COMPl$_e$ib 1.0 – user manual and quick reference

F. LEIBFRITZ * AND W. LIPINSKI †

Abstract. In this user manual, we present a short introduction to *COMPl$_e$ib* 1.0 – the *CO*nstrained *M*atrix–optimization *P*roblem *lib*rary [7], [8]. The problems are drawn from a variety of control systems engineering applications. The current version of *COMPl$_e$ib* contains a total of 124 problems coded in standard MATLAB matrix format. The advantage of this format is the platform independence. From the data defined in this test collection, it is possible to formulate several constraint matrix optimization problems, i. e. nonlinear semidefinite programs (NSDPs), bilinear matrix inequality (BMI) problems and other related matrix problems. Thus, the benchmark examples in *COMPl$_e$ib* can be used for testing and comparing a wide variety of algorithms. In particular, it is a useful tool for performing benchmarks of NSDP solvers (including BMI problem solvers) and other related matrix problem algorithms (i. e. linear SDP procedures). As a byproduct, *COMPl$_e$ib* may also serve as a direct tool for parts of control system design algorithms, i. e. algorithms for model reduction or Riccati equation solvers.

We provide a short overlook of the test examples contained in *COMPl$_e$ib* 1.0. Moreover, we state several NSDP and BMI formulations which can be built by the data defined in *COMPl$_e$ib* . Finally, we also present the usage of *COMPl$_e$ib* as a benchmark collection for linear SDP solvers like SeDuMi [11] or SDPT3 [13] in combination with the LMI parser YALMIP [10] for defining the individual SDP and calling a SDP solver (SeDuMi or SDPT3) in the MATLAB environment. Thus, we also define some linear SDPs which are deducible from the *COMPl$_e$ib* data. It is hoped that *COMPl$_e$ib* will stimulate the development of improved software for the solution of constraint matrix optimization problems, in particular, algorithms for solving nonlinear semidefinite programs.

1. Data structure and usage of *COMPl$_e$ib* . Release 1.0 of the constrained matrix optimization problem library *COMPl$_e$ib* consists of more than 120 examples collected from the engineering literature and real–life (engineering) applications for LTI control systems. In example, consider a LTI plant of order n_x with state space realization:

$$\begin{array}{rcl} \dot{x}(t) & = & Ax(t) + B_1 w(t) + Bu(t), \\ z(t) & = & C_1 x(t) + D_{11} w(t) + D_{12} u(t), \\ y(t) & = & Cx(t) + D_{21} w(t), \end{array} \tag{1.1}$$

where $x \in \boldsymbol{R}^{n_x}$, $u \in \boldsymbol{R}^{n_u}$, $y \in \boldsymbol{R}^{n_y}$, $z \in \boldsymbol{R}^{n_z}$, $w \in \boldsymbol{R}^{n_w}$ denote the state, control input, measured output, regulated output, and noise input, respectively. The current version of *COMPl$_e$ib* consists simply of the data matrices $A \in \boldsymbol{R}^{n_x \times n_x}$, $B_1 \in \boldsymbol{R}^{n_x \times n_w}$, $B \in \boldsymbol{R}^{n_x \times n_u}$, $C_1 \in \boldsymbol{R}^{n_z \times n_x}$, $D_{11} \in \boldsymbol{R}^{n_z \times n_w}$, $D_{12} \in \boldsymbol{R}^{n_z \times n_u}$, $C \in \boldsymbol{R}^{n_y \times n_x}$ and $D_{21} \in \boldsymbol{R}^{n_x \times n_w}$. In particular, all 124 test problems in *COMPl$_e$ib* 1.0 are coded and stored in standard MATLAB matrix format. We have decided to use this format, since the main advantage of MATLAB is the platform independence. The heart of *COMPl$_e$ib* is the MATLAB function file *COMPleib.m*. In a MATLAB environment, the data of the individual test example of *COMPl$_e$ib* can be accessed by calling the MATLAB function therein. For example, in MATLAB, the command

>> [A,B1,B,C1,C,D11,D12,D21,nx,nw,nu,nz,ny] = COMPleib('AC1');

returns the real data matrices A, B_1, B, C_1, C, D_{11}, D_{12} and D_{21} of (1.1) as well as the integers (dimension parameters) n_x, n_w, n_u, n_z and n_y of the *COMPl$_e$ib* example *AC1*. Together with the MATLAB function file *COMPleib.m*, *COMPl$_e$ib* is provided with several binary MATLAB data files (MAT-files) which contains the data matrices of some individual (large) test examples. In particular, release 1.0 of *COMPl$_e$ib* contains also the following MAT-files: *ac10.mat, ac13_14.mat, ac18.mat, bdt2.mat, cbm.mat, cdp.mat, cm1.mat – cm6.mat (6 files), dlr2_3.mat, he6.mat, he7.mat, hf2d1.mat – hf2d18.mat (18 files), ih.mat, iss1_2.mat, je1.mat, je2_3.mat, lah.mat, tl.mat.* Note, the name of the MAT-file corresponds to the example name in *COMPl$_e$ib* .

In order to use the *COMPleib* MATLAB function more comfortably, we offer also a MATLAB script file called *examplevector.m* with *COMPl$_e$ib* . It generates several MATLAB string vectors

*University of Trier, Department of Mathematics, D–54286 Trier, Germany. (leibfr@uni-trier.de)

†University of Trier, Department of Mathematics, D–54286 Trier, Germany. (lipi4501@uni-trier.de)

containing the names of a certain group of the *COMPl$_e$ib* examples. These vectors can be used to call several example subsets of *COMPl$_e$ib* in a row. Concretely, it provides the following MATLAB string vectors:

ExSOF	:	static output feedback examples	*ExCM*	:	cable mass models
ExAC	:	aircraft models	*Ex2om*	:	other second order models
ExHE	:	helicopter models	*ExROC*	:	reduced order control examples
ExJE	:	jet engine models	*Exlarge*	:	all large examples with $n_x \geq 50$
ExREA	:	reactor models	*Exsmall*	:	all small examples with $n_x < 50$
ExDIS	:	decentralized interconnected systems	*Exdense*	:	all dense examples (A dense)
ExWEC	:	wind energy conversion models	*Expsparse*	:	all partly sparse examples
ExEB	:	Euler Bernoulli beam models	*Exsparse*	:	all sparse examples, but
ExTF	:	terrain following examples	*Exsparse_large*	:	sparse examples of very high
ExNN	:	academic test problems			dimension saved in a sparse format
Exsof	:	other static output feedback examples	*Ex*	:	all 124 examples of *COMPl$_e$ib*
ExHF2D	:	2D heat flow 2D examples	*Ex_no_sl*	:	like *Ex* without *Exsparse_large*
Ex2OM	:	all second order models			

Note, more details on the individual example names and problem groups/classes of *COMPl$_e$ib* are given in Section 2 below (see also [8]). In a MATLAB environment, this script file can be used in combination with the MATLAB *COMPl$_e$ib* function *COMPleib*. For example, if we want to consider only the 18 aircraft examples of *COMPl$_e$ib* (AC1 – AC18), then we implement the following pseudo–code in MATLAB:

```
>> examplevector; % – script file defining example subsets of COMPl_eib
>> Set_of_Ex=ExAC; % – string vector of all aircraft examples in COMPl_eib 1.0
>> [No_Ex,dummy]=size(Set_of_Ex);
>> for i=1:No_Ex
>>     % – call COMPl_eib function
>>     [A,B1,B,C1,C,D11,D12,D21,nx,nw,nu,nz,ny]=COMPleib(Set_of_Ex(i,:));
>>     :
>>     :
>>     : (further matlab code)
>> end
```

Note, after calling the function *COMPleib*, from the example data of each aircraft model it is possible to built a special nonlinear semidefinite program or bilinear matrix inequality problem depending on the control design goal (i. e. see [7]). Thereafter, the developer of a corresponding NSDP or BMI solver can test his algorithm on this subset of *COMPl$_e$ib* .

2. Collection of test examples in *COMPl$_e$ib* release 1.0 . This section presents a short overview of all test examples which are currently implemented in *COMPl$_e$ib* 1.0. More detailed information about the origin (including references) and the application is given in our companion paper [8]. At the current stage, *COMPl$_e$ib* is divided into problem sets and the problem sets are grouped into problem classes. The problem classes and sets are listed in paragraphs below.

2.1. Static output feedback control examples . The first problem class of *COMPl$_e$ib* groups examples which are stabilizable by a static output feedback (SOF) control law, i. e.

$$u(t) = Fy(t), \quad \text{where} \quad F \in \boldsymbol{R}^{n_u \times n_y}.$$

For more details about SOF control design, i. e. see [7] and the references therein. In this class we have collected different models from the engineering literature. Table 2.1 provides a list of the individual example names used in *COMPl$_e$ib* , together with some other useful information: the dimension n_x of the data matrix A and n_u and n_y (number of controls and observations), respectively. Finally, we state a note on the structure of A, i. e. A is dense or sparse or partly sparse. In the last two cases, it may be possible to exploit the special structure of A in a solver using examples from *COMPl$_e$ib* . This group of examples can be subdivided into the following problem sets:

TABLE 2.1
Static output feedback examples

Example	n_x	n_u	n_y	Structure of A	Example	n_x	n_u	n_y	Structure of A
(AC1)	5	3	3	dense	(WEC3)	10	3	4	dense
(AC2)	5	3	3	dense	(HF1)	130	1	2	sparse
(AC3)	5	2	4	dense	(BDT1)	11	3	3	sparse
(AC4)	4	1	2	dense	(BDT2)	82	4	4	sparse
(AC5)	4	2	2	dense	(MFP)	4	3	2	dense
(AC6)	7	2	4	dense	(UWV)	8	2	2	dense
(AC7)	9	1	2	dense	(IH)	21	11	10	sparse
(AC8)	9	1	5	dense	(CSE1)	20	2	10	sparse
(AC9)	10	4	5	dense	(CSE2)	60	2	30	sparse
(AC10)	55	2	2	sparse	(EB1)	10	1	1	sparse
(AC11)	5	2	4	dense	(EB2)	10	1	1	sparse
(AC12)	4	3	4	dense	(EB3)	10	1	1	sparse
(AC13)	28	3	4	sparse	(EB4)	20	1	1	sparse
(AC14)	40	3	4	sparse	(EB5)	40	1	1	sparse
(AC15)	4	2	3	dense	(EB6)	160	1	1	sparse
(AC16)	4	2	4	dense	(PAS)	5	1	3	dense
(AC17)	4	1	2	dense	(TF1)	7	2	4	dense
(AC18)	10	2	2	dense	(TF2)	7	2	3	dense
(HE1)	4	2	1	dense	(TF3)	7	2	3	dense
(HE2)	4	2	2	dense	(PSM)	7	2	3	dense
(HE3)	8	4	6	dense	(TL)	256	2	2	dense
(HE4)	8	4	6	dense	(CDP)	120	2	2	sparse
(HE5)	8	4	2	dense	(NN1)	3	1	2	dense
(HE6)	20	4	6	dense	(NN2)	2	1	1	dense
(HE7)	20	4	6	dense	(NN3)	4	1	1	dense
(JE1)	30	3	5	partly sparse	(NN4)	4	2	3	dense
(JE2)	21	3	3	dense	(NN5)	7	1	2	dense
(JE3)	24	3	6	dense	(NN6)	9	1	4	dense
(REA1)	4	2	3	dense	(NN7)	9	1	4	dense
(REA2)	4	2	2	dense	(NN8)	3	2	2	dense
(REA3)	12	1	3	dense	(NN9)	5	3	2	dense
(REA4)	8	1	1	dense	(NN10)	8	3	3	dense
(DIS1)	8	4	4	dense	(NN11)	16	3	5	dense
(DIS2)	3	2	2	dense	(NN12)	6	2	2	dense
(DIS3)	6	4	4	dense	(NN13)	6	2	2	dense
(DIS4)	6	4	6	dense	(NN14)	6	2	2	dense
(DIS5)	4	2	2	dense	(NN15)	3	2	2	dense
(TG1)	10	2	2	dense	(NN16)	8	4	4	dense
(AGS)	12	2	2	sparse	(NN17)	3	2	1	dense
(WEC1)	10	3	4	dense	(NN18)	1006	1	1	sparse
(WEC2)	10	3	4	dense					

- Aircraft models *(AC)*
- Helicopter models *(HE)*
- Jet engine models *(JE)*
- Reactor models *(REA)*
- Decentralized interconnected systems *(DIS)*
- Euler Bernoulli beams *(EB)*
- Academic test problems *(NN)* and
- other examples from different applications like wind energy conversion models *(WEC)*, binary distillation towers *(BDT)*, terrain following models *(TF)*, compact disk player *(CDP)* and some more.

For some more details to each single example and problem sets we refer to [8].

2.2. 2D heat flow models [7] . In this section we state another group of examples arising in the design of static output feedback control laws for two dimensional heat flow models. Using standard finite difference schemes we obtain large scale finite dimensional approximations to the

infinite dimensional control problems (for more details, see [7, Section 3]). In the benchmark collection *COMPl$_e$ib* we state the data matrices of the corresponding discretized control systems. In particular, the discretization of the two dimensional heat flow models stated in [7, Section 3], [9] yields (in general) a large scale nonlinear and perturbed control system of the following form:

$$\begin{aligned} E\dot{x}(t) &= (A+\delta A)x(t) + G(x(t)) + B_1 w(t) + Bu(t), \quad x(0) = x_0, \\ z(t) &= C_1 x(t) + D_{12} u(t), \\ y(t) &= Cx(t), \end{aligned} \tag{2.1}$$

where $E \in \boldsymbol{R}^{n_x \times n_x}$ is a regular diagonal matrix and the matrices C_1 and D_{12} are defined by $C_1 = \sqrt{0.5c_1}\,[I_{n_x} \;\; 0_{n_u \times n_x}]^T$, $D_{12} = \sqrt{0.5d_1}\,[0_{n_x \times n_u} \;\; I_{n_u}]^T$ with given positive scalars $c_1, d_1 \in \boldsymbol{R}$. If $\delta A \equiv 0$ the system matrix A is not affected by a perturbation, and, if $G(x(t)) \equiv 0$, the system is linear. Depending on the corresponding heat flow model, one get linear or nonlinear control systems which can be controlled by a static output feedback control law of the form $u(t) = Fy(t)$, where $F \in \boldsymbol{R}^{n_u \times n_y}$ denotes an unknown SOF gain. For more details, we refer the interested reader to the case studies of these models in [7, Section 3] and [9]. Table 2.2 contains the list of the two dimensional

TABLE 2.2
2D heat flow models [7, Section 3]

Large model (sparse)				POD model (dense)				Property of	
Example	n_x	n_u	n_y	Example	n_x	n_u	n_y	A	model
(HF2D1)	3796	2	3	(HF2D10)	5	2	3	unstable	nonlinear
(HF2D2)	3796	2	3	(HF2D11)	5	2	3	unstable	nonlinear
(HF2D3)	4489	2	4	(HF2D12)	5	2	4	stable	linear
(HF2D4)	2025	2	4	(HF2D13)	5	2	4	stable	linear
(HF2D5)	4489	2	4	(HF2D14)	5	2	4	unstable	linear
(HF2D6)	2025	2	4	(HF2D15)	5	2	4	unstable	linear
(HF2D7)	4489	2	4	(HF2D16)	5	2	4	unstable	nonlinear
(HF2D8)	2025	2	4	(HF2D17)	5	2	4	unstable	nonlinear
(HF2D9)	3481	2	2	(HF2D18)	5	2	2	unstable	linear

heat flow models which can be found in *COMPl$_e$ib* . Note, a detailed discussion of these test examples is given in [7, Section 3] and [9]. The first nine examples represent the approximation of the discretized 2D heat flow models, while the other nine are the corresponding highly reduced order approximations of the large dimensional systems gained by the proper orthogonal decomposition (POD) approach as discussed in [9].

Due to the regularity of the so–called descriptor matrix E, it is possible to reduce the more general control system (2.1) to (1.1). In particular, if applicable, neglecting the nonlinear term $G(x(t)$ in (2.1), and redefining the data matrices $A + \delta A$, B_1, B by

$$A := E^{-1}(A + \delta A), \quad B_1 := E^{-1}B_1, \quad B := E^{-1}B$$

yields the equivalent standard linear system format (1.1) of *COMPl$_e$ib* . Thus, in *COMPl$_e$ib* , the corresponding MATLAB function returns the redefined data matrices of the (equivalent linearized) system if we call one of these heat flow models. The last column of Table 2.2 refers to the property of the data matrix A (stable/unstable) and states the source of the original model (linear/nonlinear). Finally, note, we have subdivided this group into following two problem sets: the first set contains all large scale and typically sparse models (HF2D1 – HF2D9) while the second problem set collects the low dimensional POD approximations (HF2D10 – HF2D18) of the large dimensional models.

2.3. Second order models. This example class of *COMPl$_e$ib* represents so–called second order models of the form:

$$M\ddot{q} + D\dot{q} + Sq = \hat{B}u, \quad M \text{ mass, } D \text{ damping, } S \text{ stiffness matrix} \tag{2.2}$$

which can be transformed into the first order standard system (1.1) by defining:

$$x := \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \quad A := \begin{bmatrix} 0 & I \\ -M^{-1}D & -M^{-1}S \end{bmatrix}, \quad B := \begin{bmatrix} 0 \\ M^{-1}\hat{B} \end{bmatrix}. \tag{2.3}$$

Note, in this case, the system matrices have a special structure. This is the reason why we have collected those problems in an extra class. But, note, all currently *COMPl$_e$ib* examples in this problem class are also SOF stabilizable which, in general, is not always true for second order models. Table 2.3 collects the second order benchmark examples which are currently available in

TABLE 2.3
Second order models

Example	n_x	n_u	n_y	Structure of A	Example	n_x	n_u	n_y	Structure of A
(CM1)	20	1	2	partly sparse	(DLR1)	10	2	2	dense
(CM2)	60	1	2	partly sparse	(DLR2)	40	2	2	sparse
(CM3)	120	1	2	partly sparse	(DLR3)	40	2	2	sparse
(CM4)	240	1	2	partly sparse	(ISS1)	270	3	3	sparse
(CM5)	480	1	2	partly sparse	(ISS2)	270	3	3	sparse
(CM6)	960	1	2	partly sparse	(CBM)	348	1	1	partly sparse
(TMD)	6	2	4	dense	(LAH)	48	1	1	partly sparse
(FS)	5	1	3	dense					

COMPl$_e$ib 1.0. In this table, one can also find the dimension n_x of A and the information about a possible sparsity pattern of A. Briefly this group consists of the following problem sets:

- six so–called cable mass models with very low damping *(CM)*
- three models of a space structure developed by the "Deutsche Forschungsanstalt für Luft- und Raumfahrt" *(DLR)*
- two instances of a component of the International Space Station *(ISS)*
- some other second order models, i. e. a tuned mass damper *(TMD)* example, a clamped beam model *(CBM)*, a flexible satellite *(FS)* example, and, finally, a model of the Los Angeles (university) hospital *(LAH)*

2.4. Reduced order control examples . The last group of examples in *COMPl$_e$ib* 1.0 are so–called reduced order control problems. These instances are not SOF stabilizable, but they are at least stabilizable by a reduced order output feedback control law of order $n_c \ll n_x$. Using a well known system augmentation technique, the considered system has the following form

$$\begin{aligned}
\begin{bmatrix} \dot{x}(t) \\ \dot{x}_c(t) \end{bmatrix} &= \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \end{bmatrix} + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} w(t) + \begin{bmatrix} 0 & B \\ I & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_c(t) \\ u(t) \end{bmatrix} \\
z(t) &= \begin{bmatrix} C_1 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \end{bmatrix} + D_{11} w(t) + \begin{bmatrix} 0 & D_{12} \end{bmatrix} \begin{bmatrix} \dot{x}_c(t) \\ u(t) \end{bmatrix} \\
\begin{bmatrix} x_c(t) \\ y(t) \end{bmatrix} &= \begin{bmatrix} 0 & I \\ C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_c(t) \end{bmatrix} + \begin{bmatrix} 0 \\ D_{21} \end{bmatrix} w(t)
\end{aligned} \tag{2.4}$$

and the reduced order (dynamic) output feedback control law

TABLE 2.4
Reduced order control instances

Example	n_x	n_u	n_y	n_c	Structure of A	Example	n_x	n_u	n_y	n_c	Structure of A
(ROC1)	9	2	2	1	dense	(ROC6)	5	3	3	2	dense
(ROC2)	10	2	3	1	dense	(ROC7)	5	2	3	1	dense
(ROC3)	11	4	4	2	dense	(ROC8)	9	4	4	3	dense
(ROC4)	9	2	2	1	dense	(ROC9)	6	3	3	2	dense
(ROC5)	7	3	5	1	dense	(ROC10)	6	2	4	1	dense

$$\begin{bmatrix} \dot{x}_c(t) \\ u \end{bmatrix} = F \begin{bmatrix} x_c(t) \\ y \end{bmatrix}, \quad F := \begin{bmatrix} A_c & B_c \\ C_c & D_c \end{bmatrix} \tag{2.5}$$

looks like a static output feedback controller, where $x_c \in \boldsymbol{R}^{n_c}$, $A_c \in \boldsymbol{R}^{n_c \times n_c}$, $B_c \in \boldsymbol{R}^{n_c \times n_y}$, $C_c \in \boldsymbol{R}^{n_u \times n_c}$, $D_c \in \boldsymbol{R}^{n_u \times n_y}$ and n_c denotes the ROC state. Note: $n_c = 0$ leads to the original SOF control law. For more details, we refer the interested reader to [7], [9] and the references therein.

Table 2.4 gives an overview of the currently implemented ROC problems. Therein, to our knowledge, n_c denotes the smallest possible order of the reduced output feedback controller which can be used for stabilizing the control system.

3. Constrained matrix–optimization problems deducible from *COMPl$_e$ib* . In this section we state a list of several constrained matrix–optimization problems which can be built by using the data matrices defined in *COMPl$_e$ib* . For a more detailed discussion of the underlying feedback control problems and derivation of the corresponding matrix optimization problems we refer to [1], [2], [3], [4], [6], [7], [5], [12], [14] and the references therein.

In the following subsections we collect several constrained matrix optimization problems which arise in feedback design control problems. Note, however, there are much more contributions in the control literature of problems leading to NSDPs, BMIs or SDPs. Thus, our list is far from being exhaustive.

All matrix optimization problems listed below can be easily formed from the data matrices provided by *COMPl$_e$ib* . A potential user of *COMPl$_e$ib* should take this uncomplete list of NSDPs, BMIs and SDPs as a cook book for building several matrix optimization problems from the data given in *COMPl$_e$ib* . Note, depending on the underlying (control) design problem, it is indeed possible to derive nonlinear semidefinite programs, or, equivalently, bilinear matrix inequality problems, or, even, linear semidefinte programs from the data matrices stated in *COMPl$_e$ib* .

In the subsections below, we often use the following matrix functions mapping a real matrix $F \in \boldsymbol{R}^{n_u \times n_y}$ to $\boldsymbol{R}^{n_x \times n_x}$ or $\boldsymbol{R}^{n_x \times n_w}$ or $\boldsymbol{R}^{n_z \times n_x}$ or $\boldsymbol{R}^{n_z \times n_w}$, respectively. Using the data matrices *COMPl$_e$ib* , these matrix functions are defined by

$$A(F) = A + BFC, \quad B(F) = B_1 + BFD_{21}, \quad C(F) = C_1 + D_{12}FC, \quad D(F) = D_{11} + D_{12}FD_{21},$$

respectively.

3.1. Defining nonlinear semidefinite programs by the *COMPl$_e$ib* data. In this subsection we list only some NSDPs which can be modelled by the data given in *COMPl$_e$ib* .

3.1.1. $\mathcal{H}_2$–NSDP. Using the data in *COMPl$_e$ib* , we state the following NSDP in the unknowns $F \in \boldsymbol{R}^{n_u \times n_y}, Q = Q^T \in \boldsymbol{R}^{n_x \times n_x}$ and $V = V^T \in \boldsymbol{R}^{n_x \times n_x}$:

$$\begin{array}{ll} \min_{F,Q,V} & Tr((C_1 + D_{12}FC)Q(C_1 + D_{12}FC)^T) \\ \text{s. t.} & (A + BFC)Q + Q(A + BFC)^T + B_1B_1^T = 0, \\ & (A + BFC)V + V(A + BFC)^T + I = 0,\ V \succ 0, \end{array} \tag{3.1}$$

where $V \succ 0$ denotes that V is positive definite.

An equivalent formulation of (3.1) is the following NSDP:

$$\begin{array}{ll} \min_{F,P,W} & Tr(PB_1B_1^T) \\ \text{s. t.} & (A + BFC)^TP + P(A + BFC) + (C_1 + D_{12}FC)^T(C_1 + D_{12}FC) = 0, \\ & (A + BFC)^TW + W(A + BFC) \prec 0,\ W \succ 0, \end{array} \tag{3.2}$$

in the matrix variables $F \in \boldsymbol{R}^{n_u \times n_y}, P = P^T \in \boldsymbol{R}^{n_x \times n_x}$ and $W = W^T \in \boldsymbol{R}^{n_x \times n_x}$.

Note, both NSDPs are nonlinear and non–convex matrix optimization problems subject to a constraint set of nonlinear matrix equations and inequalities. Moreover, these problems are completely defined by the example data in *COMPl$_e$ib* .

3.1.2. $\mathcal{H}_\infty$**–NSDP.** A very general version of the $\mathcal{H}_\infty$–NSDP is the following non–convex and nonlinear semidefinite program:

$$\begin{array}{ll} \min_{F,P,W,\gamma} & \gamma \\ \text{s. t.} & A(F)^TP + PA(F) + \gamma^{-1}C(F)^TC(F) + \gamma^{-1}M(F,P,\gamma)R(F,\gamma)^{-1}M(F,P,\gamma)^T = 0, \\ & \tilde{A}(F,P,\gamma)^TW + W\tilde{A}(F,P,\gamma) + I = 0, \;\; R(F,\gamma) \succ 0 \;\; W \succ 0, \;\; P \succeq 0, \;\; \gamma > 0, \end{array} \tag{3.3}$$

where the matrix functions $R : \boldsymbol{R}^{n_u \times n_y} \times \boldsymbol{R} \to \mathcal{S}^{n_w}$ and $M : \boldsymbol{R}^{n_u \times n_y} \times \mathcal{S}^{n_x} \times \boldsymbol{R} \to \boldsymbol{R}^{n_x \times n_w}$ are defined by

$$R(F,\gamma) = I_{n_w} - \gamma^{-2}D(F)^TD(F), \quad M(F,P,\gamma) = PB(F) + \gamma^{-1}C(F)^TD(F). \tag{3.4}$$

Furthermore, the non–symmetric matrix function $\tilde{A} : \boldsymbol{R}^{n_u \times n_y} \times \mathcal{S}^{n_x} \times \boldsymbol{R} \to \boldsymbol{R}^{n_x \times n_x}$ is given by

$$\tilde{A}(F,P,\gamma) = A(F) + \gamma^{-1}B(F)R(F,\gamma)^{-1}M(F,P,\gamma)^T. \tag{3.5}$$

Note, $\mathcal{S}^n$ denotes the set of all real symmetric $(n \times n)$–matrices. This version of the $\mathcal{H}_\infty$–NSDP is highly nonlinear and non–convex in the free variables $\gamma \in \boldsymbol{R}$, $F \in \boldsymbol{R}^{n_u \times n_y}$, $P = P^T \in \boldsymbol{R}^{n_x \times n_x}$ and $W = W^T \in \boldsymbol{R}^{n_x \times n_x}$.

A simplified NSDP version of (3.3) is given by

$$\begin{array}{ll} \min_{F,P,W,\gamma} & \gamma \\ \text{s. t.} & A(F)^TP + PA(F) + \gamma^{-1}C(F)^TC(F) + \gamma^{-1}PB_1B_1^TP = 0, \;\; P \succeq 0 \\ & (A(F) + \gamma^{-1}B_1B_1^TP)^TW + W(A(F) + \gamma^{-1}B_1B_1^TP) + I = 0, \;\; W \succ 0, \;\; \gamma > 0. \end{array} \tag{3.6}$$

Finally, we state the following equivalent formulations of the $\mathcal{H}_\infty$–NSDPs (3.3) and (3.6). An alternative formulation of (3.3) has the following form:

$$\begin{array}{ll} \min_{F,Q,V,\gamma} & \gamma \\ \text{s. t.} & A(F)Q + QA(F)^T + \gamma^{-1}B(F)B(F)^T + \gamma^{-1}\hat{M}(F,Q,\gamma)^T\hat{R}(F,\gamma)^{-1}\hat{M}(F,Q,\gamma) = 0, \\ & \hat{A}(F,Q,\gamma)V + V\hat{A}(F,Q,\gamma)^T + I = 0, \;\; \hat{R}(F,\gamma) \succ 0 \;\; V \succ 0, \;\; Q \succeq 0, \;\; \gamma > 0, \end{array} \tag{3.7}$$

where $\gamma \in \boldsymbol{R}$, $F \in \boldsymbol{R}^{n_u \times n_y}$, $Q = Q^T \in \boldsymbol{R}^{n_x \times n_x}$ and $V = V^T \in \boldsymbol{R}^{n_x \times n_x}$ are the free variables and the matrix functions $\hat{R}, \hat{M}, \hat{A}$ are given by

$$\hat{R}(F,\gamma) := I_{n_z} - \gamma^{-2}D(F)D(F)^T, \quad \hat{M}(F,Q,\gamma) := C(F)Q + \gamma^{-1}D(F)B(F)^T,$$

$$\hat{A}(F,Q,\gamma) := A(F) + \gamma^{-1}\hat{M}(F,Q,\gamma)^T\hat{R}(F,\gamma)^{-1}C(F).$$

Moreover, if $D_{11} = 0$ and $D_{21} = 0$, (3.7) reduces to

$$\begin{array}{ll} \min_{F,Q,V,\gamma} & \gamma \\ \text{s. t.} & A(F)Q + QA(F)^T + \gamma^{-1}B_1B_1^T + \gamma^{-1}QC(F)^TC(F)Q = 0, \;\; Q \succeq 0, \;\; \gamma > 0, \\ & (A(F) + \gamma^{-1}QC(F)^TC(F))V + V(A(F) + \gamma^{-1}QC(F)^TC(F))^T + I = 0, \;\; V \succ 0, \end{array} \tag{3.8}$$

which in turn is equivalent to (3.6).

3.1.3. $\mathcal{H}_2/\mathcal{H}_\infty$**–NSDPs.** Fixing $\gamma > 0$, a combination of the two problem classes stated in the previous two subsections motivates the following (simplified) version of the $\mathcal{H}_2/\mathcal{H}_\infty$–NSDP (see also (3.1) and (3.8)):

$$\begin{array}{ll} \min_{F,Q,V} & Tr((C_1 + D_{12}FC)Q(C_1 + D_{12}FC)^T) \\ \text{s. t.} & A(F)Q + QA(F)^T + \gamma^{-1}B_1B_1^T + \gamma^{-1}QC(F)^TC(F)Q = 0, \;\; Q \succeq 0 \\ & (A(F) + \gamma^{-1}QC(F)^TC(F))V + V(A(F) + \gamma^{-1}QC(F)^TC(F))^T + I = 0, \;\; V \succ 0. \end{array} \tag{3.9}$$

By using a dualization argument, the $\mathcal{H}_2/\mathcal{H}_\infty$–NSDP (3.9) is equivalent to the following nonlinear semidefinite program (see also (3.2) and (3.6)):

$$\begin{array}{ll} \min_{F,P,W} & Tr(PB_1B_1^T) \\ \text{s. t.} & A(F)^TP + PA(F) + \gamma^{-1}C(F)^TC(F) + \gamma^{-1}PB_1B_1^TP = 0, \;\; P \succeq 0 \\ & (A(F) + \gamma^{-1}B_1B_1^TP)^TW + W(A(F) + \gamma^{-1}B_1B_1^TP) + I = 0, \;\; W \succ 0. \end{array} \tag{3.10}$$

3.2. Building bilinear matrix optimization problems with *COMPl$_e$ib* .

3.2.1. $\mathcal{H}_2$–BMI problem. An equivalent BMI formulation of (3.1) is given by

$$\begin{array}{ll} \min_{F,Q,X} & Tr(X) \\ \text{s. t.} & (A+BFC)Q + Q(A+BFC)^T + B_1B_1^T \preceq 0, \; Q \succ 0 \\ & \left[\begin{array}{cc} X & (C_1+D_{12}FC)Q \\ Q(C_1+D_{12}FC)^T & Q \end{array} \right] \succeq 0. \end{array} \tag{3.11}$$

Note, (3.11) is bilinear in F and Q, but it is still non–convex due to the bilinearity of the free matrix variables $F \in \boldsymbol{R}^{n_u \times n_y}$, $Q = Q^T \in \boldsymbol{R}^{n_x \times n_x}$ and $X = X^T \in \boldsymbol{R}^{n_z \times n_z}$. In this formulation, it is necessary to assume that $B_1B_1^T \succ 0$. If this is not satisfied (i. e. in *COMPl$_e$ib*), one can use $B_1B_1^T + \epsilon I_{n_x}$ instead of $B_1B_1^T$ for a fixed small positive scalar ϵ.

3.2.2. $\mathcal{H}_\infty$–BMI problems. A BMI version of the optimal fixed order $\mathcal{H}_\infty$ synthesis problem (see, i. e. [3, Problem 2]) can be stated as follows:

$$\min_{F,X,\gamma} \; \gamma \;\; \text{s. t.} \;\; X \succ 0, \quad \gamma > 0, \quad \left[\begin{array}{ccc} A(F)^TX + XA(F) & XB(F) & C(F)^T \\ B(F)^TX & -\gamma\, I_{n_w} & D(F)^T \\ C(F) & D(F) & -\gamma\, I_{n_z} \end{array} \right] \prec 0. \tag{3.12}$$

Note, this BMI problem is equivalent to (3.3). Due to the bilinearity of the free matrix variables $F \in \boldsymbol{R}^{n_u \times n_y}$ and $X = X^T \in \boldsymbol{R}^{n_x \times n_x}$, the BMI problem (3.12) is a non–convex and nonlinear constrained matrix–optimization problem. Moreover, note, the left hand side of the matrix inequality lies in $\mathcal{S}^{n_x+n_w+n_z}$.

Finally, the BMI–problem

$$\min_{F,Y,\gamma} \; \gamma \;\; \text{s. t.} \;\; Y \succ 0, \quad \gamma > 0, \quad \left[\begin{array}{ccc} A(F)Y + YA(F)^T & YC(F)^T & B(F) \\ C(F)Y & -\gamma\, I_{n_z} & D(F) \\ B(F)^T & D(F)^T & -\gamma\, I_{n_w} \end{array} \right] \prec 0. \tag{3.13}$$

is equivalent to (3.12).

3.2.3. $\mathcal{H}_2/\mathcal{H}_\infty$–BMI problems. The NSPD (3.9) is equivalent to the $\mathcal{H}_2/\mathcal{H}_\infty$–BMI problem (cf. (3.11) and (3.13)):

$$\begin{array}{ll} \min_{F,Q,X} & Tr(X) \\ \text{s. t.} & \left[\begin{array}{cc} A(F)Y + YA(F)^T + \gamma^{-1}B_1B_1^T & YC(F)^T \\ C(F)Y & -\gamma\, I_{n_z} \end{array} \right] \prec 0, \\ & \left[\begin{array}{cc} X & C(F)Y \\ YC(F)^T & Y \end{array} \right] \succeq 0, \quad Y \succ 0. \end{array} \tag{3.14}$$

Moreover, the NSPD (3.10) is equivalent to the BMI problem (see also (3.12):

$$\begin{array}{ll} \min_{F,X} & Tr(XB_1B_1^T) \\ \text{s. t.} & X \succ 0, \quad \left[\begin{array}{ccc} A(F)^TX + XA(F) & XB_1 & C(F)^T \\ B_1^TX & -\gamma\, I_{n_w} & 0 \\ C(F) & 0 & -\gamma\, I_{n_z} \end{array} \right] \prec 0. \end{array} \tag{3.15}$$

3.3. Forming linear semidefinite programs from the *COMPl$_e$ib* data. For some special cases (i. e. state feedback control design), it is well known that most of the problems above has a linear SDP counterpart. In particular, setting $C := I_{n_x}$ and $D_{21} := 0$ (i. e. $y = x$, $n_y = n_x$), a clever change of variable leads to linear SDP formulations of the underlying (special) control problems. In this subsection, we state only a very uncomplete list of such SDPs. For more SDP instances, we refer the interested reader to [2] and the references therein.

3.3.1. $\mathcal{H}_2$–SDP problem. Assuming $B_1 B_1^T \succ 0$, the linear $\mathcal{H}_2$–SDP problem is defined by:

$$\begin{aligned} &\min_{X,Y,Q} \ \mathrm{Tr}\,(X) \\ \text{s.t.} \quad &\begin{bmatrix} X & C_1 Q + D_{12} Y \\ (C_1 Q + D_{12} Y)^T & Q \end{bmatrix} \succeq 0, \\ &A Q + Q A^T + B Y + Y^T B^T + B_1 B_1^T \prec 0, \quad Q \succ 0, \end{aligned} \tag{3.16}$$

where $Q = Q^T \in \boldsymbol{R}^{n_x \times n_x}$, $X = X^T \in \boldsymbol{R}^{n_z \times n_z}$ and $Y \in \boldsymbol{R}^{n_u \times n_x}$ are the optimization variable and in an optimal solution of (3.16), one can define $F = YQ^{-1} \in \boldsymbol{R}^{n_u \times n_x}$. Note, if $B_1 B_1^T \succ 0$ is not satisfied in *COMPl$_e$ib* , in the definition of (3.16) we replace $B_1 B_1^T$ by $B_1 B_1^T + \epsilon I_{n_x}$, where ϵ denotes a fixed small positive scalar.

3.3.2. $\mathcal{H}_\infty$–SDP problem. Similarly as in the subsection above, we get again a linear SDP version of the $\mathcal{H}_\infty$ problem, i. e. we have

$$\begin{aligned} &\min_{Q,Y,\gamma} \ \gamma \\ \text{s.t.} \quad &\gamma > 0, \ \ Q \succ 0, \ \ \begin{bmatrix} AQ + QA + BY + Y^T B^T & QC_1^T + Y^T D_{12}^T & B_1 \\ C_1 Q + D_{12} Y & -\gamma I_{n_z} & D_{11} \\ B_1^T & D_{11}^T & -\gamma I_{n_w} \end{bmatrix} \prec 0. \end{aligned} \tag{3.17}$$

where $Q = Q^T \in \boldsymbol{R}^{n_x \times n_x}$, $\gamma \in \boldsymbol{R}$ and $Y \in \boldsymbol{R}^{n_u \times n_x}$ are the free variable and in an optimal solution of (3.17), we set $F = YQ^{-1} \in \boldsymbol{R}^{n_u \times n_x}$.

3.3.3. $\mathcal{H}_2/\mathcal{H}_\infty$–SDP problem. The linear $\mathcal{H}_2/\mathcal{H}_\infty$–SDP is a combination of the SDPs defined in the last two paragraphs. For a given scalar $\gamma > 0$, the following linear problem represents one version of the $\mathcal{H}_2/\mathcal{H}_\infty$–SDP:

$$\begin{aligned} &\min_{Q,X,Y} \ \ \mathrm{Tr}\,(X) \\ \text{s.t.} \quad &\begin{bmatrix} AQ + QA + BY + Y^T B^T & QC_1^T + Y^T D_{12}^T & B_1 \\ C_1 Q + D_{12} Y & -\gamma I_{n_z} & 0 \\ B_1^T & 0 & -\gamma I_{n_w} \end{bmatrix} \prec 0, \\ &Q \succ 0, \ \ \begin{bmatrix} X & C_1 Q + D_{12} Y \\ (C_1 Q + D_{12} Y)^T & Q \end{bmatrix} \succeq 0, \end{aligned} \tag{3.18}$$

where $Q = Q^T \in \boldsymbol{R}^{n_x \times n_x}$, $X = X^T \in \boldsymbol{R}^{n_z \times n_z}$ and $Y \in \boldsymbol{R}^{n_u \times n_x}$ are the free variable and in an optimal solution of (3.18), we define $F = YQ^{-1} \in R^{n_u \times n_x}$.

4. Using *COMPl$_e$ib* as a benchmark collection for linear SDPs. In this section we present the usage of *COMPl$_e$ib* as a benchmark collection for linear SDP solvers like SeDuMi 1.05 [11] or SDPT3 [13] in combination with the LMI parser YALMIP 2.4[10] for defining the individual SDP and calling a SDP solver (SeDuMi or SDPT3) in the MATLAB environment. In particular, we have performed some test runs of the well known SDP solvers SeDuMi [11] and SDPT3 [13] on all test examples from *COMPl$_e$ib* 1.0. For building and solving the linear SDPs (3.16), (3.17) in MATLAB on *COMPl$_e$ib* we have used YALMIP [10] in combination with SeDuMi and SDPT3, respectively.

The results of our test runs can be found in the tables at the end of this paper. All computations results given therein were obtained on a Dell Pentium 4 PC (2 GHz CPU) with 2 GB of memory running WINDOWS 2000, using MATLAB 5.2. In each table, we list the *COMPl$_e$ib* example name (i. e. AC1), the number of optimization variables (# var.) and the number of constraints (# const.) of the corresponding SDP (3.16) or (3.17), the dimension parameters n_x, n_z, n_u, the number of iterations required until termination (# iter.) and the time in CPU-seconds (CPU–sec.). The last two columns indicate if the solver has found the optimal solution or not (fail ? yes/no). In the case of failure, we set the variable *fail* to *yes* and report the corresponding error message of the solver in the last column. On the other hand, if the algorithm terminates without any error and within the desired accuracy ($1.0 \cdot 10^{-7}$), we set *fail* to *no*.

The SDP (3.16) is a linear matrix problem in the variables $Q \in \mathcal{S}^{n_x}$, $Y \in \boldsymbol{R}^{n_u \times n_x}$ and $X \in \mathcal{S}^{n_z}$. Thus, the total number of unknowns is given by

$$\frac{1}{2}\left(n_x(n_x+1)\right) + \frac{1}{2}\left(n_z(n_z+1)\right) + n_x n_u$$

and the total number of constraints can be calculated by $2n_x^2 + (n_z+n_x)^2$.

Table 4.2 shows the results of SeDuMi for the solution of (3.16) on the whole test set of *COMPl$_e$ib* . Therein we observe that SeDuMi is not able to solve the large problems in *COMPl$_e$ib* .

Table 4.3 states the results of SDPT3 for solving (3.16). In this table, we did not list those examples for which SDPT3 runs out of memory (i. e. all examples in *COMPl$_e$ib* with $n_x > 150$). Or in other words, SDPT3 also was not able to solve the larger problems contained in *COMPl$_e$ib* .

The results of SeDuMi and SDPT3 for solving the $\mathcal{H}_\infty$–SDP (3.17) in combination with YALMIP on all *COMPl$_e$ib* examples are reported in Tables 4.4 and 4.5, respectively. Again, in Table 4.5 we skipped the examples where SDPT3 terminates due to memory problems.

In (3.17) the variables are $Q \in \mathcal{S}^{n_x}$, $Y \in \boldsymbol{R}^{n_u \times n_x}$ and $\gamma \in \boldsymbol{R}$. Hence, the total number of unknowns is defined by

$$\frac{1}{2}\left(n_x(n_x+1)\right) + n_x n_u + 1$$

and the number of constraints can be calculated by $(n_x+n_z+n_w)^2+n_x^2+1$. For these test runs, we can draw the same observations as before. Both solvers are not able to solve the larger examples which are contained in *COMPl$_e$ib* . Moreover, comparing both solvers, it seems that SeDuMi is more robust than SDPT3 since SeDuMi is able to solve more problems than SDPT3.

Note, the larger examples in *COMPl$_e$ib* were mostly not solvable because of memory problems, i. e. the solvers returns the error message *out of memory*. In order to solve also larger problem we ran the *COMPl$_e$ib* example *CDP* on a Sun Blade 1000 (Solaris 8) with 8 GB RAM and 2 UltraSparcIII CPU's (750 MHz). We used this example to solve the $\mathcal{H}_\infty$–SDP with SeDuMi in combination with YALMIP. Now, SeDuMi was able to terminate without any error message, but SeDuMi needs 22 hours, 41 minutes and 67 seconds for determining the optimal solution of (3.17) for CDP.

Finally, we present a pseudo–code MATLAB script file which we have used for solving the $\mathcal{H}_2$–SDP (3.16) with SeDuMi on *COMPl$_e$ib* in combination with YALMIP. Note, the script file for obtaining the other benchmark results looks very similar.

```
% - - - Solve linear H2–SDP by SeDuMi on COMPleib
examplevector; % – script file defining subsets of the COMPleib 1.0 examples
Set_of_Ex = Ex; % – string vector of all examples in COMPleib 1.0
[No_Ex,dummy] = size(Set_of_Ex);
.
:   (other matlab code if necessary)
for i=1:No_Ex, % – Solve linear SDP for each COMPleib test example
    % – Load COMPleib test example
    [A,B1,B,C1,C,D11,D12,D21,nx,nw,nu,nz,ny] = COMPleib(Set_of_Ex(i,:));
```

```
No_Var_Solver = 0.5 * ( nx*(nx+1) + nz*(nz+1) ) + nx*nu; % – # of variables
No_Con_Solver = 2*nx^2 + (nx+nz)^2; % – # of constraints
  if(B1*B1' ~= 0), B1 := I end
  % – Define linear H2–SDP using YALMIP
  yalmip('clear');
  Q = sdpvar(nx,nx,'symmetric','real'); % – matrix variable Q
  X = sdpvar(nz,nz,'symmetric','real'); % – matrix variable X
  Y = sdpvar(nu,nx,'full','real'); % – matrix variable Y:=FQ
  % – Define LMIs
  lmi1 = lmi('[X   C1*Q+D12*Y; Q*C1''+Y''*D12''   Q] > 0','LMIobjPosDef');
  lmi2 = lmi('A*Q+Q*A''+B*Y+Y''*B''+B1*B1'' < 0','LMIstabNegDef');
  lmiQ = lmi('Q > 0','LMIQposdef');
  LMIs = mergelmi(lmi1,lmi2,lmiQ);
  % – Define linear SDP solver and parameter options
  options = sdpsettings('Solver','sedumi') ;
  % – Solve SDP with SeDuMi
  Obj_function = trace(X);
  sol = solvesdp(LMIs,[],Obj_function,options); % – Solve SDP
  .
  .
  .
  (code for output, i. e. see tables 4.2 – 4.5)
end
```

REFERENCES

[1] D. S. Bernstein and W. M. Haddad, *LQG control with an $\mathcal{H}_\infty$ performance bound: A Riccati equation approach*, IEEE Transactions on Automatic Control, 34 (1989), pp. 293–305.

[2] S. P. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, vol. 15 of SIAM Studies in Applied Mathematics, SIAM, Philadelphia, 1994.

[3] C. W. J. Hol, C. W. Scherer, E. G. van der Meché, and O. H. Bosgra, *A nonlinear SDP approach to fixed–order controller synthesis and comparison with two other methods applied to an active suspension system*, European Journal of Control, 9 (2003).

[4] P. P. Khargonekar and M. A. Rotea, *Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control: A convex optimization approach*, IEEE Transactions on Automatic Control, 36 (1991), pp. 824–837.

[5] F. Leibfritz, *Static Output Feedback Design Problems*, Shaker Verlag, Aachen, Germany, ISBN 3-8265-4203-7, 1998.

[6] ———, *A LMI–based algorithm for designing suboptimal static $\mathcal{H}_2/\mathcal{H}_\infty$ output feedback controllers*, SIAM Journal on Control and Optimization, 39 (2001), pp. 1711–1735.

[7] F. Leibfritz, *COMPl$_e$ib: COnstrained Matrix–optimization Problem library – a collection of test examples for nonlinear semidefinite programs, control system design and related problems*, tech. report, University of Trier, Department of Mathematics, D–54286 Trier, Germany., 2003.

[8] F. Leibfritz and W. Lipinski, *Description of the benchmark examples in COMPl$_e$ib 1.0*, tech. report, University of Trier, Department of Mathematics, D–54286 Trier, Germany., 2003.

[9] F. Leibfritz and S. Volkwein, *Reduced order output feedback control design: A case study for pde systems using proper orthogonal decomposition and nonlinear semidefinite programming*, tech. report, Universität Trier, 2003. submitted.

[10] J. Löfberg, *YALMIP: Yet another LMI parser*, 2003. University of Linköpings; www.control.isy.liu.se.

[11] J. F. Sturm, *Using SeDuMi 1.02: A Matlab toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11 (1999), pp. 625–653.

[12] V. L. Syrmos, C. T. Abdallah, P. Dorato, and K. Grigoriadis, *Static output feedback – A survey*, Automatica, 33 (1997), pp. 125–137.

[13] K. C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3 – a Matlab software package for semidefinite programming*, Optimization Methods and Software, 11 (1999), pp. 545–581.

[14] K. Zhou and P. P. Khargonekar, *An algebraic Riccati equation approach to $\mathcal{H}_\infty$ optimization*, 11 (1988), pp. 85–91.

Table 4.2: *Results of SeDuMi on COMPl$_e$ib for solving the $\mathcal{H}_2$–SDP*

Ex	# var.	# const.	n_x	n_z	n_u	# iter.	CPU-sec	fail?	problems?
AC1	33	99	5	2	3	10	0.437	no	—
AC2	45	150	5	5	3	28	0.360	no	—
AC3	40	150	5	5	2	9	0.157	no	—

AC4	17	68	4	2	1	15	0.187	no	—
AC5	28	96	4	4	2	15	0.203	no	—
AC6	70	294	7	7	2	13	0.234	no	—
AC7	55	262	9	1	1	14	0.219	no	—
AC8	57	283	9	2	1	10	0.188	no	—
AC9	98	344	10	2	4	11	0.265	no	—
AC10	1665	9650	55	5	2	42	336.1	yes	numerical problems
AC11	40	150	5	5	2	16	0.234	no	—
AC12	23	57	4	1	3	9	0.140	no	—
AC13	896	4704	28	28	3	21	26.58	no	—
AC14	1006	5801	40	11	3	22	50.63	yes	numerical problems
AC15	39	132	4	6	2	11	0.172	no	—
AC16	39	132	4	6	2	11	0.156	no	—
AC17	24	96	4	4	1	9	0.140	no	—
AC18	90	425	10	5	2	4	0.141	yes	numerical problems
HE1	21	68	4	2	2	12	0.172	no	—
HE2	28	96	4	4	2	10	0.141	no	—
HE3	123	452	8	10	4	17	0.375	no	—
HE4	146	528	8	12	4	14	0.360	no	—
HE5	78	272	8	4	4	18	0.297	no	—
HE6	426	2096	20	16	4	20	3.625	no	—
HE7	426	2096	20	16	4	20	3.625	no	—
JE1	591	3244	30	8	3	26	13.50	yes	numerical problems
JE2	525	2646	21	21	3	31	15.81	yes	numerical problems
JE3	417	2241	24	9	3	40	11.36	yes	numerical problems
REA1	28	96	4	4	2	9	0.140	no	—
REA2	28	96	4	4	2	9	0.125	no	—
REA3	168	864	12	12	1	14	0.468	no	—
REA4	45	209	8	1	1	8	0.172	yes	infeasible problem
DIS1	104	384	8	8	4	10	0.234	no	—
DIS2	18	54	3	3	2	8	0.125	no	—
DIS3	66	216	6	6	4	10	0.188	no	—
DIS4	66	216	6	6	4	9	0.171	no	—
DIS5	24	81	4	3	2	13	0.188	no	—
TG1	130	600	10	10	2	17	0.453	no	—
AGS	180	864	12	12	2	12	0.438	no	—
WEC1	140	600	10	10	3	14	0.407	no	—
WEC2	140	600	10	10	3	15	0.438	no	—
WEC3	140	600	10	10	3	16	0.469	no	—
HF1	8648	51224	130	2	1	0	18.89	yes	problem size too large
BDT1	120	531	11	6	3	13	0.312	no	—
BDT2	3741	20844	82	4	4	21	1534	no	—
MFP	32	96	4	4	3	9	0.375	no	—
UWV	53	209	8	1	2	8	0.375	no	—
IH	528	1906	21	11	11	27	8.359	no	—
CSE1	328	1824	20	12	2	12	1.531	no	—
CSE2	2478	15664	60	32	2	14	309.9	no	—
EB1	68	344	10	2	1	14	0.688	no	—
EB2	68	344	10	2	1	15	0.672	no	—
EB3	68	344	10	2	1	22	0.969	no	—
EB4	233	1284	20	2	1	39	2.984	no	—
EB5	863	4964	40	2	1	25	33.38	yes	numerical problems
EB6	13043	77444	160	2	1	0	1.359	yes	problem size too large
PAS	21	86	5	1	1	43	1.813	no	—
TF1	52	219	7	4	2	11	0.593	no	—
TF2	52	219	7	4	2	11	0.578	no	—
TF3	52	219	7	4	2	11	0.578	no	—
PSM	57	242	7	5	2	9	0.453	no	—
TL	66304	393216	256	256	2	0	0	yes	problem size too large
CDP	7510	44176	120	4	2	0	13.83	yes	problem size too large
NN1	15	54	3	3	1	14	0.640	no	—
NN2	8	24	2	2	1	7	0.375	no	—
NN3	15	57	4	1	1	23	0.968	no	—
NN4	28	96	4	4	2	9	0.469	no	—
NN5	63	294	7	7	1	15	0.718	no	—

NN6	99	486	9	9	1	22	1.203	no	—
NN7	60	306	9	3	1	14	1.250	yes	numerical problems
NN8	18	54	3	3	2	8	0.437	no	—
NN9	40	131	5	4	3	9	0.469	no	—
NN10	63	228	8	2	3	20	0.907	no	—
NN11	190	873	16	3	3	36	3.484	no	—
NN12	54	216	6	6	2	10	0.500	no	—
NN13	39	153	6	3	2	12	0.563	no	—
NN14	39	153	6	3	2	12	0.562	no	—
NN15	22	67	3	4	2	13	0.609	no	—
NN16	78	272	8	4	4	22	1.016	no	—
NN17	15	43	3	2	2	7	0.407	no	—
NN18	507530	3040136	1006	2	1	0	0	yes	problem size too large
HF2D1	14428599	86488068	3796	3798	2	0	0	yes	problem size too large
HF2D2	14428599	86488068	3796	3798	2	0	0	yes	problem size too large
HF2D3	20173569	120942642	4489	4491	2	0	0	yes	problem size too large
HF2D4	4110753	24619954	2025	2027	2	0	0	yes	problem size too large
HF2D5	20173569	120942642	4489	4491	2	0	0	yes	problem size too large
HF2D6	4110753	24619954	2025	2027	2	0	0	yes	problem size too large
HF2D7	20173569	120942642	4489	4491	2	0	0	yes	problem size too large
HF2D8	4110753	24619954	2025	2027	2	0	0	yes	problem size too large
HF2D9	12134769	72732018	3481	3483	2	0	0	yes	problem size too large
HF2D10	53	194	5	7	2	11	0.547	no	—
HF2D11	53	194	5	7	2	9	0.469	no	—
HF2D12	53	194	5	7	2	12	0.578	no	—
HF2D13	53	194	5	7	2	10	0.515	no	—
HF2D14	53	194	5	7	2	11	0.546	no	—
HF2D15	53	194	5	7	2	14	0.640	no	—
HF2D16	53	194	5	7	2	12	0.594	no	—
HF2D17	53	194	5	7	2	12	0.594	yes	infeasible problem
HF2D18	53	194	5	7	2	10	0.500	no	—
CM1	236	1329	20	3	1	21	1.719	no	—
CM2	1896	11169	60	3	1	23	258.3	no	—
CM3	7386	43929	120	3	1	0	2.657	yes	problem size too large
CM4	29166	174249	240	3	1	0	0	yes	problem size too large
CM5	115926	694089	480	3	1	0	0	yes	problem size too large
CM6	462246	2770569	960	3	1	0	0	yes	problem size too large
TMD	39	153	6	3	2	30	1.500	no	—
FS	35	150	5	5	1	12	0.610	yes	infeasible problem
DLR1	78	344	10	2	2	13	0.703	no	—
DLR2	1803	9924	40	42	2	21	183.0	yes	infeasible problem
DLR3	1803	9924	40	42	2	21	183.5	yes	infeasible problem
ISS1	74796	440649	270	273	3	0	0	yes	problem size too large
ISS2	74796	440649	270	273	3	0	0	yes	problem size too large
CBM	61077	364708	348	2	1	0	0	yes	problem size too large
LAH	1230	7209	48	3	1	16	55.94	no	—
ROC1	66	283	9	2	2	21	1.140	no	—
ROC2	76	321	10	1	2	17	0.969	no	—
ROC3	176	726	11	11	4	11	0.906	no	—
ROC4	66	283	9	2	2	21	1.297	no	—
ROC5	52	179	7	2	3	20	1.141	no	—
ROC6	36	114	5	3	3	9	0.547	no	—
ROC7	31	114	5	3	2	9	0.562	no	—
ROC8	109	418	9	7	4	8	0.593	no	—
ROC9	54	193	6	5	3	7	0.484	no	—
ROC10	36	136	6	2	2	28	1.469	no	—

Table 4.3: *Results of SDPT3 on COMPl$_e$ib for solving the $\mathcal{H}_2$–SDP*

Ex	# var.	# const.	n_x	n_z	n_u	# iter.	CPU-sec	fail?	problems?
AC1	33	99	5	2	3	12	0.938	no	—
AC2	45	150	5	5	3	26	1.672	no	—
AC3	40	150	5	5	2	10	0.672	no	—
AC4	17	68	4	2	1	16	0.813	no	—

AC5	28	96	4	4	2	28	1.578	no	—
AC6	70	294	7	7	2	12	1.031	no	—
AC7	55	262	9	1	1	13	1.297	no	—
AC8	57	283	9	2	1	14	1.516	no	—
AC9	98	344	10	2	4	12	2.219	no	—
AC10	1665	9650	55	5	2	1	89.06	yes	numerical problems
AC11	40	150	5	5	2	12	0.797	no	—
AC12	23	57	4	1	3	8	0.469	no	—
AC13	896	4704	28	28	3	22	92.33	no	—
AC14	1006	5801	40	11	3	39	315.7	yes	numerical problems
AC15	39	132	4	6	2	12	0.734	no	—
AC16	39	132	4	6	2	12	0.735	no	—
AC17	24	96	4	4	1	10	0.578	no	—
AC18	90	425	10	5	2	16	2.718	no	—
HE1	21	68	4	2	2	12	0.734	no	—
HE2	28	96	4	4	2	9	0.562	no	—
HE3	123	452	8	10	4	22	3.203	no	—
HE4	146	528	8	12	4	14	2.266	no	—
HE5	78	272	8	4	4	21	2.672	no	—
HE6	426	2096	20	16	4	25	27.22	no	—
HE7	426	2096	20	16	4	25	27.23	no	—
JE1	591	3244	30	8	3	21	95.14	no	—
JE2	525	2646	21	21	3	44	60.61	yes	numerical problems
JE3	417	2241	24	9	3	34	54.30	no	—
REA1	28	96	4	4	2	9	0.563	no	—
REA2	28	96	4	4	2	9	0.547	no	—
REA3	168	864	12	12	1	13	2.657	no	—
REA4	45	209	8	1	1	11	1.140	yes	infeasible problem
DIS1	104	384	8	8	4	12	1.641	no	—
DIS2	18	54	3	3	2	10	0.562	no	—
DIS3	66	216	6	6	4	12	1.312	no	—
DIS4	66	216	6	6	4	12	1.282	no	—
DIS5	24	81	4	3	2	24	1.297	no	—
TG1	130	600	10	10	2	13	2.078	no	—
AGS	180	864	12	12	2	13	3.422	no	—
WEC1	140	600	10	10	3	16	2.703	no	—
WEC2	140	600	10	10	3	15	2.375	no	—
WEC3	140	600	10	10	3	16	2.532	no	—
HF1	8648	51224	130	2	1	0	1645	yes	problem size too large
BDT1	120	531	11	6	3	18	10.28	no	—
BDT2	3741	20844	82	4	4	15	2499	yes	numerical problems
MFP	32	96	4	4	3	12	3.250	no	—
UWV	53	209	8	1	2	10	3.218	no	—
IH	528	1906	21	11	11	38	63.67	no	—
CSE1	328	1824	20	12	2	14	19.20	no	—
CSE2	2478	15664	60	32	2	19	1025	no	—
EB1	68	344	10	2	1	12	5.719	no	—
EB2	68	344	10	2	1	16	7.500	no	—
EB3	68	344	10	2	1	18	8.406	no	—
EB4	233	1284	20	2	1	30	15.53	yes	numerical problems
EB5	863	4964	40	2	1	12	83.72	yes	infeasible problem
EB6	13043	77444	160	2	1	0	112.1	yes	problem size too large
PAS	21	86	5	1	1	17	4.406	no	—
TF1	52	219	7	4	2	14	3.968	no	—
TF2	52	219	7	4	2	14	4.000	no	—
TF3	52	219	7	4	2	14	3.985	no	—
PSM	57	242	7	5	2	12	3.891	no	—
CDP	7510	44176	120	4	2	0	2540	yes	problem size too large
NN1	15	54	3	3	1	12	2.547	no	—
NN2	8	24	2	2	1	7	1.266	no	—
NN3	15	57	4	1	1	20	4.187	no	—
NN4	28	96	4	4	2	8	2.094	no	—
NN5	63	294	7	7	1	16	5.328	no	—
NN6	99	486	9	9	1	59	25.80	yes	numerical problems
NN7	60	306	9	3	1	16	5.766	no	—

NN8	18	54	3	3	2	9	2.140	no	—
NN9	40	131	5	4	3	9	2.969	no	—
NN10	63	228	8	2	3	1	0.922	yes	numerical problems
NN11	190	873	16	3	3	19	18.66	no	—
NN12	54	216	6	6	2	10	3.766	no	—
NN13	39	153	6	3	2	11	3.985	no	—
NN14	39	153	6	3	2	11	4.375	no	—
NN15	22	67	3	4	2	13	2.719	no	—
NN16	78	272	8	4	4	1	1.125	yes	numerical problems
NN17	15	43	3	2	2	8	1.765	no	—
HF2D10	53	194	5	7	2	1	0.657	yes	numerical problems
HF2D11	53	194	5	7	2	1	0.625	yes	numerical problems
HF2D12	53	194	5	7	2	1	0.641	yes	numerical problems
HF2D13	53	194	5	7	2	1	0.641	yes	numerical problems
HF2D14	53	194	5	7	2	1	0.656	yes	numerical problems
HF2D15	53	194	5	7	2	1	0.656	yes	numerical problems
HF2D16	53	194	5	7	2	1	0.641	yes	numerical problems
HF2D17	53	194	5	7	2	1	0.640	yes	numerical problems
HF2D18	53	194	5	7	2	14	4.313	no	—
CM1	236	1329	20	3	1	25	35.05	no	—
CM2	1896	11169	60	3	1	34	3693	no	—
CM3	7386	43929	120	3	1	0	264.7	yes	problem size too large
TMD	39	153	6	3	2	28	8.547	yes	numerical problems
FS	35	150	5	5	1	37	9.844	yes	infeasible problem
DLR1	78	344	10	2	2	1	1.265	yes	numerical problems
DLR2	1803	9924	40	42	2	49	985.5	yes	infeasible problem
DLR3	1803	9924	40	42	2	49	990.6	yes	infeasible problem
LAH	1230	7209	48	3	1	16	549.6	no	—
ROC1	66	283	9	2	2	23	9.047	no	—
ROC2	76	321	10	1	2	16	7.953	no	—
ROC3	176	726	11	11	4	18	9.688	no	—
ROC4	66	283	9	2	2	30	11.58	yes	numerical problems
ROC5	52	179	7	2	3	23	6.672	no	—
ROC6	36	114	5	3	3	9	2.968	no	—
ROC7	31	114	5	3	2	13	3.641	no	—
ROC8	109	418	9	7	4	10	4.781	no	—
ROC9	54	193	6	5	3	9	3.703	no	—
ROC10	36	136	6	2	2	21	6.797	no	—

Table 4.4: *Results of SeDuMi on COMPl$_e$ib for solving the $\mathcal{H}_\infty$–SDP*

Ex	# var.	# const.	n_x	n_z	n_u	# iter.	CPU-sec	fail?	problems?
AC1	31	126	5	2	3	17	0.500	no	—
AC2	31	195	5	5	3	10	0.172	no	—
AC3	26	251	5	5	2	14	0.203	no	—
AC4	15	81	4	2	1	26	0.297	no	—
AC5	19	161	4	4	2	23	0.281	yes	$A+BYQ^{-1}$ not Hurwitz
AC6	43	491	7	7	2	27	0.453	no	—
AC7	55	278	9	1	1	24	0.343	no	—
AC8	55	523	9	2	1	23	0.390	no	—
AC9	96	585	10	2	4	24	0.453	no	—
AC10	1651	6995	55	5	2	49	393.5	yes	numerical problems
AC11	26	251	5	5	2	23	0.312	no	—
AC12	23	81	4	1	3	21	0.235	no	—
AC13	491	7841	28	28	3	22	12.77	yes	numerical problems
AC14	941	4626	40	11	3	25	39.09	yes	numerical problems
AC15	19	213	4	6	2	14	0.203	no	—
AC16	19	213	4	6	2	14	0.203	no	—
AC17	15	161	4	4	1	10	0.157	no	—
AC18	76	425	10	5	2	28	0.594	no	—
HE1	19	81	4	2	2	16	0.203	no	—
HE2	19	161	4	4	2	24	0.297	no	—
HE3	69	426	8	10	4	16	0.266	yes	$A+BYQ^{-1}$ not Hurwitz
HE4	69	849	8	12	4	19	0.438	yes	$A+BYQ^{-1}$ not Hurwitz

HE5	69	290	8	4	4	24	0.437	no	—
HE6	291	2165	20	16	4	26	2.250	no	—
HE7	291	2426	20	16	4	25	2.187	no	—
JE1	556	5525	30	8	3	53	30.23	yes	numerical problems
JE2	295	4411	21	21	3	43	15.84	yes	numerical problems
JE3	373	2098	24	9	3	37	8.531	no	—
REA1	19	161	4	4	2	20	0.234	no	—
REA2	19	161	4	4	2	20	0.266	no	—
REA3	91	1441	12	12	1	22	0.703	no	—
REA4	45	165	8	1	1	30	0.828	yes	numerical problems
DIS1	69	354	8	8	4	13	0.250	no	—
DIS2	13	91	3	3	2	16	0.203	no	—
DIS3	46	361	6	6	4	21	0.312	no	—
DIS4	46	361	6	6	4	20	0.297	no	—
DIS5	19	117	4	3	2	23	0.250	yes	$A+BYQ^{-1}$ not Hurwitz
TG1	76	1001	10	10	2	12	0.563	yes	numerical problems
AGS	103	1441	12	12	2	13	0.500	no	—
WEC1	86	1001	10	10	3	14	0.828	yes	numerical problems
WEC2	86	1001	10	10	3	16	0.937	no	—
WEC3	86	1001	10	10	3	15	0.812	no	—
HF1	8646	34590	130	2	1	0	17.73	yes	problem size too large
BDT1	100	446	11	6	3	22	0.485	no	—
BDT2	3732	14469	82	4	4	22	1640	no	—
MFP	23	161	4	4	3	10	0.140	no	—
UWV	53	186	8	1	2	32	0.453	no	—
IH	463	3251	21	11	11	40	9.063	no	—
CSE1	251	1490	20	12	2	23	1.515	no	—
CSE2	1951	12250	60	32	2	28	332.6	yes	numerical problems
EB1	66	297	10	2	1	13	0.219	no	—
EB2	66	297	10	2	1	17	0.281	no	—
EB3	66	297	10	2	1	16	0.265	no	—
EB4	231	977	20	2	1	20	0.937	yes	$A+BYQ^{-1}$ not Hurwitz
EB5	861	3537	40	2	1	22	25.47	yes	$A+BYQ^{-1}$ not Hurwitz
EB6	13041	52497	160	2	1	0	1.063	yes	problem size too large
PAS	21	90	5	1	1	30	0.329	yes	numerical problems
TF1	43	194	7	4	2	16	0.235	yes	$A+BYQ^{-1}$ not Hurwitz
TF2	43	194	7	4	2	16	0.234	yes	$A+BYQ^{-1}$ not Hurwitz
TF3	43	194	7	4	2	16	0.234	yes	$A+BYQ^{-1}$ not Hurwitz
PSM	43	246	7	5	2	12	0.172	no	—
TL	33409	655361	256	256	2	0	0	yes	problem size too large
CDP	7501	30277	120	4	2	0	12.94	yes	problem size too large
NN1	10	91	3	3	1	19	0.250	no	—
NN2	6	41	2	2	1	7	0.110	no	—
NN3	15	53	4	1	1	20	0.235	no	—
NN4	19	161	4	4	2	15	0.187	no	—
NN5	36	491	7	7	1	30	0.578	yes	$A+BYQ^{-1}$ not Hurwitz
NN6	55	811	9	9	1	35	0.922	yes	numerical problems
NN7	55	371	9	3	1	26	0.562	yes	numerical problems
NN8	13	91	3	3	2	14	0.188	no	—
NN9	31	147	5	4	3	23	0.266	no	—
NN10	61	234	8	2	3	12	0.203	no	—
NN11	185	741	16	3	3	31	1.657	no	—
NN12	34	361	6	6	2	13	0.219	yes	$A+BYQ^{-1}$ not Hurwitz
NN13	34	181	6	3	2	9	0.156	no	—
NN14	34	181	6	3	2	22	0.297	no	—
NN15	13	74	3	4	2	14	0.156	no	—
NN16	69	465	8	4	4	8	0.156	no	—
NN17	13	46	3	2	2	19	0.219	no	—
NN18	507528	2030118	1006	2	1	0	0	yes	problem size too large
HF2D1	7214299	144141717	3796	3798	2	0	0	yes	problem size too large
HF2D2	7214299	144141717	3796	3798	2	0	0	yes	problem size too large
HF2D3	10086784	201565083	4489	4491	2	0	0	yes	problem size too large
HF2D4	2055376	41030555	2025	2027	2	0	0	yes	problem size too large
HF2D5	10086784	201565083	4489	4491	2	0	0	yes	problem size too large
HF2D6	2055376	41030555	2025	2027	2	0	0	yes	problem size too large

HF2D7	10086784	201565083	4489	4491	2	0	0	yes	problem size too large
HF2D8	2055376	41030555	2025	2027	2	0	0	yes	problem size too large
HF2D9	6067384	121215387	3481	3483	2	0	0	yes	problem size too large
HF2D10	26	315	5	7	2	26	0.375	no	—
HF2D11	26	315	5	7	2	24	0.328	no	—
HF2D12	26	315	5	7	2	10	0.187	yes	infeasible problem
HF2D13	26	315	5	7	2	11	0.187	no	—
HF2D14	26	315	5	7	2	23	0.360	no	—
HF2D15	26	315	5	7	2	26	0.375	no	—
HF2D16	26	315	5	7	2	12	0.188	no	—
HF2D17	26	315	5	7	2	10	0.188	no	—
HF2D18	26	315	5	7	2	12	0.266	no	—
CM1	231	977	20	3	1	16	0.844	yes	$A+BYQ^{-1}$ not Hurwitz
CM2	1891	7697	60	3	1	19	223.4	yes	$A+BYQ^{-1}$ not Hurwitz
CM3	7381	29777	120	3	1	0	14.77	yes	problem size too large
CM4	29161	117137	240	3	1	0	0	yes	problem size too large
CM5	115921	464657	480	3	1	0	0	yes	problem size too large
CM6	462241	1850897	960	3	1	0	0	yes	problem size too large
TMD	34	137	6	3	2	20	0.234	no	—
FS	21	251	5	5	1	45	0.579	no	—
DLR1	76	297	10	2	2	24	0.391	no	—
DLR2	901	8490	40	42	2	72	102.9	no	—
DLR3	901	8490	40	42	2	72	102.9	no	—
ISS1	37396	368837	270	273	3	0	0	yes	problem size too large
ISS2	37396	368837	270	273	3	0	0	yes	problem size too large
CBM	61075	244306	348	2	1	0	0	yes	problem size too large
LAH	1225	5009	48	3	1	23	81.16	no	—
ROC1	64	251	9	2	2	14	0.235	no	—
ROC2	76	326	10	1	2	23	0.375	no	—
ROC3	111	1211	11	11	4	15	0.578	yes	$A+BYQ^{-1}$ not Hurwitz
ROC4	64	251	9	2	2	21	0.312	no	—
ROC5	50	194	7	2	3	20	0.297	no	—
ROC6	31	147	5	3	3	10	0.156	yes	$A+BYQ^{-1}$ not Hurwitz
ROC7	26	107	5	3	2	15	0.187	no	—
ROC8	82	371	9	7	4	10	0.188	yes	$A+BYQ^{-1}$ not Hurwitz
ROC9	40	181	6	5	3	9	0.141	yes	$A+BYQ^{-1}$ not Hurwitz
ROC10	34	137	6	2	2	25	0.297	no	—

Table 4.5: *Results of SDPT3 on COMPl$_e$ib for solving the $\mathcal{H}_\infty$–SDP*

Ex	# var.	# const.	n_x	n_z	n_u	# iter.	CPU-sec	fail?	problems?
AC1	31	126	5	2	3	16	1.329	yes	lack of progress
AC2	31	195	5	5	3	10	0.813	no	—
AC3	26	251	5	5	2	12	0.890	no	—
AC4	15	81	4	2	1	23	1.110	no	—
AC5	19	161	4	4	2	23	1.296	no	—
AC6	43	491	7	7	2	19	1.703	no	—
AC7	55	278	9	1	1	23	2.328	no	—
AC8	55	523	9	2	1	21	2.563	no	—
AC9	96	585	10	2	4	26	5.157	no	—
AC10	1651	6995	55	5	2	48	14070	no	—
AC11	26	251	5	5	2	17	13.13	no	—
AC12	23	81	4	1	3	13	10.30	yes	numerical problems
AC13	491	7841	28	28	3	19	128.0	no	—
AC14	941	4626	40	11	3	18	746.8	yes	numerical problems
AC15	19	213	4	6	2	13	9.297	no	—
AC16	19	213	4	6	2	13	9.281	no	—
AC17	15	161	4	4	1	8	5.390	no	—
AC18	76	425	10	5	2	20	31.75	no	—
HE1	19	81	4	2	2	14	9.985	no	—
HE2	19	161	4	4	2	16	11.42	no	—
HE3	69	426	8	10	4	13	19.59	yes	—
HE4	69	849	8	12	4	13	20.28	yes	—
HE5	69	290	8	4	4	19	28.08	no	—

HE6	291	2165	20	16	4	1	21.30	yes	numerical problems
HE7	291	2426	20	16	4	1	21.30	yes	numerical problems
JE1	556	5525	30	8	3	1	161.3	yes	numerical problems
JE2	295	4411	21	21	3	29	169.4	no	—
JE3	373	2098	24	9	3	31	239.3	no	—
REA1	19	161	4	4	2	13	9.266	no	—
REA2	19	161	4	4	2	13	9.281	no	—
REA3	91	1441	12	12	1	15	24.42	no	—
REA4	45	165	8	1	1	65	67.66	yes	numerical problems
DIS1	69	354	8	8	4	14	20.30	yes	numerical problems
DIS2	13	91	3	3	2	13	8.031	no	—
DIS3	46	361	6	6	4	17	19.08	no	—
DIS4	46	361	6	6	4	1	2.250	yes	numerical problems
DIS5	19	117	4	3	2	19	13.47	no	—
TG1	76	1001	10	10	2	17	25.09	yes	numerical problems
AGS	103	1441	12	12	2	13	20.95	no	—
WEC1	86	1001	10	10	3	1	4.094	yes	numerical problems
WEC2	86	1001	10	10	3	1	4.109	yes	numerical problems
WEC3	86	1001	10	10	3	1	4.110	yes	numerical problems
BDT1	100	446	11	6	3	12	23.81	yes	numerical problems
BDT2	3732	14469	82	4	4	19	4641	yes	numerical problems
MFP	23	161	4	4	3	10	8.000	no	—
UWV	53	186	8	1	2	26	28.92	no	—
IH	463	3251	21	11	11	34	173.1	no	—
CSE1	251	1490	20	12	2	14	64.30	yes	numerical problems
CSE2	1951	12250	60	32	2	15	1376	yes	numerical problems
EB1	66	297	10	2	1	16	21.11	no	—
EB2	66	297	10	2	1	20	26.98	no	—
EB3	66	297	10	2	1	19	25.67	no	—
EB4	231	977	20	2	1	29	108.2	yes	numerical problems
EB5	861	3537	40	2	1	417	6243	yes	numerical problems
PAS	21	90	5	1	1	36	27.09	no	—
TF1	43	194	7	4	2	16	16.08	no	—
TF2	43	194	7	4	2	16	16.08	no	—
TF3	43	194	7	4	2	16	16.08	no	—
PSM	43	246	7	5	2	13	12.06	no	—
NN1	10	91	3	3	1	13	7.313	no	—
NN2	6	41	2	2	1	8	3.968	no	—
NN3	15	53	4	1	1	20	12.58	no	—
NN4	19	161	4	4	2	13	9.391	no	—
NN5	36	491	7	7	1	16	13.86	no	—
NN6	55	811	9	9	1	18	18.94	no	—
NN7	55	371	9	3	1	14	15.77	no	—
NN8	13	91	3	3	2	15	9.203	no	—
NN9	31	147	5	4	3	25	20.91	no	—
NN10	61	234	8	2	3	1	3.250	yes	numerical problems
NN11	185	741	16	3	3	23	77.27	no	—
NN12	34	361	6	6	2	13	12.36	no	—
NN13	34	181	6	3	2	1	1.844	yes	numerical problems
NN14	34	181	6	3	2	1	1.859	yes	numerical problems
NN15	13	74	3	4	2	15	9.125	no	—
NN16	69	465	8	4	4	10	13.84	no	—
NN17	13	46	3	2	2	20	11.98	no	—
HF2D10	26	315	5	7	2	17	14.69	no	—
HF2D11	26	315	5	7	2	18	15.52	no	—
HF2D12	26	315	5	7	2	13	11.06	yes	numerical problems
HF2D13	26	315	5	7	2	11	9.687	no	—
HF2D14	26	315	5	7	2	23	19.67	no	—
HF2D15	26	315	5	7	2	21	18.00	no	—
HF2D16	26	315	5	7	2	13	11.08	yes	numerical problems
HF2D17	26	315	5	7	2	14	11.89	yes	numerical problems
HF2D18	26	315	5	7	2	9	7.750	yes	numerical problems
CM1	231	977	20	3	1	23	118.0	no	—
CM2	1891	7697	60	3	1	42	21250	no	—
TMD	34	137	6	3	2	19	17.81	no	—

FS	21	251	5	5	1	71	51.63	yes	numerical problems
DLR1	76	297	10	2	2	1	3.750	yes	numerical problems
DLR2	901	8490	40	42	2	36	614.8	yes	numerical problems
DLR3	901	8490	40	42	2	36	616.7	yes	numerical problems
LAH	1225	5009	48	3	1	30	5.267	no	—
ROC1	64	251	9	2	2	16	20.36	yes	numerical problems
ROC2	76	326	10	1	2	23	31.56	no	—
ROC3	111	1211	11	11	4	17	23.16	no	—
ROC4	64	251	9	2	2	21	27.03	no	—
ROC5	50	194	7	2	3	20	18.95	no	—
ROC6	31	147	5	3	3	12	10.53	no	—
ROC7	26	107	5	3	2	15	11.44	no	—
ROC8	82	371	9	7	4	10	14.31	no	—
ROC9	40	181	6	5	3	9	9.703	no	—
ROC10	34	137	6	2	2	29	24.89	no	—