On the implementation of primal-dual interior-point methods for semidefinite programming problems derived from the KYP lemma

Lieven Vandenberghe[1] V. Ragu Balakrishnan[2] Ragnar Wallin[3] Anders Hansson[4]

Abstract

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

$$\begin{array}{ll} \text{minimize} & q^T x + \sum_{k=1}^K \mathbf{Tr}(Q_k P_k) \\ \text{subject to} & \left[\begin{array}{cc} A_k^T P_k + P_k A_k & P_k B_k \\ B_k^T P_k & 0 \end{array}\right] \\ & \quad + \sum_{i=1}^p x_i M_{ki} \succeq N_k, \quad k = 1, \ldots, K. \end{array} \tag{1}$$

The optimization variables are $x \in \mathbf{R}^p$ and K 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 $N_k \in \mathbf{S}^{n_k+m_k}$. We assume that (A_k, B_k) is controllable for $k = 1, \ldots, K$, and that $\{\oplus_{k=1}^K M_{ki}\}_{i=1}^p$ are linearly independent. If $n_k = 0$, the kth constraint is interpreted as the linear matrix inequality (LMI) $\sum_{i=1}^p x_i M_{ki} \succeq N_k$.

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 frequency domain inequality

$$\left[\begin{array}{c} (j\omega I - A)^{-1} B \\ I \end{array}\right]^* M \left[\begin{array}{c} (j\omega I - A)^{-1} B \\ I \end{array}\right] \succ 0, \tag{2}$$

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

$$\left[\begin{array}{cc} A^T P + PA & PB \\ B^T P & 0 \end{array}\right] + M \succ 0 \tag{3}$$

with variable $P \in \mathbf{S}^n$ is feasible. Moreover, if (A, B) is controllable, then the lemma holds with nonstrict inequalities in (2) and (3). The KYP lemma forms the basis of some of the most important SDP applications in control; see, for example, [BEFB94, MR97, Jön96, BW99, HV01], The constraints in the KYP-SDP (1) have the same general form as (3), with M replaced with an affine function of the optimization variable x. If $Q_k = 0$, the KYP-SDP is therefore equivalent to minimizing $q^T x$ subject to K (nonstrict) semi-infinite frequency domain inequalities of the form (2), in which M depends affinely on x.

KYP-SDPs are difficult to solve using general-purpose SDP software packages. 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 (2) into a finite-dimensional LMI (3). For this reason, several researchers have proposed alternatives to standard SDP interior-point methods for solving KYP-SDPs. These methods include cutting-plane methods (such as the analytic center cutting-plane method) [Par00, KM01], 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].

In this paper we examine the possibility of exploiting problem structure in KYP-SDPs to speed up standard

[1]Corresponding author. UCLA Department of Electrical Engineering, 68-119 Engineering IV Building, Los Angeles, CA 90095-1594. E-mail: vandenbe@ee.ucla.edu.

[2]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-1285. E-mail: ragu@ecn.purdue.edu.

[3]Division of Automatic Control, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden. E-mail: ragnarw@isy.liu.se.

[4]Division of Automatic Control, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden. E-mail: hansson@isy.liu.se.

primal-dual interior-point methods of the type used in state-of-the-art solvers like SeDuMi, SDPT3 or SDP-PACK. Straightforward linear algebra techniques will allow us to implement the same SDP interior-point methods at a cost that is orders of magnitude less than the cost of general-purpose implementations. More specifically, if $n_k = n$ for $k = 1, \ldots, K$, 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 to n^3. Similar results have previously been obtained for special classes of SDP-KYPs, for example, KYP-SDPs derived for discrete-time FIR systems [AV02, GHNV03].

2 Interior-point algorithms for SDP

2.1 Semidefinite programming

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

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

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

$$\begin{array}{ll} \text{minimize} & \langle c, y\rangle \\ \text{subject to} & \mathcal{A}(y) + D \preceq 0 \\ & \mathcal{B}(y) + d = 0 \end{array} \tag{4}$$

with variable $y \in \mathcal{V}$ is called a *semidefinite programming problem* (SDP). The dual SDP associated with (4) is defined as

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(DZ) + d^T z \\ \text{subject to} & \mathcal{A}^{\text{adj}}(Z) + \mathcal{B}^{\text{adj}}(z) + c = 0 \\ & Z \succeq 0, \end{array} \tag{5}$$

where

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

denote the adjoints of $\mathcal{A}$ and $\mathcal{B}$. The variables in the dual problem are $Z \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_K}$, 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 $\mathcal{B}(y) + d = 0$.

2.2 Interior-point algorithms

Primal-dual interior-point methods solve the pair of SDPs (4) and (5) simultaneously. At each iteration they solve a set of linear equations of the form

$$\begin{aligned} -W\Delta Z W + \mathcal{A}(\Delta y) &= R && (6)\\ \mathcal{A}^{\text{adj}}(\Delta Z) + \mathcal{B}^{\text{adj}}(\Delta z) &= r_{\text{du}} && (7)\\ \mathcal{B}(\Delta y) &= r_{\text{pri}}, && (8) \end{aligned}$$

to compute primal and dual search directions $\Delta y \in \mathcal{V}$, $\Delta Z \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_K}$, $\Delta z \in \mathbf{R}^r$. The scaling matrix W and the righthand side R in these equations are block-diagonal and symmetric ($W, R \in \mathbf{S}^{l_1} \times \cdots \times \mathbf{S}^{l_K}$), 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. In practice the number of iterations is roughly independent of problem size (and of the order of 10–50), so the overall cost of solving the SDP is roughly proportional to the cost of solving a given set of equations of the form (6)–(8).

2.3 General-purpose solvers

A general-purpose implementation of an interior-point method will assume 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

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

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

To solve the equations (6)–(8) we can eliminate ΔZ from the first equation, and substitute $\Delta Z = W^{-1}(\mathcal{A}(\Delta y) - R)W^{-1}$ in the second equation. This yields a symmetric indefinite set of linear equations in Δy, Δz:

$$\begin{aligned} &\mathcal{A}^{\text{adj}}(W^{-1}\mathcal{A}(\Delta y)W^{-1}) + \mathcal{B}^{\text{adj}}(\Delta z) \\ &\quad = \; r_{\text{du}} + \mathcal{A}^{\text{adj}}(W^{-1}RW^{-1}) && (9)\\ &\mathcal{B}(\Delta y) = r_{\text{pri}}. && (10) \end{aligned}$$

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

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

where

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

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

$$\mathcal{A}^{\text{adj}}(W^{-1}\mathcal{A}(\Delta y)W^{-1}) = r_{\text{du}} + \mathcal{A}^{\text{adj}}(W^{-1}RW^{-1}). \tag{11}$$

i.e.,

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

The total cost of an iteration is dominated by the cost of solving these equations plus the cost of forming H. The matrix H is positive definite and almost always dense, so the cost of solving the equations is $(1/3)s^3$ flops. 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.

3 General-purpose SDP solvers and KYP-SDPs

In this section we use the observations made in §2 to estimate the cost of solving KYP-SDPs with general-purpose interior-point software.

For simplicity we assume that $K = 1$, $n_1 = n$, $m_1 = 1$, and consider the problem

$$\begin{array}{ll} \text{min.} & q^T x + \mathbf{Tr}(QP) \\ \text{s.t.} & \left[\begin{array}{cc} A^TP + PA & PB \\ B^TP & 0 \end{array}\right] + \sum_{i=1}^p x_i M_i \succeq N, \end{array} \tag{12}$$

where $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^n$, with (A, B) controllable.

We can assume, without loss of generality, that A is stable. Indeed, (A, B) is controllable by assumption, so there exists a state feedback matrix K such that $A + BK$ is stable. By applying a congruence to both sides of the constraint in (12) and noting that

$$\begin{array}{l} \left[\begin{array}{cc} I & K^T \\ 0 & I \end{array}\right] \left[\begin{array}{cc} A^TP + PA & PB \\ B^TP & 0 \end{array}\right] \left[\begin{array}{cc} I & 0 \\ K & I \end{array}\right] \\ \quad = \left[\begin{array}{cc} (A+BK)^TP + P(A+BK) & PB \\ B^TP & 0 \end{array}\right], \end{array}$$

we can transform the SDP to an equivalent KYP-SDP, with A replaced by $A + BK$.

In §3.1 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 §3.2 we then describe a straightforward technique, based on SDP duality, that reduces the cost to order n^4.

3.1 Primal method

We can express the KYP-SDP (12) as

$$\begin{array}{ll} \text{minimze} & q^Tx + \mathbf{Tr}(QP) \\ \text{subject to} & \mathcal{K}(P) + \mathcal{M}(x) \succeq N \end{array} \tag{13}$$

where

$$\mathcal{K}(P) = \left[\begin{array}{cc} A^TP + PA & PB \\ B^TP & 0 \end{array}\right], \qquad \mathcal{M}(x) = \sum_{i=1}^p x_i M_i.$$

This is in the general form (4), with $\mathcal{V} = \mathbf{S}^n \times \mathbf{R}^p$, $y = (P, x)$, $c = (Q, q)$, $D = N$, and

$$\mathcal{A}(P, x) = -\mathcal{K}(P) - \mathcal{M}(x).$$

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

$$\begin{array}{rcl} \mathcal{K}^{\text{adj}}(Z) & = & \left[\begin{array}{cc} A & B \end{array}\right] Z \left[\begin{array}{c} I \\ 0 \end{array}\right] + \left[\begin{array}{cc} I & 0 \end{array}\right] Z \left[\begin{array}{c} A^T \\ B^T \end{array}\right], \\ \mathcal{M}^{\text{adj}}(Z) & = & (\mathbf{Tr}(M_1Z), \ldots, \mathbf{Tr}(M_pZ)). \end{array}$$

The dual problem of (14) is therefore

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(NZ) \\ \text{subject to} & \mathcal{K}^{\text{adj}}(Z) = Q, \quad \mathcal{M}^{\text{adj}}(Z) = q \\ & Z \succeq 0, \end{array} \tag{14}$$

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

A general-purpose primal-dual method applied to (13) generates iterates x, P, Z. At each iteration it solves a set of linear equations of the form (6)–(8), with variables Δx, ΔP, ΔZ, by reducing it to a smaller positive definite system (11). These equations can be written in matrix-vector form as

$$\left[\begin{array}{cc} H_{11} & H_{12} \\ H_{12}^T & H_{22} \end{array}\right] \left[\begin{array}{c} \mathbf{svec}(\Delta P) \\ \Delta x \end{array}\right] = \left[\begin{array}{c} r_1 \\ r_2 \end{array}\right] \tag{15}$$

where $\mathbf{svec}(\Delta P)$ denotes the lower triangular part of ΔP stored columnwise. The exact expressions for the righthand sides r_1, r_2 are not important for our present purposes and are omitted. The blocks of the coefficient matrix are defined by the relations

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

The coefficient matrix in (15) 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 (12). The actual cost is higher since it includes the cost of constructing the matrices H_{11}, H_{12}, and H_{22}.

3.2 Dual method

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, AV02], and was applied to general KYP-SDPs in [WHV03].

We define a linear mapping $\mathcal{L} : \mathbf{R}^{n+1} \to \mathbf{S}^{n+1}$ as

$$\begin{array}{rcl} \mathcal{L}(u) & = & \left[\begin{array}{cc} \sum_{i=1}^n u_i X_i & \tilde{u} \\ \tilde{u}^T & 2u_{n+1} \end{array}\right] \\ & = & \displaystyle\sum_{i=1}^n u_i \left[\begin{array}{cc} X_i & e_i \\ e_i^T & 0 \end{array}\right] + u_{n+1} \left[\begin{array}{cc} 0 & 0 \\ 0 & 2 \end{array}\right], \end{array}$$

where $\tilde{u} = (u_1, \ldots, u_n)$ and X_i, $i = 1, \ldots, n$, are the solutions of the Lyapunov equations

$$AX_i + X_iA^T + Be_i^T + e_iB^T = 0.$$

It can be verified that

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

$$S = \mathcal{K}(P) \text{ for some } P \in \mathbf{S}^n \iff \mathcal{L}^{\rm adj}(S) = 0.$$

It follows that the first equality in the dual SDP (14) is equivalent to saying that $Z = \mathcal{L}(u) - Z_0$ for some u, where Z_0 is defined as

$$Z_0 = \begin{bmatrix} X_0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad AX_0 + X_0A^T + Q = 0.$$

Substituting in (14), and dropping the constant term $\mathbf{Tr}(NZ_0)$ from the objective, we obtain an equivalent problem

$$\begin{array}{ll} \text{maximize} & \mathcal{L}^{\rm adj}(N)^T u \\ \text{subject to} & \mathcal{L}(u) \succeq Z_0 \\ & \mathcal{M}^{\rm adj}(\mathcal{L}(u)) = q + \mathcal{M}^{\rm adj}(Z_0) \end{array} \qquad (16)$$

with variable $u \in \mathbf{R}^{n+1}$. This SDP has the form (4) with $\mathcal{V} = \mathbf{R}^{n+1}$, $y = u$,

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

and $c = -\mathcal{L}^{\rm adj}(N)$, $D = Z_0$, $d = -q - \mathcal{M}^{\rm adj}(Z_0)$.

The dual of problem (16) is

$$\begin{array}{ll} \text{minimize} & (q + \mathcal{M}^{\rm adj}(Z_0))^T v - \mathbf{Tr}(Z_0 S) \\ \text{subject to} & \mathcal{L}^{\rm adj}(S) - \mathcal{L}^{adj}(\mathcal{M}(v)) + \mathcal{L}^{\rm adj}(N) = 0 \\ & S \succeq 0, \end{array} \qquad (17)$$

with variables $v \in \mathbf{R}^p$ and $S \in \mathbf{S}^{n+1}$. Not surprisingly, the SDP (17) can be interpreted as a reformulation of the original primal problem (13). The first constraint in (17) is equivalent to

$$S - \mathcal{M}(v) + N = \mathcal{K}(P) \qquad (18)$$

for some P. Combined with $S \succeq 0$, this is equivalent to $\mathcal{K}(P) + \mathcal{M}(v) \succeq N$. Using (18) we can also express the objective function as

$$\begin{array}{l} (q + \mathcal{M}^{\rm adj}(Z_0))^T v - \mathbf{Tr}(Z_0 S) \\ \quad = \quad q^T v + \mathbf{Tr}(NZ_0) + \mathbf{Tr}(PQ). \end{array}$$

Comparing with (13), we see that the optimal v in (17) is equal the optimal x in (13). The relation (18) also allows us to recover the optimal P for (13) from the optimal solution (v, S) of (17). In summary, the pair of primal and dual SDPS (16) and (17) are equivalent to the original SDPs (13) and (14).

Now suppose we apply a primal-dual method to (16). The method generates iterates u, v, S. At each iteration a set of linear equations of the form (9)–(10) is solved, which for this SDP reduce to

$$\begin{array}{rcl} \mathcal{L}^{\rm adj}(W^{-1}\mathcal{L}(\Delta u)W^{-1}) + \mathcal{L}^{adj}(\mathcal{M}(\Delta v)) & = & r_3 \\ \mathcal{M}^{\rm adj}(\mathcal{L}(\Delta v)) & = & r_4 \end{array}$$

with variables $\Delta u \in \mathbf{R}^{n+1}$, $\Delta v \in \mathbf{R}^p$. (We omit the expressions for the righthand sides r_3 and r_4.) In matrix form,

$$\begin{bmatrix} H & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} r_3 \\ r_4 \end{bmatrix}, \qquad (19)$$

where H and G are defined by the relations

$$\begin{array}{rcl} H\Delta u & = & \mathcal{L}^{\rm adj}(W^{-1}\mathcal{L}(\Delta u)W^{-1}) \\ G\Delta v & = & \mathcal{L}^{adj}(\mathcal{M}(\Delta v)). \end{array}$$

More explicitly, suppose we define $n+1$ matrices F_i as

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

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

$$H_{ij} = \mathbf{Tr}(F_i W^{-1} F_j W^{-1})) \qquad (20)$$

$$G_{ij} = \mathbf{Tr}(F_i M_j). \qquad (21)$$

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 $G \in \mathbf{R}^{(n+1)\times p}$ 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. In practice, the matrices M_i are often sparse, so the cost of computing G is usually much lower.

At each iteration, we have to construct H and solve the equations (19). From (20), we see that the cost of constructing H is $O(n^4)$. The cost of solving the equations is $O(n^3)$ if we assume $p = O(n)$, since the matrix H is dense.

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.

4 Special-purpose implementation

We now turn to the question of exploiting additional problem structure in a special-purpose implementation. The basis of our method is the dual formulation described in §3.2.

The dual method has two limitations. First, it requires an explicit representation of the mapping $\mathcal{L}$ as $\mathcal{L}(u) = \sum_{i=1}^{n+1} u_i F_i$. This means we have to solve n Lyapunov equations, and store the solutions X_i. For large n this requires a substantial amount of memory. Second, as we have seen, the cost of the method grows as n^4, and is dominated by the cost of forming the matrix H in (20). In this section we show that it is possible to form H fast, at a cost of $O(n^3)$ operations, without explicitly computing the matrices F_i,

It can be shown that H is given by

$$H \quad = \quad \begin{bmatrix} H_1 & 0 \\ 0 & 0 \end{bmatrix} + 2 \begin{bmatrix} W_{11} \\ W_{21} \end{bmatrix} \begin{bmatrix} H_2 & 0 \end{bmatrix}$$

$$+2\left[\begin{array}{c} H_2^T \\ 0 \end{array}\right]\left[\begin{array}{cc} W_{11} & W_{12} \end{array}\right] + 2W_{22}W$$
$$+2\left[\begin{array}{c} W_{12} \\ W_{22} \end{array}\right]\left[\begin{array}{cc} W_{21} & W_{22} \end{array}\right]$$

where

$$\begin{array}{rcl} (H_1)_{ij} & = & \mathbf{Tr}(X_i W_{11} X_j W_{11}) \\ H_2 & = & \left[\begin{array}{cccc} X_1 W_{12} & X_2 W_{12} & \cdots & X_n W_{12} \end{array}\right] \end{array}$$

and W is partitioned as

$$W = \left[\begin{array}{cc} W_{11} & W_{12} \\ W_{21} & W_{22} \end{array}\right]$$

with $W_{11} \in \mathbf{S}^{n\times n}$. This formula shows that the key to constructing H fast, is to compute the two matrices H_1 and H_2 fast. In this section we discuss one method for doing this, based on the eigenvalue decomposition of A. Our assumption that (A, B) is controllable implies that it is always possible to find a linear state feedback matrix K so that $A + BK$ is stable and diagonalizable [KND85]. As mentioned in §3 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 above can be expressed as

$$\begin{array}{rcl} H_1 & = & 2\Re\Big(V^{-T}((\widetilde{\Sigma}\widetilde{W}_{11}) \odot (\widetilde{\Sigma}\widetilde{W}_{11})^T)V^{-1} \\ & & +V^{-*}(\widetilde{W}_{11} \odot (\widetilde{\Sigma}\widetilde{W}_{11}\widetilde{\Sigma}^*)^T)V^{-1}\Big) \end{array} \qquad (22)$$

$$H_2 = -V(\widetilde{\Sigma}^*\,\mathbf{diag}(\widetilde{W}_{12})^*)\bar{V}^{-1} - V\,\mathbf{diag}(\widetilde{\Sigma}\widetilde{W}_{12})V^{-1} \qquad (23)$$

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

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

$\widetilde{\Sigma} = \Sigma\,\mathbf{diag}(V^{-1}B)^*$, $\widetilde{W}_{11} = V^* W_{11} V$, and $\widetilde{W}_{12} = V^* W_{12}$. The above formulas for H and G can be evaluated in $O(n^3)$ operations, and do not require precomputing the basis matrices X_i.

5 Numerical examples

Table 1 compares two methods, applied to randomly generated KYP-SDPs of the form (12), with dimensions n = 25, 50, 100, 200, and $p = n$. Each problem was constructed so it is strictly primal and dual feasible. The execution times listed are the CPU times in seconds on a 2GHz Pentium IV PC with 1GB of memory.

	SeDuMi (primal)		SeDuMi (dual)		
n	#itrs	time/itr	prepr.	#itrs	time/itr
25	10	0.1	0.1	12	0.02
50	11	7.4	1.2	11	0.2
100	11	324.7	22.5	12	3.0
200			436.2	13	26.5

Table 1: Comparison of primal and dual method for solving KYP-SDPs with dimensions $n = p = 25, \ldots, 200$, using the general purpose solver SeDuMi.

	KYP IPM			SeDuMi (dual)		
n	prepr.	#itrs	time/itr	prepr.	#itrs	time/itr
100	1.3	9	1.2	0.7	14	1.4
200	10.1	9	8.9	3.8	15	23.2
300	32.4	10	27.3	11.9	20	127.3
400	72.2	9	62.0			
500	140.4	10	119.4			

Table 2: Comparison of fast implementation of primal-dual interior-point algorithm and general-purpose code applied to the dual problem, for five KYP-SDPs of dimension $n = 100, \ldots, n = 500$, and $p = 50$.

The first method, labeled SeDuMi (primal), solves the SDP (12) using SeDuMi version 1.05 [Stu01] via the YALMIP interface [Löf02]. We list the number of iterations ('#itrs') and the time per iteration in seconds. In the second method, labeled SeDuMi (dual), we solve the reformulated dual problem (16). Column 4 gives the time required for this preprocessing step (*i.e.*, the solution of the $n + 1$ Lyapunov equations). The other columns are the number of iterations, and time per iteration. The results confirm that the cost per iteration of a general-purpose method applied to the primal SDP (12) grows much more rapidly than the dual method.

Table 2 shows the results for a second experiment with randomly generated KYP-SDPs of dimensions $n = 100$, ..., 500, and $p = 50$. The data in the column labeled 'KYP IPM' are for a customized Matlab implementation of the primal-dual path-following method of Tütüncü, Toh, and Todd [TTT98], applied to the dual problem, and using the expressions (22) and (23) to compute the coefficient matrix of the reduced search equations. The pre-processing time for this method includes the eigenvalue decomposition of A and the computation of the matrix G in the reduced system (19). The pre-processing time and execution time per iteration grow almost exactly as n^3.

For the first three problems, we also show the results for SeDuMi applied to the dual problem. To speed up the calculation, we first transform the dual problem (16), by diagonalizing A. This corresponds to a simple change of variables, resulting in an SDP of the form (16), 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.

6 Conclusion

We have discussed techniques for avoiding the $O(n^6)$ growth of the execution time required by general-purpose SDP solvers applied to KYP-SDPs. A first approach is to reformulate the dual problem as an SDP with $n+1$ variables, and to solve the reformulated dual via a general-purpose solver. This simple method has a complexity of $O(n^4)$ flops per iteration. A second approach is to apply a standard primal-dual algorithm to the original or the reformulated KYP-SDP, and exploit problem structure to compute the primal and dual search directions fast. This results in a complexity of $O(n^3)$ per iteration.

Some topics that deserve further study are the the possible use of other canonical forms (for example, companion form), and the influence of transformations of A and B (for example, choice of state feedback matrix) on the accuracy and robustness of the algorithms.

7 Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. ECS-0200320 and the Swedish Research Council under Grant No. 271-2000-770.

References

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

[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.

[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.

[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.

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

[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 Conference*, pages 3019–3024, 2001.

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

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

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

[Par00] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, California Institute of Technology, 2000.

[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`.

[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.

[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.