Mixed LMI/Randomized Methods for Static Output Feedback Control Design

Denis Arzelier, Elena N. Gryazina, Dimitri Peaucelle and Boris T. Polyak

Abstract—This paper addresses the problem of stabilization of LTI systems via static output feedback (sof). The objective is not only to compute a stabilizing sof but rather to compute a discrete set of stabilizing sof. Two complementary mixed LMI/randomized algorithms are defined for this purpose. The main idea is to combine a particular relaxed LMI parametrization of stabilizing sof with high efficiency of Hit-and-Run method for generating random points in a given domain. Their respective relevance is analysed on several examples of the COMPl\_ib library which is intended to be the reference library for evaluating performance of reduced-order controller synthesis algorithms. Dual of the approach presented in [30], the general methodology proposed here appears to be much more effective. Finally, the paper additionally provides an extensive evaluation of the different relevant instances of the COMPl\_ib library.

I. INTRODUCTION

One of the most challenging open problems in control theory is the synthesis of static output-feedback (sof) controllers that meet desired performances and/or robustness specifications [33]. The static or reduced fixed-order dynamic output feedback problem is therefore always an active research area in the control literature. In the recent years, many attempts have been made to give efficient numerical procedures to solve related problems, [1], [21], [13], [11], [14], [15]. In [10], a numerical comparison was performed and classification into three categories (nonlinear programming, parametric optimization and convex programming approaches) was proposed. Even if these three classes may overlap, it gives a clear picture of the situation at that moment. Since then, new developments have been witnessed and the literature has been enriched by numerous contributions on the so-called nonlinear programming approach [22], [9], [8], [2] while pure convex programming methods or parametric optimization methods (which could be merged in a unique class) were scarcely still considered [26], [30]. The main reason mainly relies on the use of new powerful numerical tools based on nonsmooth optimization for the solution of static output feedback stabilization problems with closed-loop performance guarantees [9], [18]. The algorithms based on these techniques may be considered as the most numerically efficient ones at the moment as reported in different dedicated publications [3], [19], [17].

From a designer point of view, a possible limitation of these approaches is that the proposed algorithms rely on rather sophisticated theoretical tools (nonsmooth optimization) and are highly dedicated to the very purpose they have been designed for and without much room for the user to adapt the routine in a simple way. Additionally, if these last methods are rather efficient to find one solution, they seem far less competitive when dealing with the problem of the computation of sets of stabilizing sof. From a practical point of view, the designer is often more interested in the possibility of choosing a controller among many, guaranteeing different levels of heterogeneous performances in a given set than computing the optimal controller for a single performance which will often leads to poor alternative performances.

The objective of this paper is not to contest the present superiority of existing packages but rather to show that there is still room for alternative useful options on the specific issues mentioned above. In [27], a first step for designing sets of stabilizing static output feedbacks was proposed but the effective construction of samples in these ellipsoidal sets relied on usual deterministic local optimization algorithm. Recently, new randomized algorithms have been designed for the synthesis of feedback controllers in different contexts [16], [28], [12].

Our approach is based on Hit-and-Run (HR) method. It is a version of the Monte-Carlo method to generate points which are approximately uniformly distributed in a given set. The HR algorithm has been proposed by Turchin [34] and independently later by Smith [32]. It is aimed at approximately uniform generation of points in a body via random walks. Its properties have been studied in numerous works by Lovász and co-authors (see, e.g., the survey [7]).

Here, the idea is to propose a mixed LMI/randomized approach to tackle the problem of building sets of static output feedback stabilization. In [26], a new parametrization of all stabilizing sof gains based on the introduction of additional variables was defined. This parametrization leads to a convex LMI relaxation of the original problem which is dual of the one proposed later in [30]. Revisiting these earlier works, two complementary mixed LMI/randomized algorithms are defined and evaluated on the relevant part of the benchmarks of the COMPl\_ib library [20], [24], [25].
A. Notations

For conciseness reasons, some abbreviations are used. \(\text{sym}(A) = A + A'\), \([x]'BA = A'BA\).

For symmetric matrices, \(\preceq\) is the Löwner partial order \((A \succeq (\succeq) B\) if and only if \(A - B\) is positive (semi)definite \(\text{def}\) in the cone of positive (semi)definite matrices \(\mathbb{S}^+\) \((\mathbb{S}^+)\). \(\text{trace}(A)\) is the sum of diagonal elements of a (square) matrix. \(\Lambda(A)\) is the spectrum of the matrix \(A\) (the set of all eigenvalues of the matrix \(A\)). \(\mathbb{R}^{m \times n}\) is the linear space of rectangular \(m \times n\) matrices with real entries and equipped with the inner product defined as the usual inner product of the vectors representing the matrices \(< A, B > = \text{trace}(A'B)\).

B. Statement of the problem

Let the state-space model of the system be given by its minimal state-space realization:

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) \\
y(t) &= Cx(t)
\end{align*}
\]

where \(x \in \mathbb{R}^n\) is the state vector, \(u \in \mathbb{R}^m\) is the control vector and \(y \in \mathbb{R}^r\) is the output vector. All matrices are assumed to be of appropriate dimensions and it is assumed throughout the paper that \(\text{rank}(B) = m\) and \(\text{rank}(C) = r\).

The model (1) is stabilizable by static output-feedback (sof) if there exists a gain matrix \(K_{sf} \in \mathbb{R}^{m \times r}\) such that the closed-loop matrix \(A + BK_{sf}C\) is asymptotically stable \((A + BK_{sf}C)\) is a Hurwitz matrix). Moreover, the system is stabilizable by state-feedback \((sf)\) if there exists a gain matrix \(K_{sf} \in \mathbb{R}^{m \times r}\) such that the matrix \(A + BK_{sf}\) is Hurwitz. This last property is a special case of the former and corresponds to full state information output-feedback \((C = 1)\) and has found a tractable solution through convex optimization and LMI formalism [6]. Let us define the set of stabilizing state-feedback matrices:

\[
K_{sf} = \{ K_{sf} \in \mathbb{R}^{m \times n} : \Lambda(A + BK_{sf}) \subseteq \mathbb{C}^- \} \tag{2}
\]

and the set of stabilizing static output-feedback matrices:

\[
K_{sof} = \{ K_{sof} \in \mathbb{R}^{m \times r} : \Lambda(A + BK_{sof}C) \subseteq \mathbb{C}^- \} \tag{3}
\]

The problem considered in this paper is to build non trivial sets of stabilizing sof for (1).

**Problem 1: sof stabilization**

Given the model (1), build a non trivial subset \(K_{nsof} \subseteq K_{sof}\) of \(n_{sof} \geq 1\) instances.

**A. Parametrization of \(K_{sof}\)**

Two different mixed LMI/randomized algorithms may be defined. They both rely on the following parametrization of stabilizing sof matrices that has been first proposed in [26].

Let us define the following notation:

\[
M(P) = \begin{bmatrix}
A'P + PA & PB \\
B'P & 0
\end{bmatrix} \tag{4}
\]

A necessary and sufficient condition for the existence of a stabilizing sof for (1) is given in the following theorem.

**Theorem 1:** [26]

\[
\exists K_{sof} \in K_{sof} \text{ for the model (1) if and only if there exist a stabilizing sof matrix } K_{sf} \in K_{sf}, \text{ a matrix } P \in \mathbb{S}^{++} \text{ and matrices } F \in \mathbb{R}^{m \times m}, Z \in \mathbb{R}^{m \times r} \text{ solutions of the matrix inequality (5)}:
\]

\[
L(P, K_{sf}, Z, F) = M(P) + \text{sym} \left( \begin{bmatrix} K_{sf}' & 1 \\ ZC & F \end{bmatrix} \right) \prec 0 \tag{5}
\]

Moreover, \(K_{sof} = -F^{-1}Z \in K_{sof}\).

**Proof**

Note that the sof stabilizability of (1) \((\exists K_{sof} \in K_{sof})\) is equivalent to the existence of a matrix \(K_{sof} \in \mathbb{R}^{m \times r}\) and a matrix \(P \in \mathbb{S}^{++}\) solutions to the following Lyapunov inequality:

\[
(A + BK_{sof}C)'P + P(A + BK_{sof}C) = \begin{bmatrix} 1 & C'K_{sof} \\ F_1 & F_2 \end{bmatrix} M(P) \begin{bmatrix} 1 \\ K_{sof}C \end{bmatrix} \prec 0 \tag{6}
\]

Applying elimination lemma [31], this is also equivalent to the existence of matrices \(K_{sof} \in \mathbb{R}^{m \times r}, F_1 \in \mathbb{R}^{r \times m}, F_2 \in \mathbb{R}^{m \times m}\) and a matrix \(P \in \mathbb{S}^{++}\) solutions to the following matrix inequality:

\[
M(P) + \text{sym} \left( \begin{bmatrix} C'K_{sof}' & 1 \\ F_1 & F_2 \end{bmatrix} \right) \prec 0 \tag{7}
\]

The matrix \(F_2\) is always invertible since the block \((2, 2)\) reads \(F_2 + F_2' > 0\). Factorizing \(F_2\) in (7) leads to the result.

**II. CONSTRUCTION OF STABILIZING SOF SETS VIA A MIXED LMI/RANDOMIZED APPROACH**

**A. Parametrization of \(K_{sof}\)**

Let us define the following notation:

\[
M(P) = \begin{bmatrix}
A'P + PA & PB \\
B'P & 0
\end{bmatrix} \tag{4}
\]

for some \(\alpha > 0\) and \(K_{sf} = RX^{-1} \in K_{sf}\).

For a given \(K_{sf} \in K_{sf}\), the LMI convex set \(\mathcal{L}_{sof}^{K_{sf}}\) defined by:

\[
\mathcal{L}_{sof}^{K_{sf}} = \{(P, Z, F) \in \mathbb{S}^{++} \times \mathbb{R}^{m \times r} \times \mathbb{R}^{m \times m} : \exists K_{sf} \in K_{sf}, \text{ such that } L(P, K_{sf}, Z, F) < 0 \} \tag{9}
\]

is a convex parametrization that approximates the set of all stabilizing sof \(K_{sof}\). Note that this convex approximation may be empty \((\mathcal{L}_{sof}^{K_{sf}} = \emptyset)\) even if \(K_{sof} \neq \emptyset\) for a given
\( K_{stf} \in K_{stf} \). However, a complete parametrization of \( K_{sof} \) is obtained when \( K_{stf} \) covers the whole continuum of \( K_{sof} \).

\[
K_{sof} = \left\{ \bigcup_{i=1}^{\infty} K_{sof}^{i} \mid K_{stf}^{i} \in K_{stf} \right\}
\]

In fact, the problem of finding a stabilizing sof amounts to find a triplet composed of \((K_{stf}, K_{stf}, P) \in K_{stf} \times K_{stf} \times S^{+}\) verifying (5), meaning that a common Lyapunov certificate \( P \) has to be found for \( K_{stf} \) and \( K_{stf} \). This last interpretation is reminiscent of a necessary and sufficient condition of sof stabilizability proposed in [5].

**Remark 1:** Note that the parametrization (5) of \( K_{sof} \) has an equivalent one for \( K_{stf} \) since:

\[
L(P, K_{stf}, Z, F) = L(P, K_{sof}, H, F) = M(P) + \text{sym} \left( \begin{bmatrix} C' K_{sof}' & H' \\ -1 & F' \end{bmatrix} \right) < 0
\]

where \( K_{stf} = -F^{-T}H' \in K_{stf} \). Given \( K_{sof} \in K_{sof} \), one can easily get a convex parametrization approximating the set of all stabilizing sf. This seems of no consequence since the complete set \( K_{stf} \) may be parametrized by the LMI convex set defined in (8). Nevertheless, this alternate parametrization will appear useful in the sequel when looking for sets of sof.

For a given \( K_{sof} \in K_{sof} \), the LMI convex set \( K_{stf}^{K_{sof}} \) defined by:

\[
L_{stf}^{K_{sof}} = \left\{ (P, H, F) \in S^{+} \times \mathbb{R}^{m \times m} \times \mathbb{R}^{n \times m} : \exists K_{stf} \in K_{stf} \mid L(P, K_{sof}, H, F) < 0 \right\}
\]

is a convex parametrization that approximates the set of all stabilizing sf.

To make completely effective the condition of theorem 1, a precise description of the subset \( K_{sof}^{g} \subset K_{stf} \) defined by:

\[
K_{sof}^{g} = \left\{ K_{stf} \in \mathbb{R}^{m \times m} : \exists L_{stf}^{K_{sof}} \neq \emptyset \right\}
\]

would be necessary. The idea is therefore to find at least one element belonging to this set by generating a sufficient number of random samples in \( K_{stf} \) via Hit-and-Run techniques.

**B. Hit-and-Run (HR) for the stability set for matrices**

We apply HR for generating the sets of stabilizing state-feedback matrices \( K_{stf} \) (2) and static output-feedback matrices \( K_{sof} \) (3). The approach is the same for both cases.

We describe it for static output-feedback matrices (for state-feedback matrices take \( C = I, K \in \mathbb{R}^{m \times n} \)). Suppose matrices \( A, B, C \) are given and \( K \in \mathbb{R}^{m \times r} \) is a variable, \( K^0 \) belongs to the bounded set of stabilizing gains:

\[
K_{sof} = \{ K : A + BKC \text{ is Hurwitz} \}
\]

The structure of this set is analyzed in [16]. It can be nonconvex and can consist of many disjoint domains.

In every step of the HR algorithm we generate matrix \( D = Y/\|Y\|, Y = \text{randn}(m, r) \) which is uniformly distributed on the unit sphere in the space of matrices equipped with Frobenius norm. Matrix \( D \) is a random direction in the space of \( m \times r \)-matrices. We call boundary oracle an algorithm which provides \( L = \{ t \in R : tD^0 + tD \in K_{sof} \} \). We denote \( A + B(K^0 + tD)C = F + tG, F = A + B(K^0C, G = BDC \)

for a matrix \( K^0 \in K_{sof} \), then \( L = \{ t : F + tG \text{ is Hurwitz} \} \).

In the simplest case when \( K_{sof} \) is convex, this set is the interval \((-\infty, \infty) \) where \( \bar{\ell} = \sup\{ t : K^0 + tD \in K_{sof} \}, \bar{\ell} = \sup\{ t : K^0 - tD \in K_{sof} \} \). In more general situations boundary oracle provides all intersections of the straight line \( K^0 + tD, -\infty < t < +\infty \) with \( K_{sof} \). \( L \) consists of finite number of intervals, the algorithm for calculating their end points is presented in [16]. Section 4. However sometimes “brute force” approach is more simple. Introduce \( f(t) = \max \mathbb{R} \text{ e} \text{ig}(F + tG) \), then the end points of the intervals are solutions of the equation \( f(t) = 0 \) and can be found by use of standard 1D equation solvers (such as \text{command} \text{fsolve} \text{ in Matlab}).

**HR method works as follows.**

1. Find a starting point \( K^0 \in K_{sof} \), \( i = 0 \).
2. At the point \( K^i \in K_{sof} \) generate a random direction \( D_i \in \mathbb{R}^{m \times r} \) uniformly distributed on the unit sphere.
3. Apply boundary oracle procedure, i.e., define the set \( L_i = \{ t \in R : K^i + tD^i \in K_{sof} \} \).
4. Generate a point \( t_i \) uniformly distributed in \( L_i \) (we recall that \( L_i \) is, in general, a finite set of intervals), and compute a new point \( K^{i+1} = K^i + t_iD^i \).
5. Go to step 2 and increase \( i \).

The simplest theoretical result on the behavior of HR method states that if \( K_{sof} \) does not contain lower dimensional parts, then the method achieves the neighborhood of any point of \( K_{sof} \) with nonzero probability and asymptotically the distribution of points \( k_i \) tends to uniform one. The rate of convergence strongly depends on geometry of \( K_{sof} \) and its dimension.

**C. Two LMI/randomized algorithms for sof stabilization**

Two algorithms using complementary advantages of degrees of freedom offered by parametrization (5) and Hit-and-Run numerical efficiency are built in order to generate non trivial sets of stabilizing sof. The first algorithm uses Hit-and-Run only for generating a large subset \( K_{sof}^{n_{sof}} \) of \( K_{sof} \) that will serve for checking if \( L_{sof}^{K_{sof}} \) is empty.

**Algorithm 1:**

1. Compute a stabilizing sf \( K_{stf}^0 = RX^{-1} \in K_{stf} \) via the solution of the LMI problem (8).
2. From \( K_{stf}^0 \), generate a set \( K_{stf}^{n_{stf}} \subset K_{stf} \) which is the collection of \( n_{stf} \) samples of stabilizing sf matrices via HR;
3. Compute a set \( K_{sof}^{n_{sof}} \subset K_{sof} \) which is the collection of \( n_{sof} \) samples of stabilizing sof matrices:
   \[
   \forall K_{stf}^{i} \in K_{stf}^{n_{stf}}, \text{ if the LMI set } L_{sof}^{K_{stf}^{i}} \neq \emptyset, \text{ add the solution } K_i = -F_i^{-1}Z_i \text{ to the collection set } K_{sof}^{n_{sof}}.
   \]
Remark 2: (8) is not the only way to initialize algorithm 1. $K_{sf}^0$ may also be computed as the LMI solution of the $H_2$ or $H_{\infty}$ state-feedback problems (see [4] for more details).

Step 3 is performed by testing the realizability of the LMI set $\mathcal{L}_{sof}^n$ for each instance $K_{sf} \in \mathcal{K}_{sof}^n$ implying the solution of the homogeneous LMIs (7) with respect to the decision variables $(P, Z, F)$. To avoid numerical problems, this step must be done by solving the following semidefinite programming problem:

$$\min_{P, Z, F} \begin{array}{l}
\text{trace}(P) \\
\text{s.t.} \quad \text{trace}(P) > \alpha \\
M(P) + \text{sym}
\begin{bmatrix}
K_{sf}^t \\
-1
\end{bmatrix}
[ ZC F ] < 0
\end{array}$$

(15)

for some $\alpha > 0$.

The interest of running this algorithm is twofold. First, it will be a good way to evaluate the conservatism of the convex approximation of $\mathcal{K}_{sof}$ induced by parametrization (5). Indeed, the percentage $n_{sof}^1/n_{sof}$ may be defined as a quantitative measure of both the conservatism of the convex approximation as well as the difficulty to stabilize the plant via sof.

Remark 3: If $K$ is a stabilizing sof a priori known, then the set $\mathcal{L}_{sof}^n \neq \emptyset$ since the choice $Z = -FK$ will lead to the existence of $P \in \mathbb{S}^+$ and $F \in \mathbb{R}^{m \times m}$ such that:

$$M(P) + \text{sym}
\begin{bmatrix}
C^t K^t \\
-1
\end{bmatrix}
[ -FKC F ] = 0$$

$$M(P) + \text{sym}
\begin{bmatrix}
C^t K^t \\
-1
\end{bmatrix}
[ -F ] [ KC -1 ] < 0$$

(16)

Applying elimination lemma, this last condition is equivalent to the existence of a matrix $P \in \mathbb{S}^+$ such that:

$$(A + BKC)^t P + P(A + BKC) < 0$$

(17)

which is obviously true since $K \in \mathcal{K}_{sof}$.

A much more numerically efficient mixed LMI/randomized algorithm generating sets of stabilizing sof may be deduced from the previous one. It mainly avoids to run the computational burden of checking the emptiness of LMI set $\mathcal{L}_{sof}$ for every instance $K_{sf} \in \mathcal{K}_{sof}$ but starts a hit and run generation of $\mathcal{K}_{sof}^n$ as soon as an initial $K_{sof} \in \mathcal{K}_{sof}$ is found by LMI step. It proves to be much more efficient in practice to generate sets of stabilizing sof in almost every studied cases of the COMPI,ib library.

Algorithm 2: 1- Compute a stabilizing state-feedback $K_{sf}^0 = RX^{-1} \in \mathcal{K}_{sf}$ via the solution of the LMI problem (8) ;
2- From $K_{sf}^0$, generate a set $\mathcal{K}_{sf}^{n_{sof}} \subset \mathcal{K}_{sf}$ which is the collection of $n_{sof}$ samples of stabilizing sof matrices via H.R. ;
3- Find an initial stabilizing sof $K_{sof}^0 = F_0^{-1}Z_0$: Check every $K_{sf}^0 \in \mathcal{K}_{sf}^{n_{sof}}$ until $\mathcal{L}_{sof}^n \neq \emptyset$ ;
4- From $\mathcal{K}_{sof}$, compute a set $\mathcal{K}_{sof}^{n_{sof}} \subset \mathcal{K}_{sof}$ which is the collection of $n_{sof}$ samples of stabilizing sof matrices via H.R.

D. Results from the COMPI,ib library

The COMPI,ib library is composed of different LTI models (1) ranging from purely academic problems to more realistic industrial examples. The underlying systems that are already open-loop asymptotically stable have not been considered here for obvious reasons. With this last restriction, 53 different models mainly classified in six classes have been tested.

- Aerospace models: Aircraft models (AC), helicopter models (HE), jet engine models (JE)
- Reactor models (REA)
- Decentralized interconnected systems (DIS)
- Academic tests problems (NN)
- Various applications: Wind energy conversion model (WEC), binary distillation towers (BDT) and terrain following models (TF), string of high-speed vehicles (IH), strings (CSE), piezoelectric bimorph actuator (PAS),
- Second order models: A tuned mass damper (TMD), a flexible satellite (FS)
- 2D heat flow models (HF2D)

For precise details concerning each single example and benchmark, the interested reader may read [24] and [23]. The main results concerning the construction of stabilizing sof are presented in the tables I, II.

- First note that it has been possible to stabilize every model except AC10. In the first case, the HIFOO package which may be considered as one of the most effective tool for sof stabilization and for optimal $H_{\infty}$ sof control is not able either to stabilize it.
- A quick look at the reference [29] clearly shows that algorithms 1 and 2 gives far better results than this last reference. In [29], the proposed algorithm was unable to stabilize plants AC10, NN1, NN5, NN5, NN7, NN10 and NN12.
- Hard to stabilize examples may be identified as the ones for which less than 5% of the initializing state-feedback succeed in finding a stabilizing sof. AC9, AC13, AC14, AC18, HE3, JE3, BDT2, TF3, NN9, NN12, NN14 are such plants for which $n_{sof}$ obtained with algorithm 1 is rather low. This is mainly due to the bad conditioning of numerical operations (matrix inversion) and LMI optimization rather than a failure of the method in itself. This is confirmed by the reference [25] where similar failures of SDP solvers were already noticed for some of the previous examples (AC14, AC18, JE3) for the problems of $H_2$ or $H_{\infty}$ state feedback optimal control which are known to have a convex formulation. Moreover, the initialization $K_{sf} = KC$ where $K \in \mathcal{K}_{sof}$ is known a priori, does not perform well for all these examples, demonstrating that the plant matrices are poorly conditioned (see remark 3).

- It is not a surprise to note that $n_{sof}^1 \leq n_{sof}^2$ since if the initialization step succeeds $n_{sof} = 1000$ while $\max(n_{sof}^2) = 1000$. More surprising is the easiness to get $n_{sof}$ stabilizing sof for almost all examples and considering also that the complete run is really fast in
general. Size seems to be a limiting factor for LMI step but not for Hit-and-Run step.

Figure 1 shows the population stabilizing gains with the boundary for example AC7. It is interesting to note that the exact shape of the set of stabilizing sof is easily obtained with the mixed LMI/randomized algorithm 1 (see [20] for comparison). Similar results have been obtained for all two-parameters examples of the database COMPLib, labelled AC4, NN1, NN5 (see below), NN17 and HE1.

Fig. 1. Set of stabilizing sof generated with LMI/Randomized algorithm 2 for AC7 example

Figures 2 and 3 show a comparison of the populations respectively obtained via algorithms 1 and 2 for benchmark NN5.

Fig. 2. Set of stabilizing sof generated with LMI/Randomized algorithm 1 for NN5 example

Fig. 3. Set of stabilizing sof generated with LMI/Randomized algorithm 2 for NN5 example

III. CONCLUSIONS

In this article, two different mixed LMI/Hit-and-run algorithms have been proposed for the stabilization of LTI systems via static output feedback. Evaluation of these algorithms on the 53 benchmarks that are not open-loop stable shows that both may be considered as effective tools for the construction of non trivial sets of stabilizing sof. These examples exhibit relatively different structural features and different size of the state-space realization as well. Due to the relative limitations of SDP solvers in terms of computation time for large scale LTI systems, these approaches cannot be considered as competitive packages with respect to the last dedicated packages based on nonsmooth optimization [3], [17]. Nevertheless, we strongly believe that a promising new perspective is possible in the continuity of the seminal work of [12]. In particular, the results presented here should pave the way for the development of similar routines for performance optimization via static output feedback. Preliminary promising results on optimal $H_2$ and $H_\infty$ sof control are reported in the technical report [4].

APPENDIX

The following notations are used in tables I, II.
- OLS stands for Open-Loop Stable
- OLMS stands for Open-Loop Marginally Stable (a unique eigenvalue at 0 or multiple eigenvalue with 0 real part but scalar associated Jordan blocks)
- OLNS stands for Open-Loop Non Stable $\max(\text{real}(\text{eig}(A)))>0$
- $n_{sof}/n_{sf}$ stands for the portion of stabilizing sof found by the algorithm 1

<table>
<thead>
<tr>
<th>Ex.</th>
<th>$n_x$</th>
<th>$n_u$</th>
<th>$n_y$</th>
<th>OLS</th>
<th>$n_{sof}/n_{sf}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>OLMS</td>
<td>240/1000</td>
</tr>
<tr>
<td>AC2</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>OLMS</td>
<td>334/1000</td>
</tr>
<tr>
<td>AC5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>283/1000</td>
</tr>
<tr>
<td>AC9</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>OLNS</td>
<td>34/1000</td>
</tr>
<tr>
<td>AC10</td>
<td>55</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>*</td>
</tr>
<tr>
<td>AC11</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>996/1000</td>
</tr>
<tr>
<td>AC12</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>OLNS</td>
<td>997/1000</td>
</tr>
<tr>
<td>AC13</td>
<td>28</td>
<td>3</td>
<td>4</td>
<td>OLNS</td>
<td>39/1000</td>
</tr>
<tr>
<td>AC14</td>
<td>40</td>
<td>3</td>
<td>4</td>
<td>OLNS</td>
<td>13/1000</td>
</tr>
<tr>
<td>AC18</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>17/1000</td>
</tr>
<tr>
<td>HE1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>OLNS</td>
<td>93/1000</td>
</tr>
<tr>
<td>HE3</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>OLNS</td>
<td>12/1000</td>
</tr>
<tr>
<td>HE4</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>OLNS</td>
<td>1000/1000</td>
</tr>
<tr>
<td>HE5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>13/1000</td>
</tr>
<tr>
<td>HE6</td>
<td>20</td>
<td>4</td>
<td>6</td>
<td>OLNS</td>
<td>321/1000</td>
</tr>
<tr>
<td>HE7</td>
<td>20</td>
<td>4</td>
<td>6</td>
<td>OLNS</td>
<td>327/1000</td>
</tr>
<tr>
<td>DIS2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>842/1000</td>
</tr>
<tr>
<td>DIS4</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>OLNS</td>
<td>1000/1000</td>
</tr>
<tr>
<td>DIS5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>725/1000</td>
</tr>
<tr>
<td>JE2</td>
<td>21</td>
<td>3</td>
<td>3</td>
<td>OLMS</td>
<td>94/1000</td>
</tr>
<tr>
<td>JE3</td>
<td>24</td>
<td>3</td>
<td>6</td>
<td>OLNS</td>
<td>44/1000</td>
</tr>
<tr>
<td>REA1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>555/1000</td>
</tr>
<tr>
<td>REA2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>OLNS</td>
<td>965/1000</td>
</tr>
<tr>
<td>REA3</td>
<td>12</td>
<td>1</td>
<td>3</td>
<td>OLNS</td>
<td>965/1000</td>
</tr>
</tbody>
</table>

TABLE I
Numerical results for the COMPLib library
\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Ex. & n_x & n_u & n_y & OLS & n_{OLMS}/n_{OLNS} \\
\hline
WEL1 & 10 & 3 & 4 & OLNS & 177/1000 \\
BD72 & 52 & 4 & 4 & OLNS & 43/1000 \\
IH & 21 & 11 & 10 & OLNS & 63/1000 \\
CSE2 & 60 & 2 & 30 & OLNS & 10/10 \\
PAS & 5 & 1 & 3 & OLNS & 236/1000 \\
TF1 & 7 & 2 & 4 & OLNS & 817/1000 \\
TF2 & 7 & 2 & 3 & OLNS & 189/1000 \\
TF3 & 7 & 2 & 3 & OLNS & 6/1000 \\
NN1 & 3 & 1 & 2 & OLNS & 629/1000 \\
NN2 & 2 & 1 & 1 & OLNS & 1000/1000 \\
NN5 & 9 & 1 & 2 & OLNS & 84/1000 \\
NN6 & 9 & 1 & 4 & OLNS & 983/1000 \\
NN7 & 9 & 1 & 4 & OLNS & 620/1000 \\
NN9 & 5 & 3 & 2 & OLNS & 7/1000 \\
NN12 & 6 & 2 & 2 & OLNS & 28/1000 \\
NN13 & 6 & 2 & 2 & OLNS & 77/1000 \\
NN14 & 2 & 2 & 2 & OLNS & 44/1000 \\
NN15 & 3 & 2 & 2 & OLNS & 821/1000 \\
NN16 & 8 & 4 & 4 & OLNS & 61/1000 \\
NN17 & 3 & 2 & 1 & OLNS & 125/1000 \\
HF2D10 & 3 & 2 & 3 & OLNS & 997/1000 \\
HF2D11 & 5 & 2 & 3 & OLNS & 993/1000 \\
HF2D14 & 5 & 2 & 4 & OLNS & 1000/1000 \\
HF2D15 & 5 & 2 & 4 & OLNS & 1000/1000 \\
HF2D16 & 5 & 2 & 4 & OLNS & 996/1000 \\
HF2D17 & 5 & 2 & 4 & OLNS & 1000/1000 \\
HF2D18 & 5 & 2 & 2 & OLNS & 755/1000 \\
TMD & 6 & 2 & 4 & OLNS & 654/1000 \\
FS & 5 & 1 & 3 & OLNS & 977/1000 \\
\hline
\end{tabular}
\caption{Numerical results for the COMPLIB library}
\end{table}

\section*{References}


