MULTI-SCENARIO TIME-DOMAIN CONTROL DESIGN USING A NONSMOOTH APPROACH

P. Apkarian
ONERA-Toulouse, Control Dept. & Paul Sabatier Univ., Maths. Inst.
2 av. Edouard Belin, 31055 Toulouse, France
E-mail: apkarian@cert.fr

A. M. Simões
ONERA-Toulouse, French Aerospace Lab
2 av. Edouard Belin, 31055 Toulouse, France
E-mail: alberto.simoes@cert.fr

D. Noll
Paul Sabatier Univ., Maths. Inst.
2 av. Edouard Belin, 31055 Toulouse, France
E-mail: noll@mip.ups-tlse.fr

Key word: Nonsmooth optimization, structured controllers, time-domain constraints, time response shaping, tracking, decoupling, reliable control.

In this work we expand on our previous nonsmooth time-domain design technique to multiple input scenarios. A plant or plant family is submitted to a collection of test inputs and every closed-loop response so generated is called a scenario. The proposed design technique computes a controller with prescribed structure such that performance bounds on closed-loop responses are met for all scenarios. Such a design problem is very hard and only local solutions can be computed. In this difficult context, the proposed nonsmooth technique shows great promise as demonstrated on a variety of examples.

1. Introduction

Nonsmooth optimization techniques have been used recently to solve a number of difficult problems in structured linear controller design [1, 4, 2, 18]. These design methods avoid using Lyapunov variables whose size inflation is quadratic in the number of plant states. Consequently, nonsmooth techniques continue to perform even for sizable systems whereas available BMI and LMI-based techniques usually succumb. Another equally appealing feature is their flexibility to cover a vast array of practical needs and situations in brute form, that is, as posed in practice without resorting to often conservative relaxations.

One remarkable application of nonsmooth design techniques is to the synthesis of structured controllers satisfying explicit time-domain specifications. Such constraints arise naturally in realistic engineering problems, and involve rise and settling times, overshoot or undershoot, steady-state error, input amplitude and rate constraints or other plant trajectory operational limits.

It is noticeable that most of the existing linear controller synthesis methods fail to address the above criteria directly. In frequency-domain methods such as H_∞ or H_2 techniques [21], the designer tries to satisfy time-domain specifications by weighting well chosen performance channels. Unfortunately, this leads to a complicate trial-and-error procedure followed by numerous simulation checks which is always prone to failure. Similar comments apply to optimal control or eigenstructure assignment techniques, see [14] and references therein.

Other approaches closely related to the work in this paper have been reported in the literature. For example, an l_∞-norm criterion has been used in [6, 20] to minimize the amplitude of a regulated output in response to a specific bounded input, allowing to address overshoot and settling-time specifications. Unfortunately, these techniques are often restricted to discrete-time SISO systems. Similarly to convex optimization approaches in [5, 8], they rely on the Youla parametrization which generally leads to high-order controllers devoid of any particular physical structure.

In practical applications, designers are seeking solution controllers that are not only good in a nominal situation or a specific test input but also for a family of plant modes or a collection of test inputs. Multi-scenario design refers to the problem where all possible instances are grouped together to form the design requirements. The present work extends our previous work in [3, 4] to time-domain multi-scenario design: a single plant or possibly a plant family is subject to a collection of test inputs and the responses so generated are called scenarios. The proposed design technique computes a controller with prescribed structure such that time-domain performance constraints are achieved for all scenarios. Given the inherent difficulty of the design problem, only local solutions can be reached. We show in this paper through a variety of examples that the apparent discomfort attached to the local nature of solutions is widely offset by practical advantages.

The structure of the paper is as follows. The multi-scenario time-domain synthesis problem is formalized in section 2.. The main ingredients of the proposed nonsmooth minimization technique are reviewed in section 3.. Two realistic case studies are discussed in section 4.. In the first application, a tracking and decoupling controller under control amplitude and rate constraints is designed for a satellite launcher. As a second example, a fault-tolerant flight controller is designed for a combat aircraft in challenging flight conditions. The reader is referred to the paper full version and to [4, 2, 1] for further details on the proposed technique.

Notation

Let $\mathbb{R}^{n\times m}$ denote the space of $n \times m$ matrices. We use concepts from nonsmooth analysis covered by [7]. For a locally Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$, $\partial f(x)$ denotes its Clarke subdifferential at x while $f'(x; h)$ stand for its directional derivative at x in the direction h. For functions of two variables $f(x, y)$, $\partial_1 f(x, y)$ will denote the Clarke subdifferential with respect to the first variable. For differentiable functions f of two variables x and y the notation $\nabla_x f(x, y)$ stands for the gradient with respect to the first variable. The max operator applied to a vector $v \in \mathbb{R}^n$ is defined as $\max v = \max\limits_{i=1,\ldots,n} v_i$. The notation $[.]_+$ applied to a scalar α denotes the threshold function $[\alpha]_+ = \max\{0, \alpha\}$. Its generalization to a vector $v \in \mathbb{R}^n$ is defined as $[v]_+ = \max\{0, \max v\} = \max\limits_{i=1,\ldots,n} [v_i]_+$.

2. Multi-scenario time-domain synthesis set-up

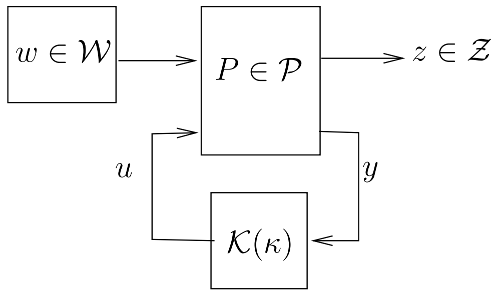


Fig. 1. multi-scenario interconnection

To begin with, we first expose the general set-up for time-domain synthesis investigated in the paper. The synthesis interconnection adopts the conventional description of the standard form ($u \in \mathbb{R}^{m_2}$ and $y \in \mathbb{R}^{p_2}$) with the following modifications, see Fig. 1. We consider a mulivalued plant $P(s)$ taking values in a finite family of linear plants $\mathcal{P} := \{P^1, \dots, P^p\}$. Each plant $P \in \mathcal{P}$ obeys the familiar state-space description of the form

$$\begin{bmatrix} \dot{x} \\ z \\ y \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x \\ w \\ u \end{bmatrix} \tag{1}$$

where indexation $i = 1, \dots, p$ for signals and matrices is omitted for simplicity.

A plant P in the family $\mathcal{P}$ in feedback loop with a single controller $K(s)$ is subject to one or several input signals w selected in a finite signal generator set $\mathcal{W} := \{w^1, \dots, w^d\}$. The closed-loop response of $P \in \mathcal{P}$ to a signal $w \in \mathcal{W}$ gives rise to a finite family of closed-loop responses $z \in \mathcal{Z}$, where $\mathcal{Z} := \{z^1, \dots, z^r\}$. Practically speaking, the signal generator is made of typical deterministic test inputs such as steps, ramps, sinusoids, etc. The above somewhat abstract description is flexible enough to reflect situations in which a single plant is submitted to various test signals as is the case when decoupling properties must be examined, or when the original system is described by multiple operating conditions or faulty modes. The latter cases are often referred to as multi-model control [13, 15] and reliable control [12, 17]. The proposed set-up also accepts more complicate formulations where each plant in the family is tested against several inputs. An instance of $\mathcal{Z}$ is called a scenario and a multi-scenario design technique consists in the search of a controller $K(s)$ such that appropriate time-domain specifications are achieved for all instances $z \in \mathcal{Z}$.

Most practical design problems dictate the use of structured controllers such as PID, decentralized, fixed-order, observer-based etc. It is thus convenient to introduce a controller parametrization in state-space

$$\kappa \in \mathbb{R}^q \to \mathcal{K}(\kappa) := \left[\begin{array}{cc} A_K(\kappa) & B_K(\kappa) \\ C_K(\kappa) & D_K(\kappa) \end{array} \right], \tag{2}$$

with corresponding frequency-domain representation

$$K(s) = C_K(\kappa)(sI - A_K(\kappa))^{-1} B_K(\kappa) + D_K(\kappa)\,.$$

It is not restrictive to assume that the mapping $\mathcal{K}: \mathbb{R}^q \to \mathbb{R}^{(m_2+k)\times(p_2+k)}$ is continuously differentiable but otherwise arbitrary. See [4, 18] for some examples. In (2), κ designates the design variables and k stands for the order of the controller where the case $k = 0$ of a static controller is included.

Multi-scenario design in the time-domain can be stated as: compute $\kappa \in \mathbb{R}^q$ such that the closed-loop time responses $z \in \mathcal{Z}$ obtained with controller $\mathcal{K}(\kappa)$ meet envelope constraints of the form

$$l_z(t) \le z(t) \le u_z(t),\ \forall t \ge 0,\ \forall z \in \mathcal{Z}, \tag{3}$$

where l_z and u_z are lower and upper bounds for z and are assumed piecewise constant in the sequel. These bounds are illustrated as dashed lines in Fig. 2 for step following and amplitude limitation specifications.

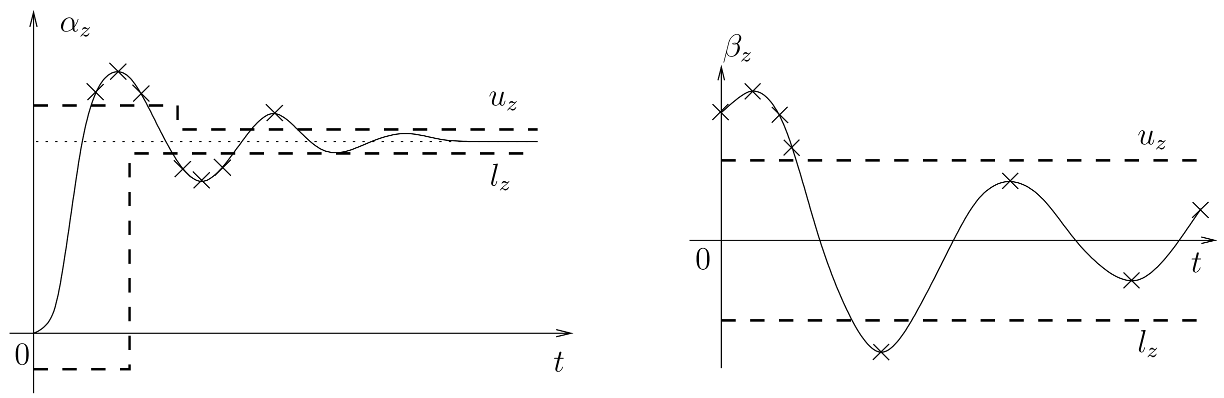


Fig. 2. shape constraints on system responses

Again for practical reasons, it may be useful to distinguish hard and soft constraints in (3). We consider a partition of $J := \{1, \dots, r\}$, the index set of $\mathcal{Z}$, into disjoint subsets S and H, i.e., $J = S \cup H$, $S \cap H = \emptyset$, where S should be seen as the index set for soft constraints and H the one for hard constraints. Correspondingly, we have a partition of the set $\mathcal{Z}$ of closed-loop responses in the form $\mathcal{Z} = \mathcal{Z}_S \cup \mathcal{Z}_H$. The notion of hard and soft constraints is clarified through the following program:

$$\begin{array}{ll} \underset{\kappa\in\mathbb{R}^q}{\text{minimize}} & \max\limits_{z\in\mathcal{Z}_S} \max\limits_{t\ge 0} \{[z(\kappa,t) - u_z(t)]_+,\ [l_z(t) - z(\kappa,t)]_+\} \\ \text{subject to} & \max\limits_{t\ge 0} z(\kappa,t) - u_z(t) \le 0,\ z \in \mathcal{Z}_H\,, \\ & \max\limits_{t\ge 0} l_z(t) - z(\kappa,t) \le 0,\ z \in \mathcal{Z}_H\,. \end{array} \tag{4}$$

A solution to program (4) necessarily meets the constraints associated with $z \in \mathcal{Z}_H$ while constraints related to $z \in \mathcal{Z}_S$ will be achieved only when the objective function falls below 0. If specifications on the signals $z \in \mathcal{Z}$ are equally important, we shall use the simpler cast

$$\underset{\kappa\in\mathbb{R}^q}{\text{minimize}} \max_{z\in\mathcal{Z}} \max_{t\ge 0} \{[z(\kappa,t) - u_z(t)]_+,\ [l_z(t) - z(\kappa,t)]_+\}\,. \tag{5}$$

By virtue of their nonconvex and nonsmooth nature, programs (4) and (5) are difficult mathematical programming problems. A specialized nonsmooth optimization technique has been developed in [3, 2] and its key ingredients are recalled in section 3..

Remark 1. *For a single plant P, with specifications $l_z(t) = u_z(t) = 0, \forall t \geq 0$ and a single excitation signal w, Program (5) reduces to*

$$\underset{\kappa\in\mathbb{R}^q}{minimize} \max_{t\geq 0} \max \left\{ [z(\kappa,t)]_+,\ [-z(\kappa,t)]_+ \right\} .$$

Since $\max \left\{ [z(\kappa,t)]_+,\ [-z(\kappa,t)]_+ \right\} = |z(\kappa,t)|$, *it becomes clear that our technique is closely related to L_∞/l_∞ norm approaches, see [20] and references therein.*

A classical approach consists in reformulating Program (4) as a smooth constrained nonlinear program, see [15] for instance. However, the use of a specialized nonsmooth optimization technique results in faster execution times, as discussed in [3].

$\diamond$

3. Key ingredients in Nonsmooth optimization

In this section we recall the key ingredients of the nonsmooth optimization method used in the experimental section 4. and refer the reader to [2] for a detailed discussion.

Introducing the notations

$$f_z(\kappa,t) := \max\{[z(\kappa,t) - u_z(t)]_+,\ [l_z(t) - z(\kappa,t)]_+\},\ z \in \mathcal{Z}_S$$

and

$$g_z(\kappa,t) := \max\{z(\kappa,t) - u_z(t),\ l_z(t) - z(\kappa,t)\},\ z \in \mathcal{Z}_H,$$

program (4) takes the more familiar form

$$\text{(6)} \qquad \begin{array}{ll} \underset{\kappa}{\text{minimize}} & f(\kappa) \\ \text{subject to} & g(\kappa) \leq 0\,, \end{array}$$

with $f(\kappa) := \max\limits_{z\in\mathcal{Z}_S} \max\limits_{t\geq 0} f_z(\kappa,t)$ and $g(\kappa) := \max\limits_{z\in\mathcal{Z}_H} \max\limits_{t\geq 0} g_z(\kappa,t)$.

To solve the constrained program (6), we follow an idea in [16] and introduce the so-called progress function for (6):

$$\text{(7)} \qquad F(\kappa^+,\kappa) = \max\{f(\kappa^+) - f(\kappa) - \mu g(\kappa)_+; g(\kappa^+) - g(\kappa)_+\},$$

where $\mu > 0$ is some fixed parameter (with $\mu = 1$ a typical value), κ represents the current iterate, and κ^+ is the next iterate or a candidate for the next iterate. Excepting the case where $\bar{\kappa}$ is a minimum of the constraint violation $g(\bar{\kappa}) > 0$, it is shown in [16] that critical points $\bar{\kappa}$ of $F(\cdot,\bar{\kappa})$ will also be critical points of the original program (6). We refer the reader to [16] and [2] for an in-depth discussion of this property.

The choice of the progress function in (7) leads to a so-called phase I/phase II method. As long as the constraint $g(\kappa) \leq 0$ is not satisfied, the right hand term in (7) is dominant and reducing it amounts to reducing constraint violation. This is phase I, which ends successfully as soon as a feasible iterate $g(\kappa^k) \leq 0$ has been found. Now phase II begins, and from now on iterates stay (strictly) feasible, and the objective function is minimized at each step. Notice that the choice of the constant $\mu > 0$ may have an influence on the behavior of the method in phase I, but has been fixed to $\mu = 1$ in our numerical implementation.

Our search for a point $\bar{\kappa}$ with $0 \in \partial_1 F(\bar{\kappa},\bar{\kappa})$ is based on an iterative descent procedure. Suppose the current iterate κ is such that $0 \notin \partial_1 F(\kappa,\kappa)$. Then it is possible to reduce the function $F(\cdot,\kappa)$

is a neighborhood of κ, that is, to find κ^+ such that $F(\kappa^+, \kappa) < F(\kappa, \kappa)$. Replacing κ by κ^+, we repeat the procedure. Unless $0 \in \partial_1 F(\kappa^+, \kappa^+)$, in which case we are done, it is possible to find κ^{++} such that $F(\kappa^{++}, \kappa^+) < F(\kappa^+, \kappa^+)$, etc. The sequence $\kappa, \kappa^+, \kappa^{++}, \ldots$ so generated is expected to converge to the sought local minimum $\bar{\kappa}$ of (6).

We find a descent step κ^+ away from the current κ by solving a tangent program at κ. Its name is derived from the fact that a first-order approximation $\widehat{F}(\cdot, \kappa)$ of $F(\cdot, \kappa)$ is built, which provides a descent direction $d\kappa$ at κ, that is, $d_1 F(\kappa, \kappa; d\kappa) < 0$, where $d_1 F$ denotes the directional derivative of $F(\cdot, \kappa)$ at κ in direction $d\kappa$. The next iterate is then $\kappa^+ = \kappa + d\kappa$, or possibly $\kappa^+ = \kappa + \alpha d\kappa$ for a suitable stepsize $\alpha \in (0, 1)$ found by a backtracking line search.

3.1. Search directions from the tangent program

In order to generate a first-order approximation $\widehat{F}(\cdot, \kappa)$ of $F(., \kappa)$ around κ, we need the set of so-called active times. To this aim, the entries of any $z(\kappa, t)$ where $z \in \mathcal{Z}_{\mathcal{S}}$ are denoted $\alpha_z(\kappa, t)$. Correspondingly, we use $l_{\alpha_z}(t)$ and $u_{\alpha_z}(t)$ to denote lower and upper bounds on α_z, respectively, and we introduce the violation functions:

$$
\begin{aligned}
f_{\alpha_z}(\kappa, t) \; &:= \; \max\{[\alpha_z(\kappa, t) - u_{\alpha_z}(\kappa, t)]_+, \; [l_{\alpha_z}(t) - \alpha_z(\kappa, t)]_+\} \\
&= \; \max\{\alpha_z(\kappa, t) - u_{\alpha_z}(t), \, l_{\alpha_z}(\kappa, t) - \alpha_z(t), \, 0\} \, .
\end{aligned} \tag{8}
$$

The set of active times for a given entry α_z is then defined as the possibly empty set

$$
T_{\alpha_z}(\kappa) := \{t \geq 0 : \exists \alpha_z, \, z \in \mathcal{Z}_S, f_{\alpha_z}(\kappa, t) = f(\kappa) > 0\},
$$

Note that the overall set $T_f(\kappa)$ of active times for f is obtained as the union of these sets and $T_f(\kappa) = \emptyset$ means $f(\kappa) = 0$, the minimum value of f has been reached.

A similar definition holds for the active times of the entries of $z(\kappa, t)$ for $z \in \mathcal{Z}_H$. Denoting these entries as $\beta_z(\kappa, t)$ with $l_{\beta_z}(t)$ and $u_{\beta_z}(t)$ as the respective lower and upper bounds, and defining the violation functions

$$
g_{\beta_z}(\kappa, t) := \max\{\beta_z(\kappa, t) - u_{\beta_z}(\kappa, t), \; l_{\beta_z}(t) - \beta_z(\kappa, t)\}, \tag{9}
$$

leads to the potentially empty sets

$$
T_{\beta_z}(\kappa) := \{t \geq 0 : \exists \beta_z, \, z \in \mathcal{Z}_H, g_{\beta_z}(\kappa, t) = g(\kappa) > 0\},
$$

As before, the set $T_g(\kappa)$ of active times for g is obtained as the union of these sets and $T_g(\kappa) = \emptyset$ means that $g(\kappa) \leq 0$, the constraints are met.

As a rule the sets $T_{\alpha_z}(\kappa)$ and $T_{\beta_z}(\kappa)$ are finite and we make this assumption from now on (see our discussion in [3]). To begin with, consider the function f and the case $f(\kappa) > 0$, because for $f(\kappa) = 0$ there is nothing to optimize. A better tangent model for f as well as for g and therefore better descent steps are obtained if we consider finite extensions $T^e_{\alpha_z}(\kappa)$ and $T^e_{\beta_z}(\kappa)$ of the active sets $T_{\alpha_z}(\kappa)$ and $T_{\beta_z}(\kappa)$, respectively. The idea behind these extensions is that enriched sets of times yield better approximations of closed-loop responses, hence a better tangent model. Since the proposed technique offers great flexibility to build such extensions, while guaranteeing convergence [2], a general characterization is the following. As an extension the active set $T^e_{\alpha_z}(\kappa)$ will contain $T_{\alpha_z}(\kappa)$, and may include additional samples with the property:

$$\forall t \in T^e_{\alpha_z}(\kappa) \text{ either } \alpha_z(\kappa,t) - u_{\alpha_z}(t) > 0 \text{ or } l_{\alpha_z}(\kappa,t) - \alpha_z(t) > 0\,.$$

For all such t, the functions f_{α_z} in (8) are differentiable in a neighborhood of κ since only one branch among three can be active in the second equation of expression (8). We infer

$$\nabla_\kappa f_{\alpha_z}(\kappa,t) = \begin{cases} \nabla_\kappa \alpha_z(\kappa,t) & \text{if } \alpha_z(\kappa,t) > u_{\alpha_z}(t) \\ -\nabla_\kappa \alpha_z(\kappa,t) & \text{if } l_{\alpha_z}(t) > \alpha_z(\kappa,t) \end{cases}$$

For all $z \in \mathcal{Z}_S$ and $t \in T^e_{\alpha_z}(\kappa)$, we collect all pairs $(\phi_f, \Phi_f) := (f_{\alpha_z}(\kappa,t), \nabla_\kappa f_{\alpha_z}(\kappa,t))$ and denote this finite set as $\mathcal{W}_f$.

A similar procedure is applied to the constraint function g. The definition of the extension set $T^e_{\beta_z}(\kappa)$ parallels that of $T^e_{\alpha_z}(\kappa)$. We infer that on such sets

$$\nabla_\kappa g_{\beta_z}(\kappa,t) = \begin{cases} \nabla_\kappa \beta_z(\kappa,t) & \text{if } \beta_z(\kappa,t) > u_{\beta_z}(t) \\ -\nabla_\kappa \beta_z(\kappa,t) & \text{if } l_{\beta_z}(t) > \beta_z(\kappa,t) \end{cases}$$

For all $z \in \mathcal{Z}_H$ and $t \in T^e_{\beta_z}(\kappa)$, we collect all pairs $(\phi_g, \Phi_g) := (g_{\beta_z}(\kappa,t), \nabla_\kappa g_{\beta_z}(\kappa,t))$ and denote this finite set as $\mathcal{W}_g$.

A tangent model of $F(.,\kappa)$ at κ is then constructed by using first-order approximations of each branch of the max-function in (7). Since the extension sets $T^e_{\alpha_z}(\kappa)$ and $T^e_{\beta_z}(\kappa)$ contain their corresponding active sets, it is straightforward to see that f and g can be expressed as

$$f(\kappa) := \max_{\{\alpha_z:\, z\in\mathcal{Z}_S\}} \max_{T^e_{\alpha_z}(\kappa)} f_{\alpha_z}(\kappa,t), \qquad g(\kappa) := \max_{\{\beta_z:\, z\in\mathcal{Z}_H\}} \max_{T^e_{\beta_z}(\kappa)} g_{\beta_z}(\kappa,t)\,.$$

With this preparation, a first-order approximation is obtained as

$$\widehat{F}(\kappa+h,\kappa) := \max\left\{ \max_{(\phi_f,\Phi_f)\in\mathcal{W}_f} \phi_f - f(\kappa) - \mu g(\kappa)_+ + \Phi_f^T h,\ \max_{(\phi_g,\Phi_g)\in\mathcal{W}_g} \phi_g - g(\kappa)_+ + \Phi_g^T h \right\},$$

where h is the displacement in the controller parameter space $\mathbb{R}^q$. Indeed, for an $\alpha_z(\kappa,t)$ the first-order approximation of $f_{\alpha_z}(\kappa+h,t)$ is $f_{\alpha_z}(\kappa+h,t) \approx f_{\alpha_z}(\kappa,t) + \nabla_\kappa f_{\alpha_z}(\kappa,t)^T h$, which leads to the term $\phi_f + \Phi_f^T h$ on the left hand branch of $\widehat{F}$, and similarly for g and its branches on the right. This gives the tangent program

$$\text{(10)} \qquad \underset{h\in\mathbb{R}^q}{\text{minimize}}\ \widehat{F}(\kappa+h,\kappa) + \tfrac{\delta}{2}\|h\|^2, \quad \text{with } \delta > 0\,.$$

It is worth noticing that an equivalent formulation for (10) is the following

$$\text{(11)} \qquad \begin{array}{lll} \underset{t,\,h\in\mathbb{R}^q}{\text{minimize}} & t + \frac{\delta}{2}\|h\|^2 & \\ \text{subject to} & \phi_f - f(\kappa) - \mu g(\kappa)_+ + \Phi_f^T h \ \leq\ t, & \forall(\phi_f,\Phi_f)\in\mathcal{W}_f\,, \\ & \phi_g - g(\kappa)_+ + \Phi_g^T h \ \leq\ t, & \forall(\phi_g,\Phi_g)\in\mathcal{W}_g\,. \end{array}$$

Program (11) is a standard convex quadratic program (CQP), and can be efficiently solved using currently available codes. Current state-of-the-art CQP codes solve problems involving several hundreds of variables and constraints in less than a second. Note that the quadratic term in (10) can be used to capture second-order information, or it may be interpreted as a trust region radius management parameter. We refer the reader to [1, 2, 18] for more eleborate variations of the present technique, and to Polak [16] for a general view on phase I/phase II methods. The key facts about (10) or (11) have been established in [1] and we state them here without proof:

- The fact that extension sets contain active sets ensures that the solution to (10) is a descent direction of $F(.,\kappa)$ at κ. If it happens that $h = 0$, then $0 \in \partial_1 F(\kappa,\kappa)$, and we are done. Clearly, a stopping test may be based on the solution to the tangent program.

- The direction h can be used in an Armijo line search [11] defined by a step α in direction h with:
$$F(\kappa + \alpha\, h, \kappa) - F(\kappa,\kappa) < \gamma\alpha F'(.,\kappa)(\kappa; h),$$
where $0 < \gamma < 1$, which terminates after finitely many steplength trials $\alpha \in (0, 1]$.

Here in both items we use the fact that $\partial_1 \widehat{F}(\kappa,\kappa) = \partial_1 F(\kappa,\kappa)$.

Having described the main features of our algorithm, we can state the pseudo-code:

Algorithm 1 nonsmooth algorithm for program (4)

Parameters: $\delta > 0$, $0 < \beta, \gamma < 1$.

1: **initialize**. Select initial κ^1.

2: **stopping test**. At counter j, stop if $0 \in \partial_1 \widehat{F}(\kappa^j, \kappa^j)$ and return κ^j. Otherwise continue.

3: **compute descent direction.** At counter j solve tangent programs (10) or (11)
$$\min_{h \in \mathbb{R}^q} \widehat{F}(\kappa^j + h, \kappa^j) + \tfrac{\delta}{2}\|h\|^2.$$
Solution is the search direction h.

4: **line search.** Find $\alpha = \beta^\nu$, $\nu \in \mathbb{N}$, satisfying the Armijo condition
$$F(\kappa^j + \alpha h, \kappa^j) - F(\kappa^j, \kappa^j) \le \gamma\alpha F'(\cdot, \kappa^j)(\kappa^j, h) < 0.$$

5: **update**. Put $\kappa^{j+1} = \kappa^j + \alpha h$, increase counter j by 1 and loop back to step 2.

3.2. Implementation details

Similarly to *iterative feedback tuning* (IFT), the proposed technique relies on simulations to compute function values as well as trajectories gradients. A comprehensive discussion on how this can be done is presented in [9, 10, 3]. This is generally the costly part of the technique since $\dim \kappa = q$ simulations for each scenario may be required in order to form the trajectories gradients. As in practice the plant trajectories are only inspected on a finite horizon, the half-line $t \ge 0$ should be replaced with $t \in [0, T]$ everywhere in the text.

Although simulations can be performed using a general-purpose ordinary differential equation solver, it is computationally more efficient for LTI systems to use the classical discrete state-propagation approach, as in the MATLAB function LSIM. This method is particularly appealing here because the simulation scenarios for a given plant in the family only differ by their input signals and consequently the dynamic equation
$$\dot{x} = Ax + B_1 w + B_2 u$$
needs only be discretized once to get the simulation dynamics
$$x_{k+1} = A^d x_k + B_1^d w_k + B_2^d u_k\,.$$
A reduction in execution time is then achieved since the data A^d,B_1^d and B_2^d can be recycled for each scenario and the rest of the computation amounts to simple matrix vector products. We

note further that the outlined procedure is amenable to parallel computing because scenarios are typically independent.

The nonsmooth technique offers the flexibility to update the simulation horizon and the sampling time along the iterations, which is in contrast with the classical smooth approach in [15].

A further important question is how to build the extension sets $T^e_{\alpha_z}$ and $T^e_{\beta_z}$ which determine the tangent program (10) and thereby the behavior of the nonsmooth algorithm. As illustrated in Fig. 2, different strategies are used for soft and hard constraints. Selected samples are represented in the figure by '$\times$' symbols. In the soft constraints case, represented on the left, $T^e_{\alpha_z}$ comprises the set of active times plus some extra samples for which constraints are violated. This is easily obtained by decimating samples provided by the numerical integrator. The extension set $T^e_{\beta_z}$, in the hard constraints case on the right, is built analogously but also includes extrema of β_z satisfying the constraints envelope. The idea here is to feed the tangent program with first-order information of g even during phase II. In return this helps preventing iterates to get stuck on the feasibility boundary.

4. Applications

The simulations and computations for the case studies presented in this section have been performed with the Matlab environment running on a 2.8GHz Pentium D processor with 1Gb RAM. Our code has been developed essentially using Matlab, with Fortran being used for the CQP tangent problem (10) to minimize the main performance bottlenecks.

4.1. Reliable flight controller

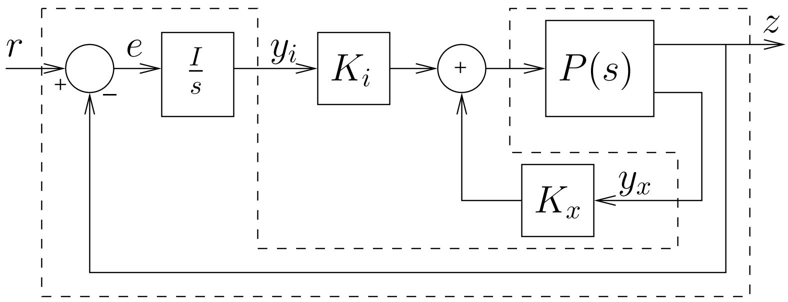


Fig. 3. closed-loop system block diagram representation

In the next example, we design a reliable flight control system for an F-16 aircraft performing high angle-of-attack maneuvers. This problem has been studied in [12] from which we borrow the model data. The primary design goal is to synthesize a stabilizing controller achieving tracking performances for the stability axis roll rate $\dot{\mu}_{rat}$, the angle-of-attack α and the sideslip angle β of the aircraft. The control system configuration is clarified in Fig. 3, with

(12) $$z^T = \left[\ \dot{\mu}_{rat} \quad \alpha \quad \beta\ \right].$$

All the aircraft states are assumed available for feedback:

(13) $$x^T = y_x^T = \left[\ u \quad w \quad q \quad v \quad p \quad r\ \right],$$

where p, q, r are respectively the roll, pitch and yaw rates, and v, w and u are the y, z and x-body axis velocities, respectively. The control vector is given as

$$u^T = \begin{bmatrix} \delta_{hr} & \delta_{hl} & \delta_{ar} & \delta_{al} & \delta_r \end{bmatrix}, \tag{14}$$

where δ_{hr}, δ_{hl}, δ_{ar}, δ_{al} and δ_r are the deflections of the right and left stabilators, the right and left ailerons and the rudder, respectively, which yields $K_x \in \mathbb{R}^{5\times 6}$ and $K_i \in \mathbb{R}^{5\times 3}$.

Given that the combat aircraft evolves in critical high angle-of-attack flight conditions, kinematics and inertial coupling phenomena become important and the control law must achieve substantial decoupling of the various channels. Additionally, the solution must guarantee closed-loop stability and satisfactory performance for any of the operation modes in table 1 in order to be eligible as a reliable controller. The linearized models P^0 and P^3 were given in [12].

Table 1. nominal and failure modes for the F-16

mode type	description
nominal operation:	$P(s) = P^0(s)$,
failure of the right stabilator:	$P(s) = P^0(s) \times \mathrm{diag}(0, 1, 1, 1, 1)$,
failure of the left stabilator:	$P(s) = P^0(s) \times \mathrm{diag}(1, 0, 1, 1, 1)$,
failure of the right aileron:	$P(s) = P^0(s) \times \mathrm{diag}(1, 1, 0, 1, 1)$,
failure of the left aileron:	$P(s) = P^0(s) \times \mathrm{diag}(1, 1, 1, 0, 1)$,
75 % symetrical impairment of the stabilators:	$P(s) = P^3(s)$,
failure in one of the redundant controllers.	$P(s) = P^0(s) \times 0.5 I_5$,

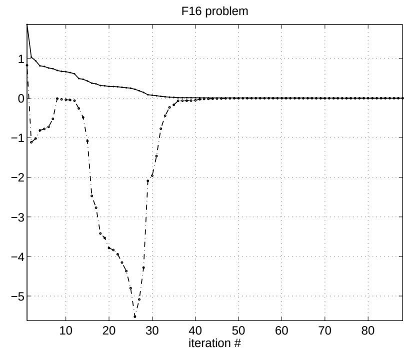
Note that the controller must achieve adequate performance not only in the nominal mode but also when either one of the failures occurs. This leads to defining 3 test inputs $w = r^1$, r^2 or r^3 described as follows:

$$r^1(t) = \begin{bmatrix} \sigma(t) \\ 0 \\ 0 \end{bmatrix}, \quad r^2(t) = \begin{bmatrix} 0 \\ \sigma(t) \\ 0 \end{bmatrix}, \quad r^3(t) = \begin{bmatrix} 0 \\ 0 \\ \sigma(t) \end{bmatrix}, \tag{15}$$

where $\sigma(t)$ stands for the unit step. Altogether, we have 3 scenarios for each mode in order to assess tracking and decoupling properties for $\dot{\mu}_{rat}$, α and β, thus giving a total of 21 scenarios. Clearly this a complicated problem involving multiple plant modes as well as multiple test inputs. This is readily incorporated within the general framework of section 2. upon defining the same test input signals as in (15) for all plants in table 1.

Additionally, it is adopted the two redundant controllers configuration depicted, following a passive redundancy strategy. Aiming to make the system reliable against an eventual failure of the controller, passive redundancy uses multiple controllers operating simultaneously in closed-loop, see [19] and the references therein. In the present example, closed-loop system stability and adequate performance must be preserved even if one of the controllers fails, as represented by the last failure mode in table 1. Note that multicontroller configurations can be easily handled by the nonsmooth technique.

For a unit step, control magnitudes and rates are limited to $15°/s$ and to $7.5°$.



(b)

Fig. 4. progress of cost and constraints functions: f (solid) and g (dash-dot)

The nonsmooth technique finds a locally optimal solution after 90 iterations corresponding to 20 minutes of cputime. Fig. 4(b) displays the evolution of functions f and g in (4) throughout the iterations sequence. We observe that hard constraints are satisfied right after the first iteration, and remain feasible in phase II until termination as theoretically predicted. The final controller found by the nonsmooth algorithm is described as:

$$
\begin{bmatrix} K_i & K_x \end{bmatrix} = \left[\begin{array}{rrr|rrrrrr}
-1.145 & -3.934 & -0.2693 & -0.06981 & 0.8754 & 22.34 & 0.003559 & 3.392 & -4.083 \\
-1.145 & -3.934 & -0.2693 & -0.06981 & 0.8754 & 22.34 & 0.003559 & 3.392 & -4.083 \\
-\mathbf{15} & 0.003514 & 1.798 & 0.1032 & -0.03847 & 2.971 & -0.7699 & 58.32 & 6.262 \\
\mathbf{15} & -0.003514 & -1.798 & -0.1032 & 0.03847 & -2.971 & 0.7699 & -58.32 & -6.262 \\
-5.004 & -0.3631 & \mathbf{15} & 0.1545 & 0.009585 & 8.99 & -2.486 & 13.7 & 110.5
\end{array}\right]. \tag{16}
$$

Figs. 5 to 7 show the closed-loop responses with the nonsmooth controller (16) for each of the seven operation modes, together with the closed-loop response under 25% and 50% impairment of the stabilators. The synthesized controller guarantees good closed-loop nominal behavior, but also closed-loop stability with limited performance deterioration even in the event of extreme failures, indicating that a reliable design has been obtained. The worst performance degradation case corresponds to the angle-of-attack tracking response under 75% impairment of the stabilators, a rather critical situation, see the central plot in Fig. 6. As expected, closed-loop responses remain satisfactory under 25% and 50% impairment of the stabilitators, even though these scenarios have not been explicitly included in the synthesis requirements. Finally, all control rate constraints are met since they were formulated as hard constraints.

Conclusion

In practical applications designers are seeking solution controllers that are not only good for a single scenario but for a collection of scenarios grouped together to form the design requirements. In this paper, we have discussed a specialized nonsmooth optimization technique to address this challenging problem class in time domain. The outcome is a more flexible design tool than traditional control design techniques since the problem is handled in brute form as posed in practice without resorting to often conservative relaxations. The proposed scenario approach also allows for a detailed analysis of each scenario individually thereby revealing the main design difficulties. Multi-scenario design is a very challenging problem for which only local solutions

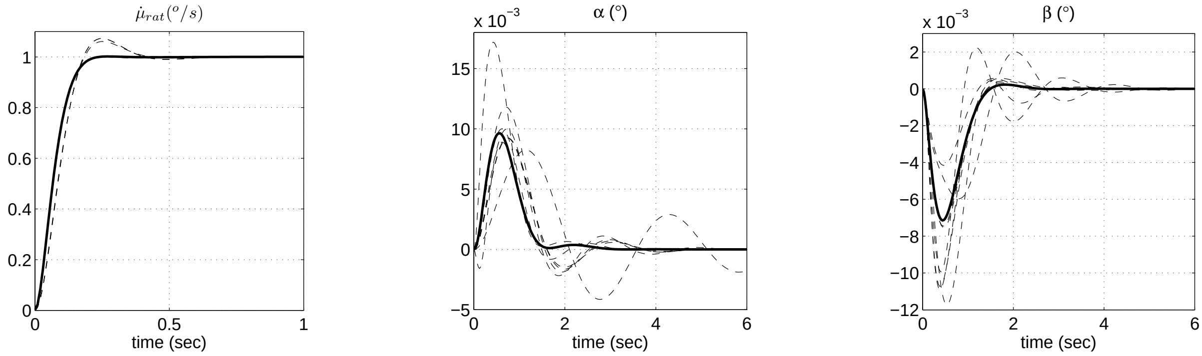


Fig. 5. closed-loop responses for a stability-axis roll rate step command (nominal: solid)

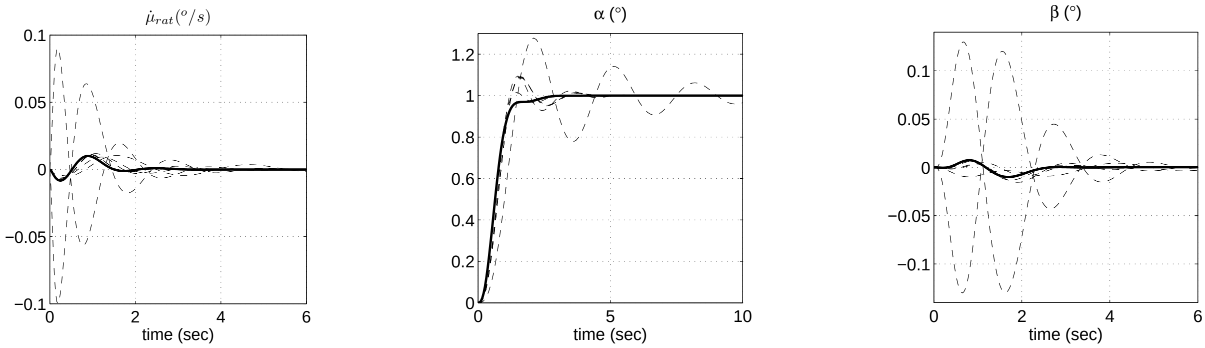


Fig. 6. closed-loop responses for an angle-of-attack step command (nominal: solid)

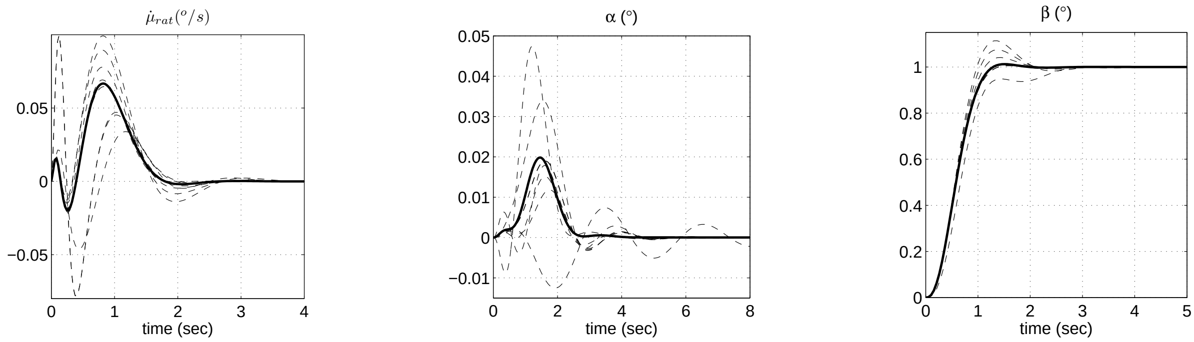


Fig. 7. closed-loop responses for a sideslip angle step command (nominal: solid)

can be reached. Despite inherent obstacles, we have shown through various case studies that the proposed technique is an effective design tool to elaborate valid practical solutions.

acknowledgement

This research was supported by ONERA and by grants from Agence Nationale de la Recherche (ANR) under contract *Survol*, and by Agence Nationale de la Recherche (ANR) under contract *Controvert*.

References

1. P. Apkarian and D. Noll. Nonsmooth H_∞ synthesis. *IEEE Trans. Aut. Control*, 51(1):71–86, 2006.
2. P. Apkarian, D. Noll, and A. Rondepierre. Mixed H_2/H_∞ control via nonsmooth optimization. *SIAM J. on Control and Optimization*, 47(3):1516–1546, 2008.
3. P. Apkarian, D. Noll, and A. Simões. Time-domain control design: a non-differentiable approach. *submitted*, 2008.
4. V. Bompart, P. Apkarian, and D. Noll. Control design in the time- and frequency-domain using nonsmooth techniques. *Syst. Control Letters*, 57(3):271–282, 2008.
5. S. Boyd and C. Barratt. *Linear Controller Design: Limits of Performance*. Prentice-Hall, 1991.
6. A. Casavola and E. Mosca. Minimization of a closed-loop response to a fixed input for SISO systems. *IEEE Trans. Aut. Control*, 42:1581–1587, 1997.
7. F. H. Clarke. *Optimization and Nonsmooth Analysis*. Canadian Math. Soc. Series. John Wiley & Sons, New York, 1983.
8. N. Elia and M.A. Dahleh. Controller design with multiple objectives. *IEEE Trans. Aut. Control*, 42(5):596–613, 1997.
9. H. Hjalmarsson. Iterative feedback tuning–an overview. *Int. J. Adaptive Contr. and Sig. Process.*, 16:373–395, 2002.
10. H. Hjalmarsson, M. Gevers, S. Gunnarsson, and O. Lequin. Iterative feedback tuning: theory and applications. *IEEE Control Syst. Mag.*, 18(4):26–41, 1998.
11. J.E. Dennis, Jr. and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM's Classics in Applied Mathematics. SIAM, 1996.
12. F. Liao, J. L. Wang, and G.-H. Yang. Reliable robust flight tracking control: An LMI approach. *IEEE Trans. Control Sys. Tech.*, 10:76–89, 2002.
13. P. Mäkilä. Multiple models, multiplicative noise and linear quadratic control-algorithm aspects. *Int. J. Control*, 54(4):921–941, October 1991.
14. A. P. Oliva and W. C. Leite Filho. Eigenstructure versus optimal control for decoupling. *Control Engineering Practice*, 10:1059–1079, 2002.
15. Y. Piguet, U. Holmberg, and R. Longchamp. A minimax approach for multi-objective controller design using multiple models. *Int. J. Control*, 72(7):716–726, may 1999.
16. E. Polak. *Optimization : Algorithms and Consistent Approximations*. Applied Mathematical Sciences, 1997.
17. G. Pujol, J. Rodellar, J.M. Rossell, and F. Pozo. Decentralised reliable guaranteed cost control of uncertain systems: an LMI design. *IET Control Theory & Applications*, 1:779–785, 2007.
18. A. Simoes, P. Apkarian, and D. Noll. A nonsmooth progress function for frequency shaping control design. *IET Control Theory & Applications*, 2(4):323–336, April 2008.
19. K. Suyama. Advanced cooperative order in a compensator set in passive redundancy. *IEEE Trans. Aut. Control*, 46(7):1151–1155, 2001.
20. M-G Yoon. Sign-weighted peak minimization problem for feedback systems. *IEEE Trans. Aut. Control*, 46:943–948, 2001.
21. K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Printice Hall, 1996.