A Gain-Based Lower Bound Algorithm for Real and Mixed μ Problems

Pete Seiler [a], Andrew Packard [b], Gary Balas [a]

[a] *Department of Aerospace Engineering and Mechanics, University of Minnesota, 55455-0153, USA*

[b] *Department of Mechanical Engineering, University of California, Berkeley, 94720-1740, USA*

Abstract

A new lower bound algorithm for real and mixed μ problems is presented. The basic idea of this algorithm is to use a related worst-case gain problem to compute the real blocks and, if the block structure is mixed, the standard power iteration to compute the complex blocks. Numerical tests indicate that the algorithm is fast and provides good lower bounds for both real and mixed μ problems of small to moderate size.

Key words: Structured Singular Value; Robust Stability; Uncertain Linear Systems

1 Introduction

The structured singular value μ, introduced in [1], can be used to analyze the robustness of linear systems subject to structured uncertainty. It is assumed that the reader is familiar with the engineering motivation for μ (see [2–4] and references therein for some discussion). It is known that computing μ is NP Hard [5,6] and for cases that include real parametric uncertainty, even approximately computing μ is NP Hard [7,8]. We refer the reader to [8] for a precise discussion of the computational complexity of ϵ-approximation problems. Thus there has been extensive research into computational algorithms that are fast and provide good lower/upper bounds for most problems of engineering interest. This paper describes an algorithm to compute lower bounds for μ with real parametric uncertainty.

For the pure complex μ problem, the power iteration [3,9] provides good lower bounds and it is fast since it relies only on matrix-vector products. The power iteration was extended to mixed μ problems in [10–12] and skew μ problems in [13–15]. Unfortunately this algorithm may fail to converge; a problem that is more common for purely real uncertainty structures [16–18]. There has been extensive research on alternative lower bound algorithms to address these issues [4,19–25,16,26–29,17].

For the pure real case, there are fundamental difficulties including the fact that real μ can be a discontinuous function of the problem data [30,31]. However, there are real μ problems of engineering interest that are well-posed. Most existing algorithms to solve these problems have a computational cost that grows exponentially with the problem size [20–23,28] and hence they are only suitable for small numbers of real parameters. A less computationally intensive approach is to regularize the problem and use the standard mixed μ power iteration. The regularization is typically accomplished by adding a small amount of complex uncertainty to each uncertain real parameter [31]. If the power iteration converges then the real parameter variations are obtained by neglecting the complex uncertainties. The magnitude of these real perturbations can then be increased to cause the system poles to lie on the imaginary axis [4]. Difficulties may arise if a large magnitude of complex uncertainty is needed to achieve power iteration convergence.

This paper describes a polynomial-time lower bound algorithm that can be applied to both pure real and mixed μ problems. We refer to this lower bound algorithm as the Gain-Based Algorithm (GBA). The basic idea of the GBA is to use a related worst-case gain problem to compute the real blocks of the perturbation. For mixed μ problems, the standard power iteration is then used to

Email addresses: `seiler@aem.umn.edu` (Pete Seiler), `pack@me.berkeley.edu` (Andrew Packard), `balas@umn.edu` (Gary Balas).

[1] This paper was not presented at any IFAC meeting. Corresponding author P. Seiler. Tel. +1 612 625 6561; Fax +1 612 626 1558.

compute the complex blocks since it is fast and has good convergence characteristics for complex μ problems. The GBA uses the *wrap-in reals* idea that exists in [16,17] but with two main distinctions. First, the use of the worst-case gain problem to compute the values of the real blocks is a new approach. Second, we do not wrap in the real blocks within each step of the power iteration. Instead, the real block is computed from scratch for each step of the GBA, wrapped-in, and then held fixed throughout the complex power iteration.

The GBA has been tested on several real-μ problems arising in industry as well as random matrices drawn from a class of no-gap mixed μ problems. It is our experience that the GBA has better convergence properties than the standard power iteration on these test problems. These results are discussed in Section 4. The improved convergence properties are due to an implicit regularization of the problem: the GBA can be viewed as adding a small complex scalar to one entry of the data matrix. Moreover, the GBA makes N_{try} attempts to find a good lower bound and N_{try} can be used to trade off computation time with the quality of the lower bound. The GBA has been integrated into the `mussv` function of Matlab's Robust Control Toolbox and can be called with the `'g'` option. This should enable other users to easily apply it to their own problems of interest.

The basic idea of computing μ via a related worst-case gain problem can potentially be applied to a variety of control analysis problems. The idea can be generalized to analyze robustness problems with other uncertainty norms, e.g. the 2-norm for ellipsoidal uncertainty [32]. This would require an algorithm to quickly and reliably solve the worst-case gain problem for the uncertainty norm of interest. The idea can also be applied to the analysis of nonlinear lumped and distributed parameter systems [33–35]. This would be applicable to analysis of finite-time control in batch and semibatch processes.

2 Notation

$\mathbb{C}^{n\times m}$ and $\mathbb{R}^{n\times m}$ are complex and real $n\times m$ matrices. For any matrix M, M^T and M^* denote the transpose and complex conjugate transpose of M. Given $A\in\mathbb{C}^{n\times m}$ and $B\in\mathbb{C}^{r\times s}$, $\text{diag}(A,B)\in\mathbb{C}^{(n+r)\times(m+s)}$ denotes the block diagonal concatenation. $\bar\sigma(M)$ and $\underline{\sigma}(M)$ denote the maximum and minimum singular values of the matrix M. Let $M\in\mathbb{C}^{(n+m)\times(n+m)}$ and $\Delta\in\mathbb{C}^{n\times n}$ be given and partition $M := \left[\begin{smallmatrix} M_{11} & M_{12}\\ M_{21} & M_{22}\end{smallmatrix}\right]$ with $M_{11}\in\mathbb{C}^{n\times n}$ and $M_{22}\in\mathbb{C}^{m\times m}$. If $I-M_{11}\Delta$ is invertible, then define $F_u(M,\Delta)$ as the linear fractional transformation (LFT) obtained by closing Δ around the upper channels of M:

$$F_u(M,\Delta) := M_{22} + M_{21}\Delta\left(I-M_{11}\Delta\right)^{-1}M_{12}$$

The notation used in this paper for the structured singular value, μ, is standard. We consider block structures consisting of r repeated real scalar blocks, c repeated complex scalar blocks, and f square full complex blocks. The restriction to square full blocks is for notational simplicity and the results can be extended to non-square full blocks. Given positive integers $k_1,k_2,\ldots,k_{r+c+f}$ define the following sets of block structured matrices:

$$\begin{aligned}
\boldsymbol{\Delta}_R &:= \{\Delta = \text{diag}(\delta_1 I_{k_1},\ldots,\delta_r I_{k_r}) \ :\ \delta_i\in\mathbb{R}\}\\
\boldsymbol{\Delta}_C &:= \big\{\Delta = \text{diag}(\delta_1 I_{k_{r+1}},\ldots,\delta_c I_{k_{r+c}},\Delta_1,\ldots,\Delta_f)\ :\\
&\qquad \delta_i\in\mathbb{C},\ \Delta_i\in\mathbb{C}^{k_{r+c+i}\times k_{r+c+i}}\big\}\\
\boldsymbol{\Delta} &:= \{\Delta = \text{diag}(\Delta_R,\Delta_C)\ :\ \Delta_R\in\boldsymbol{\Delta}_R,\ \Delta_C\in\boldsymbol{\Delta}_C\}
\end{aligned}$$

$\boldsymbol{\Delta}_R$, $\boldsymbol{\Delta}_C$, and $\boldsymbol{\Delta}$ are pure real, pure complex, and mixed real/complex block structures, respectively. The matrices in $\boldsymbol{\Delta}_R$, $\boldsymbol{\Delta}_C$, and $\boldsymbol{\Delta}$ have respective dimensions $n_R\times n_R$, $n_C\times n_C$, and $n\times n$ where $n_R := \sum_{i=1}^r k_i$, $n_C := \sum_{i=1}^{c+f} k_{r+i}$, and $n := n_R + n_C$.

The next definition, originally given in [1] for the pure complex case, is for μ in terms of the block structure defined by the set $\boldsymbol{\Delta}$. However, it also applies to other block structures such as the pure real ($\mu_{\boldsymbol{\Delta}_R}$) and pure complex ($\mu_{\boldsymbol{\Delta}_C}$) cases.

Definition 1 *[1] The structured singular value of $M\in\mathbb{C}^{n\times n}$ with respect to $\boldsymbol{\Delta}$, denoted $\mu_{\boldsymbol{\Delta}}(M)$, is defined as*

$$\mu_{\boldsymbol{\Delta}}(M) := \left(\min_{\Delta\in\boldsymbol{\Delta}}\left\{\bar\sigma(\Delta)\ :\ \det(I-M\Delta)=0\right\}\right)^{-1} \tag{1}$$

if $\exists\Delta\in\boldsymbol{\Delta}$ such that $\det(I-M\Delta)=0$ and otherwise $\mu_{\boldsymbol{\Delta}}(M):=0$.

3 Gain-Based Algorithm (GBA)

In this section, we first introduce the basic idea of the GBA (Section 3.1). Then we describe the full GBA for pure real μ problems (Section 3.2) and for mixed μ problems (Section 3.3).

3.1 Basis for GBA

Assume $M_R\in\mathbb{C}^{n_R\times n_R}$ and consider the problem of computing lower bounds for the pure real problem $\mu_{\boldsymbol{\Delta}_R}(M_R)$. Note that M_R is generally a complex matrix and the subscript R refers to the real block structure used in the μ computation. It follows from the definition of μ that any $\Delta_R\in\boldsymbol{\Delta}_R$ that satisfies $\det(I-M_R\Delta_R)=0$ yields a lower bound: $\frac{1}{\bar\sigma(\Delta_R)}\le\mu_{\boldsymbol{\Delta}_R}(M_R)$. Thus lower bounds for $\mu_{\boldsymbol{\Delta}_R}(M_R)$ can be computed by searching for a $\Delta_R\in\boldsymbol{\Delta}_R$ for which there exists nonzero $z\in\mathbb{C}^{n_R}$ and $w\in\mathbb{C}^{n_R}$ satisfying $z=M_R w$ and $w=\Delta_R z$. These equations can be represented by the LFT $F_u(M_R,\Delta_R)$ with z and w denoting the output of M_R and Δ_R, respectively. The basis for the GBA is to recast this

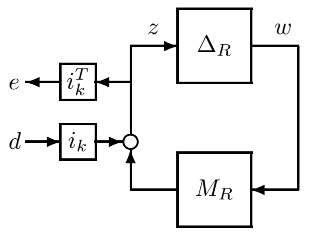

Fig. 1. M_R-Δ_R Loop With d-to-e Channels

problem into a related worst-case disturbance-to-error problem, shown in Figure 1. In this figure, a scalar disturbance is inserted at the output of M_R and a scalar error signal is pulled off the input to Δ_R. i_k denotes the k^{th} standard basis vector in $\mathbb{R}^{n_R}$ and $d, e \in \mathbb{C}$ denote the scalar disturbance and error signals.

This input/output system represents the algebraic equations $\left[\begin{smallmatrix} z \\ e \end{smallmatrix}\right] = \tilde{M}_R \left[\begin{smallmatrix} w \\ d \end{smallmatrix}\right]$ and $w = \Delta_R z$ where $\tilde{M}_R := \begin{bmatrix} M_R & i_k \\ i_k^T M_R & 1 \end{bmatrix}$. If $\det(I - M_R\Delta_R) \neq 0$ then these equations are well-posed and the disturbance to error relation is given by: $e = F_u(\tilde{M}_R, \Delta_R)d$. The key step of the GBA chooses an estimated value for the lower bound, lb_{try}, and attempts to solve the following problem:

$$\max_{\Delta_R \in \mathbf{\Delta}_R, \bar{\sigma}(\Delta_R) \leq 1/lb_{try}} |F_u(\tilde{M}_R, \Delta_R)| \qquad (2)$$

Equation 2 is in the form of a worst case performance problem. This problem is non-convex and solving for the global maximizer would be computationally intensive. However it is sufficient for our purposes to find any Δ_R achieving a large gain from d to e. We will apply the lower bound algorithm introduced in [36] for worst-case performance assessment. The algorithm is an ascent method that returns a lower bound on the maximum. Specifically, an exact maximization is used for the single parameter problem ($r = 1$) and an iterative coordinate-wise maximization is used for the general case ($r > 1$). The exact maximization along each coordinate is computed by mimicking the Hamiltonian methods for state-space H_∞ norm calculation. The lower bound algorithm in [36] is presented for $\bar{\sigma}(\Delta_R) \leq 1$. This is without loss of generality since the perturbation can be normalized.

If the maximum gain can be driven to infinity (within numerical error) then the maximizer $\Delta_{R,opt}$ satisfies $\det(I - M_R\Delta_{R,opt}) = 0$. This yields a true lower bound: $\mu_{\mathbf{\Delta}_R}(M_R) \geq \frac{1}{\bar{\sigma}(\Delta_{R,opt})}$. Restricting the search to $\bar{\sigma}(\Delta_R) \leq 1/lb_{try}$ ensures that $\Delta_{R,opt}$ will yield a lower bound that is $\geq lb_{try}$. It is more often the case that the coordinate-wise ascent algorithm will return a $\Delta_{R,opt}$ that gives an extremely large but finite gain. If we find a $\Delta_R \in \mathbf{\Delta}_R$ such that the gain from d to e is large then $I - M_R\Delta_R$ will be close to singularity. The following theorem provides two precise relations between the d-to-e gain and the distance of $I - M_R\Delta_R$ to singularity.

Theorem 1 *If* $\exists \Delta_R \in \mathbf{\Delta}_R$ *such that* $\det(I - M_R\Delta_R) \neq 0$ *and* $|F_u(\tilde{M}_R, \Delta_R)| \geq \gamma > 0$ *then:*

(A.) $\exists \delta \in \mathbb{C}$, $|\delta| \leq \gamma^{-1}$ *such that* $\det(I - M_R\Delta_R - \delta i_k i_k^T) = 0$
(B.) $\underline{\sigma}(I - M_R\Delta_R) \leq \gamma^{-1}$

Proof 1 *(A.) Since* $\det(I - M_R\Delta_R) \neq 0$*, the equations* $\left[\begin{smallmatrix} z \\ e \end{smallmatrix}\right] = \tilde{M}_R \left[\begin{smallmatrix} w \\ d \end{smallmatrix}\right]$ *and* $w = \Delta_R z$ *are well-posed, i.e. for each* $d \in \mathbb{C}$ *there exist unique* $e \in \mathbb{C}$ *and* $w, z \in \mathbb{C}^{n_R}$ *that satisfy these equations. This unique solution is:*

$$\begin{aligned} z &= (I - M_R\Delta_R)^{-1} i_k d \\ w &= \Delta_R (I - M_R\Delta_R)^{-1} i_k d \\ e &= i_k^T (I - M_R\Delta_R)^{-1} i_k d \end{aligned} \qquad (3)$$

Let $d = 1$ *and consider the* e, w, z *(note* $|e| = \gamma$*) that provide the unique solution to the input-output equations. Define* $\delta := d/e \in \mathbb{C}$ *and* $\tilde{\Delta}_R := \begin{bmatrix} \Delta_R & 0 \\ 0 & \delta \end{bmatrix}$*. Note that* $|\delta| = 1/|e| = 1/|F_u(\tilde{M}_R, \Delta_R)| \leq \gamma^{-1}$ *and* $d = \delta e$*. Thus* d, e, w, z *are a nontrivial solution to the equations* $\left[\begin{smallmatrix} z \\ e \end{smallmatrix}\right] = \tilde{M}_R \left[\begin{smallmatrix} w \\ d \end{smallmatrix}\right]$ *and* $\left[\begin{smallmatrix} w \\ d \end{smallmatrix}\right] = \tilde{\Delta}_R \left[\begin{smallmatrix} z \\ e \end{smallmatrix}\right]$*. This implies that* $I - \tilde{M}_R\tilde{\Delta}_R$ *is singular. Next, define the nonsingular transformation* $T := \begin{bmatrix} I & 0 \\ -i_k^T & 1 \end{bmatrix}$*. Singularity of* $I - \tilde{M}_R\tilde{\Delta}_R$ *implies the singularity of:*

$$T\left(I - \tilde{M}_R\tilde{\Delta}_R\right)T^{-1} = \begin{bmatrix} I - M_R\Delta_R - \delta i_k i_k^T & -\delta i_k \\ 0 & 1 \end{bmatrix}$$

This implies $I - M_R\Delta_R - \delta i_k i_k^T$ *is singular.*

(B.) By the proof of (A.), $\exists \delta \in \mathbb{C}$ *such that* $|\delta| \leq \gamma^{-1}$ *and* $\det(I - M_R\Delta_R - \delta i_k i_k^T) = 0$*. Hence* $\underline{\sigma}\left(I - M_R\Delta_R - \delta i_k i_k^T\right) = 0$*. Apply Theorem 3.3.16 of [37] to conclude*

$$\begin{aligned} \underline{\sigma}(I - M_R\Delta_R) &\leq \underline{\sigma}\left(I - M_R\Delta_R - \delta i_k i_k^T\right) + \bar{\sigma}\left(\delta i_k i_k^T\right) \\ &= |\delta| \leq \gamma^{-1} \end{aligned}$$

3.2 Real μ GBA

The full GBA to compute real μ lower bounds is presented in Table 1. The GBA is initialized with upper and lower bounds (ub and lb) on $\mu_{\mathbf{\Delta}_R}(M_R)$ as well as a perturbation (Δ_R) achieving lb. The lower bound information can simply be initialized to $lb = 0$ and $\Delta_R = 0_{n_R}$. Alternatively, the lower bound and perturbation from the standard power iteration can be used. However, the standard power iteration has convergence issues for pure real μ problems and from our experience it is not worth the computational effort to perform this iteration. The upper bound can be computed via standard methods, e.g. using the LMI [38] or Balanced [39,12] form. These upper/lower bounds are used within the iteration to obtain good estimates for lb_{try}.

For each iteration, the GBA chooses lb_{try} that restricts the perturbation search in Equation 2. It must also select the channel k in which to insert and pull off the disturbance and error signals. These choices will be described further below. The GBA then attempts to solve Equation 2 using the worst case gain lower bound algorithm given in [36]. As mentioned above, the lower bound algorithm typically returns a $\Delta_{R,try}$ that achieves a finite but large gain. Based on Theorem 1 we anticipate that $I - M_R\Delta_{R,try}$ will be close to singularity but we perform a check using the reciprocal condition number. [2] If the perturbation passes the singularity check then it and the lower bound it achieves are stored. A factor used for choosing lb_{try} is updated based on the success or failure of the singularity check. The iteration will stop if it finds a lower bound within a factor tol_{stop} of the upper bound or if it reaches the maximum number of iterations, N_{try}.

We now provide further details on the selection of lb_{try}, lb_{fac}, and k. Since the value of $\mu_{\mathbf{\Delta}_R}(M_R)$ is unknown, lb_{try} must be intelligently selected at each iteration. If lb_{try} is too large then it might exceed $\mu_{\mathbf{\Delta}_R}(M_R)$ and we will not find a Δ_R such that $I - M_R\Delta_{R,try}$ passes the singularity check. If lb_{try} is too small then our search might be successful but with only a minor increase in lb. Our solution is to set $lb_{try} = lb + (ub - lb)lb_{fac}$ where lb_{fac} is adaptively updated based on the search results. We start with $lb_{fac} = 3/4$ based on the experience that the LMI upper bound is typically within this factor of the true value of μ. This value provides a good starting point for successfully finding a lower bound. Within the iteration, we set $lb_{fac} = 1/2$ for each successful search and back off by a factor of two for each failed search. This can be roughly viewed as a stochastic bisection and we have found this provides a good compromise. Finally, we do not allow lb_{fac} to decrease below 1/32. This is based on the engineering judgment that that it is better to spend the computation time searching for a perturbation that increases the lower bound by at least this factor.

[2] The minimum singular value or the determinant could be used instead of the reciprocal condition number to check for singularity. We chose not to use the determinant since it is possible for matrices to have small determinants even if they are not necessarily close to singularity, e.g. $\det(0.9I_{200}) = 7.1\times10^{-10}$. We also found that the algorithm typically drives $\underline{\sigma}\left(I - M_R\Delta_{R,try}\right)$ to a small value but $\bar{\sigma}\left(I - M_R\Delta_{R,try}\right)$ remains O(1). Consequently the reciprocal condition number and minimum singular value are roughly of the same order and either is suitable for the singularity check.

The input/output channel for the disturbance/error is selected as $k :=\text{mod }(cnt-1, n_R)+1$ where mod denotes the modulus after division. This choice simply cycles k up from 1 to n_R and then rolls back to 1. This simple strategy appears to work well and we currently do not have a better method to determine which channel is likely to provide the best performance. This would be worth investigating. We should also point out that the algorithm can be generalized by replacing i_k with a vector $v \in \mathbb{C}^{n_R}$. The disturbance and error signals remain scalar but they are inserted into and pulled off from a linear combination of the channels. Theorem 1 continues to hold with i_k replaced by v throughout. Again, we have no method to select a good candidate v and hence our algorithm currently does not utilize this generalization. Finally, one could generalize to the case where i_k is replaced by a matrix $V \in C^{n_R\times m}$. In this case d and e are no longer scalar signals; they have dimension m. For this generalization, Theorem 1-A requires a rank m perturbation to bring $I - M_R\Delta_R$ to singularity. Initial attempts using $V = I_{n_R}$ provided worse performance than the algorithm as outlined in Table 1.

The performance of the GBA is discussed in Section 4. Our results indicate that the GBA has good convergence properties on pure real μ problems of engineering interest since the GBA implicitly regularizes the problem. Specifically, Theorem 1-A shows that the algorithm can be viewed as attempting to make $I - M_R\Delta_R$ singular by adding a complex perturbation of magnitude $\leq \gamma^{-1}$ to the (k,k) diagonal entry. The algorithm can be interpreted as minimizing the magnitude of a scalar complex uncertainty that is artificially introduced to regularize the problem.

3.3 Mixed μ GBA

Assume $M \in \mathbb{C}^{n\times n}$ and partition M conformably with the real and complex blocks of $\mathbf{\Delta}$, $M := \left[\begin{smallmatrix} M_R & M_{RC} \\ M_{CR} & M_C \end{smallmatrix}\right]$ where $M_R \in \mathbb{C}^{n_R\times n_R}$ and $M_C \in \mathbb{C}^{n_C\times n_C}$. This section addresses the problem of computing lower bounds for the mixed μ problem $\mu_{\mathbf{\Delta}}(M)$. The GBA for mixed μ problems is presented in Table 2. The GBA makes up to N_{try} attempts to find a good lower bound with lb_{try} being updated adaptively. The mixed μ GBA is initialized with upper and lower bounds (ub and lb) on $\mu_{\mathbf{\Delta}_R}(M_R)$ as well as a perturbation (Δ_R) achieving lb. The lower bound information can be initialized to $lb = 0$ and $\Delta_R = 0_{n_R}$ but it is worth the computational effort to run the power iteration first and use the lower bound and perturbation it returns to initialize lb and $\Delta \in \mathbf{\Delta}$ in the GBA.

For each attempt of the GBA, the related worst-case gain problem (Figure 1) is used to compute the real block of the perturbation. If the real block alone causes singularity then it is used to compute a valid lower bound. We can always use the complex blocks to ensure $\det(I - M\Delta) = 0$ within numerical tolerance. Hence we can

```
1.  Given : $M_R \in \mathbb{C}^{n_R \times n_R},\ \Delta_R \in \mathbf{\Delta}_R,\ lb,\ ub$
2.  Initialize : $lb_{fac} = 3/4,\ cnt = 1$
3.  while $cnt \le N_{try}$ AND $lb < ub \cdot tol_{stop}$
4.      $lb_{try} = lb + (ub - lb) lb_{fac}$
5.      $k := \mod(cnt - 1, n_R) + 1$
6.      $\tilde{M}_R := \begin{bmatrix} M_R & i_k \\ i_k^T M_R & 1 \end{bmatrix}$
7.      $\Delta_{R,try} := \arg\max_{\Delta_R \in \mathbf{\Delta}_R,\ \bar{\sigma}(\Delta_R) \le 1/lb_{try}} |F_u(\tilde{M}_R, \Delta_R)|$
8.      if $\mathrm{rcond}(I - M_R \Delta_{R,try}) < tol_{real}$
9.          $lb = \frac{1}{\bar{\sigma}(\Delta_{R,try})}$
10.         $\Delta_R = \Delta_{R,try}$
11.         $lb_{fac} := 1/2$
12.     else
13.         $lb_{fac} := \max(1/32, lb_{fac}/2)$
14.     end
15.     $cnt = cnt + 1$
16. end
17. Return : $\Delta_R, lb$
```
Table 1
The GBA for Real μ Lower Bounds

```
1.  Given : $M := \begin{bmatrix} M_R & M_{RC} \\ M_{CR} & M_C \end{bmatrix} \in \mathbb{C}^{n \times n},\ \Delta \in \mathbf{\Delta},\ lb,\ ub$
2.  Initialize : $lb_{fac} = 3/4,\ cnt = 1$
3.  while $cnt \le N_{try}$ AND $lb < ub \cdot tol_{stop}$
4.      $lb_{try} = lb + (ub - lb) lb_{fac}$
5.      $k := \mod(cnt - 1, n_R) + 1$
6.      $\tilde{M}_R := \begin{bmatrix} M_R & i_k \\ i_k^T M_R & 1 \end{bmatrix}$
7.      $\Delta_{R,try} := \arg\max_{\Delta_R \in \mathbf{\Delta}_R,\ \bar{\sigma}(\Delta_R) \le 1/lb_{try}} |F_u(\tilde{M}_R, \Delta_R)|$
8.      if $\mathrm{rcond}(I - M_R \Delta_{R,try}) < tol_{complex}$
9.          $lb = \frac{1}{\bar{\sigma}(\Delta_{R,try})}$
10.         $\Delta = \mathrm{diag}(\Delta_{R,try}, 0)$
11.         $lb_{fac} := 1/2$
12.     else
13.         $\tilde{M}_C := F_u(M, \Delta_R)$
14.         Power Iteration on $\tilde{M}_C$ to find $\Delta_{C,try} \in \mathbf{\Delta}_C$
15.         $\Delta_{try} := \mathrm{diag}(\Delta_{R,try}, \Delta_{C,try})$
16.         if $\mathrm{rcond}(I - M\Delta_{try}) < tol_{complex}$ AND $\frac{1}{\bar{\sigma}(\Delta_{try})} \ge lb$
17.             $lb = \frac{1}{\bar{\sigma}(\Delta_{try})}$
18.             $\Delta = \Delta_{try}$
19.             $lb_{fac} := 1/2$
20.         else
21.             $lb_{fac} := \max(1/32, lb_{fac}/2)$
22.         end
23.     end
24.     $cnt = cnt + 1$
25. end
26. Return : $\Delta, lb$
```
Table 2
The GBA for Mixed μ Lower Bounds

set $tol_{complex}$ to be a small factor above numerical tolerance, e.g. 100`eps`. The main point is that the mixed μ GBA, unlike the real-μ GBA presented in the previous section, returns perturbations that strictly cause $\det(I - M\Delta) = 0$ within numerical errors.

If the real block does not cause singularity, then it is wrapped into M to form $\tilde{M}_C = F_u(M, \Delta_R)$. The standard power iteration [3,9] is run on $\tilde{M}_C$ to compute the complex block, $\Delta_C \in \mathbf{\Delta}_C$. A perturbation, $\Delta \in \mathbf{\Delta}$, is then formed from the real/complex blocks and stored if it increases the current lower bound. Even though lb_{try} is chosen to be strictly larger than the current lower bound, the perturbation Δ might not increase the lower bound. In particular, if the norm of the complex block is too large ($\bar{\sigma}(\Delta_C) > 1/lb$), then the perturbation will not improve the lower bound. It seems that the adaptive selection of lb_{try} naturally balances the norms of the real and complex blocks. If lb_{try} is too large, then we will be overly restricting our search for Δ_R. As a result the gain of $F_u(M_R, \Delta_R)$ may not be very large and hence it will take a larger norm complex block to make the loop singular. If this larger norm complex block causes Δ to achieve a lower bound less than lb_{try} then lb_{try} will be decreased on the next iteration. This will increase the search for Δ_R, resulting in a larger gain for $F_u(M_R, \Delta_R)$, and hence a smaller norm Δ_C will be needed to force loop singularity. The performance of the mixed μ GBA is discussed in the next section.

3.4 Discussion of Algorithm Performance

As pointed out in the Introduction, there are fundamental computational issues associated with computing μ. In particular, this computational problem is NP Hard and approximately computing μ is also NP Hard. As a result, it is unlikely that a polynomial time algorithm can be developed that computes or approximately computes μ in all cases. Real μ problems have the additional issue that the result can be a discontinuous function of the problem data [30,31]. Given these fundamental difficulties, it is important to address the assessment of the GBA performance for real and mixed μ problems.

For real μ problems there are essentially three classes of algorithms for computing lower bounds:

(1) *Exponential-Time Algorithms*: There are several algorithms of this type listed in the introduction and some of these come with rigorous proofs that they compute μ to within a specified tolerance. The drawback is that the computation time grows exponentially with the number of real uncertainties. This limits their use to problems with only a few real uncertainties.
(2) *Polynomial-time Algorithms*: To our knowledge, the power iteration is the only algorithm in the literature in this class for real μ problems. Each iteration mainly consists of matrix-vector products and vector norm calculations. The number of these calculations per iteration and the vector/matrix dimensions scale linearly with the number and size of the uncertainty blocks. For a fixed number of iterations, this computation time grows polynomially with the number and size of the uncertainty blocks. Unfortunately this algorithm has poor convergence characteristics on real μ problems.
(3) *Approximation Algorithms*: The main algorithm in this class is the regularization [31] described in the introduction. The benefit is that it improves the convergence properties of the power iteration, which is a fast algorithm. The drawback is that the magnitude of the complex uncertainty needed to achieve power iteration convergence may be sufficiently large that it alters the original problem. The magnitude of these real perturbations can then be increased to cause the system poles to lie on the imaginary axis [4]. Difficulties may arise if a large magnitude of complex uncertainty is needed to achieve power iteration convergence.

The bulk of the computation time for the GBA occurs in solving the worst-case gain problem. The algorithm we use to solve this worst-case gain problem consists of a fixed number of coordinate-wise ascents. Each ascent step requires the solution of a number of eigenvalue problems with dimensions equal to the uncertainty block dimensions. The overall algorithm is complicated and we have neglected many details. However, it is our anticipation that the computational cost will grow polynomially with the number and size of the uncertainty blocks. On a limited class of problems, we have indeed observed the computational cost of the real μ GBA to grow polynomially with the number of real uncertainties.

Given the complexity of computing μ it is unlikely that we can prove that our algorithm computes or approximately computes μ for all problems. It should again be stressed that this is a fundamental characteristic of the problem, i.e. it is unlikely that generic performance guarantees can be proved for any polynomial-time algorithm. Instead, we could attempt to prove the GBA converges for a limited class of real μ problems. For example, an exact solution has been found for the case where M is rank 1 [40–42]. Unfortunately, our algorithm does not converge for all instances of rank 1 problems [3]. We would still like to assess the performance of the GBA on "typical" engineering problems. We have compiled and posted a set of test examples at: `http://jagger.me.berkeley.edu/~pack/gblowerbound/`. This website contains HTML codepads/results for bus steering, hydraulic servos, flight control, missile autopilots, disk drives, systems biology, spark-ignition engines, water tank systems, and mass-spring damper systems. These files compare the performance of the GBA against the only other polynomial-time algorithm, the power iteration. Results from two of these examples are presented in Section 4.

Similar comments apply concerning the computational complexity of mixed μ problems. The computation for the mixed μ power iteration grows polynomially in the number and size of the uncertainty block dimensions. This iteration has good convergence properties for many mixed μ problems. In these instances the power iteration should be used over the GBA since it has typically has lower computational cost. The GBA algorithm can be used for mixed μ problems for which the real uncertainties prevent the power iteration from converging. The GBA performance on a class of mixed μ problems is presented in Section 4.

4 Numerical Results

The GBA has been successfully applied to several engineering problems arising in industry. These problems involve only real parameter variations and the standard power iteration fails to provide useful lower bounds. In this section we demonstrate the performance of the GBA algorithm on two problems which are similar to those encountered in industry. We then test the performance of the GBA on a class of no-gap mixed μ problems.

The GBA is integrated into the `mussv` function of Matlab's Robust Control Toolbox [2] and can be called with the `'g'` option. All of the results presented in this section will use the algorithm as currently implemented in Version 3.4 of the Robust Control Toolbox. The algorithm settings are $tol_{stop} = 0.97$, $tol_{real} = 1 \times 10^{-7}$, and $tol_{comp} = 100\texttt{eps}$ where $\texttt{eps} = 2^{-52} \approx 2.22 \times 10^{-16}$ for floating point doubles in Matlab. $N_{try} = 10 + 10k$ where k is the integer following the 'g' option, i.e. `mussv(M,blk,'g6')` uses $N_{try} = 70$. Also, the worst-case coordinate-wise ascent is stopped if the gain from d to e exceeds 1×10^{12}. All results were computed on a 2.66GHz Intel Core 2 Quad CPU.

[3] Consider the rank 1 matrix $M = \left[\begin{smallmatrix} 1+j & 1+j \\ 1 & 1 \end{smallmatrix}\right]$ with $\mathbf{\Delta}_R$ consisting of two real, scalar uncertainties. $\Delta_R = \text{diag}(0,1)$ is the only $\Delta_R \in \mathbf{\Delta}_R$ for which $I - M\Delta_R$ is singular. The disturbance-to-error gain associated with either the first or second output channel is undefined at $(\delta_1, \delta_2) = (0,1)$ and remains finite in a neighborhood of this point.

4.1 Real μ GBA: Analog Filter Analysis

Anti-aliasing filters are used in flight control electronics to limit the distortion effects introduced by digital sampling. In this example we consider a low-pass Sallen-Key filter [43,44]. This is a second-order, active filter implemented with one op-amp, two resistors, and two capacitors. The input/output transfer function for the Sallen-Key filter is given by:

$$F(s) = \frac{1}{R_1 R_2 C_1 C_2 s^2 + C_2 (R_1 + R_2) s + 1} \qquad (4)$$

The variations in part values due to manufacturing tolerances, temperature, and other factors can have a significant impact on filter performance. The flight control electronics is a safety critical system and the DO-254 standard [45] lays out a requirements-based design process. In following this process, it must be verified that the anti-aliasing filters satisfy all design requirements over the range of possible part values. We will consider one such requirement: the poles of the filter shall have a damping ratio greater than $\zeta_{min} = \sqrt{2}/2$. Assume the parts are chosen: $R_1 = 17.5M\Omega \pm 10\%$, $R_2 = 0.5M\Omega \pm 10\%$, $C_1 = 1\mu F \pm 25\%$, and $C_2 = 0.1\mu F \pm 25\%$. The nominal filter has poles at $-1.03 \pm 0.29j$ and this meets the damping ratio requirement ($\zeta = 0.96 \geq \zeta_{min}$). We can determine if the filter satisfies the damping ratio requirement for all possible part values by computing μ along the line of complex points $s(\omega) := -\omega \sin(\theta) + j\omega \cos(\theta)$ where $\theta := \arcsin(\zeta_{min})$ [20,21]. At each point along this line, $1/\mu$ is the size of the smallest perturbation (normalized) that causes the filter pole to have a damping ratio of ζ_{min} and a natural frequency ω. If $\mu < 1$ for all points then filter satisfies the damping ratio requirement for all possible part variations.

The Matlab code to create the uncertain transfer function $F(s)$ and compute the μ bounds is provided in Appendix A. μ was evaluated at 100 frequency points linearly spaced between 0.6 and 2.0 rad/sec. Figure 2 shows the lower and upper bounds computed with the GBA and LMI method, respectively. The GBA was run with $N_{try} = 10$ (`'g0'`). The total time to compute upper/lower bounds at all frequency points was 46.5 seconds and the GBA accounted for roughly 7.0 seconds of this total time. We substituted the perturbations returned by the GBA into $F(s)$ and they all give poles whose damping ratio satisfies $|\zeta - \zeta_{min}| < 2 \times 10^{-14}$. The worst case lower bound is 1.107 achieved at a frequency $\omega_k = 1.109$ rad/sec. The part values returned by GBA at this frequency are $R_1^* = 15.93M\Omega$, $R_2^* = 0.54M\Omega$, $C_1^* = 1.22\mu F$, and $C_2^* = 0.077\mu F$. These part values are within the specified tolerances (deviations from nominal are -7.8%, 9.0%, 22.5%, -22.6% for R_1, R_2, C_1, and C_2, respectively) and they place the two filter poles at $-0.784 \pm 0.784j$. These poles have a damping ratio equal to ζ_{min} and hence we conclude that this filter design will not meet the requirement over all part variations.

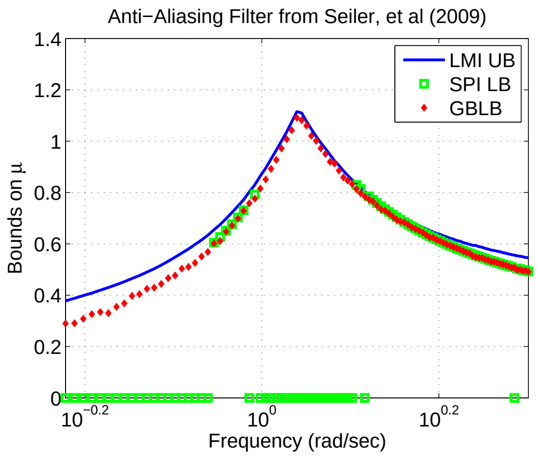

Fig. 2. μ Bounds for Anti-aliasing Filter Damping Ratio

For comparison, we computed the μ lower bounds using the standard power iteration (SPI) with 10 restarts. In this case it took 43.2 seconds to compute the upper/lower bounds with the SPI accounting for approximately 8.6 seconds of this total time. The SPI converged on 58 out of the 100 frequency points (labeled with squares in Figure 2). However, the SPI failed to converge for any frequencies between 1.0 and 1.29 rad/sec. As a result, the best lower bound returned by the power iteration is only 0.81. The results of the analysis are inconclusive when using the SPI. We can neither validate that the filter meets the ζ_{min} requirement (since the μ upper bound exceeds 1) nor can the SPI find a set of part values that demonstrate the violation of the requirement.

Finally, we performed a Monte Carlo search to study the dependence of the filter damping ratio with respect to part variations. We generated 1×10^4 random samples assuming the parts are uniformly distributed over their tolerances. The random search found a filter with damping ratio of 0.692. This confirms that the filter design will not meet the minimum damping ratio requirement. The random search took only 0.96 seconds. In this case, the Monte Carlo search was significantly faster than computing μ bounds. However, Monte Carlo search becomes prohibitive for more complicated filter designs with many uncertain part values. Moreover, the Monte Carlo search can only provide lower bounds on performance, i.e. verifying a design requirement at samples of specific part values does not provide hard guarantees that the filter implementation will satisfy the requirement. Thus this method does not meet the true objectives of the verification process for a safety critical system. The upper bounds provided by μ, on the other hand, can provide rigorous bounds to verify that a filter meets a performance requirement.

4.2 Real μ GBA: Flight Control Robustness

This section provides another example of the GBA performance on a real μ problem. This example, taken from

[4], studies the stability margins for a rigid body transport aircraft with a output feedback control law. The state equations are given by equations 2.1 and 2.2 of [4]. The model includes 14 real uncertainties that represent parametric uncertainties in the aircraft aerodynamic coefficients ($r = 14$ and $k_i = 1$ for $i = 1, \ldots, 14$). μ was evaluated at 100 frequency points logarithmically spaced between 0.1 and 1000 rad/sec. Figure 3 shows the lower and upper bounds computed with the GBA and LMI method, respectively. The GBA was run with $N_{try} = 20$ (`'g'`). The total time to compute upper/lower bounds at all 100 frequency points was 43.1 seconds and the GBA accounted for roughly 30.5 seconds of this total time. We substituted the perturbations returned by the GBA into $F_u(M_R(s), \Delta_R)$ and they all gave $\text{rcond}(I - M_R(j\omega_k)\Delta_k) < tol_{real} = 1 \times 10^{-7}$ where ω_k is the kth point in the frequency grid. Most (90 out of 100 points) gave a reciprocal condition number less than $1{\times}10^{-10}$. The worst case lower bound is 0.18 achieved at the frequency 0.64 rad/sec. The perturbation returned by the GBA at this frequency is $\Delta_R = \text{diag}(5.47, 5.47, -5.47, 5.47, -5.47, -3.35, 5.47, -5.47, 5.47, -5.47, 3.43, 5.47, 0.05, -5.47)$. This perturbation places two poles of $F_u(M_R(s), \Delta_R)$ at $-4.98 \times 10^{-10} \pm 0.64j$. For comparison, it only took 14.5 seconds to compute the LMI upper bound and the lower bound from the standard power iteration with no restarts (SPI). The power iteration took roughly 1.1 seconds of this total time but the iteration failed to converge on any of the frequency points. Increasing the number of SPI restarts to 100 (`'m100'` option) increased the total computation time to 91.3 seconds but only resulted in convergence on 6 out of the 100 frequency points. The lower bounds returned by the SPI with 100 restarts are labeled with squares in Figure 3. The figure also shows the lower bound reported by [4]. This lower bound is 0.18 at the frequency 0.63 rad/sec. It was computed using regularization to improve the convergence of the power iteration. The magnitude of the real parameter uncertainties returned by the power iteration for the regularized problem were then increased to cause the poles of the system to lie at $\pm 0.63j$.

4.3 *Mixed μ GBA*

A procedure for generating mixed μ problems for which it is known *a priori* that $\mu = 1$ (and μ is equal to the LMI upper bound) is described in [18,12]. We tested the GBA on this class of problems with a block structure consisting of r $1{\times}1$ real uncertainties, two $1{\times}1$ complex uncertainties, and a single $2{\times}2$ complex full block. This block structure was also used to test lower bound algorithms in [16,17]. Figure 4 shows the distribution of the GBA lower bounds on 500 problems for $N_{try} = 30$ (`'g2'`) with two and twelve real uncertainties. The distribution of lower bounds from the SPI and the Combined Power Algorithm (CPA) [16] are also shown for the case of two and twelve real uncertainties. The GBA

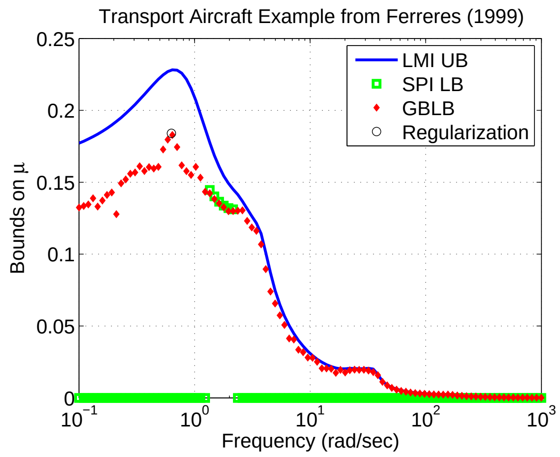

Fig. 3. μ Bounds for Flight Control Robustness

($r = 12$) curve passes through the point $(40, 0.87)$. This means the GBA returned a lower bound less than 0.87 on 40% of the problems and a lower bound ≥ 0.87 on 60% of the problems. The SPI ($r = 12$) curve passes through the point $(40, 0.18)$. This means the SPI returned a lower bound less than 0.18 on 40% of the problems and a lower bound ≥ 0.18 on 60% of the problems. Other points on the curves can be interpreted similarly. For each algorithm the curve for $r = 2$ lies above the corresponding curve for $r = 12$). It should be noted that on this plot a perfect lower bound algorithm appears as a constant horizontal line at $y = 1$.

The performance of the GBA degrades less than the SPI when the number of real blocks is increased from $r = 2$ to $r = 12$. For the $r = 2$ block structure, the GBA performs slightly worse than the CPA. We note that the default choice of tol_{stop} in the GBA causes the algorithm to stop once it achieves a lower bound exceeding 0.97. For the $r = 12$ block structure, the performance of the GBA is superior to that of the SPI. However, the lower bound of the CPA exceeds that of the GBA on approximately 60% of the problems. For this particular block and class of problems, the CPA performance is better on a majority of the test examples. The performance characteristics are dependent on the uncertainty structure and problem class. Thus the relative performance of these three algorithms will change for different problems. The GBA offers another alternative algorithm which may, depending on the particular problem, generate larger lower bounds.

Figure 5 shows the computation time per problem for the GBA and SPI as a function of the number of real blocks, r. This is averaged over 100 problems for each value of r. For the GBA there is some overhead for small problems, but the curve is basically a straight line on the log-log plot for $r \geq 20$. For this particular block structure and for this range of r, the computational cost grows polynomially with the number of real blocks. The computational growth curves will change for different block structures / test matrices.

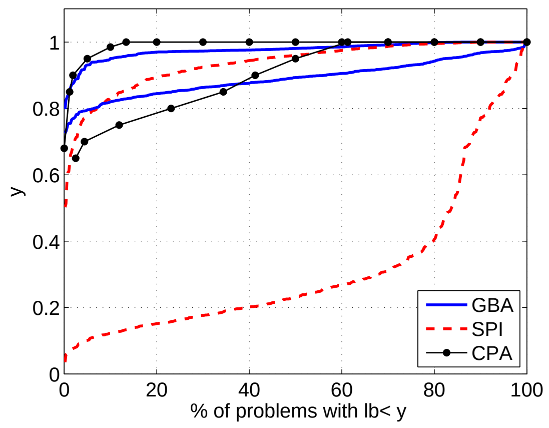

Fig. 4. Mixed μ Lower Bounds: For each algorithm the curves for $r = 2$ are above curves for $r = 12$.

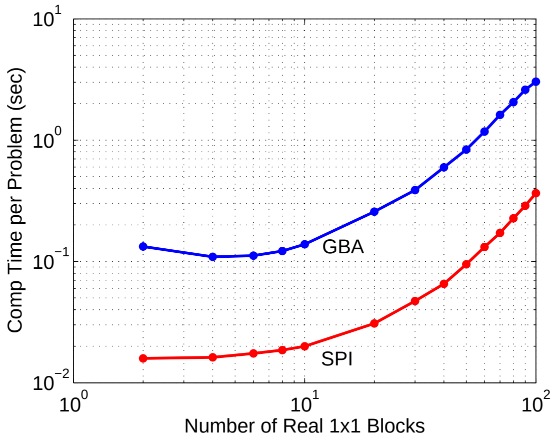

Fig. 5. Computation Time Per Problem

We also studied the computational growth with respect to the real block dimension. We briefly summarize one comparison. Again consider the block structure consisting of r 1×1 real uncertainties, two 1×1 complex uncertainties, and a single 2×2 complex full block. For $r = 10$, the gain based lower bound took 0.14 sec/problem averaged over 100 problems. Increasing the dimension of the real blocks to 10×10 increased the computation time to 3.16 sec/problem. The computation for the standard power iteration grew from 0.02 sec/problem to 0.41 sec/problem. Computational growth times will change for different block structures and test matrices.

5 Conclusions

We presented a new lower bound algorithm for real and mixed μ problems. The algorithm uses a related worst-case gain problem to compute the real blocks and, for mixed μ problems, uses the standard power iteration to compute the complex blocks. The main advantage of this algorithm is that, on the real μ test examples, it has better convergence characteristics than the power iteration and its computational requirements do not grow as rapidly as algorithms which exactly compute μ. The algorithm is integrated into the `mussv` function of Matlab's Robust Control Toolbox and can be called with the `'g'` option. This should enable other users to duplicate the results contained here and/or apply the GBA to their own problems of interest. There are several avenues for future work. First, it would be interesting to investigate the impact of generalizing the disturbance insertion and error pull-off points on algorithm performance. Second, the algorithm could also be modified by incorporating branch and bound methods. Third, the algorithm could be generalized to consider other uncertainty norms, e.g. the 2-norm for ellipsoidal uncertainty. This would require an algorithm to quickly and reliably solve the worst-case gain problem for the uncertainty norm of interest. Finally, it would be interesting to apply this idea to analyze a finite-time control in a batch process.

Acknowledgments

This research was partially supported under a NASA Langley NRA NNH077ZEA001N entitled "Analytical Validation Tools for Safety Critical Systems." The technical contract monitor is Dr. Christine Belcastro.

A Code for Anti-aliasing Filter Example

```
% Create Sallen-Key Filter with uncertain part values
R1 = ureal('R1',17.5,'Percentage',10); % MegaOhms
R2 = ureal('R2',0.5,'Percentage',10);   % MegaOhms
C1 = ureal('C1',1,'Percentage',25);     % microFarads
C2 = ureal('C2',0.1,'Percentage',25);   % microFarads
F = tf(1,[R1*R2*C1*C2, C2*(R1+R2) 1]);  % Filter

% Set the minimum allowable damping ratio
zeta_min = sqrt(2)/2;
theta = asin(zeta_min);

% Since the frequency response is not along the imag. axis,
% get LFT Representation: F(s) = lft(Delta,M(s))
[M,Delta,DeltaBlkStructure] = lftdata(F);
M = M(1:end-1,1:end-1);

% Evaluate M(s) along line of damping ratio zeta_min
NPTS = 100; ww = linspace(0.6,2,NPTS);
Mfr = zeros([size(M) NPTS]);
for i1=1:NPTS
   s = ww(i1)*( -sin(theta)+j*cos(theta) );
   Mfr(:,:,i1) = evalfr(M,s);
end

% Call mussv with options for power iteration with 10 restarts
[bnds1,muinfo1] = mussv(Mfr,DeltaBlkStructure,'am10');

% Call mussv with options for GBA with N_try=10
[bnds2,muinfo2] = mussv(Mfr,DeltaBlkStructure,'ag0');
```

References

[1] J. Doyle, Analysis of feedback systems with structured uncertainties, IEE Proc., Part D 129 (6) (1982) 242–250.

[2] G. Balas, R. Chiang, A. Packard, M. Safonov, Robust Control Toolbox, MathWorks (2009).

[3] A. Packard, J. Doyle, The complex structured singular value, Automatica 29 (1) (1993) 71–109.

[4] G. Ferreres, A practical approach to robustness analysis with aeronautical applications, Kluwer, 1999.

[5] R. Braatz, P. Young, J. Doyle, M. Morari, Computational complexity of μ calculation, IEEE Trans. on Aut. Control 39 (5) (1994) 1000–1002.

[6] J. Demmel, The componentwise distance to the nearest singular matrix, SIAM J. Matrix Anal. Appl. 13 (1) (1992) 10–19.

[7] M. Fu, The real structured singular value is hardly approximable, IEEE Trans. on Aut. Control 42 (9) (1997) 1286–1288.

[8] R. Braatz, E. Russell, Robustness margin computation for large scale systems, Computers & Chem. Eng. 23 (8) (1999) 1021–1030.

[9] A. Packard, M. Fan, J. Doyle, A power method for the structured singular value, in: CDC, 1988, pp. 2132–2137.

[10] P. Young, J. Doyle, Computation of μ with real and complex uncertainties, in: CDC, 1990, pp. 1230–1235.

[11] P. Young, J. Doyle, A lower bound for the mixed μ problem, IEEE Trans. on Aut. Control 42 (1) (1997) 123–128.

[12] P. Young, M. Newlin, J. Doyle, Practical computation of the mixed μ problem, in: ACC, 1992, pp. 2190–2194.

[13] G. Ferreres, V. Fromion, Robustness analysis using the ν tool, in: CDC, 1996, pp. 4566–4570.

[14] R. Holland, P. Young, A skew μ lower bound, in: ACC, 2002, pp. 2753–2758.

[15] R. Holland, P. Young, C. Zhu, Development of a skew μ lower bound, I. J. of Rob. and Nonl. Control 15 (11) (2005) 495–506.

[16] M. Newlin, S. Glavaski, Advances in the computation of μ lower bound, in: ACC, 1995, pp. 442–446.

[17] J. Tierno, P. Young, An improved μ lower bound via adaptive power iteration, in: CDC, 1992, pp. 3181–3186.

[18] P. Young, Robustness with parametric and dynamic uncertainty, Ph.D. thesis, Cal. Tech. (1993).

[19] D. Bates, T. Mannchen, Improved computation of mixed μ bounds for flight control law robustness analysis, Proc. of the IMECHE, Part I, J. of Sys. and Control Eng. 218 (8) (2004) 609–620.

[20] R. Dailey, A new algorithm for the real structured singular value, in: ACC, 1990, pp. 3036–3040.

[21] R. Dailey, D. Gangsaas, Worst-case analysis of flight control systems using the structured singular value, in: AHS, and ASEE, Aircraft Design, Systems and Operations Conf., no. AIAA-1989-2018, 1989.

[22] R. de Gaston, M. Safonov, Exact calculation of the multiloop stability margin, IEEE Trans. on Aut. Control 33 (2) (1988) 156–171.

[23] M. Elgersma, J. Freudenberg, B. Morton, Polynomial methods for the structured singular value with real parameters, I. J. of Rob. and Nonl. Control 6 (1996) 147–170.

[24] M. Hayes, D. Bates, I. Postlethwaite, New tools for computing tight bounds on the real structured singular value, AIAA J. of Guid., Control, and Dyn. 24 (6) (2001) 1204–1213.

[25] J. Kim, D. Bates, I. Postlethwaite, A geometrical formulation of the μ-lower bound problem, IET Control Th. and Appl. 3 (4) (2009) 465–472.

[26] M. Newlin, P. Young, Mixed μ problems and branch and bound techniques, in: CDC, 1992, pp. 3175–3180.

[27] M. Newlin, P. Young, Mixed μ problems and branch and bound techniques, Int. J. of Rob. and Nonl. Control 7 (1997) 145–164.

[28] A. Sideris, R. S. Pena, Fast computation of the multivariable stability margin for the real interrelated uncertain parameters, IEEE Trans. on Aut. Control 34 (12) (1989) 1272–1276.

[29] A. Sideris, R. S. Pena, Robustness margin calculation with dynamic and real parametric uncertainty, IEEE Trans. on Aut. Control 35 (8) (1990) 970–974.

[30] B. Barmish, P. Khargonekar, Z. Shi, Robustness margin need not be a continuous function of the problem data, Sys. and Control Letters 15 (2) (1990) 91–98.

[31] A. Packard, P. Pandey, Continuity properties of the real/complex structured singular value, IEEE Trans. on Aut. Control 38 (3) (1993) 415–428.

[32] R. Braatz, O. Crisalle, Robustness analysis for systems with ellipsoidal uncertainty, I. J. of Rob. and Nonl. Control 8 (1998) 1113–1117.

[33] D. Ma, S. Chung, R. Braatz, Worst-case performance analysis of optimal batch control trajectories, AIChE Journal 45 (7) (1999) 1469–1476.

[34] D. Ma, R. Braatz, Worst-case analysis of finite time control policies, IEEE Trans. on Control Sys. Tech. 9 (5) (2001) 766–774.

[35] Z. Nagy, R. Braatz, Worst-case and distributional robustness analysis of finite-time control trajectories for nonlinear distributed parameter systems, IEEE Trans. on Control Sys. Tech. 11 (5) (2003) 694–704.

[36] A. Packard, G. Balas, R. Liu, J. Shin, Results on worst-case performance assessment, in: ACC, 2000, pp. 2425–2427.

[37] R. Horn, C. Johnson, Topics in Matrix Analysis, Cambridge Univ. Press, 1999.

[38] M. Fan, A. Tits, J. Doyle, Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics, IEEE Trans. on Aut. Control 36 (1) (1991) 25–38.

[39] P. Young, M. Newlin, J. Doyle, μ analysis with real parametric uncertainty, in: CDC, 1991, pp. 1251–1256.

[40] J. Chen, M. Fan, C. Nett, Structured singular values and stability analysis of uncertain polynomials, part 1: the generalized μ, Sys. and Control Letters 23 (1994) 53–65.

[41] J. Chen, M. Fan, C. Nett, Structured singular values and stability analysis of uncertain polynomials, part 2: a missing link, Sys. and Control Letters 23 (1994) 97–109.

[42] P. Young, The rank one mixed μ problem and 'Kharitonov-type' analysis, Automatica 30 (12) (1994) 1899–1911.

[43] P. Horowitz, W. Hill, The Art of Electronics, Cambridge University Press, 1989.

[44] R. Sallen, E. Key, A practical method of designing RC active filters, IRE Trans. on Circuit Theory CT-2 (1955) 74–85.

[45] Radio Technical Commission for Aeronautics, DO-254, Design Assurance Guidance for Airborne Electronic Hardware (2000).