Practical Computation of the Mixed μ Problem

Peter M. Young [*] Matthew P. Newlin [*] John C. Doyle [*]

Abstract

Upper and lower bounds for the mixed μ problem have recently been developed, and this paper examines the computational aspects of these bounds. In particular a practical algorithm is developed to compute the bounds. This has been implemented as a Matlab function (m-file), and will be available shortly in a test version in conjunction with the μ-Tools toolbox. The algorithm performance is very encouraging, both in terms of accuracy of the resulting bounds, and growth rate in required computation with problem size. In particular it appears that one can handle medium size problems (less than 100 perturbations) with reasonable computational requirements.

1 Introduction

In recent years a great deal of interest has arisen with regard to robustness problems involving real (e.g. parametric) uncertainty. These problems involve uncertain parameters that are not only norm bounded, but also constrained to be real. This type of problem can be addressed within the μ framework by extending the definition of μ so as to allow *both* real and complex uncertainties in the block structure. This mixed μ problem can have fundamentally different properties from the more familiar complex μ problem, and these properties have important implications for computation (see [1] for a review of the mixed μ problem).

In particular it is now known that the general (real or) mixed μ problem is NP hard [2], and furthermore it appears that the problem of computing guaranteed bounds may be NP hard as well [3]. These results strongly suggest that any algorithm to compute (bounds for) the mixed μ problem must suffer from worst case exponential (nonpolynomial) growth rate in required computation, or from bounds which are not guaranteed to be tight, so that it appears futile to pursue exact methods for computing μ in the purely real or mixed case for even medium size problems (less than 100 real parameters). Note however that these results do not mean that one cannot develop "practical" algorithms to compute upper and lower bounds for medium size problems, where "practical" means avoiding exponential growth in computation with the number of parameters for any problems which arise in engineering applications. Important issues then become the efficient computation of the bounds and the degree to which they approximate μ, together with techniques for refining the bounds for a better approximation (at an additional computational cost).

Upper and lower bounds for the mixed μ problem have recently been developed [4, 5], and this paper examines the computational aspects of these bounds. In particular a practical algorithm is developed to compute the bounds. The theoretical bounds described in [4, 5] require some reformulation before they can be implemented in an efficient manner, and this is described in sections 3 and 4, together with details of the algorithm construction. The bounds involve solving certain optimization problems, and it is shown that the specific structure of these problems can be exploited so as to speed up the computation considerably. Some results from our extensive numerical experience with the algorithm, regarding both the quality of the bounds and the computation time, are presented in section 6, and it is seen that one can handle medium size problems with reasonable computational requirements. Finally in section 7 we briefly mention some areas for improvement on the present scheme, and in particular the use of these bounds as part of a Branch and Bound scheme to compute guaranteed bounds for the mixed μ problem (see [6]).

2 Notation and Definitions

The notation used here is fairly standard and is essentially taken from [4] and [5]. For any square complex matrix M we denote the complex conjugate transpose by M^*, and the Frobenius norm by $|M|_F$. The largest singular value and the structured singular value are denoted by $\overline{\sigma}(M)$ and $\mu_{\mathcal{K}}(M)$ respectively. The spectral radius is denoted $\rho(M)$ and $\rho_R(M) = max\{|\lambda| : \lambda \textit{ is a real eigenvalue of } M\}$, with $\rho_R(M) = 0$ if M has no real eigenvalues. For a Hermitian matrix M, then $\overline{\lambda}(M)$ denotes the largest (real) eigenvalue. For any complex vector x, then x^* denotes the complex conjugate transpose and $|x|$ the Euclidean norm. We denote the $k \times k$ identity matrix and zero matrix by I_k and O_k respectively.

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

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

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

$$\begin{aligned} X_{\mathcal{K}} = \{\Delta = block\ diag(&\delta_1^r I_{k_1}, \ldots, \delta_{m_r}^r I_{k_{m_r}}, \delta_1^c I_{k_{m_r+1}}, \ldots, \\ &\delta_{m_c}^c I_{k_{m_r+m_c}}, \Delta_1^C, \ldots, \Delta_{m_C}^C) : \\ &\delta_i^r \in \mathbf{R}, \delta_i^c \in \mathbf{C}, \Delta_i^C \in \mathbf{C}^{k_{m_r+m_c+i} \times k_{m_r+m_c+i}}\} \end{aligned} \quad (2)$$

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

Definition 1 ([7]) *The structured singular value, $\mu_{\mathcal{K}}(M)$, of a matrix $M \in \mathbf{C}^{n\times n}$ with respect to a block structure $\mathcal{K}(m_r, m_c, m_C)$ is defined as*

$$\mu_{\mathcal{K}}(M) = \left(\min_{\Delta \in X_{\mathcal{K}}} \{\overline{\sigma}(\Delta) : det(I - \Delta M) = 0\}\right)^{-1} \quad (3)$$

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

In order to develop the upper and lower bounds for μ we need to define some sets of block diagonal scaling matrices (which are also dependent on the underlying block structure).

$$\mathcal{Q}_{\mathcal{K}} = \{\Delta \in X_{\mathcal{K}} : \delta_i^r \in [-1\ 1], \delta_i^{c*}\delta_i^c = 1, \Delta_i^{C*}\Delta_i^C = I_{k_{m_r+m_c+i}}\} \quad (4)$$

$$\begin{aligned} \mathcal{D}_{\mathcal{K}} = \{block\ diag(&D_1, \ldots, D_{m_r+m_c}, d_1 I_{k_{m_r+m_c+1}}, \ldots, \\ &d_{m_C} I_{k_m}) : 0 < D_i = D_i^* \in \mathbf{C}^{k_i\times k_i}, 0 < d_i \in \mathbf{R}\} \end{aligned} \quad (5)$$

$$\begin{aligned} \mathcal{G}_{\mathcal{K}} = \{block\ diag(&G_1, \ldots, G_{m_r}, O_{k_{m_r+1}}, \ldots, O_{k_m}) : \\ &G_i = G_i^* \in \mathbf{C}^{k_i\times k_i}\} \end{aligned} \quad (6)$$

[*]Electrical Engineering, 116-81, California Institute of Technology, Pasadena, CA 91125. This work was supported by ONR, NSF, NASA, and Rockwell International.

$$\hat{\mathcal{D}}_{\mathcal{K}} = \{block\ diag(D_1, \ldots, D_{m_r+m_c}, d_1 I_{k_{m_r+m_c+1}}, \ldots, d_{m_C} I_{k_m}) : det(D_i) \neq 0, D_i \in \mathbf{C}^{k_i \times k_i}, d_i \neq 0, d_i \in \mathbf{C}\} \tag{7}$$

$$\hat{\mathcal{G}}_{\mathcal{K}} = \{block\ diag(g_1, \ldots, g_{n_r}, O_{n_c}) : g_i \in \mathbf{R}\} \tag{8}$$

where $n_r = \sum_{i=1}^{m_r} k_i$ and $n_c = n - n_r$.

3 The Lower Bound for Mixed μ

The theoretical basis for the mixed μ lower bound lies in the fact that the μ problem may be reformulated as a real eigenvalue maximization. The following theorem is taken from [5].

Theorem 1 ([5]) *For any matrix $M \in \mathbf{C}^{n \times n}$, and any compatible block structure $\mathcal{K}$*

$$\max_{Q \in \mathcal{Q}_{\mathcal{K}}} \rho_R(QM) = \mu_{\mathcal{K}}(M) \tag{9}$$

This immediately gives us a theoretical lower bound since we have that for any $Q \in \mathcal{Q}_{\mathcal{K}}$, $\rho_R(QM) \leq \mu_{\mathcal{K}}(M)$. The idea then is to find an efficient way to compute a local maximum of the function $\rho_R(QM)$ over $Q \in \mathcal{Q}_{\mathcal{K}}$. It turns out that this can be done by means of a power iteration. The iteration scheme usually converges fairly rapidly, and each iteration of the scheme is very cheap, requiring only such operations as matrix-vector multiplications and vector inner products. This gives rise to a lower bound algorithm which is much faster than would be obtained by directly solving (9) via standard optimization techniques (although this maximization is carried out implicitly by the power iteration). The theoretical development of the power iteration is fully described in [5] and we will not go into any of the details here.

In fact this scheme is a very simple power iteration, and although it usually converges to a satisfactory equilibrium point, the convergence is not always guaranteed. However in all cases one can still obtain a candidate mixed perturbation from the iteration scheme. From this one can compute a lower bound (provided that the mixed μ problem contains some complex uncertainty) by simply wrapping in the real perturbations, and then evaluating the spectral radius of the associated complex μ problem, scaled by the candidate complex perturbations. This scheme is implemented in the "rmu" code so that the algorithm always returns a valid lower bound, regardless of convergence. In fact the performance data presented in section 6 was collected regardless of whether or not the power algorithm converged on the problem (i.e. no data points were excluded).

Of course it is still desirable that the power iteration converges, since in that case it is more likely that the lower bound obtained is a good one. It is well known that the convergence properties of standard eigenvalue and singular value power algorithms (which can be obtained as special cases of this algorithm) can be improved by inverse iteration, and similar adaptations to the mixed μ power algorithm are being investigated. Preliminary results have shown an improvement in the convergence properties, and it is hoped that further refinements will enable the convergence to a local maximum of (9) to be guaranteed [8]. Note that we cannot expect to be able to guarantee to find the global maximum of (9), since the problem is not convex.

4 The Upper Bound for Mixed μ

The standard upper bound for complex μ involves a singular value minimization problem, or equivalently an eigenvalue minimization problem on a Hermitian matrix, with respect to a certain "D scaling matrix" [7]. The mixed μ upper bound takes the form of a more complicated version of the same problem, now involving an additional "G scaling matrix". The following theorem is taken from [4] (though stated here in a slightly different form).

Theorem 2 ([4]) *For any matrix $M \in \mathbf{C}^{n \times n}$, and any compatible block structure $\mathcal{K}$ suppose α_* is the result of the minimization problem*

$$\alpha_* = \inf_{\substack{D \in \mathcal{D}_{\mathcal{K}} \\ G \in \mathcal{G}_{\mathcal{K}}}} \left[\min_{\alpha \in \mathbf{R}} \{\alpha : (M^*DM + j(GM - M^*G) - \alpha D) \leq 0\} \right] \tag{10}$$

then $\mu_{\mathcal{K}}(M) \leq \sqrt{\max(0, \alpha_*)}$

Note that the above minimization involves a LMI (Linear Matrix Inequality), and hence it is convex. Note also that if we impose the restriction $G = 0_n$, we recover the standard complex μ upper bound.

Since the upper bound is a convex problem there are a whole array of numerical techniques one could use to tackle this minimization. Note however that for even medium size problems ($n < 100$) then depending on the block structure $\mathcal{K}$, the optimization over the D and G scaling matrices could involve optimizing several thousand parameters. Therefore, in order to tackle such problems with reasonable computation times, a straightforward application of brute force optimization techniques will not suffice. Instead we will exploit the specific structure of this problem, so as to develop an efficient algorithm, which can handle problems of this size.

The algorithm implementation relies heavily on the fact that the upper bound may be reformulated several different ways, as stated in the following theorem.

Theorem 3 *Suppose we have a matrix $M \in \mathbf{C}^{n \times n}$ and a real scalar $\beta > 0$, and for any $D \in \mathbf{C}^{n \times n}$ denote $M_D \doteq DMD^{-1}$. Then the following statements are equivalent:*

I. There exist matrices $D_I \in \mathcal{D}_{\mathcal{K}}, G_I \in \mathcal{G}_{\mathcal{K}}$ such that:

$$\overline{\lambda}\left(M^*D_IM + j(G_IM - M^*G_I) - \beta^2 D_I\right) \leq 0 \tag{11}$$

II. There exist matrices $D_{II} \in \hat{\mathcal{D}}_{\mathcal{K}}, G_{II} \in \hat{\mathcal{G}}_{\mathcal{K}}$ (or $D_{II} \in \mathcal{D}_{\mathcal{K}}, G_{II} \in \mathcal{G}_{\mathcal{K}}$) such that:

$$\overline{\lambda}\left(M_{D_{II}}^* M_{D_{II}} + j(G_{II}M_{D_{II}} - M_{D_{II}}^* G_{II})\right) \leq \beta^2 \tag{12}$$

III. There exist matrices $D_{III} \in \hat{\mathcal{D}}_{\mathcal{K}}, G_{III} \in \hat{\mathcal{G}}_{\mathcal{K}}$ (or $D_{III} \in \mathcal{D}_{\mathcal{K}}, G_{III} \in \mathcal{G}_{\mathcal{K}}$) such that:

$$\overline{\sigma}\left(\left(\frac{M_{D_{III}}}{\beta} - jG_{III}\right)(I_n + G_{III}^2)^{-\frac{1}{2}}\right) \leq 1 \tag{13}$$

IV. There exist matrices $D_{IV} \in \hat{\mathcal{D}}_{\mathcal{K}}, G_{IV} \in \hat{\mathcal{G}}_{\mathcal{K}}$ (or $D_{IV} \in \mathcal{D}_{\mathcal{K}}, G_{IV} \in \mathcal{G}_{\mathcal{K}}$) such that:

$$\overline{\sigma}\left((I_n + G_{IV}^2)^{-\frac{1}{4}}\left(\frac{M_{D_{IV}}}{\beta} - jG_{IV}\right)(I_n + G_{IV}^2)^{-\frac{1}{4}}\right) \leq 1 \tag{14}$$

The equivalence between I, II, III for $D_I, D_{II}, D_{III} \in \mathcal{D}_{\mathcal{K}}$ and $G_I, G_{II}, G_{III} \in \mathcal{G}_{\mathcal{K}}$ was shown in [4]. Note also that we can easily obtain the formulae to convert between the various forms (there are several more equivalent forms, slight variations on the above, which can also easily be obtained).

These different formulations, whilst mathematically equivalent, have quite different numerical properties. For the purposes of developing an upper bound algorithm, we will be concerned mostly with the formulations in (11) and (14). It is clear from (14) that as an alternative to carrying out the minimization in (10) we could compute the 'minimum' $\beta > 0$ such that

$$\inf_{\hat{D} \in \hat{\mathcal{D}}_{\mathcal{K}}, \hat{G} \in \hat{\mathcal{G}}_{\mathcal{K}}} \overline{\sigma}\left((I + \hat{G}^2)^{-\frac{1}{4}}\left(\frac{\hat{D}M\hat{D}^{-1}}{\beta} - j\hat{G}\right)(I + \hat{G}^2)^{-\frac{1}{4}}\right) \leq 1 \tag{15}$$

Note that the theoretical equivalence of the two problems breaks down at $\beta = 0$ (and so for these cases strictly speaking there is no 'minimum' β) but this presents no problem for a practical computation scheme since we merely quit if the upper bound falls below some prespecified tolerance (which can be arbitrarily small). Each of these two different formulations of the upper bound problem has its own advantages. The problem statement from (10) has the advantages that it is linear in the matrices D and G, and is convex (and hence one will not have problems associated with local minima). The problem statement from (15) has the advantages that one is trying to minimize the norm of a given matrix (which offers some numerical advantages), that $\hat{D}$ enters the problem exactly as in the standard complex μ upper bound, that $\hat{G}$ enters the problem in a balanced symmetric fashion, and that $\hat{G}$ is now a real diagonal matrix.

The upper bound algorithm implemented here uses a mixture of the formulations in theorem 3. Initially we tackle the problem in the form of (15). Here we can use some methods from the complex μ bounds, together with various other techniques, to obtain fairly good estimates of $\hat{D}$,$\hat{G}$ and β. These are then converted into an initial guess for the problem in the form of (10) and the algorithm then proceeds to improve on these. More specifically the algorithm structure is as follows:

1. First we balance the matrix. This involves computing $\hat{D}$ to solve $\inf_{\hat{D}\in\hat{\mathcal{D}}_K} |\hat{D}M\hat{D}^{-1}|_F$ using a generalization of Osborne's method [9] (as in the standard complex μ upper bound). The matrix $\hat{M} \doteq \hat{D}M\hat{D}^{-1}$ is then balanced, and this procedure generates our initial guess for $\hat{D} \in \hat{\mathcal{D}}_K$.
2. The lower bound is now computed using the algorithm from section 3, applied to the *balanced* matrix $\hat{M}$.
3. Now we have a lower bound, and $\overline{\sigma}(\hat{M})$ serves as a first guess for the upper bound. This is then improved upon in the following way. For any fixed level of β compute each block of $\hat{G}$ as $\hat{G}_i = \frac{1}{2j\beta}(\hat{M}_i - \hat{M}_i^*)$ where $\hat{M}_i$ is the corresponding sub-matrix of $\hat{M}$ (i.e. $j\hat{G}_i$ cancels the Skew-Hermitian part of $\hat{M}_i$). Then bisect on β between the lower and current upper bound to find the smallest β such that
$$\overline{\sigma}\left((I+\hat{G}^2)^{-\frac{1}{4}}\left(\frac{\hat{M}}{\beta}-j\hat{G}\right)(I+\hat{G}^2)^{-\frac{1}{4}}\right)\leq 1$$
Finally perform an eigenvalue decomposition on $\hat{G}$ as $\hat{G} = U\Lambda U^*$ (with U Unitary, Λ diagonal and real), and convert to $\hat{G} \in \hat{\mathcal{G}}_K$ by redefining $\hat{G}$ as Λ and absorbing the U matrix into $\hat{D} \in \hat{\mathcal{D}}_K$ and $\hat{M}$.
4. We now have initial guesses for $\hat{D} \in \hat{\mathcal{D}}_K$ and $\hat{G} \in \hat{\mathcal{G}}_K$. The next step is to compute a descent direction for $\hat{G} \in \hat{\mathcal{G}}_K$, together with an appropriate step length, and a new $\hat{G}$ is computed by taking this descent step. This entire procedure is then repeated once more.
5. The matrix $\hat{D} \in \hat{\mathcal{D}}_K$ is updated by computing a *diagonal* matrix $\hat{D}_d \in \hat{\mathcal{D}}_K$ (so that it commutes with $\hat{G} \in \hat{\mathcal{G}}_K$) which minimizes
$$\inf_{\substack{\hat{D}_d\in\hat{\mathcal{D}}_K\\ \hat{D}_d \text{ diagonal}}}\left|\hat{D}_d(I+G^2)^{-\frac{1}{4}}\left(\frac{\hat{M}}{\beta}-jG\right)(I+G^2)^{-\frac{1}{4}}\hat{D}_d^{-1}\right|_F$$
again using a generalized Osborne's method. We then absorb $\hat{D}_d$ into $\hat{D} \in \hat{\mathcal{D}}_K$.
6. Step 4 is repeated.
7. We now have guesses for $\hat{D} \in \hat{\mathcal{D}}_K, \hat{G} \in \hat{\mathcal{G}}_K$ and β for the upper bound problem in (15). These are converted into $D \in \mathcal{D}_K, G \in \mathcal{G}_K$ which form guesses for the upper bound problem in (10). We now improve these guesses using a descent algorithm, which iteratively computes a descent direction, and an appropriate step length, for both $D \in \mathcal{D}_K$ and $G \in \mathcal{G}_K$ *simultaneously*. At each step we compute a new upper bound by solving the associated eigenvalue problem, and quit when the bound stops decreasing (within tolerance).

The balancing in step 1 of the algorithm serves several purposes. Firstly we obtain a $\hat{D} \in \hat{\mathcal{D}}_K$ which approximately solves $\inf_{\hat{D}\in\hat{\mathcal{D}}_K} \overline{\sigma}(\hat{D}M\hat{D}^{-1})$, or in other words the standard upper bound for the associated complex μ problem. Since we have reformulated the problem in (15) so that the $\hat{D}$ matrix enters exactly as in the complex μ upper bound, and the $\hat{G}$ matrix enters in a balanced symmetric fashion, this $\hat{D}$ matrix also serves as a good first guess for the mixed μ upper bound. A good deal of numerical experience with the generalized Osborne's method for computing complex μ upper bounds has shown that is very fast and usually works well, and so by reformulating the problem in this fashion we can exploit these properties in the mixed problem as well. This balancing also numerically preconditions the problem, and can greatly improve the performance of the subsequent steps.

Step 3 of the algorithm generates our initial guess for $\hat{G}$. The approach is somewhat intuitive, but although there are no firm guarantees, it appears in general to work quite well. Thus our $\hat{D},\hat{G}$ estimates, which require very little computation time, are usually quite good *before* we enter the descent portion of the algorithm, and hence we can restrict ourselves to a small number of descent steps. This is crucial in obtaining a fast implementation, since the descent steps are quite computationally expensive.

Note that in step 7 we are required to compute a descent direction for $D \in \mathcal{D}_K, G \in \mathcal{G}_K$, together with an appropriate step length. We compute *matrix* descent directions for D, G in one shot by computing a generalized gradient of the upper bound function (details will appear elsewhere). In this way we avoid separate computation for the individual elements of the D, G matrices. This is important not only for speed of computation, but also because in the case of repeated eigenvalues there may not be a descent direction with respect to any individual elements of D, G, when there is a descent direction if all the elements are allowed to move *simultaneously*. In the case that the maximum eigenvalue is distinct, then this descent direction coincides with the usual gradient direction. The step length computation is somewhat ad-hoc, but ensures that the maximum eigenvalue of the upper bound function decreases, and that we satisfy the constraint $D > 0$. Similar comments with regard to the computation of descent directions and step lengths apply to steps 4 and 6.

This implementation of the upper bound results in an algorithm which is quite efficient, and can handle medium size problems ($n < 100$) with reasonable computational requirements. It has been implemented as as Matlab function (m-file) "rmu", and will be available shortly in a test version in conjunction with the μ-Tools toolbox [10]. The software returns upper and lower bounds for $\mu_K(M)$, together with appropriate scaling matrices $\hat{D} \in \hat{\mathcal{D}}_K$, $\hat{G} \in \hat{\mathcal{G}}_K$ for the upper bound problem in (15), and $Q \in \mathcal{Q}_K$ for the lower bound problem (9). Results regarding both the quality of the bounds and their computational requirements (as a function of problem size) are presented in section 6.

The mixed μ upper bound (in the form of (10)) can be viewed as a special case of a class of LMI problems. The solution of LMI's is a subject of much research interest right now [11], since they appear in many control problems. This algorithm represents a first attempt at solving one particular LMI. As more refined algorithms for the solution of LMI's appear, then they can be used to improve the μ upper bound computation.

5 Generating Test Matrices

It was stated in section 1 that the mixed μ problem is NP hard, which implies that the worst case performance of our (or any other) algorithm will be poor, either in terms of the accuracy of the bounds, or the growth rate in computation. In fact we can construct examples for which the bounds in theorems 1 and 2 are arbitrarily far apart. For engineering purposes then the real issue becomes whether or not we can develop a "practical" algorithm, whose *typical* performance is acceptable. In order to examine the typical performance in section 6, we will run the algorithm repeatedly on a large number of test matrices, randomly generated from within certain classes, and collect statistical data. In this section we describe three specific types of random matrices that will be used.

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

Unfortunately it is doubtful that crand matrices are very representative of those of practical interest. Since the matrices that the μ software will be run on are typically obtained from control problems, a fairly natural class of random complex matrices is to randomly generate State Space 'A,B,C,D' matrices using the μ-Tools "sysrand" command, and then evaluate the transfer matrix at some frequency (usually placed roughly in the middle of the modes). We will refer to this type of random matrix as a "sysrand" matrix.

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

1. Randomly generate matrices $D \in \mathcal{D}_{\mathcal{K}}, G \in \mathcal{G}_{\mathcal{K}}$ and $Q \in \mathcal{Q}_{\mathcal{K}}$ with the added restriction that $Q^*Q = I_n$. In addition randomly generate a Unitary matrix $Y \in \mathbf{C}^{n\times n}$, and a real nonnegative diagonal matrix $\Sigma = diag(\sigma_1 \ldots \sigma_n)$ with
$$\begin{aligned} \sigma_i &= 1 \quad for \quad i = 1 \ldots r \\ \sigma_i &< 1 \quad for \quad i = r+1 \ldots n \end{aligned} \tag{16}$$
where r is some integer satisfying $1 \le r \le n$. Finally generate a random unit norm vector $\eta \in \mathbf{C}^n$ with the restriction that:
$$\eta_i = 0 \quad for \quad i = r+1 \ldots n \tag{17}$$
2. Compute $X \in \mathbf{C}^{n\times n}$ as *any* Unitary matrix which satisfies the equation
$$X\eta = (Q^{-1} - jG)(I_n + G^2)^{-\frac{1}{2}} Y\eta \tag{18}$$
It is easy to check that the matrix $(Q^{-1} - jG)(I_n + G^2)^{-\frac{1}{2}} Y$ is Unitary, so that this is always possible.
3. Compute $M \in \mathbf{C}^{n\times n}$ as
$$M = D^{-1}\left(X\Sigma Y^*(I_n + G^2)^{\frac{1}{2}} + jG\right) D \tag{19}$$

Theorem 4 *Suppose we have a matrix $M \in \mathbf{C}^{n\times n}$ and a block structure $\mathcal{K}$. Then denoting the upper bound from theorem 2 by $\hat{\mu}$, we have that the following two conditions are equivalent:*

1. There exist matrices $D \in \mathcal{D}_{\mathcal{K}}$ and $G \in \mathcal{G}_{\mathcal{K}}$ achieving the infimum in theorem 2, and $\hat{\mu}_{\mathcal{K}}(M) = \mu_{\mathcal{K}}(M) = 1$.

2. M can be generated by the above algorithm.

The above algorithm was first developed for the purely complex case in [12]. Note that we can control the number of eigenvalues coalesced at the minimum of the upper bound function in theorem 2, and a simple extension to the algorithm allows us to also control the number of eigenvalues coalesced at the maximum of the lower bound function in theorem 1.

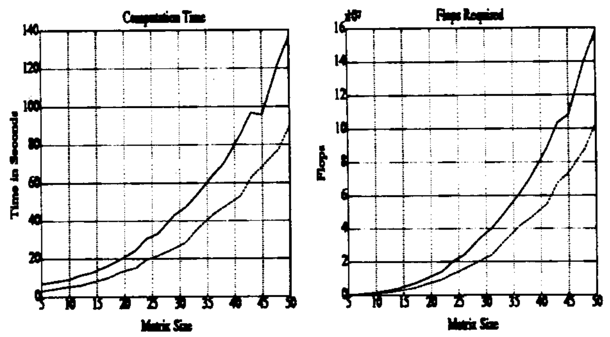

Figure 1: Typical computation requirements versus matrix size for mixed-μ problem (solid) and complex-μ problem (dashed).

Roughly speaking this algorithm allows us to randomly generate (all) problems with μ equal to its upper bound, equal to one (together with optimal scaling matrices achieving the upper and lower bounds). Note that for these problems there is no gap between the bounds from theorems 1 and 2, although the optimal lower bound requires the solution of a non convex maximization problem. We will refer to a random matrix generated by the above algorithm as a "nogap" matrix.

6 Algorithm Performance

There are many questions one could ask with regard to the algorithm performance, both in terms of computation time and accuracy of the resulting bounds. We decided to focus on the algorithm performance versus matrix size for a fixed set of uncertainty descriptions. The first test performed was to examine the average computational requirements for the algorithm implemented in Matlab. For this purpose we used crand matrices (although the results are not too different for the different classes). The computational requirements versus matrix size are shown in Figure 1 for block structures consisting of all scalar uncertainties, with 90% of them chosen as real and the rest complex. The same data for the appropriate complex μ problem is shown for comparison. The results were obtained running Matlab on a Sparc 1 workstation, and it can be seen that we can reasonably expect to handle problems of size 10 in about 10 seconds, up to problems of size 50 in about 2-3 minutes.

It can also be seen that the (experimental) growth rate in computation time for the existing implementation is approximately n^2. This is probably an artifice of the implementation in Matlab, which is an interpretive language. A more realistic measure of the computational growth rate is in terms of total floating point operations (flops). If this measure is adopted then it is seen that the (experimental) growth rate in flops is approximately n^3. In any case the algorithm growth rate appears reasonable whether measured in terms of time or flops required.

The next set of tests performed was aimed at evaluating the accuracy of the bounds. This time we compared the upper and lower mixed μ bounds, and also the mixed μ and complex μ upper bounds. The complex μ bounds were obtained by simply replacing all the real perturbations with complex ones, but without changing the matrix. Thus the complex upper bound is strictly larger than the mixed upper bound. The results are shown for sysrand matrices in Figure 2. It can be seen that the bounds are reasonably tight, even for the largest ($n = 50$) problems. Note also that we have a fairly wide spread of values for the gap between complex μ and mixed μ. The results for crand matrices were similar, except that there was typically not much of a gap between mixed μ and complex μ (see [1]).

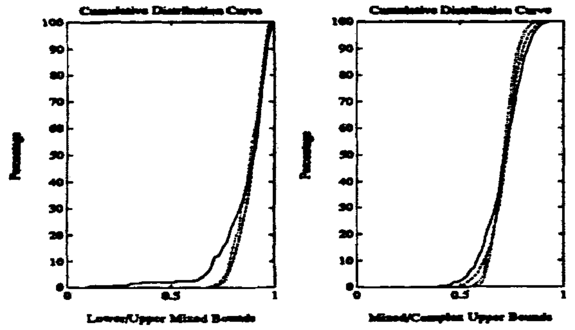

Figure 2: Typical ratios of mixed-μ lower to upper bounds, and mixed-μ to complex-μ upper bounds, for sysrand matrices of sizes 10 (solid), 20 (dashed), 30 (dotted), and 50 (dashdot).

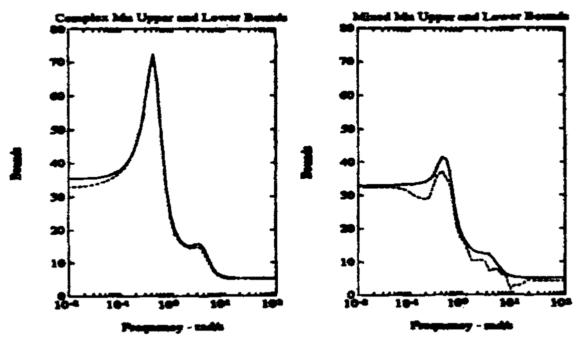

Figure 3: Complex-μ and mixed-μ upper and lower bounds versus frequency for a random system.

A number of tests were performed using the nogap matrices, and it was found that the upper bound computation was typically with 1-2% of the optimum for these matrices. The lower bound performance was not as good, and in fact the lower bound power iteration can fail to converge on this type of matrix, and yield a poor bound. Of course we cannot expect that our lower bound routine is *guaranteed* to find the correct answer, since it is attempting to maximize a non convex problem.

As a further test the bounds for the mixed μ problem were evaluated across a frequency range for some random systems (generated with "sysrand"), and compared to the bounds for the appropriate complex μ problem. Again the bounds seemed reasonably tight, and a typical example plot is shown in Figure 3.

The algorithm was also tested on a variety of other block structures and the properties appear similar to those described above. An exception to this is the pure real case ($m_c = m_C = 0$), which appears to have significantly poorer properties than any other. There are important reasons for this that seem inherent to the problem, not the computation scheme (see [1]).

Note that the above tests were aimed at evaluating the typical performance of the algorithm on an essentially random selection of problems, and it appears that the algorithm is performing well for most problems. This does not mean however that one will never encounter mixed μ problems where the gap between the upper and lower bounds is large, and it can be seen from Figure 2 that a few such cases were found.

In addition to the numerical tests described here the "rmu" software has been applied to a number of practical problems, arising from real physical systems. These include analysis of natural frequency variations for flexible structures [13], and variation of missile autopilot dynamics with angle of attack and Mach number [14]. The software worked well for these problems, providing tight bounds for the associated mixed μ problems, and is now being β-tested at sites including Honeywell, Phillips, NASA Dryden and several universities. This will provide additional experience regarding the algorithm's performance on real engineering problems.

7 Future Directions

Although the algorithm presented here will usually provide bounds that are accurate enough for engineering purposes, in a significant number of cases of interest, it will not. One possibility for such problems is to improve the algorithm for computing the bounds in theorems 1 and 2. This has been briefly discussed here, and we refer the reader to [8] for the use of adaptive power iteration to improve the lower bound performance, and [11] for the use of LMI techniques to improve the upper bound computation.

Note however that the bounds from theorems 1 and 2 may be far apart (regardless of the computation method). For these cases we must consider improving the bounds themselves. A promising approach is to use the existing bounds as part of a Branch and Bound scheme, which iteratively refines them. In this way we can develop a scheme to compute *guaranteed* bounds for the mixed μ problem. Since the problem is NP hard we must expect that the worst case computation time for such a scheme will be exponential. The real issue is whether or not we can produce a "practical" scheme, whose *typical* computation time is polynomial. We believe that it is possible to develop such a scheme, using the algorithm presented here, and this will be further pursued in [6].

References

[1] P. M. Young, M. P. Newlin, and J. C. Doyle, "μ analysis with real parametric uncertainty," in *Proceedings of the 30th Conference on Decision and Control*, pp. 1251–1256, IEEE, 1991.

[2] J. Rohn and S. Poljak, "Radius of nonsingularity." to appear in Mathematics of Control, Signals and Systems.

[3] J. Demmel, "The componentwise distance to the nearest singular matrix." to appear in SIAM Journal on Matrix Analysis and Applications.

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

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

[6] M. P. Newlin and P. M. Young, "Mixed μ problems and Branch and Bound techniques." submitted to 31st Conference on Decision and Control, 1992.

[7] J. Doyle, "Analysis of feedback systems with structured uncertainty," *IEE Proceedings, Part D*, vol. 129, pp. 242–250, Nov. 1982.

[8] J. E. Tierno and P. M. Young, "An improved μ lower bound via adaptive power iteration." submitted to 31st Conference on Decision and Control, 1992.

[9] E. E. Osborne, "On preconditioning of matrices," *Journal of the Association for Computing Machinery*, vol. 7, pp. 338–345, 1960.

[10] G. J. Balas, A. Packard, J. C. Doyle, K. Glover, and R. Smith, "Development of advanced control design software for researchers and engineers," in *Proceedings of the American Control Conference*, pp. 996–1001, 1991.

[11] C. Beck and J. C. Doyle, "Mixed μ upper bound computation using LMI optimization." submitted to 31st Conference on Decision and Control, 1992.

[12] M. K. H. Fan and A. L. Tits, "Characterization and efficient computation of the structured singular value," *IEEE Transactions on Automatic Control*, vol. AC-31, pp. 734–743, 1986.

[13] K. Lim and G. J. Balas, "Line-of-sight control of the CSI evolutionary model: $\mathcal{H}_\infty$ and μ control." to be presented at the American Control Conference, 1992.

[14] G. J. Balas and A. K. Packard, "Development and application of time-varying μ-synthesis techniques for control design of missile autopilots." John Hopkins Applied Physics Laboratories, Final Report, January, 1992.