The H_∞ control problem is solved

by

Pierre Apkarian[*] and Dominikus Noll[†]

The H_∞ control problem was posed by G. Zames in 1981* [1]*, and solved by P. Apkarian and D. Noll in 2006* [2]*. In this treatise we present the rational of H_∞ control, give a short history, and recall the milestones reached before our 2006 solution. We also discuss the recent Matlab function* `hinfstruct`*, based on work by Apkarian, Noll and Gahinet (TheMathworks) and available in the Robust Control Toolbox since R2010b.

[*]ONERA, Control System Department, Toulouse, France

[†]Université Paul Sabatier, Institut de Mathématiques, Toulouse, France

Contents

1 The H_∞ control problem

The H_∞-problem was framed by G. Zames in two plenary talks at the IEEE CDC in 1976 and the Allerton Conference in 1979, and posed formally in his 1981 paper [1]. However, the origins of the H_∞-problem are much older and date back to the 1960s, when Zames discovered the small gain theorem [3]. After more than 30 years the H_∞-problem was solved by P. Apkarian and D. Noll in 2006 [2].

In this section we introduce the H_∞ control problem formally, discuss its rationale, and present the context leading to our 2006 solution.

1.1 Some history

In their seminal 1989 paper [4] Doyle, Glover, Khargonekar and Francis show that the H_∞ problem requires the solution of two algebraic Riccati equations (AREs). Doyle [5] discusses how this milestone is reached and mentions an earlier 1984 solution. In 1994 P. Gahinet and P. Apkarian give a solution [6] of the H_∞ problem by reducing it to a linear matrix inequality (LMI), the 1995 solution. How can a problem be solved several times? What do we mean when we say that we solved the problem in 2006 [2], when there are the 1984, 1989, and 1995 solutions already?

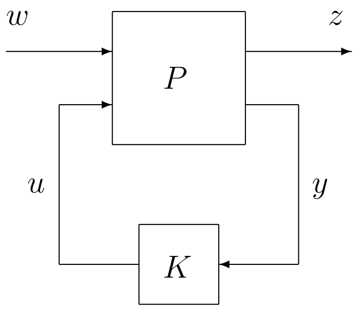
1.2 Formal statement of the problem

The H_∞ control problem can be stated as follows. Given a real rational transfer matrix $P(s)$, called the *plant*, and a space $\mathcal{K}$ of real rational transfer matrices $K(s)$, called the *controller space*, characterize and compute an optimal solution $K^* \in \mathcal{K}$ to the following optimization program

$$\begin{array}{ll} \text{minimize} & \|T_{w\to z}(P,K)\|_\infty \\ \text{subject to} & K \text{ stabilizes } P \text{ internally} \\ & K \in \mathcal{K} \end{array} \tag{1}$$

Here the objective function is the H_∞-norm of the closed-loop performance channel $T_{w\to z}(P,K)$, see (2). As we shall see the choice of the controller space $\mathcal{K}$ in (1) is the key for a proper understanding of the problem.

(2)



Let us recall the notions used to formulate (1). The plant $P(s)$ has a state-space representation of the form

$$P : \begin{cases} \dot{x} = Ax + B_1 w + B_2 u \\ z = C_1 x + D_{11} w + D_{12} u \\ y = C_2 x + D_{21} w + D_{22} u \end{cases} \qquad P(s) : \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \tag{3}$$

where $x \in \mathbb{R}^{n_p}$ is the state, $u \in \mathbb{R}^{n_u}$ the control, $y \in \mathbb{R}^{n_y}$ the measured output, $w \in \mathbb{R}^{n_w}$ the exogenous input, and $z \in \mathbb{R}^{n_z}$ the regulated output. Similarly, $K(s)$ has state-space representation

$$(4) \qquad K : \left\{ \begin{array}{rcl} \dot{x}_K &=& A_K x_K + B_K y \\ u &=& C_K x_K + D_K y \end{array} \right. \qquad K(s) : \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array} \right]$$

with $x_K \in \mathbb{R}^k$ the state of K. As soon as $D_{22} = 0$, the closed-loop transfer channel $T_{w\to z}(P,K)$ in (1) has the state-space representation

$$(5) \qquad T_{w\to z}(P,K) : \left[\begin{array}{c|c} A(K) & B(K) \\ \hline C(K) & D(K) \end{array} \right]$$

where

$$(6) \qquad A(K) = \left[\begin{array}{cc} A + B_2 D_K C_2 & B_2 C_K \\ B_K C_2 & A_K \end{array} \right], B(K) = \left[\begin{array}{c} B_1 + B_2 D_K D_{12} \\ B_K D_{21} \end{array} \right], C(K) = etc.$$

and where the state dimension is $n_p + k$. Finally, for a stable real rational transfer function $T(s)$, the H_∞-norm in (1) is defined as

$$(7) \qquad \|T\|_\infty = \max_{\omega \in \mathbb{R}} \overline{\sigma}\left(T(j\omega)\right),$$

where $\overline{\sigma}(M)$ is the maximum singular value of a complex matrix M.

With these notations we can now give the first explanation. The 1984, 1989 and 1995 solutions of the H_∞ problem (1) are all obtained within the space $\mathcal{K}_{\text{full}}$ of full-order controllers

$$\mathcal{K}_{\text{full}} = \{K : K \text{ has form (4) with size}(A_K) = \text{size}(A)\}.$$

Observe that in $\mathcal{K}_{\text{full}}$ *all* entries in A_K, B_K, C_K, D_K are free variables. Altogether there are $N := n_p^2 + n_p(n_y + n_u) + n_y n_u$ degrees of freedom and we have

$$\mathcal{K}_{\text{full}} \cong \mathbb{R}^N.$$

In particular, $\mathcal{K}_{\text{full}}$ is the largest controller space we could use in (1). Finding a solution within $\mathcal{K}_{\text{full}}$ is therefore easiest. In particular with $\mathcal{K}_{\text{full}}$ as controller space (1) is convex, as shown in [6]. When *smaller* and more practical controller spaces $\mathcal{K}$ are chosen, problem (1) is much harder to solve. Our 2006 solution addresses these difficult cases.

Solutions of the H_∞-control problem in the 1980s and 1990s refer to the full-order case, which is essentially convex.

1.3 The rationale

After closing the loop in the feedback scheme (2) we may consider the closed-loop system as a linear operator $T_{w\to z}(P,K)$ mapping input w to output z. If K stabilizes P internally, that is, if $A(K)$ in (6) is stable, then $T_{w\to z}(P,K)$ maps $w \in L^2$ into $z \in L^2$. The H_∞-norm (7) is then nothing else but the L^2-L^2-operator norm, that is,

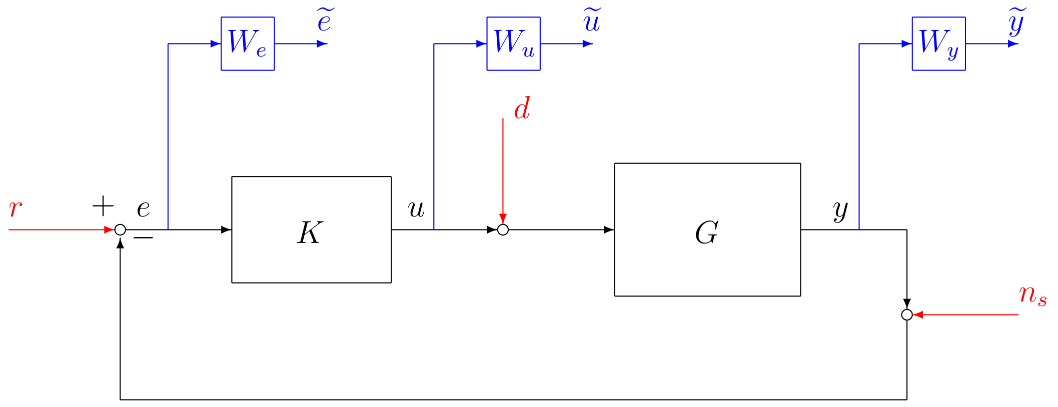
$$\|T\|_\infty = \sup_{w\neq 0} \frac{\|Tw\|_2}{\|w\|_2} = \sup_{w\neq 0} \frac{\|z\|_2}{\|w\|_2}.$$

In other words, for a closed-loop channel $w \to z$ the norm $\gamma = \|T_{w\to z}(P,K)\|_\infty$ is the factor by which the energy of the input signal is amplified in the output. Input w with energy $\|w\|_2^2$ will produce output z with energy $\|z\|_2^2$ no greater than $\gamma^2 \cdot \|w\|_2^2$, as long as controller K is used. The optimization program (1) tries to find the controller $K^* \in \mathcal{K}$ for which this amplification factor γ is smallest.

In closed-loop with controller K the input w with energy $\|w\|_2^2$ creates output z with energy $\|z\|_2^2 \leq \gamma^2\|w\|_2^2$, where $\gamma = \|T_{w\to z}(P,K)\|_\infty$. The same relation holds for power signals $w \to z$, i.e., power is amplified by no more than γ^2.

This can obviously be very useful. All we have to do is find communication channels $w \to z$, where smallness of answer z to input w tells us something useful about the system. We now give the typical context of *loopshaping*, where this idea is used.

(8)



The standard control scheme (8) features the open-loop system G, the controller K, the measured output y, the control signal u, the tracking error e. Red signals are inputs, n_s = sensor noise, d = disturbance or process noise, and r = reference signal for y, sometimes called a command. The blue signals are specifically chosen outputs, $\widetilde{e} = W_e e$, $\widetilde{u} = W_u u$, $\widetilde{y} = W_y y$. This is a special case of (2), where $w = (r, d, n_s)$ is the input, $z = (\widetilde{e}, \widetilde{u}, \widetilde{y})$, and where plant P regroups G and the filters W_e, W_u, W_y. The filters may be dynamic, which adds new states into the plant P.

What are useful transfer functions from red to blue? For instance the transfer from reference r to tracking error e

$$T_{r\to e}(K) = (I + GK)^{-1}$$

is a typical performance channel, because it describes how fast the system follows the reference r. As one typically wants to track only in the low frequency range, W_e is a low-pass filter. Now smallness of the norm

$$\|T_{r\to\widetilde{e}}(K)\|_\infty = \|W_e(I+GK)^{-1}\|_\infty$$

means that the low frequency component $\widetilde{e}$ of the tracking error e is small, so y follows the reference input r in low frequency.

Next consider a typical robustness channel. For instance, the influence of sensor noise n_s on the control signal u. Noise is typically of high frequency, but that should not lead to high frequency components in u, as this could lead e.g. to actuator fatigue. Therefore W_u is typically a high-pass filter and $\widetilde{u}$ are high frequency components of u. We find

$$T_{n_s\to\widetilde{u}}(K) = -W_u(I+KG)^{-1}K$$

and $\|T_{n_s\to\widetilde{u}}(K)\|_\infty$ puts a cost on high frequency components in u. If program (1) is successful, it will furnish an optimal $K^* \in \mathcal{K}$ which makes this cost as small as possible, thereby building robustness to sensor noise into the system.

To conclude, we can see that depending on the specific application there will be several performance and robustness channels. As (2) requires fixing a single connection $w \to z$, we will have to decide on some specific weighing between those.

Setting up the performance channel $w \to z$ in (1) could be interpreted as putting a cost on undesirable behavior of the closed-loop system.

1.4 Controller structures

The reason why the H_∞ theory of the 1980s failed to grip in practice is quickly explained. Controllers computed via algebraic Riccati equations are full order, or *unstructured*. However, for various reasons, practitioners prefer simple controllers like PIDs, or control architectures combining PIDs with filters, and such controllers are *structured*.

The discrepancy between H_∞ theory and control engineering practice is highlighted e.g. by PID control. To this day PID controllers are tuned ***instead of optimized, because software for H_∞-PID control was not available.***

During the 1990s and early 2000s a new approach to controller design based on linear matrix inequalities (LMIs) was developed. Unfortunately, LMIs have essentially the same shortcomings as AREs. H_∞ controllers computed via LMIs are still unstructured. The situation only started to improve when in the late 1990s the authors pioneered the investigation of feedback controller synthesis via bilinear matrix inequalities (BMIs). While the LMI euphoria was still in full progress, we had recognized that what was needed were algorithms which allowed to synthesize structured controllers. Here is the formal definition of structure (cf. [2]).

Definition 1. *A controller K of the form (4) is called* structured *if the state-space matrices A_K, B_K, C_K, D_K depend smoothly on a design parameter vector θ varying in some parameter space $\mathbb{R}^n$, or in a constrained subset of $\mathbb{R}^n$.* □

In other words, a controller structure $K(\cdot)$, or $K(\theta)$, consists of four smooth mappings $A_K(\cdot) : \mathbb{R}^n \to \mathbb{R}^{k\times k}$, $B_K(\cdot) : \mathbb{R}^n \to \mathbb{R}^{k\times n_y}$, $C_K(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n_u\times k}$, and $D_K(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n_u\times n_y}$.

It is convenient to indicate the presence of structure in K by the notation $K(\theta)$, where θ denotes the free parameters. In the Matlab function* `hinfstruct` *one refers to θ as the vector of tunable parameters.

1.5 Three basic examples with structure

The structure concept is best explained by examples. The transfer function of a realizable PID controller is of the form

$$K(s) = k_p + \frac{k_i}{s} + \frac{k_d s}{1 + T_f s} = d_K + \frac{r_i}{s} + \frac{r_d}{s+\tau}, \tag{9}$$

where $d_K = k_p + k_d/T_f$, $\tau = 1/T_f$, $r_i = k_i$, $r_d = -k_d T_f^2$. Realizable PIDs may therefore be represented in state-space form

$$K_{\text{pid}}(\theta) : \left[\begin{array}{cc|c} 0 & 0 & r_i \\ 0 & -\tau & r_d \\ \hline 1 & 1 & d_K \end{array}\right] \tag{10}$$

with $\theta = (r_i, r_d, d_K, \tau) \in \mathbb{R}^4$. As we can see,

$$A_K(\theta) = \begin{bmatrix} 0 & 0 \\ 0 & -\tau \end{bmatrix}, \quad B_K(\theta) = \begin{bmatrix} r_i \\ r_d \end{bmatrix}, \quad C(K) = [1 \;\; 1], \quad D_K = d_K.$$

If we use the PID structure (10) within the H_∞ framework (1), we compute an H_∞ PID controller, that is, a PID controller which minimizes the closed-loop H_∞-norm among all internally stabilizing PID controllers:

$$\|T_{w\to z}(P, K_{\text{pid}}^*)\|_\infty \le \|T_{w\to z}(P, K_{\text{pid}})\|_\infty .$$

The controller space for this structure is

$$\mathcal{K}_{\text{pid}} = \left\{ K_{\text{pid}}(\theta) : \text{ as in (10)}, \theta = (r_i, r_d, d_K, \tau) \in \mathbb{R}^4 \right\}.$$

The fact that PID is a structure in the sense of Def. 1 means that PIDs may now be optimized instead of tuned.

A second classical controller structure, related to the fundamental work of Kalman in the 1960s, is the observer-based controller, which in state-space has the form:

$$K_{\text{obs}}(\theta) : \left[\begin{array}{c|c} A + B_2 K_c + K_f C_2 & -K_f \\ \hline K_c & 0 \end{array}\right]. \tag{11}$$

Here the vector of tunable parameters θ regroups the elements of the Kalman gain matrix K_f and the state-feedback control matrix K_c. That is $\theta = (\text{vec}(K_f), \text{vec}(K_c))$. Since the plant P has n_p states, n_y outputs and n_u inputs, θ is of dimension $n_p(n_y+n_u)$, i.e., $n = n_p(n_y+n_u) < N$, which indicates that the controller is structured, even though $k = n_p$. In fact, formally the structure of observer-based controllers is defined as

$$\mathcal{K}_{\text{obs}} = \left\{K_{\text{obs}}(\theta) : \text{as in (11)}\ , \theta = (\text{vec}(K_f), \text{vec}(K_c)) \in \mathbb{R}^{n_p(n_y+n_u)}\right\}.$$

Now if we use (11) within the framework (1), we are computing an observer-based H_∞-controller. But do not observer-based ontrollers K_{obs} belong to the realm of H_2-control? This is H_∞ control! Are we mixing things? Yes we are, but why not? If we are attached to the observer-structure, and at the same time appreciate the robustness of H_∞-control, then we should by any means mix things. The result will be a controller $K_{\text{obs}}(\theta^*)$, where $\theta^* = (\text{vec}(K_f^*), \text{vec}(K_c^*))$ gives us two gain matrices K_c^* and K_f^*, neither of which is by itself optimal in any sense[1]. In particular there are no algebraic Riccati equations for K_f^* or K_c^*. Nonetheless, taken together they are optimal in the sense that

$$\|T_{w\to z}\left(P, K_{\text{obs}}(\theta^*)\right)\|_\infty \leq \|T_{w\to z}\left(P, K_{\text{obs}}(\theta)\right)\|_\infty$$

for any other observer-based controller $K_{\text{obs}}(\theta)$ which stabilizes P internally.

Observer-based H_∞-control is perhaps exotic, but it is at least made possible by our 2006 solution

A third basic controller structure are reduced order controllers. More precisely, the order of K is fixed at $k < n_p$. This is the simplest example of a structure, namely

$$\mathcal{K}_k = \{K : \ K \text{ as in (4) with size}(A_K) = k \times k\}.$$

Here the vector of tunable elements is $\theta = (\text{vec}(A_K), \text{vec}(B_K), \text{vec}(C_K), \text{vec}(D_K))$ of dimension $n = k^2 + k(n_y + n_u) + n_y n_u$. This is a structure in the spirit of our definition, because it uses fewer degrees of freedom than the full order controller, which has $N = n_p^2 + n_p(n_y + n_u) + n_y n_u$ free places.

Why is it reasonable to call $\mathcal{K}_k$ a structure as soon as $k < n_p$? The reason is that computing reduced fixed-order optimal H_∞-controllers is substantially more complicated than computing the full-order H_∞ controller. In lieu of two decoupled Riccati equations, $K^* \in \mathcal{K}_k$ requires four coupled Riccati equations, [7], and the numerical procedures proposed in the 1990s are clearly demanding. In the realm of matrix inequalities the H_∞-problem for reduced-order controllers is also well-studied. One obtains an LMI in tandem with a rank constraint, a non-convex problem which is equivalent to a BMI.

[1]The principle of separation of observation and control is no longer valid

Controllers with structure come up naturally. That's why the authors pioneered the investigation of structured H_∞-synthesis in the 1990s.

2 The solution of the H_∞-control problem

A problem which was left open for 30 years may be expected to be difficult. The difficulty in the H_∞-control problems is due to the fact that it is nonconvex, and that the objective in (1) is nonsmooth. Moreover, there is a third difficulty, which is related to stability in closed-loop.

2.1 Nonsmooth optimization

Assuming that $K(\theta)$ is structured with parameter $\theta \in \mathbb{R}^n$, we write the closed-loop transfer channel $w \to z$ in (5) as

$$T_{w\to z}(P, K(\theta)) : \left[\begin{array}{c|c} A(K(\theta)) & B(K(\theta)) \\ \hline C(K(\theta)) & D(K(\theta)) \end{array}\right].$$

Then the H_∞-objective function in (1) becomes

(12)
$$f(\theta) := \|T_{w\to z}(P, K(\theta))\|_\infty = \max_{\omega\in\mathbb{R}} \overline{\sigma}\left(C(K(\theta))(j\omega I - A(K(\theta)))^{-1}B(K(\theta)) + D(K(\theta))\right),$$

a nonsmooth, nonconvex function, which in addition, is not defined everywhere. Its domain $D_f = \{\theta \in \mathbb{R}^n : f(\theta) < \infty\}$ contains the internally stabilizing set

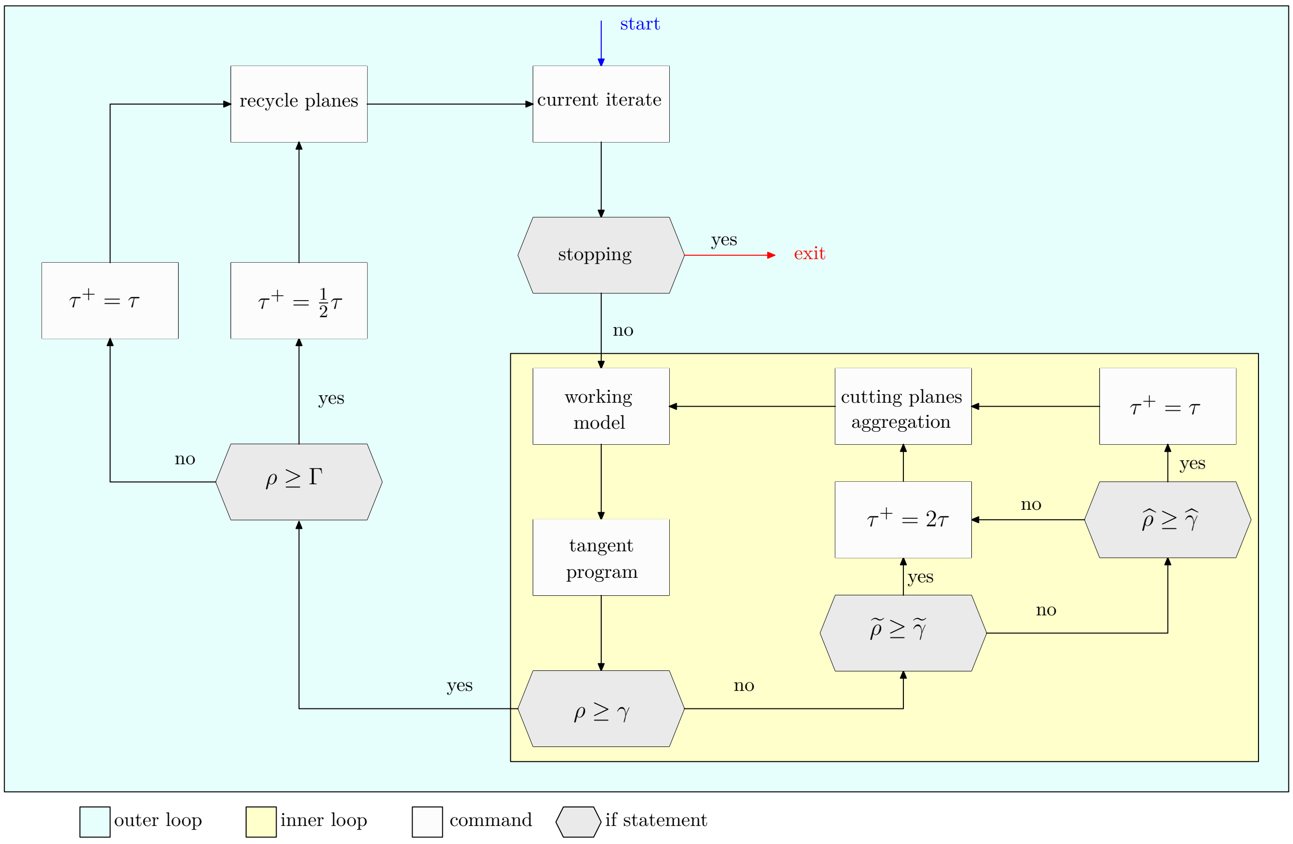
$$D_s = \{\theta \in \mathbb{R}^n : K(\theta) \text{ stabilizes } P \text{ internally}\}. \tag{13}$$

The first major step toward the solution of the H_∞ control problem in the seminal paper [2] was to characterize and compute the Clarke subdifferential of the function f. This allowed to formulate necessary optimality conditions, and thereby to characterize locally optimal solutions of (1). These conditions are of primal-dual type, which means that they are expressed in terms of primal variables θ and dual variables X, Y, which correspond to the Lyapunov variables used in the ARE and LMI solutions.

The classical solution of the H_∞-problem within $\mathcal{K}_{\text{full}}$ using AREs or LMIs for two Lyapunov matrix variables X, Y has the following particularity. The Lyapunov matrices X, Y can be interpreted as the dual variables of our own more general approach, while the primal variable, $K = (A_K, B_K, C_K, D_K)$, can be eliminated. In this very specific case the problem is convex in (X, Y).

The second major challenge was to find algorithmic tools to compute solutions of the structured H_∞-problem (1). The objective being nonconvex and nonsmooth, we had to develop new optimization methods. This was started in [2], and continued in [8–13]. We invented bundle methods for the nonconvex case. The bundle technique originated in the 1980s and is the most successful approach to deal with *convex* nonsmooth problems in Lagrangian relaxation or stochastic control. We succeeded to extend this to nonconvex functions, which represents a major breakthrough. When our new ideas, which were first published in control journals, made their appearance in a more digestible form in math journals, they were also adapted by the optimization community.

Flowchart of proximity control algorithm



2.2 Stabilization

As we stressed before, the objective $f(\theta)$ in (1), respectively (12), is only defined on the set

$$D_s = \{\theta \in \mathbb{R}^n : A(K(\theta)) \text{ is stable}\}$$

from (13). Our optimization method therefore not only has to *iterate* within this set, we first have to *find* a feasible parameter $\theta \in D_s$. Surprisingly, this is already the first difficulty.

Notice that we have to answer the following yes-or-no question:

(14) *Does there exist θ such that $A(K(\theta))$ is stable ?*

We want an algorithm which computes such a θ if the answer to (14) is "yes", and provides a certificate of emptiness of D_s if the answer is "no". And we would like these answers reasonably fast, e.g., in polynomial time.

How is this related to Kalman's classical theory of stabilizability, detectability, controllability and observability? Stabilizability of (A, B) means that we can stabilize by state feedback. And detectability of (A, C) means that we can add an observer. Therefore, if (A, B) is stabilizable and (A, C) is detectable, then the answer to question (14) is "yes" for the class $\mathcal{K}_{\rm obs}$ of observer-based controllers. Since stabilizability of (A, B) and detectability of (A, C) are conditions which can be checked by linear algebra (in polynomial time), we can say that (14) is conveniently decidable for the class of observer-based controllers $\mathcal{K}_{\rm obs}$ and for any larger class.

However, and this is the bad part of the message, for practically important controller structures $K(\theta)$ the decision (14) is NP-complete. Blondel and Tsitsiklis [14] prove this for the classes $\mathcal{K}_k$ of reduced-order controllers, including the class $\mathcal{K}_{\rm stat}$ of static controllers, and for the class $\mathcal{K}_{\rm dec}$ of decentralized controllers. It is also known the the decision is hard for PID control. For short, the most important classes in practice lead already to a difficult problem when it comes to mere stabilization.

Deciding whether a stabilizing controller $K(\theta)$ with a given structure exists is in general NP-complete.

What does this mean in practice? Complexity theory usually produces pessimistic results. The situation is by no means hopeless. Practical systems are designed to be stabilizable, so as a rule there is a good chance to find a stabilizing structured controller $K \in \mathcal{K}$ if there is one. What we expect to be hard is a certificate of non-existence when no such controller exists, because this requires an exhaustive search. Complexity also tells us that we cannot expect a linear algebra procedure as in Kalman's classical theory, at least none with polynomial complexity. We also know that for most classes $\mathcal{K}$ problem (14) is decidable, but in exponential time. This follows for instance as soon as the problem can be transformed into a polynomial decision problem, to which the Tarski-Seidenberg procedure can be applied.

2.3 Local versus global optimization

The fact that program (1) is nonconvex for practical controller structures $\mathcal{K}$ creates a dilemma. Should we go for a globally optimal solution, or should we be modest and be content with locally optimal solutions? In our approach we have opted for the local approach, as it is more realistic. This does not mean that we advise against the use of global optimization techniques. Such techniques might prove successful for small to medium size problems.

There is, however, one specific global approach on which we wish to comment, because it has contributed substantially to the field of mathematical poppycock. We are speaking about the so-called sums-of-squares (SOS) approach, which is still rumored to be suited for control problems like (1). That this is a red herring we now argue.

For most controller structures $\mathcal{K}$ it is possible to transform program (1) into a bilinear matrix inequality (BMI) using the bounded real lemma. Typically, the BMI is of the form

$$\min\{c^\top\theta : B(\theta, X, Y) \preceq 0\}, \tag{15}$$

featuring controller gains θ and Lyapunov variables X, Y as unknowns. The SOS approach interprets (15) as a systems of polynomial inequalities and uses the sums-of-squares approximation of positive polynomials to creates a hierarchy of LMI problems

$$\min\{c^\top \theta : L_i(\theta, X, Y) \preceq 0\} \tag{16}$$

with the property that the solution of (16) converges to the solution of (15). It may even happen that convergence is *finite*, meaning that there exists $i = i(B)$ such that the solution of $\min\{c^\top\theta : L_{i(B)} \preceq 0\}$ solves $\min\{c^\top\theta : B \preceq 0\}$ globally. The way this hierarchy is constructed is much inspired by the idea of a cutting plane proof for an linear integer feasibility problem $Ax \leq b$, $x \in \mathbb{Z}^n$.

Let us for simplicity assume that convergence is finite indeed. Then we might be able to write down an explicit linear matrix equality

$$\min\{c^\top \theta : L_{i(B)}(\theta, X, Y) \preceq 0\}, \tag{17}$$

which when solved gives a *globally optimal solution* of (1). (Strictly speaking, we might not be able to write down (17) directly, but only to reach it eventually by climbing up in the hierarchy until we get to $i(B)$. This would of course spoil the whole idea. But let us assume, as is often claimed in the SOS community, that we *can* write down (17) explicitly!).

Doesn't this sound nice? After all we have been told since the early 1990s that LMIs can be solved efficiently in quasi-polynomial time. So all we have to do is solve (17) quickly and get the global minimum of (15), respectively, of (1).

Of course this is all rubbish. We *know* that solving problem (1) globally is NP-complete. The SOS algorithm is even provably exponential. The size of $L_{i(B)} \preceq 0$ grows therefore exponentially in the data size(B). In fact, these problems explode extremely fast. We will need exponential space even to write down $L_{i(B)} \preceq 0$. For sizable plants we might not even be able to *store* the problem in the computer, let alone solve it. The fact that LMIs are solved in polynomial time is pointless, because we are speaking about a problem of size *polynomial*(*exponential*).

But could not something similar be said about *every* global method? Are we too severe when we call SOS a red herring? Indeed, the problem being NP-complete, *every* global method is bound to be exponential. The point is that SOS is a particularly ungainly global method, because it commits two errors, which other global methods may avoid.

The first error is that it transforms (1) to a BMI. This adds a large number of additional variables X, Y, which can be avoided e.g. by our nonsmooth approach. We have demonstrated abundantly that the presence of Lyapunov variables leads to serious ill-conditioning. To wit:

The power oscillation damping control problem which we solved in [15] using nonsmooth optimization has a system with 90 states, 3 performance connections, 1 input, 1 output, and a controller of reduced order 8. Therefore $\dim(\theta) = 81$***. Transformed to a BMI it requires additional*** $3 \cdot \frac{90\cdot 91}{2} = 12285$ ***Lyapunov variables. For the SOS approach this is just the bottom line*** $i = 1$***, where the LMI hierarchy starts. The LMI*** $L_{i(B)} \preceq 0$ ***will be of size*** exponential(12366).

The second error in the SOS approach is that it only goes for global minima. That is, it will *not* find local minima of (1) on its way toward the global. This is infelicitous, because local minima are very helpful. They may allow to improve bounds in branch-and-bound methods, and they give good practical solutions as a rule. The fact that SOS does not use this information (e.g. to infer where it is in the hierarchy $L_i \preceq 0$) is by itself already suspicious.

3 The H_2/H_∞-problem is also solved

It became already apparent in the 1-DOF scheme (8) that the L^2-L^2, respectively power-to-power, operator norm is not the only possible measure of smallness in a channel $w \to z$. Consider for instance the transfer $T_{n_s\to\widetilde{u}}$ from sensor noise n_s to the the high frequency part $\widetilde{u} = W_u u$ of the control law u. If we model n_s as white noise, then it makes sense to gauge $n_s \to \widetilde{u}$ by the operator norm from white noise at the input toward power at the output. This is the H_2-norm. For a stable transfer operator $G(s)$ the H_2-norm is given as

$$\|G\|_2 = \left(\int_0^\infty \mathrm{Tr}\left(G(j\omega)G^H(j\omega)\right) d\omega \right)^{1/2},$$

which makes it an Euclidian norm in the space of stable transfer matrices. Unlike the H_∞-norm, the H_2-norm is *not* an operator norm in the traditional sense. It becomes one as soon as stochastic signals are considered.

$\|w\|$	$\|z\|$	operator norm $\|T_{w\to z}\|$
energy	energy	H_∞
power	power	H_∞
white noise	power	H_2
Sobolev $W^{\infty,\infty}$	L^∞	worst case response norm
L^∞	L^∞	peak gain
past excitation	system ring	Hankel

In scheme (8) we might decide to use two different norms. We might assess the tracking error $r \to \widetilde{e}$ in the H_∞-norm, and the influence of sensor noise on the control $n_s \to \widetilde{u}$ by the H_2-norm. Then we obtain a mixed H_∞/H_2-control problem

$$\begin{array}{ll} \text{minimize} & \|T_{r\to\widetilde{e}}(P,K)\|_\infty \\ \text{subject to} & \|T_{n_s\to\widetilde{u}}(P,K)\|_2 \leq \gamma_2 \\ & K \text{ stabilizes } P \text{ internally} \\ & K = K(\theta) \text{ has a fixed structure} \end{array} \tag{18}$$

where γ_2 is some threshold limiting the energy of $\widetilde{u}$ in response to white noise in the input n_s. We may introduce the following more abstract setting. Consider a plant in state space form

$$P: \quad \begin{bmatrix} \dot{x} \\ \hline z_\infty \\ z_2 \\ y \end{bmatrix} = \left[\begin{array}{c|ccc} A & B_\infty & B_2 & B \\ \hline C_\infty & D_\infty & 0 & D_{\infty u} \\ C_2 & 0 & 0 & D_{2u} \\ C & D_{y\infty} & D_{y2} & 0 \end{array}\right] \begin{bmatrix} x \\ \hline w_\infty \\ w_2 \\ u \end{bmatrix} \tag{19}$$

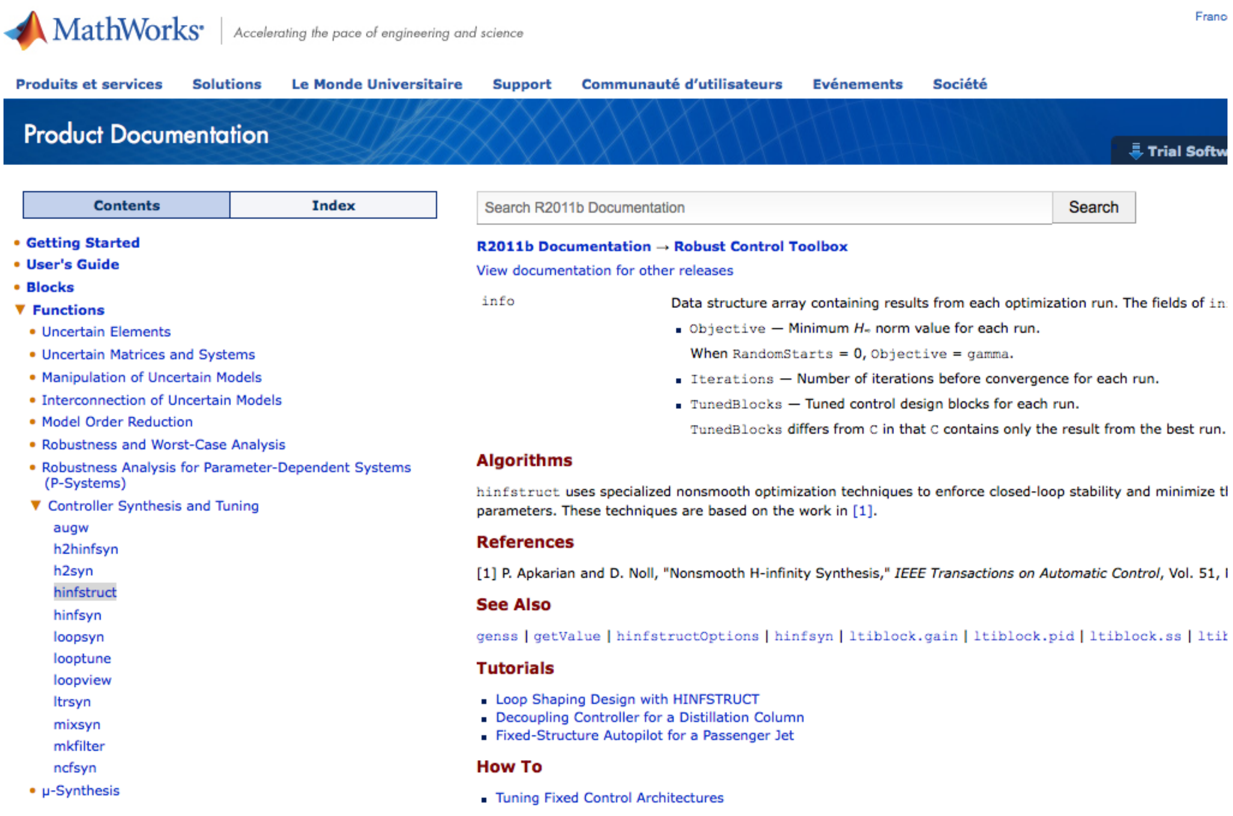
where $x \in \mathbb{R}^{n_x}$ is the state, $u \in \mathbb{R}^{n_u}$ the control, $y \in \mathbb{R}^{n_y}$ the output, and where $w_\infty \to z_\infty$ is the H_∞, $w_2 \to z_2$ the H_2 performance channel. This bring us to the following structured mixed H_2/H_∞-synthesis problem.

$$
\begin{array}{ll}
\text{minimize} & \|T_{w_2\to z_2}(P,K)\|_2 \\
\text{subject to} & \|T_{w_\infty\to z_\infty}(P,K)\|_\infty \leq \gamma_\infty \\
 & K \text{ stabilizes } P \text{ internally} \\
 & K \in \mathcal{K}
\end{array}
\tag{20}
$$

where $\mathcal{K}$ is a structured controller space as before, and γ_∞ is a suitable threshold, now for the H_∞-norm in the constraint. Notice that the H_2/H_∞- and H_∞/H_2-problems are equivalent under suitable choices of γ_2 and γ_∞.

The mixed H_2/H_∞-synthesis problem with structured controllers $K(\theta)$ is a natural extension of H_∞-control. This problem has also a long history. It was posed for the first time by Haddad and Bernstein [16] and by Doyle, Zhou, Bodenheimer [17] in 1989. We solved this problem in 2008 [18].

One may immediately think about other multi-objective extensions of (1). For instance, combining the H_∞-norm with time-domain constraints like in IFT, or H_∞/H_∞-control. For the first theme we refer the reader to our solution presented in [19, 20], while H_∞/H_∞-control will be addressed in the next section.


MathWorks | Accelerating the pace of engineering and science

Franc

Produits et services | Solutions | Le Monde Universitaire | Support | Communauté d'utilisateurs | Evénements | Société

Product Documentation

Trial Softw

Contents | Index

Search R2011b Documentation | Search

- Getting Started
- User's Guide
- Blocks
- Functions
  - Uncertain Elements
  - Uncertain Matrices and Systems
  - Manipulation of Uncertain Models
  - Interconnection of Uncertain Models
  - Model Order Reduction
  - Robustness and Worst-Case Analysis
  - Robustness Analysis for Parameter-Dependent Systems (P-Systems)
  - Controller Synthesis and Tuning
    - augw
    - h2hinfsyn
    - h2syn
    - hinfstruct
    - hinfsyn
    - loopsyn
    - looptune
    - loopview
    - ltrsyn
    - mixsyn
    - mkfilter
    - ncfsyn
  - μ-Synthesis

R2011b Documentation → Robust Control Toolbox

View documentation for other releases

`info` Data structure array containing results from each optimization run. The fields of `in`

- `Objective` — Minimum H_∞ norm value for each run.
  When `RandomStarts` = 0, `Objective` = `gamma`.
- `Iterations` — Number of iterations before convergence for each run.
- `TunedBlocks` — Tuned control design blocks for each run.
  `TunedBlocks` differs from `C` in that `C` contains only the result from the best run.

Algorithms

`hinfstruct` uses specialized nonsmooth optimization techniques to enforce closed-loop stability and minimize t
parameters. These techniques are based on the work in [1].

References

[1] P. Apkarian and D. Noll, "Nonsmooth H-infinity Synthesis," *IEEE Transactions on Automatic Control*, Vol. 51,

See Also

`genss` | `getValue` | `hinfstructOptions` | `hinfsyn` | `ltiblock.gain` | `ltiblock.pid` | `ltiblock.ss` | `lti`

Tutorials

- Loop Shaping Design with HINFSTRUCT
- Decoupling Controller for a Distillation Column
- Fixed-Structure Autopilot for a Passenger Jet

How To

- Tuning Fixed Control Architectures

4 How to use `hinfstruct`

The new Matlab function `hinfstruct` based on our seminal paper [2] allows a large variety of practical applications. This section presents several examples which will motivate the interested user to integrate structured H_∞-synthesis into his or her menu of control design methods. For more information on how to use `hinfstruct` see also

http://pierre.apkarian.free.fr/NEWALGORITHMS.html
http://www.math.univ-toulouse.fr/~noll/
http://www.mathworks.fr/help/toolbox/robust/ref/hinfstruct.html

4.1 Controller in state-space

The most general form to represent a controller $K(\theta)$ is parametrized in state-space. This is just according to Definition 1. The Matlab R2011b documentation of the Robust Control Toolbox gives the simple example

$$(21) \qquad K(\theta) = \left[\begin{array}{cc|c} 1 & a+b & -3.0 \\ 0 & ab & 1.5 \\ \hline 0.3 & 0 & 0 \end{array}\right], \qquad \begin{aligned} \dot{x}_1 &= x_1 + (a+b)\,x_2 - 3.0y \\ \dot{x}_2 &= ab\,x_2 + 1.5y \\ u &= 0.3x_1 \end{aligned}$$

where $\theta = (a, b) \in \mathbb{R}^2$ is what is called the vector of tunable parameters, and what in the optimization program (1) are the unknown variables. The commands to define this structure are

```
a = realp('a',-1);          % a is a parameter initialized as -1
b = realp('b',3);
A = [ 1 a+b ; 0 a*b ];
B = [ -3.0 ; 1.5 ];  C = [ 0.3 0 ];  D = 0;
Ksys = ss(A,B,C,D);
```

4.2 The H_∞/H_∞-control problem

It is important that the state-space structure includes the possibility of repetitions of the θ_i. For instance, in (21) both a and b are repeated. This allows us to solve H_∞ programs with several channels. For instance, the mixed H_∞/H_∞-problem can be seen as a special case of (1). Suppose we have two plants P_1 and P_2 with performance channels $w_i \to z_i$, $i = 1, 2$. Assume that the outputs y_i and inputs u_i into P_i have the same dimension, i.e., $\dim(y_1) = \mathrm{dom}(y_2)$ and $\dim(u_1) = \dim(u_2)$. Then we may connect the same controller $u_i = K(\theta)y_i$ to both plants simultaneously. That is, we may solve a program of the form

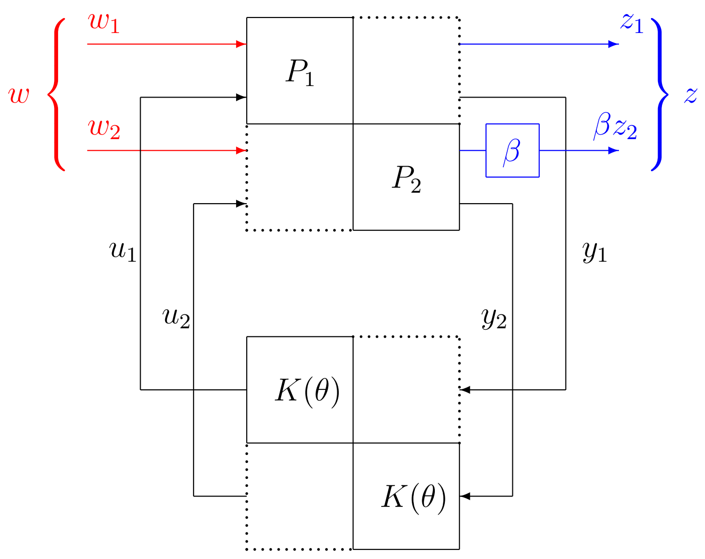
$$(22) \qquad \begin{array}{ll} \text{minimize} & \|T_{w_1\to z_1}(P_1, K)\|_\infty \\ \text{subject to} & \|T_{w_2\to z_2}(P_2, K)\|_\infty \le \gamma_2 \\ & K \text{ stabilizes } P_1 \text{ and } P_2 \\ & K = K(\theta) \text{ is structured} \end{array}$$

It turns out that we may transform (22) favorably into a program of the form

$$\begin{array}{ll} \text{minimize} & \max\{T_{w_1\to z_1}(P_1,K(\theta))\|_\infty, \beta\|T_{w_2\to z_2}(P_2,K(\theta))\|_\infty\} \\ \text{subject to} & K(\theta) \text{ stabilizes } P_1 \text{ and } P_2 \end{array} \tag{23}$$

which is sometimes called a multi-disk problem [8]. For suitable choices of γ_2 and β these two programs are equivalent. However, since the maximum of two H_∞-norms is again an H_∞-norm of an augmented plant, we can solve (23) directly via (1) with a new specific structure, which consist in repeating $K(\theta)$. Schematically

(24)



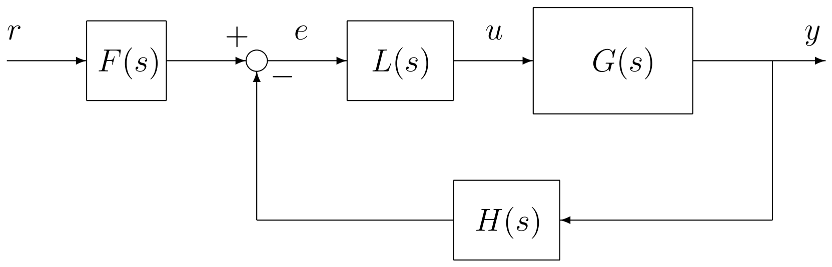
and the only connection between the two diagonal parts is the fact that the diagonal block of K is repeated. The objective of (23) is then the channel $w = (w_1, w_2) \to z = (z_1, \beta z_2)$ of the augmented plant. We may now have to update β in order to solve (24) for a specific γ_2.

4.3 Nonstandard use of H_∞/H_∞-synthesis

The standard way to use multiple H_∞ criteria is certainly in H_∞-loopshaping, and the documentation of `hinfstruct` makes this a strong point. However, there are some less obvious ideas in which one can use a program of the form (22). Two heuristics for parametric robust control, which we proposed in [21] and [22], can indeed be solved via `hinfstruct`.

4.4 Controller as transfer function

In many situations it may be preferable to avoid state-space representations of K and use the transfer function directly. Consider the following situation

(25) 

Suppose $G(s)$ and $H(s)$ are given blocks, $G(s) = \frac{1}{(s+1)^2}$ and $H(s) = \frac{5}{s+4}$. The unknown controller $K(s)$ regroups the block $L(s)$ and the prefilter $F(s)$. Let us say

$$L(s) = k_p + \frac{k_i}{s} + \frac{k_d s}{1 + T_f s}, \qquad F(s) = \frac{a}{s+a},$$

where $\theta = (a, k_p, k_i, k_d, T_f)$ is the vector of tunable parameters. Then we have the following commands to set up the controller

```
   G = tf(1,[1 2 1]);
   H = tf(5,[1 4]);
   a = realp('a',10);
   F = tf(a, [1 a ]);
   kp = realp('kp',0);
   ki = realp('ki',0);
   kd = realp('kd',0);
   Tf = realp('Tf',1);
   L = tf([kd+Tf ki*Tf+1 ki],[Tf 1 0 ] );
   T = feedback(G*L,H)*F;
```

Or we may recognize K to be a PID controller, which allows us to use a predefined structure under the form

```
   L = ltiblock.pid('L','PID');
```

The command

```
   T.Blocks
```

will show the difference. The controller K consisting of the blocks F and L could also be written in the standard form (2) as follows. Introduce

$$\widetilde{r} = Fr, \quad \widetilde{y} = Hy, \quad e = \widetilde{r} - \widetilde{y}, \quad u = Le, \quad y = Gu$$

in (25), then

$$F : \left\{ \begin{array}{rcl} \dot{\xi}_3 & = & -a\xi_3 + a\,r \\ \widetilde{r} & = & \xi_3 \end{array} \right. \qquad F(s) : \left[\begin{array}{c|c} -a & a \\ \hline 1 & 0 \end{array} \right]$$

and

$$L : \left\{ \begin{array}{rcrrr} \dot{\xi}_1 & = & & & r_i e \\ \dot{\xi}_2 & = & & -\tau\xi_2 & + r_d e \\ u & = & \xi_1 & + \xi_2 & + d\,e \end{array} \right. \qquad L(s) : \left[\begin{array}{cc|c} 0 & 0 & r_i \\ 0 & -\tau & r_d \\ \hline 1 & 1 & d \end{array} \right]$$

with the relations (9), (10). So together the controller K with three states ξ_1, ξ_2, ξ_3, inputs $\widetilde{y}$ and r, and output u, could be represented in state-space as

$$K(\theta):\left[\begin{array}{ccc|cc} 0 & 0 & r_i & -r_i & 0 \\ 0 & -\tau & r_d & -r_d & 0 \\ 0 & 0 & -a & 0 & a \\ \hline 1 & 1 & d & -d & 0 \end{array}\right]$$

where $u(s) = K(\theta, s)(\widetilde{y}(s), r(s))^T$. Here we have switched to $\theta = (a, \tau, r_i, r_d, d)$, which is equivalent to $\theta = (a, k_p, k_i, k_d, T_f)$ via (9), (10). Notice again that in both representations some of the controller gains are repeated. The corresponding plant P is found as follows. We have

$$H:\left\{\begin{array}{rcl} \dot{x}_3 & = & -4x_3 + 5y \\ \widetilde{y} & = & x_3 \end{array}\right.$$

and

$$G:\left\{\begin{array}{rcl} \dot{x}_1 & = & x_2 \\ \dot{x}_2 & = & -x_1 - 2x_2 + u \\ y & = & x_1 \end{array}\right.$$

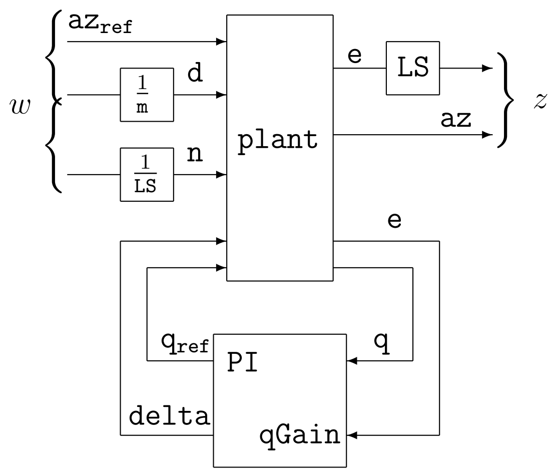
Therefore

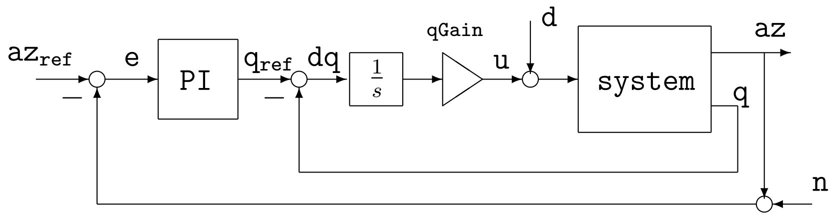
$$P:\left\{\begin{array}{rcl} \dot{x}_1 & = & x_2 \\ \dot{x}_2 & = & -x_1 - 2x_2 + u \\ \dot{x}_3 & = & 5x_1 - 4x_3 \\ \widetilde{y} & = & x_3 \end{array}\right.$$

to which we would add a performance channel $w \to z$.

4.5 Flight control 1

An example discussed in [23] and as `rct_airframe1` in the documentation of `hinfstruct` concerns flight control of an aircraft. The control architecture is given by the following scheme





Having extracted the plant from the simulink model, one marks the inputs and outputs

```
io = getlinio('rct_airframe1');
TunedBlocks = {'rct_airframe1/az Control';'rct_airframe1/q Gain'};
P = linlft('rct_airframe1',io,TundedBlocks);
P.InputName = {'azref','d','n','qref','delta' };
P.OutputName = {'e','az','ePI','qInt'};
Pdes = blkdiag(LS,eye(3)) * P * blkdiag(1/LS,1/m,eye(3));
PI0 = ltiblock.pid('azControl','pi');
qGain0 = ltiblock.gain('qGain',0);
```

After defining options, one runs

```
C = hinfstruct(Pdes,{PI0,qGain0},opt);
```

The information is retrieved by

```
PI = pid(C{1});
qGain = ss(C{2});
```

and the closed-loop LFT is obtained from

```
CL = lft(P,blkdiag(PI,qGain));
```

In this example it is trivial to obtain the controller in state-space. We have

$$(\texttt{PI}) \quad \begin{array}{rcl} \dot{x}_1 & = & k_i e \\ q_{\text{ref}} & = & x_1 + k_p e \end{array}$$

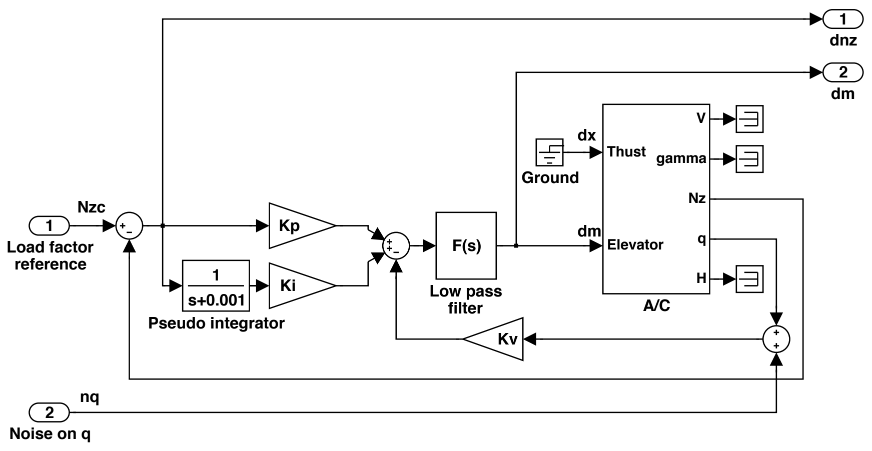
for the PI-block. Then

$$(\texttt{qGain}) \quad \begin{array}{rcl} \dot{x}_2 & = & \texttt{qGain} \cdot dq \\ u & = & x_2 \end{array}$$

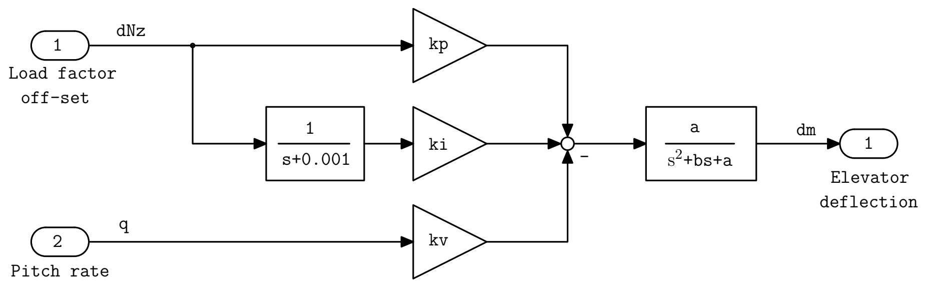
However, if more SISO controller blocks are combined, it may be preferable and more natural to work with transfer functions.

4.6 Flight control 2

A more interesting situation is the following control architecture from [24], also in the domain of flight control, where a PI-block, a gain, and a filter are combined. The overall control architecture is as follows



Extracting the controller gives the following structure, where the tunable parameters are $\theta = (k_p, k_i, k_v, a, b)$. Notice the novelty here, we are considering the filter as unknown and therefore as part of the controller. Standard procedures would design the filter first and then tune k_i, k_p, k_v. When various elements of this type are combined, we speak about a *control architecture.*



Now if we write down the state-space representation of this scheme in a straightforward way, then we may end up with

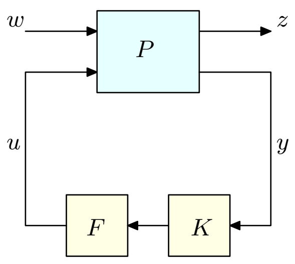
$$K(\theta):\left[\begin{array}{c}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \hline dm\end{array}\right]=\left[\begin{array}{ccc|cc}0&1&0&0&0\\ -a&-b&a&ak_p&-ak_v\\ 0&0&-0.001&k_i&0\\ \hline 1&0&0&0&0\end{array}\right]\left[\begin{array}{c}x_1\\ x_2\\ x_3\\ \hline dN_z\\ q\end{array}\right] \tag{26}$$

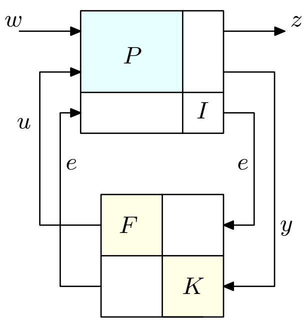
which is according to definition 1, but gives a nonlinear parametrization. If we augment the plant P artificially into a plant $\widetilde{P}$, we may obtain an equivalent affine parametrization of the controller:

$$K(\theta):\left[\begin{array}{c}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \hline dm\\ e\end{array}\right]=\left[\begin{array}{ccc|ccc}0&1&0&0&0&0\\ -a&-b&0&a&0&0\\ 0&0&-0.001&0&k_i&0\\ \hline 1&0&0&0&0&0\\ 0&0&1&0&k_p&-k_v\end{array}\right]\left[\begin{array}{c}x_1\\ x_2\\ x_3\\ \hline e\\ dN_z\\ q\end{array}\right] \tag{27}$$

This requires passing e through the plant as indicated by the following figure, which explains the meaning of the augmented plant $\widetilde{P}$.

$$P\colon \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \qquad \widetilde{P}\colon \left[\begin{array}{c|ccc} A & B_1 & 0 & B_2 \\ \hline C_1 & D_{11} & 0 & D_{12} \\ 0 & 0 & I & 0 \\ C_2 & D_{21} & 0 & D_{22} \end{array}\right]$$



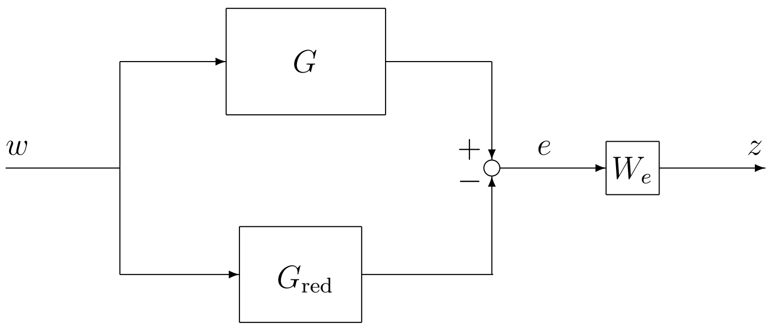


Notice that what we highlighted by red and blue in (27) is a decentralized controller structure, the one mentioned in section 2.3. Notice that if the problem is entered via the TF structure, the user will not notice the difference between (26) and (27). It should also be clear that every rational parametrization like (26) can be shuffled into an affine one using the same trick.

4.7 System reduction via `hinfstruct`

An idea already put forward in our paper [2] is H_∞-system reduction. Consider a stable system $G = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ with size$(A) = n \times n$. Suppose n is large and we want to compute a reduced stable system $G_{\rm red} = \left[\begin{array}{c|c} A_{\rm red} & B_{\rm red} \\ \hline C_{\rm red} & D_{\rm red} \end{array}\right]$ of smaller state dimension size$(A_{\rm red}) = k \ll n$ which represents G as accurately as possible.

(28)



The model matching error is $e = (G - G_{\rm red})w$, and after adding a suitable filter W_e, we might want to have $w \to z$ small in a suitable norm. The Hankel norm reduction method minimizes

$\|W_e(G - G_{\text{red}})\|_H$ in the Hankel norm $\|\cdot\|_H$, the advantage being that the solution can be obtained by linear algebra. A more natural norm would be the H_∞-norm, but the classical balanced reduction method gives only upper bounds of $\|W_e(G - G_{\text{red}})\|_\infty$.

But we can solve the H_∞-norm reduction problem directly as a special case of (1). In the case $z = e$ without filter we can pass to the standard form by considering the plant

$$P : \left[\begin{array}{c|cc} A & B & 0 \\ \hline C & D & -I \\ 0 & I & 0 \end{array}\right] = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & 0 \end{array}\right] \tag{29}$$

then G_{red} is the controller, which is of fixed reduced-order. The approach can be put to way as follows.

We have tested this with a 15th order Rolls-Royce Spey gas turbine engine model, decribed in [27, Chapter 11.8, p. 463]. The data are available for download on I. Postlethwaites's homepage as `aero0.mat`.

```
load aero0
G = G_eng;
A = G.a;
B = G.b;
C = G.c ;
D = G.d ;
```

Define the plant according to (29):

```
[nbout,nbin] = size(D);
Aplant = A;
Bplant = [ B zeros(n,nbout) ];
Cplant = [C
          zeros(nbin,n) ];
Dplant = [ D         -eye(nbout)
           eye(nbin)  zeros(nbin,nbout) ];
Plant = ss(Aplant,Bplant,Cplant,Dplant);
```

Now define the structure of the "controller", which in this case is nothing else but the reduced-order system G_{red} in (28). Let the reduced order be $k \leq n$. (In the example we have $n = 15$, $k = 6$.) Then

```
Red = ltiblock.ss('reduced',k,nbout,nbin);
```

Now we are ready to run `hinfstruct`. We could for instance do the following. Increase the maximum number of iterations to 900 (default is 300), and allow 6 random restarts.

```
Opt = hinfstructOptions('MaxIter',900,'RandomStart',6);
[Gred,gam,info] = hinfstruct(Plant,Red,Opt);
```

The result is the following output.

```
Final:  Peak gain = 0.17, Iterations = 743
Final:  Peak gain = 0.382, Iterations = 900
Final:  Peak gain = 0.808, Iterations = 553
```

```
Final:  Peak gain = 0.768, Iterations = 550
Final:  Peak gain = 0.65, Iterations = 422
Final:  Peak gain = 0.77, Iterations = 591
Final:  Peak gain = 0.236, Iterations = 900
Final:  Peak gain = 0.44, Iterations = 900
Final:  Peak gain = 0.169, Iterations = 794
Final:  Peak gain = 0.535, Iterations = 900
Final:  Peak gain = 0.794, Iterations = 538
Final:  Peak gain = 0.176, Iterations = 900
Final:  Peak gain = 0.638, Iterations = 561
Final:  Peak gain = 0.555, Iterations = 900
Final:  Peak gain = 0.43, Iterations = 788
Final:  Peak gain = 0.486, Iterations = 900
Final:  Peak gain = 0.169, Iterations = 845
Final:  Peak gain = 0.419, Iterations = 634
Final:  Peak gain = 0.49, Iterations = 900
Final:  Peak gain = 0.742, Iterations = 900
Final:  Peak gain = 0.169, Iterations = 758
```

As we can see, the best error is $\|G-G_{\rm red}\|_\infty = 0.169$, but various other local minima are found, so without testing several initial guesses (here at random), we could not rely on a single run. However, system reduction is a situation where we can do much better. Why not initialize the optimization using one of the standard reductions, like the Hankel norm reduction, or a balanced truncation? Here is how to do it.

```
[Gb,hsig]=balreal(G);
[Gh,HankInfo]=hankelmr(Gb,k);
Ah = Gh.a;
Bh = Gh.b;
Ch = Gh.c;
Dh = Gh.d;
```

So far we have the state-space form of the 6th order Hankel reduced model G_h. Now we initialize the tunable structure `Red` as this reduced-order model G_h. That is done via the structure `Value`.

```
Red.a.Value = Ah;
Red.b.Value = Bh;
Red.c.Value = Ch;
Red.d.Value = Dh;
Opt = hinfstructOptions('MaxIter',900);
[Gred,gam,info] = hinfstruct(Plant,Red,Opt);
```

This time the result is much more efficient. The output is

```
State-space model with 3 outputs, 3 inputs, and 6 states.
Final:  Peak gain = 0.169, Iterations = 25
```

That means, the seemingly global minimum with $\|G - G_{\rm red}\|_\infty = 0.169$ is reached very fast, and no random restarts are used.

This is a somewhat surprising application of* `hinfstruct`*, because we have an H_∞-filtering problem, not an H_∞-control problem. The plant P is not controllable, and it is stabilizable only because A is stable.

4.8 Control of nonlinear systems with structured H_∞-synthesis

In this section we discuss a somewhat unexpected application of structured H_∞-synthesis in the control of nonlinear systems. The class of systems we have in mind are of the form

$$P(y): \quad \begin{aligned} \dot{x} &= A(y)x + B_1(y)w + B_2(y)u \\ z &= C_1(y)x + D_{11}(y)w + D_{12}(y)u \\ y &= C_2(y)x + D_{21}(y)w + D_{22}(y)u \end{aligned} \tag{30}$$

where the system matrices depend smoothly on the measured output y. It appears therefore natural to devise a controller of the form

$$K(y): \quad \begin{array}{rcl} \dot{x}_K &=& A_K(y)x_K + B_K(y)y \\ u &=& C_K(y)x_K + D_K(y)y \end{array} \tag{31}$$

which uses the same measurement y available in real time. A natural idea, going back to [25], is to consider y like a time-varying external parameter p and pre-compute $K(p)$ for $P(p)$ for a large set $p \in \Pi$ of possible parameter values. In flight control for instance Π is the flight envelope $p = (h, V) \in \mathbb{R}^2$, indexed by altitude h and ground speed V, or sometimes by Mach number and dynamic pressure.

We now propose the following control strategy. In a first step we pre-compute the optimal H_∞ controller $K^*(p)$ for every $p \in \Pi$ using program (1):

$$\begin{array}{ll} \text{minimize} & \|T_{w\to z}\left(P(p), K\right)\|_\infty \\ \text{subject to} & K \text{ stabilizes } P(p) \text{ internally} \\ & K \in \mathcal{K} \end{array} \tag{32}$$

The solution $K^*(p)$ of (32) has the structure $\mathcal{K}$. In the terminology of [25] is the best way to control the system $P(p)$ frozen at $p(t) = y(t)$ instantaneously. In other words, at instant t, we apply the control law $K^*(y(t))$ based on the real-time measurement $y(t)$.

If we could do real-time structured H_∞-synthesis, then controller $K^*(y(t))$ would be computed and applied instantaneously at time t using (32) and the measurement $y(t)$ available at instant t. As long as this is impossible, we may pre-compute $K^*(p)$ for a large set of possible parameter values $p \in \Pi$, and as soon as $y(t)$ becomes available at time t, look $K^*(y(t))$ up in the table $\{K^*(p) : p \in \Pi\}$, and apply it instantaneously.

There are two limitations to this ideal approach. Firstly, the ideal table $\{K^*(p) : p \in \Pi\}$, computed by `hinfstruct`, may be too large. And secondly, the behavior of $K^*(p)$ as a function of p may be quite irregular. In fact, this is what has stopped this idea in the past[2]. With structured control laws $K(\theta)$ the situation is substantially improved, because one can use fewer degrees of freedom in θ.

What we have tested in [26] is a compromise between optimality of $K^*(p)$ in the sense of program (32), the necessity to avoid irregular behavior of the curves $p \mapsto K^*(p)$, and the storage requirement of such a law. We use the following definition. A controller parametrization $p \mapsto K(p)$ of the given structure $\mathcal{K}$ is *admissible* for the control of $P(y)$ if the following holds. $K(p)$ stabilizes $P(p)$ internally for every $p \in \Pi$, and

$$\|T_{w\to z}\left(P(p), K(p)\right)\|_\infty \leq (1+\alpha)\|T_{w\to z}\left(P(p), K^*(p)\right)\|_\infty \tag{33}$$

for every $p \in \Pi$, where α is some fixed threshold. Typically, $\alpha = 0.1\%$. We now seek a parametrization $K(p)$ which is close to the ideal H_∞-parametrization $K^*(p)$ in the sense that (33) is respected, but otherwise is easy to store (to embed) and shows as regular a behavior as possible. Notice that (33) allows $K(p)$ to lag behind $K^*(p)$ in performance by no more than $100\alpha\%$.

Acknowledgement

The authors acknowledge funding by Agence Nationale de Recherche (ANR) under grants *Guidage* and *Controvert*, by Fondation d'Entreprise EADS under grants *Solving Challenging Problems in Control* and *Technicom*, and by Fondation de Recherche pour l'Aéronautique et l'Espace under grant *Survol*.

References

[1] Zames, G. (1981). "Feedback and optimal sensitivity: model reference transformations, multiplicative seminorms, and approximate inverses," *IEEE Transactions on Automatic Control*, vol. AC-26, pp. 301-320.

[2] Apkarian, P., D. Noll (2006). "Nonsmooth H_∞-synthesis," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 229-244.

[3] Zames, G. (1966). "On the input-output stability of nonlinear time-varying feedback systems, part I and II," *IEEE Transactions on Automatic Control*, vol. AC-11, pp. 228 and 465.

[4] Doyle, J.C., K. Glover, P.P. Khargonekar, B.A. Francis (1989). "State-space solutions to standards H_2 and H_∞ control problems," *IEEE Transactions on Automatic Control*, vol. AC-34, no. 8, pp. 831-847.

[2]When ARE solvers were the only tools available to compute H_∞-controllers, the idea to embed such a solver into the system came obviously to mind. This failed not due to lack of speed, but due to the highly irregular behavior of $p \mapsto K^*_{\text{full}}(p)$.

[5] Doyle, J. C. (1996). "Robust and optimal control," *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, Conference Proceedings, pp. 1595 – 1598.

[6] Gahinet, P., P. Apkarian (1994). "A linear matrix inequality approach to H_∞ control". *International Journal of Robust and Nonlinear Control*, 4:421-448.

[7] D.C. Hyland, D.S. Bernstein. "The optimal projection equations for fixed-order dynamic compensation." *IEEE Trans. Autom. Control*, vol. AC-29, no. 11, pp. 1034-1037.

[8] P. Apkarian, D. Noll (2006b). "Nonsmooth optimization for multidisk H_∞ synthesis." *European Journal of Control*, vol. 12, no. 3, pp. 229 - 244.

[9] P. Apkarian, D. Noll (2007). "Nonsmooth optimization for multiband frequency domain control design." *Automatica*, vol. 43, no. 4, pp. 724 - 731.

[10] P. Apkarian, D. Noll, O. Prot (2008). "A trust region spectral bundle method for nonconvex eigenvalue optimization." *SIAM Journal on Optimization*, vol. 19, no. 1, pp. 281 – 306.

[11] D. Noll, O. Prot, A. Rondepierre (2008). "A proximity control algorithm to minimize nonsmooth and nonconvex functions." *Pacific Journal of Optimization*, vol. 4, no. 3, pp. 569-602.

[12] P. Apkarian, D. Noll, O. Prot (2009). "A proximity control algorithm to minimize nonsmooth and nonconvex semi-infinite maximum eigenvalue functions". *Journal of Convex Analysis*, vol. 16, no. 3 & 4, pp. 641 – 666.

[13] Noll, D. (2010). "Cutting plane oracles to minimize nonsmooth and nonconvex functions." *Journal of Set-Valued and Variational Analysis*, vol. 18, no. 3-4, pp. 531 – 568.

[14] V.D. Blondel, J.N. Tsitsiklis (1997). "NP-hardness of some linear control design problems", *SIAM J. of Control and Opt.,* 35:6, pp. 2118-2127.

[15] P. Apkarian, A. Simoes, D. Noll (2008), "A nonsmooth progress function algorithm for frequency shaping control design". *IET Control Theory & Applications,* vol. 2, no. 4, pp. 323 - 336.

[16] W. M. Haddad, D. S. Bernstein (1989). LQG control with a H_∞ performance bound: a Riccati equation approach. *IEEE Trans. Aut. Control*, AC-34(3):293–305.

[17] J.C. Doyle, K. Zhou, and B. Bodenheimer (1989). "Optimal control with mixed H_2 and H_∞ performance objectives". *Proceedings of the American Control Conference,* vol.3, pp. 2065–2070.

[18] P. Apkarian, D. Noll, A. Rondepierre (2008). "Mixed H_2/H_∞ control via nonsmooth optimization". *SIAM Journal on Control and Optimization*, vol. 47, no. 3, pp. 1516-1546.

[19] P. Apkarian, L. Hosseini-Ravanbod, D. Noll (2011). "Time domain constrained structured H_∞ synthesis". *International Journal of Robust and Nonlinear Control*, vol. 21, no. 2, pp. 197–217.

[20] P. Apkarian, D. Noll, A. Simões (2009). "Time domain control design: a nonsmooth approach". *IEEE Transactions on Control Systems Technology*, vol. 17, no. 6, 1439 – 1445.

[21] L. Hosseini-Ravanbod, D. Noll, P. Apkarian. (2011) "Robustness via structured H_∞/H_∞-synthesis". *International Journal of Control,* in press.

[22] P. Apkarian (2011). "Nonsmooth μ-synthesis", *International Journal of Robust and Nonlinear Control,* vol. 21(8), pp. 1493-1508.

[23] L. Hosseini-Ravanbod, D. Noll (2011). "Gain-scheduled autopilot for an aircraft". To appear.

[24] M. Gabarrou, D. Alazard, D. Noll (2010). "Structured flight control law design using nonsmooth optimization". *18th IFAC Symposium on Automatic Control in Aerospace* (ACA 2010), september 6-10, Nara, Japan.

[25] J.S. Shamma, M. Athans (1990). "Analysis of gain scheduled control for nonlinear plants", *IEEE Transactions on Automatic Control*, vol. 35, no. 4, pp. 898 – 907.

[26] L. Ravanbod-Hosseini, D. Noll (2011). "Gain-scheduled PID for imbalance compensation of a magnetic bearing". 8th *International Conference on Informatics, Automation and Robotics* (ICINCO 2011), Noordwijkerhout, The Netherlands, july 2011, Conference Proceedings, pp. 330 – 337.

[27] S. Skogestad, I. Postlethwaite. "Multivariable Feedback Control". Wiley 2005.