Interior-point algorithms for semidefinite programming problems derived from the KYP lemma[*]

Lieven Vandenberghe[**], V. Ragu Balakrishnan[***], Ragnar Wallin[†], Anders Hansson[†], and Tae Roh[**]

Summary. We discuss fast implementations of primal-dual interior-point methods for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma, a class of problems that are widely encountered in control and signal processing applications. By exploiting problem structure we achieve a reduction of the complexity by several orders of magnitude compared to general-purpose semidefinite programming solvers.

1 Introduction

We discuss efficient implementations of interior-point methods for semidefinite programming problems (SDPs) of the form

$$
\begin{aligned}
\text{minimize} \quad & q^T x + \sum_{k=1}^{L} \mathbf{Tr}(Q_k P_k) \\
\text{subject to} \quad & \begin{bmatrix} A_k^T P_k + P_k A_k & P_k B_k \\ B_k^T P_k & 0 \end{bmatrix} + \sum_{i=1}^{p} x_i M_{ki} \succeq N_k, \quad k = 1, \ldots, L.
\end{aligned}
\tag{1}
$$

The optimization variables are $x \in \mathbf{R}^p$ and L matrices $P_k \in \mathbf{S}^{n_k}$, where $\mathbf{S}^n$ denotes the space of symmetric matrices of dimension $n \times n$. The problem data are $q \in \mathbf{R}^p$, $Q_k \in \mathbf{S}^{n_k}$, $A_k \in \mathbf{R}^{n_k \times n_k}$, $B_k \in \mathbf{R}^{n_k \times m_k}$, $M_{ki} \in \mathbf{S}^{n_k+m_k}$, and $N_k \in \mathbf{S}^{n_k+m_k}$. If $n_k = 0$, the kth constraint is interpreted as the linear matrix inequality (LMI) $\sum_{i=1}^{p} x_i M_{ki} \succeq N_k$. The SDPs we study can therefore

[**]Department of Electrical Engineering, University of California, Los Angeles. `vandenbe@ee.ucla.edu`, `roh@ee.ucla.edu`.

[***]School of Electrical and Computer Engineering, Purdue University. `ragu@ecn.purdue.edu`.

[†]Division of Automatic Control, Department of Electrical Engineering, Linköping University. `ragnarw@isy.liu.se`, `hansson@isy.liu.se`.

include arbitrary LMI constraints. At the end of this section we will list several assumptions made about the problem data. The most important of these assumptions is that (A_k, B_k) is controllable for $k = 1, \ldots, L$.

We refer to SDPs of the form (1) as KYP-SDPs, and to the constraints in the problem as KYP-LMIs, for the following reason. The Kalman-Yakubovich-Popov (KYP) lemma states that the semi-infinite frequency domain inequality

$$\begin{bmatrix} (j\omega I - A)^{-1}B \\ I \end{bmatrix}^* M \begin{bmatrix} (j\omega I - A)^{-1}B \\ I \end{bmatrix} \succ 0, \qquad \omega \in \mathbf{R}, \qquad (2)$$

where $A \in \mathbf{R}^{n \times n}$ does not have imaginary eigenvalues, holds if and only if the strict LMI

$$\begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + M \succ 0$$

with variable $P \in \mathbf{S}^n$ is feasible. Moreover, if (A, B) is controllable, then the nonstrict frequency domain inequality

$$\begin{bmatrix} (j\omega I - A)^{-1}B \\ I \end{bmatrix}^* M \begin{bmatrix} (j\omega I - A)^{-1}B \\ I \end{bmatrix} \succeq 0, \qquad \omega \in \mathbf{R}, \qquad (3)$$

holds if and only if the nonstrict LMI

$$\begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + M \succeq 0 \qquad (4)$$

is feasible (for a discussion of these results from a semidefinite programming duality perspective, see [BV03]). The KYP lemma forms the basis of some of the most important applications of SDPs in control; see, for example, [BEFB94, Ran96, HB99, OB99, MR97, Jön96, BW99, HV01].

The constraints in the KYP-SDP (1) have the same general form as (4), with M replaced with an affine function of the optimization variable x. If $Q_k = 0$, the KYP-SDP is therefore equivalent to the semi-infinite SDP

$$\begin{aligned} \text{min.} \quad & q^T x \\ \text{s.t.} \quad & \begin{bmatrix} (j\omega I - A_k)^{-1}B_k \\ I \end{bmatrix}^* (\mathcal{M}_k(x) - N_k) \begin{bmatrix} (j\omega I - A_k)^{-1}B_k \\ I \end{bmatrix} \succeq 0, \qquad (5) \\ & k = 1, \ldots, L, \end{aligned}$$

with variable x, where $\mathcal{M}_k(x) = \sum_{i=1}^{p} x_i M_{ki}$. More details and examples, including some applications in which $Q_k \neq 0$, are given in §2.

KYP-SDPs are difficult to solve using general-purpose SDP software packages [Stu01, TTT02, AHN$^+$97, FKN98, Bor02, BY02, GN95, WB96]. The difficulty stems from the very high number of optimization variables $(p + \sum_k n_k(n_k + 1)/2)$. Even moderate values of n_k (say, a few hundred) result in very large scale SDPs, with several 10,000 or 100,000 variables. This is unfortunate, because in many applications the variables P_k are of little

intrinsic interest. They are introduced as auxiliary variables, in order to convert the semi-infinite frequency-domain constraint (3) into a finite-dimensional LMI (4).

For this reason, several researchers have proposed alternatives to standard interior-point methods for solving KYP-SDPs. These methods include cutting-plane methods (such as the analytic center cutting-plane method) [Par00, KM01, KMJ01, KMJ03, Hac03], interior-point methods based on alternative barrier functions for the frequency-domain constraint [KM01], and interior-point methods combined with conjugate gradients [HV00, HV01, WHV03, GH03].

In this paper we examine the possibility of exploiting KYP-SDP problem structure to speed up standard primal-dual interior-point methods of the type used in state-of-the-art solvers like SeDuMi [Stu01, Stu02] and SDPT3 [TTT02]. Straightforward linear algebra techniques will allow us to implement the same interior-point methods at a cost that is orders of magnitude less than the cost of general-purpose implementations. More specifically, if $n_k = n$, $m_k = 1$ for $k = 1, \ldots, L$, and $p = O(n)$, then the cost per iteration of a general-purpose solver grows at least as n^6 as a function of n. Exploiting structure will allow us to reduce the complexity per iteration to n^3. Similar results have previously been obtained for dual barrier methods applied to special classes of KYP-SDPs, for example, KYP-SDPs derived for discrete-time FIR systems [AV02, GHNV03, Hac03]. The results in this paper can be viewed as an extension of these techniques to general KYP-SDPs, and to primal-dual interior-point methods.

Outline of the paper

The paper is organized as follows. In §2 we give an overview of applications, illustrating that KYP-SDPs are widely encountered in control. In §3 we present some basic facts about SDPs, SDP duality, and primal-dual interior-point methods for solving them. In §4 we explain in more detail the computations involved in solving KYP-SDPs using general-purpose software, and justify our estimate of an order n^6 complexity per iteration. We also describe a dual reformulation of the KYP-SDP which can be solved at a cost of roughly $O(n^4)$ per iteration, using general-purpose software. In §5 we describe techniques that exploit additional problem structure and result in a complexity of roughly $O(n^3)$ per iteration, for either the primal or dual formulation. The relation between the methods discussed in §4 and §5 is illustrated in Table 1. The results of some numerical experiments are described in §6. In §7 we discuss extensions of the techniques in §4 and §5, to problems with multiple constraints ($L > 1$), KYP-LMIs for multi-input systems ($m_k > 1$). Conclusions and some suggestions for future research are presented in §8.

The paper also includes several appendices. Appendix A provides additional background on semidefinite programming, and a detailed summary

	Primal formulation	Dual formulation
General-purpose	$O(n^6)$ (§4.1)	$O(n^4)$ (§4.2)
Special-purpose	$O(n^3)$ (§5)	$O(n^3)$ (§5)

Table 1. Relation between the methods in §4 and §5, and estimates of their complexity per iteration (for KYP-SDPs with $L = 1$, $n_1 = n$, $m_1 = 1$, $p = O(n)$).

of the primal-dual interior-point of [TTT98]. The other appendices contain proofs of results in the paper, and discussion of relaxed assumptions.

Assumptions

We will assume that the pairs (A_k, B_k), $k = 1, \ldots, L$, are controllable. Controllability implies that the linear mappings $\mathcal{K}_k : \mathbf{S}^{n_k} \to \mathbf{S}^{n_k + m_k}$, defined by

$$\mathcal{K}_k(P) = \begin{bmatrix} A_k^T P + P A_k & P B_k \\ B^T P & 0 \end{bmatrix},$$

have full rank (see §4.2). In addition, we assume that the matrices M_{ki} are such that the mapping

$$(P_1, P_2, \ldots, P_L, x) \quad \mapsto \quad \mathbf{diag}(\mathcal{K}_1(P_1) + \mathcal{M}_1(x), \ldots, \mathcal{K}_L(P_L) + \mathcal{M}_L(x)) \quad (6)$$

has full rank, where $\mathcal{M}_k(x) = \sum_{i=1}^p x_i M_{ki}$. In other words, the lefthand sides of the constraints in (1) are zero if and only if $P_k = 0$, $k = 1, \ldots, L$, and $x = 0$.

In fact these two assumptions can be relaxed. Controllability of (A_k, B_k) can be replaced with stabilizability, provided the range of Q_k is in the controllable subspace of (A_k, B_k); see Appendix D. Moreover, a problem for which (6) does not have full rank, can always be converted to an equivalent reduced order problem for which the full rank assumption holds; see Appendix E.

Throughout the paper we assume that the problem data and parameters are real. The generalization to complex data should be straightforward.

Notation

The space of symmetric $l \times l$ matrices is denoted $\mathbf{S}^l$. For $X \in \mathbf{S}^l$, $\mathbf{svec}(X)$ denotes the $l(l+1)/2$ vector containing the lower triangular elements of X:

$$\mathbf{svec}(X) = (x_{11}, x_{21}, \ldots, x_{l1}, x_{22}, \ldots, x_{l2}, \ldots, x_{l-1,l-1}, x_{l,l-1}, x_{ll}).$$

The space of symmetric block-diagonal matrices with block dimensions l_1, $\ldots$, l_L is denoted $\mathbf{S}^{l_1} \times \mathbf{S}^{l_2} \times \cdots \times \mathbf{S}^{l_L}$. If $X_1 \in \mathbf{S}^{l_1}$, $\ldots$, $X_L \in \mathbf{S}^{l_L}$, then $\mathbf{diag}(X_1, \ldots, X_L)$ denotes the block-diagonal matrix with X_1, $\ldots$, X_L as its diagonal blocks.

The space of Hermitian $l \times l$ matrices is denoted $\mathbf{H}^l$. For $A \in \mathbf{S}^l$ ($A \in \mathbf{H}^l$), $A \succeq 0$ means A is positive semidefinite, and the set of positive semidefinite symmetric (Hermitian) matrices of dimension l is denoted $\mathbf{S}^l_+$ ($\mathbf{H}^l_+$). Similarly, $A \succ 0$ means A is positive definite; $\mathbf{S}^l_{++}$ and $\mathbf{H}^l_{++}$ are the sets of positive definite symmetric, resp. Hermitian, matrices.

The Hadamard (componentwise) product $A \circ B$ of two matrices A, B of equal dimensions is defined by $(A \circ B)_{ij} = a_{ij} b_{ij}$. The ith unit vector is denoted e_i.

2 Applications of KYP-SDPs

While the form of the KYP-SDP (1) and the KYP-LMIs (2) and (3) may appear very special, they are widely encountered in control and signal processing. We give a representative list of applications along with a brief description.

2.1 Optimization problems with frequency-domain inequalities

As we already noted, a KYP-SDP (1) with an objective that does not depend on the variables P_k (*i.e.*, $Q_k = 0$), is equivalent to an optimization problem of the form (5), in which we minimize a linear cost function subject to frequency-domain inequalities (FDIs) of the form

$$H_k(\omega, x) \succeq 0 \qquad \omega \in \mathbf{R}. \tag{7}$$

Here $H_k : \mathbf{R} \times \mathbf{R}^p \to \mathbf{H}^m$ is defined as

$$H_k(\omega, x) = \begin{bmatrix} (j\omega I - A_k)^{-1} B_k \\ I \end{bmatrix}^* (\mathcal{M}_k(x) - N_k) \begin{bmatrix} (j\omega I - A_k)^{-1} B_k \\ I \end{bmatrix}.$$

Below we list a number of applications of problems with FDI constraints. It is important to note that in these applications, x is usually the design variable that we are interested in; the matrix P in the SDP formulation is an auxiliary variable, introduced to represent an infinite family of inequalities (7) as a single matrix inequality.

Linear system analysis and design

A well-known convex reformulation of the problem of linear time-invariant (LTI) controller design for LTI systems is via the Youla parametrization; see for example [BB91]. The underlying optimization problem here is to find x such that

$$T(s, x) = T_1(s) + T_2(s) \left(\sum_{i=1}^p x_i Q_i(s) \right) T_3(s),$$

satisfies a number of affine inequalities for $s = j\mathbf{R}$, where T_i and Q_i are given stable rational transfer function matrices [BB91, HB99, OB99]. These inequalities are readily expressed as FDIs of the form (7).

Digital filter design

This application involves the discrete-time version of the FDI (7). The standard digital filter design problem consists of designing

$$T(z, x) = \sum_{i=1}^{p} x_i T_i(z),$$

where $T_i : \mathbf{C} \to \mathbf{C}$ are given transfer functions, and x is to be determined so that $G(z, x)$ satisfies certain constraints. When $T_i(z) = z^{-i}$, we have a finite-impulse response (FIR) design problem. The constraints can be *magnitude constraints* of the form

$$|T(e^{j\theta}, x)| \leq U(e^{j\theta}), \qquad \theta \in [0, 2\pi), \tag{8}$$

or *phase constraints*

$$\angle T(e^{j\theta}, x) \leq R(e^{j\theta}), \qquad \theta \in [0, 2\pi), \tag{9}$$

Extensions where T_i are more general filter banks, and when T_i are matrix-valued transfer functions are immediate [BV98]. Other variations include optimal array pattern synthesis [WBZ$^+$03].

When $U(e^{j\theta})$ and $\tan(R(e^{j\theta}))$ are given (or can be approximated) as rational functions of $e^{j\theta}$, it is straightforward to express constraints (8) as the unit-circle counterparts of inequalities (7).

Other types of filter design problems include two-sided magnitude constraints

$$L(e^{j\theta}) \leq |T(e^{j\theta}, x)| \leq U(e^{j\theta}), \qquad \theta \in [0, 2\pi),$$

and no phase constraints. These constraints can be expressed as linear FDIs via a change of variables; see [WBV98, AV01, AV02, DLS02, GHNV00].

Robust control analysis using integral quadratic constraints

Robust control [ZDG96, GL95] deals with the analysis of and design for control system models that incorporate uncertainties explicitly in them. A sufficient condition for robust stability (*i.e.*, stability of the model irrespective of the uncertainties) can be unified in the framework of integral quadratic constraints (IQCs). The numerical problem underlying the IQC-based robust stability conditions is the following [MR97, Jön96, BW99]: Find $x \in \mathbf{R}^m$ such that for $\epsilon > 0$ and for all $\omega \in \mathbf{R}$,

$$\begin{bmatrix} T(j\omega) \\ I \end{bmatrix}^* \Pi(j\omega, x) \begin{bmatrix} T(j\omega) \\ I \end{bmatrix} \preceq -2\epsilon I, \tag{10}$$

where $T : \mathbf{C} \to \mathbf{C}^{m \times m}$ is a given real-rational function, and $\Pi : \mathbf{C} \times \mathbf{R}^p \to \mathbf{C}^{2m \times 2m}$ is a linear function of x for fixed ω, and is a real-rational function of ω for fixed x. Clearly, (10) corresponds to a special instance of (3).

Multiple FDIs of the form (10) result with more sophisticated (and better) sufficient conditions for robust stability with the IQC framework; see for example [FB97, IH98].

2.2 Linear-quadratic regulators

Consider the continuous-time dynamical system model

$$\dot{x} = Ax + Bu \tag{11}$$

with initial value $x(0) = x_0$, $A \in \mathbf{R}^{n \times n}$, $B \in \mathbf{R}^{n \times m}$, $x(t) \in \mathbf{R}^n$, and $u(t) \in \mathbf{R}^m$. Assume that (A, B) is controllable.

Riccati equations

Define the cost index

$$J = \int_0^\infty \begin{bmatrix} x \\ u \end{bmatrix}^T M \begin{bmatrix} x \\ u \end{bmatrix} dt \tag{12}$$

where

$$M = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \in \mathbf{S}^{n+m}$$

with $R \succ 0$. It is well-known (see, *e.g*, [Wil71]), that the infimal value of J with respect to $u(\cdot)$ subject to (11) and such that $\lim_{t \to \infty} x(t) = 0$ is, whenever it exists, given by $x_0^T P x_0$, where $P \in \mathbf{S}^n$ solves the KYP-SDP

$$\begin{aligned} \text{maximize} \quad & x_0^T P x_0 \\ \text{subject to} \quad & \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + M \succeq 0. \end{aligned} \tag{13}$$

The optimal $u(\cdot)$ is given as a state feedback $u(t) = -R^{-1}(PB + S)^T x(t)$. Here we see an application where the variable P is of intrinsic interest and appears in the objective. For this special case, of course, the optimal P can be found by solving an algebraic Riccati equation

$$A^T P + PA + Q - (PB + S)^T R^{-1}(PB + S) = 0,$$

and very efficient methods based on the real ordered Schur form of an associated matrix pencil are available. The computational complexity of these methods is in the order of n^3. However, numerical experience have shown that for certain ill-conditioned algebraic Riccati equations the KYP-SDP-formulation is not ill-conditioned. In some cases it can therefore be beneficial to solve algebraic Riccati equations via the SDP formulation. Moreover, slight generalizations of the above problem formulation require the solution of general KYP-SDPs. An example is given next.

Quadratic constraints

Define the cost indices

$$J_i = \int_0^\infty \begin{bmatrix} x \\ u \end{bmatrix}^T M_i \begin{bmatrix} x \\ u \end{bmatrix} dt, \qquad i = 0, \ldots, p \tag{14}$$

where $M_i \in \mathbf{S}^{n+m}$. Consider the constrained optimization problem

$$\begin{aligned}
\text{minimize} \quad & J_0 \\
\text{subject to} \quad & J_i \leq c_i, \quad i = 1, \ldots, p \\
& (11) \text{ and } \lim_{t \to \infty} x(t) = 0
\end{aligned} \tag{15}$$

with respect to $u(\cdot)$. The optimal value to this problem, whenever it exists, is given by $x_0^T P x_0$, where P solves the KYP-SDP

$$\begin{aligned}
\text{maximize} \quad & x_0^T P x_0 - c^T x \\
\text{subject to} \quad & \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + M_0 + \sum_{i=1}^p x_i M_i \succeq 0 \\
& x_i \geq 0, \quad i = 1, \ldots, p
\end{aligned} \tag{16}$$

(see [BEFB94, page 151]). The optimal $u(\cdot)$ is given as a state feedback $u(t) = -R^\dagger (PB + S)^T x(t)$, where

$$\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} = M_0 + \sum_{i=1}^p x_i M_i.$$

Here we see an application where both the variables P and x are of intrinsic interest. Moreover, we have multiple constraints, some of which only involve x.

2.3 Quadratic Lyapunov function search

Consider the continuous-time dynamical system model

$$\dot{x} = f(x, u, w, t), \qquad z = g(x, u, w, t), \qquad y = h(x, u, w, t) \tag{17}$$

where $x : \mathbf{R}_+ \to \mathbf{R}^n$, $u : \mathbf{R}_+ \to \mathbf{R}^{n_u}$, $w : \mathbf{R}_+ \to \mathbf{R}^{n_w}$, $z : \mathbf{R}_+ \to \mathbf{R}^{n_z}$, and $y : \mathbf{R}_+ \to \mathbf{R}^{n_y}$. x is referred to as the state, u is the control input, w is the exogenous input, z is the output of interest and y is the measured output. Models such as (17) are ubiquitous in engineering. (We have presented a continuous-time model only for convenience; the statements we make are equally applicable to discrete-time models.)

A powerful tool for the analysis of and design for model (17) proceeds via the use of quadratic Lyapunov functions. Suppose that for some $P \in \mathbf{S}^n_{++}$, the function $V(\psi) \triangleq \psi^T P \psi$ satisfies

$$\frac{d}{dt}V(x,t) < 0 \text{ along the trajectories of (17)},\tag{18}$$

then all trajectories of model (17) go to zero. For a number of special instances of system (17), the numerical search for Lyapunov functions results in feasibility problems with KYP-LMI constraints; see, for example, [BEFB94]. As an example, consider the system

$$\dot{x} = Ax + B_p p, \qquad q = C_q x + D_{qp} p, \qquad p = \Delta(t)q, \qquad \|\Delta(t)\| \leq 1, \quad (19)$$

where $\Delta : \mathbf{R}_+ \to \mathbf{R}^{m \times m}$. The existence of a quadratic Lyapunov function such that $dV(x,t)/dt < 0$ holds along the trajectories of (19) is equivalent to the following KYP-LMI:

$$P \succ 0, \qquad \begin{bmatrix} A^T P + PA + C_q^T C_q & PB_p + C_q^T D_{qp} \\ (PB_p + C_q^T D_{qp})^T & -(I - D_{qp}^T D_{qp}) \end{bmatrix} \prec 0.\tag{20}$$

If (A, C_q) is observable, the inequality $P \succ 0$ is implied by the second LMI, which is a (strict) KYP-LMI.

Variations of this basic idea underlie a very long list of recent results in systems and control theory that lead to KYP LMIs; the following list is by no means comprehensive:

- Robust stability of norm-bound systems with structured perturbations [Doy82, Saf82, BEFB94].
- Robust stability of parameter-dependent systems [FTD91, BEFB94].
- $\mathbf{H}_\infty$ controller synthesis [AG95].
- Gain-scheduled controller synthesis [Pac94, AA98, WB02].

3 Interior-point algorithms for semidefinite programming

3.1 Semidefinite programming

Let $\mathcal{V}$ be a finite-dimensional real vector space, with inner product $\langle u, v \rangle$. Let

$$\mathcal{A} : \mathcal{V} \to \mathbf{S}^{l_1} \times \mathbf{S}^{l_2} \times \cdots \times \mathbf{S}^{l_L}, \qquad \mathcal{B} : \mathcal{V} \to \mathbf{R}^r$$

be linear mappings, and suppose $c \in \mathcal{V}$, $D = \mathbf{diag}(D_1, D_2, \ldots, D_L) \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_L}$, and $d \in \mathbf{R}^r$ are given. The optimization problem

$$\begin{aligned} \text{minimize} \quad & \langle c, y \rangle \\ \text{subject to} \quad & \mathcal{A}(y) + D \preceq 0 \\ & \mathcal{B}(y) + d = 0 \end{aligned}\tag{21}$$

with variable $y \in \mathcal{V}$ is called a *semidefinite programming problem* (SDP). The dual SDP associated with (21) is defined as

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(DZ) + d^T z \\ \text{subject to} & \mathcal{A}^{\mathrm{adj}}(Z) + \mathcal{B}^{\mathrm{adj}}(z) + c = 0 \\ & Z \succeq 0, \end{array} \tag{22}$$

where

$$\mathcal{A}^{\mathrm{adj}} : \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_L} \to \mathcal{V}, \qquad \mathcal{B}^{\mathrm{adj}} : \mathbf{R}^r \to \mathcal{V}$$

denote the adjoints of $\mathcal{A}$ and $\mathcal{B}$. The variables in the dual problem are $Z \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_L}$, and $z \in \mathbf{R}^r$. We refer to Z as the dual variable (or multiplier) associated with the LMI constraint $\mathcal{A}(y) + D \preceq 0$, and to z as the multiplier associated with the equality constraint $\mathcal{B}(y) + d = 0$.

3.2 Interior-point algorithms

Primal-dual interior-point methods solve the pair of SDPs (21) and (22) simultaneously. At each iteration they solve a set of linear equations of the form

$$-W \Delta Z W + \mathcal{A}(\Delta y) = R \tag{23}$$

$$\mathcal{A}^{\mathrm{adj}}(\Delta Z) + \mathcal{B}^{\mathrm{adj}}(\Delta z) = r_{\mathrm{du}} \tag{24}$$

$$\mathcal{B}(\Delta y) = r_{\mathrm{pri}}, \tag{25}$$

to compute primal and dual search directions $\Delta y \in \mathcal{V}$, $\Delta Z \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_L}$, $\Delta z \in \mathbf{R}^r$. The scaling matrix W and the righthand side R in these equations are block-diagonal and symmetric ($W, R \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_L}$), and W is positive definite. The value of W, as well as the values of the righthand sides R, r_{du}, and r_{pri}, change at each iteration, and also depend on the particular algorithm used. We will call these equations *Newton equations* because they can be interpreted as a linearization of modified optimality conditions. We refer to appendix A, which gives a complete description of one particular primal-dual method, for more details. Primal or dual interior-point methods give rise to equations that have the same form as (23)–(25), with different definitions of W and the righthand sides. In this paper we make no assumptions about W, other than positive definiteness, so our results apply to primal and dual methods as well..

Since in practice the number of iterations is roughly independent of problem size (and of the order of 10–50), the overall cost of solving the SDP is roughly proportional to the cost of solving a given set of equations of the form (23)–(25).

3.3 General-purpose solvers

In a general-purpose implementation of an interior-point method it is assumed that $\mathcal{V}$ is the Euclidean vector space $\mathbf{R}^s$ of dimension $s = \dim \mathcal{V}$, and that $\mathcal{A}$ and $\mathcal{B}$ are given in the canonical form

$$\mathcal{A}(y) = \sum_{i=1}^{s} y_i F_i, \qquad \mathcal{B}(y) = By.$$

The matrices $F_i \in \mathbf{S}^{l_1} \times \mathbf{S}^{l_2} \times \cdots \times \mathbf{S}^{l_L}$ and $B \in \mathbf{R}^{r \times s}$ are stored in a sparse matrix format.

The equations (23)–(25) are solved by eliminating ΔZ from the first equation, and substituting $\Delta Z = W^{-1}(\mathcal{A}(\Delta y) - R)W^{-1}$ in the second equation. This yields a symmetric indefinite set of linear equations in Δy, Δz:

$$\mathcal{A}^{\mathrm{adj}}(W^{-1}\mathcal{A}(\Delta y)W^{-1}) + \mathcal{B}^{\mathrm{adj}}(\Delta z) = r_{\mathrm{du}} + \mathcal{A}^{\mathrm{adj}}(W^{-1}RW^{-1}) \quad (26)$$

$$\mathcal{B}(\Delta y) = r_{\mathrm{pri}}. \quad (27)$$

Using the canonical representation of $\mathcal{A}$ and $\mathcal{B}$, these equations can be written as

$$\begin{bmatrix} H & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_{\mathrm{du}} + g \\ r_{\mathrm{pri}} \end{bmatrix},$$

where

$$H_{ij} = \mathbf{Tr}(F_i W^{-1} F_j W^{-1}), \quad i,j = 1, \ldots, s$$
$$g_i = \mathbf{Tr}(F_i W^{-1} R W^{-1}), \quad i = 1, \ldots, s.$$

If the SDP has no equality constraints, the equations reduce to

$$\mathcal{A}^{\mathrm{adj}}(W^{-1}\mathcal{A}(\Delta y)W^{-1}) = r_{\mathrm{du}} + \mathcal{A}^{\mathrm{adj}}(W^{-1}RW^{-1}). \quad (28)$$

i.e.,

$$H\Delta y = r_{\mathrm{du}} + g.$$

The matrix H in this system is positive definite and almost always dense, so the cost of solving the equations is $(1/3)s^3$. This is only a lower bound on the actual cost per iteration, which also includes the cost of forming H. Even though sparsity in the matrices F_i helps, the cost of constructing H is often substantially higher than the cost of solving the equations.

4 General-purpose SDP solvers and KYP-SDPs

In this section we use the observations made in §3 to estimate the cost of solving KYP-SDPs with general-purpose interior-point software. For simplicity we assume that $L = 1$, $n_1 = n$, $m_1 = 1$, and consider the problem

$$\begin{aligned} \text{minimize} \quad & q^T x + \mathbf{Tr}(QP) \\ \text{subject to} \quad & \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^{p} x_i M_i \succeq N, \end{aligned} \quad (29)$$

where $A \in \mathbf{R}^{n \times n}$, $B \in \mathbf{R}^n$, with (A, B) controllable. The extension to problems with multiple inputs ($m > 1$) and multiple constraints ($L > 1$) is discussed in §7.

In §4.1 we first make precise our earlier claim that the cost of a general-purpose solver applied to (1) grows at least as n^6, if $p = O(n)$. In §4.2 we then describe a straightforward technique, based on semidefinite programming duality, that reduces the cost to order n^4.

4.1 Primal formulation

We can express the KYP-SDP (29) as

$$\text{minimize} \quad q^T x + \mathbf{Tr}(QP) \tag{30}$$
$$\text{subject to} \quad \mathcal{K}(P) + \mathcal{M}(x) \succeq N$$

where

$$\mathcal{K}(P) = \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix}, \qquad \mathcal{M}(x) = \sum_{i=1}^{p} x_i M_i. \tag{31}$$

This is in the general form (21), with $\mathcal{V} = \mathbf{S}^n \times \mathbf{R}^p$, and

$$y = (P, x), \qquad c = (Q, q), \qquad D = N, \qquad \mathcal{A}(P, x) = -\mathcal{K}(P) - \mathcal{M}(x).$$

The adjoint of $\mathcal{A}$ is $\mathcal{A}^{\mathrm{adj}}(Z) = -\big(\mathcal{K}^{\mathrm{adj}}(Z), \mathcal{M}^{\mathrm{adj}}(Z)\big)$, where

$$\mathcal{K}^{\mathrm{adj}}(Z) = \begin{bmatrix} A & B \end{bmatrix} Z \begin{bmatrix} I \\ 0 \end{bmatrix} + \begin{bmatrix} I & 0 \end{bmatrix} Z \begin{bmatrix} A^T \\ B^T \end{bmatrix}, \qquad \mathcal{M}^{\mathrm{adj}}(Z) = \begin{bmatrix} \mathbf{Tr}(M_1 Z) \\ \vdots \\ \mathbf{Tr}(M_p Z) \end{bmatrix}.$$

The dual problem of (32) is therefore

$$\text{maximize} \quad \mathbf{Tr}(NZ) \tag{32}$$
$$\text{subject to} \quad \mathcal{K}^{\mathrm{adj}}(Z) = Q, \quad \mathcal{M}^{\mathrm{adj}}(Z) = q$$
$$Z \succeq 0,$$

with variable $Z \in \mathbf{S}^{n+1}$.

A general-purpose primal-dual method applied to (30) generates iterates x, P, Z. At each iteration it solves a set of linear equations of the form (23)–(25) with variables Δx, ΔP, ΔZ:

$$W \Delta Z W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) = R_1 \tag{33}$$
$$\mathcal{K}^{\mathrm{adj}}(\Delta Z) = R_2 \tag{34}$$
$$\mathcal{M}^{\mathrm{adj}}(\Delta Z) = r, \tag{35}$$

for some positive definite W and righthand sides R_1, R_2, r. These equations are solved by eliminating ΔZ, reducing them to a smaller positive definite system (28). The reduced equations can be written in matrix-vector form as

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{12}^T & H_{22} \end{bmatrix} \begin{bmatrix} \mathbf{svec}(\Delta P) \\ \Delta x \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}. \tag{36}$$

The blocks of the coefficient matrix are defined by the identities

$$H_{11}\,\mathbf{svec}(\Delta P) = \mathbf{svec}\left(\mathcal{K}^{\mathrm{adj}}(W^{-1}\mathcal{K}(\Delta P)W^{-1})\right)$$
$$H_{12}\Delta x = \mathbf{svec}\left(\mathcal{K}^{\mathrm{adj}}(W^{-1}\mathcal{M}(\Delta x)W^{-1})\right)$$
$$H_{22}\Delta x = \mathcal{M}^{\mathrm{adj}}(W^{-1}\mathcal{M}(\Delta x)W^{-1}).$$

The exact expressions for the righthand sides r_1, r_2, and the positive definite scaling matrix W are not important for our present purposes and are omitted; see Appendix A for details.

The coefficient matrix in (36) is dense, so the cost of solving these equations is $(1/3)(n(n+1)/2 + p)^3 = O(n^6)$ operations if we assume that $p = O(n)$. This gives a lower bound for the cost of one iteration of a general-purpose interior-point solver applied to (29). The actual cost is higher since it includes the cost of assembling the matrices H_{11}, H_{12}, and H_{22}.

4.2 Dual formulation

A reformulation based on SDP duality allows us to solve KYP-SDPs more efficiently, at a cost of roughly $O(n^4)$ per iteration. The technique is well known for discrete-time KYP-SDPs with FIR matrices [GHNV03, DTS01, AV00, AV02], and was applied to general KYP-SDPs in [WHV03].

The reformulated dual

The assumption that (A, B) is controllable implies that that the mapping $\mathcal{K}$ defined in (31) has full rank, *i.e.*, $\mathcal{K}(P) = 0$ only if $P = 0$. To see this, we can take any stabilizing state feedback matrix K, and note that $\mathcal{K}(P) = 0$ implies

$$\begin{bmatrix} I \\ K \end{bmatrix}^T \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} \begin{bmatrix} I \\ K \end{bmatrix} = (A + BK)^T P + P(A + BK) = 0,$$

and hence $P = 0$. It follows that the nullspace of $\mathcal{K}^{\mathrm{adj}}$ (a linear mapping from $\mathbf{S}^{n+1}$ to $\mathbf{S}^n$) has dimension $n + 1$. Hence there exists a mapping $\mathcal{L} : \mathbf{R}^{n+1} \to \mathbf{S}^{n+1}$ that spans the nullspace of $\mathcal{K}^{\mathrm{adj}}$:

$$\mathcal{K}^{\mathrm{adj}}(Z) = 0 \Longleftrightarrow Z = \mathcal{L}(u) \text{ for some } u \in \mathbf{R}^{n+1}$$
$$S = \mathcal{K}(P) \text{ for some } P \in \mathbf{S}^n \Longleftrightarrow \mathcal{L}^{\mathrm{adj}}(S) = 0.$$

Some practical choices for $\mathcal{L}$ will be discussed later, but first we use this observation to derive an equivalent pair of primal and dual SDPs, with a smaller number of primal and dual variables.

The first equality in the dual SDP (32) is equivalent to saying that $Z = \mathcal{L}(u) - \hat{Z}$ for some u, where $\hat{Z}$ is any symmetric matrix that satisfies

$$\mathcal{K}^{\mathrm{adj}}(\hat{Z}) + Q = 0.$$

Substituting in the dual SDP (32), and dropping the constant term $\mathbf{Tr}(N\hat{Z})$ from the objective, we obtain an equivalent problem

$$
\begin{aligned}
\text{maximize} \quad & \mathcal{L}^{\text{adj}}(N)^T u \\
\text{subject to} \quad & \mathcal{L}(u) \succeq \hat{Z} \\
& \mathcal{M}^{\text{adj}}(\mathcal{L}(u)) = q + \mathcal{M}^{\text{adj}}(\hat{Z})
\end{aligned}
\tag{37}
$$

with variable $u \in \mathbf{R}^{n+1}$. This SDP has the form (21) with $\mathcal{V} = \mathbf{R}^{n+1}$, $y = u$,

$$
\mathcal{A}(u) = -\mathcal{L}(u), \qquad \mathcal{B}(u) = \mathcal{M}^{\text{adj}}(\mathcal{L}(u)),
$$

and $c = -\mathcal{L}^{\text{adj}}(N)$, $D = \hat{Z}$, $d = -q - \mathcal{M}^{\text{adj}}(\hat{Z})$.

The dual of problem (37) is

$$
\begin{aligned}
\text{minimize} \quad & (q + \mathcal{M}^{\text{adj}}(\hat{Z}))^T v - \mathbf{Tr}(\hat{Z}S) \\
\text{subject to} \quad & \mathcal{L}^{\text{adj}}(S) - \mathcal{L}^{\text{adj}}(\mathcal{M}(v)) + \mathcal{L}^{\text{adj}}(N) = 0 \\
& S \succeq 0,
\end{aligned}
\tag{38}
$$

with variables $v \in \mathbf{R}^p$ and $S \in \mathbf{S}^{n+1}$. Not surprisingly, the SDP (38) can be interpreted as a reformulation of the original primal problem (30). The first constraint in (38) is equivalent to

$$
S - \mathcal{M}(v) + N = \mathcal{K}(P)
\tag{39}
$$

for some P. Combined with $S \succeq 0$, this is equivalent to $\mathcal{K}(P) + \mathcal{M}(v) \succeq N$. Using (39) we can also express the objective function as

$$
\begin{aligned}
(q + \mathcal{M}^{\text{adj}}(\hat{Z}))^T v - \mathbf{Tr}(\hat{Z}S) &= q^T v + \mathbf{Tr}((\mathcal{M}(v) - S)\hat{Z}) \\
&= q^T v + \mathbf{Tr}(N\hat{Z}) - \mathbf{Tr}(PK^{\text{adj}}(\hat{Z})) \\
&= q^T v + \mathbf{Tr}(N\hat{Z}) + \mathbf{Tr}(PQ).
\end{aligned}
$$

Comparing this with (30), we see that the optimal v in (38) is equal the optimal x in (30). The relation (39) also allows us to recover the optimal P for (30) from the optimal solution (v, S) of (38).

In summary, the pair of primal and dual SDPs (37) and (38) is equivalent to the original SDPs (30) and (32); the optimal solutions for one pair of SDPs are easily obtained from the solutions of the other pair.

Newton equations for reformulated dual

A primal-dual method applied to (37) generates iterates u, v, S. At each iteration a set of linear equations of the form (26)–(27) is solved, which in this case reduce to

$$
\mathcal{L}^{\text{adj}}(W^{-1}\mathcal{L}(\Delta u)W^{-1}) + \mathcal{L}^{\text{adj}}(\mathcal{M}(\Delta v)) = R
\tag{40}
$$

$$
\mathcal{M}^{\text{adj}}(\mathcal{L}(\Delta v)) = r
\tag{41}
$$

with variables $\Delta u \in \mathbf{R}^{n+1}$, $\Delta v \in \mathbf{R}^p$. (Again, we omit the expressions for W, R, r. In particular, note that W is not the same matrix as in §4.1.) In matrix form,

$$\begin{bmatrix} H & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} R \\ r \end{bmatrix}, \tag{42}$$

where H and G are defined by the identities

$$H \Delta u = \mathcal{L}^{\mathrm{adj}}(W^{-1} \mathcal{L}(\Delta u) W^{-1}), \qquad G \Delta v = \mathcal{L}^{\mathrm{adj}}(\mathcal{M}(\Delta v)).$$

The number of variables in (42) is $p + n + 1$.

Computational cost

We now estimate the cost of assembling the coefficient matrix in (42), for a specific choice of $\mathcal{L}$. To simplify the notation, we assume that the Lyapunov operator $AX + XA^T$ is invertible. This assumption can be made without loss of generality: Since (A, B) is controllable by assumption, there exists a state feedback matrix K such that $A + BK$ is stable (or, more generally, $\lambda_i(A+BK)+\lambda_j(A+BK)^* \neq 0$, for $i, j = 1, \ldots, n$). By applying a congruence to both sides of the LMI constraint in (29) and noting that

$$\begin{bmatrix} I & K^T \\ 0 & I \end{bmatrix} \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ K & I \end{bmatrix} = \begin{bmatrix} (A + BK)^T P + P(A + BK) & PB \\ B^T P & 0 \end{bmatrix},$$

we can transform the SDP (29) to an equivalent KYP-SDP

$$\begin{aligned} \text{minimize} \quad & q^T x + \mathbf{Tr}(QP) \\ \text{subject to} \quad & \begin{bmatrix} (A + BK)^T P + P(A + BK)A & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^p x_i \tilde{M}_i \succeq \tilde{N}, \end{aligned}$$

where

$$\tilde{M}_i = \begin{bmatrix} I & K^T \\ 0 & I \end{bmatrix} M_i \begin{bmatrix} I & 0 \\ K & I \end{bmatrix}, \qquad \tilde{N} = \begin{bmatrix} I & K^T \\ 0 & I \end{bmatrix} N \begin{bmatrix} I & 0 \\ K & I \end{bmatrix}.$$

We will therefore assume that the matrix A in (29) is stable.

It is then easily verified that $\mathcal{K}^{\mathrm{adj}}(Z) = 0$ if and only if $Z = \mathcal{L}(u)$ for some u, with $\mathcal{L}$ defined as

$$\mathcal{L}(u) = \sum_{i=1}^{n+1} u_i F_i,$$

where

$$F_i = \begin{bmatrix} X_i & e_i \\ e_i^T & 0 \end{bmatrix}, \quad i = 1, \ldots, n, \qquad F_{n+1} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}. \tag{43}$$

and X_i, $i = 1, \ldots, n$, are the solutions of the Lyapunov equations

$$AX_i + X_i A^T + Be_i^T + e_i B^T = 0. \tag{44}$$

With this choice of $\mathcal{L}$, the coefficient matrices H and G in (42) can be expressed as

$$H_{ij} = \mathbf{Tr}(F_i W^{-1} F_j W^{-1})), \quad i, j = 1, \ldots, n+1, \tag{45}$$

$$G_{ij} = \mathbf{Tr}(F_i M_j), \quad i = 1, \ldots, n+1, \quad j = 1, \ldots, p. \tag{46}$$

To estimate the cost of this approach we assume that $p = O(n)$. The method requires a significant amount of preprocessing. In particular we have to compute the solutions X_i of $n+1$ Lyapunov equations, which has a total cost of $O(n^4)$. The matrix G does not change during the algorithm so it can be pre-computed, at a cost of order pn^3 if the matrices M_i and X_j are dense (i.e., $O(n^4)$ if we assume $p = O(n)$). In practice, as we have seen in §2, the matrices M_i are often sparse or low-rank, so the cost of computing G is usually much lower than $O(n^4)$.

At each iteration, we have to construct H and solve the equations (42). The cost of constructing H is $O(n^4)$. The cost of solving the equations is $O(n^3)$ if we assume $p = O(n)$. The total cost is therefore $O(n^4)$, and is dominated by the cost of pre-computing the basis matrices X_i, and the cost of forming H at each iteration.

5 Special-purpose implementation

We now turn to the question of exploiting additional problem structure in a special-purpose implementation. As should be clear from the previous section, the key to a fast implementation is to solve the linear equations that arise in each iteration fast. This can be done for either the primal or the dual formulation described in §4. We will see that these two approaches lead to methods that are almost identical, and have the same complexity.

5.1 Reduced Newton equations

In §4 we noted a large difference in complexity between solving the original KYP-SDP (29) and solving the reformulated dual problem (37). The difference is due to the different dimension and structure of the Newton equations in each iteration, and the way in which special-purpose codes handle those equations. In a custom implementation, the distinction between the two formulations disappears: the equations (33)–(35) that arise when solving the primal formulation can be solved as efficiently as the equations (40)–(41) that arise when solving the dual formulation.

Solving Newton equations via dual elimination

To show this, we describe an alternative method for solving (33)–(35). As in §4.2, let $\mathcal{L} : \mathbf{R}^{n+1} \to \mathbf{S}^{n+1}$ be a linear mapping that spans the nullspace of

$\mathcal{K}^{\mathrm{adj}}$. Let Z_0 be any symmetric matrix that satisfies $\mathcal{K}^{\mathrm{adj}}(Z_0) + R_2 = 0$. The equation (34) is equivalent to saying that

$$\Delta Z = \mathcal{L}(\Delta u) - Z_0$$

for some $\Delta u \in \mathbf{R}^{n+1}$. Substituting this expression in (33) and (35), we obtain

$$W\mathcal{L}(\Delta u)W + \mathcal{K}(\Delta P) + \mathcal{M}(\Delta x) = R_1 + WZ_0W$$
$$\mathcal{M}^{\mathrm{adj}}(\mathcal{L}(\Delta u)) = r + \mathcal{M}^{\mathrm{adj}}(Z_0).$$

Next we eliminate the variable ΔP, by applying $\mathcal{L}^{\mathrm{adj}}$ to both sides of the first equation, and using the fact that $\mathcal{L}^{\mathrm{adj}}(\mathcal{K}(\Delta P)) = 0$ for all ΔP:

$$\mathcal{L}^{\mathrm{adj}}(W\mathcal{L}(\Delta u)W) + \mathcal{L}^{\mathrm{adj}}(\mathcal{M}(\Delta x)) = \mathcal{L}^{\mathrm{adj}}(R_1 + WZ_0W) \qquad (47)$$
$$\mathcal{M}^{\mathrm{adj}}(\mathcal{L}(\Delta u)) = r + \mathcal{M}^{\mathrm{adj}}(Z_0). \qquad (48)$$

This is a set of $n + p + 1$ linear equations in $n + p + 1$ variables Δu, Δx. In matrix form,

$$\begin{bmatrix} H & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta x \end{bmatrix} = \begin{bmatrix} \mathcal{L}^{\mathrm{adj}}(R_1 + WZ_0W) \\ r + \mathcal{M}^{\mathrm{adj}}(Z_0) \end{bmatrix}, \qquad (49)$$

where H and G are defined by the identities

$$H\Delta u = \mathcal{L}^{\mathrm{adj}}(W\mathcal{L}(\Delta u)W), \qquad G\Delta x = \mathcal{L}^{\mathrm{adj}}(\mathcal{M}(\Delta x)).$$

Since $\mathcal{L}$ has full rank, the matrix H is nonsingular, so the equations (49) can be solved by first solving

$$G^T H^{-1} G\Delta x = G^T H^{-1}\mathcal{L}^{\mathrm{adj}}(R_1 + WZ_0W) - r - \mathcal{M}^{\mathrm{adj}}(Z_0)$$

to obtain Δx, and then computing Δu from

$$H\Delta u = \mathcal{L}^{\mathrm{adj}}(R_1 + WZ_0W) - G\Delta x.$$

After solving (49), we can compute ΔZ as $\Delta Z = \mathcal{L}(\Delta u)$. Given ΔZ and Δx, we find ΔP by solving

$$\mathcal{K}(\Delta P) = R_1 - W\Delta ZW - \mathcal{M}(\Delta x),$$

which is an overdetermined, but solvable set of linear equations.

   We will refer to (49) as the *reduced Newton equations*.

Computational cost

We now estimate the complexity of solving the reduced Newton equations. Note that (47)–(48) have exactly the same form as (40)–(41), with different values of W and the righthand sides. In particular, our discussion of the complexity of solving (40)–(41) also applies here.

We work out the details assuming the Lyapunov operator $AX + XA^T$ is invertible. If this is not the case, the equations (33)–(35) can be transformed into an equivalent set

$$TWT^T(T^{-T}\Delta ZT^{-1})TWT^T + T\mathcal{K}(\Delta P)T^T + T\mathcal{M}(\Delta x)T^T = TR_1T^T$$
$$\mathcal{K}^{\mathrm{adj}}(T^TT^{-T}\Delta ZT^{-1}T) = R_2$$
$$\mathcal{M}^{\mathrm{adj}}(T^TT^{-T}\Delta ZT^{-1}T) = r,$$

where

$$T = \begin{bmatrix} I & K^T \\ 0 & I \end{bmatrix}, \qquad T^{-1} = \begin{bmatrix} I & -K^T \\ 0 & I \end{bmatrix}$$

and K is a stabilizing state feedback matrix. Replacing ΔZ with a new variable

$$\Delta S = T^{-T}\Delta ZT^{-1},$$

gives

$$\tilde{W}\Delta S\tilde{W} + \begin{bmatrix} (A+BK)^T\Delta P + \Delta P(A+BK) & \Delta PB \\ B^T\Delta P & 0 \end{bmatrix} + \sum_{i=1}^{p} \Delta x_i \tilde{M}_i = \tilde{R}_1$$

$$\begin{bmatrix} A+BK \\ B \end{bmatrix}^T \Delta S \begin{bmatrix} I \\ 0 \end{bmatrix} + \begin{bmatrix} I \\ 0 \end{bmatrix}^T \Delta S \begin{bmatrix} A+BK \\ B \end{bmatrix} = R_2$$

$$\mathbf{Tr}(\tilde{M}_i \Delta S) = r_i, \quad i = 1, \ldots, p,$$

where $\tilde{W} = TWT^T$, $\tilde{M}_i = TM_iT^T$, $\tilde{R}_1 = TR_1T^T$. These equations have the same structure as the original equations (33)–(35), with A replaced by $A + BK$.

If we assume that $AX + XA^T$ is invertible, we can choose $\mathcal{L}$ as in §4.2: $\mathcal{L}(u) = \sum_{i=1}^{n+1} u_i F_i$ with the matrices F_i defined as in (43). If the matrices X_i are pre-computed (at a total cost of $O(n^4)$), then the cost of constructing the coefficient matrix H in (49) is $O(n^4)$. The cost of computing G is $O(pn^3)$ if we assume the matrices X_i are known, and we do not exploit any particular structure in M_j. The cost of solving the equations (49), given H and G, is $O(n^3)$ if we assume $p = O(n)$.

Comparison with dual method

The above analysis demonstrates that a custom implementation of a primal-dual method for solving the original KYP-SDP (29) can be as efficient as a primal-dual method applied to the reformulated dual (37). Both methods are based on eliminating dual variables, either in the original dual SDP, or in the Newton equations. In fact, if we use the same mapping $\mathcal{L}$ in both methods, the reduced linear equations are identical. However, a custom implementation offers three important advantages over a general-purpose primal-dual method applied to the reformulated dual.

- In a custom implementation we can avoid the need to compute and store the basis matrices F_i (or X_i), which are required in the dual formulation (see §5.2 for details). These matrices X_i are the solution of n Lyapunov equations of order n. For large n, they are expensive to compute and store.
- Additional problem structure can be exploited in a custom implementation. Two methods that achieve an $O(n^3)$ complexity per iteration are described in §5.2.
- In a custom implementation, we can make a different choice for the mapping $\mathcal{L}$, which is used to eliminate dual variables, in each iteration.
  For example, in §4.2 we pointed out that state feedback transformations preserve the KYP structure in the SDP, while in §5.1 we made a similar observation about the Newton equations. Of course, these two viewpoints are just different interpretations of the same property. We can first use state feedback to transform the SDP and then derive the Newton equations, or we can write down Newton equations for the original SDP and then apply a state feedback transformation. Both transformations result in the same equations. However the second viewpoint opens the possibility of selecting a different state-feedback matrix K in each iteration, in order to improve the numerical stability of the elimination step.

5.2 Fast construction of reduced Newton equations

We now examine two methods that allow us to construct the matrices H and G in (49) fast, in roughly $O(n^3)$ operations.

Diagonalizable A

Suppose A is stable (or equivalently, a state feedback transformation has been applied as described in §5.1, to obtain equivalent equations with a stable A). We make the same choice of $\mathcal{L}$ as in §5.1, *i.e.*, $\mathcal{L}(u) = \sum_{i=1}^{n+1} u_i F_i$, with F_i defined in (43).

In appendix B we derive the following expression for the matrix H in (49):

$$H = \begin{bmatrix} H_1 & 0 \\ 0 & 0 \end{bmatrix} + 2 \begin{bmatrix} W_{11} \\ W_{21} \end{bmatrix} \begin{bmatrix} H_2 & 0 \end{bmatrix} + 2 \begin{bmatrix} H_2^T \\ 0 \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} \end{bmatrix}$$
$$+ 2W_{22}W + 2 \begin{bmatrix} W_{12} \\ W_{22} \end{bmatrix} \begin{bmatrix} W_{21} & W_{22} \end{bmatrix} \tag{50}$$

where

$$(H_1)_{ij} = \mathbf{Tr}(X_i W_{11} X_j W_{11}), \qquad H_2 = \begin{bmatrix} X_1 W_{12} & X_2 W_{12} & \cdots & X_n W_{12} \end{bmatrix} \tag{51}$$

and

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$$

with $W_{11} \in \mathbf{S}^{n \times n}$. Similarly,

$$G = 2 \begin{bmatrix} G_1 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} M_{1,12} & M_{2,12} & \cdots & M_{p,12} \\ M_{1,22} & M_{2,22} & \cdots & M_{p,22} \end{bmatrix}, \tag{52}$$

where

$$G_1 = \begin{bmatrix} Y_1 B & Y_2 B & \cdots & Y_p B \end{bmatrix},$$

Y_j is the solution of

$$AY_j + Y_j A^T + M_{j,11} = 0,$$

and $M_{j,11} \in \mathbf{S}^n$, $M_{j,12} \in \mathbf{R}^n$ are the 1, 1- and 1, 2-blocks of M_j. Formulas (50) and (52) show that the key to constructing H and G fast (*i.e.*, faster than in $O(n^4)$ and $O(pn^3)$ operations, respectively), is to compute the matrices H_1, H_2, and G_1 fast.

A simple approach is based on the eigenvalue decomposition of A. Our assumption that (A, B) is controllable implies that is possible to find a linear state feedback matrix K so that $A+BK$ is stable and diagonalizable [KND85]. As mentioned in §5.1, we can transform the Newton equations into an equivalent set of equations in which the matrix A is replaced by $A + BK$. We can therefore assume without loss of generality that A is diagonalizable.

Let $A = V \mathbf{diag}(\lambda)V^{-1}$ be the eigenvalue decomposition of A, with $V \in \mathbf{C}^{n \times n}$ and $\lambda \in \mathbf{C}^n$. It can be shown that the matrices H_1 and H_2 defined in (51) can be expressed as

$$H_1 = 2\Re \left(\left(V^{-T}((\widetilde{\Sigma}\widetilde{W}_{11}) \circ (\widetilde{\Sigma}\widetilde{W}_{11})^T) + V^{-*}(\widetilde{W}_{11} \circ (\widetilde{\Sigma}\widetilde{W}_{11}\widetilde{\Sigma}^*)^T) \right) V^{-1} \right) \tag{53}$$

$$H_2 = -V(\widetilde{\Sigma}^* \mathbf{diag}(\widetilde{W}_{12}))\bar{V}^{-1} - V \mathbf{diag}(\widetilde{\Sigma}\widetilde{W}_{12})V^{-1} \tag{54}$$

where $\circ$ denotes Hadamard product, $\Sigma \in \mathbf{C}^{n \times n}$ is defined as

$$\Sigma_{ij} = \frac{1}{\lambda_i + \lambda_j^*}, \qquad i, j = 1, \ldots, n,$$

$\widetilde{\Sigma} = \Sigma \mathbf{diag}(V^{-1}B)^*$, $\widetilde{W}_{11} = V^*W_{11}V$, $\widetilde{W}_{12} = V^*W_{12}$, and $\bar{V}$ is the complex conjugate of V. The above formulas for H_1 and H_2 can be evaluated in $O(n^3)$ operations, and do not require pre-computing the basis matrices X_i. We refer to appendix C for a proof of the expressions (53) and (54).

There is a similar expression for G_1:

$$G_1 = V \begin{bmatrix} (\widetilde{M}_1 \circ \Sigma)V^*B & (\widetilde{M}_2 \circ \Sigma)V^*B & \cdots & (\widetilde{M}_p \circ \Sigma)V^*B \end{bmatrix}.$$

where $\widetilde{M}_j = V^{-1}M_{j,11}V^{-*}$. The cost of computing $\widetilde{M}_j$ can be reduced by exploiting low-rank structure in $M_{j,11}$. Given the matrices $\widetilde{M}_j$, G_1 can be computed in $O(n^2p)$ operations.

A in companion form

In this section we present an $O(n^3)$ method for solving the Newton equations when A is an $n \times n$ 'shift matrix' and $B = e_n$:

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{55}$$

KYP-SDPs of this form arise in a wide variety of LMI problems in robust control [Hen03]. The method is also useful for handling matrices A in companion form: in order to solve the Newton equations for a KYP-SDP with

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_1 & -a_2 & -a_3 & \cdots & -a_n \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

we can first apply a state feedback transformation with

$$K = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix},$$

as explained in §5.1, and then solve an equivalent set of equations in which A is replaced with the shift matrix $A + BK$.

With A and B defined as in (55), the equations (33)–(35) reduce to

$$W \Delta Z W + \begin{bmatrix} 0 & 0 \\ \Delta P & 0 \end{bmatrix} + \begin{bmatrix} 0 & \Delta P \\ 0 & 0 \end{bmatrix} + \sum_{i=1}^{p} \Delta x_i M_i = R_1 \tag{56}$$

$$\begin{bmatrix} 0 & I \end{bmatrix} \Delta Z \begin{bmatrix} I \\ 0 \end{bmatrix} + \begin{bmatrix} I & 0 \end{bmatrix} \Delta Z \begin{bmatrix} 0 \\ I \end{bmatrix} = R_2 \tag{57}$$

$$\mathbf{Tr}(M_i \Delta Z) = r_i, \quad i = 1, \ldots, p. \tag{58}$$

We can follow the method of §5.1, with $\mathcal{L} : \mathbf{R}^{n+1} \to \mathbf{S}^{n+1}$ defined as

$$\mathcal{L}(u) = \begin{bmatrix} u_1 & 0 & u_2 & 0 & u_3 & \cdots & 0 & u_{k+1} \\ 0 & -u_2 & 0 & -u_3 & 0 & \cdots & -u_{k+1} & 0 \\ u_2 & 0 & u_3 & 0 & u_4 & \cdots & 0 & u_{k+2} \\ 0 & -u_3 & 0 & -u_4 & 0 & \cdots & -u_{k+2} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ u_k & 0 & u_{k+1} & 0 & u_{k+2} & \cdots & 0 & u_{2k} \\ 0 & -u_{k+1} & 0 & -u_{k+2} & 0 & \cdots & -u_{2k} & 0 \\ u_{k+1} & 0 & u_{k+2} & 0 & u_{k+1} & \cdots & 0 & u_{2k+1} \end{bmatrix}, \qquad n = 2k$$

$$\mathcal{L}(u) = \begin{bmatrix} u_1 & 0 & u_2 & 0 & u_3 & \cdots & u_{k+1} & 0 \\ 0 & -u_2 & 0 & -u_3 & 0 & \cdots & 0 & -u_{k+2} \\ u_2 & 0 & u_3 & 0 & u_4 & \cdots & u_{k+2} & 0 \\ 0 & -u_3 & 0 & u_4 & 0 & \cdots & 0 & -u_{k+3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & -u_{k+1} & 0 & -u_{k+2} & 0 & \cdots & 0 & -u_{2k+1} \\ u_{k+1} & 0 & u_{k+2} & 0 & u_{k+3} & \cdots & u_{2k+1} & 0 \\ 0 & -u_{k+2} & 0 & -u_{k+3} & 0 & \cdots & 0 & -u_{2k+2} \end{bmatrix}, \qquad n = 2k+1.$$

In other words, the even anti-diagonals of $\mathcal{L}(u)$ are zero. The elements on the odd anti-diagonals have equal absolute values and alternating signs. The nonzero elements in the first row and column are given by u.

To obtain efficient formulas for the matrix H in (49), we represent $\mathcal{L}$ as $\mathcal{L}(u) = \sum_{i=1}^{n+1} u_i F_i$, where

$$(F_i)_{jk} = \begin{cases} (-1)^{j+1} & j + k = 2i \\ 0 & \text{otherwise.} \end{cases}$$

We also factor W as $W = \sum_{k=1}^{n+1} v_k v_k^T$ (for example, using the Cholesky factorization or the eigenvalue decomposition). The i, j-element of H is

$$H_{ij} = \mathbf{Tr}(F_i W F_j W) = \sum_{k=1}^{n+1} \sum_{l=1}^{n+1} (v_l^T F_i v_k)(v_k^T F_j v_l).$$

Next we note that for $v, w \in \mathbf{R}^{n+1}$,

$$v^T F_i w = \sum_{j+k=2i} (-1)^{j+1} v_j w_k$$

$$= \sum_{k=\max\{1, 2i-n-1\}}^{\min\{n+1, 2i-1\}} (-1)^{k+1} w_k v_{2i-k}$$

$$= (v * (Dw))_{2i-1},$$

where $D = \mathbf{diag}(1, -1, 1, \ldots)$, and $v * (Dw)$ denotes the convolution of the vectors v and Dw. Therefore,

$$(v_l^T F_1 v_k, v_l^T F_2 v_k, \ldots, v_l^T F_{n+1} v_k) = E(v_l * (Dv_k))$$

where

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \in \mathbf{R}^{(n+1)\times(2n+1)}.$$

Using this notation we can express H as

$$H = E \left(\sum_{k=1}^{n+1} \sum_{l=1}^{n+1} (v_l * (Dv_k))(v_k * (Dv_l))^T \right) E^T.$$

This expression can be evaluated in $O(n^3)$ operations using the discrete Fourier transform (DFT). Let $W_{\mathrm{DFT}} \in \mathbf{C}^{(2n+1) \times (n+1)}$ be the first $n+1$ columns of the DFT matrix of length $2n+1$, *i.e.*, $W_{\mathrm{DFT}}v$ is the zero padded $(2n+1)$-point DFT of a vector $v \in \mathbf{R}^{n+1}$. Let

$$V_k = W_{\mathrm{DFT}} v_k, \qquad \tilde{V}_k = W_{\mathrm{DFT}} Dv_k, \qquad k = 1, \ldots, n+1,$$

be the DFTs of the vectors v_k and Dv_k. Then

$$H = \frac{1}{(2n+1)^2} EW_{\mathrm{DFT}}^* \left(\sum_{k=1}^{n+1} \sum_{l=1}^{n+1} (V_l \circ \tilde{V}_k)(V_k \circ \tilde{V}_l)^* \right) W_{\mathrm{DFT}} E^T$$

$$= \frac{1}{(2n+1)^2} EW_{\mathrm{DFT}}^* \left(\left(\sum_{l=1}^{n+1} V_l \tilde{V}_l^* \right) \circ \left(\sum_{k=1}^{n+1} \tilde{V}_k V_k^* \right) \right) W_{\mathrm{DFT}} E^T.$$

The matrix in the middle is the Hadamard product of the $(2n+1) \times (2n+1)$ matrix $\sum_k V_k \tilde{V}_k^*$ with its conjugate transpose, so the cost of evaluating this expression is $O(n^3)$.

The matrix G in (49) has elements

$$G_{ij} = \mathbf{Tr}(F_i M_j), \quad i = 1, \ldots, n+1, \quad j = 1, \ldots, p,$$

and is easily computed in $O(n^2 p)$ operations, since only $O(n)$ elements of F_i are nonzero. For sparse or low-rank M_j the cost is even lower.

6 Numerical examples

In Table 2 we compare four algorithms, applied to randomly generated KYP-SDPs of the form (29), with dimensions $n = 25, 50, 100, 200$, and $p = n$. Each problem was constructed so it is strictly primal and dual feasible. (However the algorithms were started at infeasible starting points.) The execution times listed are the CPU times in seconds on a 2GHz Pentium IV PC with 1GB of memory. All times are averages over five randomly generated instances.

The first method, SeDuMi (primal), solves the SDP (29) using the general-purpose solver SeDuMi (version 1.05R5) [Stu01]. The second method, SDPT3 (primal), solves the same problem using the general-purpose solver SDPT3 (version 3.02) [TTT02]. Both solvers were called via the YALMIP interface [Löf02]. We skip the last problem ($n = 200$), due to excessive computation time and memory requirements. The numbers T_s are the CPU times needed to solve each problem, divided by the number of iterations.

The other methods, SeDuMi (dual) and SDPT3 (dual), solve the reformulated dual problem (37), for the choice of basis matrices described in §5.1. In

n	SeDuMi (primal)		SDPT3 (primal)		SeDuMi (dual)			SDPT3 (dual)		
	#itrs	T_s	#itrs	T_s	T_p	#itrs	T_s	T_p	#itrs	T_s
25	10	0.3	8	0.8	0.1	12	0.04	0.1	9	0.06
50	11	8.1	9	4.9	1.1	11	0.3	1.1	9	0.26
100	11	307.1	8	107.2	21.4	14	3.3	21.4	10	1.4
200					390.7	12	30.9	390.7	10	15.3

Table 2. Computational results for the general-purpose SDP solvers SeDuMi and SDPT3 applied to KYP-SDPs with dimensions $n = p = 25, \ldots, 200$, applied to the original problem ('Primal'), and to the reformulated dual SDP ('Dual'). T_p is the time in seconds required for preprocessing, and consists mainly of the cost of computing an explicit basis for the solution set of the dual equality constraints. T_s is the solution time per iteration, excluding preprocessing time.

addition to the number of iterations and the time per iteration T_s, we also give T_p, the preprocessing time required to compute the parameters in the reformulated dual problem (37). This preprocessing step is dominated by the cost of of solving the $n + 1$ Lyapunov equations (44).

The results confirm that the cost per iteration of a general-purpose method applied to the primal SDP (29) grows much more rapidly than the same method applied to the reformulated dual problem (37).

Table 3 shows the results for a second experiment with randomly generated KYP-SDPs of dimensions $n = 100, \ldots, 500$, and $p = 50$. Again, all values are averages over five problem instances. The data in the column KYP-IPM are for

n	KYP-IPM			SeDuMi (dual)			SDPT3 (dual)		
	T_p	#itrs	T_s	T_p	#itrs	T_s	T_p	#itrs	T_s
100	1.0	9	0.6	0.5	12	1.5	3.6	12	1.3
200	8.3	9	4.7	3.5	13	24.4	44.4	13	13.8
300	28.1	10	16.7	11.7	12	155.3	194.2	14	77.7
400	62.3	10	36.2	26.7	12	403.7			
500	122.0	10	70.3	51.9	12	1068.4			

Table 3. Results for KYP-SDPs of dimension $n = 100, \ldots, n = 500$, and $p = 50$. The first method is a customized Matlab implementation of a primal-dual method as described in §5, using the formulas (53) and (54). The second method is SeDuMi applied to the reformulated dual SDP (37), after first diagonalizing A. The third method solves the same reformulated dual SDP using SDPT3, without the diagonalization of A (except in the preprocessing).

a customized Matlab implementation of the primal-dual interior-point method of Tütüncü, Toh, and Todd [TTT98, TTT02], applied to the dual problem, and using the expressions (53) and (54) to compute the coefficient matrix of the reduced Newton equations. The preprocessing time for this method

includes the eigenvalue decomposition of A and the computation of the matrix G in the reduced system (42). The table shows that the preprocessing time and execution time per iteration grow almost exactly as n^3.

For the same set of problems, we also show the results for SeDuMi applied to the reformulated dual problem. To speed up the calculation in SeDuMi, we first transform the dual problem (37), by diagonalizing A. This corresponds to a simple change of variables, replacing Z with $V^{-1}ZV^{-*}$ and $\tilde{z}$ with $V^{-1}\tilde{z}$. We then eliminate the (1,1)-block in the dual variable as in the SeDuMi (dual) method of Table 2, which gives a problem of form (37), with complex data and variables. Since A is diagonal, the basis matrices X_i are quite sparse and easier to compute (at a cost of $O(n^3)$ total). Despite the resulting savings, it is clear from the table that the execution time per iteration grows roughly as n^4.

The third column (SDTP3 (dual)) gives the results for SDPT3 applied to the reformulated dual problem. Since the version of SDPT3 we used does not accept complex data, we only used diagonalization of A in the preprocessing step, to accelerate the solution of the Lyapunov equations (44). Results are reported for the first three problems only, due to insufficient memory. As for SeDuMi, the results show an $O(n^4)$-growth for the solution time per iteration.

7 Extensions

In this section we discuss some extensions of the techniques of §4 and §5 to the general problem (1).

7.1 Multiple constraints

Consider a KYP-SDP with multiple constraints,

$$
\begin{aligned}
\text{minimize} \quad & q^T x + \sum_{k=1}^{L}(Q_k P_k) \\
\text{subject to} \quad & \mathcal{K}_k(P_k) + \mathcal{M}_k(x) \succeq N_k, \quad k = 1, \ldots, L,
\end{aligned}
$$

where $\mathcal{K}_k : \mathbf{S}^{n_k} \to \mathbf{S}^{n_k+1}$ and $\mathcal{M}_k : \mathbf{R}^p \to \mathbf{S}^{n_k+1}$ are defined as

$$
\mathcal{K}_k(P_k) = \begin{bmatrix} A_k^T P_k + P_k A_k & P_k B_k \\ B_k^T P_k & 0 \end{bmatrix}, \qquad \mathcal{M}_k(x) = \sum_{i=1}^{p} x_i M_{ki}.
$$

We assume (A_k, B_k) is controllable for $k = 1, \ldots, L$. The Newton equations that need to be solved at each iteration take the form

$$
\begin{aligned}
W_k \Delta Z_k W_k + \mathcal{K}_k(\Delta P_k) + \mathcal{M}_k(\Delta x) &= R_{\text{pri},k}, \quad k = 1, \ldots, L \\
\mathcal{K}_k^{\text{adj}}(\Delta Z_k) &= R_{\text{du},k}, \quad k = 1, \ldots, L \\
\sum_{k=1}^{L} \mathcal{M}_k^{\text{adj}}(\Delta Z_k) &= r,
\end{aligned}
$$

with variables $\Delta P_k \in \mathbf{S}^{n_k}$, $\Delta x \in \mathbf{R}^p$, $\Delta Z_k \in \mathbf{S}^{n_k+1}$. The values of the positive definite matrix W_k and the righthand sides change at each iteration. As in the single-constraint case, we solve the Newton equations by eliminating some of the dual variables, and expressing ΔZ_k as

$$\Delta Z_k = \mathcal{L}_k(\Delta u_k) - \hat{Z}_k,$$

where $\mathcal{L}_k : \mathbf{R}^{n_k+1} \to \mathbf{S}^{n_k+1}$ parametrizes the nullspace of $\mathcal{K}_k^{\mathrm{adj}}$, and $\hat{Z}_k$ satisfies

$$\mathcal{K}_k^{\mathrm{adj}}(\hat{Z}_k) + R_{\mathrm{du},k} = 0.$$

We then apply $\mathcal{L}_k^{\mathrm{adj}}$ to both sides of the first group of equations and obtain

$$\mathcal{L}_k^{\mathrm{adj}}(W\mathcal{L}_k(\Delta u_k)W) + \mathcal{L}_k^{\mathrm{adj}}(\mathcal{M}_k(\Delta x)) = \mathcal{L}_k^{\mathrm{adj}}(R_{\mathrm{pri},k} + W_k\hat{Z}_kW_k),$$

for $k = 1, \ldots, L$, and

$$\sum_{k=1}^{L} \mathcal{M}_k^{\mathrm{adj}}(\mathcal{L}_k(\Delta u_k)) = r + \sum_{k=1}^{L} \mathcal{M}_k^{\mathrm{adj}}(\hat{Z}_k).$$

In matrix form,

$$
\begin{bmatrix}
H_1 & 0 & \cdots & 0 & G_1 \\
0 & H_2 & \cdots & 0 & G_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & H_L & G_L \\
G_1^T & G_2^T & \cdots & G_L^T & 0
\end{bmatrix}
\begin{bmatrix}
\Delta u_1 \\
\Delta u_2 \\
\vdots \\
\Delta u_L \\
\Delta x
\end{bmatrix}
=
\begin{bmatrix}
\mathcal{L}_1^{\mathrm{adj}}(R_{\mathrm{pri},1} + W_1\hat{Z}_1W_1) \\
\mathcal{L}_2^{\mathrm{adj}}(R_{\mathrm{pri},2} + W_2\hat{Z}_2W_2) \\
\vdots \\
\mathcal{L}_L^{\mathrm{adj}}(R_{\mathrm{pri},L} + W_L\hat{Z}_LW_L) \\
r + \sum_{k=1}^{L} \mathcal{M}_k^{\mathrm{adj}}(\hat{Z}_k)
\end{bmatrix}.
\tag{59}
$$

To solve these equations we first solve

$$\sum_{k=1}^{L} G_k^T H_k^{-1} G_k \Delta x = -r - \sum_{k=1}^{L} \left(\mathcal{M}_k^{\mathrm{adj}}(\hat{Z}_k) - G_k^T H_k^{-1} \mathcal{L}_k^{\mathrm{adj}}(R_k + W_k\hat{Z}_kW_k) \right) \tag{60}$$

for Δx, and then solve

$$H_k \Delta u_k = \mathcal{L}_k^{\mathrm{adj}}(R_{\mathrm{pri},k} + W_k\hat{Z}_kW_k) - G_k \Delta x, \quad k = 1, \ldots, L,$$

to determine Δu_k. As in the single-constraint case, the cost of this method is dominated by the cost of forming the coefficient matrices H_k and G_k, the coefficient matrix of (60), and the cost of solving this system. The matrices G_k can be pre-computed. Assuming $n_k = O(n)$ for $k = 1, \ldots, L$, the cost of forming H_k is $O(n^4)$, or $O(n^3)$ if we use one of the methods of §5. Assuming $p = O(n)$, the cost of forming and solving the equations (60) is $O(Ln^3)$. Overall, the cost increases linearly with L.

7.2 Multivariable systems

The extension to systems with multiple inputs $(m_k > 1)$ is also quite straight-forward, although the formulas get more involved. The Newton equations include constraints

$$\mathcal{K}_k^{\mathrm{adj}}(\Delta Z_k) = \begin{bmatrix} A_k \\ B_k \end{bmatrix}^T \Delta Z_k \begin{bmatrix} I \\ 0 \end{bmatrix} + \begin{bmatrix} I \\ 0 \end{bmatrix}^T \Delta Z_k \begin{bmatrix} A \\ B \end{bmatrix} = R_{\mathrm{pri},k}, \qquad k = 1, \ldots, L.$$

By representing ΔZ_k as

$$\Delta Z_k = \mathcal{L}_k(\Delta u_k) - \hat{Z}_k$$

where $\mathcal{L} : \mathbf{R}^{n_k m_k + m_k(m_k+1)/2} \rightarrow \mathbf{S}^{m_k+n_k}$, we can obtain reduced Newton equations of the form (59). If $m_k \ll n_k$ this offers a practical and efficient alternative to standard general-purpose methods.

8 Conclusion

We have described techniques for exploiting problem structure in interior-point methods for KYP-SDPs, a class of large-scale SDPs that are common in control applications. The method is very effective if the SDP includes one or more inequalities with large state space dimension, and a relatively small number of inputs. Preliminary numerical results illustrate that a special-purpose interior-point implementation based on these techniques can achieve a dramatic gain in efficiency compared with the best general-purpose solvers.

Several open questions remain.

- There is considerable freedom in choosing the mapping $\mathcal{L}$, used to eliminate a subset of the dual variables. The representation used in §5.1, for example, is parametrized by a state feedback matrix K. It is not clear how this choice affects the numerical stability of the method.
- The main idea in our approach is to use direct linear algebra techniques to solve the Newton equations in an interior-point method fast. This allows us to speed up the computation, without compromising the reliability and speed of convergence of a primal-dual interior-point method. It seems likely that other common classes of SDPs in control can benefit from similar techniques.

Acknowledgments

A Primal-dual interior-point method for semidefinite programming

In this appendix we review the definition and key properties of the semidefinite programming problem (SDP). We also describe a primal-dual method for solving SDPs.

A.1 Optimality conditions

We first state a few basic properties of the pair of primal and dual SDPs (21) and (22). We will express the (primal) SDP (21) as

$$
\begin{aligned}
\text{minimize} \quad & \langle c, y \rangle \\
\text{subject to} \quad & \mathcal{A}(y) + S + D = 0 \\
& \mathcal{B}(y) + d = 0 \\
& S \succeq 0,
\end{aligned}
\tag{61}
$$

where $S \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_L}$ is an additional variable.

The *duality gap* associated with primal feasible points y, S and a dual feasible Z is defined as

$$\mathbf{Tr}(SZ).$$

It is easily verified that

$$\mathbf{Tr}(SZ) = \langle c, y \rangle - \mathbf{Tr}(DZ) - d^T z$$

if y, S, Z, z are primal and dual feasible. In other words the duality gap is equal to the difference between the objective values.

If strong duality holds, then y, S, Z, z are optimal if and only if they are feasible, *i.e.*,

$$
S \succeq 0, \qquad \mathcal{A}(y) + S + D = 0, \qquad \mathcal{B}(y) + d = 0,
\tag{62}
$$

and

$$
Z \succeq 0, \qquad \mathcal{A}^{\mathrm{adj}}(Z) + \mathcal{B}^{\mathrm{adj}}(z) + c = 0,
\tag{63}
$$

and the duality gap is zero:

$$
SZ = 0.
\tag{64}
$$

The last condition is referred to as *complementary slackness*.

A.2 Algorithm

We briefly describe an infeasible primal-dual interior-point method for solving the pair of SDPs (61) and (22). Except for a few details, the method is the algorithm of [TTT98, TTT02], which has been implemented in the state-of-the-art SDP solver SDPT3.

We assume that the mapping $(\mathcal{A}, \mathcal{B})$ has full rank, *i.e.*, $\mathcal{A}(y) = 0$ and $\mathcal{B}(y) = 0$ imply $y = 0$. We define $m = l_1 + l_2 + \cdots + l_L$.

Outline

The algorithm starts at initial y, z, S, Z satisfying $S \succ 0$, $Z \succ 0$ (for example, $y = 0$, $z = 0$, $S = I$, $Z = I$). We repeat the following five steps.

1. *Evaluate stopping criteria.* Terminate if the following three conditions are satisfied:

$$\|\mathcal{A}(y) + S + D\| \le \epsilon_{\text{feas}} \max\{1, \|B\|\}$$
$$\|\mathcal{B}(y) + d\| \le \epsilon_{\text{feas}} \max\{1, \|d\|\}$$
$$\|\mathcal{A}^{\text{adj}}(Z) + \mathcal{B}^{\text{adj}}(z) + c\| \le \epsilon_{\text{feas}} \max\{1, \|c\|\}$$
$$\mathbf{Tr}(SZ) \le \max\{\epsilon_{\text{abs}}, -\epsilon_{\text{rel}}\langle c, y\rangle, \epsilon_{\text{rel}}(\mathbf{Tr}(DZ) + d^T z)\},$$

   where ϵ_{feas}, ϵ_{abs}, ϵ_{rel} are given positive tolerances, or if a specified maximum allowable number of iterations is reached. Otherwise go to step 2.
2. *Compute the scaling matrix R.* The scaling matrix is a block-diagonal matrix, and defines a congruence that jointly diagonalizes S^{-1} and Z:

$$R^T S^{-1} R = \mathbf{diag}(\lambda)^{-1}, \qquad R^T Z R = \mathbf{diag}(\lambda) \tag{65}$$

   where $\lambda \in \mathbf{R}^m_{++}$.
3. *Compute the affine scaling directions Δy^{a}, ΔS^{a}, ΔZ^{a}, Δz^{a}, by solving the set of linear equations*

$$\mathcal{H}(\Delta Z^{\text{a}} S + Z \Delta S^{\text{a}}) = -\mathbf{diag}(\lambda)^2 \tag{66}$$
$$\Delta S^{\text{a}} + \mathcal{A}(\Delta y^{\text{a}}) = -(\mathcal{A}(y) + S + D) \tag{67}$$
$$\mathcal{A}^{\text{adj}}(\Delta Z^{\text{a}}) + \mathcal{B}^{\text{adj}}(\Delta z^{\text{a}}) = -(\mathcal{A}^{\text{adj}}(Z) + \mathcal{B}^{\text{adj}}(z) + d) \tag{68}$$
$$\mathcal{B}(\Delta y^{\text{a}}) = -(\mathcal{B}(y) + d), \tag{69}$$

   where $\mathcal{H}$ is defined as

$$\mathcal{H}(X) = \frac{1}{2}(R^T X R^{-T} + R^{-1} X^T R).$$

4. *Compute the centering-corrector steps Δy^{c}, ΔZ^{c}, ΔS^{c}, by solving the set of linear equations*

$$\mathcal{H}(\Delta Z^{\text{c}} S + Z \Delta S^{\text{c}}) = \mu I - \mathcal{H}(\Delta Z^{\text{a}} \Delta S^{\text{a}}) \tag{70}$$
$$\Delta S^{\text{c}} + \mathcal{A}(\Delta y^{\text{c}}) = 0 \tag{71}$$
$$\mathcal{A}^{\text{adj}}(\Delta Z^{\text{c}}) + \mathcal{B}^{\text{adj}}(\Delta z^{\text{c}}) = 0 \tag{72}$$
$$\mathcal{B}(\Delta y^{\text{c}}) = 0. \tag{73}$$

   The coefficient μ is given by

$$\mu = \frac{\mathbf{Tr}(SZ)}{m} \left(\frac{\mathbf{Tr}((S + \alpha \Delta S^{\text{a}})(Z + \beta \Delta Z^{\text{a}}))}{\mathbf{Tr}(SZ)} \right)^\delta,$$

where

$$\alpha = \min\{1, \sup\{\alpha \mid S + \alpha \Delta S^{\mathrm{a}} \succeq 0\}\}$$
$$\beta = \min\{1, \sup\{\beta \mid Z + \beta \Delta Z^{\mathrm{a}} \succeq 0\}\}$$

and δ is an algorithm parameter. Typical values of δ are $\delta = 1, 2, 3$.

5. *Update the primal and dual variables* as

$$y := y + \alpha \Delta y, \qquad S := S + \alpha \Delta S, \qquad Z := Z + \beta \Delta Z, \qquad z := z + \beta \Delta z,$$

where $\Delta y = \Delta y^{\mathrm{a}} + \Delta y^{\mathrm{c}}$, $\Delta S = \Delta S^{\mathrm{a}} + \Delta S^{\mathrm{c}}$, $\Delta Z = \Delta Z^{\mathrm{a}} + \Delta Z^{\mathrm{c}}$, $\Delta z = \Delta z^{\mathrm{a}} + \Delta z^{\mathrm{c}}$, and

$$\alpha = \min\{1, 0.99 \sup\{\alpha \mid S + \alpha \Delta S \succeq 0\}\}$$
$$\beta = \min\{1, 0.99 \sup\{\beta \mid Z + \beta \Delta Z \succeq 0\}\}.$$

Go to step 1.

Discussion

Starting point

The method is called *infeasible* because it does not require feasible starting points. The initial values of y, z, S, Z must satisfy $S \succ 0$ and $Z \succ 0$, but do not have to satisfy the linear equations $\mathcal{A}(y) + S + D = 0$, $\mathcal{B}(y) + d = 0$, $\mathcal{A}^{\mathrm{adj}}(Z) + \mathcal{B}^{\mathrm{adj}}(z) + c = 0$.

The update rule in Step 5 ensures that $S \succ 0$, $Z \succ 0$ throughout the algorithm. If started at a feasible starting point, the iterates in the algorithm will remain feasible. This is easily verified from the definition of the search directions in Steps 3 and 4.

Termination

If we start at feasible points, the iterates satisfy $\mathcal{A}(y) + D + S = 0$, $\mathcal{B}(y) + d = 0$, $\mathcal{A}^{\mathrm{adj}}(Z) + \mathcal{B}^{\mathrm{adj}}(z) + c = 0$ throughout the algorithm, so the first three conditions are automatically satisfied. If we start at infeasible points, these conditions ensure that at termination the primal and dual residuals $\mathcal{A}(y) + D + S$, $\mathcal{B}(y) + d$, $\mathcal{A}^{\mathrm{adj}}(Z) + \mathcal{B}^{\mathrm{adj}}(z) + c$ are sufficiently small. A typical value for ϵ_{feas} is 10^{-8}.

At each iteration, we have $S \succ 0$, $Z \succ 0$, and hence $\mathbf{Tr}(SZ) > 0$. The third condition is therefore satisfied if one of the following conditions holds:

- $\mathbf{Tr}(SZ) \le \epsilon_{\mathrm{abs}}$
- $\langle c, y \rangle \le 0$ and $\mathbf{Tr}(SZ) \le \epsilon_{\mathrm{rel}} |\langle c, y \rangle|$
- $\mathbf{Tr}(DZ) + d^T z > 0$ and $\mathbf{Tr}(SZ) \le \epsilon_{\mathrm{rel}} (\mathbf{Tr}(DZ) + d^T z)$.

Assuming y, S, z, Z are feasible, the first of these conditions implies that the duality gap is less than ϵ_{abs}, and therefore

$$\langle c, y \rangle - p^{\star} \le \epsilon_{\text{abs}}, \qquad d^{\star} - \mathbf{Tr}(DZ) - d^T z \le \epsilon_{\text{abs}},$$

i.e., the absolute errors between the primal and dual objective values and their optimal values $p^{\star}$ and $d^{\star}$ are less than ϵ_{abs}.

If either the second or the third condition holds, then

$$\frac{\langle c, y \rangle - p^{\star}}{|p^{\star}|} \le \epsilon_{\text{rel}}, \qquad \frac{d^{\star} - \mathbf{Tr}(DZ) - d^T z}{|d^{\star}|} \le \epsilon_{\text{rel}},$$

i.e., we have determined the optimal values with a relative accuracy of at least ϵ_{abs}. Typical values of ϵ_{abs}, ϵ_{rel} are $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = 10^{-8}$.

Scaling matrix

The scaling matrix R is efficiently computed as follows. We first compute the Cholesky factorization of S and Z:

$$S = L_1 L_1^T, \qquad Z = L_2 L_2^T,$$

where L_1 and L_2 are block-diagonal with lower-triangular diagonal blocks of dimensions $m_1, \ldots, m_L$. Next, we compute the SVD of $L_2^T L_1$:

$$L_2^T L_1 = U \, \mathbf{diag}(\lambda) V^T,$$

where U and V are block-diagonal with block dimensions $l_1, \ldots, l_L$, $U^T U = I$, $V^T V = I$, and $\mathbf{diag}(\lambda)$ is a positive diagonal matrix of size $m \times m$. Finally, we form

$$R = L_1 V \, \mathbf{diag}(\lambda)^{-1/2}.$$

It is easily verified that $R^T S^{-1} R = \mathbf{diag}(\lambda)^{-1}$ and $R^T Z R = \mathbf{diag}(\lambda)$.

Search directions

The definition of $\mathcal{H}$ and the definition of the affine scaling and centering-corrector directions may be justified as follows. The *central path* for the pair of primal and dual SDPs (61) and (22) is defined as the set of points $(y(\mu), S(\mu), Z(\mu))$ that satisfy $S(\mu) \succ 0$, $Z(\mu) \succ 0$, and

$$\begin{aligned} \mathcal{A}(y(\mu)) + S(\mu) + D = 0, \qquad \mathcal{B}(y(\mu)) + d = 0 \\ \mathcal{A}^{\text{adj}}(Z(\mu)) + \mathcal{B}^{\text{adj}}(z(\mu)) + c = 0 \\ Z(\mu) S(\mu) = \mu I, \end{aligned} \qquad (74)$$

for some $\mu > 0$. In the limit for $\mu \to 0$, these equations reduce to the optimality conditions (62)–(64). Central points with parameter μ have duality

gap $\mathbf{Tr}(S(\mu)Z(\mu)) = m\mu$. Most interior-point methods can be interpreted as damped Newton methods for solving a *symmetrized* version of the central-path equations (74), for a decreasing sequence of values of μ.

A unified description of different symmetric formulations of the central path was developed by Zhang [Zha98], who notes that positive definite matrices S, Z satisfy $SZ = \mu I$ if and only if there exists a nonsingular matrix P such that

$$\frac{1}{2}(P^T Z S P^{-T} + P^{-1} S Z P) = \mu I.$$

The algorithm outlined above uses $P = R$ defined in Step 2 (known as the *Nesterov-Todd* scaling matrix), but many other choices are possible.

Using Zhang's parametrization, the central path equations can be expressed as

$$\mathcal{H}(Z(\mu)S(\mu)) = \mu I$$
$$S(\mu) + \mathcal{A}(y(\mu)) + D = 0$$
$$\mathcal{A}^{\mathrm{adj}}(Z(\mu)) + \mathcal{B}^{\mathrm{adj}}z + c = 0$$
$$\mathcal{B}(y(\mu)) + d = 0.$$

The Newton directions at some y, $Z \succ 0$, $S \succ 0$ are obtained by linearizing these equations and solving the linearized equations

$$\mathcal{H}(\Delta Z S + Z \Delta S) = \mu I - \mathcal{H}(ZS) \tag{75}$$
$$\Delta S + \mathcal{A}(\Delta y) = -(\mathcal{A}(y) + S + D) \tag{76}$$
$$\mathcal{A}^{\mathrm{adj}}(\Delta Z) + \mathcal{B}^{\mathrm{adj}}(\Delta z) = -(\mathcal{A}^{\mathrm{adj}}(Z) + \mathcal{B}^{\mathrm{adj}}(z) + c). \tag{77}$$
$$\mathcal{B}(\Delta y) = -(\mathcal{B}(y) + d) \tag{78}$$

We can now interpret and justify the search directions defined in Steps 3, 4, and 5 as Newton directions. We first note that, if we choose R as in Step 2,

$$\mathcal{H}(ZS) = \frac{1}{2}(R^T Z S R^{-T} + R^{-1} S Z R) = \mathbf{diag}(\lambda)^2,$$

so the Newton equations (75)–(78) reduce to

$$\mathcal{H}(\Delta Z S + Z \Delta S) = \mu I - \mathbf{diag}(\lambda)^2 \tag{79}$$
$$\Delta S + \mathcal{A}(\Delta y) = -(\mathcal{A}(y) + S + D) \tag{80}$$
$$\mathcal{A}^{\mathrm{adj}}(\Delta Z) + \mathcal{B}^{\mathrm{adj}}(\Delta z) = -(\mathcal{A}^{\mathrm{adj}}(Z) + \mathcal{B}^{\mathrm{adj}}(z) + c). \tag{81}$$
$$\mathcal{B}(\Delta y) = -(\mathcal{B}(y) + d) \tag{82}$$

Comparing this system with the sets of equations (66)–(69) and (70)–(73), we see that, except for the term $\mathcal{H}(\Delta Z^{\mathrm{a}} \Delta S^{\mathrm{a}})$, these equations are identical to the Newton equations. More precisely, if we delete the term $\mathcal{H}(\Delta Z^{\mathrm{a}} \Delta S^{\mathrm{a}})$,

the solution (79)–(82) is given by $\Delta y = \Delta y^{\mathrm{a}} + \Delta y^{\mathrm{c}}$, $\Delta S = \Delta S^{\mathrm{a}} + \Delta S^{\mathrm{c}}$, $\Delta Z = \Delta Z^{\mathrm{a}} + \Delta Z^{\mathrm{c}}$, $\Delta z = \Delta z^{\mathrm{a}} + \Delta z^{\mathrm{c}}$,

A distinguishing feature of the predictor-corrector method is that the Newton equations are solved in two steps, by solving the two sets of linear equations (66)–(69) and (70)–(73) separately, instead of solving a single set of equations (79)–(82). This strategy has proven to be very successful in primal-dual methods for linear programming [Meh91], and offers two advantages. The first, and most important, advantage is that it allows us to select the value of μ adaptively. In the algorithm described above, this idea is implemented as follows. In Step 3 we compute the affine scaling direction, $i.e.$, the limit of the Newton direction for $\mu \to 0$. In Step 4, we first assess the 'quality' of the affine direction as a search direction, by computing the ratio

$$\eta = \frac{\mathbf{Tr}((S + \alpha\Delta S^{\mathrm{a}})(Z + \beta\Delta Z^{\mathrm{a}}))}{\mathbf{Tr}(SZ)},$$

where we take $\alpha = 1$, $\beta = 1$ if possible, and otherwise take the maximum α and β that satisfy $S + \alpha\Delta S^{\mathrm{a}} \succeq 0$, resp. $Z + \beta\Delta Z^{\mathrm{a}} \succeq 0$. The ratio η gives the reduction in $\mathbf{Tr}(SZ)$ that we can achieve by using the affine scaling direction. If the ratio is small, we assume the affine scaling direction is a good search direction and we choose a small value of μ; if the ratio η is large, we choose a larger value of μ. Choosing $\mu = \eta^{\delta}\,\mathbf{Tr}(SZ)/m$ means that we select the Newton step for central points $y(\mu)$, $S(\mu)$, $Z(\mu)$, with

$$\mathbf{Tr}(S(\mu)Z(\mu)) = \eta^{\delta}\,\mathbf{Tr}(SZ).$$

The second advantage of solving two linear systems is that we can add a higher-order correction term when linearizing the equation $\mathcal{H}(Z(\mu)S(\mu)) = \mu I$. In Newton's method we linearize this equation by expanding

$$\mathcal{H}((Z + \Delta Z)(S + \Delta S)) = \mathcal{H}(ZS) + \mathcal{H}(\Delta ZS + Z\Delta S) + \mathcal{H}(\Delta Z\Delta S)$$

and omitting the second-order term, which yields a linear equation

$$\mathcal{H}(ZS) + \mathcal{H}(Z\Delta S + \Delta ZS) = \mu I.$$

The combined directions, $\Delta Z = \Delta Z^{\mathrm{a}} + \Delta Z^{\mathrm{c}}$, $\Delta S = \Delta S^{\mathrm{a}} + \Delta S^{\mathrm{c}}$, used in the predictor-corrector method, on the other hand, satisfies

$$\mathcal{H}(ZS) + \mathcal{H}((\Delta Z^{\mathrm{a}} + \Delta Z^{\mathrm{c}})S + Z(\Delta S^{\mathrm{a}} + \Delta S^{\mathrm{c}})S) + \mathcal{H}(\Delta Z^{\mathrm{a}}\Delta S^{\mathrm{a}}) = \mu I,$$

which includes part of the second-order term, and can therefore be expected to be more accurate.

We conclude by pointing out that the two sets of linear equations (66)–(69) and (70)–(73) only differ in the righthand side, so the cost of solving both systems is about the same as the cost of solving one system.

Step size

After computing the search directions, we update the variables in step 5. We use different step sizes α and β for the primal and dual variables. If possible, we make a full step ($\alpha = 1$, $\beta = 1$). If this is unacceptable because it results in values of S and Z that are not positive definite, we decrease α and/or β, and make a step equal to a fraction 0.99 of the maximum steps that satisfy $S + \alpha \Delta S \succeq 0$ and $Z + \beta \Delta Z \succeq 0$.

A.3 Solving the Newton equations

When applied to an SDP that is primal and dual feasible, the predictor-corrector method usually converges in 10–50 iterations. As a rule of thumb, the overall cost of solving the SDP and its dual is therefore equal to the cost of solving 10–50 linear equations of the form

$$\mathcal{H}(\Delta Z S + Z \Delta S) = D_1 \tag{83}$$

$$\Delta S + \mathcal{A}(\Delta y) = D_2 \tag{84}$$

$$\mathcal{A}^{\mathrm{adj}}(\Delta Z) + \mathcal{B}^{\mathrm{adj}}(\Delta z) = D_3 \tag{85}$$

$$\mathcal{B}(\Delta y) = D_4. \tag{86}$$

It can be shown that these equations have a unique solution if $(\mathcal{A}, \mathcal{B})$ has full rank [TTT98, p.777].

The Newton equations can be simplified by eliminating ΔS. Using the definition of R in (65), we first note that equation (83) can be written as

$$(R^T \Delta Z R + R^{-1} \Delta S R^{-T}) \, \mathbf{diag}(\lambda) + \mathbf{diag}(\lambda)(R^{-1} \Delta S R^{-T} + R^T \Delta Z R) = 2 D_1.$$

The general solution of the homogeneous equation ($D_1 = 0$) is $\Delta S = -R R^T \Delta Z R R^T$. A particular solution is $\Delta Z = 0$,

$$\Delta S = 2 R (D_1 \circ G) R^T$$

where $G_{ij} = 1/(\lambda_i + \lambda_j)$. All solutions of (83) can therefore be written as

$$\Delta S = -R R^T \Delta Z R R^T + 2 R (D_1 \circ G) R^T.$$

Substituting in (84) gives an equivalent set of linear equations

$$-W \Delta Z W + \mathcal{A}(\Delta y) = D \tag{87}$$

$$\mathcal{A}^{\mathrm{adj}}(\Delta Z) + \mathcal{B}^{\mathrm{adj}}(\Delta z) = D_3 \tag{88}$$

$$\mathcal{B}(\Delta y) = D_4 \tag{89}$$

where $W = R R^T$, $D = D_2 - 2 R (D_1 \circ G) R^T$.

A general-purpose SDP solver like SDPT3 solves (87)–(89) by eliminating ΔZ from the first equation, which yields

$$\mathcal{A}^{\mathrm{adj}}(W^{-1} \mathcal{A}(\Delta y) W^{-1}) + \mathcal{B}^{\mathrm{adj}}(\Delta z) = D_3 + \mathcal{A}^{\mathrm{adj}}(W^{-1} D W^{-1}) \tag{90}$$

$$\mathcal{B}(\Delta y) = D_4, \tag{91}$$

and solving for Δy and Δz.

B Derivation of (50) and (52)

B.1 Expression for H

Let $H \in \mathbf{S}^{n+1}$ be defined as

$$H_{ij} = \mathbf{Tr}(F_i W F_j W), \qquad i, j = 1 \ldots, n+1,$$

where

$$F_i = \begin{bmatrix} X_i & e_i \\ e_i^T & 0 \end{bmatrix}, \quad i = 1, \ldots, n, \qquad F_{n+1} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix},$$

and $X_i \in \mathbf{S}^n$. To simplify the expressions for H we first note that if we partition W as

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{12}^T & W_{22} \end{bmatrix},$$

with $W_{11} \in \mathbf{S}^n$, $W_{12} \in \mathbf{R}^n$, and $W_{22} \in \mathbf{R}$, then

$$W F_i = \begin{bmatrix} W_{11} X_i + W_{12} e_i^T & W_{11} e_i \\ W_{12}^T X_i + W_{22} e_i^T & W_{12}^T e_i \end{bmatrix}, \quad i = 1, \ldots, n, \qquad W F_{n+1} = \begin{bmatrix} 0 & 2W_{12} \\ 0 & 2W_{22} \end{bmatrix}.$$

The leading $n \times n$ block of H is given by

$$\begin{aligned}
H_{ij} &= \mathbf{Tr}((W_{11} X_i + W_{12} e_i^T)(W_{11} X_j + W_{12} e_j^T)) + e_i^T W_{11}(X_j W_{12} + e_j W_{22}) \\
&\quad + (W_{12}^T X_i + W_{22} e_i^T) W_{11} e_j + e_i^T W_{12} W_{12}^T e_j \\
&= \mathbf{Tr}(W_{11} X_i W_{11} X_j) + 2 e_i^T W_{11} X_j W_{12} + 2 W_{12}^T X_i W_{11} e_j + 2 W_{22} e_i^T W_{11} e_j \\
&\quad + 2 e_i^T W_{12} W_{12}^T e_j
\end{aligned}$$

for $i, j = 1, \ldots, n$. The last column is given by

$$\begin{aligned}
H_{i,n+1} &= 2(W_{12}^T X_i + W_{22} e_i^T) W_{12} + 2 e_i^T W_{12} W_{22} \\
&= 2 W_{12}^T X_i W_{12} + 4(e_i^T W_{12}) W_{22}
\end{aligned}$$

for $i = 1, \ldots, n$, and

$$H_{n+1,n+1} = 4 W_{22}^2.$$

In summary,

$$\begin{aligned}
H &= \begin{bmatrix} H_1 & 0 \\ 0 & 0 \end{bmatrix} + 2 \begin{bmatrix} W_{11} \\ W_{12}^T \end{bmatrix} \begin{bmatrix} H_2 & 0 \end{bmatrix} + 2 \begin{bmatrix} H_2^T \\ 0 \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} \end{bmatrix} \\
&\quad + 2 W_{22} \begin{bmatrix} W_{11} & W_{12} \\ W_{12}^T & W_{22} \end{bmatrix} + 2 \begin{bmatrix} W_{12} \\ W_{22} \end{bmatrix} \begin{bmatrix} W_{12}^T & W_{22} \end{bmatrix}
\end{aligned}$$

where

$$\begin{aligned}
(H_1)_{ij} &= \mathbf{Tr}(X_i W_{11} X_j W_{11}), \quad i, j = 1, \ldots, n \\
H_2 &= \begin{bmatrix} X_1 W_{12} & X_2 W_{12} & \cdots & X_n W_{12} \end{bmatrix}.
\end{aligned}$$

This proves (50).

B.2 Expression for G

Let $G \in \mathbf{R}^{(n+1) \times p}$ be defined by

$$G_{ij} = \mathbf{Tr}(F_i M_j), \qquad i = 1, \ldots, n, \quad j = 1, \ldots, p.$$

We will partition M_j as

$$M_j = \begin{bmatrix} M_{j,11} & M_{j,12} \\ M_{j,12}^T & M_{j,22} \end{bmatrix},$$

with $M_{j,11} \in \mathbf{S}^n$. We have

$$G_{ij} = \mathbf{Tr}(X_i M_{j,11}) + 2 e_i^T M_{j,12}, \quad i = 1, \ldots, n, \qquad G_{ij} = 2 M_{j,22}.$$

From this it is easy to see that

$$G = 2 \begin{bmatrix} Y_1 B & Y_2 B & \cdots & Y_p B \\ 0 & 0 & \cdots & 0 \end{bmatrix} + 2 \begin{bmatrix} M_{1,12} & M_{2,12} & \cdots & M_{p,12} \\ M_{1,22} & M_{2,22} & \cdots & M_{p,22} \end{bmatrix},$$

with Y_j is the solution of $A Y_j + Y_j A^T + M_{j,11} = 0$.

C Derivation of (53) and (54)

Let $X(v)$ be the solution of the Lyapunov equation

$$AX(v) + X(v)A^T + v B^T + B v^T = 0,$$

i.e., $X(v) = \sum_{i=1}^n v_i X_i$. The matrices H_1 and H_2 satisfy

$$v^T H_1 v = \mathbf{Tr}(X(v) W_{11} X(v) W_{11}), \qquad H_2 v = X(v) W_{12}$$

for all v.

C.1 Expression for H_1

First suppose A is diagonal, $A = \mathbf{diag}(\lambda)$, with $\lambda \in \mathbf{C}^n$. Define $\Sigma \in \mathbf{H}^{n \times n}$ as

$$\Sigma_{ij} = \frac{1}{\lambda_i + \bar{\lambda}_j}, \quad i, j = 1, \ldots, n.$$

The solution of $\mathbf{diag}(\lambda) Y + Y \mathbf{diag}(\lambda)^* + G w^* + w G^* = 0$, where $G \in \mathbf{C}^n$, is given by

$$Y(w) = -(G w^* + w G^*) \circ \Sigma. \tag{92}$$

Therefore, for general $S \in \mathbf{H}^{(n+1) \times (n+1)}$,

$$\mathbf{Tr}(Y(w)SY(w)S) = \mathbf{Tr}(((Gw^* + wG^*) \circ \Sigma)S((Gw^* + wG^*) \circ \Sigma)S)$$
$$= \mathbf{Tr}(D_G\Sigma D_w^* S D_G \Sigma D_w^* S) + \mathbf{Tr}(D_G \Sigma D_w^* S D_w \Sigma D_G^* S)$$
$$+ \mathbf{Tr}(D_w \Sigma D_G^* S D_G \Sigma D_w^* S) + \mathbf{Tr}(D_w \Sigma D_G^* S D_w \Sigma D_G^* S)$$

where $D_x = \mathbf{diag}(x)$. Now we use the property that for $A, B \in \mathbf{C}^{n \times n}$,

$$\mathbf{Tr}(D_x A D_y B) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i A_{ij} y_j B_{ji} = x^T (A \circ B^T) y.$$

This gives

$$\mathbf{Tr}(Y(w)SY(w)S)$$
$$= w^*((SD_G\Sigma) \circ (SD_G\Sigma)^T)\bar{w} + w^*(S \circ (\Sigma D_G^* S D_G \Sigma)^T)w$$
$$+ w^T((\Sigma D_G^* S D_G \Sigma) \circ S^T)\bar{w} + w^T((\Sigma D_G^* S) \circ (\Sigma D_G^* S)^T)w$$
$$= 2\Re(w^T((\Sigma D_G^* S) \circ (\Sigma D_G^* S)^T)w) + 2\Re(w^*(S \circ (\Sigma D_G^* S D_G \Sigma)^T)w). \quad (93)$$

Now suppose A is not diagonal, but diagonalizable, with $AV = V\,\mathbf{diag}(\lambda)$. The solution of $AX + XA^T + Bv^T + vB^T = 0$ is given by

$$X(v) = VY(V^{-1}v)V^*$$

where $Y(w)$ is the solution of

$$\mathbf{diag}(\lambda)Y + Y\,\mathbf{diag}(\lambda)^* + V^{-1}Bw^* + wB^T V^{-*} = 0.$$

Therefore, for $W_{11} \in \mathbf{S}^{n+1}$,

$$\mathbf{Tr}(X(v)W_{11}X(v)W_{11}) = \mathbf{Tr}(Y(V^{-1}v)V^*W_{11}VY(V^{-1}v)V^*W_{11}V),$$

so we can apply (93) with $w = V^{-1}v$, $S = V^*W_{11}V$, $G = V^{-1}B$, and

$$\mathbf{Tr}(X(v)W_{11}X(v)W_{11})$$
$$= 2\Re(v^T V^{-T}((\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{11}V) \circ (\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{11}V)^T)V^{-1}v)$$
$$+ 2\Re(v^T V^{-*}(V^* W_{11}V) \circ (\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{11}V\,\mathbf{diag}(V^{-1}B)\Sigma)^T)V^{-1}v).$$

In conclusion,

$$H_1 = 2\Re(V^{-T}\left((\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{11}V) \circ (\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{11}V)^T\right)V^{-1})$$
$$+ 2\Re(V^{-*}\left((V^* W_{11}V) \circ (\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{11}V\,\mathbf{diag}(V^{-1}B)\Sigma)^T\right)V^{-1}).$$

C.2 Expression for H_2

With $Y(w)$ defined as in (92), and $s \in \mathbf{C}^n$,

$$\begin{aligned}
Y(w)s &= -((Gw^* + wG^*) \circ \Sigma)s \\
&= -D_G \Sigma D_w^* s - D_w \Sigma D_G^* s \\
&= -D_G \Sigma D_s \bar{w} - D_w \Sigma D_G^* s \\
&= -D_G \Sigma D_s \bar{w} - \mathbf{diag}(\Sigma D_G^* s)w.
\end{aligned}$$

To determine H_2 we apply this expression with $s = V^* W_{12}$, $G = V^{-1}B$, and $w = V^{-1}v$:

$$\begin{aligned}
X(v)W_{12} &= VY(V^{-1}v)V^* W_{12} \\
&= -V\,\mathbf{diag}(V^{-1}B)\Sigma\,\mathbf{diag}(V^* W_{12})\bar{V}^{-1}v \\
&\quad - V\,\mathbf{diag}(\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{12})V^{-1}v.
\end{aligned}$$

Therefore,

$$H_2 = -V\,\mathbf{diag}(V^{-1}B)\Sigma\,\mathbf{diag}(V^* W_{12})\bar{V}^{-1} - V\,\mathbf{diag}(\Sigma\,\mathbf{diag}(V^{-1}B)^* V^* W_{12})V^{-1}.$$

D Non-controllable (A, B)

In this appendix we discuss how the assumption that (A_k, B_k) is controllable can be relaxed to (A_k, B_k) stabilizable, provided the range of Q_k is in the controllable subspace of (A_k, B_k). For simplicity we explain the idea for problems with one constraint, and omit the subscripts k, as in (29). We define $\mathcal{M}(x) = \sum_{i=1}^{p} x_i M_i - N_i$, and assume the problem is strictly (primal) feasible.

Let T be a unitary state transformation such that

$$\tilde{A} = \begin{bmatrix} \tilde{A}_1 & \tilde{A}_{12} \\ 0 & \tilde{A}_2 \end{bmatrix} = T^T A T, \qquad \tilde{B} = \begin{bmatrix} \tilde{B}_1 \\ 0 \end{bmatrix} = BT$$

where $(\tilde{A}_1, \tilde{B}_1)$ is controllable and $\tilde{A}_2$ is Hurwitz. Note that

$$\begin{bmatrix} T^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A^T P + PA & PB \\ B^T P & 0 \end{bmatrix} \begin{bmatrix} T & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} \tilde{A}^T \tilde{P} + \tilde{P}\tilde{A} & \tilde{P}\tilde{B} \\ \tilde{B}^T \tilde{P} & 0 \end{bmatrix}$$

where $\tilde{P} = T^T PT$. Let

$$\tilde{P} = \begin{bmatrix} \tilde{P}_1 & \tilde{P}_{12} \\ \tilde{P}_{12}^T & \tilde{P}_2 \end{bmatrix}, \qquad \tilde{Q} = \begin{bmatrix} \tilde{Q}_1 & \tilde{Q}_{12} \\ \tilde{Q}_{12}^T & \tilde{Q}_2 \end{bmatrix} = TQT^T$$

and let

$$\mathcal{M} = \begin{bmatrix} \tilde{\mathcal{M}}_1 & \tilde{\mathcal{M}}_{12} & \tilde{\mathcal{M}}_{13} \\ \tilde{\mathcal{M}}_{12}^T & \tilde{\mathcal{M}}_2 & \tilde{\mathcal{M}}_{23} \\ \tilde{\mathcal{M}}_{13}^T & \tilde{\mathcal{M}}_{23}^T & \tilde{\mathcal{M}}_3 \end{bmatrix} = \begin{bmatrix} T^T & 0 \\ 0 & I \end{bmatrix} \mathcal{M}(x) \begin{bmatrix} T & 0 \\ 0 & I \end{bmatrix}.$$

Then it holds that (29) with strict inequality is equivalent to

$$\text{minimize} \quad q^T x + \mathbf{Tr}(\tilde{Q}_1 \tilde{P}) + 2\,\mathbf{Tr}(\tilde{Q}_{12} P_{12}) + \mathbf{Tr}(\tilde{Q}_2 \tilde{P}_2)$$

$$\text{subject to} \quad \begin{bmatrix} \tilde{P}_1 \tilde{A}_1 + \tilde{A}_1^T \tilde{P}_1 & \tilde{P}_1 \tilde{A}_{12} + \tilde{P}_{12}\tilde{A}_2 + \tilde{A}_1^T \tilde{P}_{12} & \tilde{P}_1 \tilde{B}_1 \\ * & \tilde{P}_{12}^T \tilde{A}_{12} + \tilde{P}_2 \tilde{A}_2 + \tilde{A}_{12}^T \tilde{P}_{12} + \tilde{A}_2^T \tilde{P}_2 & \tilde{P}_{12}^T \tilde{B}_1 \\ * & * & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} \tilde{\mathcal{M}}_1 & \tilde{\mathcal{M}}_{12} & \tilde{\mathcal{M}}_{13} \\ \tilde{\mathcal{M}}_{12}^T & \tilde{\mathcal{M}}_2 & \tilde{\mathcal{M}}_{23} \\ \tilde{\mathcal{M}}_{13}^T & \tilde{\mathcal{M}}_{23}^T & \tilde{\mathcal{M}}_3 \end{bmatrix} \succ 0. \tag{94}$$

By the Schur complement formula the above constraint is equivalent to

$$\begin{bmatrix} \tilde{P}_1 \tilde{A}_1 + \tilde{A}_1^T \tilde{P}_1 & \tilde{P}_1 \tilde{B}_1 \\ \tilde{B}_1^T \tilde{P}_1 & 0 \end{bmatrix} + \begin{bmatrix} \tilde{\mathcal{M}}_1 & \tilde{\mathcal{M}}_{13} \\ \tilde{\mathcal{M}}_{13}^T & \tilde{\mathcal{M}}_3 \end{bmatrix} \succ 0$$

and

$$\tilde{P}_{12}^T \tilde{A}_{12} + \tilde{P}_2 \tilde{A}_2 + \tilde{A}_{12}^T \tilde{P}_{12} + \tilde{A}_2^T \tilde{P}_2 + \tilde{\mathcal{M}}_2$$
$$- \begin{bmatrix} \tilde{P}_1 \tilde{A}_{12} + \tilde{P}_{12}\tilde{A}_2 + \tilde{A}_1^T \tilde{P}_{12} + \tilde{\mathcal{M}}_{12} \\ B_1^T \tilde{P}_{12} + \tilde{\mathcal{M}}_{23}^T \end{bmatrix}^T \begin{bmatrix} \tilde{P}_1 \tilde{A}_1 + \tilde{A}_1^T \tilde{P}_1 + \tilde{\mathcal{M}}_1 & \tilde{P}_1 \tilde{B}_1 + \tilde{\mathcal{M}}_{13} \\ \tilde{B}_1^T \tilde{P}_1 + \tilde{\mathcal{M}}_{13}^T & \tilde{\mathcal{M}}_3 \end{bmatrix}^{-1}$$
$$\times \begin{bmatrix} \tilde{P}_1 \tilde{A}_{12} + \tilde{P}_{12}\tilde{A}_2 + \tilde{A}_1^T \tilde{P}_{12} + \tilde{\mathcal{M}}_{12} \\ \tilde{B}_1^T \tilde{P}_{12} + \tilde{\mathcal{M}}_{23}^T \end{bmatrix} \succ 0.$$

Now by our assumption that the range of Q is in the controllable subspace of (A, B), we have $\tilde{Q}_2 = 0$ and $\tilde{Q}_{12} = 0$. Then $\tilde{P}_{12}$ and $\tilde{P}_2$ only appear in the latter matrix inequality. This shows that it is possible to partition the optimization problem into one problem of the original form for which $(\tilde{A}_1, \tilde{B}_1)$ is controllable involving the variables x and $\tilde{P}_1$, and a feasibility problem involving $\tilde{P}_{12}$ and $\tilde{P}_2$. Notice that feasible $\tilde{P}_{12}$ and $\tilde{P}_2$ can be found by solving a Lyapunov equation for $\tilde{P}_2$. Hence all results presented in this article extend to the case when (A, B) is stabilizable. Notice however that there does not exist strictly dual feasible Z if (A, B) is not controllable.

E Linear independence

In this appendix, we relax the assumption that the mapping (6) has full rank.

E.1 Change of variables

Consider the constraint in (29) which can be written as

$$-\mathcal{A}(P, x) = \begin{bmatrix} PA + A^T P & PB \\ B^T P & 0 \end{bmatrix} + \sum_{i=1}^{p} x_i \begin{bmatrix} M_{1,i} & M_{12,i} \\ M_{12,i}^T & M_{2,i} \end{bmatrix} \succeq \begin{bmatrix} N_1 & N_{12} \\ N_{12}^T & N_2 \end{bmatrix}.$$

Let P_i solve

$$A^T P_i + P_i A = M_{1,i}, \quad i = 1, \ldots, p$$

and let $\tilde{M}_{12,i} = M_{12,i} - P_i B$, $i = 1, \ldots, p$. Then with

$$\bar{P} = P - \sum_{i=1}^{p} x_i P_i$$

it holds that the above LMI is satisfied for $P = P^T$ and some x if and only if $\bar{P}$ and x satisfy

$$-\tilde{\mathcal{A}}(\bar{P}, x) = \begin{bmatrix} \bar{P}A + A^T\bar{P} & \bar{P}B \\ B^T\bar{P} & 0 \end{bmatrix} + \sum_{i=1}^{p} x_i \tilde{M}_i \succeq N$$

where

$$\tilde{M}_i = \begin{bmatrix} 0 & \tilde{M}_{12,i} \\ \tilde{M}_{12,i}^T & M_{2,i} \end{bmatrix}.$$

Hence it is no loss in generality to assume that the LMI constraint is of a form where the $M_{1,i}$-entries are zero. The objective function is transformed to $\tilde{q}^T x + \mathbf{Tr}\, Q\bar{P}$, where $\tilde{q}_i = q_i + \mathbf{Tr}\, QP_i$. We remark that this structure of the constraint is inherent in certain applications such as IQCs as they are defined in the Matlab IQC-toolbox. Moreover, notice that we could have defined the change of variables slightly differently using an affine change of variables such that N would also have had a zero 1,1-block. However, the notation would have been more messy, and it would also complicate the presentation in what follows.

E.2 Linear independence

The full rank property of $\mathcal{A}(P, x)$ is needed for uniqueness of the solution of the Newton equations. In this subsection we show that a sufficient and more easily verified condition is that A is Hurwitz and $\tilde{M}_i$, $i = 1, \ldots, p$, are linearly independent. We make use of the specific structure developed above to show that $\mathcal{A}(P, x) = 0$ implies $(P, x) = 0$. By the change of variables in the previous subsection, $\mathcal{A}(P, x) = 0$ is equivalent to $\tilde{\mathcal{A}}(\bar{P}, x) = 0$. In the 1,1-position this equation reads

$$\bar{P}A + A^T\bar{P} = 0$$

and since A is Hurwitz it follows that $\bar{P} = 0$. This implies that

$$\sum_{i=1}^{p} x_i \tilde{M}_i = 0$$

and since $\tilde{M}_i$, $i = 1, \ldots, p$, are linearly independent it follows that $x = 0$. By the definition of the change of variables it is now true that $(P, x) = 0$. We remark that the above proof easily extends to the general problem formulation in the introduction.

E.3 Linear dependence

In case $\tilde{M}_i$, $i = 1, \ldots, p$, are linearly dependent, then either the objective function is not bounded from below, or the problem can be reduced to an equivalent problem with fewer variables for which $\tilde{M}_i$, $i = 1, \ldots, p$, are linearly independent. To this end define

$$\tilde{M} = \begin{bmatrix} \mathbf{svec}(\tilde{M}_1) \ \mathbf{svec}(\tilde{M}_2) \ \cdots \ \mathbf{svec}(\tilde{M}_p) \end{bmatrix}.$$

Apply a singular value decomposition to $\tilde{M}$:

$$\tilde{M} = U \begin{bmatrix} \Sigma_1 \ 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix},$$

where Σ_1 has full column rank. Define a change of variables for x via

$$\bar{x} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} = \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} x$$

Now clearly $\tilde{M}x = \bar{M}\bar{x}_1$, where $\bar{M} = U\Sigma_1$. Therefore we can rewrite the constraint in the variables $\bar{x}_1$ and with a set of linearly independent matrices, $\bar{M}_i$, given by the inverse symmetric vectorization of the columns of $\bar{M}$. The part of the primal objective function involving x can be written as as

$$c^T x = c^T \begin{bmatrix} V_1 \ V_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} x = \bar{c}_1^T \bar{x}_1 + \bar{c}_2^T \bar{x}_2$$

where $\bar{c}_1 = V_1^T c$, $\bar{c}_2 = V_2^T c$. Since $\bar{x}_2$ is not present in the constraint the objective function is bounded below only if

$$\bar{c}_2 = V_2^T c = 0$$

Hence, this is a necessary condition for the optimization problem to have a solution. Therefore either the problem is not bounded below or there is an equivalent problem involving fewer variables for which $\bar{M}_i$, $i = 1, \ldots, p$ are linearly independent.

The cost of the above operation is $O(n^3)$, where we assume that p is of $O(n)$.

References

[AA98]   P. Apkarian and R. J. Adams. Advanced gain-scheduled techniques for uncertain systems. *IEEE Trans. Control Sys. Tech.*, 6(1):21–32, January 1998.

[AG95]   P. Apkarian and P. Gahinet. A convex characterization of gain-scheduled $\mathbf{H}_\infty$ controllers. *IEEE Transactions on Automatic Control*, 40(5):853–864, May 1995.

[AHN⁺97] F. Alizadeh, J. P. Haeberly, M. V. Nayakkankuppam, M. L. Overton, and S. Schmieta. SDPPACK *User's Guide, Version 0.9 Beta.* NYU, June 1997.

[AV00]   B. Alkire and L. Vandenberghe. Handling nonnegative constraints in spectral estimation. In *Proceedings of the 34th Asilomar Conference on Signals, Systems, and Computers*, pages 202–206, 2000.

[AV01]   B. Alkire and L. Vandenberghe. Interior-point methods for magnitude filter design. In *Proceedings of the 2001 IEEE International Conference on Acoustics. Speech, and Signal Processing*, volume VI, pages SPTM–11, 2001.

[AV02]   B. Alkire and L. Vandenberghe. Convex optimization problems involving finite autocorrelation sequences. *Mathematical Programming Series A*, 93:331–359, 2002.

[BB91]   S. Boyd and C. Barratt. *Linear Controller Design: Limits of Performance.* Prentice Hall, 1991.

[BEFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics.* SIAM, Philadelphia, PA, June 1994.

[Bor02]   B. Borchers.   *CSDP 4.2 User's Guide*, 2002.   Available from www.nmt.edu/~borchers/csdp.html.

[BV98]   V. Balakrishnan and L. Vandenberghe. Linear Matrix Inequalities for signal processing: An overview. In *Proceedings of the 32nd Annual Conference on Information Sciences and Systems*, Princeton, New Jersey, March 1998.

[BV03]   V. Balakrishnan and L. Vandenberghe. Semidefinite programming duality and linear time-invariant systems. *IEEE Trans. Aut. Control*, 48:30–41, 2003.

[BW99]   V. Balakrishnan and F. Wang. Efficient computation of a guaranteed lower bound on the robust stability margin for a class of uncertain systems. *IEEE Trans. Aut. Control*, AC-44(11):2185–2190, November 1999.

[BY02]   S. J. Benson and Y. Ye. *DSDP4 — A Software Package Implementing the Dual-Scaling Algorithm for Semidefinite Programming*, 2002. Available from www-unix.mcs.anl.gov/~benson.

[DLS02]   T. N. Davidson, Z.-Q. Luo, and J. F. Sturm. Linear matrix inequality formulation of spectral mask constraints with applications to FIR filter design. *IEEE Transactions on Signal Processing*, 50(11):2702–2715, 2002.

[Doy82]   J. Doyle. Analysis of feedback systems with structured uncertainties. *IEE Proc.*, 129-D(6):242–250, November 1982.

[DTS01]   B. Dumitrescu, Ioan Tabus, and Petre Stoica. On the parametrization of positive real sequences and MA parameter estimation. *IEEE Transactions on Signal Processing*, 49(11):2630–9, November 2001.

[FB97]   M. Fu and N. E. Barabanov. Improved Upper Bounds for the Mixed Structured Singular Value. *IEEE Trans. Aut. Control*, 42(10):1447–1452, October 1997.

[FKN98]   K. Fujisawa, M. Kojima, and K. Nakata. *SDPA User's Manual*, 1998. Available from www.is.titech.ac.jp/~yamashi9/sdpa.

[FTD91]   M. K. H. Fan, A. L. Tits, and J. C. Doyle. Robustness in the presence of mixed parametric uncertainty and unmodeled dynamics. *IEEE Trans. Aut. Control*, AC-36(1):25–38, January 1991.

[GH03]   J. Gillberg and A. Hansson. Polynomial complexity for a Nesterov-Todd potential-reduction method with inexact search directions. In *Proceedings of thet 42nd IEEE Conference on Decision and Control*, 2003.

[GHNV00] Y. Genin, Y. Hachez, Yu. Nesterov, and P. Van Dooren. Convex optimization over positive polynomials and filter design. In *Proceedings of the 2000 UKACC International Conference on Control*, Cambridge University, 2000.

[GHNV03] Y. Genin, Y. Hachez, Yu. Nesterov, and P. Van Dooren. Optimization problems over positive pseudo-polynomial matrices. *SIAM Journal on Matrix Analysis and Applications*, 25(3):57–79, 2003.

[GL95]   M. Green and D. J. N. Limebeer. *Linear Robust Control*. Information and System sciences. Prentice Hall, Englewood Cliffs, NJ, 1995.

[GN95]   P. Gahinet and A. Nemirovskii. *LMI Control Toolbox: the LMI Lab*. The MathWorks, Inc., 1995.

[Hac03]   Y. Hachez. *Convex Optimization over Non-Negative Polynomials: Structured Algorithms and Applications*. PhD thesis, Université catholique de Louvain, Belgium, 2003.

[HB99]   H. Hindi and S. Boyd. Multiobjective $\mathcal{H}_2/\mathcal{H}_\infty$-optimal control via finite-dimensional Q-parametrization and linear matrix inequalities. In *Proc. American Control Conf.*, June 1999.

[Hen03]   D. Henrion. LMI formulations of polynomial matrix problems in robust control. Draft., 2003.

[HV00]   A. Hansson and L. Vandenberghe. Efficient solution of linear matrix inequalities for integral quadratic constraints. In *Proc. IEEE Conf. on Decision and Control*, pages 5033–5034, 2000.

[HV01]   A. Hansson and L. Vandenberghe. A primal-dual potential reduction method for integral quadratic constraints. In *2001 American Control Conference*, pages 3013–3018, Arlington, Virginia, June 2001.

[IH98]   T. Iwasaki and S. Hara. Well-posedness of feedback systems: Insights into exact robustness analysis and approximate computations. *IEEE Trans. Aut. Control*, AC-43(5):619–630, May 1998.

[Jön96]   U. Jönsson. *Robustness Analysis of Uncertain and Nonlinear Systems*. PhD thesis, Lund Institute of Technology, Sweden, 1996.

[KM01]   C.-Y. Kao and A. Megretski. Fast algorithms for solving IQC feasibility and optimization problems. In *Proc. American Control Conf.*, pages 3019–3024, 2001.

[KMJ01]   C.-Y. Kao, A. Megretski, and U. T. Jönsson. A cutting plane algorithm for robustness analysis of periodically time-varying systems. *IEEE Trans. Aut. Control*, 46(4):579–592, 2001.

[KMJ03]   C.-Y. Kao, A. Megretski, and U. Jönsson. Specialized fast algorithms for IQC feasibility and optimization problems. 2003. Submitted to Automatica.

[KND85]   J. Kautsky, N. K. Nichols, and P. Van Dooren. Robust pole assignment in linear state feedback. *Int. J. Control*, 41:1129–1155, 1985.

[Löf02]   J. Löfberg. YALMIP. *Yet another LMI parser*. University of Linköping, Sweden, 2002.

[Meh91]   S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, November 1991.

[MR97]   A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Trans. Aut. Control*, 42(6):819–830, June 1997.

[OB99]   J. Oishi and V. Balakrishnan. Linear controller design for the NEC laser bonder via LMI optimization. In Laurent El Ghaoui and Silviu-Iulian Niculescu, editors, *Advances in Linear Matrix Inequality Methods in Control*, Advances in Control and Design. SIAM, 1999.

[Pac94]   A. Packard. Gain scheduling via linear fractional transformations. *Syst. Control Letters*, 22:79–92, 1994.

[Par00]   P. A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, Pasadena, California, May 2000.

[Ran96]   A. Rantzer. On the Kalman-Yakubovich-Popov lemma. *Syst. Control Letters*, 28(1):7–10, 1996.

[Saf82]   M. G. Safonov. Stability margins of diagonally perturbed multivariable feedback systems. *IEE Proc.*, 129-D:251–256, 1982.

[Stu01]   J. F. Sturm. *Using SEDUMI 1.02, a Matlab Toolbox for Optimization Over Symmetric Cones*, 2001. Available from `fewcal.kub.nl/sturm/software/sedumi.html`.

[Stu02]   J. F. Sturm. Implementation of interior point methods for mixed semidefinite and second order cone optimization problems. *Optimization Methods and Software*, 17(6):1105–1154, 2002.

[TTT98]   M. J. Todd, K. C. Toh, and R. H. Tütüncü. On the Nesterov-Todd direction in semidefinite programming. *SIAM J. on Optimization*, 8(3):769–796, 1998.

[TTT02]   K. C. Toh, R. H. Tütüncü, and M. J. Todd. *SDPT3 version 3.02. A Matlab software for semidefinite-quadratic-linear programming*, 2002. Available from `www.math.nus.edu.sg/~mattohkc/sdpt3.html`.

[WB96]   S.-P. Wu and S. Boyd. SDPSOL: *A Parser/Solver for Semidefinite Programming and Determinant Maximization Problems with Matrix Structure. User's Guide, Version Beta*. Stanford University, June 1996.

[WB02]   F. Wang and V. Balakrishnan. Improved stability analysis and gain-scheduled controller synthesis for parameter-dependent systems. *IEEE Trans. Aut. Control*, 47(5):720–734, May 2002.

[WBV98]   S.-P. Wu, S. Boyd, and L. Vandenberghe. FIR filter design via spectral factorization and convex optimization. In B. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, volume 1, pages 215–245. Birkhauser, 1998.

[WBZ$^+$03] F. Wang, V. Balakrishnan, P. Zhou, J. Chen, R. Yang, and C. Frank. Optimal array pattern synthesis using semidefinite programming. *IEEE Trans. Signal Processing*, 51(5):1172–1183, May 2003.

[WHV03]   R. Wallin, H. Hansson, and L. Vandenberghe. Comparison of two structure-exploiting optimization algorithms for integral quadratic constraints. In *4th IFAC Symposium on Robust Control Design*, Milan, Italy, 25-27 June 2003. IFAC.

[Wil71]   J. C. Willems. Least squares stationary optimal control and the algebraic Riccati equation. *IEEE Transactions on Automatic Control*, AC-16(6):621–634, 1971.

[ZDG96]   K. Zhou, J. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1996.

[Zha98]   Y. Zhang. On extending some primal-dual interior-point algorithms from linear programming to semidefinite programming. *SIAM J. on Optimization*, 8:365–386, 1998.