

Global Optimization in Control System Analysis and Design

Venkataramanan Balakrishnan
Stephen Boyd

Information Systems Laboratory
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
U. S. A.

I. INTRODUCTION

Many problems in control system analysis and design can be posed in a setting where a system with a fixed model structure and nominal parameter values is affected by parameter variations. An example is parametric robustness analysis, where the parameters might represent physical quantities that are known only to within a certain accuracy, or vary depending on operating conditions etc. Frequently asked questions here deal with performance issues: “How bad can a certain performance measure of the system be over all possible values of the parameters?” Another example is parametric controller design, where the parameters represent degrees of freedom available to the control system designer. A typical question here would be: “What is the best choice of parameters, one that optimizes a certain design objective?”

Many of the questions above may be directly restated as optimization

problems: If q denotes the vector of parameters, $\mathcal{Q}_{\text{init}}$ the set of values that q might assume and $f(q)$ an objective function, then the questions above translate into solving one of the following optimization problems:

$$\text{P1: } \min_{q \in \mathcal{Q}_{\text{init}}} f(q),$$

or

$$\text{P2: } \max_{q \in \mathcal{Q}_{\text{init}}} f(q).$$

In general, (P1) and (P2) are *non-convex* optimization problems, and are much harder¹ to solve than say, *convex* optimization problems, for which there exist a number of effective algorithms. Solving (P1) or (P2) where $\mathcal{Q}_{\text{init}}$ is a set with finite or countably infinite elements is the well-studied combinatorial optimization problem [2].

Since solving (P1) or (P2) is hard in general, it is worth discussing the costs and benefits associated with approximate or suboptimal solutions such as local optimization methods, Monte Carlo methods, gridding etc. The attractiveness of such methods stems from the ease with which they may be performed — they typically require much less computation than global optimization methods. The cost associated with a suboptimal solution depends upon the underlying physical problem. For example, suppose that problem (P1) arises from robustness analysis, where one seeks the worst (smallest) possible value of a certain performance measure $f(q)$. Then confidence misplaced on the local minimum returned by a local optimization procedure might be potentially disastrous, that is, local optimization might not find the worst-case parameter which might, for example, render the system unstable. In this case, the cost associated with not finding the global minimum would be high. (In such situations, methods that yield lower bounds are of greater value, evidenced by the vast research into conservative analytical techniques for robustness analysis.) On the other hand, if (P1) arises from a design problem, where one seeks the parameters that yield the smallest value of a design objective, a local minimization method would yield a possibly conservative upper bound for the global minimum. In this case, the cost associated with not finding the global minimum would usually be acceptable.

¹For a quantitative description of the term “hard”, see for example, [1].

There exist several popular methods for solving global optimization problems (P1) or (P2) (see [3], for example). *Simulated Annealing* (see [4] and the references therein) describes a family of iterative methods where every iteration consists of taking a step in parameter space with a probability that decays exponentially with an “energy” function associated with the new parameter value. This technique reportedly performs well and has been applied widely to computer-aided design of electronic integrated circuits, design of error-correcting codes etc. It must be mentioned, however, that simulated annealing does not maintain both upper and lower bounds of the global optimum that it seeks. As a result, the algorithm has no stopping criterion; and its termination, at any time, does not yield any bounds for the optimum. This is not a serious drawback in the many applications where it has been successfully used, since it has led to designs that are significantly better than any found before; the cost associated with not finding the globally optimal design in these cases is acceptable. However, since annealing does not yield any guarantees about the solutions it yields — though some probabilistic statements can be made about the convergence to the global optimum — it cannot be used for problems such as robustness analysis where the cost associated with not finding the global optimum is usually high. Another approach to global optimization is to apply interior point algorithms [5], which have been reported to perform well on some integer programming problems. Here again, as with annealing, there are no guaranteed bounds on the optimum.

In contrast with the above techniques for global optimization, *branch and bound* algorithms, as they progress, do maintain upper and lower bounds for the global optimum; thus termination at any time yields guaranteed bounds for the optimum. These algorithms derive their name from the way they proceed: They break up the parameter region into subregions (“branching”) to derive bounds for the global optimum over the original region (“bounding”). The branching is done based on some heuristic rules. Though these heuristics often work well, it must be emphasized that these algorithms are worst-case combinatoric. Thus they may require unacceptably long computation times on some simple problems.

Traditionally, branch and bound algorithms have been used in discrete programming problems (see [6, 7] for early articles, [8, 9] for surveys and [10, 11, 3] for texts). In [12], Hansen combines interval analysis with a

branch and bound scheme very similar to ours for global optimization. A more recent application of a branch and bound algorithm on a parametric robustness problem arising in control systems analysis is in [13], where De Gaston and Safonov use a branch and bound algorithm for computing the robust stability margin for systems with uncorrelated uncertain parameters. Sideris and Peña [14] extend this algorithm to the case when the parameters are real and may be correlated. In [15], Chang *et al.* describe a similar branch and bound algorithm for computing the real structured singular value and the real multivariable stability margin. Vicino *et al.* [16] use a branch and bound algorithm with geometric programming ideas to compute the robust stability margin. Demarco *et al.* [17] use a branch and bound algorithm to study stability problems arising in power systems.

In this chapter, we restrict our attention to following setup: *We consider linear systems with a number of constant, unknown parameters that lie in intervals.* Thus, the parameter region $\mathcal{Q}_{\text{init}}$ is a rectangle. For such systems, we consider problems of parameter robustness analysis and parameter selection (that is, design). We show how a branch and bound technique may be used to solve these problems.

The organization of the chapter is as follows. Section II describes the basic branch and bound algorithm, its convergence properties and a simple extension. Section III discusses some problems that arise in parameter dependent linear systems, Section IV discusses the computation of bounds for these problems and Section V presents some simple examples that illustrate the performance of the branch and bound algorithm on these problems. Section VI makes some closing remarks.

Some Notation

\mathbf{R} (\mathbf{C}) denotes the set of real (complex) numbers. For $c \in \mathbf{C}$, $\text{Re } c$ is the real part of c . The set of $m \times n$ matrices with real (complex) entries is denoted $\mathbf{R}^{m \times n}$ ($\mathbf{C}^{m \times n}$). P^T stands for the transpose of P , and P^* , the complex conjugate transpose. I denotes the identity matrix, with size determined from context.

For a matrix $P \in \mathbf{R}^{n \times n}$ (or $\mathbf{C}^{n \times n}$), $\lambda_i(P)$, $1 \leq i \leq n$ denotes the i th eigenvalue of P (with no particular ordering). $\text{Tr}(P)$ stands for the trace (sum of the diagonal entries) of P . $\bar{\sigma}(P)$ denotes the maximum singular

value of P , defined as

$$\bar{\sigma}(P) = \max_{1 \leq i \leq n} \sqrt{\lambda_i(P^*P)},$$

and $\underline{\sigma}(P)$ the minimum singular value of P , defined as

$$\underline{\sigma}(P) = \min_{1 \leq i \leq n} \sqrt{\lambda_i(P^*P)}.$$

The *condition number* of a matrix P with positive minimum singular value is the ratio $\bar{\sigma}(P)/\underline{\sigma}(P)$ (it is defined to be ∞ if $\underline{\sigma}(P) = 0$). $\|P\|_F$ is the Frobenius norm of P , given by $\sqrt{\text{Tr}(P^*P)}$. The definitions for $\bar{\sigma}(P)$ and $\|P\|_F$ hold also for $P \in \mathbf{R}^{m \times n}$ (or $\mathbf{C}^{m \times n}$).

II. A BRANCH AND BOUND ALGORITHM

Most of the material (Subsections A and B) in this section is from [18]. We reproduce it here for completeness.

The branch and bound algorithm we present here finds the global minimum of a function $f : \mathbf{R}^m \rightarrow \mathbf{R}$ over an m -dimensional rectangle $\mathcal{Q}_{\text{init}}$. (Of course, by replacing f by $-f$, the algorithm can also be used to find the global maximum.)

For a rectangle $\mathcal{Q} \subseteq \mathcal{Q}_{\text{init}}$ we define

$$\Phi_{\min}(\mathcal{Q}) = \min_{q \in \mathcal{Q}} f(q).$$

Then, the algorithm computes $\Phi_{\min}(\mathcal{Q}_{\text{init}})$ to within an absolute accuracy of $\epsilon > 0$, using two functions $\Phi_{\text{lb}}(\mathcal{Q})$ and $\Phi_{\text{ub}}(\mathcal{Q})$ defined over $\{\mathcal{Q} \mid \mathcal{Q} \subseteq \mathcal{Q}_{\text{init}}\}$ (which, presumably, are easier to compute than $\Phi_{\min}(\mathcal{Q})$). These two functions satisfy the following conditions.

$$(R1) \quad \Phi_{\text{lb}}(\mathcal{Q}) \leq \Phi_{\min}(\mathcal{Q}) \leq \Phi_{\text{ub}}(\mathcal{Q}).$$

Thus, the functions Φ_{lb} and Φ_{ub} compute a lower and upper bound on $\Phi_{\min}(\mathcal{Q})$, respectively.

(R2) As the maximum half-length of the sides of \mathcal{Q} , denoted by $\text{size}(\mathcal{Q})$, goes to zero, the difference between upper and lower bounds *uniformly* converges to zero, *i.e.*,

$$\forall \epsilon > 0 \exists \delta > 0 \text{ such that} \\ \forall \mathcal{Q} \subseteq \mathcal{Q}_{\text{init}}, \text{ size}(\mathcal{Q}) \leq \delta \implies \Phi_{\text{ub}}(\mathcal{Q}) - \Phi_{\text{lb}}(\mathcal{Q}) \leq \epsilon.$$

Roughly speaking, then, the bounds Φ_{lb} and Φ_{ub} become sharper as the rectangle shrinks to a point.

We now describe the algorithm. We start by computing $\Phi_{\text{lb}}(\mathcal{Q}_{\text{init}})$ and $\Phi_{\text{ub}}(\mathcal{Q}_{\text{init}})$. If $\Phi_{\text{ub}}(\mathcal{Q}_{\text{init}}) - \Phi_{\text{lb}}(\mathcal{Q}_{\text{init}}) \leq \epsilon$, the algorithm terminates. Otherwise we partition $\mathcal{Q}_{\text{init}}$ as a union of subrectangles as $\mathcal{Q}_{\text{init}} = \mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \dots \cup \mathcal{Q}_N$, and compute $\Phi_{\text{lb}}(\mathcal{Q}_i)$ and $\Phi_{\text{ub}}(\mathcal{Q}_i)$, $i = 1, 2, \dots, N$. Then

$$\min_{1 \leq i \leq N} \Phi_{\text{lb}}(\mathcal{Q}_i) \leq \Phi_{\text{min}}(\mathcal{Q}_{\text{init}}) \leq \min_{1 \leq i \leq N} \Phi_{\text{ub}}(\mathcal{Q}_i),$$

so we have new bounds on $\Phi_{\text{min}}(\mathcal{Q}_{\text{init}})$. If the difference between the new bounds is less than or equal to ϵ , the algorithm terminates. Otherwise, the partition of $\mathcal{Q}_{\text{init}}$ is further refined and the bounds updated.

If a partition $\mathcal{Q}_{\text{init}} = \cup_{i=1}^N \mathcal{Q}_i$ satisfies $\text{size}(\mathcal{Q}_i) \leq \delta$, $i = 1, 2, \dots, N$, then by condition (R2) above,

$$\min_{1 \leq i \leq N} \Phi_{\text{ub}}(\mathcal{Q}_i) - \min_{1 \leq i \leq N} \Phi_{\text{lb}}(\mathcal{Q}_i) \leq \epsilon;$$

thus a “ δ -grid” ensures that $\Phi_{\text{min}}(\mathcal{Q}_{\text{init}})$ is determined to within an absolute accuracy of ϵ . However, for the “ δ -grid”, the number of rectangles forming the partition (and therefore the number of upper and lower bound calculations) grows exponentially with $1/\delta$. The branch and bound algorithm applies a heuristic rule for partitioning $\mathcal{Q}_{\text{init}}$, which in most cases leads to a reduction of the number of calculations required to solve the problem compared to the δ -grid. The heuristic is this: Given any partition $\mathcal{Q}_{\text{init}} = \cup_{i=1}^N \mathcal{Q}_i$ that is to be refined, pick a rectangle \mathcal{Q} from the partition such that $\Phi_{\text{lb}}(\mathcal{Q}) = \min_{1 \leq i \leq N} \Phi_{\text{lb}}(\mathcal{Q}_i)$, and split it into two halves. The rationale behind this rule is that since we are trying to find the minimum of a function, we should concentrate on the “most promising” rectangle. We must emphasize that this is a heuristic, and in the worst case will result in a δ -grid.

A. The general branch and bound algorithm

In the following description, k stands for the iteration index. \mathcal{L}_k denotes the list of rectangles, L_k the lower bound and U_k the upper bound for $\Phi_{\text{min}}(\mathcal{Q}_{\text{init}})$, at the end of k iterations.

Algorithm I

```

 $k = 0;$ 
 $\mathcal{L}_0 = \{Q_{\text{init}}\};$ 
 $L_0 = \Phi_{\text{lb}}(Q_{\text{init}});$ 
 $U_0 = \Phi_{\text{ub}}(Q_{\text{init}});$ 
while  $U_k - L_k > \epsilon$ , {
    pick  $Q \in \mathcal{L}_k$  such that  $\Phi_{\text{lb}}(Q) = L_k$ ;
    split  $Q$  along one of its longest edges into  $Q_I$  and  $Q_{II}$ ;
     $\mathcal{L}_{k+1} := (\mathcal{L}_k - \{Q\}) \cup \{Q_I, Q_{II}\};$ 
     $L_{k+1} := \min_{Q \in \mathcal{L}_{k+1}} \Phi_{\text{lb}}(Q);$ 
     $U_{k+1} := \min_{Q \in \mathcal{L}_{k+1}} \Phi_{\text{ub}}(Q);$ 
     $k := k + 1;$ 
}

```

The requirement that we split the chosen rectangle along a longest edge may seem mysterious at this point. This splitting rule controls the condition number of the rectangles in the partition; see the proof of convergence in Subsection B.

At the end of k iterations, U_k and L_k are upper and lower bounds respectively for $\Phi_{\min}(Q_{\text{init}})$. We prove in Subsection B that if the bounds $\Phi_{\text{lb}}(Q)$ and $\Phi_{\text{ub}}(Q)$ satisfy condition (R2), $U_k - L_k$ is guaranteed to converge to zero, and therefore the branch and bound algorithm will terminate in a finite number of steps.

It is clear that in the branching process described above, the number of rectangles is equal to the number of iterations N . However, we can often eliminate some rectangles from consideration; they may be *pruned* since $\Phi_{\min}(Q_{\text{init}})$ cannot be achieved in them. This is done as follows. At each iteration:

Eliminate from list \mathcal{L}_k the rectangles $Q \in \mathcal{L}_k$ that satisfy

$$\Phi_{\text{lb}}(Q) > U_k.$$

If a rectangle $Q \in \mathcal{L}_k$ satisfies this condition, then $q \in Q \Rightarrow f(q) > U_k$; however the minimum of $f(q)$ over Q_{init} is *guaranteed* to be less than U_k , and therefore cannot be found in Q .

Though pruning is not necessary for the algorithm to work, it does reduce storage requirements. The algorithm often quickly prunes a large

portion of $\mathcal{Q}_{\text{init}}$, and works with only a small remaining subset. The set \mathcal{L}_k , the union of the rectangles in the pruned list, acts as an approximation of the set of minimizers of f . In fact, every minimizer of f is guaranteed to be in \mathcal{L}_k .

The term *pruning* comes from the following. The algorithm can be viewed as growing a binary tree of rectangles representing the current partition \mathcal{L}_k , with the nodes corresponding to rectangles and the children of a given node representing the two halves obtained by splitting it. By removing a rectangle from consideration, we prune the tree.

As we noted at the beginning of this section, the above algorithm can also be used for global maximization, merely by minimizing $-f$. However, we will find it convenient to have a version of the algorithm for directly finding $\Psi_{\max}(\mathcal{Q}_{\text{init}}) = \max_{q \in \mathcal{Q}_{\text{init}}} f(q)$. Here, for $\mathcal{Q} \subseteq \mathcal{Q}_{\text{init}}$, $\Psi_{\text{lb}}(\mathcal{Q})$ and $\Psi_{\text{ub}}(\mathcal{Q})$ denote lower and upper bounds for $\Psi_{\max}(\mathcal{Q})$, and are required to satisfy

$$(R1') \quad \Psi_{\text{lb}}(\mathcal{Q}) \leq \Psi_{\max}(\mathcal{Q}) \leq \Psi_{\text{ub}}(\mathcal{Q}).$$

$$(R2')$$

$\forall \epsilon > 0 \exists \delta > 0$ such that

$$\forall \mathcal{Q} \subseteq \mathcal{Q}_{\text{init}}, \text{ size}(\mathcal{Q}) \leq \delta \implies \Psi_{\text{ub}}(\mathcal{Q}) - \Psi_{\text{lb}}(\mathcal{Q}) \leq \epsilon.$$

In the following, L_k and U_k give lower and upper bounds for $\Psi_{\max}(\mathcal{Q}_{\text{init}})$ at the end of k iterations.

Algorithm II

```

k = 0;
 $\mathcal{L}_0 = \{\mathcal{Q}_{\text{init}}\};$ 
 $L_0 = \Psi_{\text{lb}}(\mathcal{Q}_{\text{init}});$ 
 $U_0 = \Psi_{\text{ub}}(\mathcal{Q}_{\text{init}});$ 
while  $U_k - L_k > \epsilon$ , {
    pick  $\mathcal{Q} \in \mathcal{L}_k$  such that  $\Psi_{\text{ub}}(\mathcal{Q}) = U_k$ ;
    split  $\mathcal{Q}$  into  $\mathcal{Q}_I$  and  $\mathcal{Q}_{II}$  along the longest edge;
     $\mathcal{L}_{k+1} := (\mathcal{L}_k - \{\mathcal{Q}\}) \cup \{\mathcal{Q}_I, \mathcal{Q}_{II}\};$ 
     $L_{k+1} := \max_{\mathcal{Q} \in \mathcal{L}_{k+1}} \Psi_{\text{lb}}(\mathcal{Q});$ 
     $U_{k+1} := \max_{\mathcal{Q} \in \mathcal{L}_{k+1}} \Psi_{\text{ub}}(\mathcal{Q});$ 
     $k := k + 1;$ 
}

```

The corresponding pruning step is

Eliminate from list \mathcal{L}_k , the rectangles $Q \in \mathcal{L}_k$ that satisfy

$$\Psi_{\text{ub}}(Q) < L_k.$$

B. Analysis of Convergence of the Branch and Bound Algorithm

We now show that the branch and bound algorithm converges in a finite number of steps, provided the bound functions $\Phi_{\text{lb}}(\cdot)$ and $\Phi_{\text{ub}}(\cdot)$ satisfy conditions (R1) and (R2) listed at the beginning of this section. (We will only consider Algorithm I, since the proof for Algorithm II then follows analogously.)

An upper bound on the number of branch and bound iterations

The derivation of an upper bound on the number of iterations of the branch and bound algorithm involves the following steps. We first show that after a large number of iterations k , the partition \mathcal{L}_k must contain a rectangle of small volume. (The volume of a rectangle is defined as the product of the lengths of its sides.) We then show that this rectangle has a small size, and this in turn implies that $U_k - L_k$ is small.

First, we observe that the number of rectangles in the partition \mathcal{L}_k is just k (without pruning, which in any case does not affect the number of iterations). The total volume of these rectangles is $\text{vol}(Q_{\text{init}})$, and therefore

$$\min_{Q \in \mathcal{L}_k} \text{vol}(Q) \leq \frac{\text{vol}(Q_{\text{init}})}{k}. \quad (1)$$

Thus, after a large number of iterations, at least one rectangle in the partition has small volume.

Next, we show that small volume implies small size for a rectangle in any partition. We define the *condition number* of a rectangle $Q = \prod_i [l_i, u_i]$ as

$$\text{cond}(Q) = \frac{\max_i (u_i - l_i)}{\min_i (u_i - l_i)}.$$

We then observe that our splitting rule, which requires that we split rectangles along a longest edge, results in an upper bound on the condition number of rectangles in our partition.

Lemma 1 For any k and any rectangle $\mathcal{Q} \in \mathcal{L}_k$,

$$\text{cond}(\mathcal{Q}) \leq \max\{\text{cond}(\mathcal{Q}_{\text{init}}), 2\}. \quad (2)$$

Proof

It is enough to show that when a rectangle \mathcal{Q} is split into rectangles \mathcal{Q}_1 and \mathcal{Q}_2 ,

$$\text{cond}(\mathcal{Q}_1) \leq \max\{\text{cond}(\mathcal{Q}), 2\}, \quad \text{cond}(\mathcal{Q}_2) \leq \max\{\text{cond}(\mathcal{Q}), 2\}.$$

Let ν_{\max} be the maximum edge length of \mathcal{Q} , and ν_{\min} , the minimum. Then $\text{cond}(\mathcal{Q}) = \nu_{\max}/\nu_{\min}$. When \mathcal{Q} is split into \mathcal{Q}_1 and \mathcal{Q}_2 , our splitting rule requires that \mathcal{Q} be split along an edge of length ν_{\max} . Thus, the maximum edge length of \mathcal{Q}_1 or \mathcal{Q}_2 can be no larger than ν_{\max} . Their minimum edge length could be no smaller than the minimum of $\nu_{\max}/2$ and ν_{\min} , and the result follows. \blacksquare

We note that there are other splitting rules that also result in a uniform bound on the condition number of the rectangles in any partition generated. One such rule is to cycle through the index on which we split the rectangle. If \mathcal{Q} was formed by splitting its parent along the i th coordinate, then when we split \mathcal{Q} , we split it along the $(i + 1)$ modulo m coordinate.

We can bound the size of a rectangle \mathcal{Q} in terms of its volume and condition number, since

$$\begin{aligned} \text{vol}(\mathcal{Q}) &= \prod_i (u_i - l_i) \\ &\geq \max_i (u_i - l_i) \left(\min_i (u_i - l_i) \right)^{m-1} \\ &= \frac{(2 \text{size}(\mathcal{Q}))^m}{\text{cond}(\mathcal{Q})^{m-1}} \\ &\geq \left(\frac{2 \text{size}(\mathcal{Q})}{\text{cond}(\mathcal{Q})} \right)^m. \end{aligned}$$

Thus,

$$\text{size}(\mathcal{Q}) \leq \frac{1}{2} \text{cond}(\mathcal{Q}) \text{vol}(\mathcal{Q})^{1/m}. \quad (3)$$

Combining equations (1), (2) and (3) we get

$$\min_{\mathcal{Q} \in \mathcal{L}_k} \text{size}(\mathcal{Q}) \leq \frac{1}{2} \max\{\text{cond}(\mathcal{Q}_{\text{init}}), 2\} \left(\frac{\text{vol}(\mathcal{Q}_{\text{init}})}{k} \right)^{1/m}. \quad (4)$$

Thus, for large k , the partition \mathcal{L}_k must contain a rectangle of small size.

Finally, we show that if a partition has a rectangle of small size, the upper and lower bounds cannot be too far apart. More precisely, we show that given some $\epsilon > 0$, there is some N such that $U_N - L_N \leq \epsilon$ for some $N \leq k$.

First, let δ be small enough such that if $\text{size}(\mathcal{Q}) \leq 2\delta$ then $\Phi_{\text{ub}}(\mathcal{Q}) - \Phi_{\text{lb}}(\mathcal{Q}) \leq \epsilon$ (recall requirement (R2) at the beginning of this section). Let k be large enough such that

$$\max\{\text{cond}(\mathcal{Q}_{\text{init}}), 2\} \left(\frac{\text{vol}(\mathcal{Q}_{\text{init}})}{k} \right)^{1/m} \leq 2\delta. \quad (5)$$

Then from equation (4), some $\mathcal{Q} \in \mathcal{L}_k$ satisfies $\text{size}(\mathcal{Q}) \leq \delta$. Then the rectangle $\tilde{\mathcal{Q}}$, one of whose halves is \mathcal{Q} , must satisfy $\text{size}(\tilde{\mathcal{Q}}) \leq 2\delta$, and therefore

$$\Phi_{\text{ub}}(\tilde{\mathcal{Q}}) - \Phi_{\text{lb}}(\tilde{\mathcal{Q}}) \leq \epsilon.$$

However, since $\tilde{\mathcal{Q}}$ was split at some previous iteration, it must have satisfied $\Phi_{\text{lb}}(\tilde{\mathcal{Q}}) = L_N$ for some $N \leq k$. Thus

$$U_N - L_N \leq \Phi_{\text{ub}}(\tilde{\mathcal{Q}}) - L_N \leq \epsilon,$$

or we have an upper bound on the number of branch and bound iterations.

C. Simultaneous maximization of multiple objectives

In many cases, the maximization (or minimization) of several objectives at the same time and over the same parameter rectangle $\mathcal{Q}_{\text{init}}$ may be of interest. In this setting, we have M objective functions $f^{(1)}, f^{(2)}, \dots, f^{(M)}$, defined over the same parameter region $\mathcal{Q}_{\text{init}}$, and we seek to maximize each of these objectives over $\mathcal{Q}_{\text{init}}$. One way to do this is to simply apply the Algorithm II M times, maximizing one objective function each time. However, if the functions $f^{(i)}$, $i = 1, \dots, M$ are “correlated” — this is often the case in robustness analysis where the same set of parameters leads to the worst possible performance with several different objectives, or in controller design where the same set of parameters is optimal for more than one performance measure — this sequential approach would prove wasteful. We now present a heuristic method that exploits any correlation

between the objectives; in the worst-case, this method behaves like the branch and bound algorithm applied M times, without pruning.

We use $\Psi_{\text{lb}}^{(j)}(\mathcal{Q})$ and $\Psi_{\text{ub}}^{(j)}(\mathcal{Q})$ respectively, to denote the lower and upper bounds of the maximum $\Psi_{\text{max}}^{(j)}(\mathcal{Q})$ of the j th objective function $f^{(j)}$ over the parameter rectangle \mathcal{Q} . Then our heuristic algorithm for simultaneous maximization runs as follows: We start by computing the M upper and lower bounds over the initial parameter rectangle, that is, we compute $\Psi_{\text{lb}}^{(j)}(\mathcal{Q}_{\text{init}})$ and $\Psi_{\text{ub}}^{(j)}(\mathcal{Q}_{\text{init}})$ for $i = 1, \dots, M$. If $\Psi_{\text{ub}}^{(j)}(\mathcal{Q}_{\text{init}}) - \Psi_{\text{lb}}^{(j)}(\mathcal{Q}_{\text{init}}) \leq \epsilon$, for every j , the algorithm terminates. Otherwise, we partition $\mathcal{Q}_{\text{init}}$ and proceed to refine the bounds.

The major difference between the multiple objective maximization and the single objective maximization is this: In the maximization of a single objective, given any partition $\mathcal{Q}_{\text{init}} = \cup_{i=1}^N \mathcal{Q}_i$ that is to be refined, we pick a rectangle \mathcal{Q} from the partition such that $\Psi_{\text{ub}}(\mathcal{Q}) = \max_{1 \leq i \leq N} \Psi_{\text{ub}}(\mathcal{Q}_i)$, and split it into two halves. However, in the multiple objective case, there might be no such candidate rectangle to be split, since in general there is no rectangle \mathcal{Q} all of whose upper bounds satisfy $\Psi_{\text{ub}}^{(j)}(\mathcal{Q}) = \max_{1 \leq i \leq N} \Psi_{\text{ub}}^{(j)}(\mathcal{Q}_i)$, $j = 1, \dots, M$. To address this issue, we propose the following heuristic rule: We *cycle* through the objective functions to determine which rectangle to split. More precisely, if at some iteration, we split a rectangle \mathcal{Q} that satisfies $\Psi_{\text{ub}}^{(j)}(\mathcal{Q}) = \max_{1 \leq i \leq N} \Psi_{\text{ub}}^{(j)}(\mathcal{Q}_i)$, then at the next iteration, we split a rectangle \mathcal{Q} that satisfies $\Psi_{\text{ub}}^{(j_{\text{new}})}(\mathcal{Q}) = \max_{1 \leq i \leq N} \Psi_{\text{ub}}^{(j_{\text{new}})}(\mathcal{Q}_i)$, where $j_{\text{new}} = (j + 1) \bmod M$. The iterations continue till the difference between the upper and lower bounds for every objective is less than or equal to ϵ .

Since the original branch and bound algorithm converges, so does this new algorithm. In the following description of the algorithm, L_k and U_k now denote *vectors* of length m . The j th component of L_k , denoted $L_k^{(j)}$, is the lower bound of $\Psi_{\text{max}}^{(j)}(\mathcal{Q}_{\text{init}})$ at the end of k iterations, and similarly for U_k .

Algorithm III

```

 $k = 0; \mathcal{L}_0 = \{\mathcal{Q}_{\text{init}}\};$ 
for  $j=1, \dots, M$  {
     $L_0^{(j)} = \Psi_{\text{lb}}^{(j)}(\mathcal{Q}_{\text{init}});$ 
     $U_0^{(j)} = \Psi_{\text{ub}}^{(j)}(\mathcal{Q}_{\text{init}});$ 
}
until  $\max_j \{U_k^{(j)} - L_k^{(j)}\} \leq \epsilon, \{$ 
     $j_{\text{new}} = k \bmod M;$ 
    pick  $\mathcal{Q} \in \mathcal{L}_k$  such that  $\Psi_{\text{ub}}^{(j_{\text{new}})}(\mathcal{Q}) = U_k^{(j_{\text{new}})};$ 
    split  $\mathcal{Q}$  along one of its longest edges into  $\mathcal{Q}_I$  and  $\mathcal{Q}_{II}$ ;
     $\mathcal{L}_{k+1} := (\mathcal{L}_k - \{\mathcal{Q}\}) \cup \{\mathcal{Q}_I, \mathcal{Q}_{II}\};$ 
    for  $j = 1, \dots, M, \{$ 
         $L_{k+1}^{(j)} := \max_{\mathcal{Q} \in \mathcal{L}_{k+1}} \Psi_{\text{lb}}^{(j)}(\mathcal{Q});$ 
         $U_{k+1}^{(j)} := \max_{\mathcal{Q} \in \mathcal{L}_{k+1}} \Psi_{\text{ub}}^{(j)}(\mathcal{Q});$ 
    }
     $k := k + 1;$ 
}

```

The pruning step needs to be modified, so that the rectangles that are eliminated from the rectangle list are those where *none* of the functions $f^{(j)}$ can achieve their maximum:

Eliminate from list \mathcal{L}_k the rectangles $\mathcal{Q} \in \mathcal{L}_k$ that satisfy

$$\Psi_{\text{ub}}^{(j)}(\mathcal{Q}) < L_k^{(j)}, \text{ for every } j = 1, 2, \dots, M.$$

III. SOME PARAMETER PROBLEMS IN LTI CONTROLLER ANALYSIS AND DESIGN

A. Our Framework

We now apply the branch and bound algorithms of Section II to the computation of some quantities that arise in the analysis and design of parameter-dependent linear systems. Our framework consists of a family of linear

time-invariant systems described by

$$\begin{aligned}
\dot{x} &= Ax + B_u u + B_w w, & x(0) &= x_0, \\
y &= C_y x + D_{yu} u + D_{yw} w, \\
z &= C_z x + D_{zu} u + D_{zw} w, \\
u &= \Delta y,
\end{aligned} \tag{6}$$

where $x(t) \in \mathbf{R}^n$, $w(t) \in \mathbf{R}^{n_i}$, $z(t) \in \mathbf{R}^{n_o}$, $u(t), y(t) \in \mathbf{R}^p$, and A , B_u , B_w , C_y , C_z , D_{yu} , D_{yw} , D_{zu} and D_{zw} are real matrices of appropriate sizes. Δ is a diagonal matrix, parametrized by a vector of parameters $q = [q_1, q_2, \dots, q_m]$, and is given by

$$\Delta = \text{diag}(q_1 I_1, q_2 I_2, \dots, q_m I_m), \tag{7}$$

where I_i is an identity matrix of size p_i . Of course, $\sum_i^m p_i = p$. The rectangle in which q lies is given by $\mathcal{Q}_{\text{init}} = [l_1, u_1] \times [l_2, u_2] \times \dots \times [l_m, u_m]$. Figure 1 shows a block diagram of our setup.

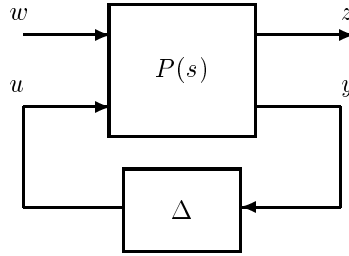


Fig. 1. System in standard form.

We also define

$$\begin{aligned}
P_{yu} &= C_y(sI - A)^{-1}B_u + D_{yu}, \\
P_{yw} &= C_y(sI - A)^{-1}B_w + D_{yw}, \\
P_{zu} &= C_z(sI - A)^{-1}B_u + D_{zu}, \\
P_{zw} &= C_z(sI - A)^{-1}B_w + D_{zw}.
\end{aligned} \tag{8}$$

P_{yu} is the (open-loop, *i.e.* $\Delta = 0$) transfer matrix from u to y and so on.

Eliminating u and y from equations (6) yields the closed-loop system

equations:

$$\begin{aligned}
\dot{x} &= (A + B_u \Delta (I - D_{yu} \Delta)^{-1} C_y) x + \\
&\quad (B_u \Delta (I - D_{yu} \Delta)^{-1} D_{yw} + B_w) w, \\
z &= (C_z + D_{zu} \Delta (I - D_{yu} \Delta)^{-1} C_y) x + \\
&\quad (D_{zu} \Delta (I - D_{yu} \Delta)^{-1} D_{yw} + D_{zw}) w.
\end{aligned} \tag{9}$$

For convenience, we let

$$\begin{aligned}
\mathcal{A}(q) &= A + B_u \Delta (I - D_{yu} \Delta)^{-1} C_y, \\
\mathcal{B}(q) &= B_u \Delta (I - D_{yu} \Delta)^{-1} D_{yw} + B_w, \\
\mathcal{C}(q) &= C_z + D_{zu} \Delta (I - D_{yu} \Delta)^{-1} C_y, \\
\mathcal{D}(q) &= D_{zu} \Delta (I - D_{yu} \Delta)^{-1} D_{yw} + D_{zw}.
\end{aligned}$$

We note that the entries of $\mathcal{A}(q)$, $\mathcal{B}(q)$, $\mathcal{C}(q)$ and $\mathcal{D}(q)$ are *rational functions* of the parameter vector q .

The closed-loop transfer matrix from w to z is denoted $P_{cl}(q)$ and is given by

$$P_{cl}(q) = P_{zw} + P_{zu} \Delta (I - P_{yu} \Delta)^{-1} P_{yw}. \tag{10}$$

Loosely speaking, the above framework describes linear systems with fixed, unknown gains that lie in intervals. $P(s)$ is often called the *open-loop system*, and corresponds to the case when all the gains are set to zero. Δ , on one hand, might represent unknown parameters, in which case it has the interpretation of a perturbation to a linear system; on the other hand, the entries of Δ might represent gains which a designer may choose at will, in which case Δ has the interpretation of a design variable. w consists of the inputs to the system and z the outputs, and the closed loop transfer matrix $P_{cl}(q)$ consists of all transfer functions of interest.

A number of parameter problems in control can be addressed in this setting: study of Kharitonov polynomials and interval matrices, parametric controller design etc. (We refer the reader to [18] for how the first two problems can be cast into our setup.) Roughly speaking, any system with state-space matrices whose entries are rational functions of the uncertain parameters can be considered in our framework². However, we wish to emphasize the restriction that the uncertain parameters lie in a rectangle;

²The precise condition is that none of the rational functions that make up the state space entries have a singularity at $q = 0$.

we cannot, for example, directly consider a situation where the uncertain parameters lie in a general polytope or an ellipsoid.

There are a number of important quantities associated with systems described by (6). We describe some of them below.

B. Well-posedness

One of the most fundamental properties of the feedback system in Figure 1 is well-posedness: We say that the system is *well-posed* if it is well-defined for all $q \in \mathcal{Q}_{\text{init}}$, that is, if

$$\det(I - D_{yu}\Delta) \neq 0 \quad \text{for all } q \in \mathcal{Q}_{\text{init}}. \quad (11)$$

Condition (11) is necessary and sufficient for equations (9) to be well-defined for all $q \in \mathcal{Q}_{\text{init}}$. Obviously, this condition is equivalent to none of the rational matrices $\mathcal{A}(q)$, $\mathcal{B}(q)$, $\mathcal{C}(q)$ and $\mathcal{D}(q)$ having singularities in $\mathcal{Q}_{\text{init}}$.

If Δ has the interpretation of an uncertainty, then the question of well-posedness is one of robustness analysis, where one asks if there is any choice of parameters that makes $I - D_{yu}\Delta$ singular. If Δ , on the other hand, contains design parameters, then the question is whether there is any choice of parameters that makes $I - D_{yu}\Delta$ nonsingular. Of course, this question has a particularly simple answer: $I - D_{yu}\Delta$ is either nonsingular for almost all values of q or it is singular for *every* q — this follows from the fact that $\det(I - D_{yu}\Delta)$ is a rational function of q — and therefore very simple algorithms can be devised to answer the design question.

The answer to the question of well-posedness is a Boolean “yes” or “no”. One may also define a quantitative measure of well-posedness as, say, the condition number of $I - D_{yu}\Delta$. In that case, the robustness analysis question is the worst (*i.e.* largest) possible condition number of $I - D_{yu}\Delta$ over all possible Δ . The corresponding design problem would seek the choice of parameters that minimizes the condition number of $I - D_{yu}\Delta$.

For simplicity, we will henceforth assume that the system in Figure 1 is well-posed over $\mathcal{Q}_{\text{init}}$.

C. Stability degree

The *stability degree* of an LTI system $\dot{x} = Fx$ where $F \in \mathbf{R}^{n \times n}$ is denoted $\lambda_{\text{sd}}(F)$ and defined as

$$\lambda_{\text{sd}}(F) = - \max_{1 \leq i \leq n} \text{Re } \lambda_i(F).$$

The stability degree gives the slowest possible decay rate of the solutions of the system and thus may be regarded as a stability measure for the system:

$$\lambda_{\text{sd}}(F) = \sup \left\{ \alpha \mid \lim_{t \rightarrow \infty} e^{\alpha t} x(t) = 0 \text{ whenever } \dot{x} = Fx \right\}.$$

We note that the stability degree equals the negative of the largest Lyapunov exponent of the system [19].

For a linear system, a “large” stability degree means that the solutions decay “fast”. Based on this observation, the stability degree may be used to define a robustness measure for an uncertain system, or a design goal for parametric design. We describe these below.

1. Minimum Stability Degree (\mathcal{D}_{min})

A robustness measure for the parameter-dependent system in Figure 1 is the *minimum stability degree* (\mathcal{D}_{min}), the smallest or worst-case stability degree of the system over all possible value of the parameters:

$$\mathcal{D}_{\text{min}}(\mathcal{Q}_{\text{init}}) = \min_{q \in \mathcal{Q}_{\text{init}}} \lambda_{\text{sd}} \mathcal{A}(q).$$

\mathcal{D}_{min} and other quantities that we describe below are functions of the system given by equations (6); we will not show this dependence explicitly for convenience.

The system in Figure 1 is robustly stable, *i.e.* the eigenvalues of the system have negative real parts for all Δ if and only if its \mathcal{D}_{min} is positive. Moreover, \mathcal{D}_{min} gives the worst-case decay rate of the solutions $x(t)$ of the state equations:

$$\mathcal{D}_{\text{min}}(\mathcal{Q}_{\text{init}}) = \min_{q \in \mathcal{Q}_{\text{init}}} \sup \left\{ \alpha \mid \lim_{t \rightarrow \infty} x(t) e^{\alpha t} = 0 \text{ whenever } \dot{x} = \mathcal{A}(q)x \right\}.$$

From the point of view of robustness analysis, it is desirable to have a large, positive \mathcal{D}_{min} . This ensures that the states decay fast, irrespective of the uncertain parameter vector q and the initial condition.

2. Maximum Stability Degree (\mathcal{D}_{\max})

Treating the stability degree of the system as a design objective, one may define the *maximum stability degree* (\mathcal{D}_{\max}) as

$$\mathcal{D}_{\max}(\mathcal{Q}_{\text{init}}) = \max_{q \in \mathcal{Q}_{\text{init}}} \lambda_{\text{sd}} \mathcal{A}(q).$$

Thus, with the components of q regarded as design parameters, the problem is one of finding the set of parameters that maximizes the slowest possible decay rate of the solutions to the system, that is,

$$\mathcal{D}_{\max}(\mathcal{Q}_{\text{init}}) = \max_{q \in \mathcal{Q}_{\text{init}}} \sup \left\{ \alpha \mid \lim_{t \rightarrow \infty} x(t) e^{\alpha t} = 0 \text{ whenever } \dot{x} = \mathcal{A}(q)x \right\}.$$

Computing \mathcal{D}_{\max} checks *stabilizability*: There exists a choice of parameters that stabilizes the system if and only if \mathcal{D}_{\max} is positive.

D. \mathbf{H}_{∞} norm

For the system in Figure 1, another quantity of interest is $\|P_{\text{cl}}\|_{\infty}$, the closed-loop \mathbf{H}_{∞} norm from w to z (see equation (10)), where $\|\cdot\|_{\infty}$ refers to the \mathbf{H}_{∞} norm:

$$\|G\|_{\infty} = \sup_{\text{Re } s > 0} \bar{\sigma}(G(s)).$$

$\|P_{\text{cl}}\|_{\infty}$ is just the *root mean square gain* (RMS-gain) of the system between the input w and the output z , *i.e.*,

$$\|P_{\text{cl}}\|_{\infty} = \max_{w(t) \neq 0} \frac{\|z\|_{\text{rms}}}{\|w\|_{\text{rms}}},$$

where the RMS value of a signal $w(t)$ is defined as

$$\|w\|_{\text{rms}} = \left(\lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T w(t)^2 dt \right)^{1/2},$$

provided the limit exists.

Often w has the interpretation of a disturbance and z , that of some error signal; it is then desirable to have $\|P_{\text{cl}}\|_{\infty}$ small.

1. Maximum \mathbf{H}_∞ norm ($\mathcal{H}_{\infty,\max}$)

With q regarded as an uncertain parameter vector, we define the *maximum \mathbf{H}_∞ norm* from w to z (denoted $\mathcal{H}_{\infty,\max}$) for the system in Figure 1 as

$$\mathcal{H}_{\infty,\max}(\mathcal{Q}_{\text{init}}) = \max_{q \in \mathcal{Q}_{\text{init}}} \|P_{\text{cl}}(q)\|_\infty.$$

Thus $\mathcal{H}_{\infty,\max}$ is the worst-case RMS gain of the system from w to z over all possible parameters.

We note that system in Figure 1 is robustly stable if and only if $\mathcal{H}_{\infty,\max}$ is finite; if the system is unstable for some choice of parameters q , then $\|P_{\text{cl}}(q)\|_\infty = \infty$. Moreover, $\mathcal{H}_{\infty,\max}$ then serves as a measure of robust stability: A smaller $\mathcal{H}_{\infty,\max}$ means a “more robustly stable” system.

2. Minimum \mathbf{H}_∞ norm ($\mathcal{H}_{\infty,\min}$)

In contrast with $\mathcal{H}_{\infty,\max}$, we may regard q as a design parameter and ask what choice of parameters yields the smallest possible RMS-gain between w and z . This is the the *minimum \mathbf{H}_∞ norm* from w to z (denoted $\mathcal{H}_{\infty,\min}$) in Figure 1:

$$\mathcal{H}_{\infty,\min}(\mathcal{Q}_{\text{init}}) = \min_{q \in \mathcal{Q}_{\text{init}}} \|P_{\text{cl}}(q)\|_\infty.$$

Clearly, the system is stabilizable if and only if $\mathcal{H}_{\infty,\min}$ is finite.

E. \mathbf{H}_2 norm

We consider finally the \mathbf{H}_2 norm of the closed-loop transfer matrix from w to z , which, for a stable, strictly proper³ transfer matrix P_{cl} is defined as

$$\|P_{\text{cl}}\|_2 = \left(\text{Tr} \frac{1}{2\pi} \int_{-\infty}^{\infty} P_{\text{cl}}(j\omega) P_{\text{cl}}(j\omega)^* d\omega \right)^{1/2}.$$

$\|P_{\text{cl}}\|_2 = \infty$ if P_{cl} is either unstable or not strictly proper. $\|P_{\text{cl}}\|_2$ has the interpretation of the RMS value of the output z when the components of the input w are independent white noises with unit power spectral density.

³ P_{cl} is said to be strictly proper if $P_{\text{cl}}(\infty) = 0$.

1. Maximum \mathbf{H}_2 norm ($\mathcal{H}_{2,\max}$)

When q represents uncertainties, we define the *maximum \mathbf{H}_2 norm* from w to z (denoted $\mathcal{H}_{2,\max}$) in Figure 1 as

$$\mathcal{H}_{2,\max}(\mathcal{Q}_{\text{init}}) = \max_{q \in \mathcal{Q}_{\text{init}}} \|P_{\text{cl}}(q)\|_2.$$

Thus $\mathcal{H}_{2,\max}$ is the worst-case RMS value of z when w is driven by white noise whose power spectral density is the identity.

We note that $\mathcal{H}_{2,\max}$ is finite if and only if the system in Figure 1 is robustly stable and strictly proper for all $q \in \mathcal{Q}_{\text{init}}$. Moreover, a smaller $\mathcal{H}_{2,\max}$ means that the output is less susceptible to noises at the input; thus $\mathcal{H}_{2,\max}$ serves a measure of robust performance.

We also note that the computation of the maximum total state covariance of a system driven by white noise can be cast as a problem of computing $\mathcal{H}_{2,\min}$ [20].

2. Minimum \mathbf{H}_2 norm ($\mathcal{H}_{2,\min}$)

The design problem corresponding to the \mathbf{H}_2 norm measuring the size of the closed-loop transfer matrix is that of finding the minimum \mathbf{H}_2 norm ($\mathcal{H}_{2,\min}$). This is the choice of parameters q that minimizes the closed-loop \mathbf{H}_2 norm:

$$\mathcal{H}_{2,\min}(\mathcal{Q}_{\text{init}}) = \min_{q \in \mathcal{Q}_{\text{init}}} \|P_{\text{cl}}(q)\|_2.$$

F. Remarks on Complexity

We now make some general observations regarding the computation of the quantities defined so far. First, we note that the fundamental problem of well-posedness (*cf.* equation (11)) is NP-hard in general [21, 22, 23, 24]; roughly speaking, in the worst case, the number of computations required to establish well-posedness increases more than polynomially with the problem size m (which is the size of D_{yu} and Δ). This makes it likely that *any* algorithm that computes any of the six quantities above to within some fixed accuracy also requires, in the worst case, computations that increase more than polynomially with the problem size. This conjecture is especially interesting in light of the fact that maximum number of branch and bound algorithm iterations increases exponentially with m for a given accuracy

(see Subsection II B); therefore, if the conjecture were true, no algorithm would perform substantially better than a branch and bound algorithm on these problems.

We know of no existing algorithms that compute any of the quantities \mathcal{D}_{\min} , \mathcal{D}_{\max} , $\mathcal{H}_{\infty,\max}$, $\mathcal{H}_{\infty,\min}$, $\mathcal{H}_{2,\max}$ or $\mathcal{H}_{2,\min}$ for the general framework in equations (6). However, much work has been done in special cases. Kharitonov’s theorem [25, 26] gives a very efficient method for determining robust stability for the special case when the coefficients of the characteristic polynomial of $\mathcal{A}(q)$ are just the uncertain parameters q_i . Kharitonov’s theorem has been extended to cover the case in which the characteristic polynomial is an affine function of q [27, 28]. Another problem that may be considered in our setup is the study of interval matrices [29, 30, 31].

In [32], Anderson *et al.* observe that the robust stability question is *decidable*, which means that by evaluating a finite number of polynomial functions of the input data (the entries of the state-space matrices, and the l_i , u_i), we can determine whether the system is robustly stable. It turns out, however, that these decision procedures involve an extraordinarily large number of polynomials, even for small systems with few parameters. Moreover the number of polynomials that need to be checked grows very rapidly (more than exponentially) with system size and number of parameters (see also [33]).

Though most of the methods described above do not directly consider the computation of the six quantities described in our framework, they do provide useful *bounds* for these quantities. Local optimization procedures provide lower (upper) bounds for the maximization (minimization) problems; there exist many analytical techniques (small gain theorem, Lyapunov theory based methods etc.) that yield bounds in the other direction. We will use some of these techniques to derive bounds for \mathcal{D}_{\min} , \mathcal{D}_{\max} , $\mathcal{H}_{\infty,\max}$, $\mathcal{H}_{\infty,\min}$, $\mathcal{H}_{2,\max}$ and $\mathcal{H}_{2,\min}$ in the next section.

IV. COMPUTATION OF BOUNDS

A. A Loop Transformation

Before we go on to describing the computation of bounds, we describe a loop transformation that converts the problem of finding bounds over an

arbitrary rectangle to that of finding bounds over the cube $\mathcal{U} = [-1, 1]^m$. We refer the reader to [34] for a complete discussion of loop transformations.

The loop transformation is best explained through Figure 2, where the symbols $\tilde{H}(s)$ and $\tilde{\Delta}$ refer to the “new” system and the “normalized” perturbation.

The loop transformation can be interpreted as translating \mathcal{Q} to the origin, and then scaling it to the cube $[-1, 1]^m$.

$$K = \text{diag}\left(\frac{u_1 + l_1}{2}I_1, \frac{u_2 + l_2}{2}I_2, \dots, \frac{u_m + l_m}{2}I_m\right),$$

$$F = \text{diag}\left(\frac{u_1 - l_1}{2}I_1, \frac{u_2 - l_2}{2}I_2, \dots, \frac{u_m - l_m}{2}I_m\right)$$

are the matrices that accomplish this.

A state-space representation of the loop-transformed system $\tilde{P}(s)$ is given by $\{\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}\}$, where

$$\tilde{A} = A + B_u T K C_y; \quad \tilde{B} = \begin{bmatrix} \underbrace{B_w + B_u T K D_{yw}}_{\tilde{B}_z} & \underbrace{B_u T F^{\frac{1}{2}}}_{\tilde{B}_u} \end{bmatrix};$$

$$\tilde{C} = \begin{bmatrix} \underbrace{\tilde{C}_z}_{C_z + D_{zu} T K C_y} \\ \underbrace{F^{\frac{1}{2}}(I + D_{yu} T K) C_y}_{\tilde{C}_y} \end{bmatrix}; \quad (12)$$

$$\tilde{D} = \begin{bmatrix} \underbrace{\tilde{D}_{zw}}_{D_{zw} + D_{zu} T K D_{yw}} & \underbrace{\tilde{D}_{zu}}_{D_{zu} T K F^{\frac{1}{2}}} \\ \underbrace{F^{\frac{1}{2}}(I + D_{yu} T K) D_{yw}}_{\tilde{D}_{yw}} & \underbrace{F^{\frac{1}{2}} D_{yu} T F^{\frac{1}{2}}}_{\tilde{D}_{yu}} \end{bmatrix}.$$

$T = (I - K D_{yu})^{-1}$, and I is the identity matrix of appropriate size. We remind the reader of the assumption that the system is well-posed, which guarantees that $(I - K D_{yu})$ is invertible.

We make the obvious but important remark that all the six quantities we have defined in the previous subsection remain invariant under the loop transformation. Thus any bounds for these quantities computed with the loop transformed system in Figure 2 are valid for the system in Figure 1.

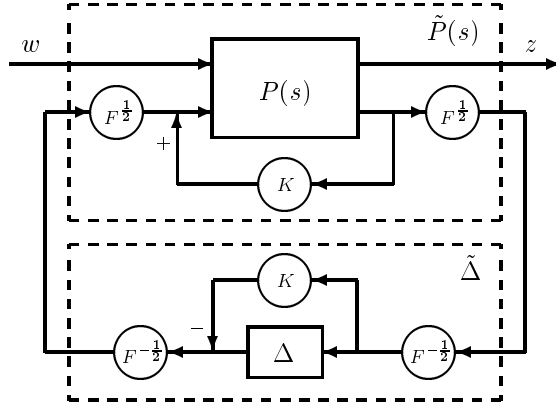


Fig. 2. Loop Transformation.

B. Bounds for \mathcal{D}_{\min}

Computation of \mathcal{D}_{\min} is a global minimization problem and we will therefore use Algorithm I of Section II. In the notation used to describe Algorithm I, we have $f(q) = \lambda_{\text{sd}}(\mathcal{A}(q))$ and $\Phi_{\min}(\mathcal{U}) = \mathcal{D}_{\min}(\mathcal{U})$. We now need to compute a lower bound $\Phi_{\text{lb}}(\mathcal{U})$ and an upper bound $\Phi_{\text{ub}}(\mathcal{U})$ for $\mathcal{D}_{\min}(\mathcal{U})$.

Upper bounds

One simple upper bound on \mathcal{D}_{\min} over the cube \mathcal{U} is just the stability degree of the system evaluated at the *midpoint* of the cube. Thus:

$$\Phi_{\text{ub}}(\mathcal{U}) = \lambda_{\text{sd}}(\mathcal{A}(0)) = \lambda_{\text{sd}}(A). \quad (13)$$

This upper bound can be improved quite easily by local optimization methods such as a gradient search (see any text on optimization, [35] for example). Another heuristic is to set the upper bound over \mathcal{U} to be equal to the minimum of the stability degrees over the vertices and the center:

$$\Phi_{\text{ub}}(\mathcal{U}) = \min_{q \in \{\text{vertices of } \mathcal{U}\} \cup \{0\}} \lambda_{\text{sd}}(\mathcal{A}(q)). \quad (14)$$

There are two justifications for this heuristic: First, in many special cases, \mathcal{D}_{\min} is always achieved at a vertex; secondly, since the number of local minima for \mathcal{D}_{\min} is finite, as the size of \mathcal{Q} (before loop-transformation)

becomes small, \mathcal{D}_{\min} is more likely to be achieved on the boundary of \mathcal{Q} than in the interior; therefore the vertices are more likely to yield better bounds on \mathcal{D}_{\min} than the center.

Lower bounds

Computation of lower bounds is a little more involved. It is based on the application of a generalized *small gain theorem* (SGT) which states that for every $q \in \mathcal{U}$, the (closed-loop) system in Figure 1 has the same number of stable poles as $P(s)$, the open-loop system, provided $\|P_{yu}\|_{\mathbf{L}_\infty} < 1$, where

$$\|H\|_{\mathbf{L}_\infty} = \sup_{\omega \in \mathbf{R}} \bar{\sigma}(H(j\omega))$$

is the \mathbf{L}_∞ norm of the transfer matrix H . This theorem is a simple extension of the conventional small gain theorem which can be found, for example, in [34]. Thus, if $\|P_{yu}\|_{\mathbf{L}_\infty} < 1$,

$$A \text{ stable} \implies \mathcal{D}_{\min}(\mathcal{U}) > 0, \quad (15)$$

and

$$A \text{ unstable} \implies \mathcal{D}_{\max}(\mathcal{U}) < 0. \quad (16)$$

Thus, if A is stable and $\|P_{yu}\|_{\mathbf{L}_\infty} < 1$, (15) gives a lower bound of zero for $\mathcal{D}_{\min}(\mathcal{U})$. Otherwise, we may conclude nothing.

In order to derive a better lower bound on $\mathcal{D}_{\min}(\mathcal{U})$, we consider the exponentially time-weighted system

$$\begin{aligned} \dot{x}_e &= (A + \alpha I)x_e + B_u u + B_w w, & x_e(0) &= x_0, \\ y &= C_y x_e + D_{yu} u + D_{yw} w, \\ z &= C_z x_e + D_{zu} u + D_{zw} w, \\ u &= \Delta y, \end{aligned} \quad (17)$$

where $\alpha < \lambda_{\text{sd}}(A)$. Note that $(A + \alpha I)$ is guaranteed to be *stable*. The solutions of equations (17) and (6) are simply related by $x_e(t) = e^{\alpha t} x(t)$, and therefore we conclude that

$$\mathcal{D}_{\min}(\mathcal{U}) > \alpha, \quad \text{whenever } \|P_{yu}\|_{\infty, \alpha} < 1,$$

where

$$\|H\|_{\infty, \alpha} = \sup_{\{s = -\alpha + j\omega \mid \omega \in \mathbf{R}\}} \bar{\sigma}(H(s))$$

is the α -shifted \mathbf{L}_∞ norm of H [36]. Therefore, we define $\Phi_{\text{lb}}(\mathcal{U})$ as

$$\Phi_{\text{lb}}(\mathcal{U}) = \sup\{\alpha \mid \alpha < \lambda_{\text{sd}}(A), \|P_{yu}\|_{\infty, \alpha} < 1\}. \quad (18)$$

Note that if $\|P_{yu}\|_{\infty, \alpha} \geq 1$ for all α , then $\Phi_{\text{lb}}(\mathcal{U}) = -\infty$. (This occurs only if $\bar{\sigma}(D_{yu}) \geq 1$.)

The condition in (18) is readily checked by forming an appropriate Hamiltonian matrix and checking its eigenvalues (see [37, 38]); a simple bisection can be used to compute Φ_{lb} .

We note that our procedure for computing $\Phi_{\text{lb}}(\mathcal{U})$ is just an application of the ‘‘shifted circle criterion’’ (Anderson and Moore [39]).

Very often, the lower bound above turns out to be too conservative; the reason for this lies in the application of the SGT in (15) (and (16)). The very special structure of Δ (real, diagonal, and with possibly many repeated entries) has been ignored, and this means that the SGT may be extremely conservative in guaranteeing the robust stability of the exponentially weighted closed-loop system. Eliminating or reducing this conservatism of the SGT that arises due to ‘‘structured perturbations’’ is a major area of research in itself (structured singular value [40], scaling or the scaled singular value [41]). The scaled singular value (which we will abbreviate as SSV) is directly relevant to our problem, and we will give a brief and informal discussion here.

The motivation for the SSV arises from the following simple observation: The system shown in Figure 3 is equivalent to the system in Figure 1 (in the sense that the solution to the closed-loop state equations as well as the closed-loop transfer matrices from w to z are equal) for all nonzero $\alpha \in \mathbf{C}$ and invertible matrices⁴ $S_\Delta \in \mathbf{C}^{p \times p}$ such that $S_\Delta \Delta = \Delta S_\Delta$. In other words, the structure of Δ makes the closed-loop system invariant under the scaling of the open loop transfer matrix described by

$$\begin{bmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{bmatrix} \longrightarrow \begin{bmatrix} \alpha I_1 & 0 \\ 0 & S_\Delta \end{bmatrix} \begin{bmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{bmatrix} \begin{bmatrix} \alpha I_2 & 0 \\ 0 & S_\Delta \end{bmatrix}^{-1},$$

where $\alpha \in \mathbf{C}$, I_1 and I_2 are identity matrices of sizes n_o and n_i respectively, and $S_\Delta \in \mathbf{C}^{p \times p}$ is an invertible matrix that commutes with Δ .

⁴Both α and S_Δ can be functions of s rather than simply constants, but we will not consider this more general setting here.

We let

$$S_{\text{left}} = \begin{bmatrix} \alpha I_1 & 0 \\ 0 & S_\Delta \end{bmatrix} \quad \text{and} \quad S_{\text{right}} = \begin{bmatrix} \alpha I_2 & 0 \\ 0 & S_\Delta \end{bmatrix}.$$

S_{left} and S_{right} are referred to as the *left* and *right scalings*.

The set of matrices that commute with Δ is denoted \mathbf{S}_Δ :

$$\mathbf{S}_\Delta = \{ S_\Delta \mid S_\Delta \Delta = \Delta S_\Delta, S_\Delta \in \mathbf{C}^{p \times p} \}.$$

\mathbf{S}_Δ determines the set of left and right scalings, denoted \mathbf{S}_{left} and $\mathbf{S}_{\text{right}}$ respectively, that leave the closed-loop system invariant:

$$\mathbf{S}_{\text{left}} = \left\{ S_{\text{left}} \mid S_{\text{left}} = \begin{bmatrix} \alpha I_1 & \\ & S_\Delta \end{bmatrix}, \alpha \in \mathbf{C}, S_\Delta \in \mathbf{S}_\Delta \right\}, \quad (19)$$

$$\mathbf{S}_{\text{right}} = \left\{ S_{\text{right}} \mid S_{\text{right}} = \begin{bmatrix} \alpha I_2 & \\ & S_\Delta \end{bmatrix}, \alpha \in \mathbf{C}, S_\Delta \in \mathbf{S}_\Delta \right\}. \quad (20)$$

It is not hard to see that for our case, every $S_\Delta \in \mathbf{S}_\Delta$ is of the form

$$S_\Delta = \begin{bmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_p \end{bmatrix},$$

where $D_i \in \mathbf{C}^{p_i \times p_i}$ and invertible, $i = 1, 2, \dots, p$.

In the computation of a lower bound for \mathcal{D}_{\min} through equation (18), the transfer matrix from u to y is the only one of interest. The scaling of $P(s)$ to $S_{\text{left}}P(s)S_{\text{right}}^{-1}$ results in the scaling of $P_{yu}(s)$ to $S_\Delta P_{yu}(s)S_\Delta^{-1}$. Then we may define a new, possibly improved lower bound as

$$\Phi_{\text{lb}}(\mathcal{U}) = \sup \left\{ \alpha \mid \alpha < \lambda_{\text{sd}}(A), \inf_{S_\Delta \in \mathbf{S}_\Delta} \|S_\Delta P_{yu} S_\Delta^{-1}\|_{\infty, \alpha} < 1 \right\}. \quad (21)$$

There are many heuristics for performing the optimization in equation (21). We will not describe any of them here. However, we note that for a fixed α , computation of the scaling that minimizes $\|S_\Delta P_{yu} S_\Delta^{-1}\|_{\infty, \alpha}$ can be formulated as a quasi-convex optimization problem (see [36], chapters 13 and 14), and therefore can be performed by effective methods. Scaling a *constant* matrix (as opposed to a transfer matrix) optimally is a

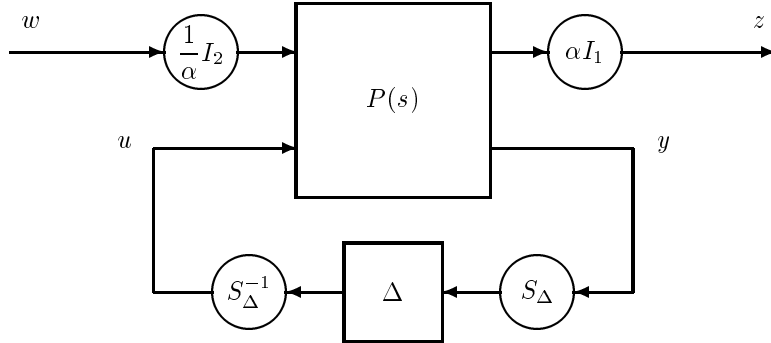


Fig. 3. The standard form with scaling. $\alpha \in \mathbf{C}$, I_1 and I_2 are identity matrices of sizes n_i and n_o respectively, $S_\Delta \in \mathbf{C}^{p \times p}$ is invertible and commutes with Δ .

well studied problem. A (by no means exhaustive) list of references is [42, 43, 44, 45, 46, 47, 48].

Note that the SSV has ignored the fact that Δ is real. This aspect has been addressed in a recent work that accounts for real perturbations [49]. The results therein may be incorporated into the bounds in (18) to improve the lower bound further.

For more details, including a proof that the bounds we have described for \mathcal{D}_{\min} satisfy conditions (R1) and (R2) stated at the beginning of Section II, see [18].

C. Bounds for \mathcal{D}_{\max}

Computation of \mathcal{D}_{\max} is a global maximization problem, so we will use Algorithm II of Section II. Our notation is then $f(q) = \lambda_{\text{sd}}(\mathcal{A}(q))$ and $\Psi_{\max}(\mathcal{U}) = \mathcal{D}_{\max}(\mathcal{U})$; we now describe the computation of a lower bound $\Psi_{\text{lb}}(\mathcal{U})$ and an upper bound $\Psi_{\text{ub}}(\mathcal{U})$ for $\mathcal{D}_{\max}(\mathcal{U})$.

Lower bounds

A simple lower bound on \mathcal{D}_{\max} over the cube \mathcal{U} is just the stability degree of the system evaluated at the *midpoint* of the cube. Thus:

$$\Psi_{\text{lb}}(\mathcal{U}) = \lambda_{\text{sd}}(\mathcal{A}(0)) = \lambda_{\text{sd}}(A). \quad (22)$$

The above lower bound can be improved by the same heuristic as in equation (14):

$$\Psi_{\text{lb}}(\mathcal{U}) = \max_{q \in \{\text{vertices of } \mathcal{U}\} \cup \{0\}} \lambda_{\text{sd}}(\mathcal{A}(q)). \quad (23)$$

Upper bounds

Our upper bound for $\mathcal{D}_{\text{max}}(\mathcal{U})$ is

$$\Psi_{\text{ub}}(\mathcal{U}) = \inf \{ \alpha \mid \alpha > \lambda_{\text{sd}}(A), \|P_{yu}\|_{\infty, \alpha} < 1 \}. \quad (24)$$

Note that if $\|P_{yu}\|_{\infty, \alpha} \geq 1$ for all α , then $\Phi_{\text{lb}}(\mathcal{U}) = \infty$.

This bound can be established using a derivation identical to that leading to the bound in equation (18). Instead, let us give a brief intuitive explanation of this bound.

The bound in equation (18) is just the largest amount of “anti-damping” α we may add to system in Figure (1) with the SGT proving robust stability. In contrast, the bound in equation (24) is just the smallest anti-damping α we must add to system in Figure (1) for the SGT to prove robust *instability*, *i.e.* to guarantee that the closed-loop system in Figure 1 is unstable for all $q \in \mathcal{U}$; the negative of the maximum real part of the eigenvalues of the closed-loop system can be no larger than this anti-damping.

This upper bound can be improved using the same scaling techniques as in equation (21):

$$\Psi_{\text{ub}}(\mathcal{U}) = \inf \left\{ \alpha \mid \alpha > \lambda_{\text{sd}}(A), \inf_{S_{\Delta} \in \mathbf{S}_{\Delta}} \|S_{\Delta} P_{yu} S_{\Delta}^{-1}\|_{\infty, \alpha} < 1 \right\}. \quad (25)$$

It can be shown that the bounds for \mathcal{D}_{max} satisfy conditions (R1') and (R2') stated in Section II.

D. Bounds for $\mathcal{H}_{\infty, \text{max}}$

We next describe the computation of a lower bound $\Psi_{\text{lb}}(\mathcal{U})$ and an upper bound $\Psi_{\text{ub}}(\mathcal{U})$ for $\mathcal{H}_{\infty, \text{max}}$.

Lower Bounds

A simple lower bound for $\mathcal{H}_{\infty, \text{max}}(\mathcal{U})$ is just the \mathbf{H}_{∞} norm of the closed-loop system with the parameter vector set to the midpoint of the parameter

region \mathcal{U} :

$$\Psi_{\text{lb}}(\mathcal{U}) = \|P_{\text{cl}}(0)\|_{\infty} = \|P_{zw}\|_{\infty}. \quad (26)$$

This bound may be improved using the same heuristic as in equations (14) and (23), by setting the lower bound of $\mathcal{H}_{\infty, \text{max}}$ over \mathcal{U} to be equal to the maximum of the \mathbf{H}_{∞} norm from w to z computed with the parameters assuming the values at the vertices and the center:

$$\Psi_{\text{lb}}(\mathcal{U}) = \max_{q \in \{\text{vertices of } \mathcal{U}\} \cup \{0\}} \|P_{\text{cl}}(q)\|_{\infty}. \quad (27)$$

Lower bounds

We now describe a simple scheme for computing an upper bound that is based on a small gain based robust stability condition due to Doyle [40] and Safonov [45] (see [36, p239-241]).

We define

$$P_{\beta} = \begin{bmatrix} \frac{P_{zw}}{\beta} & \frac{P_{zu}}{\sqrt{\beta}} \\ \frac{P_{yw}}{\sqrt{\beta}} & P_{yu} \end{bmatrix}, \quad (28)$$

where $\beta > 0$. Then

$$\|P_{\beta}\|_{\infty} < 1 \implies \sup_{\|\Delta\|_{\infty} \leq 1} \|[P_{zw} + P_{zu}\Delta(I - P_{yu}\Delta)^{-1}P_{yw}]\|_{\infty} < \beta.$$

Our upper bound is:

$$\Psi_{\text{ub}}(\mathcal{U}) = \inf \{ \beta \mid \|P_{\beta}\|_{\infty} < 1 \}, \quad (29)$$

with the convention that the infimum of a function over the empty set is infinity.

As with the lower bounds for \mathcal{D}_{min} , Ψ_{ub} may be rapidly computed using a bisection on Hamiltonian matrices [37, 38]. We may also use scaling to improve the upper bound for $\mathcal{H}_{\infty, \text{max}}$ in equation (29). The possibly improved upper bound is then given by

$$\Psi_{\text{ub}}(\mathcal{U}) = \inf \left\{ \beta \mid \inf_{S_{\text{left}} \in \mathbf{S}_{\text{left}}, S_{\text{right}} \in \mathbf{S}_{\text{right}}} \|S_{\text{left}} P_{\beta} S_{\text{right}}^{-1}\|_{\infty} < 1 \right\}, \quad (30)$$

where \mathbf{S}_{left} and $\mathbf{S}_{\text{right}}$ are given by equations (20) and (19).

We refer the reader to [50] for a proof that the bounds for $\mathcal{H}_{\infty, \text{max}}$ satisfy conditions (R1') and (R2') stated in Section II.

E. Bounds for $\mathcal{H}_{\infty, \min}$

We describe next the computation of an upper bound $\Phi_{\text{ub}}(\mathcal{U})$ and a lower bound $\Phi_{\text{lb}}(\mathcal{U})$ for $\Phi_{\min}(\mathcal{U}) = \mathcal{H}_{\infty, \min}(\mathcal{U})$. The bounds we present here make the possibly unrealistic assumption that system in Figure 1 is robustly stable. In terms of design, this requires the designer to apply the algorithm only over parameter ranges where the system is guaranteed to be stable.

Upper bounds

A simple upper bound for $\mathcal{H}_{\infty, \min}(\mathcal{U})$ is just the \mathbf{H}_{∞} norm of the closed-loop system with the parameter vector set to the midpoint of the parameter region \mathcal{U} :

$$\Phi_{\text{lb}}(\mathcal{U}) = \|P_{\text{cl}}(0)\|_{\infty} = \|P_{zw}\|_{\infty}. \quad (31)$$

This upper bound may be improved by employing the same heuristic as in equations (14), (23) and (27):

$$\Phi_{\text{lb}}(\mathcal{U}) = \min_{q \in \{\text{vertices of } \mathcal{U}\} \cup \{0\}} \|P_{\text{cl}}(q)\|_{\infty}. \quad (32)$$

Lower bound

If $\|P_{yu}\|_{\infty} < 1$, we may compute a lower bound based on simple norm inequalities:

$$\begin{aligned} \|P_{\text{cl}}(q)\|_{\infty} &= \|P_{zw} + P_{zu}\Delta(I - P_{yu}\Delta)^{-1}P_{yw}\|_{\infty} \\ &\geq \|P_{zw}\|_{\infty} - \frac{\|P_{zu}\|_{\infty}\|P_{yw}\|_{\infty}}{1 - \|P_{yu}\|_{\infty}}. \end{aligned} \quad (33)$$

If $\|P_{yu}\|_{\infty} \geq 1$, our lower bound is merely 0.

We refer the reader to [50] for more details, including a proof that the bounds for $\mathcal{H}_{\infty, \min}$ satisfy conditions (R1) and (R2) stated in Section II.

F. Bounds for $\mathcal{H}_{2, \max}$

We describe next the computation of an upper bound $\Psi_{\text{ub}}(\mathcal{U})$ and a lower bound $\Psi_{\text{lb}}(\mathcal{U})$ for $\Psi_{\max}(\mathcal{U}) = \mathcal{H}_{2, \max}(\mathcal{U})$. The bounds are based on the observation that \mathbf{H}_2 norms can be computed by solving Lyapunov equations (see for example, [36]). More precisely, if the stable, strictly proper transfer function $G(s)$ has a state-space realization $\{A, B, C\}$, then $\|G\|_2 =$

$\sqrt{\text{Tr}(CW_cC^T)}$, where $W_c = W_c^T > 0$ is the unique solution of the Lyapunov equation

$$AW_c + W_cA^T + BB^T = 0. \quad (34)$$

($\|G\|_2 = \infty$ if A is unstable or if G is not strictly proper.)

Lower bounds

A simple lower bound $\Psi_{\text{lb}}(\mathcal{U})$ for $\mathcal{H}_{2,\text{max}}$ is given by the \mathbf{H}_2 norm of the system evaluated with parameters assuming values at the center of \mathcal{U} :

$$\Psi_{\text{lb}}(\mathcal{U}) = \|\mathcal{P}_{\text{cl}}(0)\|_2. \quad (35)$$

This lower bound may be improved by simple heuristics as in equation (14) or by local optimization methods; in fact, there is a whole body of research on this problem. We refer the reader to the survey by Toivonen and Mäkilä [51].

Upper bound

Noting that (34) is just a system of linear equations, we may compute an upper bound $\Psi_{\text{ub}}(\mathcal{U})$ based on a simple perturbation analysis. We present the bound below, omitting tedious details.

In what follows, L_A is the Lyapunov operator associated with A , given by $A \otimes I + I \otimes A$ (“ \otimes ” is the Kronecker product, see for example, [52]), $N_1 = \min\{n_i, n\}$, $N_2 = \min\{n_o, n\}$. For convenience, we let

$$\begin{aligned} a &= \frac{\bar{\sigma}(B_u)\bar{\sigma}(C_y)}{1 - \bar{\sigma}(D_{yu})}, \\ b &= \frac{\bar{\sigma}(B_w)\bar{\sigma}(D_{yw})}{1 - \bar{\sigma}(D_{yu})}, \\ c &= \frac{\bar{\sigma}(C_y)\bar{\sigma}(D_{zu})}{1 - \bar{\sigma}(D_{yu})}, \\ d &= \frac{1}{\underline{\sigma}(L_A) - 2a} \left(2a\|W_c\|_F + b^2\sqrt{N_1} + 2b\|B_w\|_F \right), \\ e &= \left(2c\sqrt{N_2}\|C_zW_c\|_F + c^2N_2\|W_c\|_F + dN_2(\bar{\sigma}(C_z) + c)^2 \right)^{1/2}. \end{aligned} \quad (36)$$

W_c is the controllability Gramian of the system with $\Delta = 0$ and satisfies $AW_c^T + W_cA + B_wB_w^T = 0$.

Then, if $\bar{\sigma}(D_{yu}) < 1$ and $\underline{\sigma}(L_A) > 2a$,

$$\Psi_{\text{ub}}(\mathcal{U}) = ((\Psi_{\text{lb}}(\mathcal{U}))^2 + e^2)^{1/2}. \quad (37)$$

If $\bar{\sigma}(D_{yu}) \geq 1$ or $\underline{\sigma}(L_A) \leq 2a$, the upper bound is only ∞ .

It can be shown that the bounds for $\mathcal{H}_{2,\text{max}}$ satisfy conditions (R1') and (R2') of Section II.

G. Bounds for $\mathcal{H}_{2,\text{min}}$

An upper bound $\Phi_{\text{ub}}(\mathcal{U})$ and a lower bound $\Phi_{\text{lb}}(\mathcal{U})$ for $\Phi_{\text{min}}(\mathcal{U}) = \mathcal{H}_{2,\text{min}}(\mathcal{U})$ can be derived analogously to those for $\mathcal{H}_{2,\text{max}}$.

Upper bounds

An upper bound $\Phi_{\text{ub}}(\mathcal{U})$ for $\mathcal{H}_{2,\text{min}}$ is given by the \mathbf{H}_2 norm of the system evaluated with parameters assuming values at the center of \mathcal{U} :

$$\Phi_{\text{ub}}(\mathcal{U}) = \|P_{\text{cl}}(0)\|_2. \quad (38)$$

This upper bound may be improved by local optimization methods.

Lower bound

If $\bar{\sigma}(D_{yu}) < 1$, $\underline{\sigma}(L_A) > 2a$ and $\Phi_{\text{ub}}(\mathcal{U}) > e$,

$$\Phi_{\text{lb}}(\mathcal{U}) = \max \left\{ 0, ((\Phi_{\text{ub}}(\mathcal{U}))^2 - e^2)^{1/2} \right\}, \quad (39)$$

where a , b , c , d and e are given by equations (36). If $\bar{\sigma}(D_{yu}) \geq 1$ or $\underline{\sigma}(L_A) \leq 2a$ or $\Phi_{\text{ub}}(\mathcal{U}) \leq e$, the lower bound is merely 0. As with the lower bound for $\mathcal{H}_{\infty,\text{min}}$, this bound requires that the system (6) be robustly stable.

It can be shown that the bounds for $\mathcal{H}_{2,\text{max}}$ satisfy conditions (R1) and (R2) of Section II.

V. EXAMPLES

We consider a mechanical plant consisting of two masses connected by a spring with the left-hand mass driven by a force, as shown in Figure 4.

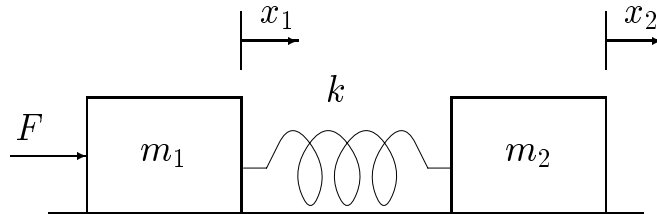


Fig. 4. The plant consisting of two masses connected by a spring.

We note that this example is so simple that many of the quantities associated with robustness analysis and design could be computed by hand; of course, the value of the methods that we present is in solving more complex problems that cannot be solved by hand or simple *ad hoc* procedures.

A. Examples for Analysis

We will first study the robustness properties of this system under variations of m_2 and k . More specifically, we will examine the common perception that an LQR-optimal state-feedback is “robust”. To this end, we let $[x_1 \ \dot{x}_1 \ x_2 \ \dot{x}_2]^T$ be the state vector, and employ a state-feedback law $F = -k_{\text{LQR}}x$ which is LQR optimal for the parameter values $m_1 = m_2 = k = 1$ (with weights $Q = I$, $\rho = 1$). In other words, $F(t) = -k_{\text{LQR}}x(t)$ is the input that minimizes the cost function

$$\int_0^{\infty} (x(t)^T x(t) + (F(t))^2) dt,$$

(irrespective of the initial condition) where the state equations are those of the system in Figure 4.

We will now study the robustness of this LQR-optimal closed-loop system with respect to variations in the parameters m_2 and k in a range between $2/3$ and $3/2$:

$$2/3 \leq m_2 \leq 3/2, \quad 2/3 \leq k \leq 3/2. \quad (40)$$

Thus, these physical parameters can vary over a range exceeding $2 : 1$. The closed-loop transfer function P_{cl} that we will examine is the closed-loop

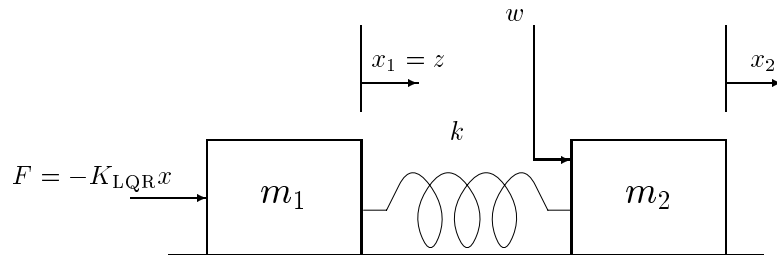


Fig. 5. The closed-loop system with LQR-optimal state feedback.

$x = [x_1 \ \dot{x}_1 \ x_2 \ \dot{x}_2]^T$ is the state vector. $m_1 = 1$ is fixed, and m_2 and k are uncertain parameters known to lie in $[2/3, 3/2]$. w is the force on the second mass and z is the position of the first mass.

transfer function from the force on the second mass to the position of the first mass (see Figure 5).

1. \mathcal{D}_{\min} of the system with the LQR-optimal controller

For this nominally LQR-optimal system, we compute the first measure of robustness, *i.e.* \mathcal{D}_{\min} over the parameter ranges in (40).

Figure 6 shows the results obtained from applying the branch and bound algorithm to this problem. The solid lines correspond to the bounds in (18) and (13), and the dashed lines to the “improved” bounds⁵ in (21) and (14). We note the following:

- *The system is robustly stable.* The system is robustly stable if and only if \mathcal{D}_{\min} is positive. The lower bound for \mathcal{D}_{\min} is positive at the end of 8 iterations using the lower bound (18) and at the end of 6 iterations using the lower bound (21), which establishes robust stability in each case.
- $0.1853 \leq \mathcal{D}_{\min} \leq 0.1862$. Thus the algorithm can prove that the decay rate of the state $x(t)$ is at least 0.1853, irrespective of the initial state x_0 and q . (For reference, the stability degree (λ_{sd}) of the

⁵The optimization problem to perform the scaling in equation (21) was not solved exactly; instead, a lower bound to the optimum was used.

nominal system is 0.3738. Thus the parameter variations can degrade the stability degree by a factor of about 2.)

- With the bounds in (18) and (13), the algorithm takes 307 iterations to determine \mathcal{D}_{\min} to within an absolute accuracy of 0.001, whereas with the improved bounds in (21) and (14), it takes only 176 iterations; this is offset by the increased computation accompanying the improved bounds.
- The algorithm returns the worst-case parameters $m_2 = 2/3$ and $k = 3/2$, which happen to lie on a vertex of the parameter box. This is not the case in general.
- Figure 6 also shows the number of rectangles in the partition as well as the pruned volume percentage as a function of iterations, for the two different sets of bounds. It is clear that the “improved” bounds do show improved performance.
- Figure 7 shows the parameter region under consideration at various stages of the algorithm. The algorithm can *prove* that \mathcal{D}_{\min} cannot be achieved outside the shaded region.

2. $\mathcal{H}_{\infty, \max}$ of the system with the LQR-optimal controller

We now turn to our second measure of robustness: We will compute the largest possible \mathbf{H}_{∞} norm of the closed-loop transfer function P_{cl} due to the parameter variations in (40).

Figure 8 shows the results from the branch and bound algorithm applied to the computation of $\mathcal{H}_{\infty, \max}$. The solid lines correspond to the bounds in (26) and (29), and the dashed lines once again to the “improved” bounds⁶ (in (27) and in (30)). We note the following:

- *The system is robustly stable.* Either upper bound (from (29) or (30)) is finite only if the system is robustly stable. The upper bound on $\mathcal{H}_{\infty, \max}$ from (29) is finite at the end of 8 iterations, (and the upper

⁶The optimal scaling problem in (30) was again not solved exactly; instead, an upper bound to the optimum was used.

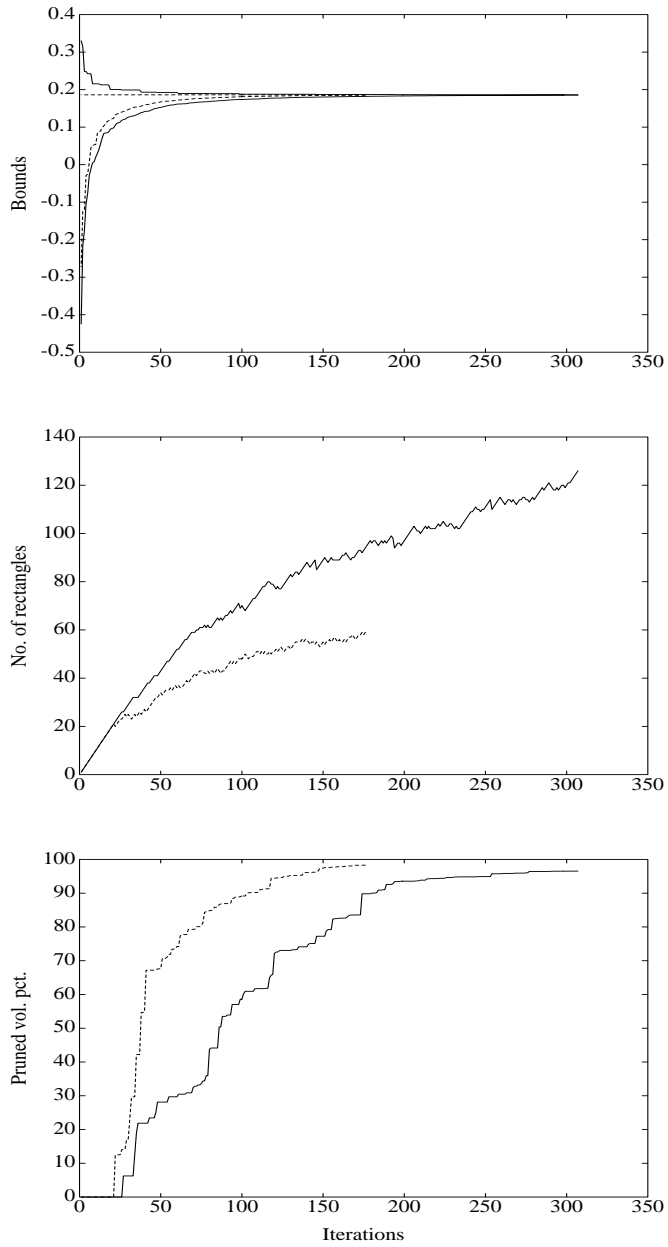
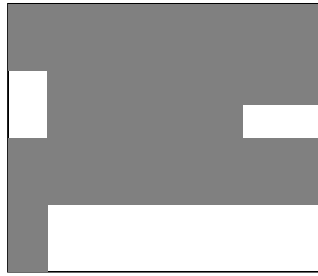
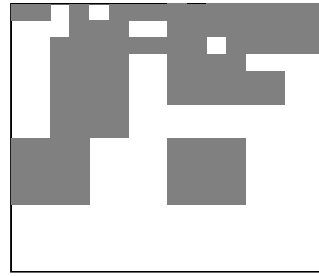


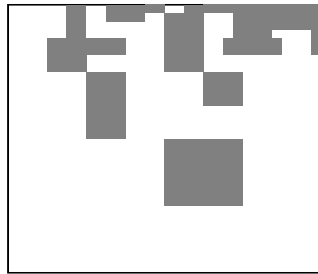
Fig. 6. Results from the branch and bound algorithm for \mathcal{D}_{\min} . Solid lines correspond to bounds from equations (18) and (13). Dotted lines correspond to bounds from equations (21) and (14).



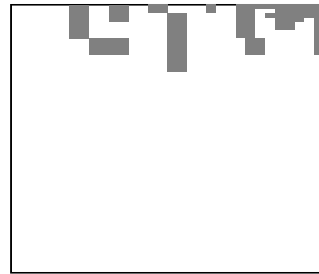
After 50 iterations



After 100 iterations



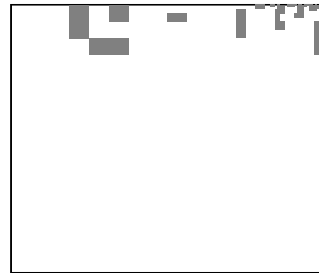
After 150 iterations



After 200 iterations



After 250 iterations



After 300 iterations

Fig. 7. The unpruned parameter region at various stages of the algorithm during the computation of \mathcal{D}_{\min} . The x - and y -coordinates are k and $1/m_2$ respectively. The algorithm can guarantee that \mathcal{D}_{\min} cannot be achieved outside the shaded region.

bound from (30) by 6 iterations), indicating that the system is robustly stable. Note that this is consistent with our observations from \mathcal{D}_{\min} computation.

- With bounds from (26) and (29), at the end of 122 iterations, the algorithm guarantees that $2.499 \leq \mathcal{H}_{\infty, \max} \leq 2.500$.

The performance with the bounds from (27) and (30) is considerably better; at the the end of only 40 iterations, the algorithm guarantees that $2.499 \leq \mathcal{H}_{\infty, \max} \leq 2.500$. For reference, the \mathbf{H}_{∞} norm from w to z for the nominal system is 1.008.

- The algorithm returns worst-case parameters $m_2 = 3/2$ and $k = 2/3$, which are different than the worst-case parameters for \mathcal{D}_{\min} ; thus, \mathcal{D}_{\min} and $\mathcal{H}_{\infty, \max}$ are inequivalent measures of robustness for our problem.
- Figure 8 also shows the number of rectangles in the partition as well as the pruned volume percentage as a function of iterations. It is again clear that the “improved” bounds do show improved performance.

3. $\mathcal{H}_{2, \max}$ of the system with the LQR-optimal controller

We finally consider the worst-case \mathbf{H}_2 norm of the closed-loop transfer function, our third measure of robustness.

The results from the branch and bound algorithm are shown in Figure 9 corresponding to the bounds in (35) and (37). We note that:

- *The system is robustly stable.* The upper bound from (37) is finite only if the system is robustly stable. The upper bound on $\mathcal{H}_{2, \max}$ from (37) is finite at the end of 467 iterations, indicating that the system is robustly stable.

The branch and bound algorithm applied towards computing \mathcal{D}_{\min} and $\mathcal{H}_{\infty, \max}$ establishes robust stability at the end of just 8 iterations, while it takes 467 iterations with $\mathcal{H}_{2, \max}$. The reason is that the bounds for \mathcal{D}_{\min} and $\mathcal{H}_{\infty, \max}$ use the same test for robust stability while the bound for $\mathcal{H}_{2, \max}$ uses a different, more conservative test.

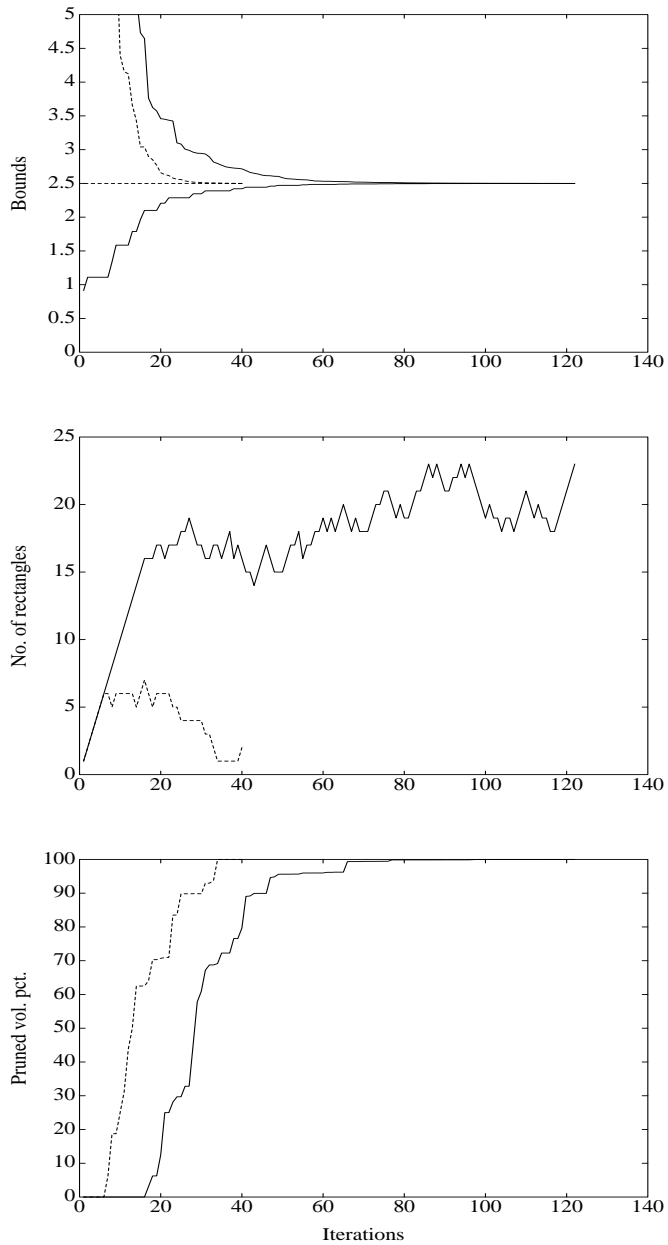


Fig. 8. Results from the branch and bound algorithm for $\mathcal{H}_{\infty, \max}$. Solid lines correspond to bounds from equations (26) and (29). Dotted lines correspond to bounds from equations (27) and (30).

- The algorithm takes 15,000 iterations to return $1.1304 \leq \mathcal{H}_{2,\max} \leq 1.1404$. (For reference, the nominal \mathbf{H}_2 norm from w to z is 0.6922.) Clearly, the progress here is much slower than with \mathcal{D}_{\min} or $\mathcal{H}_{\infty,\max}$. This is because the upper bound for $\mathcal{H}_{2,\max}$ is rather poor.

This illustrates an important point about the branch and bound algorithm: If the bounds are bad, the algorithm may take a very long time to converge.

- The algorithm returns the same set of worst-case parameters ($m_2 = 3/2$ and $k = 2/3$) as with $\mathcal{H}_{\infty,\max}$.

B. Examples for Design

We consider the system shown in Figure 4 with the parameters assuming the nominal values $m_1 = 1$, $m_2 = 1$ and $k = 1$; for this system, we consider the problem of designing a state feedback that does not use the position or velocity of the second mass m_2 , that is, we design a state feedback of the form $F = -(k_1 x_1 + k_2 \dot{x}_1)$. The closed-loop transfer function P_{cl} of interest is the transfer function from the force on the second mass w to the position of the first mass x_1 (see Figure 10).

We will restrict the state feedback gains k_1 and k_2 to satisfy

$$1/2 \leq k_1 \leq 1, \quad 1/2 \leq k_2 \leq 1, \quad (41)$$

The above parameter range has been chosen so that the system is stable for all values of k_1 and k_2 lying in it.

1. \mathcal{D}_{\max} -optimal incomplete state feedback

Our first design objective is to maximize \mathcal{D}_{\max} : We design an incomplete state feedback that maximizes the slowest decay rate of the system, with the gains restricted to lie in (41).

Figure 11 shows the result from the branch and bound algorithm applied to the computation of \mathcal{D}_{\max} . The solid lines correspond to the bounds in (22) and (24), and the dashed lines once again to the “improved” bounds⁷ (in (23) and in (25)). We note that:

⁷The optimal scaling problem in (25) was again not solved exactly; instead, an upper bound to the optimum was used.

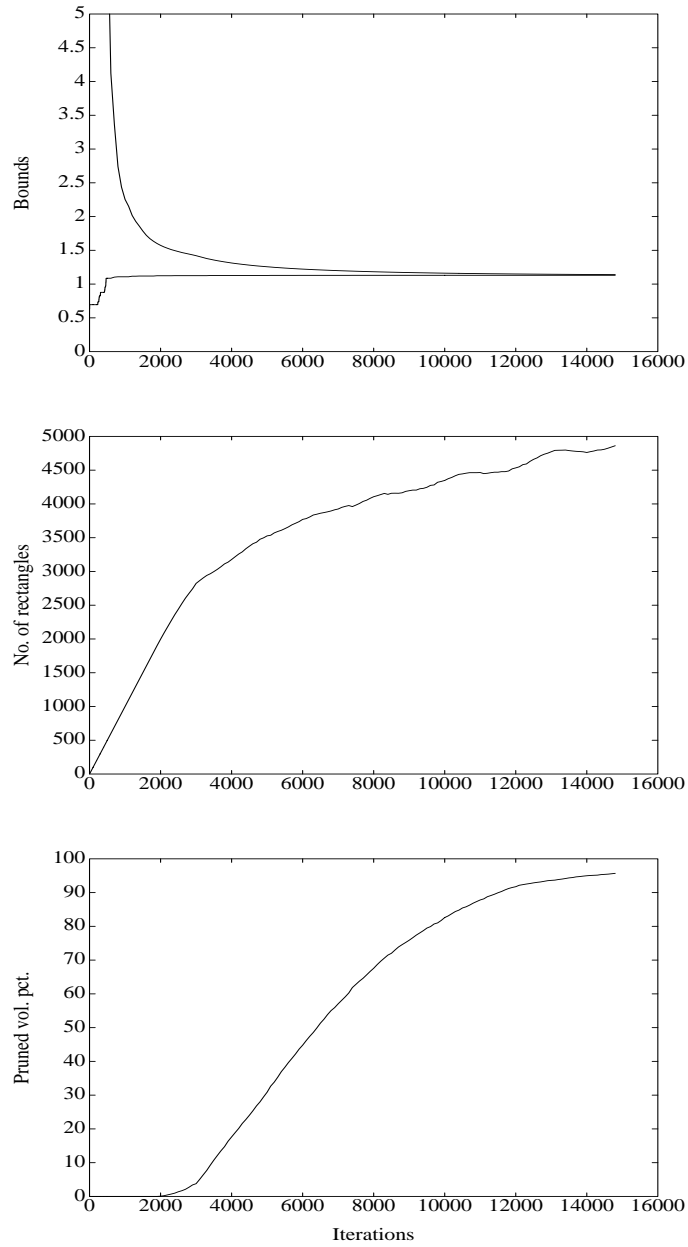


Fig. 9. Results from the branch and bound algorithm for $\mathcal{H}_{2,\max}$.

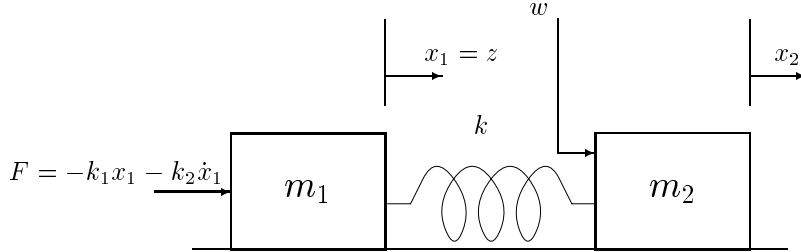


Fig. 10. The closed-loop system with incomplete state feedback $F = -k_1 x_1 - k_2 \dot{x}_1$, where k_1 and k_2 are the design parameters, each constrained to lie in $[1/2, 1]$. $[x_1 \ \dot{x}_1 \ x_2 \ \dot{x}_2]^T$ is the state vector. $m_1 = 1$, $m_2 = 1$ and $k = 1$ are fixed, w is the force on the second mass and z is the position of the first mass.

- Using the bounds from from (22) and (24), the branch and bound algorithm returns $0.2133 \leq \mathcal{D}_{\max} \leq 0.2141$ after 52 iterations. The algorithm takes 43 iterations to yield the same result using bounds (23) and (25).
- The upper bound from (25) performs only marginally better than the bound from (24). Thus, scaling does not help the upper bound much in this particular example.
- The algorithm returns the optimal gains $k_1 = 0.5$ and $k_2 = 1.0$, which corresponds to a vertex of the rectangle in (41).

2. $\mathcal{H}_{\infty, \min}$ -optimal incomplete state feedback

We next compute the optimal values of the gains k_1 and k_2 that yield the smallest possible \mathbf{H}_{∞} norm of the closed-loop transfer function between w , the force on the second mass and z , the position of the first mass, with the gains restricted as in (41).

Figure 12 shows the result from the branch and bound algorithm applied to the computation of $\mathcal{H}_{\infty, \min}$. We note the following:

- After 275 iterations, the algorithm returns $2.5928 \leq \mathcal{H}_{2, \min} \leq 2.6006$.

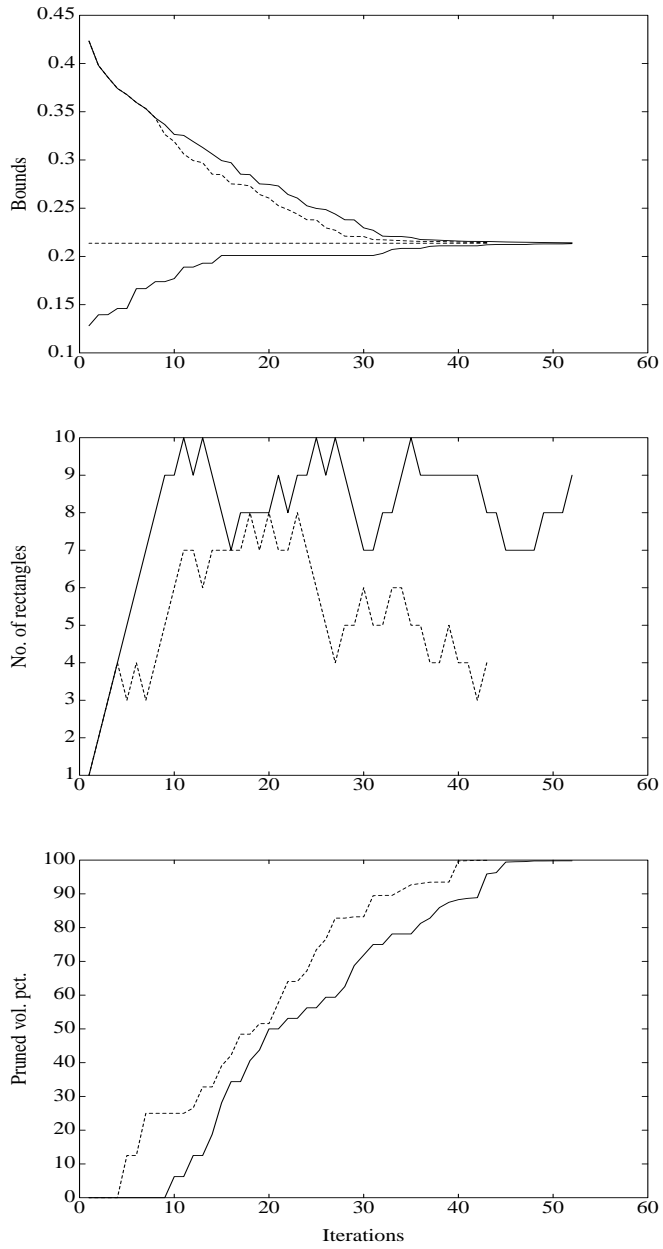


Fig. 11. Results from the branch and bound algorithm for \mathcal{D}_{\max} .

- The algorithm returns optimal gains $k_1 = 0.831$, $k_2 = 0.999$, which does *not* correspond to a vertex of the parameter rectangle. In fact, the minimum value among the \mathbf{H}_∞ norms at the vertices is 2.6225, which is slightly worse than $\mathcal{H}_{\infty,\min}$ returned by the algorithm.
- Not only does the algorithm return a set of parameters that achieves the upper bound (2.6006) on $\mathcal{H}_{\infty,\min}$, but it also can *prove* that the smallest achievable $\mathcal{H}_{\infty,\min}$ must be at least 2.5928.

3. $\mathcal{H}_{2,\min}$ -optimal incomplete state feedback

We finally compute the optimal feedback gains k_1 and k_2 that minimize the \mathbf{H}_2 norm of the transfer function from w to z with the gains restricted as in (41).

The results from the branch and bound algorithm are shown in Figure 13. We note that:

- After 17,500 iterations, the algorithm returns $0.9900 \leq \mathcal{H}_{2,\min} \leq 1.0002$. Thus, the performance of the algorithm, as with the case of $\mathcal{H}_{2,\max}$, is rather slow. This can be traced once again to the poor quality of the bounds (39) and (38) for $\mathcal{H}_{2,\min}$.
- The optimal set of gains for $\mathcal{H}_{2,\min}$ is $k_1 = 1$ and $k_2 = 1$, which is different than the gains returned for either \mathcal{D}_{\max} or $\mathcal{H}_{\infty,\min}$. Almost all of the work done by the algorithm has gone towards establishing the lower bound of 0.9900. The algorithm can *guarantee* that for every choice of parameters in (41), the \mathbf{H}_2 norm from w to z is at least 0.9900.

C. Simultaneous maximization of multiple objectives

Finally, we present an example that illustrates the branch and bound algorithm applied towards maximization of several objectives at the same time. We consider a robustness analysis problem and will use the same setup as in subsection A (see Figure 5). For this system, we consider the simultaneous maximization of the \mathbf{H}_∞ norms from w to x_1 and w to \dot{x}_1 .

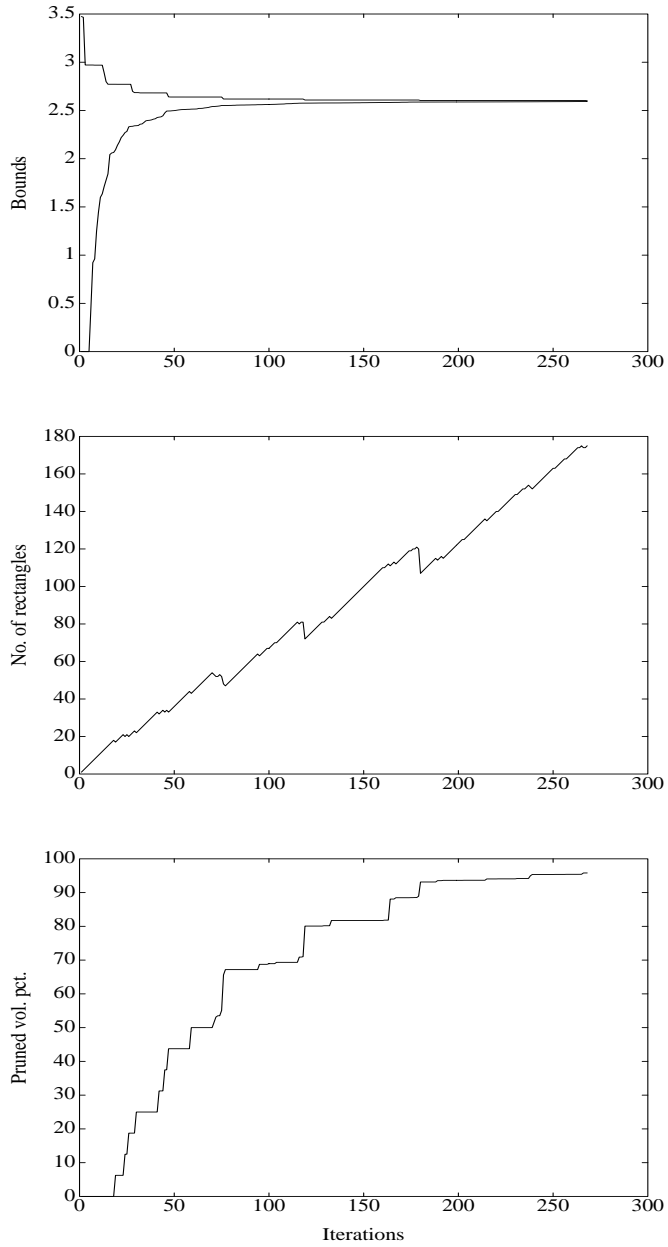


Fig. 12. Results from the branch and bound algorithm for $\mathcal{H}_{\infty, \min}$

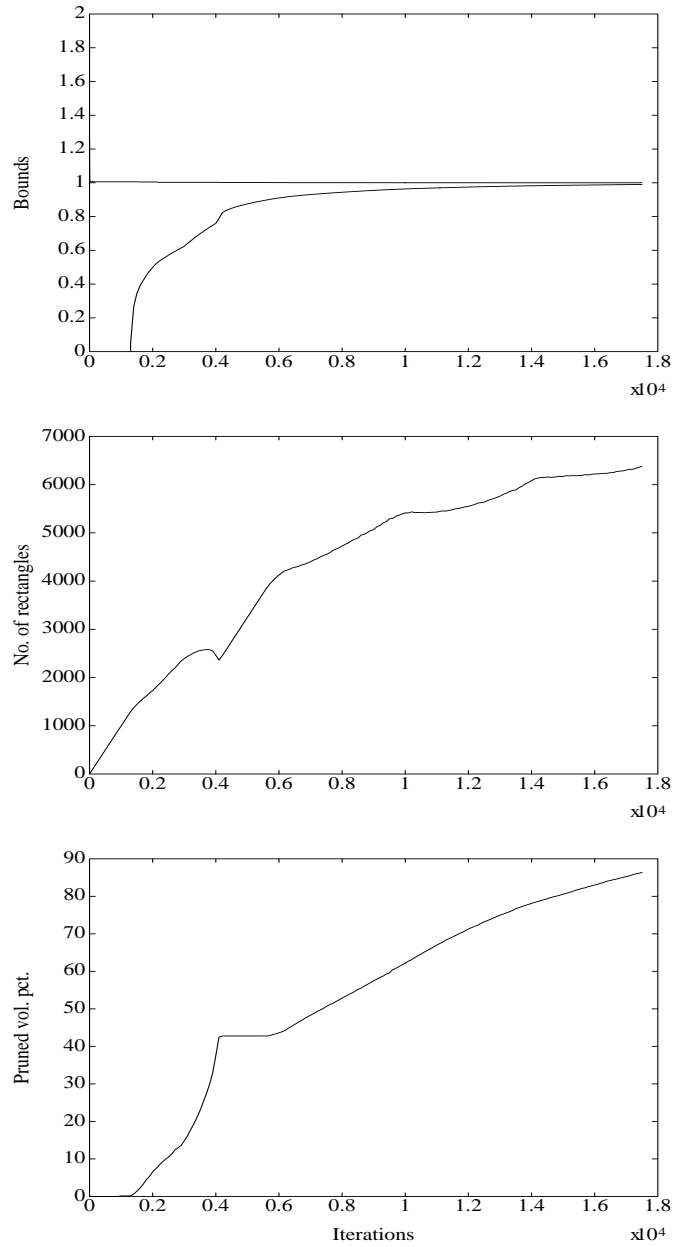


Fig. 13. Results from the branch and bound algorithm for $\mathcal{H}_{2,\min}$.

Figure 14 shows the result from the branch and bound algorithm applied to this problem using the bounds (26) and (29). The solid lines show the bounds $\mathcal{H}_{\infty, \max}$ between w and x_1 and the dashed lines, the bounds between w and \dot{x}_1 . It is clear that the algorithm takes quite a few iterations more compared to the maximization of just $\mathcal{H}_{\infty, \max}$ between w and x_1 (see Figure 8). The reason for this is clear from Figure 15, which shows the parameter region under consideration. The two objective functions that we seek to maximize achieve their maxima at two opposite corners of the parameter region. Thus, in effect, the simultaneous maximization algorithm has to do two separate individual branch and bound maximizations, interlaced.

VI. CONCLUSION

We have described some parameter problems in control systems analysis and design which may be cast as global optimization problems, and how a branch and bound algorithm may be used to solve them. *Our main point is that we may combine recent (and continuing) gains in computing power with advances in theory to answer questions that were previously unanswerable.* We know of no existing methods that compute exactly any of the six quantities that we have described in this chapter.

We must emphasize that branch and bound algorithm can be computation intensive (worst-case combinatoric), and requires much more computation than most local optimization methods, which, in many instances, do yield the global optimum. The main strength of the branch and bound algorithm, however, is in yielding *guaranteed* bounds for the global optimum. These bounds may be used, in turn, to inspire confidence in corresponding local optimization methods. For instance, if for a certain class of problems, a local minimization procedure consistently returns objective values that are not much larger than the lower bound on the minimum returned by the branch and bound algorithm, we may conclude that the local minimization method is “good enough”, especially if it involves far less computation.

Clearly, there is a trade-off between the computational effort spent on the branch and bound iterations and that spent on computing upper and lower bounds during each iteration: If the bounds are very good, they most likely require some computational effort; however, fewer branch and bound

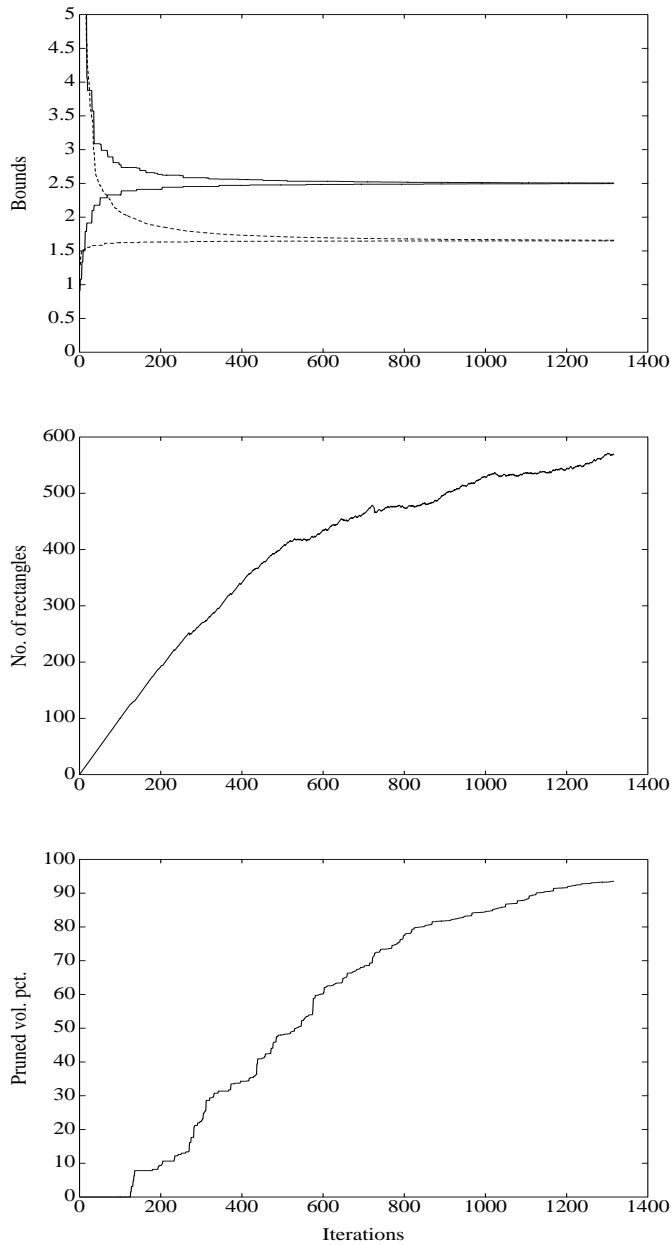


Fig. 14. Results from the branch and bound algorithm for multiple $\mathcal{H}_{\infty, \max}$ computations. The solid lines show bounds on $\mathcal{H}_{\infty, \max}$ between w and x_1 and the dashed lines, the bounds between w and \dot{x}_1 .

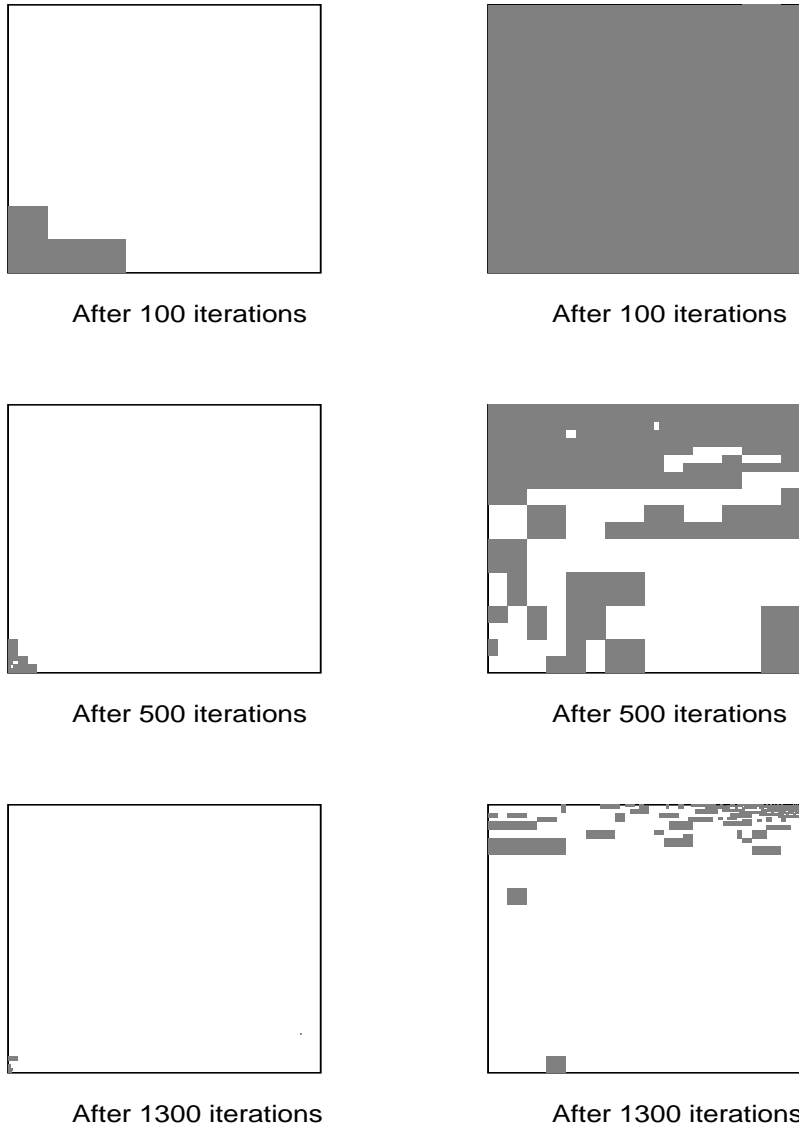


Fig. 15. The unpruned parameter region for either objective at various stages of the algorithm during the computation of $\mathcal{H}_{\infty, \max}$. The left hand side corresponds to $\mathcal{H}_{\infty, \max}$ between w and x_1 and the right hand side to $\mathcal{H}_{\infty, \max}$ between w and \dot{x}_1 . The x - and y -coordinates are k and $1/m_2$ respectively.

iterations will be needed. On the other hand, if the bounds are loose, the branch and bound algorithm may take a very long time to converge.

The basic branch and bound algorithm is easily extended to other problems, with the bound computation being the problem-specific task. We must mention, however, that the robust synthesis problem, which involves finding the design parameters that minimize the maximum of the objective over the uncertain parameters — the so-called “minimax” problem — cannot be handled in our setting. We refer the reader to [50] for work on this topic.

In conclusion,

- A number of global optimization problems in control systems analysis and design can be solved within a reasonable amount of time on present day computers.
- With advances in computing power and theory, the list of problems that can be solved using these methods is likely to grow.
- There will always be problems which will take inordinate amounts of time to solve with these methods.

VII. ACKNOWLEDGEMENTS

We would like to thank Silvano Balemi, coauthor of [18] and [50], with whom major parts of this work were completed. This research supported in part by NSF under ECS-85-52465 and by AFOSR under 89-0228.

VIII. REFERENCES

1. A. Nemirovsky and D. Yudin, *Problem Complexity and Method Efficiency in Optimization*, John Wiley & Sons, (1983).
2. C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, (1982).
3. P. M. Pardalos and J. B. Rosen, *Constrained Global Optimization: Algorithms and Applications*, Springer-Verlag, (1987).

4. R. H. J. M. Otten and L. P. P. P. van Ginneken, *The Annealing Algorithm*, Kluwer Academic Publishers, (1989).
5. Y. Ye, “Interior-point algorithms for global optimization”, *Annals of Operations Research*, **25**, pp. 59–74, (1990).
6. A. H. Land and A. G. Doig, “An automatic method for solving discrete programming problems”, *Econometrica*, **28**, pp. 497–520, (1960).
7. R. J. Dakin, “A tree search algorithm for mixed integer programming problems”, *The Computer Journal*, **8**, pp. 250–255, (1965).
8. E. L. Lawler and D.E. Wood, “Branch-and-bound methods: A survey”, *Operations Research*, **14**, pp. 699–719, (1966).
9. K. Spielberg, “Enumerative methods in integer programming”, *Annals of Discrete Mathematics*, **5**, pp. 139–183, (1979).
10. A. Schrijver, *Theory of Linear and Integer Programming*, Wiley-Interscience series in discrete mathematics. John Wiley & Sons, (1986).
11. M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, volume 2, Springer-Verlag, (1988).
12. E. Hansen, “Global optimization using interval analysis – the multi-dimensional case”, *Numerische Mathematik*, **34**, pp. 247–270, (1980).
13. R. R. E. De Gaston and M. G. Safonov, “Exact calculation of the multiloop stability margin”, *IEEE Trans. Aut. Control*, **33**(2), pp. 156–171, (1988).
14. A. Sideris and R. S. S. Peña, “Fast computation of the multivariable stability margin for real interrelated uncertain parameters”, *IEEE Trans. Aut. Control*, **34**(12), pp. 1272–1276, (1989).
15. B. C. Chang, O. Ekdal, H. H. Yeh, and S. S. Banda, “Computation of the real structured singular value via polytopic polynomials”, *J. of Guidance*, **14**(1), pp. 140–147, (1991).
16. A. Vicino, A. Tesi, and M. Milanese, “Computation of nonconservative stability perturbation bounds for systems with nonlinearly correlated uncertainties”, *IEEE Trans. Aut. Control*, **35**(7), pp. 835–841, (1990).

17. C. DeMarco, V. Balakrishnan, and S. Boyd, "A branch and bound methodology for matrix polytope stability problems arising in power systems", In *Proc. IEEE Conf. on Decision and Control*, , pp. 3022–3027, Honolulu, Hawaii, (1990).
18. V. Balakrishnan, S. Boyd, and S. Balemi, "Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems", *Int. J. of Robust and Nonlinear Control*, **1**(4), pp. 295–317, (1991).
19. T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*, Springer-Verlag, (1989).
20. V. Balakrishnan and S. Boyd, "Computation of the worst-case covariance for linear systems with uncertain parameters", In *Proc. IEEE Conf. on Decision and Control*, Brighton, U. K, (1991).
21. J. Rohn, "Systems of linear interval equations", *Linear Algebra and its Applications*, **126**, pp. 39–78, (1989).
22. J. Rohn, "Nonsingularity under data rounding", *Linear Algebra and its Applications*, **139**, pp. 171–174, (1990).
23. J. Rohn and S. Poljak, "Radius of nonsingularity", *Mathematics of Control, Signals, and Systems*, (1991-92), To appear.
24. J. Demmel, "The componentwise distance to the nearest singular matrix", *SIAM J. on Matrix Analysis and Applications*, (1991-92), To appear.
25. V. L. Kharitonov, "Asymptotic stability of an equilibrium position of a family of systems of linear differential equations", *Differentsial'nye Uraveniya*, **14**(11), pp. 1483–1485, (1978).
26. B. R. Barmish, "Invariance of the strict Hurwitz property for polynomials with perturbed coefficients", *IEEE Trans. Aut. Control*, **AC-29**, pp. 935–936, (1984).
27. A. C. Bartlett, C. V. Hollot, and H. Lin, "Root locations of an entire polytope of polynomials: it suffices to check the edges", *Mathematics of Control, Signals, and Systems*, **1**(1), pp. 61–71, (1989).

28. M. Fu and B. R. Barmish, “Polytopes of polynomials with zeros in a prescribed region”, *IEEE Trans. Aut. Control*, **34**(5), pp. 544–546, (1989).
29. S. Bialas, “A necessary and sufficient condition for stability of interval matrices”, *Int. J. Control*, **37**(4), pp. 717–722, (1983).
30. B. R. Barmish and C. V. Hollot, “Counterexamples to a recent result on the stability of interval matrices by S. Bialas”, *Int. J. Control*, **39**(5), pp. 1103–1104, (1984).
31. B. Ross Barmish, M. Fu, and S. Saleh, “Stability of a polytope of matrices: Counterexamples”, *IEEE Trans. Aut. Control*, **33**(6), pp. 569–572, (1988).
32. B. D. Anderson, N. K. Bose, and E. I. Jury, “Output feedback stabilization and related problems—Solution via decision methods”, *IEEE Trans. Aut. Control*, **AC-20**, pp. 53–66, (1975).
33. E. Zeheb, “Necessary and sufficient conditions for root clustering of a polytope of polynomials in a simply connected domain”, *IEEE Trans. Aut. Control*, **34**, pp. 986–990, (1989).
34. C. A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, Academic Press, New York, (1975).
35. D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, Mass., 2nd edition, (1984).
36. S. Boyd and C. Barratt, *Linear Controller Design: Limits of Performance*, Prentice-Hall, (1991).
37. S. Boyd, V. Balakrishnan, and P. Kabamba, “A bisection method for computing the \mathbf{H}_∞ norm of a transfer matrix and related problems”, *Mathematics of Control, Signals, and Systems*, **2**(3), pp. 207–219, (1989).
38. S. Boyd and V. Balakrishnan, “A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its \mathbf{L}_∞ -norm”, *Syst. Control Letters*, **15**, pp. 1–7, (1990).

39. B. Anderson and J. B. Moore, "Linear system optimization with prescribed degree of stability", *Proc. IEEE*, **116**(12), pp. 2083–2087, (1969).
40. J. Doyle, "Analysis of feedback systems with structured uncertainties", *IEE Proc.*, **129-D**(6), pp. 242–250, (1982).
41. M. G. Safonov, "Stability margins of diagonally perturbed multivariable feedback systems", *IEE Proc.*, **129-D**, pp. 251–256, (1982).
42. J. Doyle, J. E. Wall, and G. Stein, "Performance and robustness analysis for structured uncertainties", In *Proc. IEEE Conf. on Decision and Control*, , pp. 629–636, (1982).
43. M. K. Fan and A. L. Tits, "Characterization and efficient computation of the structured singular value", *IEEE Trans. Aut. Control*, **AC-31**(8), pp. 734–743, (1986).
44. M. K. Fan and A. L. Tits, " m -form numerical range and the computation of the structured singular value", *IEEE Trans. Aut. Control*, **33**(3), pp. 284–289, (1988).
45. M. G. Safonov, "Exact calculation of the multivariable structured-singular-value stability margin", In *Proc. IEEE Conf. on Decision and Control*, , pp. 1224–1225, Las Vegas, NV, (1984).
46. M. G. Safonov, "Optimal diagonal scaling for infinity-norm optimization", *Syst. Control Letters*, **7**, pp. 257–260, (1986).
47. M. G. Safonov and J. Doyle, "Optimal scaling for multivariable stability margin singular value computation", In *Proceedings of MECO/EES 1983 Symposium*, (1983).
48. M. Saeki, "A method of robust stability analysis with highly structured uncertainties", *IEEE Trans. Aut. Control*, **31**(10), pp. 935–940, (1986).
49. M. K. H. Fan, A. L. Tits, and J. C. Doyle, "Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics", *IEEE Trans. Aut. Control*, **36**(1), pp. 25–38, (1991).

50. S. Balemi and V. Balakrishnan, “Global optimization of \mathbf{H}_∞ -norm of parameter-dependent linear systems”, Technical Report # 91.15, Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), Zürich, (1991).
51. H. T. Toivonen and P. M. Mäkilä, “Computer-aided design procedure for multiobjective LQG control problems”, *Int. J. Control*, **49**(2), pp. 655–666, (1989).
52. R. A. Horn and C. A. Johnson, *Matrix Analysis*, Cambridge University Press, (1985).