Parametric robust structured control design

Pierre Apkarian, Minh Ngoc Dao, and Dominikus Noll

***Abstract*—We present a new approach to parametric robust controller design, where we compute controllers of arbitrary order and structure which minimize the worst-case H_∞ norm over a pre-specified set of uncertain parameters. At the core of our method is a nonsmooth minimization method tailored to functions which are semi-infinite minima of smooth functions. A rich test bench and a more detailed example illustrate the potential of the technique, which can deal with complex problems involving multiple possibly repeated uncertain parameters.**

***Index Terms*—Real uncertain parameters, structured H_∞-synthesis, parametric robust control, nonsmooth optimization, local optimality, inner approximation.**

I. Introduction

PARAMETRIC uncertainty is among the most challenging problems in control system design due to its NP-hardness. Albeit, being able to provide solutions to this fundamental problem is a must for any practical design tool worthy of this attribute. Not surprisingly, therefore, parametric uncertainty has remained high up on the agenda of unsolved problems in control for the past three decades.

It is of avail to distinguish between analysis and synthesis techniques for parametric robustness. Analysis refers to assessing robustness of a closed-loop system when the controller is already given. If the question whether this given controller renders the closed loop parametrically robustly stable is solved exhaustively, then it is already an NP-hard problem [1]. Parametric robust synthesis, that is, computing a controller which is robust against uncertain parameters, is even harder, because it essentially involves an iterative procedure where at every step an analysis problem is solved. Roughly, we could say that in parametric robust synthesis we have to optimize a criterion, a single evaluation of which is already NP-hard.

For the analysis of parametric robustness, theoretical and practical tools with only mild conservatism and acceptable CPUs have been proposed over the years [2]. In contrast, no tools with comparable merits in terms of quality and CPU are currently available for synthesis. It is fair to say that the parametric robust synthesis problem has remained open. The best currently available techniques for synthesis are the μ tools going back to [3], made available to designers through the MATLAB Robust Control Toolbox. These rely on upper bound approximations of μ and follow a heuristic which alternates between analysis and synthesis steps. When this works, it gives performance and stability certificates, but the approach may turn out conservative, and the computed controllers are often too complicated for practice.

The principal obstruction to efficient robust synthesis is the inherent nonconvexity and nonsmoothness of the mathematical program underlying the design. These obstacles have to some extent been overcome by the nonsmooth approaches [4], [5], [6] for control. These techniques allow to address multi-model and multi-objective structured control design as discussed in [7], [8], [9], [10], [11]. These have become available to designers through synthesis tools like `HINFSTRUCT` or `SYSTUNE` from [12]. However, these methods can no longer be used for the substantially harder parametric robust synthesis problem, and this has been an incentive for our present work, where we investigate new classes of optimization programs involving upper-C^1 stability and performance criteria.

In order to understand our approach, it is helpful to distinguish between inner and outer approximations of the robust control problem on a set $\mathbf{\Delta}$ of uncertain parameters. Outer approximations relax the problem over $\mathbf{\Delta}$ by choosing a larger, but more convenient, set $\widetilde{\mathbf{\Delta}} \supset \mathbf{\Delta}$, the idea being that the problem on $\widetilde{\mathbf{\Delta}}$ becomes accessible to computations. If solved successfully on $\widetilde{\mathbf{\Delta}}$, this provides performance and robustness certificates for $\mathbf{\Delta}$. Typical tools in this class are the upper bound approximation $\overline{\mu}$ of the structured singular value μ developed in [13], the DK-iteration function `DKSYN` of [12], or LMI-based approaches like [14]. The principal drawback of outer approximations is the inherent conservatism, which increases significantly with the number of uncertainties and their repetitions, and the fact that failures occur more often.

Inner approximations are preferred in practice and relax the problem by solving it on a smaller typically finite subset $\mathbf{\Delta}_a \subset \mathbf{\Delta}$. This avoids conservatism and leads to acceptable CPUs, but has the disadvantage that no immediate stability or performance certificate for $\mathbf{\Delta}$ is obtained. Our principal contribution here is to show a way how this shortcoming can be avoided. We present an efficient technique to compute an inner approximation with structured controllers with a local optimality certificate such that robust stability and performance are achieved over $\mathbf{\Delta}$ in the majority of cases. We then also show how this can be certified a posteriori over $\mathbf{\Delta}$, when combined with outer approximation for analysis. The new method we propose is termed *dynamic inner approximation*, as it generates the inner approximating set $\mathbf{\Delta}_a$ dynamically. The idea of using inner approximations, and thus multiple models, to solve robust synthesis problems is not new and was employed in different contexts, see e.g. [15], [16], [17].

To address the parametric robust synthesis problem we use a nonsmooth optimization method tailored to minimizing a cost function, which is itself a semi-infinite minimum of smooth

P. Apkarian is with the Control System Department, ONERA, 2, av. Ed. Belin, 31055, Toulouse, France - Tel: +33562252784, Fax: +33562252564, `Pierre.Apkarian@onera.fr`

M. N. Dao is with the Department of Mathematics and Informatics, Hanoi National University of Education, Vietnam, and also with the Institut de Mathématiques de Toulouse, 118 route de Narbonne F-31062, Toulouse, France - Tel: +33561557656, Fax: 33561557599, `minhdn@hnue.edu.vn`

D. Noll is with the Institut de Mathématiques de Toulouse, 118 route de Narbonne F-31062, Toulouse, France - Tel: +33561558622, Fax: 33561557599, `noll@mip.ups-tlse.fr`

functions. This is in contrast with previously discussed nonsmooth optimization problems, where a semi-infinite maximum of smooth functions is minimized, and which have been dealt with successfully in [9]. At the core of our new approach is therefore understanding the principled difference between a min-max and a min-min problem, and the algorithmic strategies required to solve them successfully. Along with the new synthesis approach, our key contributions are

- an in-depth and rigorous analysis of worst-case stability and worst-case performance problems over a compact parameter range.
- the description of a new resolution algorithm for worst-case programs along with a proof of convergence in the general nonsmooth case.

Note that convergence to local minima from an arbitrary, even remote, starting point is proved. Convergence to a global minimum is as a rule unrealistic in terms of CPU due to the NP-hardness of the problems.

The paper is organized as follows. Section II states the problem formally, and Subsection II-B presents our novel dynamic inner approximation technique and the elements needed to carry it out. Section III highlights the principal differences between nonsmooth min-min and min-max problems. Sections IV-A and IV-B examine the criteria which arise in the optimization programs, the H_∞-norm, and the spectral abscissa. Section V presents the optimization method we designed for min-min problems and Subsections V-B, V-C are dedicated to convergence analysis. Section VI-A presents an assessment and a comparison of our algorithm on a bench of test examples. Section VI-B gives a more refined study of a challenging missile control problem.

NOTATION

For complex matrices X^H denotes conjugate transpose. For Hermitian matrices, $X \succ 0$ means positive definite, $X \succeq 0$ positive semi-definite. We use concepts from nonsmooth analysis covered by [18]. For a locally Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$, $\partial f(x)$ denotes its (compact and convex) Clarke subdifferential at $x \in \mathbb{R}^n$. The Clarke directional derivative at x in direction $d \in \mathbb{R}^n$ can be computed as

$$f^\circ(x,d) = \max_{g \in \partial f(x)} g^T d\,.$$

The symbols $\mathcal{F}_l$, $\mathcal{F}_u$ denote lower and upper Linear Fractional Transformations (LFT) [19]. For partitioned 2×2 block matrices, $\star$ stands for the Redheffer star product [20].

II. PARAMETRIC ROBUSTNESS

A. Setup

We consider an LFT plant as in Fig. 1 with real parametric uncertainties $\mathcal{F}_u(P,\Delta)$ where

$$P(s): \begin{cases} \dot{x} &= Ax + B_p p + B_w w + Bu \\ q &= C_q x + D_{qp} p + D_{qw} w + D_{qu} u \\ z &= C_z x + D_{zp} p + D_{zw} w + D_{zu} u \\ y &= Cx + D_{yp} p + D_{yw} w + Du \end{cases} \tag{1}$$

and $x \in \mathbb{R}^{n_x}$ is the state, $u \in \mathbb{R}^{m_2}$ the control, $w \in \mathbb{R}^{m_1}$ the vector of exogenous inputs, $y \in \mathbb{R}^{p_2}$ the output, and $z \in \mathbb{R}^{p_1}$ the regulated output. The uncertainty channel is defined as $p = \Delta q$ where the uncertain matrix Δ is without loss assumed to have the block-diagonal form

$$\Delta = \operatorname{diag}\left[\delta_1 I_{r_1}, \ldots, \delta_m I_{r_m}\right] \tag{2}$$

with $\delta_1, \ldots, \delta_m$ representing real uncertain parameters, and r_i giving the number of repetitions of δ_i. We assume without loss that $\delta = 0$ represents the nominal parameter value. Moreover, we consider $\delta \in \mathbf{\Delta}$ in one-to-one correspondence with the matrix Δ in (2).

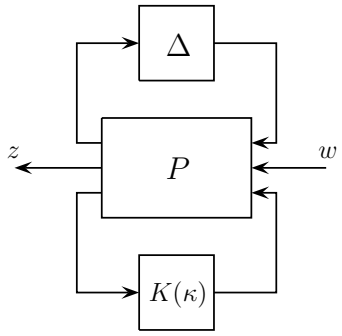

Fig. 1: Robust synthesis interconnection

Given a compact convex set $\mathbf{\Delta} \subset \mathbb{R}^m$ containing $\delta = 0$, the parametric robust structured H_∞ control problem consists in computing a structured output-feedback controller $u = K(\kappa^*)y$ with the following properties:

(i) **Robust stability**. $K(\kappa^*)$ stabilizes $\mathcal{F}_u(P,\Delta)$ internally for every $\delta \in \mathbf{\Delta}$.

(ii) **Robust performance**. Given any other robustly stabilizing controller $K(\kappa)$ with the same structure, the optimal controller satisfies

$$\max_{\delta\in\mathbf{\Delta}} \|T_{zw}(\delta,\kappa^*)\|_\infty \le \max_{\delta\in\mathbf{\Delta}} \|T_{zw}(\delta,\kappa)\|_\infty.$$

Here $T_{zw}(\delta,\kappa) := \mathcal{F}_l\left(\mathcal{F}_u(P,\Delta(\delta)),\, K(\kappa)\right)$ is the closed-loop transfer function of the performance channel $w \to z$ of (1) when the control loop with $K(\kappa)$ and the uncertainty loop with Δ are both closed.

We recall that according to [5] a controller

$$K(\kappa): \begin{cases} \dot{x}_K &= A_K(\kappa)x_K + B_K(\kappa)y \\ u &= C_K(\kappa)x_K + D_K(\kappa)y \end{cases} \tag{3}$$

in state-space form is called *structured* if $A_K(\kappa), B_K(\kappa), \ldots$ depend smoothly on a design parameter κ varying in a design space $\mathbb{R}^n$ or in some constrained subset of $\mathbb{R}^n$. Typical examples of structure include PIDs, reduced-order controllers, observer-based controllers, or complex control architectures combining controller blocks such as set-point filters, feed-forward, washout or notch filters, and much else [9]. In contrast, full-order controllers are state-space representations with the same order as $P(s)$ without particular structure, and are sometimes referred to as *unstructured*, or as *black-box controllers*.

Parametric robust control is among the most challenging problems in linear feedback control. The structured singular value μ developed in [19] is the principled theoretical tool to describe problem (i), (ii) formally. In the same vein, based on the spectral abscissa

$$\alpha(A) = \max\{\mathrm{Re}(\lambda) : \lambda \text{ eigenvalue of } A\}$$

of a square matrix A, criterion (i) may be written as

$$\max_{\delta\in\mathbf{\Delta}} \alpha\left(A(\delta,\kappa^*)\right) < 0, \tag{4}$$

where $A(\delta,\kappa)$ is the A-matrix of the closed-loop transfer function $T_{zw}(\delta,\kappa)$.

If the uncertain parameter set is a cube $\mathbf{\Delta} = [-1,1]^m$, which is general enough for applications, then the same information is obtained from the distance to instability in the maximum-norm

$$d^* = \min\{\|\delta\|_\infty : \alpha\left(A(\delta,\kappa^*)\right) \geq 0\}, \tag{5}$$

because criterion (i) is now equivalent to $d^* > 1$. It is known that the computation of any of these elements, μ, (4), or (5) is NP-complete, so that their practical use is limited to analysis of small problems, or to the synthesis of tiny ones. Practical approaches have to rely on intelligent relaxations, or *heuristics*, which use either inner or outer approximations.

In the next chapters we will develop our *dynamic inner approximation* method to address problem (i), (ii). We solve the problem on a relatively small set $\mathbf{\Delta}_a \subset \mathbf{\Delta}$, which we construct iteratively.

B. Dynamic inner approximation

The following *static* inner approximation to (i), (ii) is near at hand. After fixing a sufficiently fine approximating static grid $\mathbf{\Delta}_s \subset \mathbf{\Delta}$, one solves the multi-model H_∞-problem

$$\min_{\kappa\in\mathbb{R}^n} \max_{\delta\in\mathbf{\Delta}_s} \|T_{zw}\left(\delta,\kappa\right)\|_\infty. \tag{6}$$

This may be addressed with the methods of [5], [11], [21], and the software tools `HINFSTRUCT` and `SYSTUNE` [22], [7], [8], [12], but has a high computational burden due to the large number of scenarios in $\mathbf{\Delta}_s$, which makes it prone to failure. Straightforward gridding becomes very quickly intractable for sizable $\dim(\delta)$.

Here we advocate a different strategy, which we call *dynamic* inner approximation, because it operates on a substantially smaller set $\mathbf{\Delta}_a \subset \mathbf{\Delta}$ generated dynamically, whose elements are called the *active scenarios*, which we update a couple of times by applying a search procedure locating problematic parameter scenarios in $\mathbf{\Delta}$. This leads to a rapidly converging procedure, much less prone to failure than (6). The method can be summarized as shown in Algorithm 1.

The principal elements of Algorithm 1 will be analyzed in the following sections. We will focus on the optimization programs v^* in step 4, α^* in step 3, and d^*, h^* in step 6, which represent a relatively unexplored type of nonsmooth programs, with some common features which we shall put into evidence here. In contrast, program v_* in step 2 is accessible to numerical methods through [5] and can be addressed with tools like `HINFSTRUCT` or `SYSTUNE` available through [12], or `HIFOO` available through [10]. Note that our approach is heuristic in so far as we have relaxed (i) and (ii) by computing locally optimal solutions, so that a global stability/performance certificate is only provided in the end as a result of step 6.

Algorithm 1. Dynamic inner approximation for parametric robust synthesis over $\mathbf{\Delta}$

Parameters: $\varepsilon > 0$.

▷ **Step 1** (`Nominal synthesis`)**.** Initialize the set of active scenarios as $\mathbf{\Delta}_a = \{0\}$.

▷ **Step 2** (`Multi-model synthesis`)**.** Given the current finite set $\mathbf{\Delta}_a \subset \mathbf{\Delta}$ of active scenarios, compute a structured multi-model H_∞-controller by solving

$$v_* = \min_{\kappa\in\mathbb{R}^n} \max_{\delta\in\mathbf{\Delta}_a} \|T_{zw}\left(\delta,\kappa\right)\|_\infty.$$

The solution is the structured H_∞-controller $K(\kappa^*)$.

◇ **Step 3** (`Destabilization`)**.** Try to destabilize the closed-loop system $T_{zw}\left(\delta,\kappa^*\right)$ by solving the destabilization problem

$$\alpha^* = \max_{\delta\in\mathbf{\Delta}} \alpha\left(A(\delta,\kappa^*)\right).$$

If $\alpha^* \geq 0$, then the solution $\delta^* \in \mathbf{\Delta}$ destabilizes the loop. Include δ^* in the active scenarios $\mathbf{\Delta}_a$ and go back to step 2. If no destabilizing δ was found then go to step 4.

▷ **Step 4** (`Degrade performance`)**.** Try to degrade the robust H_∞-performance by solving

$$v^* = \max_{\delta\in\mathbf{\Delta}} \|T_{zw}\left(\delta,\kappa^*\right)\|_\infty.$$

The solution is δ^*.

◇ **Step 5** (`Stopping test`)**.** If $v^* < (1+\varepsilon)v_*$ degradation of performance is only marginal. Then exit, or optionally, go to step 6 for post-processing. Otherwise include δ^* among the active scenarios $\mathbf{\Delta}_a$ and go back to step 2.

◇ **Step 6** (`Post-processing`)**.** Check robust stability (i) and performance (ii) of $K(\kappa^*)$ over $\mathbf{\Delta}$ by computing the distance d^* to instability (5), and its analogue $h^* = \min\{\|\delta\|_\infty : \|T_{zw}(\delta,\kappa^*)\|_\infty \geq v^*\}$. Possibly use μ-tools from [12] to assess d^*, h^* approximately. If all δ^* obtained satisfy $\delta^* \notin \mathbf{\Delta}$, then terminate successfully.

III. Nonsmooth min-max versus min-min programs

A. Classification of the programs in Algorithm 1

Introducing the functions $a_\pm(\delta) = \pm\alpha\left(A(\delta)\right)$, the problem of step 3 can be equivalently written in the form

$$\begin{array}{ll} \text{minimize} & a_-(\delta) = -\alpha\left(A(\delta)\right) \\ \text{subject to} & \delta \in \mathbf{\Delta} \end{array} \tag{7}$$

for a matrix $A(\delta)$ depending smoothly on the parameter $\delta \in \mathbb{R}^m$. Here the dependence of the matrix on controller $K(\kappa^*)$ is omitted for simplicity, as the latter is fixed in step 3 of the algorithm. Similarly, if we introduce $h_\pm(\delta) = \pm\|G(\delta)\|_\infty$,

with $G(s,\delta)$ a transfer function depending smoothly on $\delta \in \mathbb{R}^m$, then problem of step 4 has the abstract form

$$\begin{array}{ll} \text{minimize} & h_-(\delta) = -\|G(\delta)\|_\infty \\ \text{subject to} & \delta \in \mathbf{\Delta} \end{array} \tag{8}$$

where again controller $K(\kappa^*)$ is fixed in step 4, and therefore suppressed in the notation. In contrast, the H_∞-program v_* in step 2 of Algorithm 1 has the form

$$\begin{array}{ll} \text{minimize} & h_+(\kappa) = \|G(\kappa)\|_\infty \\ \text{subject to} & \kappa \in \mathbb{R}^n \end{array} \tag{9}$$

which is of the more familiar min-max type. Here we use the well-known fact that the H_∞-norm may be written as a semi-infinite maximum function $h_+(\kappa) = \max_{\omega\in[0,\infty]} \overline{\sigma}\left(G(\kappa, j\omega)\right)$. The maximum over the finitely many $\delta \in \mathbf{\Delta}_a$ in step 2 complies with this structure and may in principle be condensed into the form (9), featuring only a single transfer $G(s,\kappa)$. In practice this is treated as in [11].

Due to the minus sign, programs (7) and (8), written in the minimization form, are now of the novel min-min type, which is given special attention here. This difference is made precise by the following

Definition 1 (Spingarn [23])**.** A locally Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$ is lower-C^1 at $x_0 \in \mathbb{R}^n$ if there exist a compact space $\mathbb{K}$, a neighborhood U of x_0, and a mapping $F : \mathbb{R}^n \times \mathbb{K} \to \mathbb{R}$ such that

$$f(x) = \max_{y\in\mathbb{K}} F(x,y) \tag{10}$$

for all $x \in U$, and F and $\partial F/\partial x$ are jointly continuous. The function f is said to be upper-C^1 if $-f$ is lower-C^1. □

We expect upper- and lower-C^1 functions to behave quite differently in descent algorithms. Minimization of lower-C^1 functions, as required in (9), should lead to a genuinely nonsmooth problem, because iterates of a descent method move toward the points of nonsmoothness. In contrast, minimization of upper-C^1 functions as required in (7) and (8) is expected to be better behaved, because iterates move away from the nonsmoothness. Accordingly, we will want to minimize upper-C^1 functions in (7) and (8) in much the same way as we optimize smooth functions in classical nonlinear programming, whereas the minimization of lower-C^1 functions in (9) requires specific techniques like nonconvex bundle methods [24], [25], [5]. See Fig. 2 for an illustration.

Remark 1 (Distance to instability)**.** Note that the computation of the distance to instability d^* defined in (5) for step 6 of Algorithm 1 has also the features of a min-min optimization program. Namely, when written in the form

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -t \le \delta_i \le t,\ i = 1,\dots,m \\ & -\alpha\left(A(\delta)\right) \le 0 \end{array} \tag{11}$$

with variable $(\delta, t) \in \mathbb{R}^{m+1}$, the Lagrangian of (5) is

$$L(\delta,t,\lambda,\mu_\pm) = t + \sum_{i=1}^{m} \mu_{i-}\left(-t-\delta_i\right) + \mu_{i+}(\delta_i - t) - \lambda\alpha\left(A(\delta)\right)$$

for Lagrange multipliers $\lambda \ge 0$ and $\mu_\pm \ge 0$. In particular, if $(\delta^*, t^*, \lambda^*, \mu_\pm^*)$ is a Karush-Kuhn-Tucker point of (11), (see [18, Prop. 6.6.4], or [26, Sect. 3.3.1]), then the local minimum (δ^*, t^*) we are looking for is also a critical point of the unconstrained program

$$\min_{\delta\in\mathbb{R}^m, t\in\mathbb{R}} L(\delta, t, \lambda^*, \mu_\pm^*),$$

which features the function a_- and is therefore of min-min type. Therefore, in solving (5), we expect phenomena of min-min type to surface rather than those of a min-max program. A similar comment applies to the computation of h^* in step 6 of the algorithm.

Remark 2 (Well-posedness)**.** Yet another aspect of Algorithm 1 is that in order to be robustly stable over the parameter set $\mathbf{\Delta}$, the LFTs must be well-posed in the sense that $(I - \Delta\mathcal{D})^{-1}$ exists for every $\delta \in \mathbf{\Delta}$, where $\mathcal{D}$ is the closed-loop D-matrix. Questioning well-posedness could therefore be included in step 3 of the algorithm, or added as posterior testing in step 6. It can be formulated as yet another min-min program

$$\begin{array}{ll} \text{minimize} & -\overline{\sigma}((I - \Delta\mathcal{D})^{-1}) \\ \text{subject to} & \delta \in \mathbf{\Delta} \end{array} \tag{12}$$

where one would diagnose the solution δ^* to represent an ill-posed scenario as soon as it achieves a large negative value. Program (12) exhibits the same properties as minimizing h_- in Section IV-A and is handled with the same novel techniques.

For programs v^* in step 4, α^* in step 3, and d^*, h^* in step 6 of Algorithm 1, well-posedness (12) is a prerequisite. However, we have observed that it may not be necessary to question well-posedness over $\mathbf{\Delta}$ at every step, since questioning stability over $\mathbf{\Delta}$ has a similar effect. Since the posterior certificate in step 6 of the algorithm covers also well-posedness, this is theoretically justified.

Remark 3. Our notation makes it easy for the reader to distinguish between min-min and min-max programs. Namely, minimizations over the controller variable κ turn out the min-max ones, while minimizations over the uncertain parameters δ lead to the min-min type.

Remark 4. Note that min-min programs (7), (8) require only smoothness of $A(\delta)$, respectively, $G(\delta)$. This is practical in problems where LFT representations are not readily available.

B. Highlighting the difference between min-max and min-min

In this section we look at the typical difficulties which surface in min-max and min-min programs. This is crucial for the understanding of our algorithmic approach. Consider first a min-max program of the form

$$\min_{\kappa\in\mathbb{R}^n} \max_{i\in I} f_i(\kappa), \tag{13}$$

where the f_i are smooth. When the set I is finite, we may simply dissolve this into a classical nonlinear programming (NLP) using one additional dummy variable $t \in \mathbb{R}$:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & f_i(\kappa) \le t,\ i \in I. \end{array}$$

The situation becomes more complicated as soon as the set I is infinite, as is for instance the case in program v_* in step 2 of

Algorithm 1. The typical difficulty in min-max programs is to deal with this semi-infinite character, and one is beholden to use a tailored solution, as for instance developed in [5], [25], [24]. Altogether this type of difficulty is well-known and has been thoroughly studied.

In contrast, a min-min program

$$\min_{\delta\in\mathbb{R}^n}\min_{i\in I} f_i(\delta) \tag{14}$$

cannot be converted into an NLP even when I is finite. The problem has disjunctive character, and if solved to global optimality, min-min programs lead to combinatorial explosion. On the other hand, a min-min problem has some favorable features when it comes to solely finding a good local minimum. Namely, when meeting a nonsmooth iterate δ^j, where several branches f_i are active, we can simply pick one of those branches and continue optimization as if the objective function were smooth. In the subsequent sections we prove that this intuitive understanding is indeed correct. Our experimental section will show that good results are obtained if a good heuristic is used.

The above considerations lead us to introduce the notion of active indices and branches for functions $f(\delta)$ defined by the inner $\max$ and $\min$ in (13) and (14).

Definition 2. The set of active indices for f at δ is defined as

$$I(\delta) := \{i \in I : f_i(\delta) = f(\delta)\} \ .$$

Active branches of f at δ are those corresponding to active indices, i.e, f_i, $i \in I(\delta)$.

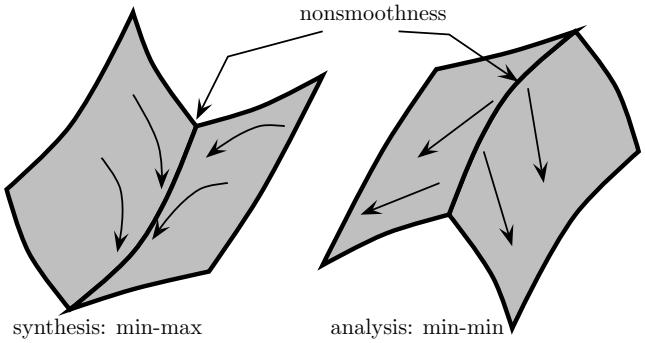

Fig. 2: Minimization of a minimum (right) may require deciding along which branch to proceed, hence the potentially combinatorial character of (14). Minimizing a maximum (left) requires simultaneous minimization of active branches, hence the non smoothness and the potentially semi-infinite character of (13).

IV. Computing subgradients

In this section we briefly discuss how the subgradient information needed to minimize h_- and a_- is computed.

A. Case of the H_∞-norm

We start by investigating the case of the H_∞-norm $h_\pm$. We recall that function evaluation is based on the Hamiltonian algorithm of [27], [28] and its further developments [29]. Computation of subgradients of h_- in the sense of Clarke can be adapted from [5], see also [30]. We assume the controller is fixed in this section and investigate the properties of h_- as a function of δ. To this aim, the controller loop is closed by substituting the structured controller (3) in (1), and we obtain the transfer function $M(\kappa) := \mathcal{F}_l(P, K(\kappa))$. Substantial simplification in Clarke subdifferential computation is then obtained by defining the 2×2-block transfer function

$$\begin{bmatrix} * & T_{qw}(\delta) \\ T_{zp}(\delta) & T_{zw}(\delta) \end{bmatrix} := \begin{bmatrix} 0 & I \\ I & \Delta \end{bmatrix} \star M \,, \tag{15}$$

where the dependence on κ has now been suppressed, as the controller will be fixed to κ^* after step 2. It is readily seen that T_{zw} coincides with the closed-loop transfer function where both controller and uncertainty loops are closed.

Now consider the function $h_-(\delta) := -\|T_{zw}(\delta)\|_\infty$, which is well defined on its domain $\mathbb{D} := \{\delta \in \mathbb{R}^m : T_{zw}(\delta) \text{ is internally stable}\}$. We have the following

Proposition 1. *The function h_- is everywhere Clarke subdifferentiable on $\mathbb{D}$. The Clarke subdifferential at $\delta \in \mathbb{D}$ is the compact and convex set*

$$\partial h_-(\delta) = \Big\{ \phi_Y : Y = (Y_\omega), \omega \in \Omega(\delta),\, Y_\omega \succeq 0, \\ \textstyle\sum_{\omega\in\Omega(\delta)} \mathrm{Trace}(Y_\omega) = 1 \Big\},$$

where the i-th entry of ϕ_Y is $\mathrm{Trace}\left(\Delta_i^T \Phi_Y\right)$ with $\Delta_i = \partial\Delta/\partial\delta_i$, and

$$\Phi_Y = -\sum_{\omega\in\Omega(\delta)} \mathrm{Re}\left(T_{qw}(\delta, j\omega) P_\omega Y_\omega Q_\omega^H T_{zp}(\delta, j\omega)\right)^T .$$

Here $\Omega(\delta)$ is the set of active frequencies at δ, Q_ω is a matrix whose columns are the left singular vectors associated with the maximum singular value of $T_{zw}(\delta, j\omega)$, P_ω is the corresponding matrix of right singular vectors, and Y_ω is an Hermitian matrix of appropriate size.

Proof: Computation of the Clarke subdifferential of h_- can be obtained from the general rule $\partial(-h) = -\partial h$, and knowledge of ∂h_+, see [5]. Note that in that reference the Clarke subdifferential is with respect to the controller and relies therefore on the Redheffer star product

$$P \star \begin{bmatrix} K(\kappa) & I \\ I & 0 \end{bmatrix} .$$

Here we apply this in the upper loop in Δ, so we have to use the analogue expression (15) instead.

■

Remark 5. In the case where a single frequency ω_0 is active at δ and the maximum singular value $\overline{\sigma}$ of $T_{zw}(\delta, j\omega_0)$ has multiplicity 1, h_- is differentiable at δ and the gradient is

$$\frac{\partial h_-(\delta)}{\partial \delta_i} = -\mathrm{Trace}\,\mathrm{Re}\,\left(T_{qw}(\delta, j\omega_0) p_{\omega_0} q_{\omega_0}^H T_{zp}(\delta, j\omega_0)\right)^T \Delta_i,$$

where p_{ω_0} and q_{ω_0} are the unique right and left singular vectors of $T_{zw}(\delta, j\omega_0)$ associated with $\overline{\sigma}(T_{zw}(\delta, j\omega_0)) = h_+(\delta)$.

Proposition 2. *Let $\mathbb{D} = \{\delta : T_{zw}(\delta) \text{ is internally stable}\}$. Then $h_+ : \delta \mapsto \|T_{zw}(\delta)\|_\infty$ is lower-C^1 on $\mathbb{D}$, so that $h_- : \delta \mapsto -\|T_{zw}(\delta)\|_\infty$ is upper-C^1 there.*

Proof: Recall that the maximum singular value has the variational representation

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

Now observe that $z \mapsto |z|$, being convex, is lower-C^1 as a mapping $\mathbb{R}^2 \to \mathbb{R}$, so we may write it as

$$|z| = \sup_{l \in \mathbb{L}} \Psi(z, l)$$

for Ψ jointly of class C^1 and $\mathbb{L}$ compact. Then

$$h_+(\delta) = \sup_{j\omega \in \mathbb{S}^1} \sup_{\|u\|=1} \sup_{\|v\|=1} \sup_{l \in \mathbb{L}} \Psi\left(u^T T_{zw}(\delta, j\omega) v, l\right), \quad (16)$$

where $\mathbb{S}^1 = \{j\omega : \omega \in \mathbb{R} \cup \{\infty\}\}$ is homeomorphic with the 1-sphere. This is the desired representation (10), where the compact space $\mathbb{K}$ is obtained as $\mathbb{K} := \mathbb{S}^1 \times \{u : \|u\| = 1\} \times \{v : \|v\| = 1\} \times \mathbb{L}$, F as $F(\delta, j\omega, u, v, l) := \Psi\left(u^T T_{zw}(\delta, j\omega) v, l\right)$ and y as $y := (j\omega, u, v, l)$. ∎

B. Case of the spectral abscissa

For the spectral abscissa the situation is more complicated, as $a_\pm$ is not locally Lipschitz everywhere. Recall that an eigenvalue λ_i of $A(\delta)$ is called active at δ if $\mathrm{Re}(\lambda_i) = \alpha(A(\delta))$. We use $I(\delta)$ for the indices of active eigenvalues. Let us write the LFT describing $A(\delta)$ as $A(\delta) = \mathcal{A} + \mathcal{C}\Delta(I - \mathcal{D}\Delta)^{-1}\mathcal{B}$, where dependence on controller parameters κ is again omitted and considered absorbed into the state-space data $\mathcal{A}$, $\mathcal{B}$, etc.

Proposition 3. *Suppose all active eigenvalues λ_i, $i \in I(\delta)$ of $A(\delta)$ at δ are semi-simple. Then $a_\pm(\delta) = \pm\alpha(A(\delta))$ is Clarke subdifferentiable in a neighborhood of δ. The Clarke subdifferential of a_- at δ is $\partial a_-(\delta) = \{\phi_Y : Y = (Y_i)_{i \in I(\delta)}, Y_i \succeq 0, \sum_{i \in I(\delta)} \mathrm{Trace}(Y_i) = 1\}$, where the i-th entry of ϕ_Y is $-\mathrm{Trace}\, {\Delta_i}^T \Phi_Y$ with $\Delta_i = \partial\Delta/\partial\delta_i$, and*

$$\Phi_Y = \sum_{i \in I(\delta)} \mathrm{Re}\left((I - \mathcal{D}\Delta)^{-1}\mathcal{C} V_i Y_i U_i^H \mathcal{B}(I - \Delta\mathcal{D})^{-1}\right)^T.$$

Here V_i is a column matrix of right eigenvectors, U_i^H a row matrix of left eigenvectors of $A(\delta)$ associated with the eigenvalue λ_i, and such that $U_i^H V_i = I$.

Proof: This follows from [31]. See also [32]. A very concise proof that semi-simple eigenvalue functions are locally Lipschitz could also be found in [33]. ∎

When every active eigenvalue is simple, Y_i reduces to a scalar y_i and a fast implementation is possible. We use the LU-decomposition to solve for $\widetilde{u}_i$ and $\widetilde{v}_i$ in the linear systems

$$\widetilde{u}_i^H (I - \Delta\mathcal{D}) := u_i^H \mathcal{B},\ (I - \mathcal{D}\Delta)\widetilde{v}_i := \mathcal{C} v_i.$$

Given the particular structure (2) of Δ, subgradients with respect to the kth entry are readily obtained as a sum over $i \in I(\delta)$ of inner products of the form $y_i \mathrm{Re}\, \widetilde{u}_i(J(k))^H \widetilde{v}_i(J(k))$, where $J(k)$ is a selection of indices associated with the rows/columns of δ_k in $\Delta(\delta)$. Similar inner products arise in the computation of H_∞ norm subgradients.

It was observed in [31] that $a_\pm$ may fail to be locally Lipschitz at δ if $A(\delta)$ has a derogatory active eigenvalue.

Proposition 4. *Suppose every active eigenvalue of $A(\delta)$ is simple. Then a_- is upper-C^1 in a neighborhood of δ.*

Proof: If active eigenvalues are simple, then a_+ is the maximum of C^1 functions in a neighborhood of δ. The result follows from $a_- = -a_+$. ∎

V. Algorithm for min-min programs

In this section we present our descent algorithm to solve programs (7) and (8). We consider an abstract form of the min-min program with f a general objective function of this type:

$$\begin{array}{ll} \text{minimize} & f(\delta) \\ \text{subject to} & \delta \in \mathbf{\Delta} \end{array} \quad (17)$$

As we already pointed out, the crucial point is that we want to stay as close as possible to a standard algorithm for smooth optimization, while assuring convergence under the specific form of upper nonsmoothness in these programs.

Algorithm 2. Descent method for min-min programs

Parameters: $0 < \gamma < \Gamma < 1$, $0 < \theta < \Theta < 1$.

▷ **Step 1** (`Initialize`)**.** Put outer loop counter $j = 1$, choose initial guess $\delta^1 \in \mathbf{\Delta}$, fix memory stepsize $t_1^\sharp > 0$.

◇ **Step 2** (`Stopping`)**.** If δ^j is a Karush-Kuhn-Tucker point of (17) then exit, otherwise go to inner loop.

▷ **Step 3** (`Inner loop`)**.** At current iterate δ^j call the step finding Subroutine (Subroutine 1) started with last memorized stepsize $t_j^\sharp$ to find a step $t_k > 0$ and a new serious iterate δ^{j+1} such that

$$\rho_k = \frac{f(\delta^j) - f(\delta^{j+1})}{f(\delta^j) - \phi_k^\sharp(\delta^{j+1}, \delta^j)} \geq \gamma.$$

◇ **Step 4** (`Stepsize update`)**.** If $\rho_k \geq \Gamma$ then update memory stepsize as $t_{j+1}^\sharp = \theta^{-1} t_k$, otherwise update memory stepsize as $t_{j+1}^\sharp = t_k$. Increase counter j and go back to step 2.

In order to understand Algorithm 2 and its step finding subroutine (Subroutine 1), we recall from [34], [21] that

$$\phi^\sharp(\eta, \delta) = f(\delta) + f^\circ(\delta, \eta - \delta)$$

is the standard model of f at δ, where $f^\circ(\delta, d)$ is the Clarke directional derivative of f at δ in direction d [18]. This model can be thought of as a substitute for a first-order Taylor expansion at δ and can also be represented as

$$\phi^\sharp(\eta, \delta) = f(\delta) + \max_{g \in \partial f(\delta)} g^T(\eta - \delta), \quad (18)$$

where $\partial f(\delta)$ is the Clarke subdifferential of f at δ. In the subroutine we generate lower approximations $\phi_k^\sharp$ of $\phi^\sharp$ using finite subsets $\mathcal{G}_k \subset \partial f(\delta)$, putting

$$\phi_k^\sharp(\eta, \delta) = f(\delta) + \max_{g \in \mathcal{G}_k} g^T(\eta - \delta).$$

We call $\phi_k^\sharp$ the working model at inner loop counter k.

Subroutine 1. Descent step finding for min-min programs

Input: Current serious iterate δ, last memorized stepsize $t^\sharp > 0$. Flag.

Output: Next serious iterate δ^+.

▷ **Step 1** (`Initialize`)**.** Put linesearch counter $k = 1$, and initialize search at $t_1 = t^\sharp$. Choose subgradient $g_0 \in \partial f(\delta)$. Put $\mathcal{G}_1 = \{g_0\}$.

▷ **Step 2** (`Tangent program`)**.** Given $t_k > 0$, a finite set of Clarke subgradients $\mathcal{G}_k \subset \partial f(\delta)$, and the corresponding working model $\phi_k^\sharp(\cdot,\delta) = f(\delta) + \max_{g\in\mathcal{G}_k} g^T(\cdot-\delta)$, compute solution $\eta^k \in \mathbf{\Delta}$ of the convex quadratic tangent program

$$\text{(TP)} \qquad \min_{\eta\in\mathbf{\Delta}} \phi_k^\sharp(\eta,\delta) + \tfrac{1}{2t_k}\|\eta-\delta\|^2.$$

◇ **Step 3** (`Armijo test`)**.** Compute

$$\rho_k = \frac{f(\delta) - f(\eta^k)}{f(\delta) - \phi_k^\sharp(\eta^k,\delta)}$$

If $\rho_k \geq \gamma$ then return $\delta^+ = \eta^k$ successfully to Algorithm 2. Otherwise go to step 4

▷ **Step 4** (`If Flag = strict. Cutting and aggregate plane`)**.** Pick a subgradient $g_k \in \partial f(\delta)$ such that $f(\delta) + g_k^T(\eta^k - \delta) = \phi^\sharp(\eta^k,\delta)$, or equivalently, $f^\circ(\delta, \eta^k - \delta) = g_k^T(\eta^k - \delta)$. Include g_k into the new set $\mathcal{G}_{k+1}$ for the next sweep. Add the aggregate subgradient g_k^* into the set $\mathcal{G}_{k+1}$ to limit its size.

◇ **Step 5** (`Step management`)**.** Compute the test quotient

$$\widetilde{\rho}_k = \frac{f(\delta) - \phi_{k+1}^\sharp(\eta^k,\delta)}{f(\delta) - \phi_k^\sharp(\eta^k,\delta)}.$$

If $\widetilde{\rho}_k \geq \widetilde{\gamma}$ then select $t_{k+1} \in [\theta t_k, \Theta t_k]$, else keep $t_{k+1} = t_k$. Increase counter k and go back to step 2.

Remark 6. Typical values are $\gamma = 0.0001$, $\widetilde{\gamma} = 0.0002$, and $\Gamma = 0.1$. For backtracking we use $\theta = \frac{1}{4}$ and $\Theta = \frac{3}{4}$.

A. Practical aspects of Algorithm 2

The subroutine of the descent Algorithm 2 looks complicated due to step 4, but as we now argue, it reduces to a standard backtracking linesearch in the majority of cases. Note that if f is certified upper-C^1, then we completely dispense with step 4 and keep $\mathcal{G}_k = \{g_0\}$, which by force reduces the subroutine to a linesearch along a projected gradient direction. This is what we indicate by `flag = upper` in step 4 of the subroutine.

If f is known to have a strict standard model $\phi^\sharp$ in (18), without being certified upper-C^1, which corresponds to `flag = strict`, then step 4 of the subroutine is in principle needed. However, even then we expect the subroutine to reduce to a standard linesearch. This is clearly the case when the Clarke subdifferential $\partial f(\delta)$ at the current iterate δ is singleton, because $\phi_k^\sharp(\eta,\delta) = f(\delta) + \nabla f(x)^T(\eta-\delta)$ is then independent of k, so $\rho_k \geq \gamma$ reads

$$f(\eta^k) \leq f(\delta^j) + \gamma \nabla f(\delta^j)^T(\eta^k - \delta^j),$$

which is the usual Armijo test [26]. Moreover, η^k is then a step along the projected gradient $P_{\mathbf{\Delta}-\delta}(-\nabla f(\delta))$, which is easy to compute due to the simple structure of $\mathbf{\Delta}$. More precisely, for $\mathbf{\Delta} = [-1,1]^m$ and stepsize $t_k > 0$, the solution η of tangent program (TP) in step 2 can be computed coordinatewise as

$$\min\left\{\gamma_i\eta + (2t_k)^{-1}\eta^2 + \left(\gamma_i - \delta_i t_k^{-1}\right)\eta : -1 \leq \eta \leq 1\right\},$$

where $\gamma_i := \partial f(\delta)/\partial\delta_i$. Cutting plane and aggregate plane in step 4 become redundant, and the quotient $\widetilde{\rho}_k$ in step 5 is also redundant as it is always equal to 1.

Remark 7. There is only one case in which step 4 is fully executed, and that is when f is *not* certified upper-C^1, and in addition the subgradient $g_0 \in \partial f(\delta)$ in step 1 of Subroutine 1 does *not* satisfy $f(\delta) + g_0^T(\eta^k - \delta) = \phi^\sharp(\eta^k,\delta)$. In that rare event step 4 requires computation of a new subgradient $g_k \in \partial f(\delta)$ which *does* satisfy $f(\delta) + g_k^T(\eta^k - \delta) = \phi^\sharp(\delta, \eta^k - \delta)$. From here on the procedure changes. The sets $\mathcal{G}_{k+1}$ may now grow, because we add g_k into $\mathcal{G}_{k+1}$. This corresponds to what happens in a bundle method. The tangent program (TP) is now solved numerically using a QP-solver. Fortunately we may limit the number of elements of $\mathcal{G}_{k+1}$ using the aggregate subgradient of Kiwiel [35], so this is still very fast.

Remark 8. For the spectral abscissa $f(\delta) = a_-(\delta)$, which is not certified upper-C^1, we use this cautious variant, where the computation of g_k in step 4 may be required. For $f = a_-$ this leads to a low-dimensional semidefinite program.

Remark 9. The stopping test in step 2 of Algorithm 2 can be delegated to Subroutine 1. Namely, if δ^j is a Karush-Kuhn-Tucker point of (17), then $\eta^k = \delta^j$ is solution of the tangent program (TP). This means we can use the following practical stopping tests: If the inner loop at iterate δ^j finds $\delta^{j+1} \in \mathbf{\Delta}$ such that

$$\frac{\|\delta^{j+1} - \delta^j\|}{1 + \|\delta^j\|} < \mathrm{tol}_1, \quad \frac{|f(\delta^{j+1}) - f(\delta^j)|}{1 + |f(\delta^j)|} < \mathrm{tol}_2,$$

then we decide that δ^{j+1} is optimal and stop. That is, the $(j+1)$st inner loop is not started. On the other hand, if the inner loop at δ^j has difficulties finding a new iterate and provides five consecutive unsuccessful backtracks η^k such that

$$\frac{\|\eta^k - \delta^j\|}{1 + \|\delta^j\|} < \mathrm{tol}_1, \quad \frac{|f(\eta^k) - f(\delta^j)|}{1 + |f(\delta^j)|} < \mathrm{tol}_2,$$

or if a maximum $k_{\max}$ of linesearch steps k is exceeded, then we decide that δ^j was already optimal and stop. In our experiments we use $\mathrm{tol}_1 = 10^{-4}$, $\mathrm{tol}_2 = 10^{-4}$, $k_{\max} = 50$.

B. Convergence analysis for the negative H_∞-norm

Algorithm 2 was studied in much detail in [34], and we review the convergence result here, applying them directly to the functions a_- and h_-. The significance of the class of upper-C^1 functions for convergence lies in the following

Proposition 5. *Suppose f is upper-C^1 at $\bar{\delta}$. Then its standard model $\phi^\sharp$ is strict at $\bar{\delta}$ in the following sense: For every $\varepsilon > 0$ there exists $r > 0$ such that the one-sided Taylor type estimate*

$$f(\eta) \leq \phi^\sharp(\eta, \delta) + \varepsilon\|\eta - \delta\| \tag{19}$$

is satisfied for all $\delta, \eta \in B(\bar{\delta}, r)$.

Proof: The following, even stronger property of upper-C^1 functions was proved in [36], see also [37], [34]. Suppose $\delta^k \to \bar{\delta}$ and $\eta^k \to \bar{\delta}$, and let $g_k \in \partial f(\delta_k)$ be arbitrary. Then there exist $\varepsilon_k \to 0$ such that

$$f(\eta^k) \leq f(\delta^k) + g_k^T(\eta^k - \delta^k) + \varepsilon_k\|\eta^k - \delta^k\| \tag{20}$$

is satisfied. ∎

Theorem 1 (Worst-case H_∞ norm on $\boldsymbol{\Delta}$)**.** *Let $\delta^j \in \boldsymbol{\Delta}$ be the sequence generated by Algorithm* 2 *with standard linesearch for minimizing program* (8)*. Then the sequence δ^j converges to a Karush-Kuhn-Tucker point δ^* of* (8)*.*

Proof: The method of proof of [21, Theorem 6.6] shows that every accumulation point of the $\delta^j \in \boldsymbol{\Delta}$ is a critical point of (8), because of the one-sided Taylor estimate (19). However, the proof in [21] still needs the extended version of Subroutine 1, which we want to avoid.

Now, using the stronger hypothesis that f is upper-C^1, we find indeed that the full construction of the step finding subroutine is not needed, because regardless how the cutting planes are chosen, the working models always satisfy a one-sided Taylor estimate $f(\eta^k) \leq \phi_k^\sharp(\eta^k, \delta^k) + \varepsilon_k\|\eta^k - \delta^k\|$ with $\varepsilon_k \to 0$ as $\eta^k, \delta^k \to \delta$. This freedom allows us to simplify the step finding Subroutine 1 to a standard linesearch, while maintaining the convergence argument of [21]. More details are given in [36].

Since h_- is upper-C^1 on $\mathbb{D}$ by Proposition 2, the above applies to h_-. Note that hypotheses assuring boundedness of the sequence δ^j used in [21], [34], [36] are not needed here, since $\boldsymbol{\Delta}$ is bounded.

Convergence to a single critical point is now assured through [34, Cor. 1], because G in (8) depends analytically on δ, so h_- is a subanalytic function, and hence satisfies the Łojasiewicz inequality [38]. Subanalyticity of h_- can be derived from the following fact [39]: If $F : \mathbb{R}^n \times \mathbb{K} \to \mathbb{R}$ is subanalytic, and $\mathbb{K}$ is subanalytic and compact, then $f(\delta) = \min_{y\in\mathbb{K}} F(\delta, y)$ is subanalytic. We apply this to the negative of (16). ∎

Remark 10. The lightning function $f : \mathbb{R} \to \mathbb{R}$ in [40] is an example which has a strict standard model but is not upper-C^1. It is Lipschitz with constant 1 and has $\partial f(x) = [-1, 1]$ for every x. The standard model of f is strict, because for all x, y there exists $\rho = \rho(x, y) \in [-1, 1]$ such that

$$\begin{aligned} f(y) = f(x) + \rho|y - x| &\leq f(x) + \operatorname{sign}(y - x)(y - x) \\ &\leq f(x) + f^\circ(x, y - x) = \phi^\sharp(x, y - x), \end{aligned}$$

using the fact that $\operatorname{sign}(y-x) \in \partial f(x)$. At the same time f is certainly not upper-C^1, because it is not semi-smooth in the sense of [41]. This shows that the class of functions f with a strict standard model offers a scope of its own, justifying the effort made in the step finding subroutine.

C. Convergence analysis for the negative spectral abscissa

While we obtained an ironclad convergence certificate for the H_∞-programs (8), and similarly, for (12), theory is more complicated with program (7). In our numerical testing $a_-(\delta) = -\alpha(A(\delta))$ behaves consistently like an upper-C^1 function, and we expect this to be true at least if all active eigenvalues of $A(\delta^*)$ are semi-simple. We now show that a_- has a strict standard model as a rule.

Since $A(\delta)$ depends analytically on δ, the eigenvalues are roots of a characteristic polynomial $p_\delta(\lambda) = \lambda^m + a_1(\delta)\lambda^{m-1} + \cdots + a_m(\delta)$ with coefficients $a_i(\delta)$ depending analytically on δ. For fixed $d \in \mathbb{R}^m$, every eigenvalue $\lambda_\nu(t)$ of $A(\delta^* + td)$ has therefore a Newton-Puiseux expansion of the form

$$\lambda_\nu(t) = \lambda_\nu(0) + \sum_{i=k}^{\infty} \lambda_{\nu,i-k+1} t^{i/p} \tag{21}$$

for certain $k, p \in \mathbb{N}$, where the coefficients $\lambda_{\nu,i} = \lambda_{\nu,i}(d)$ and leading exponent k/p can be determined by the Newton polygon [42]. If all active eigenvalues of $a_-(\delta) = -\alpha(A(\delta))$ are semi-simple, then a_- is Lipschitz around δ^* by Proposition 3, so that necessarily $k/p \geq 1$ in (21). It then follows that either $a'_-(\delta^*, d) = 0$ when $k/p > 1$ for all active ν, or $a'_-(\delta^*, d) = -\operatorname{Re}\lambda_{\nu,1} \leq a^\circ_-(\delta^*, d)$ for the active $\nu \in I(\delta^*)$ if $k/p = 1$. In both cases a_- satisfies the strictness estimate (19) *directionally*, and we expect a_- to have a strict standard model. Indeed, for $k/p = 1$ we have $a_-(\delta^* + td) \leq a_-(\delta^*) + a^\circ_-(\delta^*, d)t - \operatorname{Re}\lambda_{\nu,2} t^{(p+1)/p} + \mathrm{o}(t^{(p+1)/p})$, while the case $k/p > 1$ gives $a'_-(\delta^*, d) = 0$, hence $a^\circ_-(\delta^*, d) \geq 0$, and so $a_-(\delta^* + td) \leq a_-(\delta^*) - \operatorname{Re}\lambda_{\nu,1} t^{k/p} + \mathrm{o}(t^{k/p}) \leq a_-(\delta^*) + a^\circ_-(\delta^*, d)t - \operatorname{Re}\lambda_{\nu,1} t^{k/p} + \mathrm{o}(t^{k/p})$. As soon as these estimates hold uniformly over $\|d\| \leq 1$, a_- has indeed a strict standard model, i.e., we have the following

Lemma 1. *Suppose every active eigenvalue of $A(\delta^*)$ is semi-simple, and suppose the following two conditions are satisfied:*

$$\begin{aligned} &\lim_{t\to 0} \sup_{\|d\|\leq 1} \sup_{\nu\in I(\delta^*), k/p=1} \sum_{i=k+1}^{\infty} \operatorname{Re}\lambda_{\nu,i-k+1}(d) t^{i/p-1} \geq 0 \\ &\lim_{t\to 0} \sup_{\|d\|\leq 1} \sup_{\nu\in I(\delta^*), k/p>1} \sum_{i=k}^{\infty} \operatorname{Re}\lambda_{\nu,i-k+1}(d) t^{i/p-1} \geq 0. \end{aligned} \tag{22}$$

Then the standard model of a_- is strict at δ^*, i.e., a one-sided Taylor estimate of the form (19) *is satisfied.* □

Even though these conditions are not easy to check, they seem to be verified most of the time, so that the following result reflects what we observe in practice for the min-min program of the negative spectral abscissa a_-.

Theorem 2 (Worst-case spectral abscissa on $\boldsymbol{\Delta}$)**.** *Let $\delta^j \in \boldsymbol{\Delta}$ be the sequence generated by Algorithm* 2 *for program* (7)*, where the step finding subroutine is carried out with step* 4 *activated. Suppose every accumulation point δ^* of the sequence δ^j is simple or semi-simple and satisfies condition* (22)*. Then the sequence converges to a unique Karush-Kuhn-Tucker point of program* (7)*.*

Proof: We apply [34, Cor. 1], using that a_- satisfies the Łojasiewicz inequality at all accumulation points. ∎

Remark 11. The results of this section give a fair theoretical explanation why the step finding subroutine reduces to a standard linesearch in almost all cases. Indeed, as $\partial a_-(\delta)$ is singleton almost everywhere in the neighborhood of a semi-simple eigenvalue due to Proposition 3 and Rademacher's theorem, step 4 becomes redundant almost everywhere.

D. Multiple performance measures

Practical applications often feature several design requirements combining H_∞ and H_2 performances with spectral constraints related to pole locations. The results in Section V-B easily extend to this case upon defining $H(\kappa,\delta) := \max_{i\in I} h_i\left(T_{z_i,w_i}(\kappa,\delta)\right)$, where several performance channels $w_i \to z_i$ are assessed against various requirements h_i, as in [22], [8]. All results developed so far carry over to multiple requirements, because the worst-case multi-objective performance in step 4 of Algorithm 1 involves $H_- = -H$ which has the same min-min structure as before.

VI. Experiments

A. Algorithm testing

In this section our dynamic inner approximation technique (Algorithm 1) is tested on a bench of 14 examples of various sizes and structures. All tests have been adapted from the literature and are described in Table I. Some tests have been made more challenging by adding uncertain parameters in order to illustrate the potential of the technique for higher-dimensional parametric domains $\mathbf{\Delta}$. The notation $[r_1\ r_2\ \ldots\ r_m]$ in the rightmost column of the table stands for the block sizes in $\Delta = \mathrm{diag}\,[\delta_1 I_{r_1},\ldots,\delta_m I_{r_m}]$. Uncertain parameters have been normalized so that $\mathbf{\Delta} = [-1,1]^m$, with nominal value $\delta = 0$.

The dynamic inner approximation of Algorithm 1 is first compared to static inner approximation (6), which uses a dense enough static grid $\mathbf{\Delta}_s$ of $\mathbf{\Delta}$ to perform a multi-model synthesis for a large number $\mathrm{card}(\mathbf{\Delta_s})$ of models where card stands for set cardinality. In consequence, static approximation cannot be considered practical. Namely,

- Dense grids become quickly intractable for high-dimensional $\mathbf{\Delta}$.
- Static approximation may lead to overly optimistic answers in terms of worst-case performance if critical parametric configurations are missed by gridding.

This is what is observed in columns 7 - 9 of Table II, where we used a 5^m-point grid with $m = \dim(\delta)$ the number of uncertain parameters. Worst-case performance is missed in tests 6, 9, 12 and 14, as we verified by Algorithm 1. Running times may rise to hours or even days for cases 1, 2, 5 and 10. On the other hand, when the worst case δ^* found by Algorithm 1 is close to the grid $\mathbf{\Delta}_s$, then static approximation and Algorithm 1 find the same result. Dynamic inner approximation (Algorithm 1) can therefore be considered a cheap, and therefore very successful, way to cover the uncertainty box. The number of scenarios in $\mathbf{\Delta}_a$ rarely exceeds 10 in our testing. Computations were performed with MATLAB R2013b on OS Windows 7 Home Premium with CPU Intel Core i5-2410M running at 2.30 Ghz and 4 GB of RAM.

The results h_∞ achieved by Algorithm 1 shown in column 4 of Table II are underestimates of the worst-case H_∞ performance on the unit cube. We therefore certify them *a posteriori* through the mixed μ upper bound [13], which can be computed using the routine `wcgain` of [12]. This is Step 6 in Algorithm 1, which gives the final validation of the design. This gives an overestimate $\bar\gamma_{\mathrm{A1}}$ of the worst-case performance on the unit cube $\mathbf{\Delta}$, which we report in column 5 of Table II. Since $h_\infty \le \mu \le \bar\gamma_{\mathrm{A1}}$, we know that as soon as lower and upper worst-case estimates nearly coalesce, $h_\infty \approx \bar\gamma_{\mathrm{A1}}$, controllers designed using Algorithm 1 are certified on the unit cube.

Our last comparison is between Algorithm 1 and `DKSYN` for complex- and mixed-μ syntheses. To put comparison on an equal basis, we scale the performance channels by $1/h_\infty$, where h_∞ are the values computed via Algorithm 1 (column 4 of Table II). If we now validate our method using `wcgain` as before, this gives the values $\bar\gamma_{\mathrm{A1}} \ge 1$ reported in column 2 of Table III. This allows to compare results to one and to decide whether a technique improves over Algorithm 1.

For the purpose of comparison we now run complex- and mixed-μ syntheses for solving

$$\min_{K(s)} \max_{\delta\in\mathbf{\Delta}} \frac{1}{h_\infty} \|T_{zw}\left(\delta, K(s)\right)\|_\infty. \tag{23}$$

The optimal controllers $K(s)$ are then again analyzed in closed-loop via `wcgain`, which gives the estimates $\bar\gamma_{\mathbb{C}}$ and $\bar\gamma_{\mathbb{R}}$ reported in columns 3 and 4 of Table III.

Table III shows that all synthesis techniques perform equally well in tests 4 and 6, as indicated by $\bar\gamma_{\mathrm{A1}} \approx \bar\gamma_{\mathbb{C}} \approx \bar\gamma_{\mathbb{R}}$. Mixed-$\mu$ synthesis performs better in case 8, but with a very high order controller, 61 states versus a simple PID for Algorithm 1. In all other cases, Algorithm 1 achieves better worst-case performance $\bar\gamma_{\mathrm{A1}} \ll \bar\gamma_{\mathbb{C}}, \bar\gamma_{\mathbb{R}}$ with simpler structured controllers $K(\kappa)$. It also proves competitive in terms of CPU.

Going a step further, it can be shown using the min-min program in Step 4 in Algorithm 1, or alternatively the routine `wcgain`, that the upper estimates $\bar\gamma_{\mathrm{A1}}$, $\bar\gamma_{\mathbb{C}}$ and $\bar\gamma_{\mathbb{R}}$ are tight. We therefore conclude that the conservatism observed in μ synthesis is not an artifact of the analysis routine, but reflects the conservatism of the synthesis approach. The results of Tables II and III can be checked by downloading the entire benchmark at `http://pierre.apkarian.free.fr/BENCH.zip`

TABLE I: Test cases

No.	Benchmark name	Ref.	States	Uncertainty block structure
1	Flexible Beam	[43]	8	[1 1 1 3 1]
2	Mass-Spring-Dashpot	[44]	12	[1 1 1 1 1 1]
3	DC Motor	[45]	5	[1 2 2]
4	DVD Drive	[46]	5	[1 3 3 3 1 3]
5	Four Disk	[47]	10	[1 3 3 3 3 3 1 1 1 1]
6	Four Tank	[48]	6	[1 1 1 1]
7	Hard Disk Drive	[49]	18	[1 1 1 2 2 2 2 1 1 1 1]
8	Hydraulic Servo	[50]	7	[1 1 1 1 1 1 1 1]
9	Mass-Spring System	[51]	4	[1 1]
10	Tail Fin Controlled Missile	[52]	23	[1 1 1 6 6 6]
11	Robust Filter Design 1	[53]	4	[1]
12	Robust Filter Design 2	[54]	2	[1 1]
13	Satellite	[55]	5	[1 6 1]
14	Mass-Spring-Damper	[12]	8	[1]

TABLE II: Comparisons of Algorithm 1 with static approximation on unit box, running times in sec.

No.	order	Algorithm 1				Static relaxation		
		# scen.	h_∞	$\bar\gamma_{\rm A1}$	time	# scenarios	H_∞ norm	time
1	3	4	1.290	1.290	25	3125	I	∞
2	5	16	2.960	2.990	261	15625	I	∞
3	PID	2	0.500	0.510	6	125	0.500	128
4	5	1	45.455	45.455	2	15625	45.454	4909
5	6	6	0.682	0.682	68	9765625	I	∞
6	6	4	5.568	5.568	42	625	5.564	3872
7	4	4	0.026	0.026	35	48828125	I	∞
8	PID	3	0.701	0.708	10	390625	I	∞
9	4	4	0.745	0.745	23	25	0.759	67
10	12	6	1.810	1.828	159	15625	I	∞
11	4	4	2.636	2.636	17	5	2.636	7
12	1	3	2.793	2.793	8	25	2.660	23
13	6	5	0.154	0.154	48	125	0.156	876
14	5	3	1.643	1.643	39	5	1.644	27

I = intractable

TABLE III: Comparisons of Algorithm 1 (A1) & DKSYN complex- ($\mathbb{C}$) and mixed-μ ($\mathbb{R}$)
Worst-case H_∞ performance on unit box computed using μ upper bound

No.	worst-case H_∞			Controller order			Running times		
	$\bar\gamma_{\rm A1}$	$\bar\gamma_{\mathbb{C}}$	$\bar\gamma_{\mathbb{R}}$	A1	$\mathbb{C}$	$\mathbb{R}$	A1	$\mathbb{C}$	$\mathbb{R}$
1	1.00	1.36	U	3	52	102	25	80	86
2	1.01	1.56	1.22	5	42	82	261	123	141
3	1.02	24.86	26.10	PID	47	47	6	76	27
4	1.00	1.00	1.00	5	5	5	2	27	53
5	1.00	4.04	3.40	6	54	10	68	131	316
6	1.00	1.00	1.00	6	6	22	42	18	29
7	1.01	63.65	F	4	18	F	35	159	F
8	1.01	1.20	0.95	PID	45	47	10	101	134
9	1.00	1.10	1.21	4	24	28	23	48	113
10	1.01	2.83	2.05	12	149	385	159	1412	7612
11	1.00	1.04	1.49	4	14	16	17	14	22
12	1.00	1.02	1.07	1	10	14	8	16	21
13	1.00	2.76	8.41	6	269	259	48	183	257
14	1.00	1.32	1.27	5	14	16	39	17	47

U: unstable, F: failure, times in seconds

B. Tail fin controlled missile

We now illustrate our robust synthesis technique in more depth for a tail fin controlled missile. This problem is adapted from [52, Chapter IV] and has been made more challenging by adding parametric uncertainties in the most critical parameters. The linearized rigid body dynamics of the missile are

$$\begin{bmatrix}\dot\alpha\\ \dot q\end{bmatrix} = \begin{bmatrix}Z_\alpha & 1\\ M_\alpha & M_q\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}Z_d\\ M_d\end{bmatrix}u$$
$$\begin{bmatrix}\eta\\ q\end{bmatrix} = \begin{bmatrix}V/kG\,Z_\alpha & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}V/kG\,Z_d\\ 0\end{bmatrix}u$$

where α is the angle of attack, q the pitch rate, η the vertical acceleration and u the fin deflection. Both η and q are measured through appropriate devices as described below. A more realistic model also includes bending modes of the missile structure. In this application, we have 3 bending modes whose contribution to η and q is additive and described as:

$$\begin{bmatrix}\eta_i(s)\\ q_i(s)\end{bmatrix} = \frac{1}{s^2+2\zeta\omega_i s+\omega_i^2}\begin{bmatrix}s^2\Xi_{\eta_i}\\ s\Xi_{q_i}\end{bmatrix},\ i=1,2,3\,.$$

It is also important to account for actuator and detector dynamics. The actuator is modeled as a 2nd-order transfer function with damping 0.7 and natural frequency 188.5 rad./sec. Similarly, the accelerometer and pitch rate gyrometer are 2nd-order transfer functions with damping 0.7 and natural frequencies 377 rad./sec. and 500 rad./sec., respectively.

Uncertainties affect both rigid and flexible dynamics and the deviations from nominal are 30% for Z_α, 15% for M_α, 30% for M_q, and 10% for each ω_i. This leads to an uncertain model with uncertainty structure given as

$$\Delta = \mathrm{diag}\left[\delta_{Z_\alpha}, \delta_{M_\alpha}, \delta_{M_q}, \delta_{\omega_1} I_6, \delta_{\omega_2} I_6, \delta_{\omega_3} I_6\right]\,,$$

which corresponds to $\delta\in\mathbb{R}^6$ and repetitions $[1\ 1\ 1\ 6\ 6\ 6]$ in the terminology of Table I. The controller structure includes both feed-forward $K_{\rm ff}(s)$ and feedback $K_{\rm fb}(s)$ actions

$$u_c = K_{\rm ff}(s)\eta_r + K_{\rm fb}(s)\begin{bmatrix}\eta_r-\eta_m\\ -q_m\end{bmatrix} = K(s)\begin{bmatrix}\eta_r-\eta_m\\ q_m\\ \eta_r\end{bmatrix},$$

where η_r is the acceleration set-point and η_m, q_m are the detectors outputs. The total number of design parameters κ in $K(\kappa,s)$ is 85, as a tridiagonal state space representation of a 12-th order controller was used.

The missile autopilot is optimized over $\kappa\in\mathbb{R}^{85}$ to meet the following requirements:

- The acceleration η_m should track the reference input η_r with a rise time of about 0.5 seconds. In terms of the transfer function from η_r to the tracking error $e:=\eta_r-\eta_m$ this is expressed as $||W_e(s)T_{e\eta_r}||_\infty\le 1$, where the weighting function $W_e(s)$ is
$$W_e(s) := 1/M\frac{s/\omega_B+M}{s/\omega_B+A},\ A=0.05,\ M=1.5,\ \omega_B=10\,.$$
- Penalization of the high-frequency rate of variation of the control signal and roll-off are captured through the constraint $||W_u(s)T_{u\eta_r}||_\infty\le 1$, where $W_u(s)$ is a high-pass weighting $W_u(s):=(s/100(0.001s+1))^2$.
- Stability margins at the plant input are specified through the H_∞ constraint $\|W_o(s)S(s)W_i(s)\|_\infty\le 1$, where S is the input sensitivity function $S:=(I+K_{\rm fb}G)^{-1}$ and with static weights $W_o=W_i=0.4$.

Finally, stability and performance requirements must hold for the entire range of parametric uncertainties, where $\mathbf{\Delta}$ is the $\mathbb{R}^6$-hyperbox with limits in percentage given above. The resulting nonsmooth program v^* to be solved in step 2 of Algorithm 1 takes the form

$$\min_{\kappa\in\mathbb{R}^{85}}\ \max_{\delta\in\mathbf{\Delta}_a\subset\mathbb{R}^6}\ \|T_{zw}(\delta,\kappa)\|_\infty.$$

We have observed experimentally that controllers $K(s)$ of order greater than 12 do not improve much. The order of the augmented plant including flexible modes, detector and actuator dynamics, and weighting filters is $n_x=23$.

The evolution of the worst-case H_∞ performance vs. iterations in Algorithm 2 (and its Subroutine 1) is problem-dependent. For the missile example, a destabilizing uncertainty is found at the 1st iteration. The algorithm then settles very quickly in 5 iterations on a final set $\mathbf{\Delta}_a$ consisting of 6 scenarios. The number of scenarios in the final $\mathbf{\Delta}_a$ coincides with the number of iterations in Algorithm 1 plus the nominal

scenario, and can be seen in column 3 of Table II. Note that the evolution of the worst-case H_∞ performance is not always monotonic. Typically the curve may bounce back when a bad parametric configuration δ is discovered by the algorithm. This is the case e.g. for the mass-spring example.

The achieved values of the H_∞ norm and corresponding running times are given in Table II. Responses to a step reference input for 100 models from the uncertainty set $\mathbf{\Delta}$ are shown in Fig. 3 to validate the robust design. Good tracking is obtained over the entire parameter range. The magnitude of the 3 controller gains of $K(s)$ is plotted in Fig. 4. Robust roll-off and notching of flexible modes are clearly achieved. Potential issues due to pole-zero cancellations are avoided as a consequence of allowing parameter variations in the model. Finally, Fig. 5 displays the Nichols plots for 100 models sampled in the uncertainty set. We observe that good "rigid" margins as well as attenuation of the flexible modes over $\mathbf{\Delta}$ has been achieved.

Remark 12. Mixed-μ synthesis turned out time-consuming, exceeding two hours in the missile example. The controller order inflates to 385 and conservatism is still present as compared to dynamic approximation via Algorithm 1 as shown in Table III. Resorting to interpreting uncertain parameters as complex cannot be considered an acceptable workaround. Even when it delivers a result, this approach as a rule leads to high-order controllers (149 states in the missile example) and is also conservative, as we expected. It appears that scaling- or multiplier-based approaches using outer approximations [56], [2] encounter difficulties in terms of conservatism and controller complexity for repeated parametric uncertainties whereas our approach is not affected by these issues.

Remark 13. Static inner approximation remains intractable even for a coarse grid of 5 points in each dimension. See Table II.

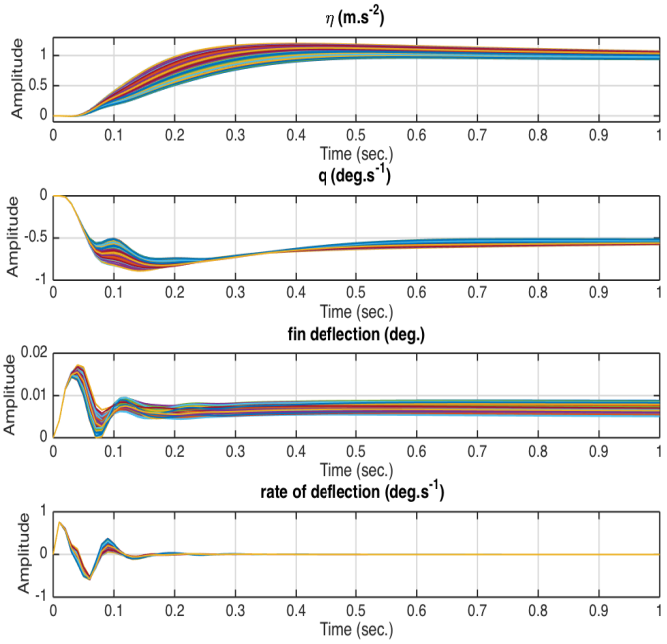

Fig. 3: Step responses of controlled missile for 100 sampled models in uncertainty range

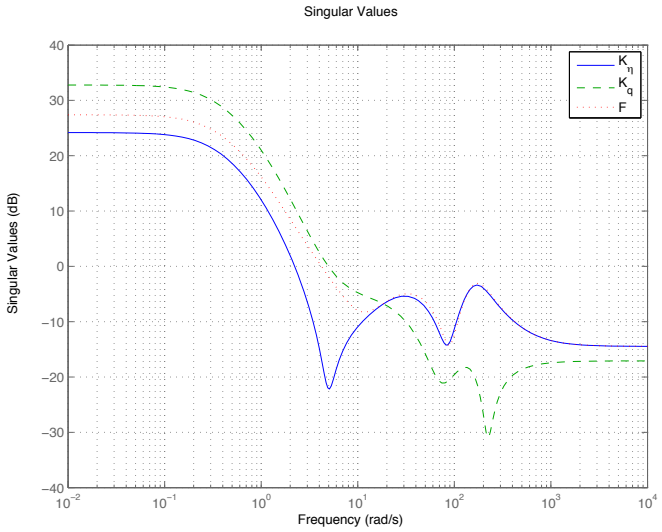

Fig. 4: Feedback and feed-forward gains

VII. Conclusion

We have presented a novel algorithmic approach to parametric robust H_∞ control with structured controllers. A new inner approximation technique termed *dynamic inner approximation*, adapting a set of parameter scenarios $\mathbf{\Delta}_a$ iteratively, was developed and shown to work rapidly without introducing conservatism. Global robustness and performance certificates are then best obtained *a posteriori* by applying analysis tools based on outer approximations. At the core our new method is leveraged by sophisticated nonsmooth optimization techniques tailored to the class of upper-C^1 stability and performance functions. The approach was tested on a bench of challenging examples, and within a case study. The results indicate that the proposed technique is a valid practical tool, capable of solving challenging design problems with parametric uncertainty. The new method discussed in this paper will be available in the R2015b release of MATLAB's Robust Control Toolbox.

References

[1] H. Özbay. O. Toker, "On the $\mathcal{NP}$-hardness of the purely complex μ computation, analysis/synthesis, and some related problems in multi-dimensional systems," in *Proc. American Control Conf.*, Seattle, June 1995, pp. 447–451.

[2] A. Packard, J. C. Doyle, and G. J. Balas, "Linear, multivariable robust control with a μ perspective," *J. Dyn. Sys., Meas., Control, Special Edition on Control*, vol. 115, no. 2b, pp. 426–438, June 1993.

[3] 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.

[4] E. Polak and Y. Wardi, "Nondifferentiable optimization algorithm for designing control systems having singular value inequalities," *Automatica–J. IFAC*, vol. 18, no. 3, pp. 267–283, 1982.

[5] P. Apkarian and D. Noll, "Nonsmooth H_∞ synthesis," *IEEE Trans. Automat. Control*, vol. 51, no. 1, pp. 71–86, January 2006.

[6] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton, "Stabilization via nonsmooth, nonconvex optimization," *IEEE Trans. Automat. Control*, vol. 51, no. 11, pp. 1760–1769, November 2006.

[7] P. Gahinet and P. Apkarian, "Automated tuning of gain-scheduled control systems," in *Proc. IEEE Conf. on Decision and Control*, Florence, December 2013, pp. 2740 – 2745.

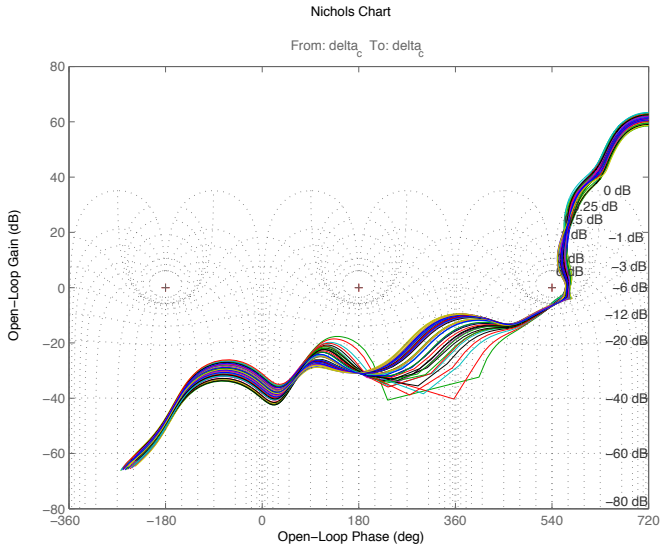

Fig. 5: Nichols plots for 100 sampled models in uncertainty range

[8] P. Apkarian, "Tuning controllers against multiple design requirements," in *Proc. American Control Conf.*, Washington, June 2013, pp. 3888 – 3893.

[9] P. Apkarian and D. Noll, "Optimization-based control design techniques and tools," in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. Springer-Verlag, 2015.

[10] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton, "HIFOO - A Matlab package for fixed-order controller design and H_∞ optimization," in *5th IFAC Symposium on Robust Control Design*, Toulouse, July 2006.

[11] P. Apkarian and D. Noll, "Nonsmooth optimization for multidisk H_∞ synthesis," *Eur. J. Control*, vol. 12, no. 3, pp. 229–244, May-June 2006.

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

[13] M. K. H. Fan, A. L. Tits, and J. C. Doyle, "Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics," *IEEE Trans. Automat. Control*, vol. 36, no. 1, pp. 25–38, 1991.

[14] V. Balakrishnan, "Linear matrix inequalities in robustness analysis with multipliers," *Systems Control Lett.*, vol. 25, no. 4, pp. 265–272, 1995.

[15] R. H. Nyström, K. V. Sandström, T. K. Gustafsson, and H. T. Toivonen, "Multimodel robust control of nonlinear plants: a case study," *J. Process Contr.*, vol. 9, no. 2, pp. 135–150, 1999.

[16] J.-F. Magni, Y. Le Gorrec, and C. Chiappa, "A multimodel-based approach to robust and self-scheduled control design," in *Proc. IEEE Conf. on Decision and Control*, vol. 3, 1998, pp. 3009–3014.

[17] J. Ackermann, A. Bartlett, D. Kaesbauer, W. Sienel, and R. Steinhauser, *Robust control. Systems with Uncertain Physical Parameters*, ser. Comm. Control Engrg. Ser. London: Springer-Verlag London, Ltd., 1993.

[18] F. H. Clarke, *Optimization and Nonsmooth Analysis*, ser. Canad. Math. Soc. Ser. Monogr. Adv. Texts. New York: John Wiley & Sons, Inc., 1983.

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

[20] R. M. Redheffer, "On a certain linear fractional transformation," *J. Math. and Phys.*, vol. 39, pp. 269–286, 1960.

[21] D. Noll, O. Prot, and A. Rondepierre, "A proximity control algorithm to minimize nonsmooth and nonconvex functions," *Pac. J. Optim.*, vol. 4, no. 3, pp. 571–604, 2008.

[22] P. Apkarian, P. Gahinet, and C. Buhr, "Multi-model, multi-objective tuning of fixed-structure controllers," in *European Control Conf. (ECC)*, Strasbourg, June 2014.

[23] J. E. Spingarn, "Submonotone subdifferentials of Lipschitz functions," *Trans. Amer. Math. Soc.*, vol. 264, no. 1, pp. 77–89, 1981.

[24] P. Apkarian, D. Noll, and O. Prot, "A proximity control algorithm to minimize nonsmooth and nonconvex semi-infinite maximum eigenvalue functions," *J. Convex Anal.*, vol. 16, no. 3-4, pp. 641–666, 2009.

[25] ——, "A trust region spectral bundle method for nonconvex eigenvalue optimization," *SIAM J. Optim.*, vol. 19, no. 1, pp. 281–306, 2008.

[26] D. P. Bertsekas, *Nonlinear Programming*. Belmont: Athena Scientific, 1999.

[27] S. Boyd, V. Balakrishnan, and P. Kabamba, "A bisection method for computing the $\mathbf{H}_\infty$ norm of a transfer matrix and related problems," *Math. Control Signals Systems*, vol. 2, no. 3, pp. 207–219, 1989.

[28] S. Boyd and V. Balakrishnan, "A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its $\mathbf{L}_\infty$-norm," *Systems Control Lett.*, vol. 15, no. 1, pp. 1–7, 1990.

[29] P. Benner, V. Sima, and M. Voigt, "$\mathcal{L}_\infty$-norm computation for continuous-time descriptor systems using structured matrix pencils," *IEEE Trans. Automat. Control*, vol. 57, no. 1, pp. 233–238, 2012.

[30] S. Boyd and C. Barratt, *Linear Controller Design: Limits of Performance*. New York: Prentice Hall, 1991.

[31] J. V. Burke and M. L. Overton, "Differential properties of the spectral abscissa and the spectral radius for analytic matrix-valued mappings," *Nonlinear Anal.*, vol. 23, no. 4, pp. 467–488, 1994.

[32] V. Bompart, P. Apkarian, and D. Noll, "Non-smooth techniques for stabilizing linear systems," in *Proc. American Control Conf.*, New York, July 2007, pp. 1245–1250.

[33] S. H. Lui, "Pseudospectral mapping theorem II," *Electron. Trans. Numer. Anal.*, vol. 38, pp. 168–183, 2011.

[34] D. Noll, "Convergence of non-smooth descent methods using the Kurdyka-Łojasiewicz inequality," *J. Optim. Theory Appl.*, vol. 160, no. 2, pp. 553–572, 2014.

[35] K. C. Kiwiel, "An aggregate subgradient method for nonsmooth convex minimization," *Math. Programming*, vol. 27, no. 3, pp. 320–341, 1983.

[36] M. N. Dao, "Bundle method for nonconvex nonsmooth constrained optimization," 2014, submitted.

[37] D. Noll, "Cutting plane oracles to minimize non-smooth non-convex functions," *Set-Valued Var. Anal.*, vol. 18, no. 3-4, pp. 531–568, 2010.

[38] J. Bolte, A. Daniilidis, and A. Lewis, "The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems," *SIAM J. Optim.*, vol. 17, no. 4, pp. 1205–1223, 2006.

[39] E. Bierstone and P. D. Milman, "Semianalytic and subanalytic sets," *Inst. Hautes Études Sci. Publ. Math.*, vol. 67, pp. 5–42, 1988.

[40] D. Klatte and B. Kummer, *Nonsmooth Equations in Optimization. Regularity, Calculus, Methods and Applications*, ser. Nonconvex Optim. Appl. Dordrecht: Kluwer Academic Publishers, 2002, vol. 60.

[41] R. Mifflin, "Semismooth and semiconvex functions in constrained optimization," *SIAM J. Control Optimization*, vol. 15, no. 6, pp. 959–972, 1977.

[42] J. Moro, J. V. Burke, and M. L. Overton, "On the Lidskii-Vishik-Lyusternik perturbation theory for eigenvalues of matrices with arbitrary Jordan structure," *SIAM J. Matrix Anal. Appl.*, vol. 18, no. 4, pp. 793–817, 1997.

[43] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*. New York: Macmillan Publishing Company, 1992.

[44] C. S. Resnik, "A method for robust control of systems with parametric uncertainty motivated by a benchmark example," Master's thesis, June 1991.

[45] U. Chaiya and S. Kaitwanidvilai, "Fixed-structure robust DC motor speed control," in *Proc. International MultiConference of Engineers and Computer Scientists (IMECS)*, vol. II, Hong Kong, March 2009, pp. 1533–1536.

[46] G. Filardi, O. Sename, A. Besancon-Voda, and H.-J. Schroeder, "Robust H_∞ control of a DVD drive under parametric uncertainties," in *European Control Conf. (ECC)*, Cambridge, September 2003.

[47] D. F. Enns, "Model reduction for control system design," Ph.D. dissertation, Stanford University, 1984.

[48] R. Vadigepalli, E. P. Gatzke, and F. J. Doyle III, "Robust control of a multivariable experimental four-tank system," *Ind. Eng. Chem. Res.*, vol. 40, no. 8, pp. 1916–1927, 2001.

[49] D. W. Gu, P. H. Petkov, and M. M. Konstantinov, *Robust Control Design with Matlab*. London: Springer-Verlag, 2005.

[50] Y. Cheng and B. L. R. D. Moor, "Robustness analysis and control system design for a hydraulic servo system," *IEEE Trans. on Control System Technology*, vol. 2, no. 3, pp. 183–197, 1994.

[51] D. Alazard, C. Cumer, P. Apkarian, M. Gauvrit, and G. Ferreres, *Robustesse et Commande Optimale*. Toulouse: Cépaduès Éditions, 1999.

[52] D. L. Krueger, "Parametric uncertainty reduction in robust multivariable control," Ph.D. dissertation, Naval Postgraduate School, September 1993.

[53] C. W. Scherer and I. E. Köse, "Robustness with dynamic IQCs: an exact state-space characterization of nominal stability with applications to robust estimation," *Automatica J. IFAC*, vol. 44, no. 7, pp. 1666–1675, 2008.
[54] Y.-M. Kim, "Robust and reduced order H-Infinity filtering via LMI approach and its application to fault detection," Ph.D. dissertation, Wichita State University, May 2006.
[55] D. Noll, M. Torki, and P. Apkarian, "Partially augmented Lagrangian method for matrix inequality constraints," *SIAM J. Optim.*, vol. 15, no. 1, pp. 161–184, 2004.
[56] P. M. Young, "Controller design with real parametric uncertainty," *Internat. J. Control*, vol. 65, no. 3, pp. 469–509, 1996.

Pierre Apkarian received the Ph.D. degree in control engineering from the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace (ENSAE), France, in 1988. He was qualified as a Professor from University of Toulouse (France) in both control engineering and applied mathematics in 1999 and 2001, respectively. Since 1988, he has been a Research Scientist at ONERA (Office National d'Etudes et de Recherches Aérospatiales) and an Associate Professor at the University of Toulouse. Pierre Apkarian has served as an associate editor for the IEEE Transactions on Automatic Control. His research interests include robust and gain-scheduling control theory through LMI methods or optimization-based techniques. More recently, his research has focused on the development of specialized non-smooth programming techniques for control system design. He is co-author with Dominikus Noll and Pascal Gahinet of the `HINFSTRUCT` and `SYSTUNE` software in MATLAB's Robust Control Toolbox.

Minh Ngoc Dao received his B.Sc. and M.Sc. in mathematics from Hanoi National University of Education (Vietnam) in 2004 and 2006, and his Ph.D. in applied mathematics from the University of Toulouse (France) in 2014. He has held a lecturer position at Hanoi National University of Education since 2004, and is currently a Postdoctoral Fellow at the University of British Columbia, Kelowna (Canada). Dr. Dao's research interests include nonlinear optimization, projection methods, monotone operator theory and automatic control.

Dominikus Noll received his Ph.D. and habilitation in 1983 and 1989 from Universität Stuttgart (Germany). Since 1995 he is a professor of applied mathematics at the University of Toulouse (France), and a distinguished professor of mathematics since 2009. Dr. Noll has held visiting positions at Upsala University, Dalhousie University, the University of Waterloo, Simon Fraser University, and the University of British Columbia. Dr. Noll's current research interests include nonlinear optimization, optimal control, projection-based iterative schemes, and robust feedback control design. He is co-inventor of the synthesis tools `HINFSTRUCT` and `SYSTUNE`. Dr. Noll is Associate Editor of the Journal of Convex Analysis.