Systems Modeling Analysis and Control (SMAC) Toolbox: a complete software suite to compute tight bounds on the (skewed) structured singular value

Clément Roos, Jérémy Lesprier and Jean-Marc Biannic

GT MOSAR meeting, April 23rd, 2015

ONERA

THE FRENCH AEROSPACE LAB

retour sur innovation

Context and objectives

Since its introduction, μ-analysis has been extensively studied both in the academic and in the industrial worlds.

Several methods have been developed in the last 30 years in order to tackle the problem of computing the structured singular value μ.

Due to NP-hardness, bounds are computed instead of the exact value.

1. Most of the computationally tractable techniques to compute μ upper bounds rely on the so-called (D, G) scalings formulation.
   $\rightarrow$ the resulting bounds are sometimes quite conservative
   $\rightarrow$ the computational time is sometimes quite large

2. A wide number of very different approaches have been developed to compute μ lower bounds.
   $\rightarrow$ no extensive comparison is available

Context and objectives

Main contributions

- Present a thorough comparison of the most significant μ lower bound algorithms on a wide set of real-world applications.
- Propose simple improvements to these algorithms to approach the exact value of μ with a reasonable computational cost.
- Develop several techniques to reduce the gap between μ and its upper bound with a moderate computational time.

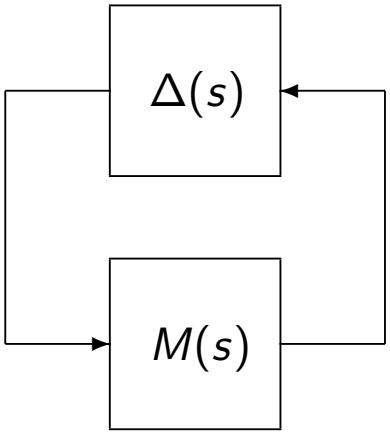
Objectives

- Compute the (almost) exact value of μ in (almost) all cases, even for medium/large size problems addressed by control engineers.
- Propose a user-friendly Matlab Toolbox, which implements state-of-the-art algorithms.

Outline

Problem statement

Stability/performance of LTI systems with time-invariant uncertainties.



$M(s)$ is a stable and proper real-rational transfer function $\Rightarrow$ nominal system.

$\Delta(s) = \mathrm{diag}(\Delta_1(s), \ldots, \Delta_N(s))$ is a block-diagonal operator $\Rightarrow$ model uncertainties. $\Delta_i(s)$ can be:

- a time-invariant diagonal matrix $\Delta_i(s) = \delta_i I_{n_i}$, where δ_i is a real/complex parametric uncertainty,
- a stable and proper real-rational unstructured transfer function representing neglected dynamics.

$\mathbf{\Delta}$ is the set of all matrices with the same block-diagonal structure and the same nature (real or complex) as $\Delta(j\omega) \Rightarrow$ admissible uncertainties.

$k\mathcal{B}_{\mathbf{\Delta}} = \{\Delta \in \mathbf{\Delta} : \overline{\sigma}(\Delta) < k\} \Rightarrow$ maximum size of the uncertaintes.

Problem statement

Definition

Let $\omega \in \mathbb{R}_+$ be a given frequency. If no matrix $\Delta \in \mathbf{\Delta}$ makes $I - M(j\omega)\Delta$ singular, then the structured singular value $\mu_{\mathbf{\Delta}}(M(j\omega))$ is zero. Otherwise:

$$\mu_{\mathbf{\Delta}}(M(j\omega)) = \left[\min_{\Delta \in \mathbf{\Delta}} \{\overline{\sigma}(\Delta),\ \det(I - M(j\omega)\Delta) = 0\} \right]^{-1}$$

Lemma

The interconnection $M(s) - \Delta(s)$ is stable $\forall \Delta(s) \in k_r \mathcal{B}_{\mathbf{\Delta}}$, where the robust stability margin k_r is defined as the inverse of the largest value of $\mu_{\mathbf{\Delta}}(M(j\omega))$ over the whole frequency range:

$$k_r = \left[\max_{\omega \in \mathbb{R}_+} \mu_{\mathbf{\Delta}}(M(j\omega)) \right]^{-1}$$

The main objective of this work is to determine k_r, *i.e.* to compute the best possible μ upper and lower bounds over the whole frequency range.

Outline

Survey of existing μ lower bound algorithms

All methods that can be applied to real-world benchmarks are mentioned.

μ lower bounds and destabilizing values of $\Delta(s)$ are obtained in all cases.

1 Power algorithm

- a non-concave optimization problem is solved: $\mu_{\mathbf{\Delta}}(M) = \max_{Q \in \mathcal{Q}} \rho_R(QM)$
- a local maximum (μ lower bound) is computed using a fixed point iteration

2 Gain-based algorithm

- the problem is reformulated as a worst-case H_∞ performance problem
- real uncertainties are computed so as to bring the H_∞ norm to infinity
- complex uncertainties are obtained using the power algorithm

3 Poles migration techniques

- the idea is to move an eigenvalue of the state matrix A_0 of the interconnection between $M(s)$ and Δ towards the imaginary axis
- **first approach:** use of a first-order characterization of the variation $d\lambda_i$ in the ith eigenvalue $\lambda_i(A_0)$ of A_0 caused by a small variation $d\Delta$ of Δ

Survey of existing μ lower bound algorithms

- a series of such perturbations $d\Delta$ are computed, which progressively move the eigenvalues towards the imaginary axis
- two algorithms exist:
  1. the uncertainty with minimum Froebenius norm which brings an eigenvalue on the imaginary axis is determined, and then the one with minimum $\overline{\sigma}$ norm such that the system remains at the limit of stability
  2. the power algorithm is applied for a few frequencies on a regularized problem (addition of a small amount of complex uncertainty) ; using the results as an initialization, a series or LP problems are solved until one of the eigenvalues of the interconnection becomes unstable.
- **second approach:** resolution of an optimization problem with a nonsmooth objective function and a non-convex constraint: $\min_{\Delta \in \mathbf{\Delta}} \overline{\sigma}(\Delta)$ such that $\lambda_{max}(A_0) = \max_i \Re\left(\lambda_i(A_0)\right) = 0$
- Matlab standard nonlinear optimization algorithms are used
- a relaxed condition is considered in practice to avoid convergence issues: the equality constraint is replaced with $|\lambda_{max}(A_0)| \leq \epsilon$

Survey of existing μ lower bound algorithms

4 Direct optimization-based techniques

- direct resolution of an optimization problem with a nonsmooth objective function and a non-convex constraint: $\min_{\Delta \in \mathbf{\Delta}} \overline{\sigma}(\Delta)$ such that $\det(I - M\Delta) = 0$
- a relaxed condition is considered in practice to avoid convergence issues: the equality constraint is replaced with $\underline{\sigma}(I - M\Delta) \leq \epsilon$ or $|\det(I - M\Delta)| \leq \epsilon$
- different optimization tools:
  1. Matlab standard nonlinear optimization algorithms (function `fmincon`)
  2. nonsmooth optimization algorithms
- convergence to a local minimum is ensured in the latter case

5 Geometrical approach

- the signs of the real and the imaginary parts of $\det(I - M\Delta)$ are computed for randomly selected points on the surface of a given hyperbox in $\mathbb{R}^N$
- this hyperbox is enlarged until the four possible sign combinations are found, which means that it might contain values of $\delta_1, \ldots, \delta_N$ s.t. $\det(I - M\Delta) = 0$
- a series of contractions and expansions involving nonlinear optimization are performed to approach the singular region

Selected μ lower bound algorithms

Eight μ lower bound algorithms are compared in this work:

	Description	Uncertainties	Matlab code
1	Power algorithm	all	Robust Control Toolbox
2	Gain-based algorithm	all but complex	Robust Control Toolbox
3	Poles migration technique	all	Carsten Döll
4	Poles migration technique	real	SMAC Toolbox
5	Poles migration technique	all	Mark Halton
6	Direct nonlinear optimization	all	Mark Halton
7	Direct nonsmooth optimization	all	Alberto Simoes
8	Geometrical approach	real	Jongrae Kim

Some other algorithms exist but are not considered here:

- exponential-time algorithms,
- methods which can only be applied in very specific cases,
- slight variants of already selected techniques.

Selected μ lower bound algorithms

Classical strategy: compute μ lower bounds on a fixed frequency grid:

- 50 logarithmically-spaced points within the system bandwidth
- 50 additional points used to refine the grid in some frequency regions corresponding to weakly damped modes

The same grid is used for all grid-based methods (1-2-6-7-8).

Alternative: compute bounds on a set of frequency intervals whose union covers the whole frequency range (requires to solve a skew-μ problem).

Interval-based implementations are available for methods 6 and 7 $\Rightarrow$ the system bandwidth is divided into 6 and 4 frequency intervals of equal size on a logarithmic scale respectively.

Special case of poles migration techniques (3-4-5): no tight grid is required since frequency is naturally optimized:

- coarse 10-point grid used as an initialization for method 4
- no frequency grid at all for methods 3 and 5

List of benchmarks

Large set of **36 challenging benchmarks**:

- a few academic systems and many real-world applications
- purely real, mixed real/complex or purely complex uncertainties
- large number of states (from 2 up to 70)
- large number of uncertainties (from 1 up to 28), repeated or not, even highly repeated in some cases (size of Δ up to 507×507)
- both rigid & highly flexible models (aircraft, telescope mockup, satellite...)
- several fields of application (civilian & fighter aircraft, launcher, re-entry vehicle, satellite, telescope, helicopter, spacecraft, missile, hard disk drive, biochemical network, car, hydraulic servo system, spark ignition engine...)

32 of these 36 benchmarks are available in the control literature. The other 4 ones have been developed by ONERA in cooperation with industrial partners. See the papers for all details and references!

Results for purely complex and mixed problems

Only algorithms 1-3-5-6-7 can be applied to benchmarks 30-36.

Algo	No. of times the gap w.r.t. the best μ lower bound is			Mean gap w.r.t. the best μ lower bound	Mean CPU time
	$=0\%$	$\leq 5\%$	$\leq 25\%$		
1	4	7	7	0.10%	1.1 s
3	0	4	6	16.60%	0.9 s
5	2	2	3	57.12%	140.8 s
6 (g)	1	7	7	0.34%	2648.8 s
6 (i)	0	5	7	4.26%	874.2 s
7 (g)	4	4	6	10.63%	3972.6 s
7 (i)	2	5	6	7.91%	249.5 s

The most relevant algorithm is the **power algorithm of Young & Doyle** [TAC, p.123-128, 1997] with the highest accuracy and almost the lowest computational time.

Improvement for purely complex and mixed problems

$\rightarrow$ The power algorithm is applied on a fixed frequency grid, which usually does not contain the frequency corresponding to the exact value of μ.

$\rightarrow$ It almost always converges in the presence of complex uncertainties, but it sometimes requires a quite large number of iterations.

$\rightarrow$ The considered optimization problem is non-concave and the results strongly depend on the initialization.

Idea: better exploit the power algorithm

1. algorithm 1 is applied on a rough frequency grid (*e.g.* 20 frequency points)
2. the grid is gradually tightened around the peak frequencies until improvement in the μ lower bound becomes marginal

At each frequency, algorithm 1 is not only initialized with the best result obtained at the previous frequency, but also with one or more random values.

Improvement for purely complex and mixed problems

This strategy has been implemented in the **Systems Modeling Analysis and Control (SMAC) Toolbox** for Matlab developed at ONERA.

Benchmark	Initial μ lower bound		Improved μ lower bound	
	value	time	value	time
30	1.6828	0.4 s	1.6828	0.4 s
31	3.7245	0.3 s	3.7245	0.2 s
32	0.8908	1.6 s	0.8908	2.0 s
33	0.4346	0.7 s	0.4362	0.6 s
34	0.9587	1.1 s	0.9606	1.3 s
35	0.9910	1.7 s	0.9927	2.1 s
36	15.0296	1.7 s	15.0296	1.1 s

The best lower bound is obtained for all benchmarks with this strategy.

Results for purely real problems

All algorithms are applied to benchmarks 1-29.

Algo	No. of times the gap w.r.t. the best μ lower bound is			Mean gap w.r.t. the best μ lower bound	Mean CPU time
	$=0\%$	$\leq 5\%$	$\leq 25\%$		
1	6	9	13	44.10%	1.7 s
2	2	19	24	11.87%	27.1 s
3	5	16	22	18.81%	1.0 s
4	26	27	29	0.88%	0.9 s
5	22	26	26	8.95%	22.8 s
6 (g)	5	18	24	11.09%	448.8 s
6 (i)	9	20	23	15.72%	97.8 s
7 (g)	8	17	23	17.44%	1694.9 s
7 (i)	24	25	25	9.36%	124.3 s
8	0	8	17	24.38%	749.4 s

The most relevant algorithm is the **poles migration technique of Ferreres & Biannic** [CEP, p.1267-1278, 2001] with by far the highest accuracy and also the lowest computational time.

Improvement for purely real problems

Improving the poles migration technique is not a trivial issue.

Idea: combine several algorithms

1. algorithm 4 is executed first (most efficient technique in almost all cases)
2. algorithm 2 is then executed only for a few selected frequencies using the previous results as an initialization
3. particle swarm optimization is finally applied using the previous results as an initialization

This strategy has been implemented in the **SMAC Toolbox**.

Bench	Algorithm 4		Other algorithms			Combination	
	value	time	best value	time	algo	value	time
20	0.9380	0.5 s	0.9947	41.6 s	7	0.9947	5.5 s
26	0.9881	2.2 s	1.2134	184.9 s	7	1.2144	18.0 s
29	724.15	46.9 s	733.86	2864.8 s	2	753.10	580.0 s

The best lower bound is obtained for all benchmarks with this strategy.

Outline

μ upper bound computation

Strategies exist to compute the best possible μ lower bound in all cases.

What about conservatism, which is measured as the gap with respect to the exact value of μ?

The next step is to **compute μ upper bounds** to evaluate this gap.

(D, G) scalings based characterization

Let $\beta > 0$. If there exist matrices $D \in \mathcal{D}$ and $G \in \mathcal{G}$ which satisfy one of the following relations:

$$M(j\omega)^* D M(j\omega) + j(GM(j\omega) - M(j\omega)^* G) \leq \beta^2 D$$

$$\overline{\sigma}\left((I + G^2)^{-\frac{1}{4}}\left(\frac{DM(j\omega)D^{-1}}{\beta} - jG\right)(I + G^2)^{-\frac{1}{4}}\right) \leq 1$$

where $\begin{cases} \mathcal{D} = \{D = D^* > 0 : \forall \Delta \in \mathbf{\Delta}, D\Delta = \Delta D\} \\ \mathcal{G} = \{G = G^* : \forall \Delta \in \mathbf{\Delta},\ G\Delta = \Delta^* G\} \end{cases}$, then $\mu_{\mathbf{\Delta}}(M(j\omega)) \leq \beta$.

μ upper bound computation

Computing a μ upper bound on $\mathbb{R}_+$ involves an infinite number of frequency-domain constraints $\Rightarrow$ usually solved on a finite frequency grid.

Problem

The frequency for which the maximal value of μ is reached is generally not part of the grid, since it is unknown $\Rightarrow$ k_r can be over-evaluated.

In this context, the frequency grid must be quite dense, which can lead to a prohibitive computational cost. But even so, it is still possible to miss a critical frequency...

Alternative method implemented in the SMAC Toolbox

- A μ upper bound is first determined at some frequency.
- A hamiltonian-based technique is then applied to determine all frequency intervals on which this bound remains valid.
- This strategy is repeated and a guaranteed μ upper bound is finally obtained when the union of all intervals covers the whole frequency range.

Gap between μ lower and upper bounds

A μ upper bound is computed for each of the 36 benchmarks using the SMAC Toolbox for Matlab using the aforementioned algorithm.

The mean value of the gap between the lower and the upper bounds $\check{\mu}$ and $\hat{\mu}$ is:

- **12.71%** for purely real problems
- **0.39%** for purely complex and mixed real/complex problems

Is it the lower or the upper bound on μ which is responsible for this gap?

Claim

The μ lower bound almost always equals the exact value of μ.

No proof, but true in many practical cases!

Focus on purely complex and mixed problems

For 6 benchmarks out of 7, the gap between $\check{\mu}$ and $\hat{\mu}$ is less than 0.00%.

For the last one, a branch & bound algorithm is applied to improve $\hat{\mu}$.

		Without branch & bound		With branch & bound	
Bench	**Value of $\check{\mu}$**	**Value of $\hat{\mu}$**	**Gap**	**Value of $\hat{\mu}$**	**Gap**
33	0.4346	0.4479	2.68%	0.4346	0.00%

The mean value of the gap over all 7 benchmarks is now **less than 0.00%**.

For all benchmarks, it has been proved that the μ lower bound is equal to the exact value of μ.

Focus on purely real problems

For 19 benchmarks out of 29, the gap between $\check{\mu}$ and $\hat{\mu}$ is less than 0.00%.
For the 10 others, a branch & bound algorithm is applied to improve $\hat{\mu}$.

		Without branch & bound		With branch & bound	
Bench	**Value of $\check{\mu}$**	**Value of $\hat{\mu}$**	**Gap**	**Value of $\hat{\mu}$**	**Gap**
3	1.7111	2.4131	41.03%	1.7111	0.00%
10	1.6200	1.9757	21.96%	1.6200	0.00%
16	0.8182	0.8187	0.06%	0.8182	0.00%
17	0.6667	0.6875	3.12%	0.6667	0.00%
19	0.9966	1.0875	9.12%	0.9966	0.00%
20	0.9947	1.0073	1.27%	0.9947	0.00%
22	0.1842	0.2283	23.94%	0.1842	0.00%
23	1.6668	2.2237	33.41%	1.7000	1.99%
27	0.8619	2.7235	215.99%	0.9481	10.00%
29	753.10	894.36	18.76%	754.00	0.12%

The mean value of the gap over all 29 benchmarks is now only **0.42%**.

For all except one benchmarks, it has been proved that the μ lower bound is (almost) equal to the exact value of μ.

Problem: computing tight μ upper bounds can be extremely long.

Outline

Towards a reduction of conservatism

The gap between μ upper and lower bounds is usually very reasonable, but it can unfortunately be quite large in some cases.

This is essentially due to the conservatism of the μ upper bound. From experience, the μ lower bound is usually very close to the true value of μ.

Several solutions can be considered to reduce conservatism and CPU time:

- branch & bound $\rightarrow$ already implemented in the **SMAC Toolbox**,
- μ-sensitivities to focus on the most relevant uncertainties,
- less conservative μ upper bound characterizations,
- faster LMI solvers (SeDuMi...).

} available in a future release

This work focuses on the first three items for problems with purely real uncertainties.

Branch-and-bound algorithms

Standard branch & bound algorithm

Partition the uncertainty domain in more and more subsets until the gap between:

- the highest lower bound computed on all subsets and
- the highest upper bound computed on all subsets

becomes lower than a user-defined threshold η.

$\rightarrow$ Conservatism can be reduced to an arbitrarily small value for systems with only real parametric uncertainties.

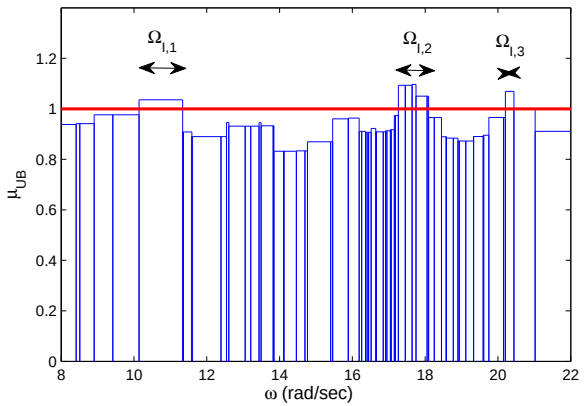
$\rightarrow$ Computational complexity grows exponentially as a function of the number of uncertain parameters (it can take hours and even days to get a tight μ upper bound for some benchmarks).

Branch-and-bound algorithms

Improved branch & bound algorithm

Assume that for a given subset $\mathcal{D}_i$, stability can only be proved for frequencies $\omega \in \Omega_V \subset \mathbb{R}_+$:

- $\rightarrow$ $\mathcal{D}_i$ is partitioned,
- $\rightarrow$ for each subset of $\mathcal{D}_i$, the analysis is restricted to the frequency domain $\Omega_I = \mathbb{R}_+ \setminus \Omega_V$.



After a few steps, each analysis is restricted to narrow frequency intervals.

$\Rightarrow$ drastic reduction of computational cost, but still not sufficient...

Use of the μ-sensitivities

The μ-sensitivities introduced by Braatz & Morari [ACC, p.231-236, 1991] aim at determining how the value of μ changes with a small variation of a single uncertainty in Δ.

μ-sensitivities

Let $M = M(j\omega)$ be the frequency response of $M(s)$ at a frequency ω. Let $\Delta = \text{diag}(\delta_1 I_{n_1}, \ldots, \delta_N I_{n_N})$ and $A(\epsilon_j) = \text{diag}(I_{n_1}, \ldots, (1-\epsilon_j) I_{n_j}, \ldots, I_{n_N})$. The jth μ-sensitivity is defined as:

$$S_j^\mu = \frac{\partial}{\partial \epsilon_j} \mu_{\mathbf{\Delta}}(M_{\epsilon_j}) = \lim_{\epsilon_j \to 0^+} \frac{|\mu_{\mathbf{\Delta}}(M) - \mu_{\mathbf{\Delta}}(M_{\epsilon_j})|}{\epsilon_j}$$

where $M_{\epsilon_j} = A(\epsilon_j) M A(\epsilon_j)$.

The μ-sensitivities can thus be used to identify which uncertainties have the largest influence on the μ upper bound.

Use of the μ-sensitivities

1 Application to the (D, G) scalings based characterization

The decision variables associated to the most "critical" uncertainties are optimized using an LMI solver, while the others are obtained using a suboptimal and faster gradient descent method.

$$\Delta = \text{diag}(\delta_1 I_{n_1}, ..., \delta_N I_{n_N}) \Rightarrow \begin{cases} \mathcal{D} = \{\text{diag}(D_1, ..., D_N), D_i \in \mathbb{C}^{n_i \times n_i}, D_i = D_i^*\} \\ \mathcal{G} = \{\text{diag}(G_1, ..., G_N), G_i \in \mathbb{C}^{n_i \times n_i}, G_i = G_i^*\} \end{cases}$$

$\rightarrow$ only some of the D_i and G_i matrices are optimized using an LMI solver

Bench	$\hat{\mu}^{LMI}$	$\hat{\mu}^{pLMI}$		$\hat{\mu}^{GD}$	
	CPU time	**Gap w.r.t $\hat{\mu}^{LMI}$**	**CPU time**	**Gap w.r.t $\hat{\mu}^{LMI}$**	**CPU time**
25	63 s	0.01%	20 s	20.41%	9 s
26	306 s	0.57%	30 s	20.47%	6 s
27	3546 s	26.58%	100 s	63.37%	36 s
satellite	2555 s	97.34%	49 s	230.48%	22 s

μ upper bounds close to $\hat{\mu}^{LMI}$ are obtained with a computational time close to the one needed to compute $\hat{\mu}^{GD}$.

Use of the μ sensitivities

2 Application to the multiplier based characterization

Focus on purely real uncertainties: $\Delta = \text{diag}(\delta_1 I_{n_1}, \ldots, \delta_N I_{n_N})$.

Multiplier based characterization

Let $\alpha > 0$. If there exist a matrix $C \in \mathbb{C}^{n \times n}$, where $n = \sum_{i=1}^{N} n_i$, such that:

$$He\ C(I_n - \Delta M(j\omega)) < 0 \quad \text{for all} \quad \Delta = \frac{1}{\alpha}\text{diag}(\pm I_{n_1}, \ldots, \pm I_{n_N})$$

then $\mu_{\mathbf{\Delta}}(M(j\omega)) \leq \alpha$.

+ less conservative than the (D, G) scalings based characterization,
- unstructured multiplier $C \Rightarrow$ large number of decision variables,
- 2^N constraints $\Rightarrow$ exponential growth of computational complexity,
- only applicable at a single frequency $\Rightarrow$ use of a frequency grid.

Use of the μ-sensitivities

Proposed solutions:

- exploit rank deficiency: if $M = AB^*$, where $A, B \in \mathbb{C}^{n \times q}$ and $q = \text{rank}(M)$, the size of C can be restricted to $q \times q$ instead of $n \times n$,
- use the μ-sensitivities: if the r most "critical" uncertainties are handled using the multiplier approach and the others using the (D, G) scalings approach, the number of constraints drops from 2^N to 2^r,
- use a hamiltonian-based technique to compute μ upper bounds which are guaranteed on the whole frequency range and not only on a grid.

Benchmark	(D, G) **scalings**		**Multiplier**		
	Gap w.r.t. $\check{\mu}$	**CPU time**	r	**Gap w.r.t.** $\check{\mu}$	**CPU time**
3	41.03%	1 s	2	1.23%	42 s
10	27.59%	6 s	2	14.20%	120 s
			4	0.09%	477 s
22	24.46%	11 s	5	12.88%	133 s
			9	5.34%	1190 s

Conservatism is significantly reduced with a reasonable computation time.

Use of the μ-sensitivities

3 Application to branch & bound

The uncertainty domain is cut along the most "sensitive" edges instead of the longest edges.

Benchmark	Gap η	Standard B&B	Use of the μ-sensitivities
		CPU time	CPU time
19	2%	1787 s	104 s
22	5%	348 s	180 s
23	5%	688 s	481 s
27	10%	∞	901 s
	85%	280 s	10 s
29	10%	36269 s	1485 s

Conservatism is significantly reduced with a much more reasonable computational time.

Conclusion

A thorough comparative analysis of the main existing methods to compute μ lower bounds has been presented. The most relevant algorithms are:

- the **poles migration technique** of Ferreres & Biannic [CEP, p.1267-1278, 2001] for purely real problems
- the **power algorithm** of Young & Doyle [TAC, p.123-128, 1997] for purely complex or mixed real/complex problems

36 challenging benchmarks have been considered, with various fields of application, system dimensions and structures of the uncertainties.

A strategy has been proposed to combine the most efficient techniques. In almost all cases, the μ lower bound is equal to the exact value of μ.

Several solutions have been proposed to reduce conservatism and computational time when computing μ upper bounds: **branch & bound**, **μ-sensitivities**, **alternative μ upper bound characterizations**...

Similar results are obtained for skew-μ problems.

Overview of the SMART Library of the SMAC Toolbox

Matlab routines to compute tight bounds on μ are available in the **Skew-Mu Analysis based Robustness Tools (SMART) Library** of the **SMAC Toolbox**:

http://w3.onera.fr/smac

More generally, the SMART Library implements most of the μ-analysis based algorithms developed at ONERA/DCSD and allows to compute:

- the (skewed) structured singular value,
- the (skewed) robust stability margin,
- the worst-case H_∞ performance level,
- the worst-case gain, modulus, phase and delay margins.

It can be applied to high-order systems with numerous uncertainties.

It is currently the most efficient available software.

A new release of the **SMART Library** will be issued in 2015!

List of benchmarks

Benchmark	Description	States	Uncertainty block Δ	
			Size	Structure
1	Academic example	5	1	1×1
2	Academic example	4	3	3×1
3	Academic example	4	4	2×2
4	Inverted pendulum	4	3	3×1
5	Anti-aliasing filter	2	5	$3\times1 + 1\times2$
6	DC motor	4	5	$3\times1 + 1\times2$
7	Bus steering system	9	5	$1\times2 + 1\times3$
8	Satellite	9	4	$2\times1 + 1\times2$
9	Bank-to-turn missile	6	4	4×1
10	Aeronautical vehicle	8	4	4×1
11	Four-tank system	10	4	4×1
12	Re-entry vehicle	6	8	$1\times2 + 2\times3$
13	Missile	14	4	4×1
14	Cassini spacecraft	17	4	4×1
15	Mass-spring-damper	7	6	6×1
16	Spark ignition engine	4	7	7×1
17	Hydraulic servo system	8	8	8×1
18	Academic example	41	5	$3\times1 + 1\times2$
19	Drive-by-wire vehicle	4	16	$2\times1 + 7\times2$
20	Re-entry vehicle	7	13	$3\times1 + 1\times4 + 1\times6$
21	Space shuttle	34	9	9×1
22	Rigid aircraft	9	14	14×1

List of benchmarks

Benchmark	Description	States	Uncertainty block Δ	
			Size	Structure
23	Fighter aircraft	10	27	$7\times1 + 1\times2 + 1\times3 + 1\times15$
24	Flexible aircraft	46	20	20×1
25	Telescope mockup	70	20	20×1
26	Hard disk drive	29	27	$19\times1 + 4\times2$
27	Launcher	30	45	$16\times1 + 10\times2 + 1\times3 + 1\times6$
28	Helicopter	12	120	4×30
29	Biochemical network	7	507	13×39
30	Himat fighter aircraft	16	4	$2\times2(c)$
31	F14 fighter aircraft	52	8	$1\times2(c) + 1\times6(c)$
32	DC motor	4	6	$3\times1 + 1\times2 + 1\times1(c)$
33	Four-tank system	12	6	$4\times1 + 1\times2(c)$
34	Missile	19	6	$4\times1 + 2\times1(c)$
35	Hydraulic servo system	9	9	$8\times1 + 1\times1(c)$
36	Space shuttle	46	18	$9\times1 + 1\times9(c)$

The notation $m\times p$ in the last column means that Δ contains m blocks of size $p\times p$.
All blocks are real unless (c) is specified. All real/complex blocks are diagonal/full.

All benchmarks can be freely downloaded from the **SMAC website**:

http://w3.onera.fr/smac/smart

References

μ lower bound computation

[1] A. Fabrizi, C. Roos and J-M. Biannic, "A detailed comparative analysis of μ lower bound algorithms", in Proceedings of the ECC, Strasbourg, France, 2014, pp. 220-226.

[2] C. Roos and J-M. Biannic, "A detailed comparative analysis of all practical algorithms to compute lower bounds on the structured singular value", submitted to Control Engineering Pratice, 2015.

μ upper bound computation

[3] C. Roos and J-M. Biannic. "Efficient computation of a guaranteed stability domain for a high-order parameter dependent plant", in Proceedings of the ACC, Baltimore, Maryland, 2010, pp. 3895-3900.

[4] J. Lesprier, C. Roos and J-M. Biannic, "Improved μ upper bound computation using the μ-sensitivities", in Proceedings of the IFAC ROCOND, Bratislava, Slovak Republic, 2015.

[5] J. Lesprier, C. Roos and J-M. Biannic, "Efficient computation of the multiplier based μ upper bound on large frequency intervals", submitted to the IEEE CDC, Osaka, Japan, 2015.

SMART Library of the SMAC Toolbox

[6] C. Roos, "Systems Modeling, Analysis and Control Toolbox: an insight into the robustness analysis library", in Proceedings of the IEEE MSC, Hyderabad, India, 2013, pp. 176-181.

[7] C. Roos, F. Lescher, J-M. Biannic, C. Doll and G. Ferreres, "A set of μ-analysis based tools to evaluate the robustness properties of high-dimensional uncertain systems", in Proceedings of the IEEE MSC, Denver, Colorado, 2011, pp. 644-649.