An Augmented Lagrangian Method for a Class of LMI-Constrained Problems in Robust Control Theory

B.Fares[*] *P. Apkarian*[‡] *D. Noll*[¶]

Abstract

We present a new approach to a class of non-convex LMI-constrained problem in robust control theory. The problems we consider may be recast as the minimization of a linear objective subject to linear matrix inequality (LMI) constraints in tandem with non-convex constraints related to rank conditions. We solve these problems using an extension of the augmented Lagrangian technique. The Lagrangian function combines a multiplier term and a penalty term governing the non-convex constraints. The LMI constraints, due to their special structure, are handled explicitly and not included in the Lagrangian. Global and fast local convergence of our approach is then obtained either by an LMI-constrained Newton type method including line search or by a trust-region strategy. This procedure may therefore be regarded as a *sequential semi-definite programming* (SSDP) method, inspired by the sequential quadratic programming (SQP) in nonlinear optimization. The method is conveniently implemented with available SDP interior-point solvers. We compare its performance to the well-known D-K iteration scheme in robust control. Two test problems are investigated and demonstrate the power and efficiency of our approach.

Key words: Nonlinear Programming, Semi-Definite Programming, Robust control, Linear Matrix Inequality approach.

1 INTRODUCTION

A large variety of problems in robust control can be cast as minimizing a linear objective subject to linear matrix inequality (LMI) constraints and additional nonlinear constraints which represent rank deficiency conditions. More formally, this can be stated as

$$\text{minimize } c^T x \tag{1}$$

$$\mathcal{L}(x) \leq 0, \tag{2}$$

$$\text{Rank}\, \mathcal{A}(x) = r, \tag{3}$$

[*]ONERA-CERT, Control Systems and Flight Mechanics Dept., 2 av. Edouard Belin, 31055 Toulouse, FRANCE - Email : fares@cict.fr

[‡]Corresponding author - ONERA-CERT, Control Systems and Flight Mechanics Dept., 2 av. Edouard Belin, 31055 Toulouse, FRANCE - Email : apkarian@cert.fr - Tel : +33 5.62.25.27.84 - Fax : +33 5.62.25.27.64

[¶]Université Paul Sabatier, Mathématiques pour l'Industrie et la Physique, 118, route de Narbonne, 31062 Toulouse, FRANCE - Email : noll@dumbo.ups-tlse.fr - Tel : +33 5.61.55.86.22 - Fax : +33 5.61.55.61.83

where c and r are given and x denotes the vector of decision variables. Inequality (2) represents LMI constraints, while (3) is a rank condition on $\mathcal{A}(x)$, with both $\mathcal{A}, \mathcal{L}$ affine matrix-valued functions of x. Synthesis problems that can be formulated as (1) - (3) are:

- fixed-order H_∞ synthesis,
- robust synthesis with different classes of scalings or multipliers,
- reduced-order linear parameter-varying (LPV) synthesis.

The rank condition (3) renders these synthesis problems [11] highly complex. Due to their practical importance, however, various heuristics and ad hoc methods have been developed in recent years to obtain solutions to these difficult problems. The $D-K$ iteration procedure is a popular example of this type, [7, 21]. Most currently used methods are based on *coordinate descent schemes* which alternatively and iteratively fix parts of the coordinates of the decision vector, while trying to optimize the remaining indices. This is conceptually simple and easily implemented as long as the intermediate steps are convex LMI programs. The latter may often be guaranteed through an appropriate choice of the decision variables held fixed at each step. However, a major drawback of coordinate descent schemes is that they may (and often will) fail to converge even for starting points close to a local solution. As a result, solutions obtained via such methods are highly questionable and bear the risk of additional conservatism in the synthesis task.

In this paper, we follow a quite different line of attack initiated in [5]. The rank constraints (3) are incorporated into an augmented Lagrangian function with a suitably defined penalty term and a term involving Lagrange multiplier variables. The LMI constraints (2), due to their infinite character, are treated explicitely and not included in the Lagrangian. Instead, the augmented Lagrangian function is minimized subject to these LMI constraints, using an increasing sequence of penalty parameters and a first-order update rule for the Lagrange multiplier estimates. At each step, the minimization of the augmented Lagrangian is performed either by a Newton type method including a line search or via a trust-region strategy. The entire scheme may be considered as a *sequential semi-definite programming* (SSDP) method which at each step requires solving a convex LMI program. It therefore lends itself to currently available LMI solvers [15] based on semi-definite programming (SDP). Even though more sophisticated than most coordinate descent schemes, the advantages of the new approach are at hand:

- The decision variables do not have to be treated separately. The entire vector x of decision variables is updated at each step.
- The method, being of descent type, is guaranteed to converge globally, that is, to a local minimum from any feasible, even remote, starting point. Moreover, the rate of convergence is at least linear.

From a control theory viewpoint, the first observation is important since it means that there is no need to separate Lyapunov and scaling variables from control variables. All these parameters are processed jointly during the iteration.

In this paper, we focus on the robust synthesis problem which, in a sense, is the most difficult among the problems mentioned above, with rank constraint of the form $\mathrm{Rank}\mathcal{A}(x) = 0$. The

paper is organized as follows. Section 2 recalls the setting of the robust control problem. Section 3 gives a detailed description of the augmented Lagrangian method. The trust-region technique which may as an option replace the Newton step is discussed in Section 3.4. Numerical aspects of the algorithms are presented in Section 4.

The notation used throughout the paper is fairly standard. $\mathcal{S}^n$ denotes the set of $n \times n$ symmetric matrices. M^T is the transpose of the matrix M. The notation $\mathrm{Tr}\, M$ stands for the trace of M. For Hermitian or symmetric matrices , $M > N$ means that $M - N$ is positive definite and $M \geq N$ means that $M - N$ is positive semi-definite. We shall use $\mathrm{Herm}\, \mathrm{M}$ for $1/2(M + M^T)$. The notation $\mathrm{co}\{p_1, \cdots, p_L\}$ stands for the convex hull of the set $\{p_1, \cdots, p_L\}$. In symmetric block matrices or long matrix expressions, we use $*$ as an ellipsis for terms that are induced by symmetry. The operator svec which maps the set of symmetric matrices $\mathcal{S}^n$ into R^ℓ where $\ell = n(n+1)/2$ is defined as :

$$\mathrm{svec}(X) = (X_{11}, \cdots, X_{1n}, X_{22}, \cdots, X_{2n}, \cdots X_{nn})^T$$

By introducing the diagonal matrix T defined as:

$$T := \mathrm{diag}(1, \sqrt{2}, \ldots, \sqrt{2}, 1, \sqrt{2}, \ldots, 1) \ \in \ \mathbf{R}^{\ell \times \ell},$$

where the unit entries correspond to diagonal terms of X whereas entries $\sqrt{2}$ are associated with terms strictly above diagonal, we can define the operator $\circledast$ which generalizes the usual Kronecker product $\otimes$ to the set of symmetric matrices and is compatible with the inner product of symmetric matrices [1]. We have the properties,

- $\Big[M \circledast N\Big]\, T\, \mathrm{svec}(X) = \ T\, \mathrm{svec}\Big[\frac{1}{2}\,(NXM^T + MXN^T)\Big]$.
- $\mathrm{Tr}\, XY = \mathrm{svec}^{\,T} X\, T^2\, \mathrm{svec}\, Y$.

Finally, the gradient of a real-valued function $f(x)$ is denoted $\nabla f(x)$ and its Hessian $\nabla^2 f(x)$. Throughout the paper we shall use the fact that a convex quadratic problem can be recast as an LMI program. Namely, the convex quadratic minimization

$$\overline{x} = \underset{x}{\mathrm{argmin}} \left\{ g^T x + x^T H x \right\}$$

is equivalent to solving an SDP problem

$$\min \left\{ t \ : \ \begin{pmatrix} t - g^T x & x^T \\ x & H^{-1} \end{pmatrix} \geq 0 \right\}.$$

2 ROBUST CONTROL SYNTHESIS

This section gives a brief review of a basic result in control which we exploit throughout the paper. We are concerned with the synthesis of a robust controller for an uncertain plant subject to structured parametric linear fractional transformation (LFT) uncertainties

Consider the uncertain plant governed by:

$$\begin{pmatrix} \dot{x} \\ z_\theta \\ z \\ y \end{pmatrix} = \begin{pmatrix} A & B_\Theta & B_1 & B_2 \\ C_\Theta & D_{\Theta\Theta} & D_{\Theta 1} & D_{\Theta 2} \\ C_1 & D_{1\Theta} & D_{11} & D_{12} \\ C_2 & D_{2\Theta} & D_{21} & 0 \end{pmatrix} \begin{pmatrix} x \\ w_\theta \\ w \\ u \end{pmatrix} \tag{4}$$

$$w_\Theta = \Theta(t) z_\Theta$$

where $\Theta(t)$ is a time-varying matrix-valued parameter ranging over a polytope $\mathcal{P}$, i.e.,

$$\Theta(t) \in \mathcal{P} = \mathrm{co}\,\{\Theta_{v_1}, \ldots, \Theta_{v_N}\}, \qquad \forall t \geq 0\,, \tag{5}$$

with Θ_{v_i}'s the vertices of the polytope $\mathcal{P}$. Straightforward computations lead to the state-space form with LFT plant

$$\begin{pmatrix} \dot{x} \\ z \\ y \end{pmatrix} = \left\{ \begin{pmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & 0 \end{pmatrix} + \begin{pmatrix} B_\Theta \\ D_{1\Theta} \\ D_{2\Theta} \end{pmatrix} \Theta(t) \Big(I - D_{\Theta\Theta}\Theta(t) \Big)^{-1} \begin{pmatrix} C_\Theta & D_{\Theta 1} & D_{\Theta 2} \end{pmatrix} \right\} \begin{pmatrix} x \\ w \\ u \end{pmatrix}. \tag{6}$$

The plant with inputs w, u and outputs z, y has state-space entries which are fractional functions of the time-varying parameter $\Theta(t)$, hence the name. The meaning of the signals is the following:

- u is the control input,
- w is the vector of exogenous signals,
- z is the vector of regulated variables,
- y is the measurement signal.

The synthesis task for the uncertain plant (4) is now to find a linear time-invariant (LTI) controller

$$\begin{cases} \dot{x}_K = A_K x_K + B_K y \\ u = C_K x_K + D_K y \end{cases} \tag{7}$$

such that

- the closed-loop system is internally stable,
- the L_2-induced gain of the closed-loop operator mapping w to z is bounded by γ.

Moreover, the above specifications must hold for all parameter trajectories $\Theta(t)$ defined by (5), hence the robustness of the device.

It is now well-known that such problems can be handled via a suitable generalization of the Bounded Real Lemma which translates these items into the existence of a Lyapunov matrix $X_{c\ell}$

and scalings Q, S, R such that $X_{c\ell} > 0$ and

$$\begin{pmatrix} A_{c\ell}^T X_{c\ell} + X_{c\ell} A_{c\ell} & * & * \\ B_{c\ell}^T X_c + \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} C_{c\ell} & \begin{pmatrix} Q & 0 \\ 0 & -\gamma I \end{pmatrix} + \left(\begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} D_{c\ell} + * \right) & * \\ C_{c\ell} & D_{c\ell} & -\begin{pmatrix} R^{-1} & 0 \\ 0 & \gamma I \end{pmatrix} \end{pmatrix} < 0 \qquad (8)$$

where the scalings Q, R and S must satisfy the LMI constraints :

$$\begin{pmatrix} \Theta_{v_i} \\ I \end{pmatrix}^T \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \begin{pmatrix} \Theta_{v_i} \\ I \end{pmatrix} > 0, \quad i = 1, \cdots, N; \qquad \begin{pmatrix} -Q & 0 \\ 0 & R \end{pmatrix} > 0 \,. \qquad (9)$$

The state-space data $A_{c\ell}, B_{c\ell}, C_{c\ell}, D_{c\ell}$ determine the closed-loop system (4) and (7) with the loop $w_\Theta = \Theta(t) z_\Theta$ open, that is,

$$\begin{cases} \begin{pmatrix} \dot{x} \\ \dot{x}_K \end{pmatrix} &= A_{c\ell} x_{c\ell} + B_{c\ell} \begin{pmatrix} w_\theta \\ w \end{pmatrix} \\ \begin{pmatrix} z_\theta \\ z \end{pmatrix} &= C_{c\ell} x_{c\ell} + D_{c\ell} \begin{pmatrix} w_\theta \\ w \end{pmatrix}. \end{cases} \qquad (10)$$

The Bounded Real Lemma conditions (8) are further simplified by means of the Projection Lemma [14, 19], and we obtain the following characterization amenable to numerical computations.

Theorem 2.1 *Consider the LFT plant (4) where Θ is ranging over the polytope $\mathcal{P}$ defined in (5). Let $\mathcal{N}_X$ and $\mathcal{N}_Y$ denote any bases of the null spaces of $(C_2, D_{2\Theta}, D_{21}, 0)$ and $(B_2^T, D_{\Theta 2}^T, D_{12}^T, 0)$, respectively. Then the existence of a controller (7) such that the closed-loop system is well posed and the above Bounded Real Lemma conditions hold with L_2-gain performance γ is guaranteed if there exists a pair of symmetric matrices (X, Y) in tandem with scalings $Q, S, R, \widetilde{Q}, \widetilde{S}$ and $\widetilde{R}$ such that the LMIs (11)-(14) and the nonlinear algebraic constraints (15) below are satisfied:*

$$\mathcal{N}_X^T \begin{pmatrix} A^T X + XA & XB_\Theta + C_\Theta^T S^T & XB_1 & C_\Theta^T R & C_1^T \\ B_\Theta^T X + SC_\Theta & Q + SD_{\Theta\Theta} + D_{\Theta\Theta}^T S^T & SD_{\Theta 1} & D_{\Theta\Theta}^T R & D_{1\Theta}^T \\ B_1^T X & D_{\Theta 1}^T S^T & -\gamma I & D_{\Theta 1}^T R & D_{11}^T \\ RC_\Theta & RD_{\Theta\Theta} & RD_{\Theta 1} & -R & 0 \\ C_1 & D_{1\Theta} & D_{11} & 0 & -\gamma I \end{pmatrix} \mathcal{N}_X < 0 \qquad (11)$$

$$\mathcal{N}_Y^T \begin{pmatrix} AY + YA^T & YC_\Theta^T + B_\Theta \widetilde{S} & YC_1^T & B_\Theta \widetilde{Q} & B_1 \\ C_\Theta Y + \widetilde{S}^T B_\Theta^T & D_{\Theta\Theta}\widetilde{S} + \widetilde{S}^T D_{\Theta\Theta} - \widetilde{R} & \widetilde{S}^T D_{1\Theta}^T & D_{\Theta\Theta}\widetilde{Q} & D_{\Theta 1} \\ C_1 Y & D_{1\Theta}\widetilde{S} & -\gamma I & D_{1\Theta}\widetilde{Q} & D_{11} \\ \widetilde{Q} B_\Theta^T & \widetilde{Q} D_{\Theta\Theta}^T & \widetilde{Q} D_{1\Theta}^T & \widetilde{Q} & 0 \\ B_1^T & D_{\Theta 1}^T & D_{11}^T & 0 & -\gamma I \end{pmatrix} \mathcal{N}_Y < 0 \qquad (12)$$

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} > 0 \,, \quad \begin{pmatrix} -Q & 0 \\ 0 & R \end{pmatrix} > 0 \,, \qquad (13)$$

$$\begin{pmatrix} \Theta_{v_i} \\ I \end{pmatrix}^T \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \begin{pmatrix} \Theta_{v_i} \\ I \end{pmatrix} > 0, \, i = 1, \cdots, N. \tag{14}$$

$$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix}^{-1} = \begin{pmatrix} \widetilde{Q} & \widetilde{S} \\ \widetilde{S}^T & \widetilde{R} \end{pmatrix}. \tag{15}$$

Notice here that due to the nonlinear constraints (15), the problem under consideration is non-convex and has even been shown to have non-polynomial (NP) complexity [11]. This is in sharp contrast with the traditional linear parameter-varying (LPV) control problem for which the LMI constraints (11)-(14) are the same but the nonlinear constraints (15) causing the trouble is not required. Theorem 2.1 is cited from [19], and the reader is referred to [16, 2, 3, 20] for related texts and further details.

3 AUGMENTED LAGRANGIAN METHOD

In this section, we present our approach to finding local solutions, in a sense to be defined later, to the robust synthesis problem in Theorem 2.1. We recast it as an optimization problem using a cost function which combines the L_2-gain performance index γ and a penalty term accounting for the nonlinear constraint (15), attributing a high cost to infeasible points. The LMI-constraints, being different in nature, are not included in the objective but kept explicitely. Our approach is known in nonlinear optimization with a finite number of equalities [13] as an *augmented Lagrangian method*, and we extend it here in a natural way to include LMI constraints. The entire procedure is then a sequential semi-definite programming (SSDP) scheme inspired by the sequential quadratic programming (SQP) method in classical optimization.

In order to simplify the expressions, we shall use the following notations. Define new variables $P, \widetilde{P}$ as :

$$P = \begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} \;\; , \;\; \widetilde{P} = \begin{pmatrix} \widetilde{Q} & \widetilde{S} \\ \widetilde{S}^T & \widetilde{R} \end{pmatrix}. \tag{16}$$

Let $\mathcal{X}$ be the convex set of LMI constraints (11)-(14), and x be the complete vector of decision variables $x = (\gamma, P, \tilde{P})$. The robust control problem is equivalently formulated as :

$$\min \left\{ \gamma \; : \; P\widetilde{P} - I = 0 \; , \; x \in \mathcal{X} \right\}, \tag{17}$$

i.e., we are looking for the best L_2-gain threshold γ for which a robust closed loop controller (7) for (4) exists.

The key idea in solving (17) is now to eliminate the non-convex constraints (15) by including them into a partially augmented Lagrangian function. This allows us to break the difficult non-convex synthesis problem into a series of easier LMI subproblems. The non-convex problem (17) is now approximated by a series of new optimization problems each of which involves minimizing the augmented Lagrangian function, $\Phi_c(x, \Lambda)$, defined as:

$$\Phi_c(x, \Lambda) = \gamma + \sum_{ij} \Lambda_{ij} (P\widetilde{P} - I)_{ij} + \frac{c}{2} \sum_{ij} (P\widetilde{P} - I)_{ij}^2$$

subject to the LMI-constraints (2). In matrix form, the new objective is:

$$\Phi_c(x,\Lambda) = \gamma + \mathrm{Tr}\Big(\Lambda(P\widetilde{P} - I)\Big) + \frac{c}{2}\mathrm{Tr}\Big((P\widetilde{P} - I)^T(P\widetilde{P} - I)\Big), \tag{18}$$

where c is a positive penalty parameter and Λ is a Lagrange multiplier. Each of the new optimization problems

$$\begin{array}{ll} \text{minimize} & \Phi_c(x,\Lambda) \\ \text{subject to} & x \in \mathcal{X} \end{array} \tag{19}$$

can then be solved by a sequence of SDPs. At the current position x, a new iterate x^+ is obtained by minimizing a second-order Taylor series approximation of $\Phi_c(\cdot,\Lambda)$ about the current x subject to the LMI-constraints. We must keep in mind, however, that the variables γ, P, $\widetilde{P}$ are linked to the Lyapunov variables X and Y appearing in the LMI constraints (11)-(14).

Let x^* be a local minimizer of problem (17), and suppose there exists a Lagrange multiplier matrix Λ^* associated with x^* such that the first and second order optimality conditions hold. Letting

$$\Phi(x,\Lambda) \;=\; \gamma + \mathrm{Tr}(\Lambda(P\tilde{P} - I))$$

the Lagrangian of (17), and observing that for every $c > 0$, $\Phi(x^*,\Lambda^*) = \Phi_c(x^*,\Lambda^*)$, $\nabla\Phi(x^*,\Lambda^*) = \Phi_c(x^*,\Lambda^*)$ and $\nabla^2\Phi(x^*,\Lambda^*) \leq \nabla^2\Phi_c(x^*,\Lambda^*)$ at the optimum (x^*,Λ^*), we obtain the following optimality conditions for the augmented Lagrangian:

$$\nabla\Phi_c^T(x^*,\Lambda^*)(x - x^*) \geq 0,\ \forall x \in \mathcal{X}$$

$$(x - x^*)^T\nabla^2\Phi_c(x^*,\Lambda^*)(x - x^*) > 0, \quad \forall x \neq x^*,\ x - x^* \text{ tangential},$$

valid for every $c > 0$. Here tangential refers to variables x satisfying $(x - x^*)^T\nabla\Phi(x^*,\Lambda^*) = (x - x^*)^T\nabla\Phi_c(x^*,\Lambda^*) = 0$. As a consequence of these conditions,

$$\Phi_c(x,\Lambda^*) \geq \Phi_c(x^*,\Lambda^*) + \frac{\eta}{2}\|x - x^*\|^2, \quad \forall x \in \mathcal{X},\ \|x - x^*\| < \varepsilon$$

for certain $\eta, \varepsilon < 0$ and every $c > 0$.

There are now two mechanisms by which the minimization of (19) can yield points close to x^* [13]. Clearly, when Λ is close to Λ^*. If this is not the case, it is still reasonable to infer that exists a local minimizer of $\Phi_c(.,\Lambda)$ close to x^* if c is chosen sufficiently large, say $c \geq \bar{c}$ for a certain threshold $\bar{c}$. In fact, by taking c large, we attribute a high cost to infeasible points, so the local minimizer of $\Phi_c(.,\Lambda)$ will be nearly feasible and we can expect $\Phi_c(x,\Lambda)$ to be close to γ for nearly feasible x. This suggests that in both cases, we can obtain good approximations to x^*.

To ensure that Λ tends to Λ^* during the iteration, we consider an intelligent update of Λ^j based on first-order Lagrange multiplier estimates [13, 8]:

$$\Lambda^{j+1} = \Lambda^j + c^j(P_{j+1}\widetilde{P}_{j+1} - I)\,.$$

This updating rule improves the convergence to a local minimizer x^* even when the penalty parameter c is not large [13] and thus, numerical ill conditioning is avoided. The reader is

referred to [8, 10] for more details about the convergence properties of this method which is often called the *first-order Lagrange multiplier method.*

When implementing a second-order algorithm, we need to compute the gradient and Hessian of the augmented Lagrangian (18). It is obtained about the point $x = (\gamma, P, \widetilde{P})$ as

$$\Phi_c(x+\delta,\Lambda) = \Phi_c(x,\Lambda) + \nabla\Phi_c(x,\Lambda)^T\delta + \frac{1}{2}\delta^T\nabla^2\Phi_c(x,\Lambda)\delta + O(\|\delta\|^2)\,,$$

where δ is the increment vector, $\nabla\Phi_c(x,\Lambda)$ is the gradient vector that can be computed by using the properties of the svec operator as:

$$\frac{\partial\Phi_c}{\partial P} = 2T^2\mathrm{svec}\Big[\mathrm{Herm}(\widetilde{P}\Lambda + c.P\widetilde{P}^2 - c.\widetilde{P})\Big] \tag{20}$$

$$\frac{\partial\Phi_c}{\partial\widetilde{P}} = 2T^2\mathrm{svec}\Big[\mathrm{Herm}(\Lambda P + c.\widetilde{P}P^2 - c.P)\Big],\quad \frac{\partial\Phi_c}{\partial\gamma} = 1\,. \tag{21}$$

The Hessian $\nabla^2\Phi_c$ of the augmented Lagrangian is computed in a similar manner:

$$\frac{\partial^2\Phi_c}{\partial P^2} = c.T\Big[(\widetilde{P})^2 \circledast I\Big]T,\quad \frac{\partial^2\Phi_c}{\partial\widetilde{P}^2} = c.T\Big[(P)^2 \circledast I\Big]T \tag{22}$$

$$\frac{\partial^2\Phi_c}{\partial P\partial\widetilde{P}} = T\Big[\Lambda \circledast I + c.(\widetilde{P}P - I) \circledast I + c.\widetilde{P} \circledast P\Big]T \tag{23}$$

while second-order derivatives involving γ are identically zero. These results are similar to those given in [6] (Proposition 4.1), and proofs are omitted for brevity. Finally, this first-order Lagrange multiplier method is described in the next section.

3.1 Algorithm description

Step 0. Initialization. Initialize the algorithm by determining a feasible point of the LMI constraints: For fixed large enough $\gamma = \gamma_0$, find an initial point that renders the LMIs (11)-(14) maximally negative by solving the SDP:

$$\min\Big\{t \;:\; \text{LMIs (11)-(14)} < t\,I\Big\}.$$

Then, determine X_0, Y_0, P_0 and $\widetilde{P}_0$ so that $P_0\widetilde{P}_0 - I$ is as close as possible to zero. This can be done using the techniques in [5, 6]. Then initialize the penalty parameter $c^0 > 0$ and the Lagrange multiplier Λ^0.

Step 1. Lagrangian minimization. For $j = 0, 1, \cdots$ minimize the augmented Lagrangian $\Phi_j(x) := \Phi_{c^j}(x, \Lambda^j)$ associated with Λ^j, c^j subject to the LMI constraints $x \in \mathcal{X}$. The solution so obtained is $x^{j+1} = (\gamma_{j+1}, P_{j+1}, \tilde{P}_{j+1})$.

Step 2. Update penalty and multiplier.

$$\Lambda^{j+1} = \Lambda^j + c^j(P_{j+1}\widetilde{P}_{j+1} - I).$$

$$c^{j+1} = \begin{cases} \rho c^j & \text{if } \|P_{j+1}\widetilde{P}_{j+1} - I\|_F > \mu\|P_j\widetilde{P}_j - I\|_F \\ c^j & \text{if } \|P_{j+1}\widetilde{P}_{j+1} - I\|_F \le \mu\|P_j\widetilde{P}_j - I\|_F \end{cases} \tag{24}$$

for given ρ and μ.

Step 3. Terminating phase. Due to non-linearity the algebraic constraint (15) is never exactly satisfied at x^{j+1}. It is, however, possible to terminate the program without strict satisfaction of the nonlinear constraints by a simple perturbation technique [5], which is applicable as long as the LMIs (11)-(14) are strictly satisfied. One can then replace $\widetilde{P}_{j+1}$ with P_{j+1}^{-1} and check whether the LMI constraints (11)-(14) hold, possibly with new X and Y. In this case a controller is readily obtained. Dually, we can replace P_{j+1} with $\widetilde{P}_{j+1}^{-1}$ and check the LMI constraints (11)-(14), with the scaling constraint in (14) suitably replaced with its dual form

$$\begin{pmatrix} I \\ -\Theta_{v_i}^T \end{pmatrix}^T \widetilde{P}_{j+1} \begin{pmatrix} I \\ -\Theta_{v_i}^T \end{pmatrix} < 0 \quad , \quad \forall i = 1, \cdots, N.$$

If the perturbation technique fails, set $j = j + 1$ and return to **Step 1**.

3.2 Choice of parameters

An important practical question is how to select the initial multiplier Λ^0 and the penalty parameter sequence c^j. Any prior knowledge should be exploited to select Λ^0 as close as possible to Λ^*, but this is generally difficult. Concerning the penalty parameter sequence c^j some important remarks are in order:

- the initial value of c^0 should not be too large. This increases the number of steps, and the method may fail to converge when the algebraic nonlinear constraint (15) measured in the Frobenius norm is still too large.

- c^j should not be increased too fast to a point where the sub-problem (19) becomes ill-conditioned.

- c^j should not be increased too slowly, at least in the early steps, because otherwise the first-order update of the Lagrange multiplier has a poor convergence rate.

A good practical scheme is to choose a moderate value c^0, and then during an initial phase increase c^j by a factor $\rho > 1$ only if the constraint violation measured by $\|P\widetilde{P} - I\|_F$ is not decreased by a factor $0 < \mu < 1$ over the previous minimization as in (24). Typical values are $\rho = 4$ and $\mu = 0.2$. The heuristic presented in the Appendix justifies the update in step 2 and suggests that during the terminal phase, the terms

$$\Lambda_{k\ell}^{j+1}(P_{j+1}\tilde{P}_{j+1} - I)_{k\ell} + c^{j+1}(P_{j+1}\tilde{P}_{j+1} - I)_{k\ell} \approx \Lambda_{k\ell}^{j}(P_j\tilde{P}_j - I)_{k\ell} + c^{j}(P_j\tilde{P}_j - I)_{k\ell}$$

should be held approximately constant.

Yet another possibility is to use an individual penalty parameter $c_{k\ell}$ for every element in the constraint matrix and to increase by a certain factor the penalty parameters of the constraints that are violated most.

3.3 Modified Newton method

We have not specified in which way the minimization of $\Phi_j(x) = \Phi_{c^j}(x, \Lambda^j)$ in step 1 of the algorithm should be achieved. As a first option, we propose to use a Newton type method which minimizes the second order Taylor polynomial

$$\psi(\delta) = \Phi_j(x) + \nabla\Phi_j(x)^T\delta + \frac{1}{2}\delta^T\nabla^2\Phi_j(x)\delta \tag{25}$$

of $\Phi_j(x)$ about the current iterate x and subject to the constraint set $x + \delta \in \mathcal{X}$ in order to obtain the next iterate x^+. When combined with a line search, this provides the new iterate $x^+ = x + t\delta$ with an appropriate $t > 0$.

A difficulty with Newton's method occurs when the Hessian $\nabla^2\Phi_j(x)$ is not positive definite. In this case, modifying the inertia of $\nabla^2\Phi_j(x)$ may be advised, for instance by adding a diagonal correction matrix D rendering the matrix $\nabla^2\Phi_j(x) + D$ positive definite and reasonably well conditioned. Different techniques have been proposed in the literature [9, ?]. In most schemes, a modified Cholesky algorithm is used and pivot entries are sequentially introduced to meet the positive definiteness condition. The modified Cholesky *factorization* LL^T of $H := \nabla^2\Phi_j(x) + D$, yields the diagonal D matrix described as:

$$D_{\ell\ell} = \begin{cases} 0 & \text{if } \mu_1 < H_{\ell\ell} - \sum_{m=1}^{\ell-1} L_{\ell m}^2 \\ \mu_2 - \left(H_{\ell\ell} - \sum_{m=1}^{\ell-1} L_{\ell m}^2 \right) & \text{otherwise ,} \end{cases}$$

with $\mu_1 = 0$ and μ_2 chosen in general larger than μ_1.

Actually, we found it even more efficient to adopt an argument known from the Gauss-Newton method which neglects the term $\widetilde{P}_j P_j - I$ in the Hessian matrix (22)-(23), while performing a modified Cholesky factorization of the resulting term. This is motivated by the fact that by dropping the constraint term, the new matrix is positive semi-definite, while coming closer to the correct Hessian matrix as soon as the neglected term $P_j\widetilde{P}_j - I$ approaches zero, that is when the nonlinear constraint (15) nearly holds.

3.4 Trust-region techniques

While modification techniques as above are employed to address the possible indefiniteness of the Hessian $\nabla^2\Phi_j(x)$, one may replace the Newton step (plus line search) by a trust-region strategy. Trust-region methods form a popular class of iterative optimization algorithms, in which the cost function $\Phi_j(x)$ is approximated by a local model which is minimized in a *neighborhood* of the current iterate x. This model is generally the second order Taylor polynomial (25) of Φ_j about the current x. The method will only "trust" this model within a limited neighborhood of the point x, defined by the constraint $\|\delta\| \leq \Delta$, where Δ is the trust-region radius. We will therefore at each step minimize $\psi(\delta)$ subject to the constraints $\|\delta\| \leq \Delta$ and $x + \delta \in \mathcal{X}$ in order to find the new iterate $x^+ = x + \delta$. The appropriate Δ is found by matching the model and the true cost function: If the value of the cost function $\Phi_j(x + \delta)$ computed at the trial point $x + \delta$ produces a

decrease in cost comparable to the decrease predicted by the model, the radius Δ and hence the trial point $x+\delta$ is accepted as the next iterate x^+. Otherwise, the trial step cannot be trusted and Δ has to be decreased.

The details of the trust-region algorithm are described in the following.

Step 1. Select the initial trust region bound $\Delta_0 > 0$ and specify the constants

$$0 < \eta_1 < \eta_2 < 1$$

Typical values are $\eta_1 = \frac{1}{4}, \eta_2 = \frac{3}{4}$.

Step 2. Solve the optimization problem

$$\min\ \psi(\delta) := \Phi_j(x) + \nabla\Phi_j^T(x)\delta + \frac{1}{2}\delta^T\nabla^2\Phi_j(x)\delta \tag{26}$$

$$\text{over}\ \ \mathrm{x} \in \mathcal{X} \text{ and } \|\delta\| \leq \Delta$$

compute

$$\rho := \frac{\Phi_j(x) - \Phi_j(x+\delta)}{\Phi_j(x) - \psi(\delta)}.$$

Step 3. Set the new estimate of x :

$$x^+ = \begin{cases} x+\delta & \text{if } \rho \leq \eta_1 \\ x & \text{otherwise.} \end{cases}$$

Step 4. Update The Trust-Region radius Δ as :

$$\Delta^+ = \begin{cases} \alpha_1\|\delta\| & \text{if } \rho \leq \eta_1 \\ \Delta & \text{if } \eta_1 < \rho < \eta_2 \\ \max\{\alpha_2\|\delta\|, \Delta\} & \text{if } \rho \geq \eta_2 \end{cases}$$

for some $0 < \alpha_1 < 1 < \alpha_2$.

The value of ρ indicates how well the model predicts the change in the cost value. If ρ is small ($\rho \leq \eta_1$), then the actual change in the cost value $\Phi_j(x)$ is much smaller than that predicted by $\psi_k(\delta)$. This indicates that the model cannot be trusted for a bound as large as Δ. In this case the step δ will be rejected and Δ will be reduced. If ρ is large ($\rho \geq \eta_2$), then the model is adequately predicting the change in the cost value, this suggests that the model can be trusted over an even wider region. In this case the bound Δ will be increased.

The choice of the initial trial value for Δ is crucial here; if it is chosen too large, a large number of steps may be required before a cost improvement occurs. If it is chosen too small the convergence rate may be poor. In particular, to ensure better convergence properties of the algorithm, we should use a good strategy to determine a maximal initial trust-region radius Δ_0 that guarantees a sufficient agreement between the model and the cost function in the direction $g_0 = -\nabla\Phi_j(x_0)$ or the Newton direction $g_0 = -\nabla^2\Phi_j(x_0)^{-1}\nabla_x\Phi_j(x_0)$, using an iterative search

along this direction [18]. At each iteration i of the search, given a radius estimate $\Delta_0^{(i)}$, the model and the cost values are computed at the point $x_0 - \Delta_0^{(i)}.\hat{g}_0$, as:

$$\psi_0^{(i)} := \psi(x_0 - \Delta_0^{(i)}.\hat{g}_0)\ ,\ \Phi_{j_0}^{(i)} := \Phi_j(x_0 - \Delta_0^{(i)}.\hat{g}_0)$$

where $\hat{g}_0 = g_0/\|g_0\|$. The ratio

$$\rho_0^{(i)} = \frac{\Phi_{j_0} - \Phi_{j_0}^{(i)}}{\Phi_{j_0} - \psi_0^{(i)}} \quad ;\ \Phi_{j_0} = \Phi_j(x_0)$$

is also computed, and the algorithm then stores the maximal value among the estimates $\Delta_0^{(\ell)}$ $(\ell = 0, \cdots, i)$, whose associated $\rho_0^{(\ell)}$ is "close enough to one" . The update of the current estimate $\Delta_0^{(i)}$ is given as $\Delta_0^{(i+1)} = \beta_0^{(i)}\Delta_0^{(i)}$ where

$$\beta_0^{(i)} = \begin{cases} \alpha_1 & \text{if } |\rho_0^{(i)} - 1| > \mu_1 \\ \alpha_2 & \text{if } |\rho_0^{(i)} - 1| \leq \mu_2 \\ 1 & \text{otherwise,} \end{cases}$$

for some $0 \leq \mu_2 < \mu_1$ and $0 < \alpha_1 < 1 < \alpha_2$.

Notice that the trust region approach is particularly interesting if the Hessian of the augmented Lagrangian $\nabla^2\Phi_j(x)$ is *not* positive definite, but curvature information should be used to obtain the new iterate. However, an indefinite Hessian does not lead to a convex LMI-subproblem. In order to guarantee convexity, we are forced to use the same modification techniques as presented in Section 3.4, that is, adding a diagonal D to $\nabla^2\Phi_j(x)$, or use the Gauss-Newton approximation. While this may often destroy the major advantage of the trust region approach, the full expoitation of the indefinite Hessian, renders step 1 of the algorithm more expensive in CPU. It is therefore not too surprising that the trust region approach is inferior to Newton's method under these circumstances, as corroborated by our numerical tests.

4 NUMERICAL EXPERIMENTS

This section provides two applications of the methods just discussed. As compared to previously developed techniques like Frank & Wolfe [4], our approach is in many respects preferable. Firstly, the Frank & Wolfe algorithm is not guaranteed to find a local optimal solution and is often subject to zigzagging in the final steps. In contrast, our augmented Lagrangian method shows good convergence properties and we usually observe a linear convergence rate from practically any feasible starting point. Furthermore, the approach here is more general than the Frank & Wolfe approach since a best γ level is computed, whereas the former technique will require a less efficient dichotomy scheme to minimize γ. Based on two examples, we shall also compare the efficiency of our approach to the popular $D - K$ iteration method.

4.1 Robust control of Flexible Actuator

Consider the unbalanced oscillator described in Figure 1. The plant is built with a cart (weight M) fixed to a vertical plane by a linear spring k and constrained to move only along the z axis.

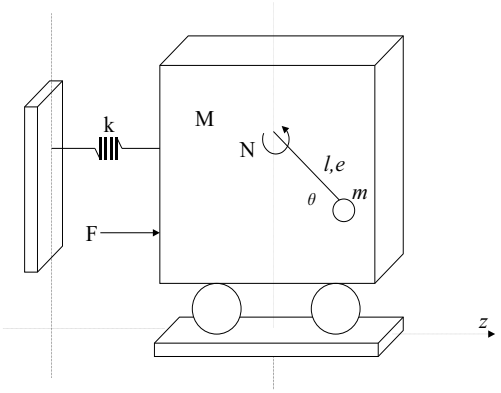

Figure 1: Flexible Actuator

An embedded pendulum (with mass m and moment of inertia I) is attached to the center of mass of the cart and can be rotated in the horizontal plane. The cart is submitted to an external disturbance F and a control torque N is applied to the pendulum.

The nonlinear equations of motion are:

$$(M+m)\ddot{Z} + me\ddot{\theta}\cos\theta = me\dot{\theta}^2\sin\theta - kZ + F$$

$$me\ddot{Z}\cos\theta + (I+me^2)\ddot{\theta} = N$$

where θ and $\dot{\theta}$ denote the angular position and velocity of the pendulum, and $Z, \dot{Z}$ denote the position and velocity of the cart, respectively. We normalize these equations as in [12]:

$$\ddot{\zeta} + \varepsilon\ddot{\theta}\cos\theta = \varepsilon\dot{\theta}^2\sin\theta - \zeta + w, \qquad \varepsilon\ddot{\zeta}\cos\theta + \ddot{\theta} = u$$

where $[\zeta\ \dot{\zeta}\ \theta\ \dot{\theta}]^T$ is the new state vector. We express the nonlinear terms of the equations through the parameters δ_1, δ_2 defined by $\delta_1 = \cos\theta$ and $\delta_2 = \dot{\theta}\sin\theta$. The LFT model of the plant is then derived and numerical data are given in Appendix B. Table 2 displays the behavior of Algorithm 3.1 in this example. We can see that for both line search and trust-region techniques, the Lagrangian method achieves good values of γ already after a few iterations. The nonlinear constraints decrease with an approximately linear rate. In practice, one may stop the algorithm whenever γ is not reduced over a certain number of iteration and the nonlinear constraint is sufficiently small, say smaller that 10^{-3} or 10^{-4}. The final steps in the table are only for illustration of the asymptotic behavior of the method. Note that the number of decision variables is 94 in this example.

4.2 Autopilot robust control of missile

Consider the missile-airframe control problem illustrated in Figure 2. The control problem requires that the autopilot generates the required tail deflection (δ) to produce an angle of attack α corresponding to a maneuver called by the guidance law. Sensor measurements for feedback include missile rotational rates q (rate gyros) and α. For the problem considered here, it is desired to track step input commands α_c with a steady state accuracy of 1% and to achieve

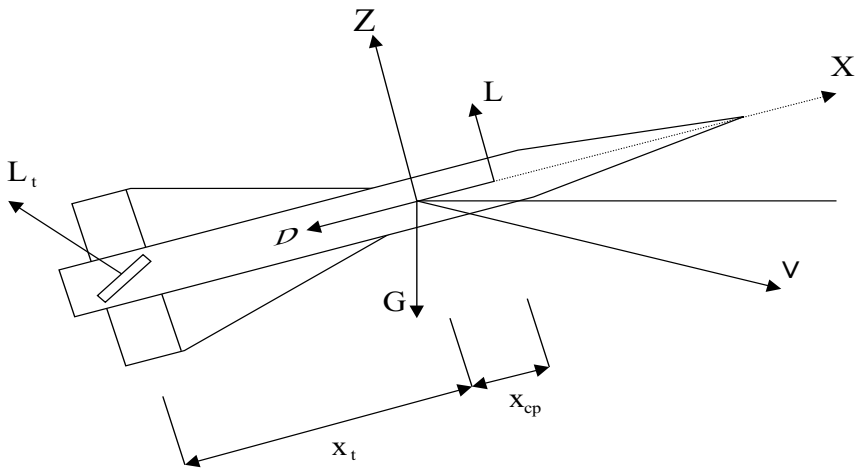

Figure 2: Longitudinal model for air to air missile

a rise time less than 0.2 second, and limited overshoot to 2% over a wide range of angles of attack ± 20 deg., and variations in Mach number 2.5 to 3.5. The nonlinear state equations of the missile are given as [17] :

$$\dot{\alpha} = K_\alpha M C_n(\alpha, \delta, M)\cos\alpha + q$$

$$\dot{q} = K_q M^2 C_m(\alpha, \delta, M)$$

where $[\alpha;\ q]$ is the state vector. The aerodynamics coefficients C_n, C_m are defined by a polynomial expression in α and δ and given as :

$$C_n(\alpha, \delta, M) = a_n\alpha^3 + b_n|\alpha|\alpha + c_n(2 - \frac{M}{3})\alpha + d_n\delta$$

$$C_m(\alpha, \delta, M) = a_m\alpha^3 + b_m|\alpha|\alpha + c_m(-7 + 8\frac{M}{3})\alpha + d_m\delta$$

with the values in Table 1 of Appendix C.

The missile tail fin actuator is modeled as a first-order system with time constant of $1/150$ seconds. The numerical data of the LFT plant are given in appendix C. The proposed optimization techniques discussed in this paper are then readily applicable and results are shown in Table 3. We note that the method behaves in much the same way as for the flexible actuator: good values of γ are achieved after a few iterations with a similiar rate of decrease of the nonlinear constraints.

For $\gamma = 0.952$ and using the trust-region technique, the optimal controller is given as:

$$\left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array}\right] = \left[\begin{array}{cccc|cc} -80.347 & 73.162 & 3627.94 & -668.44 & -32.266 & -2916.3 \\ 8.46719 & -8.5839 & -373.33 & 57.2022 & 2.60852 & 299.040 \\ 192.075 & -181.66 & -8694.6 & 366.130 & 11.5816 & 6979.91 \\ 2.1114 & -0.0529 & -689.32 & -7003.9 & 8.82356 & 552.939 \\ \hline 0.4619 & 0.4110 & 24.112 & -4.4648 & -0.21534 & -19.38 \end{array}\right].$$

Computational experience on a set of different examples indicates that the number of iterations in terms of SDPs is almost independent of the problem dimension (132 decision variables) while the CPU time is of course strongly dependent on the efficiency of the SDP solver. As it turns out, in its actual state the bottleneck of our approach *is* the SDP-solver. The public domain software for SDP we tested could be reliably used to problem sizes up to $500 - 1000$ decision variables. For larger sizes, the method may fail due to failure of the SDP-solver to find a feasible starting point or to solve the LMI-subproblem. Solvers exploiting at best the structure of the problem under consideration may then be required.

4.2.1 Comparison with D-K iteration

In this section, we provide a brief comparison with $D-K$-iteration method. The general scheme is as follows.

Step 1. Find an initial controller that stabilizes the closed-loop system.

Step 2. Analysis phase : for a fixed controller, find the optimal γ subject to the LMI constraints (8)-(9).

Step 3. Compute the scaling $\widetilde{Q}, \widetilde{S}$ and $\widetilde{R}$ so that the nonlinear constraints (15) holds.

Step 4. Synthesis phase : for fixed scaling, minimize γ subject to the LMI constraints (11)-(13).

Step 5. Compute the new controller and return to **Step 2**, until convergence.

Table 4 shows the best behavior that we have obtained so far. In many instances the algorithm is often cycling or the cost increases before reaching a smaller value. We observe that this coordinate descent technique fails to achieve an adequate value of γ, Table 4, as compared to the Lagrangian method in Table 3. This is due to the fact that the method is not garanteed to provide a local minimizer. Saddle points are typical difficulties in coordinate descent schemes. Also, the convergence is fairly slow and exhibits a typical gradient behavior.

5 CONCLUDING REMARKS

In this paper, we have developed an augmented Lagrangian technique for finding local solutions to robust control problems. The proposed technique is an extension of the classical augmented Lagrangian method. It comprises LMI-constraints which are handled explicitly in the course of the algorithm. The method is comfortably implemented with available SDP codes, and may be highlighted as a *successive semi-definite programming method*. We found the approach highly reliable (as we demonstrated on a set of test examples), exhibiting good convergence properties, and applicable to a rich list of problems in robust control theory. In conclusion, SSDP provides considerable advantages in terms of efficiency and reduced conservatism as compared to customary D-K-iteration schemes.

A Multiplier update rule

Consider the optimization problem

$$(P) \qquad \begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) = 0 \end{array}$$

which we solve via an augmented Lagrangian approach:

$$(AL) \qquad \text{minimize } \Phi_c(x,\lambda) = f(x) + \lambda g(x) + \frac{c}{2} g(x)^2.$$

Assuming that we are in fact allowed to stop the minimization at the threshold $g(x) \approx \varepsilon$, we should compare the minimization of $\Phi_c(\cdot,\lambda)$ to the relaxed problem

$$(P_\varepsilon) \qquad \begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h(x) := \frac{1}{2}(g(x)^2 - \varepsilon^2) \le 0 \end{array}$$

Assuming that the Kuhn-Tucker point x_ε for (P_ε) has multiplier μ and the constraint is active, matching the Kuhn-Tucker conditions from both problems (AL) and (P_ε) suggest that for iterates x having $g(x) \approx \varepsilon$, we expect

$$\nabla f(x_\varepsilon) + \mu g(x_\varepsilon) \nabla g(x_\varepsilon) \approx \nabla f(x) + (\lambda + c g(x)) \nabla g(x),$$

which suggests that $\lambda + cg(x)$ should be closed to the fixed value $\mu g(x_\varepsilon)$ and therefore change only slightly towards the end of the iteration, that is, when the method approaches iterates x having $g(x) \approx \varepsilon$. This heuristic should at least give an indication on how to proceed in the more complicated case involving LMI-constraints, and it seems that our update rule (step 2 of the algorithm) confirms this.

B Actuator state-space data

The numerical data for the first example subject to LFT plant are:

$$\begin{pmatrix} \dot\zeta \\ \ddot\zeta \\ \dot\theta \\ \ddot\theta \\ \hline z_\zeta \\ z_\theta \\ z_u \\ \hline \zeta \\ \theta \end{pmatrix} = \left\{ \left(\begin{array}{cccc|c|c} 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -0.2 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & -0.2 & 1 \\ \hline -0.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right) + \right.$$

$$\left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ \frac{\varepsilon}{2} & -\frac{\varepsilon}{2} & -\frac{\varepsilon}{2} & 5\frac{\varepsilon}{2} \\ 0 & 0 & 0 & 0 \\ \frac{\varepsilon}{2} & \frac{\varepsilon}{2} & 0 & -\frac{\varepsilon}{2} \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \Theta(t) \left(I - \begin{bmatrix} -\frac{\varepsilon}{2} & 0 & \frac{\varepsilon}{2} & -2\varepsilon \\ 0 & \varepsilon & \varepsilon & -3\varepsilon \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Theta(t) \right)^{-1}$$

$$\times \left(\begin{array}{cccc|c|c} 0.84 & 0 & 0 & 0 & -0.84 & 0.16 \\ 1.23 & 0 & 0 & 0\ -1.23 & 0.23 & \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right) \Bigg\} \left(\begin{array}{c} \zeta \\ \dot{\zeta} \\ \theta \\ \dot{\theta} \\ \hline w \\ \hline u \end{array}\right)$$

where ε, a coupling parameter, is chosen in this problem as 0.02. The vector of regulated variables z consists of three components, z_ζ, z_θ correspond to damping specifications on ζ, θ , and z_u to penalize the control activity. The exogenous input w is the external force F.

C Missile state-space data

The numerical data for LFT air-to-air missile are:

a_n	$0.103 \times 10^{-}3$	a_m	0.215×10^3	K_α	1.185
b_n	-0. 945	b_m	-0.0195	K_q	70.526
c_n	-0.1696	c_m	0.051	K_z	0.6659
d_n	-0.034	d_m	-0.206	M_0	3

Table 1: Numerical values

$$\left(\begin{array}{c} \dot{\alpha} \\ \dot{q} \\ \dot{\delta} \\ \hline \dot{x_w} \\ \hline z_e \\ z_\delta \\ \hline \alpha_c - \alpha \\ n - q \end{array}\right) = \Bigg\{ \left(\begin{array}{cccc|ccc} -0.8767 & 1 & -0.1209 & 0 & 0 & 0 & 0 \\ 8.9117 & 0 & -130.755 & 0 & 0 & 0 & 0 \\ 0 & 0 & -150 & 0 & 0 & 0 & 150 \\ -1 & 0 & 0 & -0.05 & 0 & 1 & 0 \\ \hline -0.25 & 0 & 0 & 3.4875 & 0 & 0.25 & 0 \\ 0 & 0 & -3 & 0 & 0 & 0 & 3 \\ \hline -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0.01 & 0 & 0 \end{array}\right) +$$

$$\left(\begin{array}{ccccc} 0.2737 & 0.201 & 1.185 & 0 & 0 \\ 23.46 & 86.324 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \Theta(t) \left(I - \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.038 & 0.0283 & 0 & 0 & 0 \\ 1.303 & 4.796 & 0 & 0 & 0 \\ 3.91 & 14.387 & 0 & .5 & 0 \end{array}\right] \Theta(t) \right)^{-1}$$

$$\left(\begin{array}{cccc|cc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.1233 & 0 & -0.0170 & 0 & 0 & 0 & 0 \\ 0.4951 & 0 & -7.2642 & 0 & 0 & 0 & 0 \\ 1.4853 & 0 & -21.7925 & 0 & 0 & 0 & 0 \end{array}\right) \Bigg\} \left(\begin{array}{c} \alpha \\ q \\ \delta \\ \hline x_w \\ \hline n \\ \hline \alpha_c \\ \hline \delta_e \end{array}\right)$$

where the time-varying matrix-valued parameter $\Theta(t)$ is defined as :

$$\Theta(t) = \begin{pmatrix} \Delta_\alpha & 0 \\ 0 & \Delta_M I_4 \end{pmatrix}.$$

The vector of regulated variables z consists of two components. The first, z_e, corresponds to a frequency-weighted sensitivity design goal, for tracking error accuracy, while z_δ serves to limit

the tail-fin actuator rate $\dot{\delta}$ and indirectly to bound the controller bandwidth to avoid problems with unmodeled flexible modes. The vector of exogenous inputs w includes the command α_c and the pitch rate sensor noise n.

Acknowledgments

We thank Annick Sartenaer for her useful suggestions and comments on trust-region techniques, and for providing an advanced version of a forthcoming paper [18].

References

[1] F. Alizadeh, J.-P. Haeberly, and M. L. Overton, *Primal-dual interior-point methods for semi-definite programming: convergence rates, stability, and numerical results*, In revision for J. of Optimization, (1997).

[2] P. Apkarian and P. Gahinet, *A Convex Characterization of Gain-Scheduled H_∞ Controllers*, IEEE Trans. Aut. Control, 40 (1995), pp. 853–864. See also pp. 1681.

[3] P. Apkarian, P. Gahinet, and G. Becker, *Self-Scheduled H_∞ Control of Linear Parameter-Varying Systems: A Design Example*, Automatica, 31 (1995), pp. 1251–1261.

[4] P. Apkarian and H. D. Tuan, *Concave Programming in Control Theory*, (1998). to appear in J. of Global Optimization.

[5] ——, *Robust Control via Concave Minimization - Local and Global Algorithms*, (1998). to appear in IEEE Trans. on Automatic Control.

[6] ——, *A Sequential SDP/Gauss-Newton Algorithm for Rank-Constrained LMI Problems*, in Proc. IEEE Conf. on Decision and Control, 1999.

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

[8] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, Academic Press, London, 1982.

[9] ——, *Nonlinear Programming*, Athena Scientific, USA, Belmont, Mass., 1995.

[10] A. R. Conn, N. Gould, A. Sartenaer, and P. L. Toint, *Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and nonlinear constraints*, SIAM J. on Optimization, 6 (1996), pp. 674–703.

[11] J. David, *Algorithms for Analysis and Design of Robust Controllers*, Ph.D., Dept. of Elect. Eng., K. U. Leuven, Belgium, 1994.

[12] S. Dussy and L. E. Ghaoui, *Measurement-scheduled Control for RATC problem*, Int. J. Robust and Nonlinear Control, (1997).

[13] R. FLETCHER, *Practical Methods of Optimization*, John Wiley & Sons, 1987.

[14] P. GAHINET AND P. APKARIAN, *A Linear Matrix Inequality Approach to H_∞ Control*, Int. J. Robust and Nonlinear Control, 4 (1994), pp. 421–448.

[15] P. GAHINET, A. NEMIROVSKI, A. J. LAUB, AND M. CHILALI, *LMI Control Toolbox* , The MathWorks Inc., 1995.

[16] A. PACKARD, *Gain Scheduling via Linear Fractional Transformations*, Syst. Control Letters, 22 (1994), pp. 79–92.

[17] R. T. REICHERT, *Robust Autopilot Design Using μ-Synthesis*, in Proc. American Control Conf., may 1990, pp. 2368–2373.

[18] A. SARTENAER, *Automatic determination of an initial trust region in nonlinear programming*, J. sisci, 18 (1997), pp. 1788–1803.

[19] C. W. SCHERER, *A Full Block $\mathcal{S}$-Procedure with Applications*, in Proc. IEEE Conf. on Decision and Control, San Diego,USA, 1997, pp. 2602–2607.

[20] G. SCORLETTI AND L. E. GHAOUI, *Improved Linear Matrix Inequality Conditions for Gain-Scheduling*, in Proc. IEEE Conf. on Decision and Control, New Orleans, LA, Dec. 1995, pp. 3626–3631.

[21] K. ZHOU, J. C. DOYLE, AND K. GLOVER, *Robust and Optimal Control*, Printice Hall, 1996.

	Trust-Region			*Constrained Newton*		
step	γ	$\|\|P\widetilde{P}-I\|\|_F$	c	γ	$\|\|P\widetilde{P}-I\|\|_F$	c
0	7	8.018e-005	0.5	7	8.018e-005	0.5
1	2.028	4.328 e-003		1.824	2.477 e-002	
2	1.887	1.480 e-002		1.734	2.911 e-002	
3	1.806	2.303 e-002		1.702	2.771 e-002	
4	1.759	2.667 e-002		1.685	2.555 e-002	
5	1.730	2.742 e-002		1.667	2.258 e-002	
6	1.706	2.685 e-002		1.702	3.351 e-003	2
7	1.682	2.508 e-002		1.712	1.863 e-003	
8	1.671	2.366 e-002		1.720	2.250 e-004	8
9	1.697	1.852 e-003	2	1.723	1.219 e-004	
10	1.703	2.237 e-004	8	1.725	1.424 e-005	32
11	1.710	1.079 e-004	32	1.725	1.287 e-006	128
12	1.713	1.433 e-005	128	1.725	5.253 e-007	
13	1.715	1.280 e-006	512	1.725	6.205 e-008	512
14	1.715	5.221 e-007	1024	1.725	5.290 e-009	
15	1.715	6.171 e-008				

Table 2: Behavior of Algorithm 3.1 for Flexible Actuator
Computations on PC with cpu Pentium II 333 MHz and using the LMI Control Toolbox [15].

	Trust-Region			*Constrained Newton*		
step	γ	$\|\|P\widetilde{P}-I\|\|_F$	c	γ	$\|\|P\widetilde{P}-I\|\|_F$	c
0	5	8.037 e-004	0.25	5	8.037 e-004	0.25
1	1.550	2.205 e-000		3.602	5.417 e-000	
2	1.246	3.578 e-000		3.020	7.371 e-000	
3	1.246	3.578 e-000		2.150	9.215 e-000	
4	0.995	2.751 e-000		1.025	6.284 e-000	
5	0.979	2.897 e-000		0.982	4.572 e-000	
6	0.950	2.218 e-000		0.969	3.912 e-000	
7	0.935	1.712 e-000		0.952	2.617 e-001	
8	0.932	3.046 e-001		0.941	9.675 e-002	
9	0.937	1.123 e-001	1	0.940	7.270 e-002	
10	0.942	2.472 e-002	4	0.945	3.043 e-003	1
11	0.947	3.526 e-003	16	0.950	7.125 e-004	4
12	0.950	5.247 e-004	64	0.953	1.925 e-004	16
13	0.951	3.756 e-005	256	0.955	2.357 e-005	64
14	0.952	7.652 e-006	1024	0.955	3.253 e-006	256
15	0.952	1.257 e-006				

Table 3: Behavior of Algorithm 3.1 for Missile autopilot
Computations on PC with cpu Pentium II 333 MHz and using the LMI Control Toolbox

	Classical D-K iteration Approach				
step	phase	γ	step	phase	γ
0	-	5	5	A	2.346
1	A	4.871	6	S	2.320
2	S	3.237	7	A	2.287
3	A	2.401	8	S	2.280
4	S	2.373	9	A	2.252
			10	S	fail

Table 4: Behavior of the classical D-K iteration approach
A: analysis - S: synthesis