Reduced-order output feedback control design with *specSDP*, a code for linear / nonlinear SDP problems

J.B. Thevenet[*] and **P. Apkarian**[†] and **D. Noll**[‡]

[*] ONERA-CERT, Control System department, 2 Av. Edouard Belin, 31055 Toulouse, France,
MIP, Université Paul Sabatier, 118, route de Narbonne, 31062 Toulouse, France,
e-mail: thevenet@cert.fr

[†] ONERA-CERT and MIP,
e-mail: apkarian@cert.fr

[‡] MIP,
e-mail: noll@mip.ups-tlse.fr

Abstract

Optimization problems with bilinear matrix inequalities (BMI) constraints arise frequently in automatic control and are difficult to solve due to the inherent non-convexity. The purpose of this paper is to give an overview of the spectral SDP (semidefinite programming) method already presented in [Thevenet et al., 2004], along with practical information on how to basically use *specSDP*, a software designed to solve LMI / BMI-constrained optimization problems, with a linear or quadratic objective function. *specSDP* is a Fortran code, interfaced with Matlab. It is tested here against several examples of reduced order control problems from the benchmark collection *COMPlib* [Leibfritz and Lipinsky, 2003].

Keywords: Bilinear matrix inequality, spectral penalty function, reduced order control synthesis.

1 INTRODUCTION

The problem of minimizing an objective function subject to bilinear matrix inequality (BMI) constraints :

$$\begin{array}{ll} \text{minimize} & c^T x \ \left(+\frac{1}{2}\, x^T Q x\right),\ x \in \mathbb{R}^n \\ \text{subject to} & \mathcal{B}(x) \preceq 0, \end{array} \tag{1}$$

where $\preceq 0$ means negative semi-definite and

$$\mathcal{B}(x) \ = \ A_0 + \sum_{i=1}^{n} x_i A_i + \sum_{1 \le i < j \le n} x_i x_j C_{ij} \tag{2}$$

is a bilinear matrix-valued operator with data $A_i, C_{ij} \in \mathbb{S}^m$, is a general cast for a wide range of problems such as control synthesis, structure problems, filtering .... These problems are central in many industrial applications.

The importance of BMIs was recognized over the past decade and different solution strategies have been proposed. Earliest ideas use interior point methods (imported from semidefinite programming), concave programming or coordinate descent methods employing successions of SDP subproblems. The success of these methods is up to now very limited, and even moderately sized programs with no more than 100 variables usually may cause failure. For semidefinite programming, [Zibulevsky, 1996], and [Ben-Tal and Zibulevsky, 1997] propose a modified augmented Lagrangian method, which in [Kocvara and Stingl, 2003] has been expanded to include BMI-problems. The paper presents *specSDP* (spectral Non-Linear SDP), a code designed to solve any problem of type (1), with the option to treat pure LMI constraints explicitely. It also handles cases where the objective function includes a quadratic term, at least for LMI constraints. The code detects these cases and, as a consequence, selects the best way of solving them, which results in improved computational times. Using *specSDP* is relatively easy, and efficient, as it has been tested against many difficult numerical examples. See for instance [Apkarian et al., 2004] and [Thevenet et al., 2004], the latter containing in addition a full theoretical description of the method. The user is given the opportunity of choosing a number of miscellaneous options, as the default ones might be suboptimal on a specific problem. The software will be available soon for academics on request to the authors. The structure of the paper is as follows. In section 2, we recall briefly some relevant features of our spectral SDP method. Section 3 gives information on how to use the software basically. Finally applications to reduced order output feedback control design are examined in section 4.

NOTATION

Our notation is standard. We let $\mathbb{S}^m$ denote the set of $m \times m$ symmetric matrices, M^T the transpose of the matrix M and $\mathrm{Tr}\, M$ its trace. We equip $\mathbb{S}^m$ with the euclidean scalar prod-

uct $\langle X, Y\rangle = X \bullet Y = \mathrm{Tr}\,(XY)$. For symmetric matrices , $M \succ N$ means that $M - N$ is positive definite and $M \succeq N$ means that $M - N$ is positive semi-definite. We define the operator svec , which maps the set of symmetric matrices $\mathbb{S}^m$ into $\mathbb{R}^l$ where $l = n\,(n+1)/2$, as:

$$\mathrm{svec}\, X = [X_{11},\ \cdots,\ X_{1n},\ X_{21},\ \cdots,\ X_{2n},\ \cdots,\ X_{nn}]^T\,.$$

2 SPECTRAL NONLINEAR SDP METHOD

In this chapter we present the main aspects of our method (See [Thevenet et al., 2004] for a complete description).

2.1 General outline

The method we are about to present applies to more general classes of objective functions than (1). We shall consider matrix inequality constrained programs of the form

$$\begin{array}{ll} \text{minimize} & f(x),\ x \in \mathbb{R}^n \\ \text{subject to} & \mathcal{F}(x) \preceq 0, \end{array} \tag{3}$$

where f is a class $\mathcal{C}^2$ function, $\mathcal{F} : \mathbb{R}^n \to \mathbb{S}^m$ a class $\mathcal{C}^2$ operator. Our method is a penalty/barrier multiplier algorithm as proposed in [Zibulevsky, 1996, Mosheyev and Zibulevsky, 2000]. According to that terminology, a spectral penalty (SP) function for (3) is defined as

$$F(x,p) = f(x) + \mathrm{Tr}\,\Phi_p(\mathcal{F}(x)), \tag{4}$$

where $\varphi_p : \mathbb{R} \to \mathbb{R}$ is a parametrized family of scalar functions, which generates a family of matrix-valued operators $\Phi_p : \mathbb{S}^m \to \mathbb{S}^m$ upon defining:

$$\Phi_p(X) := S\mathrm{diag}\,[\varphi_p\,(\lambda_i(X))]\,S^T. \tag{5}$$

Here $\lambda_i(X)$ stands for the ith eigenvalue of $X \in \mathbb{S}^m$ and S is an orthonormal matrix of associated eigenvectors. An alternative expression for (4) using (5) is:

$$F(x,p) = f(x) + \sum_{i=1}^{m} \varphi_p(\lambda_i(\mathcal{F}(x))\,. \tag{6}$$

As φ_p is chosen strictly increasing and satisfies $\varphi_p(0) = 0$, each of the following programs (parametrized through a penalty $p > 0$)

$$\begin{array}{ll} \text{minimize} & f(x),\ x \in \mathbb{R}^n \\ \text{subject to} & \Phi_p(\mathcal{F}(x)) \preceq 0 \end{array} \tag{7}$$

is equivalent to (3). Thus, $F(x,p)$ may be understood as a penalty function for (3). Forcing $p \to 0$, we expect the solutions to the unconstrained program $\min_x F(x,p)$ to converge to a solution of (3). It is well-known that pure penalty methods run into numerical difficulties as soon as penalty constants get large. Similarly, using pure SP functions as in (4) would lead to ill-conditioning for small $p > 0$. The epoch-making idea of Hestenes [Hestenes, 1969] and Powell [Powell, 1969], known as the augmented Lagrangian approach, was to avoid this phenomenon by including a linear term carrying a Lagrange multiplier estimate into the objective. In the present context, we follow the same line, but incorporate Lagrange multiplier information by a nonlinear term. We define the augmented Lagrangian function associated with the matrix inequality constraints in (1) as

$$\begin{aligned} L(x,V,p) &= f(x) + \mathrm{Tr}\,\Phi_p(V^T\mathcal{F}(x)V)\,, \\ &= f(x) + \sum_{i=1}^{m} \varphi_p(\lambda_i(V^T\mathcal{F}(x)V)). \end{aligned} \tag{8}$$

In this expression, the matrix variable V has the same dimension as $\mathcal{F}(x) \in \mathbb{S}^m$ and serves as a factor of the Lagrange multiplier variable $U \in \mathbb{S}^m$, $U = VV^T$. In contrast with classical augmented Lagrangians, however, the Lagrange multiplier U is not involved linearly in (8). We nevertheless reserve the name of an augmented Lagrangian for $L(x,V,p)$, as its properties resemble those of the classical augmented Lagrangian. Schematically, the augmented Lagrangian technique is as follows:

Spectral augmented Lagrangian algorithm

1. **Initial phase.** Set constants $\gamma > 0, \rho < 1$. Initialize the algorithm with x_0, V_0 and a penalty parameter:

$$p_0 > 0. \tag{9}$$

2. **Optimization phase.** For fixed V_j and p_j solve the unconstrained subproblem

$$\text{minimize}_{x\in\mathbb{R}^n} \quad L(x, V_j, p_j) \tag{10}$$

Let x_{j+1} be the solution. Use the previous iterate x_j as a starting value for the inner optimization.

3. **Update penalty and multiplier.** Apply first-order rule to estimate Lagrange multiplier:

$$\begin{aligned} V_{j+1}V_{j+1}^T = V_j S\,\big[\mathrm{diag}\varphi_p'\,\big(\lambda_i\,\big(V_j^T \\ \cdot\mathcal{F}(x_{j+1})V_j)\big)\big]\,S^T V_j^T, \end{aligned} \tag{11}$$

where S diagonalizes $V_j^T\mathcal{F}(x_{j+1})V_j$.
Update the penalty parameter using:

$$p_{j+1} = \begin{cases} \rho p_j, & \text{if } \max(0, \lambda_{\max}(\mathcal{F}(x_{j+1}))) \\ & \quad > \gamma \max(0, \lambda_{\max}(\mathcal{F}(x_j))) \\ p_j, & \text{else} \end{cases} \tag{12}$$

Increase j and go back to step 2.

In our implementation, following the recommendation in

[Mosheyev and Zibulevsky, 2000], we have used the log-quadratic penalty function $\varphi_p(t) = p\varphi_1(t/p)$ where

$$\varphi_1(t) = \begin{cases} t + \frac{1}{2}t^2 & \text{if} \quad t \geq -\frac{1}{2} \\ -\frac{1}{4}\log(-2t) - \frac{3}{8} & \text{if} \quad t < -\frac{1}{2}, \end{cases} \tag{13}$$

but other choices could be used (see for instance [Zibulevsky, 1996] for an extended list).

The multiplier update formula (11) requires the full machinery of differentiability (see for instance [Thevenet et al., 2004] and [Lewis, 1996]) of the spectral function $\mathrm{Tr}\,\Phi_p$ and will not be derived here.

2.2 Solving the subproblem - implementational issues

Efficient minimization of $L(x, V_j, p_j)$ for fixed V_j, p_j is at the core of our approach and our implementation in the genuine BMI case is based on a Newton trust region method, following the lines of Lin and Moré [Lin and More, 1998]. As compared to Newton line search algorithms or other descent direction methods, trust regions can take better advantage of second-order information. This is witnessed by the fact that negative curvature in the tangent problem Hessian, frequently arising in BMI-minimization when iterates are still far away from any neighbourhood of local convergence, may be taken into account. Furthermore, trust region methods often (miraculously) find good local minima, leaving the bad ones behind. This additional benefit is in contrast with what line search methods achieve, and is explained to some extent by the fact that, at least over the horizon specified by the current trust region radius, the minimization in the tangent problem is a global one. This is why *specSDP* uses a trust region method to solve BMI problems.

However, for pure LMI (convex) problems, and when the conditioning is not too bad, the line search method may behave extremely well. As a result, the implementation default option for LMI problems in *specSDP* gives priority to the line search, but the user is still given the choice to switch to a trust region approach.

As implemented, both the trust region and line search variant of our method apply a Cholesky factorization to the tangent problem Hessian. This serves either to build an efficient preconditioner, or to compute the Newton direction in the second case. When the Hessian is close to indefinite, a modified Cholesky factorization is used ($\nabla^2 L + E = JJ^T$, with J a lower triangular matrix and E a shift matrix of minimal norm). The method from Lin and Moré [Lin and More, 1998] has been set as the default option, but the user may choose the variant from [Schnabel and Eskow, 1999], which is able to control efficiently the condition number of the shifted Hessian. This is sometimes useful for ill-conditioned problems.

3 USING specSDP

3.1 Calling specSDP

Hereafter we describe how to basically call the *specSDP* code from Matlab. This is done by executing the following command:

```
>> x = specSDP(constraints, objective)
```

in which 'x' is the value of the decision vector at the computed optimum, and *'constraints'*, *'objective'* are Matlab structures, including the considered problem data. This describes the standard call of *specSDP*, but extra optional arguments can be used and additional returned information can be requested. Indeed by typing:

```
>> [x,f,lmax,diagno]=specSDP(constraints,objective,X,opts);
```

from a Maltab prompt, the user will be returned some more information besides the optimum 'x': the objective function at the computed optimum, the maximum eigenvalue of the BMI and a dignostic concerning the success of the minimization. The structures *'X'* and *'opts'* enable the user to specify a panel of options when the default ones are deemed inappropriate. Both structures include several fields, among them an initial estimate of the solution, bounds on the variables, tolerances on the termination tests.... For convenience, the user may modify one or several fields of each structure without assigning the remaining ones. In that case, the unassigned fields will be set internally to the default values. All these features are detailed in the user's guide, which can be downloaded from

"http://www.cert.fr/dcsd/THESES/thevenet/publi.html".

Below is an example of how the previously mentioned structures should be built. Let us consider the following simple problem, with $nbvar$ $(= 3)$ variables, $nbmi$ $(= 2)$ BMI constraints, and a linear objective function:

$$\begin{array}{lll} \text{min.} & c^T x & \\ \text{s.t.} & A_0^1 + x_1 A_1^1 + x_2 A_2^1 + x_1 x_3 C_{13}^1 + x_2 x_3 C_{23}^1 \preceq 0 \\ & A_0^2 + x_2 A_2^2 + x_3 A_3^2 \preceq 0, \end{array} \tag{14}$$

where

$$c = [0\,0\,1]^T,\ x = [x_1\, x_2\, x_3]^T \in \mathbb{R}^3,$$

and

$$A_l^1 \in \mathbb{S}^2,\ C_{ij}^1 \in \mathbb{S}^2,\ A_l^2 \in \mathbb{S}^3,\ C_{ij}^2 \in \mathbb{S}^3,$$

for

$$l = \{0, ..., 3\},\ (i, j) \in \{1, 2\} \times \{3, 3\} := I_{ind} \times J_{ind},$$

$A_1^2,\ A_3^1,\ C_{13}^2,\ C_{23}^2$ being null matrices.

3.1.1 Structure *'constraints'*: this is the first argument that has to be passed when running *specSDP*. It requires a minimum number of fields, which are the following:

- ***'A0'***, which should be set as: $constraints.A_0 = \mathrm{svec}\, A_0$, where svec has been defined in section 1,

- ***'AAt'***, which must be built as: $\mathbf{constraints.AAt} = \mathbf{sparse}(\mathbf{AA^T})$, where

$$AA = \begin{bmatrix} \text{svec}\, A_1^1 & \cdots & \text{svec}\, A_{nbvar}^1 \\ \vdots & \vdots & \vdots \\ \text{svec}\, A_1^{nbmi} & \cdots & \text{svec}\, A_{nbvar}^{nbmi} \end{bmatrix}.$$

  The l^{th}column of AA is formed by stacking below each other the "svectorized" matrices A_l^k, where $k = 1, ..., nbmi$, that are associated with the l^{th} component x_l of the decision vector x in (14). Notice that, for convenience, the transposed matrix has to be passed, and that the Matlab sparse format should be used, since in many applications AA will have a limited number of nonzeros entries. *specSDP* has been designed to take advantage of sparse data in the computations.

- ***'dims'***, which is an integer row vector whose k^{th}component, $k = 1, ..., nbmi$, determines the size of the left-hand side matrix of the k^{th} LMI / BMI, what we will denote $\mathcal{B}^k(x)$ in the sequel. In our example we set:
  $constraints.dims = \begin{bmatrix} 2 & 3 \end{bmatrix}$.

When treating BMI cases some additional fields have to be passed :

- ***'Iind'*** and ***'Jind'***. These two fields are integer column vectors of length r, $(= 2$ in our example), the number of nonzeros nonlinear matrices C_{ij}, where:

$$C_{ij} = \begin{bmatrix} C_{ij}^1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & C_{ij}^{nbmi} \end{bmatrix},$$

  C_{ij}^k $(k = 1, .., nbmi)$ being as in (14).

  For any $m = 1, ..., r$, we define:

$$\begin{aligned} constraints.Iind(m) &= i, \\ constraints.Jind(m) &= j, \end{aligned}$$

  when referring to the previous notation, the considered nonlinear term being $x_i x_j C_{ij}$. Here this will give:

$$\begin{aligned} constraints.Iind &= \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \\ constraints.Jind &= \begin{bmatrix} 3 \\ 3 \end{bmatrix}. \end{aligned}$$

- ***'constraints.CCt'***. Again $constraints.CCt = sparse(CC^T)$, with:

$$CC = \begin{bmatrix} \text{svec}\, C_{I(1)J(1)}^1 & \cdots & \text{svec}\, C_{I(r)J(r)}^1 \\ \vdots & \vdots & \vdots \\ \text{svec}\, C_{I(1)J(1)}^{nbmi} & \cdots & \text{svec}\, C_{I(r)J(r)}^{nbmi} \end{bmatrix}.$$

3.1.2 Structure *'objective'*: now we focus on the second argument *'objective'* . It must have at least one field, ***'c'***, and at most two, ***'c'*** and ***'Q'***. The first situation arises when the objective function is linear, and the second one when it is quadratic.
Naturally, *'objective.c'* will be assigned the corresponding objective value ($objective.c = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ for the example above). As for *'objective.Q'*, the field will not be assigned in linear cases (as in (14)) and will be set to $\text{svec}\,(Q)$, as defined in section 1, when the objective is quadratic: $f(x) = c^T x + x^T Q x$, with $Q \in \mathbb{S}^n$. This way only the lower part of Q, including the diagonal, is stored, which avoids unwarranted extra storage.
Notice that the code is also able to solve pure feasibility problems, that is, programs without any objective function. In that case a null vector, with as many elements as the unknown vector 'x', has to be passed to the solver.
Finally, let us mention that *specSDP* has not been designed to solve genuine BMI problems with quadratic criterion. That means that ***'objective.Q'*** can be assigned only when the constraint is a pure LMI.

3.2 Display

Once the code has been launched, information is displayed in the course of the iterations, as shown below:

```
>> [xopt, f, lmax] = specSDP(constraints, objective, X);
 Initializing
 initial feasability -0.034267828520259266
 Number of variables :  10
 Number of BMIs :  2
 Size of BMIs :  4 3
 Version :  1

 Ext |    Obj.   |  Compl.   |   Stat.   | Int |  Tot. |    lmax    |  Penalty
   1 | 0.100E+03 | 0.278E-03 | 0.990E-01 |   1 |     1 | -.343E-01  | 0.100E-01
     |           |           |           |     |       | -.418E+00  | 0.100E-01
   2 | 0.100E+03 | 0.273E-03 | 0.000E+00 |   1 |     2 | -.343E-01  | 0.100E-01
     |           |           |           |     |       | -.418E+00  | 0.100E-01
   3 | 0.100E+03 | 0.269E-03 | 0.000E+00 |   1 |     3 | -.343E-01  | 0.100E-01
     |           |           |           |     |       | -.418E+00  | 0.100E-01
   4 | 0.558E-02 | 0.128E-01 | 0.994E+01 |  32 |    35 | 0.788E-02  | 0.500E-02
     |           |           |           |     |       | -.482E+00  | 0.500E-02
   5 | 0.299E-02 | 0.778E-02 | 0.259E-03 |  19 |    54 | 0.164E-02  | 0.250E-02
     |           |           |           |     |       | -.701E+00  | 0.250E-02
   6 | 0.154E-02 | 0.391E-02 | 0.144E-03 |  13 |    67 | 0.309E-03  | 0.125E-02
     |           |           |           |     |       | -.282E+00  | 0.125E-02
   7 | 0.960E-03 | 0.193E-02 | 0.583E-04 |  11 |    78 | 0.602E-06  | 0.625E-03
     |           |           |           |     |       | -.148E+00  | 0.625E-03
   8 | 0.960E-03 | 0.101E-02 | 0.000E+00 |   1 |    79 | 0.602E-06  | 0.313E-03
     |           |           |           |     |       | -.148E+00  | 0.313E-03
   9 | 0.960E-03 | 0.523E-03 | 0.000E+00 |   1 |    80 | 0.602E-06  | 0.156E-03
     |           |           |           |     |       | -.148E+00  | 0.156E-03
  10 | 0.424E-03 | 0.256E-03 | 0.536E-04 |   4 |    84 | -.727E-06  | 0.781E-04
     |           |           |           |     |       | -.664E-01  | 0.781E-04

 Ext |    Obj.   |  Compl.   |   Stat.   | Int |  Tot. |    lmax    |  Penalty
  11 | 0.188E-03 | 0.125E-03 | 0.235E-04 |   7 |    91 | 0.115E-06  | 0.391E-04
     |           |           |           |     |       | -.304E-01  | 0.391E-04
  12 | 0.188E-03 | 0.649E-04 | 0.000E+00 |   1 |    92 | 0.115E-06  | 0.195E-04
     |           |           |           |     |       | -.304E-01  | 0.195E-04
  13 | 0.988E-04 | 0.299E-04 | 0.894E-05 |  14 |   106 | 0.128E-07  | 0.977E-05
     |           |           |           |     |       | -.540E-02  | 0.977E-05
  14 | 0.983E-04 | 0.155E-04 | 0.510E-07 |   2 |   108 | -.241E-07  | 0.488E-05
     |           |           |           |     |       | -.515E-02  | 0.488E-05

Minimization succeeded
```

On the top of the shot, "initial feasibility" gives the maximum eigenvalue at the initial point x_0:

$$\lambda_1(\mathcal{B}(x_0)) = \max_{k=1,\,\dots,\,nbmi} (\lambda_1(\mathcal{B}^k(x_0)))$$

(On the example above, the initial point is "feasible").

Also general figures about the optimization problem

are provided, such as the number of variables involved, the number of LMIs / BMIs, their size..., as well as information about each "tangent" subproblem (current values of the objective function, penalty parameters, number of inner / outer iterations....)

4 APPLICATION: REDUCED ORDER FEEDBACK SYNTHESIS

In this section, we test our code against several examples of so-called reduced order control problems from the benchmark collection *COMPlib* [Leibfritz and Lipinsky, 2003]. Notice that we have only focused on stabilizing the closed-loop sytems, without minimizing any H_∞ (respectively H_2) performance. However *specSDP* could have treated those problems via the bounded real lemma ([Anderson and Vongpanitlerd, 1973]) for instance. Anyway the order of the resulting controllers has been reduced in many cases, when compared to the "best" compensators mentioned in *COMPlib*.

4.1 BMI characterization

We first recall a BMI characterization of the classical output feedback synthesis problem, before extending it to the fixed-order synthesis. To this end let $P(s)$ be an LTI (Linear Time Invariant) system with state-space equations:

$$P(s): \qquad \begin{bmatrix} \dot{x} \\ y \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix}, \tag{15}$$

where

- $x \in \mathbb{R}^n$ is the state vector,
- $u \in \mathbb{R}^m$ is the vector of control inputs,
- $y \in \mathbb{R}^p$ is the vector of measurements.

Our aim is to compute K such that the closed-loop system is internally stable: for $w = 0$ the state vector of the closed-loop system (15) and ($u = Ky$) tends to zero as time goes to infinity. It is well known that this problem is equivalent to finding K and a matrix $X \in \mathbb{S}^n$ such that:

$$\begin{aligned} (A+BKC)^T X + X(A+BKC) &\prec 0 \\ X &\succ 0. \end{aligned} \tag{16}$$

Fixed-order synthesis is concerned with the design of a dynamic controller $K(s) = C_K(sI - A_K)^{-1}B_K + D_K$ where $A_K \in \mathbb{R}^{k\times k}$ and $k < n$. It can be regarded as a static gain synthesis problem for an augmented system. Consequently, (16) applies if we perform the following substitutions:

$$\begin{aligned} K &\to \begin{bmatrix} A_K & B_K \\ C_K & D_K \end{bmatrix}, \quad A \to \begin{bmatrix} A & 0 \\ 0 & 0_k \end{bmatrix} \\ B &\to \begin{bmatrix} 0 & B \\ I_k & 0 \end{bmatrix}, \qquad C \to \begin{bmatrix} 0 & I_k \\ C & 0 \end{bmatrix}. \end{aligned} \tag{17}$$

4.2 Discussion

In pratice, attacking (16) with *specSDP* is somewhat awkward, since this formulation results in introducing an infinity of equivalent feasible points. Indeed, as soon as the couple (K_0, X_0) is feasible for the BMI (16), $(K_0, \alpha X_0)$ is also feasible for any scalar α. An accurate way to avoid this phenomenon would be to constraint explicitly the norm of the Lyapunov matrix X. Since *specSDP* has not been designed to handle equality constraints, we found that replacing the LMI in (16) by:

$$I \quad \succ \quad X \quad \succ \quad \epsilon I,$$

where $0 < \epsilon < 1$, was a good alternative. Notice that the LMI $X \succ \epsilon I$ is a safeguard against converging to the infeasible point $(0, 0)$. Paradoxically ϵ should be chosen small enough to allow a bad conditioning of X, as it is often the case when the real part of the closed-loop poles is close to the imaginary axis.
Similary, there exists for K an infinity of state-space realizations, which correspond to as many equivalent local minima, for a given X. As a result is is essential to give a particular structure to the controller. Although this choice might be discussed, we chose for K an observable companion structure, as it is easily implemented.

4.3 Numerical examples

The following table summerizes the results obtained on the examples from *COMPlib*, where n_x, n_u and n_y stand respectively for the number of states, the number of outputs and the number of inputs of the LTI plants. n_c denotes the smallest order of a stabilizing output feeback controller found with a previously existing approach. Finally n_{spec} gives the order of the stabilizing compensator obtained by *specSDP*. The notations "$n_{spec} = e$" (respectively "$n_{spec} = s$") mean that the order of the "best" controller found by *specSDP* is equal (respectively superior) to n_c.

Example	n_x	n_u	n_y	n_c	n_{spec}
(ROC1)	9	2	2	1	0
(ROC2)	10	2	3	1	e
(ROC3)	11	4	4	2	s
(ROC4)	9	2	2	1	0
(ROC5)	7	3	5	1	e
(ROC6)	5	3	3	2	1
(ROC7)	5	2	3	1	0
(ROC8)	9	4	4	3	e
(ROC9)	6	3	3	2	1
(ROC10)	6	2	4	1	s

Table 1: Reduced order control instances.

The table shows that on 5 examples out of 10, the order reduction given by *specSDP* represents an improvement over the best result obtained so far with other methods. Below are listed the corresponding controllers, with the same notation

as in (17), such that K_i denotes the controller obtained on the example (ROCi):

$$K_1 = \begin{bmatrix} \text{-96.9936} & 1.1757 \\ 12.2472 & \text{-0.3010} \end{bmatrix},$$

$$K_4 = \begin{bmatrix} \text{-100} & 0 \\ 0 & \text{-0.0917} \end{bmatrix},$$

$$K_6 = \begin{bmatrix} \text{-96.9315} & 1 & 80.5295 & 21.3260 \\ 83.9037 & \text{-96.6169} & 7.1625 & 10.8001 \\ 28.7578 & 94.3267 & \text{-99.8624} & \text{-32.8591} \\ \text{-49.6697} & \text{-37.1984} & 67.4075 & 20.3091 \end{bmatrix},$$

$$K_7 = \begin{bmatrix} \text{-81.0282} & \text{-80.6694} & 40.9775 \\ 82.9061 & 83.3068 & \text{-41.9307} \end{bmatrix},$$

$$K_9 = \begin{bmatrix} \text{-24.6240} & 1 & 4.9110 & \text{-1.2712} \\ 6.5284 & \text{-421.1192} & \text{-15.4894} & \text{-10.7716} \\ \text{-17.2231} & 3.9298 & \text{-57.1176} & 15.8847 \\ 13.3235 & \text{-10.0564} & \text{-3.3616} & 0.5402 \end{bmatrix}.$$

Furthermore, on 8 examples out of 10, *specSDP* was at least as efficient as other approaches, according to Table 1.
Finally, the solver failed to stabilize the models of the examples $(ROC3)$ (respectively $(ROC10)$) with controllers of order 2 (respectively 1). It is hard to determine precisely the cause of failure for a given problem, since many parameters are involved. However we found that balancing the data before solving and providing a "good" initial point were essential in many cases.

5 CONCLUSION

A spectral penalty augmented Lagrangian method for matrix inequality constrained nonlinear optimization programs and the associated software *specSDP* was presented in this paper. The algorithm performs robustly, as witnessed by the numerical examples in [Apkarian et al., 2004] and in [Thevenet et al., 2004], and finally by the applications of reduced order control synthesis considered in this paper. It is an efficient solver for small / medium scale examples, that is when the number of variables is up to a few hundreds of variables in the BMI case, up to 1500 variables in the LMI case. Moreover running the code is quite easy, as a limited number of data is needed for using it. A wide choice of optional arguments allows the user to make an expert use of it for particularly hard problems. Besides we remind the reader that, similarly to our previous approaches, the proposed algorithm, when tested against BMI-constrained (nonconvex) problems, is a local optimization method, which gives no certificate as to finding the global optimal solution of the program.

References

[Anderson and Vongpanitlerd, 1973] Anderson, B. D. O. and Vongpanitlerd, S. (1973). *Network Analysis*. Printice Hall, Englewood Cliffs.

[Apkarian et al., 2004] Apkarian, P., Noll, D., Thevenet, J. B., and Tuan, H. D. (2004). A Spectral Quadratic-SDP Method with Applications to Fixed-Order $\mathcal{H}_2$ and $\mathcal{H}_\infty$ Synthesis. In *Asian Control Conference, to be published in European Journal of Control*.

[Ben-Tal and Zibulevsky, 1997] Ben-Tal, A. and Zibulevsky, M. (1997). Penalty/barrier multiplier methods for convex programming problems. *SIAM J. on Optimization*, 7:347–366.

[Hestenes, 1969] Hestenes, M. R. (1969). Multiplier and gradient method. *J. Optim. Theory Appl.*, 4:303 – 320.

[Kocvara and Stingl, 2003] Kocvara, M. and Stingl, M. (2003). A Code for Convex Nonlinear and Semidefinite Programming. *Optimization Methods and Software*, 18(3):317–333.

[Leibfritz and Lipinsky, 2003] Leibfritz, F. and Lipinsky, W. (2003). Description of the benchmark examples in Complib 1.0. Technical report, University of Trier, Dpt. of Mathematics.

[Lewis, 1996] Lewis, A. (1996). Derivatives of spectral functions. *Mathematics of Operations Research*, 21:576–588.

[Lin and More, 1998] Lin, C. and More, J. (1998). Newton's method for large bound–constrained optimization problems. Technical Report ANL/MCS-P724–0898, Mathematics and Computer Sciences Division, Argonne National Laboratory.

[Mosheyev and Zibulevsky, 2000] Mosheyev, L. and Zibulevsky, M. (2000). Penalty/barrier multiplier algorithm for semidefinite programming. *Optimization Methods and Software*, 13(4):235–261.

[Powell, 1969] Powell, M. J. D. (1969). A method for nonlinear constraints in minimization problem. In Fletcher, R., editor, *Optimization*. Academic Press, London, New York.

[Schnabel and Eskow, 1999] Schnabel, R. B. and Eskow, E. (1999). A revised modified cholesky factorization algorithm. *SIAM J. on Optimization*, 9(4):1135–1148.

[Thevenet et al., 2004] Thevenet, J. B., Noll, D., and Apkarian, P. (2004). Nonlinear spectral SDP method for BMI-constrained problems : Applications to control design. In *ICINCO*, volume 1, pages 237–248, Setubal, Portugal.

[Zibulevsky, 1996] Zibulevsky, M. (1996). *Penalty/Barrier Multiplier Methods for Large-Scale Nonlinear and Semidefinite Programming*. Ph. D. Thesis, Technion Israël Institute of Technology.