Computing the structured distance to instability

Pierre Apkarian* Dominikus Noll† Laleh Ravanbod†

Abstract

We analyze robust stability and performance of dynamical systems with real uncertain parameters. We compute criteria like the distance to instability, the worst-case spectral abscissa, or the worst-case H_∞-norm, which quantify the degree of robustness of such a system when parameters vary in a given set $\mathbf{\Delta}$. As computing these indices is NP-hard, we present a heuristic which finds good lower bounds fast and reliably. Posterior certification is then obtained by an intelligent global strategy. A test bench of 87 systems with up to 70 states 39 uncertain parameters with up to 11 repetitions demonstrates the potential of our approach.

1 Problem Specification.

Robustness specifications limit the loss of performance and stability in a system where differences between the mathematical model and reality crop up. Robustness against real parametric uncertainties is among the most challenging calls in this regard. Already deciding whether a given system with uncertain real parameters δ is robustly stable over a given parameter range $\delta \in \mathbf{\Delta}$ is NP-hard, and this is aggravated when it comes to deciding whether a given level of H_2- or H_∞-performance is guaranteed over that range. In this work we address this type of uncertainty by computing three key criteria, which quantify the degree of parametric robustness of a system. These are (a) the worst-case H_∞-norm, and (b) the worst-case spectral abscissa over a given parameter range, and (c) the distance to instability, or stability margin, of a system with uncertain parameters.

Consider a Linear Fractional Transform [23] with real parametric uncertainties as in Figure 1, where

$$(1.1) P(s): \left\{ \begin{array}{rcl} \dot{x} &=& Ax + B_p p + B_w w \\ q &=& C_q x + D_{qp} p + D_{qw} w \\ z &=& C_z x + D_{zp} p + D_{zw} w \end{array} \right.$$

and $x \in \mathbb{R}^n$ is the state, $w \in \mathbb{R}^{m_1}$ a vector of exogenous inputs, and $z \in \mathbb{R}^{p_1}$ a vector of regulated outputs. The uncertainty channel is defined as $p = \Delta q$, where the time-invariant uncertain matrix Δ has the block-diagonal form

$$(1.2) \qquad \Delta = \mathrm{diag}\left[\delta_1 I_{r_1}, \ldots, \delta_m I_{r_m}\right],$$

with $\delta_1, \ldots, \delta_m$ representing real uncertain parameters, and r_i giving the number of repetitions of δ_i. Here we assume without loss that $\delta = 0 \in \mathbf{\Delta}$ represents the nominal parameter value, and we consider $\delta \in \mathbf{\Delta}$ in one-to-one correspondence with the matrix Δ in (1.2). For practical applications it is generally sufficient to consider the case $\mathbf{\Delta} = [-1, 1]^m$.

To analyze the performance of (1.1) in the presence of the uncertain $\delta \in \mathbb{R}^m$ we compute the worst-case H_∞-performance

$$(1.3) \qquad h^* = \max\{\|T_{wz}(\delta)\|_\infty : \delta \in \mathbf{\Delta}\},$$

where $\|\cdot\|_\infty$ is the H_∞-norm, and where $T_{wz}(s, \delta)$ is the transfer function $z(s) = \mathcal{F}_u(P(s), \Delta)w(s)$, obtained by closing the loop between (1.1) and $p = \Delta q$ with (1.2) in Figure 1. The solution $\delta^* \in \mathbf{\Delta}$ of (1.3) represents a worst possible choice of the parameters $\delta \in \mathbf{\Delta}$, which may be an important element in analyzing performance and robustness of the system, see e.g. [1].

Our second criterion is similar in nature, as it allows to verify whether the uncertain system (1.1) is robustly stable over a given parameter range $\mathbf{\Delta}$. This can be tested by maximizing the spectral abscissa of the system A-matrix over the parameter range

$$(1.4) \qquad \alpha^* = \max\{\alpha(A(\delta)) : \delta \in \mathbf{\Delta}\},$$

where $A(\delta) = A + B_p \Delta (I - D_{pq}\Delta)^{-1} C_q$, and where the spectral abscissa of a square matrix A is defined as $\alpha(A) = \max\{\mathrm{Re}\,\lambda : \lambda \text{ eigenvalue of } A\}$. Since A is stable if and only if $\alpha(A) < 0$, robust stability of (1.1) over $\mathbf{\Delta}$ is certified as soon as $\alpha^* < 0$, while a destabilizing $\delta^* \in \mathbf{\Delta}$ is found as soon as $\alpha^* \geq 0$.

Note however that a decision in favor of robust stability over $\mathbf{\Delta}$ based on $\alpha^* < 0$ is only valid when the global maximum over $\mathbf{\Delta}$ is computed. This renders (1.4) a difficult problem, and it is in fact known that solving (1.4) globally is NP-hard. In [18] Poljak and Rohn have shown that for a given set of matrices $A_0, \ldots, A_k$, deciding whether $A_0 + r_1 A_1 + \ldots + r_k A_k$ is stable for all $r_i \in [0, 1]$ is NP-hard, and Braatz *et al.* [6] have shown that deciding whether a system with real (or mixed or complex) uncertainties is robustly stable over a range $\mathbf{\Delta} = [-1, 1]^m$ is harder than globally solving a nonconvex quadratic programming problem, hence is NP-hard.

*ONERA, Control System Department, Toulouse, France
†Université de Toulouse, Institut de Mathématiques

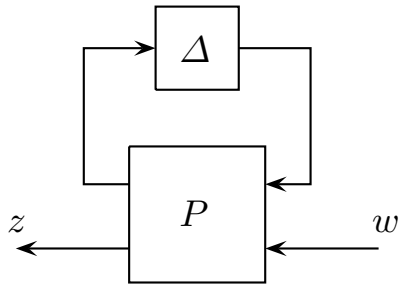

Figure 1: Robust system interconnection

Our third analysis problem is related to the previous ones and concerns computation of the distance to instability. Assuming $A(\delta)$ stable at the nominal value $\delta = 0$, we ask for the largest variation in the parameter δ under which the system remains stable. This leads to

$$(1.5)\quad d^* = \max\{d : A(\delta) \text{ stable for all } |\delta|_\infty < d\},$$

where $|\delta|_\infty = \max\{|\delta_1|, \ldots, |\delta_m|\}$ is the maximum norm. This quantity is also known as the stability margin, or the radius of stability of (1.1). A formulation of (1.5) which does not require $A(0)$ to be stable is

$$d^* = \min\{|\delta|_\infty : A(\delta) \text{ unstable}\},$$

and this still works when $A(\delta)$ is stable for all δ, because then $d^* = \min \emptyset = +\infty$. This version can be given the form of a constrained optimization program

$$(1.6)\quad \begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -t \le \delta_i \le t, i = 1, \ldots, m \\ & -\alpha(A(\delta)) \le 0 \end{array}$$

with decision variable $(t, \delta) \in \mathbb{R}^{1+m}$.

2 Methods

The fact that the analysis problems (1.3) – (1.5) are NP-hard has practical consequences. It means that it is vain to address them frontally, and that heuristic methods and tailored approaches are required. Once the importance of these problems had been recognized in the 1980s, a number of heuristic attempts were made, altogether with limited success. The most principled among them are based on the structured singular value $\mu_{\mathbf{\Delta}}$, introduced in [10], and use overestimations. Since P is certified robust over $\mathbf{\Delta} = [-1, 1]^m$ as soon as $\mu_{\mathbf{\Delta}}(P) < 1$, it is near at hand to construct outer approximations $\widetilde{\mu}_{\mathbf{\Delta}} \ge \mu_{\mathbf{\Delta}}$ which are easier to compute, and then seek a certificate by showing $\widetilde{\mu}_{\mathbf{\Delta}}(P) < 1$. If only $\widetilde{\mu}_{\mathbf{\Delta}}(P) = 1 + \epsilon$ can be proved, one has at least a certificate over the box $(1+\epsilon)^{-1}\mathbf{\Delta}$. The approximation [19] is a prominent example in this class.

In terms of program (1.4), outer approximations work similarly. Find an overestimation $\alpha^* \le \overline{\alpha}$ which is easier to compute, and try to prove $\overline{\alpha} < 0$, as this entails the desired certificate $\alpha^* < 0$. Examples of such $\overline{\alpha}$ are Trefethen's pseudo-spectral abscissa [21], or the approximation proposed by Hinrichsen and Pritchard [14]. LMI-relaxations of the robust analysis problem are also in this class, see [4].

In this work we prefer inner approximations. In a first stage this means that we compute good lower bounds for h^*, α^*, d^* using a local optimization algorithm. These bounds are then in a second stage used within, or in tandem with, global optimization techniques to get certificates. There is general agreement that inner approximations give better practical results, but it is often held against them that they are less rigorous, the argument being that they give no immediate certificates. This argument is not tenable, because there is no guarantee either in an outer approximation technique that it succeeds in computing a certificate. And even when an outer approximation method obtains a certificate, it is usually conservative, so that an inner approximation technique, had it to deal with such a sub-optimal value only, would not have any particular difficulty certifying it. We hold that inner approximations are stronger in both respects. They give better heuristic results, and they present also the theoretically preferable way to certify them.

3 Optimization program

In this section we discuss a general optimization program of the form

$$(3.7)\quad \begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in C \end{array}$$

where C is a convex set with a simple structure, and where $f : \mathbb{R}^m \to \mathbb{R}$ is a nonsmooth and nonconvex objective, which in this work has the specific property of being representable as a semi-infinite minimum of smooth functions. As we shall see, this covers programs (1.3) – (1.5), and we show how this can be exploited algorithmically. To understand the structure of programs (1.3) – (1.5), we recall the following

DEFINITION 1. **(Spingarn [20])**. *A locally Lipschitz function $f : \mathbb{R}^m \to \mathbb{R}$ is lower-C^1 at $\boldsymbol{x}_0 \in \mathbb{R}^m$ if there exists a neighborhood U of $\boldsymbol{x}_0$, a compact Hausdorff space $\mathbb{K}$, and a function $F : \mathbb{R}^m \times \mathbb{K} \to \mathbb{R}$ such that for every $\boldsymbol{x} \in U$,*

$$(3.8)\qquad f(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \mathbb{K}} F(\boldsymbol{x}, \boldsymbol{y})$$

and F and $\partial F/\partial \boldsymbol{x}$ are jointly continuous. The function f is upper-C^1 at $\boldsymbol{x}_0$ if $-f$ is lower-C^1 at $\boldsymbol{x}_0$. □

Lower- and upper-C^1 functions behave very differently when minimized. In a lower-C^1 function the nonsmoothness goes downward, so in descending we are moving into the zone of nonsmoothness, where we expect difficulties. In contrast, an upper-C^1 function has its nonsmoothness going upward. Therefore on minimizing it we move away from the zone of nonsmoothness, which is why we expect less trouble.

Minimizing lower-C^1 functions is of min-max type. This form of nonsmoothness is preponderant in nonsmooth optimization and therefore well-studied. A typical case in control is H_∞-synthesis [2], where the semi-infinite character of the objective is at the origin of the difficulty. To cope with min-max problems one has to minimize active or near active branches simultaneously.

In contrast, minimizing upper-C^1 functions is of min-min type. This highlights the difficulty, as it is disjunctive and may run into a combinatorial explosion. Yet min-min problems have some favorable features when local minima are computed. Namely, as opposed to the min-max case, when several branches are active at $\boldsymbol{x}$ in a min-min problem, then it is enough to select one and proceed. This is what we exploit favorably.

4 Algorithm

In order to solve (3.7) algorithmically, we propose a nonsmooth trust-region strategy, which we now explain. At the current iterate $\boldsymbol{x}$ consider the standard model

$$\phi^\sharp(\boldsymbol{y},\boldsymbol{x}) = f(\boldsymbol{x}) + f^\circ(\boldsymbol{x},\boldsymbol{y}-\boldsymbol{x}) \tag{4.9}$$

of f at $\boldsymbol{x}$, where $f^\circ(\boldsymbol{x},\boldsymbol{d})$ is the Clarke directional derivative of f at $\boldsymbol{x}$ in direction $\boldsymbol{d}$; cf. [8]. We may regard (4.9) a natural nonsmooth analogue of the first-order Taylor expansion at $\boldsymbol{x}$.

Recall that $f^\circ(\boldsymbol{x},\boldsymbol{d}) = \max\{\boldsymbol{g}^\top\boldsymbol{d} : \boldsymbol{g} \in \partial f(\boldsymbol{x})\}$, where $\partial f(\boldsymbol{x})$ is the Clarke subdifferential of f at $\boldsymbol{x}$. We generate finite approximations $\phi_k^\sharp$ of $\phi^\sharp$ by selecting a finite subset $\mathcal{G}_k$ of $\partial f(\boldsymbol{x})$, and putting

$$\phi_k^\sharp(\boldsymbol{y},\boldsymbol{x}) = f(\boldsymbol{x}) + \max\{\boldsymbol{g}^\top(\boldsymbol{y}-\boldsymbol{x}) : \boldsymbol{g} \in \mathcal{G}_k\}, \tag{4.10}$$

where k is the counter of the inner loop of our algorithm. Note that $\phi_k^\sharp(\boldsymbol{x},\boldsymbol{x}) = \phi^\sharp(\boldsymbol{x},\boldsymbol{x})$, $\phi_k^\sharp \le \phi^\sharp$, and $\partial_1\phi_k^\sharp(\boldsymbol{x},\boldsymbol{x}) \subset \partial_1\phi^\sharp(\boldsymbol{x},\boldsymbol{x})$.

At serious iterate $\boldsymbol{x}$ and inner loop counter k, current trust-region radius R_k, and current working model $\phi_k^\sharp$, we solve the trust-region tangent program

$$\begin{array}{lll} & \text{minimize} & \phi_k(\boldsymbol{y},\boldsymbol{x}) \\ (4.11) & \text{subject to} & \boldsymbol{y} \in C \\ & & \|\boldsymbol{y}-\boldsymbol{x}\| \le R_k \end{array}$$

Let $\boldsymbol{y}^k$ be an optimal solution of (4.11). Fixing $0 < \theta < 1$, we allow as a *trial step* any $\boldsymbol{z}^k \in B(\boldsymbol{x},R_k)$ satisfying

$$f(\boldsymbol{x}) - \phi_k^\sharp(\boldsymbol{z}^k,\boldsymbol{x}) \ge \theta\left(f(\boldsymbol{x}) - \phi_k^\sharp(\boldsymbol{y}^k,\boldsymbol{x})\right). \tag{4.12}$$

Trial steps $\boldsymbol{z}^k$ are candidates to become the next serious step $\boldsymbol{x}^+$, and this is tested by the usual condition

$$\rho_k = \frac{f(\boldsymbol{x}) - f(\boldsymbol{z}^k)}{f(\boldsymbol{x}) - \phi_k(\boldsymbol{z}^k,\boldsymbol{x})} \stackrel{?}{\ge} \gamma$$

where $0 < \gamma < 1$. In the case of acceptance $\rho_k \ge \gamma$ we call $\boldsymbol{x}^+ = \boldsymbol{z}^k$ a *serious step*. The crucial question is what action to take when $\rho_k < \gamma$, in which case $\boldsymbol{z}^k$ is not accepted and termed a *null step*. Here we have to form a better model $\phi_{k+1}^\sharp(\cdot,\boldsymbol{x})$, and we may have to reduce the trust-region radius. The details are covered by the following trust-region algorithm:

Algorithm 1. Trust-region method for program (3.7)

Parameters: $0 < \gamma < \widetilde{\gamma} < 1$, $0 < \gamma < \Gamma$, $0 < \theta \ll 1$.

▷ **Step 1** (`Initialize`)**.** Choose $\boldsymbol{x}^1 \in C$ and initial $R_1^\sharp > 0$. Put outer loop counter $j = 1$.

▷ **Step 2** (`Stopping`)**.** Stop if outer loop iterate $\boldsymbol{x}^j$ at counter j is critical. Otherwise goto inner loop.

▷ **Step 3** (`Initialize inner loop`)**.** Put $k = 1$. Initialize $\phi_1^\sharp(\cdot,\boldsymbol{x}^j) = f(\boldsymbol{x}^j) + \boldsymbol{g}_j^\top(\cdot - \boldsymbol{x}^j)$, where $\boldsymbol{g}_j \in \partial f(\boldsymbol{x}^j)$. Initialize trust-region radius $R_1 = R_j^\sharp$.

▷ **Step 4** (`Trial step`)**.** Given $\phi_k^\sharp$ and R_k, compute solution $\boldsymbol{y}^k$ of (4.11). Find trial step $\boldsymbol{z}^k$ with (4.12).

▷ **Step 5** (`Acceptance`)**.** If

$$\rho_k = \frac{f(\boldsymbol{x}^j) - f(\boldsymbol{z}^k)}{f(\boldsymbol{x}^j) - \phi_k(\boldsymbol{z}^k,\boldsymbol{x}^j)} \ge \gamma$$

put $\boldsymbol{x}^{j+1} = \boldsymbol{z}^k$ (serious step) and go to step 7. Otherwise find $\boldsymbol{g}_k \in \partial f(\boldsymbol{x}^j)$ with $f^\circ(\boldsymbol{x}^j, \boldsymbol{z}^k - \boldsymbol{x}^j) = \boldsymbol{g}_k^\top(\boldsymbol{z}^k - \boldsymbol{x}^j)$, and add it to $\mathcal{G}_{k+1}$. Build new $\phi_{k+1}^\sharp$.

▷ **Step 6** (`Update trust-region radius`)**.** If

$$\widetilde{\rho}_k = \frac{f(\boldsymbol{x}^j) - \phi_{k+1}^\sharp(\boldsymbol{z}^k,\boldsymbol{x}^j)}{f(\boldsymbol{x}^j) - \phi_k^\sharp(\boldsymbol{z}^k,\boldsymbol{x}^j)} \ge \widetilde{\gamma}$$

put $R_{k+1} = \frac{1}{2}R_k$, else $R_{k+1} = R_k$. Increase k and goto step 4.

▷ **Step 7** (`Update memory trust-region radius`)**.**

$$R_j^\sharp = \begin{cases} R_k & \text{if } \rho_k < \Gamma \\ 2R_k & \text{if } \rho_k \ge \Gamma \end{cases}$$

Increase outer loop counter j and go back to step 2.

It is crucial to observe that as soon as $\boldsymbol{x}^j$ is a point of strict differentiability of f, then $\partial f(\boldsymbol{x}^j) = \{\nabla f(\boldsymbol{x}^j)\}$, hence $\phi^\sharp(\cdot, \boldsymbol{x}^j) = \phi_k^\sharp(\cdot, \boldsymbol{x}^j) = f(\boldsymbol{x}^j) + \nabla f(\boldsymbol{x}^j)^\top(\cdot - \boldsymbol{x}^j)$ for all k. Then algorithm 1 reduces to the standard smooth (first-order) trust-region method, where the only action in case of a null step is reduction of the trust-region radius. Namely, subgradients $\boldsymbol{g}_j$ in step 3 and $\boldsymbol{g}_k$ in step 5 then all coincide with $\nabla f(\boldsymbol{x}^j)$, and the test quotient $\widetilde{\rho}_k$ equals 1. Therefore if f is strictly differentiable almost everywhere on C, and if $\boldsymbol{z}^k$ is drawn at random according to a continuous probability law on C, then algorithm 1 is the classical trust-region scheme with probability 1. This is what happens in practice and explains why our method works fast and reliably.

5 Convergence

The convergence properties of algorithm 1 will be considered next. We recall a concept which was first used in [17] to prove convergence of a bundle method. We specialize it here to the standard model. We say that the standard model $\phi^\sharp$ of f is strict at $\boldsymbol{x}$ if for any sequences $\boldsymbol{x}_k, \boldsymbol{y}_k \to \boldsymbol{x}$ there exists $\epsilon_k \to 0^+$ such that $f(\boldsymbol{y}^k) \leq f(\boldsymbol{x}^k) + \phi^\sharp(\boldsymbol{y}^k, \boldsymbol{x}^k) + \epsilon_k \|\boldsymbol{y}^k - \boldsymbol{x}^k\|$. The following was proved in [16]:

LEMMA 5.1. *Suppose f is upper-C^1 at $\boldsymbol{x}$. Then the standard model $\phi^\sharp$ is strict at $\boldsymbol{x}$.* □

We have the following

THEOREM 5.1. *Suppose the standard model $\phi^\sharp$ of f is strict. Let $\boldsymbol{x}^1 \in C$ be such that $\{\boldsymbol{x} \in C : f(\boldsymbol{x}) \leq f(\boldsymbol{x}^1)\}$ is bounded. Then every accumulation point $\boldsymbol{x}^*$ of the $\boldsymbol{x}^j$ generated by algorithm 1 is a critical point of (3.7).* □

The proof is outside the scope of the present contribution and can be found in [3]. By an observation first made in [9], it is possible to say more when f is upper-C^1. Namely, we may then fix $\boldsymbol{g}_j \in \partial f(\boldsymbol{x}^j)$ drawn in step 3 of the algorithm, letting $\boldsymbol{g}_k = \boldsymbol{g}_j$ for the $\boldsymbol{g}_k$ in step 5. Then, regardless whether or not $\boldsymbol{x}^j$ is a point of strict differentiability of f, the algorithm reduces to its classical first-order antecedent.

THEOREM 5.2. *Let f be upper-C^1 and suppose $\{\boldsymbol{x} \in C : f(\boldsymbol{x}) \leq f(\boldsymbol{x}^1)\}$ is bounded. Let $\boldsymbol{x}^j \in C$ be generated by the classical first-order trust-region algorithm. Then every accumulation point $\boldsymbol{x}^*$ of the $\boldsymbol{x}^j$ is critical.* □

6 Amenability

We still have to show that the criteria used in programs (1.3) – (1.5) are amenable to the theory presented in section 5. We start with the following

LEMMA 6.1. *Let f be defined as $f(\delta) = -\|T_{wz}(\delta)\|_\infty$, then f is upper-C^1 on the set $D = \{\delta \in \mathbb{R}^m : T_{wz}(\delta)$ is internally stable$\}$.*

Proof. We use the variational representation of the maximum singular value

$$\overline{\sigma}(G) = \sup_{\|u\|=1} \sup_{\|v\|=1} \left| u^\top G v \right|.$$

Observing that $z \to |z|$ is convex and therefore lower-C^1, we write it in the form $|z| = \sup_{l \in \mathbb{L}} \Psi(z, l)$ for Ψ and $\partial\Psi/\partial z$ jointly continuous and a suitable compact set $\mathbb{L}$. Then

$$-f(\delta) = \sup_{j\omega \in \mathbb{S}^1} \sup_{\|u\|=1} \sup_{\|v\|=1} \sup_{l \in \mathbb{L}} \Psi(u^\top T_{zw}(j\omega, \delta) v, l),$$

which is a representation of the form (3.8) with the compact space $\mathbb{K} = \mathbb{S}^1 \times \{u : \|u\| = 1\} \times \{v : \|v\| = 1\} \times \mathbb{L}$ and $F(\delta, j\omega, u, v, l) = \Psi(u^\top T_{wz}(j\omega, \delta) v, l)$. Hence $-f$ is lower-C^1. □

As a consequence, algorithm 1 may be applied to program (1.3) even in its reduced classical form. The constraint set is $C = \boldsymbol{\Delta}$, a hypercube, and it is therefore natural to choose the maximum norm $|\cdot|_\infty$ for the trust-regions. That makes (4.11) a linear program, which can even be solved explicitly, as we may restrict the number of cuts in $\phi_k^\sharp$ to one.

Let us next look at the spectral abscissa, $f(\delta) = -\alpha(A(\delta))$. Here the situation is more complicated. It has been observed in [7] that $\alpha(A(\delta))$ may even fail to be locally Lipschitz, so difficulties may be expected when $\alpha(A(\delta))$ is minimized. Here we are minimizing $-\alpha(A(\delta))$, which is better behaved. In our experiments we observe that $f(\delta) = -\alpha(A(\delta))$ behaves consistently like an upper-C^1 function. Nonetheless, since this is not always guaranteed, it is prudent to treat f under the weaker hypothesis that it has a strict standard model $\phi^\sharp$, in which case the full version of algorithm 1 may be needed. It turns out that if all active eigenvalues at $\alpha(A(\delta))$ are semi-simple, then f is locally Lipschitz in a neighborhood of δ, and the strictness condition on $\phi^\sharp$ is satisfied at least directionally. This gives at least a partial theoretical explanation why algorithm 1 works successfully on problems (1.4) and (1.5).

For (1.4) we choose $C = \boldsymbol{\Delta}$ and $|\cdot|_\infty$, so that (4.11) is again a linear program. As for (1.5), using exact penalization this can be transformed to

$$\min\{t + c\max(0, -\alpha(A(\delta))) : -t \leq \delta_i \leq t, i = 1, \ldots, m\},$$

with $\boldsymbol{x} = (t, \delta)$. Here $C = \{(t, \delta) \in \mathbb{R}^{1+m} : -t \leq \delta_i \leq t, i = 1, \ldots, m\}$ is again a simply structured convex set, and the standard model of the objective has properties similar to those of $-\alpha(A(\delta))$. With these choices local solutions of (1.5) are computed fast and reliably.

7 Stopping

Stopping a nonsmooth optimization method is not an easy task in general due to the occasionally slow convergence. Moreover, one can easily convince oneself that contrary to the smooth case the subdifferential $\partial f(\boldsymbol{x}^j)$ does not give a valid test, because in general $\min\{\|g\| : g \in \partial(f + i_C)(\boldsymbol{x}^j)\} \not\to 0$ as $\boldsymbol{x}^j \to \boldsymbol{x}^*$, where i_C is the indicator function of the set C; cf. [8]. Fortunately, the optimality conditions of (4.11) give the indication how to proceed. The solution $\boldsymbol{y}^k$ gives $\boldsymbol{g}_k \in \partial\left(\phi_k(\cdot, \boldsymbol{x}^j) + i_C\right)(\boldsymbol{y}^k)$ and $\boldsymbol{v}_k$ in the normal cone to the ball $B(\boldsymbol{x}^j, R_k)$ at $\boldsymbol{y}^k$ such that $\boldsymbol{g}_k + \boldsymbol{v}_k = 0$. Following classical lines in bundle methods, $\boldsymbol{g}_k$ is called the aggregate subgradient. The convergence proof in [3] shows that if $\boldsymbol{g}_j$ is the aggregate subgradient at acceptance of $\boldsymbol{z}^k = \boldsymbol{x}^{j+1}$ for the corresponding $\boldsymbol{y}^k = \tilde{\boldsymbol{x}}^{j+1}$, then $\boldsymbol{g}_j \to 0$. That leads to the following stopping test. If on acceptance of $\boldsymbol{x}^{j+1}$ in the outer loop

$$\frac{\|\boldsymbol{g}_j\|}{|f(\boldsymbol{x}^{j+1})| + 1} < \mathrm{tol}_1$$

is satisfied, then $\boldsymbol{x}^{j+1}$ is returned as optimal. On the other hand, it is also necessary to stop the algorithm in case the inner loop has difficulties converging. That can be tested as follows. If the inner loop gives ν consecutive unsuccessful trial steps $\boldsymbol{y}^k$ where

$$\frac{\|\boldsymbol{g}_k\|}{|f(\boldsymbol{x}^j)| + 1} < \mathrm{tol}_2,$$

then $\boldsymbol{x}^j$ is returned as optimal. It may also help to add tests as to the progress of the method, like

$$\frac{\|\boldsymbol{x}^j - \boldsymbol{x}^{j+1}\|}{\|\boldsymbol{x}^j\| + 1} < \mathrm{tol}_3, \qquad \frac{f(\boldsymbol{x}^j) - f(\boldsymbol{x}^{j+1})}{|f(\boldsymbol{x}^j)| + 1} < \mathrm{tol}_4,$$

but one has to be aware that the latter are based on convergence of the sequence $\boldsymbol{x}^j$, not on convergence *to a critical point*. Note that $\boldsymbol{g}_j \in \partial(f + i_C)(\boldsymbol{x}^j)$ is of the form $\boldsymbol{g}_j = \boldsymbol{p}_j + \boldsymbol{q}_j$, where $\boldsymbol{p}_j \in \partial f(\boldsymbol{x}^j)$ and $\boldsymbol{q}_j$ a normal vector to C at $\boldsymbol{x}^j$ in the boundary of $\boldsymbol{\Delta}$. That means, smallness of $P_C(-\boldsymbol{p}_j)$ is the correct indicator of optimality, where P_C is the orthogonal projection of C.

8 Experiments

Our first test concerns program (1.3), which we solve by Algorithm 1. Table 1 shows the characteristics of 27 benchmarks with n states, and uncertain structure $[r_1 \dots r_m]$, where $m = \mathrm{length}(\delta)$, and r_i the number of repetitions in (1.2). Here [1 1 1 3 1] is further compressed to $1^3 3^1 1^1$. The values achieved by algorithm 1 are shown as h^* in column 3 of table 2, the CPU is t^*.

To certify the result h^* of algorithm 1, we use the function `WCGAIN` of [5, 24], which is a branch-and-bound method combined with a frequency sweep. `WCGAIN` computes lower and upper bounds $\underline{h}, \overline{h}$ for the global maximum of (1.3) in t_{wc} seconds, and a solution δ_{wc} realizing $\underline{h}$. These results are shown in table 2.

♯	**Benchmark**	n	**Structure**
1	Beam1	11	$1^3 3^1 1^1$
2	Beam2	11	$1^3 3^1 1^1$
3	DC motor1	7	$1^1 2^2$
4	DC motor2	7	$1^1 2^2$
5	DVD driver1	10	$1^1 3^3 1^1 3^1$
6	Four-disk system1	16	$1^1 3^5 1^4$
7	Four-disk system2	16	$1^1 3^5 1^4$
8	Four-tank system1	12	1^4
9	Four-tank system2	12	1^4
10	Hard disk driver1	22	$1^3 2^4 1^4$
11	Hard disk driver2	22	$1^3 2^4 1^4$
12	Hydraulic servo1	9	1^9
13	Hydraulic servo2	9	1^9
14	Mass-spring1	8	1^2
15	Mass-spring2	8	1^2
16	Missile1	35	$1^3 6^3$
17	Missile2	35	$1^3 6^3$
18	Filter1	8	1^1
19	Filter2	8	1^1
20	Filter-Kim1	3	1^2
21	Filter-Kim2	3	1^2
22	Satellite1	11	$1^1 6^1 1^1$
23	Satellite2	11	$1^1 6^1 1^1$
24	Mass-spring-damper1	13	1^1
25	Mass-spring-damper2	13	1^1
26	RobToy1	3	$1^1 2^1$
27	RobToy2	3	$1^1 2^2 3^1$

Table 1: Benchmark problems for (1.3) and (1.5).

The interpretation of table 2 is that algorithm 1 and `wcgain` are in agreement in the majority of cases 1-5,7-9,11-13,16,17. Case 15 leaves a doubt, while cases 6,10,14,24,27 are failures of `WCGAIN`. On average algorithm 1 was 121-times faster than `WCGAIN`. Altogether, the fact that both methods are in good agreement in the majority of cases is a good sign and can be considered as an endorsement of our method.

8.1 Robust stability over $\boldsymbol{\Delta}$. Our second test concerns program (1.4). We use a bench of 32 cases (numbers 28-59) gathered in Table 3. Algorithm 1 converges very fast and delivers the values α^* in column 3 of table 4 in t^* seconds.

To certify α^* we have implemented the integral global optimization method of [22], also known as the

♯	$\underline{h}$	h^*	$\overline{h}$	t^*	$\overline{h}/h^*$	$t_{\rm wc}/t^*$
1	1.70	1.71	1.70	1.02	0.99	13.29
2	1.29	1.29	1.29	0.36	1	32.68
3	0.72	0.72	0.72	0.51	1.01	14.49
4	0.50	0.50	0.50	0.13	1	45.02
5	45.45	45.45	45.46	0.23	1	189.31
6	3.50	4.56	3.50	0.44	0.77	343.35
7	0.69	0.68	0.69	0.34	1.01	558.03
8	5.60	5.60	5.60	0.32	1	5.72
9	5.60	5.57	5.60	0.29	1	7.32
10	243.9	7526.6	Inf	0.96	Inf	73.10
11	0.03	0.03	0.03	0.20	1.12	314.92
12	1.17	1.17	1.17	0.34	1	10.94
13	0.7	0.70	0.7	0.33	1.01	11.69
14	3.71	6.19	3.71	0.31	0.60	3.54
15	6.84	6.84	7.16	0.13	1.05	7.05
16	5.12	5.15	5.12	0.46	0.99	272.54
17	1.83	1.82	1.83	0.22	1	1183.5
18	4.86	4.86	4.86	0.32	1	3.41
19	2.63	2.64	2.63	0.27	1	4.06
20	2.95	2.96	2.95	0.24	1	3.4
21	2.79	2.79	2.79	0.07	1	12.95
22	0.16	0.17	0.16	0.33	1	86.17
23	0.15	0.15	0.15	0.70	1	41.09
24	7.63	8.85	7.63	0.21	0.86	4.88
25	1.65	1.65	1.65	0.08	1	13.70
26	0.12	0.12	0.12	0.56	1	4.24
27	20.85	21.70	20.91	0.24	0.96	29.19

Table 2: Results for worst-case H_∞-norm on $\mathbf{\Delta}$

Zheng-method, for short ZM. In algorithm 2, μ is any continuous finite Borel measure on $\mathbf{\Delta}$. Numerical implementations use Monte-Carlo methods to compute the integral, and we refer to [22] for details. Our numerical tests are performed with $2000 \cdot m$ initial samples, where $m = \dim(\delta)$, and the stopping criterion is chosen at variance $= 10^{-12}$. The result obtained by ZM is $\alpha_{\rm ZM}$ in column 3 of table 4, obtained within $t_{\rm ZM}$ seconds CPU.

Algorithm 2. Zheng-method for $\alpha^* = \max_{\boldsymbol{x}\in\mathbf{\Delta}} f(\boldsymbol{x})$

▷ **Step 1** (`Initialize`)**.** Choose initial $\alpha < \alpha^*$.

▷ **Step 2** (`Loop`)**.** Compute $\alpha^+ = \dfrac{\int_{[f\geq\alpha]} f(x)\, d\mu(x)}{\mu[f \geq \alpha]}$.

▷ **Step 3** (`Stopping`)**.** If progress of α^+ over α is marginal, stop, otherwise $\alpha \to \alpha^+$ and goto step 2.

An interesting feature of ZM is that it can be initialized with the lower bound α^*, and this often leads to a significant speedup. Altogether ZM and our local method are in very good agreement on the test bench, which we consider an argument in favor of our approach.

♯	**Benchmark**	n	**Structure**
28	Beam3	11	$1^3 3^1 1^1$
29	Beam4	11	$1^3 3^1 1^1$
30	Dashpot system1	17	1^6
31	Dashpot system2	17	1^6
32	Dashpot system3	17	1^6
33	DC motor3	7	$1^1 2^2$
34	DC motor4	7	$1^1 2^2$
35	DVD driver2	10	$1^1 3^3 1^1 3^1$
36	Four-disk system3	16	$1^1 3^5 1^4$
37	Four-disk system4	16	$1^1 3^5 1^4$
38	Four-disk system5	16	$1^1 3^5 1^4$
39	Four-tank system3	12	1^4
40	Four-tank system4	12	1^4
41	Hard disk driver3	22	$1^3 2^4 1^4$
42	Hard disk driver4	22	$1^3 2^4 1^4$
43	Hydraulic servo3	9	1^9
44	Hydraulic servo4	9	1^9
45	Mass-spring3	8	1^2
46	Mass-spring4	8	1^2
47	Missile3	35	$1^3 6^3$
48	Missile4	35	$1^3 6^3$
49	Missile5	35	$1^3 6^3$
50	Filter3	8	1^1
51	Filter4	8	1^1
52	Filter-Kim3	3	1^2
53	Filter-Kim4	3	1^2
54	Satellite3	11	$1^1 6^1 1^1$
55	Satellite4	11	$1^1 6^1 1^1$
56	Satellite5	11	$1^1 6^1 1^1$
57	Mass-spring-damper3	13	1^1
58	Mass-spring-damper4	13	1^1
59	Mass-spring-damper5	13	1^1

Table 3: Benchmarks for (1.4).

8.2 Distance to instability In this last part we apply our algorithm to program (1.5) using the test bench in Table 5. The distance computed by algorithm 1 is d^* in column 2 of Table 6. To certify d^* we use three different tools. Firstly, ZM is used in the following way. For a given confidence level $\gamma = 0.05$ we compute

$$\underline{\alpha} = \max\{\alpha(A(\delta)) : \delta \in (1-\gamma)d^*\mathbf{\Delta}\} \tag{8.13}$$

and

$$\overline{\alpha} = \max\{\alpha(A(\delta)) : \delta \in (1+\gamma)d^*\mathbf{\Delta}\}. \tag{8.14}$$

Then d^* is certified by ZM with confidence level γ if $\underline{\alpha} < 0$ and $\overline{\alpha} > 0$. This happened in all cases except 87.

A very elegant way to certify parametric robust stability over a given set $\mathbf{\Delta}$ uses the method of Lasserre

♯	α^*	$\alpha_{\rm ZM}$	t^*	$t_{\rm ZM}$
28	-1.2e-7	-1.2e-7	0.19	32.70
29	-1.7e-7	-1.7e-7	0.04	33.00
30	0.0186	0.0185	0.23	90.25
31	-1.0e-6	-1.0e-6	0.39	39.63
32	-1.0e-6	-1.0e-6	0.08	39.63
33	-0.0010	-0.0010	0.02	20.63
34	-0.0010	-0.0010	0.02	20.74
35	-0.0165	-0.0165	0.04	49.29
36	0.0089	0.0089	0.10	159.91
37	-7.5e-7	-7.5e-7	0.29	73.86
38	-1.0e-7	-1.0e-7	0.29	74.63
39	-6.0e-6	-6.0e-6	0.02	26.03
40	-6.0e-6	-6.0e-6	0.02	26.2
41	266.70	266.70	0.09	297.21
42	-1.6026	-1.6026	0.06	80.4
43	-0.3000	-0.3000	0.04	51.41
44	-0.3000	-0.3000	0.02	50.95
45	-0.0054	-0.0054	0.01	31.59
46	-0.0368	-0.0370	0.01	16.94
47	22.6302	22.1682	0.07	104.18
48	-0.5000	-0.5000	0.07	51.78
49	-0.5000	-0.5000	0.07	52.24
50	-0.0148	-0.0148	0.06	7.05
51	-0.0148	-0.0148	0.02	6.89
52	-0.2500	-0.2500	0.01	12.83
53	-0.2500	-0.2500	0.01	12.9
54	3.9e-5	3.9e-5	0.02	44.02
55	-0.0269	-0.0269	0.02	26.02
56	-0.0268	-0.0268	0.02	26.08
57	0.2022	0.2022	0.01	8.30
58	-0.1000	-0.1000	0.01	6.91
59	-0.1000	-0.1000	0.01	6.94

Table 4: Results for worst-case spectral abscissa (1.4).

[15], where BMI problems are solved globally by a hierarchy of LMI approximations. When written as minimization problems, the value v_k of the kth approximating LMI is an upper bound of, and decreases toward, the global minimum value v_* of the BMI as $k \to \infty$. Moreover, convergence is often finite, that is, there exists k such that $v_k = v_*$. In that case, to solve the BMI, it suffices to find k and solve the corresponding kth LMI.

Following [12] one can certify robust stability over $\mathbf{\Delta}$ by showing that the value of the following polynomial optimization problem is > 0:

$$\begin{array}{ll} \text{minimize} & \det(H(\delta)) \\ \text{subject to} & \delta \in \mathbf{\Delta} \end{array} \tag{8.15}$$

where $H(\delta)$ is the so-called Hermite-matrix; see [12]. For $\mathbf{\Delta} = [-1, 1]^m$ in (8.15), the method [15] gives finite convergence. We follow [12] and apply GloptiPoly [13] to (8.15), where Maple 14 is used beforehand to compute the determinant of $H(\delta)$ formally. Based on (8.13) and (8.14) this leads to a procedure to certify or reject our heuristic d^*.

♯	**Benchmark**	n	**Structure**
60	Academic example	5	1^1
61	Academic example	4	1^3
62	Academic example	4	2^2
63	Inverted pendulum	4	1^3
64	DC motor	4	$1^3 2^1\ 1^1$
65	Bus steering system	9	$2^1 3^1$
66	Satellite	9	$2^1 1^2$
67	Bank-to-turn missile	6	1^4
68	Aeronautical vehicle	8	1^4
69	Four-tank system	10	1^4
70	Re-entry vehicle	6	$3^1 2^1 3^1$
71	Missile	14	1^4
72	Cassini spacecraft	17	1^4
73	Mass-spring-damper	7	1^6
74	Spark ignition engine	4	1^7
75	Hydraulic servo system	8	1^8
76	Academic example	41	$2^1 1^3$
77	Drive-by-wire vehicle	4	$1^2 2^7$
78	Re-entry vehicle	7	$1^3 6^1 4^1$
79	Space shuttle	34	1^9
80	Rigid aircraft	9	1^{14}
81	Fighter aircraft	10	$3^1 15^1 1^6 2^1 1^1$
82	Flexible aircraft	46	1^{20}
83	Telescope mockup	70	1^{20}
84	Hard disk drive	29	$1^8 2^4 1^1 1$
85	Launcher	30	$1^2 2^2 1^2 3^1 6^1 1^{12} 2^8$
86	Helicopter	12	30^4
87	Biochemical network	7	39^{13}

Table 5: Benchmarks for (1.5) available in [25].

The method was indeed able to certify d^* in cases 20, 21, 26 and 27. In the tests of table 6 the method was not able to furnish a decision even when the feasibility radius of the SDP-solver SeDuMi was enlarged to 10^3, and a large number of LMIs was considered. The bottleneck of the proposed method appears to be slow convergence $v_k \to v^*$, the fact that lower bounds cannot be taken into account, and the necessity to compute the determinant of $H(\delta)$ formally, which is impossible for matrices larger that 11×11. In all other aspects the method remains very promising.

Finally, we have also matched our heuristic d^* in Table 6 with the result $d_{\rm F}$ of another heuristic [11], which is tailored to problem (1.5). The results are in total agreement (see column 3 of table 6), and given the highly dedicated nature of [11], this can be understood as a further endorsement of our approach.

♯	d^*	$d_{\rm F}/d^*$	$\mathbf{D}_{\rm ZM}$	t^*	$t_{\rm ZM}$
60	0.79	1	$\surd$	0.15	7.3
61	3.41	1	$\surd$	0.13	23.9
62	0.58	1	$\surd$	0.15	97.4
63	0.84	1	$\surd$	0.22	24.7
64	1.25	1	$\surd$	0.19	37.7
65	1.32	0.99	$\surd$	0.37	13.8
66	1.01	0.99	$\surd$	0.3	20.2
67	0.60	0.99	$\surd$	0.17	167.7
68	0.61	0.99	$\surd$	0.19	38.9
69	6.67	0.99	$\surd$	0.27	24.9
70	6.20	1	$\surd$	0.44	21.8
71	7.99	1	$\surd$	0.25	24.9
72	0.06	1	$\surd$	0.13	25.1
73	1.17	1	$\surd$	0.17	2536.3
74	1.22	0.99	$\surd$	0.41	42.8
75	1.50	0.99	$\surd$	0.41	62.8
76	1.18	0.99	$\surd$	0.57	36.5
77	1	0.99	$\surd$	0.96	97.0
78	1.02	0.98	$\surd$	0.42	132.4
79	0.79	0.99	$\surd$	0.8	60.9
80	5.42	1	$\surd$	0.54	252.5
81	0.59	0.99	$\surd$	1.31	171.3
82	0.22	0.99	$\surd$	1.26	180.3
83	0.02	0.99	$\surd$	1.37	274.8
84	0.82	1	$\surd$	2.87	202.1
85	1.16	0.99	$\surd$	4.08	271.2
86	0.08	0.99	$\surd$	0.85	70.7
87	1.4e-3	1	failed	36.76	-

Table 6: Results for (1.5).

9 Conclusion

We have presented a local trust-region optimization method, which allows to compute good lower bounds for three NP-hard criteria in parametric robustness analysis. These are (a) the worst-case H_∞-norm of a system with real uncertain parameters over a given range, (b) the worst-case spectral abscissa of an uncertain system over that range, and (c) the distance to instability of a nominally stable system with real parametric uncertainty. A bench of 87 cases with systems up to 70 states, and with up to 39 uncertain parameters repeated up to 13 times, has been tested. It was shown that our local method gives excellent lower bounds, which in the majority of cases can be certified *a posteriori* as globally optimal using different global optimization tools.

References

[1] P. Apkarian, M.N. Dao, D. Noll, Parametric robust structured control design. *IEEE Trans. Autom. Control*, to appear 2015.

[2] P. Apkarian, D. Noll, Nonsmooth H_∞ synthesis. *IEEE Trans. Autom. Control* 51(1):2006,71-86.

[3] P. Apkarian, D. Noll, L. Ravanbod, Nonsmooth bundle trust-region algorithm. http://arxiv.org/abs/1504.00648

[4] V. Balakrishnan, Linear matrix inequalities in robustness analysis with multipliers, *Systems Control Letters*, 25(4):1995,265-272.

[5] G. J. Balas, J. C. Doyle, K. Glover, A. Packard, and R. Smith, μ-Analysis and synthesis toolbox: User's Guide. The MathWorks, Inc., 1991.

[6] R. D. Braatz, P. M. Young, J. C. Doyle, M. Morari, Computational complexity of μ calculation. *IEEE Trans. Autom. Control*, 39:1994,1000-1002.

[7] J. V. Burke, M. L. Overton, Differential properties of the spectral abscissa and the spectral radius for analytic matrix-valued mappings. *Nonlinear Anal.* 23(4):1994,467-488.

[8] F. H. Clarke, *Optimization and Nonsmooth Analysis*. John Wiley & Sons, Inc., New York, 1983.

[9] M. N. Dao, Bundle method for nonconvex nonsmooth constrained optimization, *J. Convex Analysis*, 2015.

[10] M. K. H. Fan, A. L. Tits, J. C. Doyle, Robustness in the presence of mixed parametric uncertainty and unmodelized dynamics, *IEEE Trans. Aut. Con.*, AC-36:1991, 25-38.

[11] A. Fabrizi, C. Roos, J.M. Biannic, A detailed comparative analysis of lower bound algorithms. *European Control Conference* 2014, Strasbourg, France.

[12] D. Henrion, J. B. Lasserre, GloptiPoly: global optimization over polynomials with Matlab and SeDuMi. *ACM Trans. Math. Software*, 29(2):2003,165-194.

[13] D. Henrion, D. Arzelier, D. Peaucelle, J.-B. Lasserre, On parameter-dependent Lyapunov functions for robust stability of linear systems. *43rd IEEE Conf. Dec. Con.*, 2004 Atlantis, Paradise Island, Bahamas

[14] D. Hinrichsen, A.J. Pritchard, Mathematical System Theory I. Modeling, State Space Analysis, Stability and Robustness. Springer Verlag 2005.

[15] J.-B. Lasserre, Global optimization with polynomials and the problem of moments. *SIAM J. Optim.* 11:2001,796-817.

[16] D. Noll, Convergence of non-smooth descent methods using the Kurdyka-Łojasiewicz inequality. *J. Optim. Theory Appl.* 160(2):2014, 553-572.

[17] D. Noll, O. Prot, A. Rondepierre, A proximity control algorithm to minimize nonsmooth and nonconvex functions. *Pac. J. Optim.* 4(3):2008,571-604.

[18] S. Poljak, J. Rohn, Checking robust nonsingularity is NP-hard. *Math. Control Signals Sys.* 6:1993,1-9.

[19] J. Sreedhar, P. van Dooren, A.L. Tits, A fast algorithm to compute the real structured stability radius. *Int. Series of Numer. Math.,* Birkhäuser Verlag, pp. 219-230:1996.

[20] J. E. Spingarn, Submonotone subdifferentials of Lipschitz functions. *Trans. Amer. Math. Soc.* 264(1):1981,77-89.

[21] L. N. Trefethen, Computation of pseudospectra. *Acta Numerica*, vol. 8,1999.

[22] Q. Zhen and D. Zhuang, Integral global minimization: algorithms, implementations, and numerical tests. *Journal of Global Optimization,* 7:1995,421-454.

[23] K. Zhou, J. C. Doyle, K. Glover. *Robust and Optimal Control*. Prentice Hall, New Jersey, 1996.

[24] Robust Control Toolbox 5.0. MathWorks, Natick, MA, USA, Sept 2013.

[25] SMAC Toolbox, ONERA 2012-15, http://w3.onera.fr/smac