mathrm{dom}(\mu_g)$ implies that $\mu_g(M)$ is well defined. Let $\begin{bmatrix} z \\ y \end{bmatrix} = M \begin{bmatrix} x \\ u \end{bmatrix}$ as in Figure 6.5. Then the following are equivalent.

- $\mu_g(M) < 1$.
- $\nexists\, (z, y, x, u) : \|x_j\| \le \|z_j\| \,\forall j \in J$ and $\|y\| \le \|u\|$.
- $\|x_j\| \le \|z_j\| \,\forall j \in J \implies \|y\| > \|u\|$.
- $\min_{\Delta\in\mathbf{B}_{\mathbf{\Delta}}} \underline{\sigma}(\Delta \star M) > 1$.

Thus the theorem is proved. ■

The negation of Theorem 6.4 is stated as a corollary below because we use it directly in the proof that μ_g solves the model validation problem.

Corollary 6.5 *Let $\mu(M_{JJ}) < 1$, and $M \in \mathrm{dom}(\mu_g)$. Then*

$$\{\, \Delta \mid \Delta \in \mathbf{B}_{\mathbf{\Delta}}, \underline{\sigma}(\Delta \star M) \le 1 \} \neq \emptyset$$

where the block structure, $\widetilde{\mathbf{\Delta}}$, for the μ_g problem is $\mathbf{\Delta}_J = \mathbf{\Delta}$, $\mathbf{\Delta}_K = \mathbb{C}^{\dim(u)\times\dim(y)}$, and $\widetilde{\mathbf{\Delta}} = \mathrm{diag}(\mathbf{\Delta}_J, \mathbf{\Delta}_K)$.

Comparing $\mu_g(M)$ to 1 in Theorem 6.4 obscures the relationship for other values. For completeness we state below a scaled version of the theorem.

Corollary 6.6 *Let $\mu(M_{JJ}) < \beta$, and $M \in \mathrm{dom}(\mu_g)$. Then*

$$\min_{\substack{\Delta\in\mathbf{\Delta} \\ \|\Delta\|\le 1/\beta}} \underline{\sigma}(\Delta \star M) > 1/\beta \iff \mu_g(M) < \beta,$$

where the block structure, $\widetilde{\mathbf{\Delta}}$, for the μ_g problem is $\mathbf{\Delta}_J = \mathbf{\Delta}$, $\mathbf{\Delta}_K = \mathbb{C}^{\dim(u)\times\dim(y)}$, and $\widetilde{\mathbf{\Delta}} = \mathrm{diag}(\mathbf{\Delta}_J, \mathbf{\Delta}_K)$.

Application to the Model Validation Problem

Recall the model validation problem from Section 6.1. This problem is a special case of the generalized μ problem. Define the matrix $\widetilde{P}$, formed from P, W_n, y, and u, as

$$\widetilde{P} = \begin{bmatrix} P_{11} & P_{12} & P_{13}u \\ 0 & 0 & 1 \\ W_n^{-1}P_{21} & W_n^{-1}P_{22} & W_n^{-1}(P_{23}u - y) \end{bmatrix}. \tag{6.6}$$

Define a block structure, $\widetilde{\mathbf{\Delta}}$, for $\mu_g(\widetilde{P})$ in the following manner.

$$\begin{array}{lcl} \mathbf{\Delta}_J & \triangleq & \left\{ \mathrm{diag}(\Delta, \Delta_d) \mid \Delta \in \mathbf{\Delta},\ \Delta_d \in \mathbb{C}^{\dim(d)\times 1} \right\} \\ \mathbf{\Delta}_K & \triangleq & \left\{ \Delta_n \mid \Delta_n \in \mathbb{C}^{1\times \dim(n)} \right\} \\ \widetilde{\mathbf{\Delta}} & \triangleq & \mathrm{diag}(\mathbf{\Delta}_J, \mathbf{\Delta}_K). \end{array} \tag{6.7}$$

With these definitions, $\mu_g(\widetilde{P})$ is well posed: these dimensions partition $\widetilde{P}$ such that $\widetilde{P}_{KK} = W_n^{-1}(P_{23}u - y)$; $\underline{\sigma}(\widetilde{P}_{KK}) > 0$ because $P_{23}u - y \neq 0$ by assumption (so that the model validation problem is nontrivial), and $\dim(\widetilde{P}_{KK}) = \dim(y)\times 1$.

The following theorem gives the solution to the model validation problem in the μ_g framework.

Theorem 6.7 *Let $\widetilde{P}$ and $\widetilde{\mathbf{\Delta}}$ be defined as in Equations (6.6) and (6.7) above. Then*

$$\mu_g(\widetilde{P}) \geq 1 \iff \exists \Delta \in \mathbf{B}_{\mathbf{\Delta}},\ \|d\| \leq 1,\ \textit{and}\ \|n\| \leq 1 \textit{ such that } y = W_n n + (\Delta \star P)\begin{bmatrix} d \\ u \end{bmatrix}.$$

Proof: Simple algebra shows

$$y = W_n n + (\Delta \star P)\begin{bmatrix} \Delta_d \\ u \end{bmatrix} \iff n = -(\Delta_J \star \widetilde{P}).$$

(Observe that Δ_d and $(\Delta_J \star \widetilde{P})$ are vectors.) Corollary 6.5 implies that the following are equivalent.

- $\mu_g(\widetilde{P}) \geq 1$.
- $\exists\ \Delta_J \in \mathbf{B}_{\mathbf{\Delta}_J}$: $\underline{\sigma}(\Delta_J \star \widetilde{P}) \leq 1$.
- $\exists \Delta \in \mathbf{B}_{\mathbf{\Delta}}$, d: $\|d\| \leq 1$, $\|n\| \leq 1$, where $y = W_n n + (\Delta \star P)\begin{bmatrix} d \\ u \end{bmatrix}$.

Thus the theorem is proved. ∎

Note that $\mu_g(\widetilde{P}) < 1$ (or any upper bound to μ_g being less than one) implies that the model is invalid. Therefore, the calculation of an upper bound is critical to obtaining conclusive information from the model validation problem. A convex optimization problem that gives an upper bound to μ_g is given in Section 6.4.

6.4 The Upper Bound

In this section we show that an LMI optimization problem provides an upper bound to μ_g. LMIs are convex optimization problems with convex constraints, and as such are computable with a wide variety of techniques. For simplicity, we treat only the case where the blocks of Δ are full complex blocks. The more general structures of Chapter 4 follow easily. Although the LMI developed here is new, the various optimization methods that have been applied to existing LMI problems are easily modified to address the computation of our LMI.

Invertible matrices are used to scale and sum the inequalities in the definition of μ_g. Let

$$\mathbf{d} \triangleq \{ (\ldots d_i \ldots) \mid d_i > 0 \; \forall \, i \in J \text{ and } d_i < 0 \; \forall \, i \in K \}.$$

Given a $d \in \mathbf{d}$, define the scaling matrices $D_{\mathbb{X}}(d)$ and $D_{\mathbb{Z}}(d)$ by

$$D_{\mathbb{X}}(d) \triangleq \operatorname{diag}(\ldots d_i I_{\mathbb{X}i} \ldots)$$

$$D_{\mathbb{Z}}(d) \triangleq \operatorname{diag}(\ldots d_i I_{\mathbb{Z}i} \ldots)$$

where $I_{\mathbb{X}i}$ and $I_{\mathbb{Z}i}$ result from the partitioning of the identity operators on $\mathbb{X}$ and $\mathbb{Z}$ respectively.

Note that $D_{\mathbb{X}}$ and $D_{\mathbb{Z}}$ may differ in dimension. In the case where all Δ_i are square, $D_{\mathbb{X}}(d)$ and $D_{\mathbb{Z}}(d)$ are identical. In any case, $D_{\mathbb{X}}(d)$ and $D_{\mathbb{Z}}(d)$ satisfy

$$D_{\mathbb{X}} \begin{bmatrix} \Delta_J & 0 \\ 0 & \Delta_K \end{bmatrix} = \begin{bmatrix} \Delta_J & 0 \\ 0 & \Delta_K \end{bmatrix} D_{\mathbb{Z}}$$

for all $d \in \mathbf{d}$.

These scaling matrices differ from the μ case in that the d_i scalings corresponding to the Δ_K block are negative. In the μ case, where K is empty, all scalings are positive.

Now define $I(\gamma)$ as

$$I(\gamma) \triangleq \operatorname{diag}(\gamma I_J, \gamma^{-1} I_K),$$

where I_J is the identity operator on $\mathbb{X}_J$ and I_K is the identity operator on $\mathbb{X}_K$. The upper bound can be stated as follows.

Theorem 6.8 *Let $d \in \mathbf{d}$, $D_{\mathbb{X}}(d)$, $D_{\mathbb{Z}}(d)$, and $I(\gamma)$ with $\gamma > 0$ be as above. Then*

$$M^* D_{\mathbb{Z}} M - I(\gamma)^2 D_{\mathbb{X}} < 0 \implies \gamma > \mu_g(M). \tag{6.8}$$

This gives a means of computing an upper bound; for a specified γ, searching for $D_{\mathbb{X}}(d)$ and $D_{\mathbb{Z}}(d)$ is an LMI problem. One can iterate on γ to find the smallest γ such that Equation (6.8) is satisfied for some d. The search for d and γ is typically simultaneous for greater efficiency.

Proof: Let $z = Mx$ and $I = J \bigcup K$. Consider the converse of Equation (6.8). If $\gamma \leq \mu_g(M)$ then, by Equation (6.5), $\exists x \neq 0$ such that

$$\begin{array}{rcll} \|x_j\|\,\gamma & \leq & \|z_j\| & \forall\, j \in J, \\ \|z_k\|\,\gamma & \leq & \|x_k\| & \forall\, k \in K. \end{array}$$

By squaring these conditions, we see that they are equivalent to the existence of $x \neq 0$ such that

$$\begin{array}{rcll} x_j^* x_j \gamma^2 & \leq & z_j^* z_j & \forall\, j \in J, \\ z_k^* z_k \gamma^2 & \leq & x_k^* x_k & \forall\, k \in K. \end{array} \tag{6.9}$$

Scaled versions of these equations also hold. Therefore Equation (6.9) is equivalent to the existence of $x \neq 0$ such that

$$\begin{array}{ll} x_j^* d_j x_j \gamma^2 \leq z_j^* d_j z_j & \forall\, j \in J, \\ x_k^* d_k x_k \gamma^{-2} \leq z_k^* d_k z_k & \forall\, k \in K, \end{array} \tag{6.10}$$

for all $d_j > 0$ and all $d_k < 0$. Equivalently, $\exists x \neq 0$ such that

$$\begin{array}{ll} x^*(M^* I_{j,I}^* d_j I_{j,I} M - \gamma^2 I_{j,I}^* d_j I_{j,I}) x \geq 0 & \forall\, j \in J, \\ x^*(M^* I_{k,I}^* d_k I_{k,I} M - \gamma^{-2} I_{k,I}^* d_k I_{k,I}) x \geq 0 & \forall\, k \in K, \end{array} \tag{6.11}$$

for all $d_j > 0$ and all $d_k < 0$. Here $I_{j,I}$, for example, is a partitioning of the appropriate identity operator. Note that

$$\sum_{i \in I} I_{i,I}^* d_i I_{i,I} \;=\; D_{\mathbb{Z}}(d) \text{ or } D_{\mathbb{X}}(d)$$

depending on whether I operates on $\mathbb{X}$ or on $\mathbb{Z}$.

The sum of the positive terms of Equation (6.11) is also positive. In other words, $\gamma \leq \mu_g(M)$ implies that, for all $d \in \mathbf{d}$ and corresponding $D_{\mathbb{X}}(d)$ and $D_{\mathbb{Z}}(d)$, $\exists x \neq 0$ such that

$$x^*(M^* D_{\mathbb{Z}} M - I(\gamma)^2 D_{\mathbb{X}}) x \geq 0.$$

The negation of this statement gives the following. If there exists a $d \in \mathbf{d}$ and a corresponding $D_{\mathbb{X}}(d)$ and $D_{\mathbb{Z}}(d)$ such that

$$M^* D_{\mathbb{Z}} M - I(\gamma)^2 D_{\mathbb{X}} < 0,$$

then $\gamma > \mu_g(M)$. ∎

When the Upper Bound is Exact

This section demonstrates the quality of the LMI upper bound for the μ_g problem of Section 6.2 by showing it is exact for three or fewer blocks in the $\mathbf{\Delta}$ structure in Equation (6.2), just as in the standard μ case. It might be reasonably expected, then, that the LMI upper bound for μ_g for more than three blocks is as accurate as the LMI upper bound for μ for more than three blocks. In the standard μ case with more than three blocks, the gap between the upper bound and μ is typically only a few percent and is very rarely more than ten percent. With more elaborate block structures, however, the gap can on occasion be quite large, so computation continues to be the subject of research.

The main theorem below states the result. The proof relies on the lemma presented here and on a theorem from Fradkov and Yakubovich in [17].

We begin with the main result.

Theorem 6.9 *Let M and $\mathbf{\Delta}$ be defined as in Section 6.2 with $\mathbf{\Delta}$ having three or fewer blocks. Let $d \in \mathbf{d}$, $D_{\mathbb{X}}(d)$, $D_{\mathbb{Z}}(d)$, and $I(\gamma)$ with $\gamma > 0$ be defined compatibly with M and $\mathbf{\Delta}$. Then*

$$\mu_g(M) < \gamma \iff \exists d \in \mathbf{d} : \; M^* D_{\mathbb{Z}}(d) M - I(\gamma)^2 D_{\mathbb{X}}(d) < 0.$$

In order to prove the theorem, we need the following lemma.

Lemma 6.10 *Let $\mathbb{Q}^+ = \{ z \mid z \in \mathbb{R}^3, \; z_l > 0 \text{ for } l \in \{1,2,3\} \}$, $L = \{ x \mid x \in \mathbb{X}, \; x \neq 0 \}$, and $I = J \bigcup K$. Let*

$$\begin{aligned} \varsigma_j(x) &= x^*(M^* I_{j,I}^* I_{j,I} M - \gamma^2 I_{j,I}^* I_{j,I}) x \quad \forall \, j \in J, \\ \varsigma_k(x) &= -x^*(M^* I_{k,I}^* I_{k,I} M - \gamma^{-2} I_{k,I}^* I_{k,I}) x \quad \forall \, k \in K, \end{aligned}$$

and $\varsigma(x) = (\ldots \varsigma_i(x) \ldots)$. Then

$$\begin{aligned} & \exists d \in \mathbf{d} : \; M^* D_{\mathbb{Z}}(d) M - I(\gamma)^2 D_{\mathbb{X}}(d) < 0 \\ \iff \; & \exists \text{ a separating hyperplane between } \varsigma(L) \text{ and } \mathbb{Q}^+. \end{aligned}$$

Proof: Given $d \in \mathbf{d}$, let $q = (\ldots d_j \ldots -d_k \ldots) \in \mathbb{Q}^+$, and let q_i be the elements of q. Then the following are equivalent.

- $M^* D_{\mathbb{Z}}(d) M - I(\gamma)^2 D_{\mathbb{X}}(d) \; < \; 0.$
- $x^*(M^* D_{\mathbb{Z}}(d) M - I(\gamma)^2 D_{\mathbb{X}}(d)) x \; < \; 0 \; \forall x \in L.$
- $\sum_{i \in I} q_i \varsigma_i(x) \; < \; 0 \; \forall x \in L.$
- $<q, \varsigma(x)> \; < \; 0 \; \forall x \in L.$

The separating hyperplane is defined by its normal, q. ∎

Note that the ς_i are Hermitian forms.

Proof of Theorem 6.9: If $\gamma > \mu_g$ so that there are no solutions, x, to Equation (6.5), then $\varsigma(x) \bigcap \mathbb{Q}^+ = \emptyset \ \forall \ x \in L$, and, by the S-procedure losslessness theorem for three or fewer Hermitian forms on a complex space (in [17]), there is a q such that $<q, \varsigma(x)> \in \mathbb{R}^- \ \forall x \in L$. Lemma 6.10 tells us this is equivalent to $M^* D_{\mathbb{Z}} M - I(\gamma)^2 D_{\mathbb{X}} < 0$. ■

Thus, in this case, $\mu_g(M) = \inf_{d \in \mathbf{d}} \{ \gamma \mid M^* D_{\mathbb{Z}}(d) M - I(\gamma)^2 D_{\mathbb{X}}(d) < 0 \}$.

6.5 The Lower Bound

The upper bound for the generalized μ problem allows us to invalidate models: there are no Δ, d, and n, with norms less than the inverse of the upper bound, that are consistent with Equation (6.1) represented by the block diagram in Figure 6.1. In contrast, the lower bound does not allow us to validate models: the lower bound finds a Δ, d, and n with norms less than or equal to the inverse of the lower bound. The difference is that the upper bound allows us to say "this data is inconsistent, hence the model is not representative of the true system" while the lower bound only allows us to say "this data is consistent, but more might not be;" we cannot conclude that the model is representative of the true system. The lower bound provides no hard information about the relevance of the model.

The value of the lower bound is that we typically want more than a simple yes or no answer; we want to fine tune our model by adjusting the sizes of the uncertainties and disturbances. Further, the lower bound can verify the quality of the upper bound: if they are close together, then both bounds are good, while if they are not close together, it may be that the data invalidates the model and the upper bound fails to indicate this. Under these circumstances it is difficult to tell what is consistent and what is a reasonable model. Without the lower bound, we have no indication of the quality of the upper bound.

This section develops a lower bound for the generalized μ problem. The lower bound algorithms of Chapters 3 and 5 rely upon alignment conditions (see [28] and [41]) derived from rather inaccessible results about the analyticity of eigenvalue decompositions ([19]). These results are not applicable to the generalized μ problem. Instead, we pursue a simpler perturbation analysis that contains the conditions for the standard lower bound as a special case.

The Form of Solutions

We used the form of the definition of μ_g in Equation (6.5) only to simplify the subsequent exposition. The following characterization of μ_g is necessary

to explain the lower bound computation for μ_g. We want solutions of the following form.

$$\begin{aligned} z &= Mx \\ v &= \Delta^* w \\ w &= M^* v \\ x &= \Delta z \\ \Delta_K &= \frac{b}{\|b\|}\sigma_K \frac{z_K^*}{\|z_K\|}, \end{aligned}$$

where b is a vector and σ is a scalar.

For regular blocks—those with the subscript J—this is just as for regular μ: Δ enforces the constraint between z_J and x_J. The w and v are found easily from the Δ. For the generalized blocks—those with the subscript K—the Δ_K above ensures $\|x_K\| = \|\Delta_K\| \, \|z_K\|$, so $\|\Delta_K\|^{-1}$ is the gain from x_K to z_K. Note that $\Delta_K = k x_K z_K^*$ for some $k \in \mathbb{R}$. Although $z \in \ker(I - M\Delta)$ as before, now the operator Δ depends on the kernel.

The Solution Occurs on the Boundary

An important feature of the generalized μ problem is that the maximum in Equation (6.5) occurs on the boundary. The proof of the following theorem treats the general case where there may be real blocks in the Δ structure. Note that $\mathbb{R} \subset \mathbb{C}$ has no interior. This is no different than the standard case in Chapter 3. The proof relies on a nondegeneracy assumption that the problem does not decouple into two independent problems so we may assume that none of the blocks of the vector z are zero.

Theorem 6.11 *Subject to the nondegeneracy assumption, a definition equivalent to Definition (6.5) of the positive function $\mu_g(M)$ for $M \in \mathrm{dom}(\mu_g)$ is*

$$\mu_g(M) \triangleq \max_{\|x\|=1} \left\{ \gamma \;\middle|\; \begin{array}{rcl} \|x_j\| \, \gamma &=& \|z_j\| \, , \forall j \in J \\ \|z_k\| \, \gamma &=& \|x_k\| \, , \forall k \in K \end{array} \right\}. \tag{6.12}$$

Proof: If $z_j = 0$ for some j, put Δ_j on the boundary. Thus without loss of generality, assume $z_j \neq 0$ for all j in J.

Assume $(I - M_{11}\Delta_1)^{-1}$ is well defined.

Assume there is a solution with Δ_2 not on the boundary:

$$\begin{aligned} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} &= M \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \end{aligned}$$

In the following, we construct a solution of the form

$$\begin{bmatrix} z_1 + \epsilon\zeta_1 \\ z_2 + \epsilon\zeta_2 \end{bmatrix} = M \begin{bmatrix} x_1 + \epsilon\chi_1 \\ x_2 \end{bmatrix}$$
$$\begin{bmatrix} x_1 + \epsilon\chi_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \widetilde{\Delta}_1 & 0 \\ 0 & \widetilde{\Delta}_2 \end{bmatrix} \begin{bmatrix} z_1 + \epsilon\zeta_1 \\ z_2 + \epsilon\zeta_2 \end{bmatrix}$$

with $\widetilde{\Delta}_1$ strictly better than Δ_1 (smaller gain J blocks and larger gain K blocks) and $\widetilde{\Delta}_2$ still not on the boundary, which contradicts the above assumption.

Away from $\det(I - M_{11}\Delta_1) = 0$, the three vectors

$$\begin{aligned} z_1 &= (I - M_{11}\Delta_1)^{-1} M_{12} x_2 \\ x_1 &= \Delta_1 (I - M_{11}\Delta_1)^{-1} M_{12} x_2 \\ z_2 &= (M_{22} + M_{21}\Delta_1 (I - M_{11}\Delta_1)^{-1} M_{12}) x_2 \end{aligned}$$

are all continuous in Δ_1, thus an order ϵ change in Δ_1 results in order ϵ changes in z_1, x_1, and z_2.

Next, multiply the standard blocks of Δ_1 by $1 - \frac{\epsilon}{\sigma}$ and the generalized blocks by $1 + \frac{\epsilon}{\sigma}$. With this new Δ_1 and the same x_2, the vectors z_1, x_1, and z_2 are replaced by $z_1 + \epsilon\zeta_1$, $x_1 + \epsilon\chi$, and $z_2 + \epsilon\zeta_2$. Since these new generalized blocks of Δ_1 are not in the standard form of Equation 6.12, we replace them blockwise as follows. For these blocks,

$$\begin{aligned} x + \epsilon\chi &= (1 + \frac{\epsilon}{\sigma})\Delta(z + \epsilon\zeta) \\ &= \frac{b}{\|b\|}(\sigma + \epsilon)\frac{z^*}{\|z\|}(z + \epsilon\zeta) \\ &= \frac{b}{\|b\|}(\sigma + \epsilon)\,\|z + \epsilon\zeta\|\,\frac{z^*(z + \epsilon\zeta)}{\|z\|\,\|z + \epsilon\zeta\|} \\ &= \frac{b}{\|b\|}(\sigma + \epsilon)\,\|z + \epsilon\zeta\|\,(1 + O\epsilon^2) \\ &= \left(\frac{be^{j\theta}}{\|b\|}\,\|1 + O\epsilon^2\|\,(\sigma + \epsilon)\frac{(z + \epsilon\zeta)^*}{\|z + \epsilon\zeta\|}\right)(z + \epsilon\zeta), \\ \widetilde{\Delta_1} &= \left(\frac{be^{j\theta}}{\|b\|}\,\|1 + O\epsilon^2\|\,(\sigma + \epsilon)\frac{(z + \epsilon\zeta)^*}{\|z + \epsilon\zeta\|}\right). \end{aligned}$$

Note that this $\widetilde{\Delta_1}$ is in our standard form and is a strict improvement over the old block of Δ_1. Note also that we have assumed $z \neq 0$.

Next, we must find a $\widetilde{\Delta}_2$ that satisfies

$$x_2 = \widetilde{\Delta}_2(z_2 + \epsilon\zeta_2) = \widetilde{\Delta}_2(z_2 + \epsilon M_{21}\chi)$$

while remaining internal. If the 2 block is a standard block (a J block), this is trivial and the size of $\widetilde{\Delta}_2$ only increases by $\frac{\|z_2\|}{\|z_2 + \epsilon M_{21}\chi\|}$, so the

change is of order ϵ. If the 2 block is a K block, the required change in size is similarly of order ϵ:

$$\begin{aligned} x_2 &= \frac{b}{\|b\|}\sigma\frac{z^*}{\|z\|}z \\ &= \left(\frac{b}{\|b\|}\frac{\|z\|\,\sigma}{\|z+\epsilon\zeta\|}\frac{(z+\epsilon\zeta)^*}{\|z+\epsilon\zeta\|}\right)(z+\epsilon\zeta), \\ \widetilde{\Delta}_2 &= \left(\frac{b}{\|b\|}\frac{\|z\|\,\sigma}{\|z+\epsilon\zeta\|}\frac{(z+\epsilon\zeta)^*}{\|z+\epsilon\zeta\|}\right). \end{aligned}$$

Thus we have an improved solution in the sense that the loop equations are satisfied with a Δ whose sizes are strictly better, so the original solution is not a local maximum. We conclude that interior points are not local maximums. ■

A Perturbation Analysis

Here we perturb a solution and ignore the higher order terms. Let $\widetilde{\alpha}$ be the additive perturbation to α so that the perturbed α is $\alpha+\widetilde{\alpha}$. The loop equations

$$\begin{aligned} z &= Mx \\ v &= \Delta^* w \\ w &= M^* v \\ x &= \Delta z \end{aligned}$$

with no perturbation to M imply

$$\begin{aligned} & x-\Delta z = 0 \\ \Longrightarrow\quad & \widetilde{x}-\widetilde{\Delta}z-\Delta\widetilde{z}=0 \\ \Longrightarrow\quad & w^*\widetilde{\Delta}z = w^*\widetilde{x}-w^*\Delta\widetilde{z} \\ & \qquad = v^*M\widetilde{x}-v^*M\widetilde{x} \\ & \qquad = 0; \end{aligned}$$

any perturbation which still satisfies the loop equations must also satisfy $w^*\widetilde{\Delta}z=0$ to first order.

A Family of Perturbations

Since local maximums occur on the boundary, we are interested in perturbations to Δ from $\partial\mathbf{B}_{\mathbf{\Delta}}$. Although this need not parametrize all such perturbations, we shall consider perturbations of the form

$$\widetilde{\Delta} = G\Delta.$$

It is easily verified that if $G+G^* < 0$ and G is small enough then $\overline{\sigma}(I+G) < 1$ while if $G+G^* > 0$ then $\underline{\sigma}(I+G) > 1$. Denote the sets of such G as G_m and G_p respectively:

$$\begin{aligned} G_m &= \{\, G \mid G+G^* < 0,\ \overline{\sigma}(I+G) < 1 \,\} \\ G_p &= \{\, G \mid G+G^* > 0 \,\}. \end{aligned}$$

We define the set $\mathbb{G}$, which parametrizes a set of allowable perturbations to Δ, blockwise as follows.

Full blocks: $\mathbb{G}_{\mathbf{C}} = \{\, G \mid G \in G_m \bigcap \mathbf{\Delta_C} \,\}$

$\delta_c I$ blocks: $\mathbb{G}_{\mathbf{c}} = \{\, g_c I \mid g_c I \in G_m \bigcap \mathbf{\Delta_c} \,\}$

$\delta_r I$ blocks: on the boundary relative to $\mathbb{R}$, $\mathbb{G}_{\mathbf{r}} = \{\, g_r I \mid g_r I \in G_m \bigcap \mathbf{\Delta_r} \,\}$, and in the interior relative to $\mathbb{R}$, $\mathbb{G}_{\mathbf{r}} = \{\, g_r I \mid g_r I \in \mathbf{\Delta_r} \,\}$

Δ_K blocks: $\mathbb{G}_k = \{\, G \mid G \in G_p \bigcap \mathbf{\Delta}_k \,\}$

For the Δ_K blocks note that

$$\begin{aligned} & (I+G)\frac{b}{\|b\|}\sigma\frac{z^*}{\|z\|} \\ = \ & \frac{(I+G)b}{\|(I+G)b\|}\left(\frac{\|(I+G)b\|}{\|b\|}\sigma\right)\frac{z^*}{\|z\|} \end{aligned}$$

and that $\left(\frac{\|(I+G)b\|}{\|b\|}\sigma\right) > \sigma$. Also recall that $\overline{\sigma}(AB) \le \overline{\sigma}(A)\overline{\sigma}(B)$ and that $\|Ab\| \ge \underline{\sigma}(A)\,\|b\|$. If we further constrain $\mathbb{G}$ to have small enough norm, then it is easy to see that $G \in \mathbb{G}$ implies that $(I+G)\Delta$ is an improvement over Δ in that it provides a larger value in the maximization problem if $\det(I-(I+G)\Delta M) = 0$.

The Alignment Conditions

A solution to the equations is not a local maximum of Equation (6.12) if there is a $G \in \mathbb{G}$ (which corresponds to a better Δ) that satisfies $w^*\widetilde{\Delta}z = w^*Gx = 0$. Thus a necessary condition for a solution to be a local maximum is that $w^*\mathbb{G}x \bigcap 0 = \emptyset$. We satisfy this condition if all the $w_i^*\mathbb{G}_i x_i \subset \mathbb{C}^-$. (Other halfplanes correspond to other eigenvector normalizations.)

In the following presentation of the consequent alignment conditions for each type of block, the subscripts are omitted.

Full blocks: $w = \alpha x,\ \alpha \in \mathbb{R}^+ \implies w^*\mathbb{G}x = \alpha x^*\mathbb{G}x \subset \mathbb{C}^-$. If $w \neq \alpha x$ (and $x \neq 0 \neq w$), then there is a z such that $z^*x = 0$, $z^*w \neq 0$, and $G = \epsilon(-zz^* - w^{\perp}w^{\perp *}) \in \mathbb{G}$, but $w^*Gx = 0$. Thus we require $w = \alpha x$.

$\delta_c I$ **blocks:** $w^*x \in \mathbb{R}^+ \implies w^*\mathbb{G}x \subset \mathbb{C}^-$. If $w^*x \notin \mathbb{R}^+$, then there is a $G \in \mathbb{G}$ such that $w^*Gx \in \mathbb{C}^+$. Thus we require $w^*x \in \mathbb{R}^+$.

$\delta_r I$ **blocks:** $w^*x \in \mathbb{C}^+ \implies w^*\mathbb{G}x \subset \mathbb{C}^-$. If $w^*x \notin \mathbb{C}^+$ and this block is on the boundary, then there is a $G \in \mathbb{G}$ such that $w^*Gx \in \mathbb{C}^+$. Thus we require $w^*x \in \mathbb{C}^+$ for boundary blocks. If the block is internal, however, we need $\mathrm{Re}(w^*x) = 0$. Otherwise there is a $G \in \mathbb{G}$ such that $w^*Gx \in \mathbb{C}^+$. When $\mathrm{Re}(w^*x) = 0$, $w^*\mathbb{G}x = 0 \notin \mathbb{C}^-$, but this is not an issue because it does not allow us to improve any of the boundary Δ; we might be able to move the internal Δ, but this does not help us find a better solution. There is also the curious case of the perturbation that is internal in the sense that $\mathrm{Re}(w^*x) = 0$, but also happens to be the size of the perturbations on the boundary.

Δ_J **blocks:** $w = \alpha x$, $\alpha \in \mathbb{R}^- \implies w^*\mathbb{G}x = \alpha x^*\mathbb{G}x \subset \mathbb{C}^-$. If $w \neq \alpha x$ (and $x \neq 0 \neq w$), then there is a z such that $z^*x = 0$, $z^*w \neq 0$, and $G = \epsilon(zz^* + w^{\perp}w^{\perp *}) \in \mathbb{G}$, but $w^*Gx = 0$. Thus we require $w = \alpha x$.

These alignment conditions, satisfied at all local maximums, imply certain conditions on the Δ at alignment, as follows.

Full blocks: $w = \alpha x, \alpha \in \mathbb{R}^+, x = \Delta z$ together imply that Δ is wz^* normalized plus something orthogonal.

$\delta_c I$ **blocks:** $w^*x \in \mathbb{R}^+, x = \Delta z$ imply that Δ is the identity times z^*w normalized.

$\delta_r I$ **blocks:** $\mathrm{Re}(w^*z) = 0$ puts no constraint on Δ, while $w^*x \in \mathbb{C}^+, x = \Delta z$ imply that Δ is the identity times $\mathrm{sgn}(\mathrm{Re}(w^*z))$, normalized.

Δ_K **blocks:** $w = \alpha x, \alpha \in \mathbb{R}^-, x = \Delta z$ together imply that Δ is $-wz^*$ normalized.

The normalizations are determined by the fact that the solution occurs on the boundary. Note that we have not assumed anywhere that the matrix M is not rank one.

An Equivalent Rank One Problem

For each local maximum of a generalized μ problem there is a corresponding rank one generalized μ problem. Both problems satisfy the same loop equations and alignment conditions at the local maximum, and the local maximum is also the global maximum of the rank one problem. Let M denote the original problem, and M_r denote the corresponding rank one problem. Then $z = M_r x$ and $w = M_r^* v$ imply $M_r = \alpha z w^*$, and $z = \frac{zw^*}{w^*x}x$ further implies that $M_r = \frac{zw^*}{w^*x}$.

Thus we need z, w, and x (or M, w, and x) to construct M_r from the original problem. It is routine to verify that the remainder of the alignment conditions are satisfied with this rank one matrix.

For the generalized μ power algorithm, however, we need a different rank one matrix. At a local maximum the alignment conditions are satisfied and $w_K = \frac{-x_K\|v_K\|}{\beta\|x_K\|}$, where β is the size of the local maximum. This follows directly from $w_K = \alpha x_K$, $\alpha \in \mathbb{R}^-$, and $v_K = \Delta_K^* w_K$. Similarly, the alignment conditions imply that $z_K = \frac{-v_K\|x_K\|}{\beta\|v_K\|}$.

When we replace w_K and z_K, the resulting rank one matrix is

$$M_r = \frac{\begin{bmatrix} z_J \\ \frac{-v_K\|x_K\|}{\|v_K\|\beta} \end{bmatrix} \begin{bmatrix} w_J \\ \frac{-x_K\|v_K\|}{\|x_K\|\beta} \end{bmatrix}^*}{\begin{bmatrix} w_J \\ \frac{-x_K\|v_K\|}{\|x_K\|\beta} \end{bmatrix}^* \begin{bmatrix} x_J \\ x_K \end{bmatrix}}. \tag{6.13}$$

If we use this rank one matrix in a power iteration for the generalized μ lower bound, then the flow of information in the $\widehat{M}$ problem is preserved. Equivalently, the use of this rank one problem encompasses the ROA on the $\widehat{M}$ problem as a special case.

The Rank One Solution

Here we consider the calculation of generalized μ for the rank one matrix zw^*. Note that this is different than the rank one matrix in Equation (6.13) and that generalized μ is not linear in scalar multiplication (typically, $\mu(\alpha M) \neq \alpha\mu(M)$, $\alpha \in \mathbb{R}^+$). The solution to the rank one generalized μ satisfies the following equation.

$$\begin{aligned}
0 &= \det\left(I - zw^* \begin{bmatrix} \Delta_J & 0 \\ 0 & \Delta_K \end{bmatrix}\right) \\
&= 1 - w^* \begin{bmatrix} \Delta_J & 0 \\ 0 & \Delta_K \end{bmatrix} z \\
&= 1 - \sum w_i^* \Delta_i z_i \\
&= 1 - \left(\mu^{-1} e^{j\theta_c} L_c + \mu e^{j\theta_K} L_K + \mu^{-1} \sum_r \delta_r w_r^* z_r\right), \\
L_c &= \left\| \sum_r w_c^* \Delta_c z_c \right\| \\
L_K &= \left\| \sum_K w_K^* \Delta_K z_K \right\|
\end{aligned}$$

where L_c and L_K are chosen so that L_c is as large as possible, as in the standard case in Section 2.5, and so that L_K is as small as possible. If there are two or

more K blocks, then L_c can be taken to be zero, and the choice of Δ_K is not unique.

Scaling the above equation by μ, we wish to satisfy

$$\mu = e^{j\theta_c} L_c + \mu^2 e^{j\theta_K} L_K + \sum_r \delta_r w_r^* z_r$$

with μ as large as possible. (A larger value for μ makes $\mu^2 e^{j\theta_K} L_K$ too large for a solution.) The L_c and L_K combine optimally as $(\mu^2 L_K - L_c)e^{j\theta}$.

If L_K is zero, the solution is just as in the standard case. Otherwise, the contribution of the real blocks is better the more negative it is and the larger the complex part. (Remember that we wish to maximize μ and that a μ that is too large makes $(\mu^2 L_K - L_c)e^{j\theta}$ too large.) The graphical solution is similar to the solution in Section 2.5.

A Power Iteration

At equilibrium, a power iteration must be consistent with the loop equations

$$\begin{aligned} z &= Mx \\ v &= \Delta^* w \\ w &= M^* v \\ x &= \Delta z, \end{aligned}$$

and Δ and the vectors must be consistent with the alignment conditions.

To define a particular power algorithm, we must select the order and way we use the equations for updates as well as how and when we make Δ consistent with the alignment conditions.

Key to a successful power iteration is the direction of flow. To allow the largest solution to dominate, we wish to use the power steps as follows. Use the equation

$$\begin{bmatrix} z_J \\ z_K \end{bmatrix} = M \begin{bmatrix} x_J \\ x_K \end{bmatrix}$$

to find a new z_J and x_K, and the equation

$$\begin{bmatrix} w_J \\ w_K \end{bmatrix} = M^* \begin{bmatrix} v_J \\ v_K \end{bmatrix}$$

to find a new w_J and v_K. With these choices, the flow of information in the $\widehat{M}$ problem is preserved. Analogous to the algorithms of Chapter 3, we use these z_J, x_K, w_J, and v_K to update Δ according to the alignment conditions.

Thus a natural generalization of power algorithms for the largest eigenvalue and largest singular value is

- initialize

▶ get new z_J, x_K, and β from $z = Mx$

- select Δ from the rank one solution for the matrix in Equation (6.13)
- get new v_J and w_K from $v = \Delta^* w$
- get new w_J, v_K, and β from $w = M^* v$
- select Δ from the rank one solution for the matrix in Equation (6.13)
- get new x_J and z_K from $x = \Delta z$
- if converged then done, else go to ▶

Many of the refinements and techniques of Chapter 3 are applicable to the computation of generalized μ, as are techniques of Chapter 5. These topics are not covered here, however.

Chapter 7

Implicit Formulations: A Unifying Framework

In previous chapters we have seen that μ solves many robustness analysis problems and that μ can be computed effectively despite poor worst-case computation. We have also seen that the definition and computation of μ can be extended to salve a related but different problem, the model validation problem. This chapter discusses extensions to the μ paradigm in a more general way.

The main motivation behind this chapter is the belief that control theory needs to play a broader role in technology; in fact, of the multiple technological problems involving mathematical tools of the dynamical systems theory, only a small fraction reduce to the design of a feedback system for a well defined plant. A broad class of problems in modeling, system identification, system design, simulation, and optimization are addressed with similar mathematical tools; a natural objective is therefore the development of a more unified theory, in which a common language of mathematical and computational machinery is used to perform the previous range of activities. Although this goal may seem ambitious, this chapter documents some progress in this direction by exhibiting a framework that captures as special cases robustness analysis and system identification.

The connection between these two problems is that they both are special cases of the following analysis question:

> **Q**: *Given a mathematical description in terms of equations involving uncertainty, are there values of the uncertainty in a given class such that the equations have a solution?*

For the case of robustness analysis, this question is specialized as follows. The mathematical description is a dynamical system with uncertainty, for example parametric or dynamic uncertainty. Loosely speaking, to solve a robust stability problem, (such problems encompass many robust performance prob-

lems,) is to test whether there are uncertainty values for which loop equations admit nontrivial solutions, and is therefore a special case of answering **Q**.

Most activity in robustness analysis has focused on obtaining computable answers to this question for a rich variety of uncertainty descriptions, which can usually be fit into the LFT and μ paradigms. A class of perturbation structures which lead to particularly tractable computation are those corresponding to Integral Quadratic Constraints (IQCs, see Megretski and Treil, [22]) on signals: in this case the problem reduces to a convex feasibility problem, the solution of an LMI. For tighter descriptions of uncertainty such as real parameters, worst-case computation is provably NP hard (see Chapter 2).

A recent development in robustness analysis (see Paganini et al., [30]), exploited in this chapter, is that a larger class of analysis problems, involving uncertain systems and an arbitrary number of IQCs, can be formulated and solved in implicit form. This is related to the behavioral paradigm for system theory (see Willems, [38], and references therein) and is described in Section 7.1.

A large research field under the umbrella of control theory is the area of system identification, which obtains dynamical models from experimental data, and the related model validation problem, which checks consistency of a model with data. Mathematically, this area has relied on extending methods of statistics, mainly time series analysis. A standard reference is Ljung ([21]). Since these models are typically stochastic, it has been difficult to reconcile this theory with robust control, which relies on deterministic descriptions.

The main argument to base this unification is the recognition that model validation/ID problems are special cases of **Q**: given a model and experimental data, are values of the uncertainty (parameters, disturbances, etc.) that solve the equations? This is, strictly speaking, a model validation question; system identification involves additionally finding the parameter values, but this is often no harder—existence of a solution is usually shown by finding a solution.

It is shown in Section 7.2 that the implicit LFT formulation over constant matrices provides a natural framework in which to cast a large class of model validation/ID problems, and therefore is a unifying paradigm.

In addition to providing a conceptual framework in which to relate robustness analysis and identification, the formulation pursued here suggests that computational tools developed to deal with the implicit robustness analysis formulation could be applied to system identification. These tools are discussed in Section 7.4.

Finally, Section 7.5 contains an example where the conventional least squares identification problem is reviewed from this perspective.

7.1 Robustness Analysis in Implicit Form

We begin by considering an uncertain LFT system in implicit form, depicted in Figure 7.1. Note that the output is constrained to be zero.

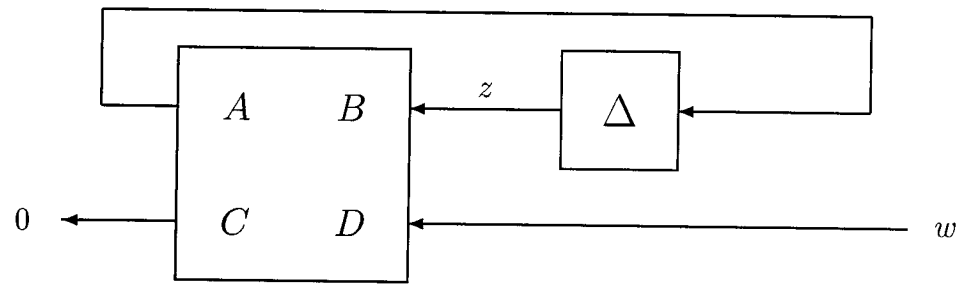


Figure 7.1: An Implicit LFT system.

In this formulation, $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ is in principle an arbitrary map, and Δ is a structured uncertainty operator, described below. The manifest variables w (the signals of interest) are described implicitly by the above equations, since $(\Delta \star M)w = 0$, where it is assumed that $(I - A\Delta)^{-1}$ is well defined. Note that there is no partition of the manifest variables into inputs and outputs. In this respect, this formulation is consistent with the behavioral framework for system theory, introduced by Willems ([38]). D'Andrea and Paganini ([10, 9]) provide motivation and introductory material on these representations.

An internal description of the same system in terms of the generalized state z is

$$\varphi(\Delta, M) \begin{bmatrix} z \\ w \end{bmatrix} = 0, \tag{7.1}$$

$$\varphi(\Delta, M) = \begin{bmatrix} I - \Delta A & -\Delta B \\ C & D \end{bmatrix}. \tag{7.2}$$

As in standard robust control, Δ varies in $\mathbf{\Delta}$, which is the class of structured uncertainty operators of the form

$$\Delta = \mathrm{diag}\,[\delta_1 I_{r_1}, \ldots, \delta_L I_{r_L}, \Delta_{L+1}, \ldots, \Delta_{L+F}] \tag{7.3}$$

where the blocks in Δ represent real parameters or dynamic (linear time-invariant (LTI), linear time-varying, or nonlinear) perturbations. Usually these perturbations are restricted to $\mathbf{B}_{\mathbf{\Delta}} = \{\, \Delta \mid \|\Delta\| \leq 1 \,\}$ in some operator norm.

In addition to allowing the representation of any standard input-output uncertain LFT system, the implicit representation allows the formulation of

over constrained problems, which have more equations than free variables w. A special case of this is considered by Paganini et al. in [30], where it is shown that a finite number of IQCs on w can be given a kernel representation as in Figure 7.1. An example of the use of such additional constraints is the case of whiteness constraints to pose a robust $\mathcal{H}_2$ performance problem (see [30]). Therefore a richer class of robust performance analysis problems can be formulated in this paradigm and converted to a robust stability test in the sense described below. The following definition, and much of the remainder of this section, is from Paganini and Doyle ([31]).

Definition 7.1 *Consider the implicit system in Equation (7.1), where A, B, C, D, and Δ are linear operators in a vector valued $\mathcal{L}_2$ space. The system has robust $\mathcal{L}_2$ stability if for each $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$,*

$$\inf_{z,w} \left\{ \left\| \varphi(\Delta, M) \begin{bmatrix} z \\ w \end{bmatrix} \right\| \;\middle|\; z, w \in \mathcal{L}_2, \left\| \begin{bmatrix} z \\ w \end{bmatrix} \right\| = 1 \right\} > 0. \qquad (7.4)$$

According to the definition (where $\mathcal{L}_2$ could be replaced by any Banach space of signals), $\mathcal{L}_2$ stability implies that there are no nontrivial signals satisfying Equation (7.1) for any $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$. This is a condition of the type expressed in the analysis question **Q** of the introduction; $\mathcal{L}_2$ stability is equivalent to a negative answer to **Q**.

Note that in this infinite dimensional case, the $\mathcal{L}_2$ stability definition has the technical requirement that nontrivial approximate solutions z and w to Equation (7.1) cannot exist with an arbitrarily small amount of equation error. This is equivalent to saying that apart from being injective, the operator $\varphi(\Delta, M)$ has a left inverse which is a bounded operator in $\mathcal{L}_2$.

The following proposition from [31] reduces the representation to a simpler form.

Proposition 7.2 *The implicit system in Equation (7.1) has robust $\mathcal{L}_2$ stability if and only if both (i) and (ii) hold.*

(i) D has a bounded left inverse L.

(ii) The implicit system $\begin{bmatrix} I - \Delta\widehat{A} \\ \widehat{C} \end{bmatrix} z = 0$ *has robust $\mathcal{L}_2$ stability,*

where $\widehat{A} = A - BLC$, $\widehat{C} = C - DLC$.

Condition (i) is a nominal stability condition which, if not satisfied, says there are nontrivial solutions to $Dw = 0$ and the system is not stable at $\Delta = 0$. In (ii), where from now on we replace $\widehat{A}$ and $\widehat{C}$ by A and C, the left invertibility condition on $\begin{bmatrix} I - \Delta A \\ C \end{bmatrix}$ for each $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$ resembles a PBH test

for detectability of the pair A, C. Tests for robust $\mathcal{L}_2$ stability of this implicit representation are given in [31].

In many important cases, the robustness analysis can be conducted in a constant matrix representation, which is essential if computational tests are to be derived. These have the form

$$\begin{bmatrix} I - \Delta A \\ C \end{bmatrix} z = 0 \tag{7.5}$$

where A and C are constant matrices and $\Delta \in \mathbf{\Delta} \subset \mathbb{C}^{n \times n}$.

One such case is that of state space implicit descriptions in discrete time (see Paganini and D'Andrea, [9, 30]). Assume that M in Figure 7.1 is a finite dimensional LTI system. By writing a state space realization of this system we obtain a new implicit description, where M is replaced by a constant matrix, the delta structure Δ is replaced by an augmented structure $\Delta_S = \text{diag}[\lambda I, \Delta]$, where λ is the delay operator. It is shown by Paganini et al. in [30] that under mild assumptions the analysis can be reduced to a constant matrix problem such as Equation (7.5).

Another constant matrix case is when M and Δ are time-invariant. Then the robust stability test reduces to (see [31])

$$\ker \begin{bmatrix} I - \Delta_0 A(j\omega) \\ C(j\omega) \end{bmatrix} = 0 \quad \forall \Delta_0 \in \mathbf{B}_{\mathbf{\Delta}_0} \ \ \forall \omega, \tag{7.6}$$

where $A(j\omega)$ and $C(j\omega)$ are the frequency responses of the LTI systems A and C, and Δ_0 is a constant complex perturbation with the same structure as the original Δ. This is a constant matrix test at each frequency.

These conditions are reminiscent of μ, which corresponds to the case where $C = 0$. We give the following definition from [31]:

Definition 7.3 *The structured singular value of the matrix A with respect to the structure $\mathbf{\Delta} \subset \mathbb{C}^{n \times n}$, subject to the implicit constraints C is defined as*

$$\mu_{\mathbf{\Delta},C}(A) \triangleq 0 \quad \textit{if} \quad \ker \begin{bmatrix} I - \Delta A \\ C \end{bmatrix} = 0 \ \forall \Delta \in \mathbf{\Delta}, \ \textit{otherwise}$$

$$\mu_{\mathbf{\Delta},C}(A) \triangleq \left(\min \left\{ \bar{\sigma}(\Delta) \;\middle|\; \Delta \in \mathbf{\Delta}, \ker \begin{bmatrix} I - \Delta A \\ C \end{bmatrix} \neq 0 \right\} \right)^{-1}. \tag{7.7}$$

Note that the structure $\mathbf{\Delta}$ could be specified to be real, or to be mixed with real and complex blocks.

When we formulate the special case of the analysis question $\mathbf{Q}$ for these constant matrix problems as follows

$\mathbf{Q}_\mu$: *Given Equations (7.5) where A, C are constant matrices and $\mathbf{B}_\Delta \subset \mathbb{C}^{n\times n}$ as in Equation (7.3), is there a $\Delta \in \mathbf{B}_\Delta$ such that the Equations (7.5) admit nontrivial solutions?*

We see that $\mathbf{Q}_\mu$ can be restated as the test Is $\mu_{\mathbf{\Delta},C}(A) \geq 1$? A negative answer to $\mathbf{Q}_\mu$ is equivalent to robust stability.

Thus the class of analysis questions which can be stated in the form $\mathbf{Q}_\mu$ can all be answered if we can compute the quantity $\mu_{\mathbf{\Delta},C}(A)$. As in the standard case ($C = 0$), exact computation of $\mu_{\mathbf{\Delta},C}(A)$ is difficult in general. In Section 7.4 we consider upper and lower bounds for this problem and their computation.

7.2 Model Validation and Identification

The basic element of any quantitative approach to scientific and technological problems is a mathematical model. Typically, the model is obtained using some combination of first principles analysis and identification from experimental data. In general, one might start with a model with some a priori structure, perhaps using some first principles knowledge, that includes some description of the a priori uncertainty (parameters, disturbances, etc.). After performing an experiment, one is faced with the mathematical problem of finding values of the uncertainty that agree with our data.

An extensive field of research pursues the answer to this problem. For the case of dynamical models, a standard reference is Ljung ([21]). A canonical example of the methods of standard system identification is the fitting of a parametric model by using prediction error methods (PEMs) described next. Assume the following model structure,

$$y = G(\lambda, \theta)u + H(\lambda, \theta)d \tag{7.8}$$

where λ is the shift operator, θ is a vector of parameters, G and H are discrete time systems, and d is a disturbance. Given data u and y, these methods attempt to find values of θ and d which agree with the data. Since many solutions may exist, the standard approach is to search for the solution which minimizes some norm of d.

A related problem is model validation: given a model and data, is the model consistent with the data? In the PEM example, it may be that values of θ have already been chosen, and we wish to determine whether the model is consistent with a set of data with a plausible (e.g. small enough) instance of d. The model validation problem clearly fits into the analysis question $\mathbf{Q}$ stated in the introduction: given the equations, the parameters and the data, that must check whether there are values of the uncertainty (in this case d, satisfying some constraints, e.g. $\|d\| \leq \gamma$) which verify the equations.

The identification problem is different in that the parameters θ are also unknown. Consequently in **Q** we inquire whether there are values of d and θ verifying the equations. To minimize $\|d\|$, we can ask **Q** for various sizes of d. Additionally, in conventional identification in general and in the PEM problem in particular, we want to find the values of θ that give the affirmative answer to **Q**. Traditionally, the model is said to be identified when a fixed value of θ is chosen.

More generally, we might prefer a final identified model where, for example, some parametric uncertainty is left. This entails a somewhat expanded notion of identification. Other choices will arise as we include other sources of uncertainty. Also, we may have multiple experiments, where some of the uncertainty is fixed to have a common value across experiments, and other uncertainty (e.g., noise, parametric variations due to changes in experimental conditions, unmodeled dynamics) is allowed to vary from one experiment to another.

Therefore, a general methodology for model validation and system identification (henceforth denoted MV/ID) should provide computational tools for answering the above general question **Q** for rich uncertainty structures, including noise, unmodeled dynamics, and parameters.

MV/ID in an Implicit LFT Setting

We now consider a general class of MV/ID problems which are described in terms of LFTs. In Chapter 6 we see how this type of problem is strongly related to robustness analysis machinery. In this section it is shown how these problems are naturally formulated as implicit LFT analysis problems considered in Section 7.1.

Figure 7.2 shows a generic input-output MV/ID structure, where we assume all the elements in the diagram are constant vectors and matrices. In Section 7.3 we briefly explain how dynamical models based on finite time histories may be converted to this form. The vector of unknown inputs (disturbances) d is constrained by $\|d\| \leq 1$; Δ is in $\mathbf{B}_{\mathbf{\Delta}}$; u and y are the measured inputs and outputs. The MV/ID problem is, again, to find values of Δ and d consistent with Figure 7.2. The problem is assumed to be well posed in the sense that there are no nontrivial solutions with $d = 0$ when y and u are 0.

This LFT structure captures a rich variety of linear identification problems. As a simple example, consider the standard linear regression problem

$$y = M\theta + d \tag{7.9}$$

where M and y are known, θ is a vector of unknown parameters and d is a vector of unknown errors. These equations are of the form of Figure 7.2, with

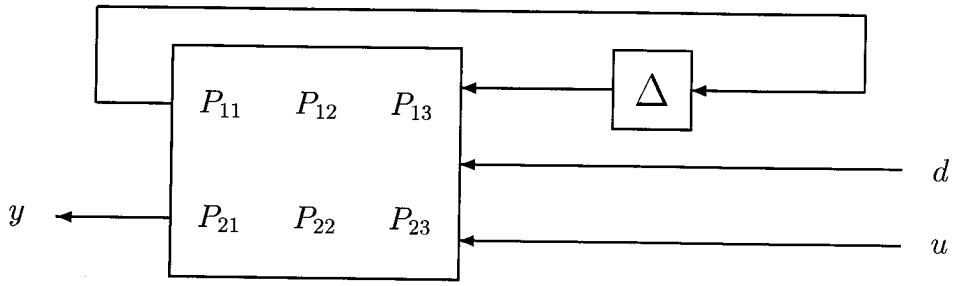


Figure 7.2: A standard input-output MV/ID setup.

$\Delta = \theta$, $u = 1$, and

$$P = \begin{bmatrix} 0 & 0 & 1 \\ M & I & 0 \end{bmatrix}.$$

In Figure 7.3 the equations of Figure 7.2 are represented in implicit form, with u and y combined into the vector v which includes all the known data:

$$v = \begin{bmatrix} u \\ y \end{bmatrix}, \quad \widetilde{P}_{13} = \begin{bmatrix} P_{13} & 0 \end{bmatrix}, \quad \widetilde{P}_{23} = \begin{bmatrix} P_{23} & -I \end{bmatrix}. \tag{7.10}$$

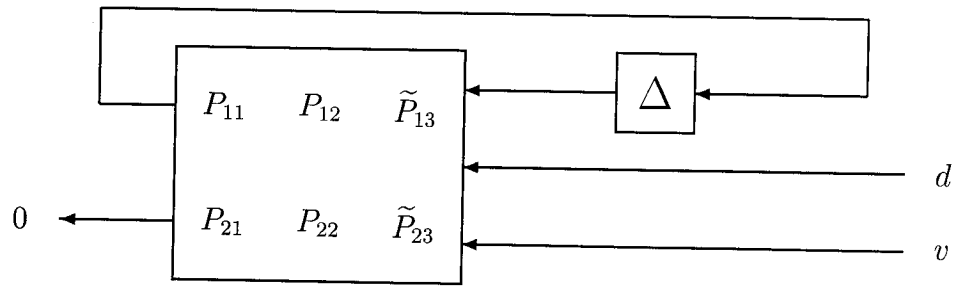


Figure 7.3: A standard MV/ID setup in implicit form.

The distinction between input and output in Figure 7.2 has been eliminated in Equation (7.10). Note that the input-output partition has been eliminated from the model. In fact, it could well be that we wish to validate some model based on observations of a system where this distinction is not available. Then we would arrive directly at Figure 7.3. For example, in the linear regression above, the input is an artifice of the representation.

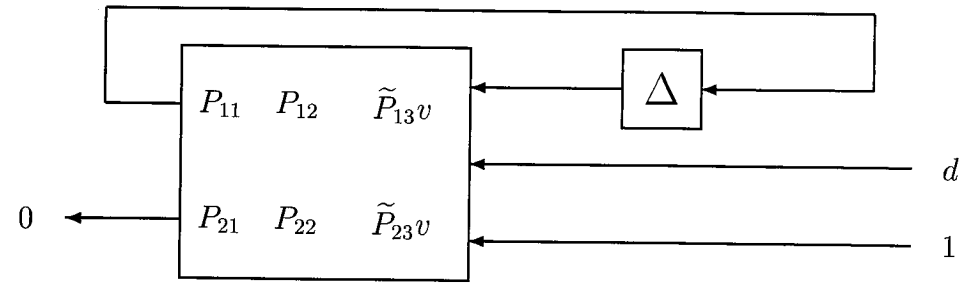


Figure 7.4: The MV/ID setup with data inside the matrix.

We can now incorporate the data v into the matrix by considering a fictitious scalar input of value 1. This results in Figure 7.4.

The representation has up to now two different sources of uncertainty: Δ and d. This distinction disappears and the constraint $\|d\| \leq 1$ is included in the problem when we introduce the uncertainty block $\Delta_d = d$, $\|\Delta_d\| \leq 1$, and write $d = \Delta_d 1$. This is shown in Figure 7.5. Note that Δ_d is just a new name for d that reflects its location in the diagram.

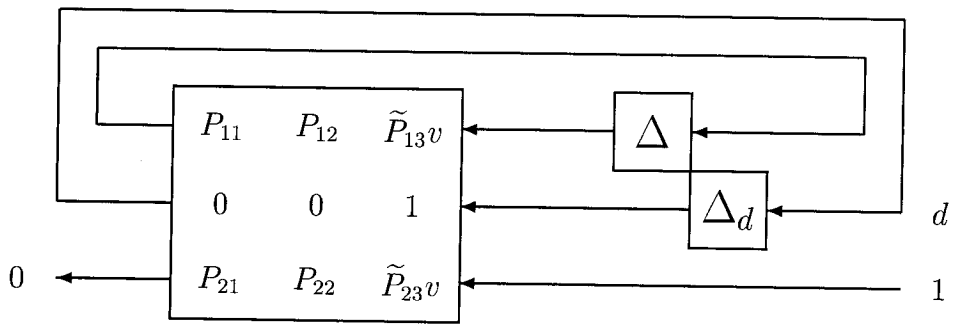


Figure 7.5: MV/ID as an analysis problem.

The MV/ID question is now reduced to the question Are there nontrivial signals satisfying the implicit equations of Figure 7.5 for $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$? This is an instance of a constant matrix problem of Figure 7.1. The fact that one signal is constrained to be 1 is irrelevant since everything can be normalized by linearity if the problem is well posed.

Unless the nominal model satisfies the data, $\widetilde{P}_{23}v$ is left invertible (it is a nonzero column). Then Proposition 7.2 tells us the MV/ID analysis can

be reduced to the question Is $\ker \begin{bmatrix} I - \Delta A \\ C \end{bmatrix} \neq 0$ for some $\Delta \in \mathbf{B}_{\mathbf{\Delta}}$? This question is precisely $\mathbf{Q}_\mu$ posed in Section 7.1, and entails the same computation as in the robustness analysis problem. A positive answer to $\mathbf{Q}_\mu$ is equivalent to establishing that the model and the data are consistent.

7.3 Time Domain Data and Dynamical Models

Section 7.2 was based on static representations for models and data, with no explicit time variable. System Identification, however, deals with dynamical models and observations across time. Since the time horizon is finite, the dynamic MV/ID problem is always represented by a finite, albeit large, number of equations that can be represented in terms of constant matrices. To illustrate, consider an autoregressive model

$$A(\lambda)y = d, \quad A(\lambda) = 1 + a_1\lambda + \ldots + a_m\lambda^m \tag{7.11}$$

with y known and the a_i and d unknown. Over a finite horizon $[0, N]$, the equations can be written as

$$\begin{bmatrix} y(0) & \cdot & \cdots & \cdot \\ y(1) & y(0) & \cdots & \cdot \\ y(2) & y(1) & \cdots & \cdot \\ \vdots & \vdots & \vdots & \vdots \\ y(N) & \cdot & \cdots & y(N-m) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} d(0) \\ d(1) \\ d(2) \\ \vdots \\ d(N) \end{bmatrix}, \tag{7.12}$$

which has the form $y = M\theta + d$, a special case of Equation (7.9). Thus this problem has a constant matrix LFT representation.

This example is particularly simple because it reduces to a linear regression. More generally the disturbance d may enter the equations through some dynamics which contain uncertainty. Such is the case of the ARMA model $A(\lambda)y = B(\lambda)d$. Here, the unknown parameters in $B(\lambda)$ are convolved in time with the unknown disturbance. A constant matrix representation of these equations features repetition of the parameters in $B(\lambda)$, resulting in large δI blocks in Δ and consequently more difficult computation. Progress has been made recently in computation with large problems involving such repetition. See Tierno and Doyle ([36]) for more details on the general formulation of finite time horizon analysis problems and the related computation.

7.4 Computation for Implicit Analysis

In this section we address the issue of obtaining computational tools to answer the general question $\mathbf{Q}_\mu$, of which both constant matrix robustness analysis and MV/ID are special cases. The idea is to extend the upper and lower bounds for standard μ analysis ($C = 0$ in Definition 7.3). Except in special cases, the upper and lower bounds must be close to know that either is close to $\mu_{\mathbf{\Delta},C}(A)$.

If the lower bound is ≥ 1, then we have a solution to the Equations (7.5) and the system is not robustly stable. Thus a lower bound ≥ 1 is a sufficient condition for the absence of robust stability. For MV/ID, a lower bound ≥ 1 is a sufficient condition for the existence of a solution to the problem.

On the other hand, an upper bound < 1 guarantees the absence of a solution and is thus a sufficient condition for robust stability and a sufficient condition for the absence of a solution to the MV/ID problem.

Upper Bounds

The upper bounds to the structured singular value extend in a natural way to our implicit formulation. We first define the set $\mathbf{X}$ of positive scaling matrices which commute with the elements in $\mathbf{\Delta}$. They have the structure

$$X = \mathrm{diag}\,[X_1, \ldots, X_L, x_{L+1}I, \ldots, x_{L+F}I]\,. \tag{7.13}$$

Now consider the two LMI feasibility conditions

$$\exists X \in \{\, \mathbb{X} \mid A^*XA - \beta^2 X - C^*C < 0 \,\} \tag{7.14}$$

$$\exists X \in \{\, \mathbb{X} \mid C_\perp(A^*XA - \beta^2 X)C_\perp^* < 0 \,\}\,. \tag{7.15}$$

In Equation (7.15), $C_\perp^*$ is a matrix whose columns form a basis for the kernel of C. It is not difficult to show that these LMIs are equivalent.

Define the upper bound to μ

$$\widehat{\mu}_{\mathbf{\Delta},C}(A) = \inf_{\beta>0} \{\, \beta \mid (7.14) \text{ is satisfied} \,\}\,. \tag{7.16}$$

It is easy to show (see Paganini and Doyle, [31]) that $\mu_{\mathbf{\Delta},C}(A) \leq \widehat{\mu}_{\mathbf{\Delta},C}(A)$. Thus if Equation (7.14) is satisfied for $\beta < 1$ then the answer to the question $\mathbf{Q}_\mu$ is negative.

A natural question is under which conditions the upper bound is exact ($\mu = \widehat{\mu}$). From the point of view of $\mathbf{\Delta}$, we inquire which structures are μ-simple (i.e. which structures ensure $\mu = \widehat{\mu}$ for any matrices A, C). The answer from [31] is the following:

Proposition 7.4 *The following structures are μ-simple in the implicit case.*

(i) $\mathbf{\Delta} = \{\, \delta I \mid \delta \in M_{KJ} \}$
(ii) $\mathbf{\Delta} = \mathbb{C}^{n\times n}$
(iii) $\mathbf{\Delta} = \{\, \mathrm{diag}[\Delta_1, \Delta_2] \mid \Delta_i \in \mathbb{C}^{m_i \times m_i} \}$
(iv) $\mathbf{\Delta} = \{\, \mathrm{diag}[\Delta_1, \Delta_2] \mid \Delta_i \in \mathbb{R}^{m_i \times m_i} \}$, *for A, C real*

Remarks: In standard μ, two additional complex structures are μ-simple ($\mathrm{diag}[\delta_1 I, \Delta_2]$ and $\mathrm{diag}[\Delta_1, \Delta_2, \Delta_3]$) (see [27]), but this does not carry through to the implicit case (see [31]).

The upper bound is an exact robust stability test for the dynamic problem where the uncertainty is in a class of time-varying operators (see [31]).

Even if the structure is not μ-simple, special cases on the matrices A, C can yield $\mu = \widehat{\mu}$. An example of this is when A is rank 1 in the kernel of C.

In the case where $\mathbf{\Delta}$ contains blocks of repeated real perturbations, a tighter upper bound can be defined as Chapter 4 by adding a term $j(A^*G - GA)$ to the LMI in Equation (7.14). In this case $G = G^* = \mathrm{diag}[G_1 \ldots G_r, 0 \ldots 0]$, where the G_i blocks correspond to the real δI blocks.

Although the upper bound can in principle be computed exactly by solving an LMI, MV/ID problems can have very large matrices with extensive repetition in uncertainty blocks. In these cases the LMI involves very large full blocks, which standard LMI solvers cannot handle. By taking advantage of special structure in these repeated problems (see Tierno and Doyle, [36]), some reduction in complexity should be possible, but further research is required.

Lower Bounds

In contrast to an upper bound, which guarantees that there there are no solutions to the equations in $\mathbf{Q}_\mu$, a lower bound guarantees that there is a solution to the equations. This suggests an equivalent definition of $\mu_{\mathbf{\Delta},C}(A)$:

$$\mu_{\mathbf{\Delta},C}(A) \triangleq \max \left\{ \beta \;\middle|\; (\beta, \Delta) \in S \bigcup \{(0,0)\} \right\} \tag{7.17}$$

$$S \triangleq \left\{ (\beta, \Delta) \;\middle|\; \|\Delta\| = 1, \ker \begin{bmatrix} \beta I - \Delta A \\ C \end{bmatrix} \neq 0 \right\}. \tag{7.18}$$

If a point is found in S with $\beta \geq 1$, then the answer to the question $\mathbf{Q}_\mu$ is yes.

For a general problem, the set S may be disjoint, it may have isolated points, or it may be empty. We do not consider the general problem here;

our problems are easier. The MV/ID problems are continuous in the following sense: if we have a point in S then a small change in uncertainty or unknown parameters, along with the appropriate small change in measurement noise or equation error, provides us with another point in S. We can find a point in S for a MV/ID problem by choosing any values for the uncertainty and finding the output noise that is consistent. This allows a perturbation analysis of optimality.

Proposition 7.5 *Let P parametrize $\ker(C)$ and let $0 = (\beta I - \Delta A)Px$ so that $(\beta, \Delta) \in S$. If $\exists\, \widehat{\beta} > 0$, $\widehat{\Delta} \leq 0$, and $\widehat{x}$ such that*

$$0 = (\widehat{\beta} - \widehat{\Delta}A)Px + (\beta - \Delta A)P\widehat{x} \tag{7.19}$$

then (β, Δ) is not a local maximum of Equation (7.17).

Remarks: Note that this test is a linear programming feasibility test on $\widehat{\beta}$, $\widehat{\Delta}$, and $\widehat{x}$.

More generally we may encounter problems that are not continuous. We then face the question of the physical or engineering interpretation of such a problem. Often such a problem is poorly posed. If one chooses to proceed with a discontinuous problem, one can go through a regularization procedure completely analogous to the one in Section 2.1.

The approach to the computation of the maximization problem in Equation (7.17) is heavily influenced by the fact that the problem is NP hard (μ is a special case of Equation (7.17) and is NP complete).

As before, to obtain acceptable computation for the large problems we are interested in, one is forced either to consider special cases, e.g. μ-simple problems, or relax the requirement for exact computation.

Lower bound computation for the special case $C = 0$ in Chapter 3 shows that specialized algorithms are much better than standard optimization code. The successful algorithms can be separated roughly into the two classes discussed below, which look for a local maximum of β in S.

The first class of algorithms is based on conditions on β, Δ, and x consistent with a local maximum. These are called alignment conditions. The algorithms try to satisfy these conditions directly. If these local maximum conditions are not achieved, then no lower bound is provided. The SPA and ROA described in Chapter 3 are of this type and are efficient and usually calculate the structured singular value reasonably accurately. The principal difficulty in generalizing the algorithms of this type is that the conditions for a local maximum derived from Proposition 7.5 are considerably more expensive to compute than the alignment conditions of the $C = 0$ case. Since it is difficult to analyze a power

algorithm mathematically, the performance of such algorithms is measured by testing the algorithm on a large set of representative problems.

The second class of algorithms moves from one point in S to another, so a lower bound is constantly being provided and improved upon. The first step is to find a point in S, which is easy in our case. Next, we check conditions in Equation (7.19) to decide if we are done. If we are not done, then a better point can be found nearby. This can be proved using a contraction mapping and starting in the direction of $(\widehat{\beta}, \widehat{\Delta})$. Although standard optimization code is in this second class of algorithms, the algorithms developed in Chapter 3 suggest that specialized code incorporating elements of algorithms of the first class will be much better.

7.5 The Least Squares Problem Revisited

A canonical example of a computationally easy problem in system identification is the least squares problem: in the linear regression setup $y = M\theta + d$ of Equation (7.9) we wish to find the values of θ which satisfy the equations and minimize the 2 norm of the vector d. Assuming M is full column rank, this problem has an explicit solution $\widehat{\theta} = (M^*M)^{-1}M^*y$, which gives a minimum of $\|d\|^2$ equal to

$$\gamma_0^2 = y^*(I - M(M^*M)^{-1}M^*)y. \tag{7.20}$$

An important requirement to validate our approach to MV/ID is to ensure that this simple problem does not turn into a hard one when recast in our analysis framework. Of course, no solution is more efficient than the least squares solution, and we would not expect a method which is developed to encompass quite general problems to be optimal in a special case. We do, however, want the problem to be tractable in the new formulation. Fortunately, as is shown below, the resulting implicit analysis problem is indeed tractable and we can recover the least squares solution.

We first specialize the MV/ID setup of Figure 7.5 to the linear regression case. This is done by following the steps in Section 7.2, and yields the diagram on Figure 7.6.

The only modification to the setup in Figure 7.5 is that scalings k, γ are added to fix the allowable sizes of θ, d respectively. In the least squares problem, we attempt to find the smallest value of γ such that the corresponding $\mathbf{Q}_\mu$ gives an affirmative answer for some value of k (we do not have constraints on parameter size in this case). So the strategy is to let $k \implies \infty$ in the analysis, and attempt to minimize γ.

For simplicity assume $\|y\| = 1$. Therefore the D matrix in the implicit representation (refer to Figure 7.1) is left invertible, and Proposition 7.2 converts

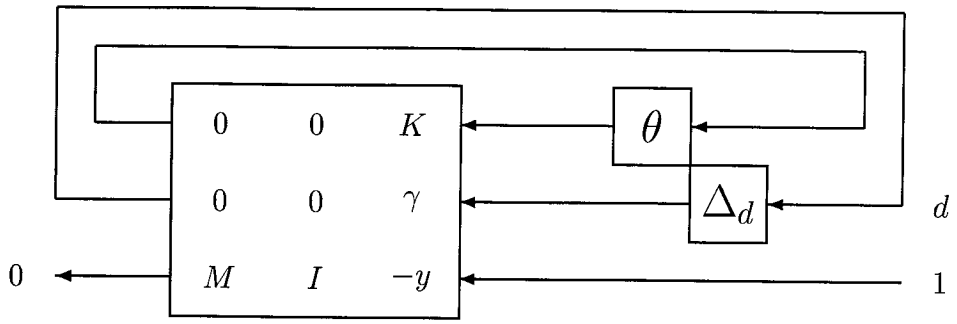


Figure 7.6: The least squares problem.

it to the standard form of Equation (7.5), with

$$\begin{aligned} A &= \begin{bmatrix} ky^*M & ky^* \\ \gamma y^*M & \gamma y^* \end{bmatrix} \\ C &= (I - yy^*) \begin{bmatrix} M & I \end{bmatrix}. \end{aligned} \tag{7.21}$$

The structure $\mathbf{\Delta}$ consists in this case in two real full blocks. Although they are nonsquare (they are columns) the analysis in Proposition 7.4 (*iv*) remains valid; therefore the structure is μ-simple and the LMIs in Equations (7.14) and (7.15) with $\beta = 1$ capture exactly the answer to $\mathbf{Q}_\mu$. This fact is already an indication that our problem remains tractable in the new formulation, although a solution based on LMIs is less efficient than the least squares solution.

To make the point clearer, we can show explicitly that the LMI approach gives the least squares solution as $k \implies \infty$. Consider the LMI in Equation (7.15) with $\beta = 1$; the X scaling in this case consists of two scalar parameters, one for each of the full blocks. Let us call the first one $x > 0$, and we can normalize the second one to 1. Some algebra gives $C_\perp^* = \begin{bmatrix} I & 0 \\ -M & y \end{bmatrix}$,

$$C_\perp (A^*XA - X) C_\perp^* = \begin{bmatrix} -(xI + M^*M) & -M^*y \\ -y^*M & k^2x + \gamma^2 - 1 \end{bmatrix}. \tag{7.22}$$

A Schur complement operation reduces Equation (7.15) to

$$\exists x \in \left\{ x \;\middle|\; k^2x + \gamma^2 - 1 + y^*M(xI + M^*M)^{-1}M^*y < 0 \right\}. \tag{7.23}$$

As $k \implies \infty$, the unknown x must go to zero if (7.23) is to be satisfied; this implies that $(xI + M^*M)^{-1} \implies (M^*M)^{-1}$ and the LMI is feasible for large k if and only if

$$\gamma^2 < 1 - y^*M(M^*M)^{-1}M^*y = \gamma_0^2. \tag{7.24}$$

Recapitulating,

- For $\gamma < \gamma_0$ the LMI is feasible and therefore $\widehat{\mu}_{\mathbf{\Delta},C}(A) = \mu_{\mathbf{\Delta},C}(A) < 1$, which in turn implies that the answer to $\mathbf{Q}_\mu$ is negative (no solutions with $\|d\| \leq \gamma$).

- For $\gamma > \gamma_0$ and large enough k the LMI is not feasible, $\widehat{\mu}_{\mathbf{\Delta},C}(A) = \mu_{\mathbf{\Delta},C}(A) \geq 1$ and therefore there are solutions to the MV/ID problem.

So we again find that γ_0 is the minimum norm for d, as expected. We have not shown how to solve for θ and d, but this information also can be obtained from the LMI approach.

We reiterate that we are not advocating this method for a least squares problem; this is a method suitable for a large class of problems. We have simply shown that it remains tractable in this simple case.

Bibliography

[1] V. Balakrishnan, Stephen Boyd, and S. Balemi. Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems. *International Journal of Robust and Nonlinear Control*, 1(4):295–317, October–December 1991.

[2] Gary J. Balas, John C. Doyle, Keith Glover, Andrew K. Packard, and Roy Smith. The μ analysis and synthesis toolbox. MathWorks and MUSYN, 1991.

[3] Gary J. Balas and Andrew K. Packard. Development and application of time-varying μ-synthesis techniques for control design of missile autopilots. John Hopkins Applied Physics Laboratories, Final Report, January, 1992.

[4] Gary J. Balas, Peter M. Young, and John C. Doyle. μ based control design as applied to a large space structure: Control design for the minimast facility. NASA CSI/GI final report, June 1992.

[5] A. C. Bartlett, C. V. Hollot, and H. Lin. Root locations of an entire polytope of polynomials: It suffices to check the edges. In *Mathematics of Control, Signals and Systems*. Springer Verlag, 1988.

[6] Carolyn Beck and John C. Doyle. Mixed μ upper bound computation. In *Proceedings of the 31st Conference on Decision and Control*, pages 3187–3192, 1992.

[7] Richard D. Braatz, Peter M. Young, John C. Doyle, and Manfred Morari. Computational complexity of μ calculation. *IEEE Transactions on Automatic Control*, 39:1000–1002, 1994.

[8] Jie Chen, Michael K. H. Fan, and Carl N. Nett. The structured singular value and stability of uncertain polynomials: A missing link. *Control of Systems with Inexact Dynamic Models, ASME*, pages 15–23, 1991.

[9] Raffaello D'Andrea and Fernando Paganini. Interconnection of uncertain behavioral systems for robust control. In *Proceedings of the 32nd Conference on Decision and Control*, pages 3642–3647, 1993.

[10] Raffaello D'Andrea, Fernando Paganini, and John C. Doyle. Uncertain behavior. In *Proceedings of the 32nd Conference on Decision and Control*, pages 3891–3896, 1993.

[11] Raymond R. E. de Gaston and Michael G. Safonov. Exact calculation of the multiloop stability margin. *IEEE Transactions on Automatic Control*, 33:156–171, 1988.

[12] James Demmel. The componentwise distance to the nearest singular matrix. *SIAM Journal on Matrix Analysis and Applications*, 13:10–19, 1992.

[13] John C. Doyle. Analysis of feedback systems with structured uncertainty. *IEE Proceedings, Part D*, 129(6):242–250, November 1982.

[14] John C. Doyle, Andrew K. Packard, and Kemin Zhou. Review of LFTs, LMIs and μ. In *Proceedings of the 30th Conference on Decision and Control*, pages 1227–1232, 1991.

[15] Michael K. H. Fan and Andre L. Tits. Characterization and efficient computation of the structured singular value. *IEEE Transactions on Automatic Control*, AC-31:734–743, 1986.

[16] Michael K. H. Fan, Andre L. Tits, and John C. Doyle. Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics. *IEEE Transactions on Automatic Control*, AC-36:25–38, 1991.

[17] A. L. Fradkov and V. A. Yakubovich. The S-procedure and duality theorems for nonconvex problems of quadratic programming. Technical report, Leningrad. Univ., 1973.

[18] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP Completeness.* W. H. Freeman, New York, 1979.

[19] Tosio Kato. *A Short Introduction to Perturbation Theory for Linear Operators.* Springer Verlag, New York, 1982.

[20] V. L. Kharitonov. Asymptotic stability of an equilibrium position of a family of systems of linear differential equations. *Differential Equations*, 14:1483–1485, 1979.

[21] Lennart Ljung. *System Identification, Theory for the User.* Information and System Sciences Series. Prentice-Hall, New Jersey, 1987.

[22] A. Megretski and S. Treil. Power distribution inequalities in optimization and robustness of uncertain systems. *Journal of Mathematical Systems, Estimation, and Control*, 3, No. 3:301–319, 1993.

[23] A. Megretsky. Power distribution approach in robust control. Technical report, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden, 1992.

[24] E. E. Osborne. On preconditioning of matrices. *Journal of the Association for Computing Machinery*, 7:338–345, 1960.

[25] Andrew K. Packard. *What's new with μ: Structured Uncertainty in Multivariable Control.* PhD thesis, University of California, Berkeley, 1988.

[26] Andrew K. Packard and John C. Doyle. Structured singular value with repeated scalar blocks. In *Proceedings of the American Control Conference*, pages 1213–1218, 1988.

[27] Andrew K. Packard and John C. Doyle. The complex structured singular value. *Automatica*, 29:71–109, 1993.

[28] Andrew K. Packard, Michael K. H. Fan, and John C. Doyle. A power method for the structured singular value. In *Proceedings of the 27^{th} Conference on Decision and Control*, pages 2132–2137, 1988.

[29] Andrew K. Packard and Pradeep Pandey. Continuity properties of the real/complex structured singular value. *IEEE Transactions on Automatic Control*, AC-38:415–428, 1993.

[30] Fernando Paganini, Raffaello D'Andrea, and John C. Doyle. Behavioral approach to robustness analysis. In *Proceedings of the American Control Conference*, pages 2782–2786, 1994.

[31] Fernando Paganini and John C. Doyle. Analysis of implicitly defined systems. In *Proceedings of the 33^{rd} Conference on Decision and Control*, pages 3673–3678, 1994.

[32] J. Rohn and S. Poljak. Checking robust nonsingularity is NP-hard. *Mathematics of Control, Signals and Systems*, 6(1):1–9, 1993.

[33] Jeff S. Shamma. Robustness analysis for time-varying systems. In *Proceedings of the 31^{st} Conference on Decision and Control*, pages 3163–3168, 1992.

[34] Athanasios Sideris and Ricardo S. Sanchez Pena. Fast computation of the multivariable stability margin for real interrelated uncertain parameters. *IEEE Transactions on Automatic Control,*, 34(12):1272–1276, December 1989.

[35] Roy S. Smith and John C. Doyle. Model validation: A connection between robust control and identification. *IEEE Transactions on Automatic Control*, 37(7):942–952, July 1992.

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

[37] Jorge E. Tierno and Peter M. Young. An improved μ lower bound via adaptive power iteration. In *Proceedings of the 31st Conference on Decision and Control*, pages 3181–3186, 1992.

[38] Jan C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 36:259–294, 1991.

[39] Peter M. Young. *Robustness with Parametric and Dynamic Uncertainty.* PhD thesis, California Institute of Technology, 1993.

[40] Peter M. Young. The rank one mixed μ problem and "Kharitonov-type" analysis. *Automatica*, 30(12):1899–1911, December 1994.

[41] Peter M. Young and John 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.

[42] Peter M. Young and John C. Doyle. Properties of the mixed μ problem and its bounds, 1993. Submitted to IEEE Transactions on Automatic Control.

[43] Peter M. Young, Matthew P. Newlin, and John C. Doyle. μ analysis with real parametric uncertainty. In *Proceedings of the 30th Conference on Decision and Control*, pages 1251–1256. IEEE, 1991.