Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

A tailored inexact interior-point method for systems analysis

Janne Harju Johansson
Department of Electrical Engineering
Linköping University
581 83 Linköping, Sweden
harju@isy.liu.se

Anders Hansson
Department of Electrical Engineering
Linköping University
581 83 Linköping, Sweden
hansson@isy.liu.se

Abstract— **Within the area of systems analysis there are several problem formulations that can be rewritten as semidefinite programs. Increasing demand on computational efficiency and ability to solve large scale problems make the available generic solvers inadequate. In this paper structure knowledge is utilized to derive tailored calculations and to incorporate adaptation to the different properties that appear in a proposed inexact interior-point method.**

I. INTRODUCTION

In this paper a structured semidefinite programming (SDP) problem is defined and a tailored algorithm is proposed and evaluated. The problem formulation, can for example be applied to analysis of polytopic linear differential inclusions (LDIs). The reformulation from systems analysis problems to SDPs is described in [9] and [17].

The software packages available to solve SDP problems are numerous. For example, if YALMIP, [27], is used as an interface, nine available solvers can be applied. Some examples of solvers are SDPT3, [33], DSDP, [2] and SeDuMi, [32], [29] all of which are interior-point methods. These solvers solve the optimization problem on a general form. The problem size will increase with the number of constraints and the number of matrix variables. Hence, for large scale problems, generic solvers will not have an acceptable solution time or terminate within an acceptable number of function calls. It is necessary to utilize the problem structure to speed up the performance. Here an algorithm is described that uses inexact serch directions for an infeasible interior-point method. A memory efficient iterative solver is used to compute the search directions in each step of the interior-point method. In each step of the algorithm, the error tolerance for the iterative solver decreases, and hence the initial steps are less expensive to calculate than the last ones.

Iterative solvers for linear systems of equations are well studied in the literature. For applications to optimization and preconditioning for interior-point methods see [12], [7], [11] [5], [19], [22] and [35]. Here a SDP problem is studied and hence are the algorithms in [12], [7], [11] and [5] not applicable. In [19], [22] and [35] a potential reduction method is applied and an iterative solver for the search directions is used. In [19] a feasible interior-point method is used and hence the inexact solutions to the search direction equations need to be projected into the feasible space at a high cost. In [22] and [35] this was circumvented by solving one linear system of equations for the primal search direction and another linear system of equations for the dual search direction, however also at a higher computational cost. Furthermore, solving the normal equations in [19] resulted in an increasing number of iterations in the iterative solver when tending towards the optimum. In this paper the augmented equations are solved, which results in an indefinite linear system of equations. No increase in the number of iterations in the iterative solver has been observed. The behavior of constant number of iterations in the iterative solver has also been observed in [21] and [11]. In [12] the same behavior was noted for linear programming. There the augmented equations are solved when the iterate is close to the optimum.

A problem similar to the one discussed in this paper has been investigated in [34], [36] and [19]. However, the problem classes do not coincide since the constraints in this work share the matrix variable P, defined later in the paper.

In [25] some preliminary results were presented. However, it was noted that the convergence of the iterative solver was only satisfactory initially in the algorithm and hence further work was needed to cover a larger class of problems. The two stage method described in this paper overcomes this problem in many cases.

The remaining part of the paper is organized as follows. First the optimization problem is formulated and some mathematical preliminaries are presented. Then a brief discussion of optimality conditions and the inexact interior-point method is presented. When the overall algorithm is defined the equations to find the search directions are given and the solution of that linear system of equations is discussed. A new preconditioner is suggested and described in detail. Finally some computational results are presented where the proposed algorithm is compared to the SDPT3 solver.

II. PROBLEM FORMULATION

Denote the space of symmetric matrices of size n as $\mathbb{S}^n$. The optimization problem to be solved is

$$\begin{aligned} &\min\ c^T x + \langle C, P\rangle \\ &\text{s.t. } \mathcal{F}_i(P) + \mathcal{G}_i(x) + M_{i,0} = S_i,\ i = 1,\dots,n_i \\ &S_i \succeq 0 \end{aligned} \tag{1}$$

where the decision variables are $S_i \in \mathbb{S}^{n+m}$, $P \in \mathbb{S}^n$ and $x \in \mathbb{R}^{n_x}$,

$$\mathcal{F}_i(P) = \begin{bmatrix} \mathcal{L}_i(P) & PB_i \\ B_i^T P & 0 \end{bmatrix} = \begin{bmatrix} A_i^T P + PA_i & PB_i \\ B_i^T P & 0 \end{bmatrix} \tag{2}$$

This work was supported by CENIIT.

and

$$\mathcal{G}_i(x) = \sum_{k=1}^{n_x} x_k M_{i,k}, \tag{3}$$

with $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m}$, $C \in \mathbb{S}^n$ and $M_{i,k} \in \mathbb{S}^{n+m}$. The inner product $\langle C, P\rangle$ is Trace(CP), and $\mathcal{L}_i : \mathbb{S}^n \to \mathbb{S}^n$ is the Lyapunov operator $\mathcal{L}_i(P) = A_i^T P + P A_i$ with adjoint $\mathcal{L}_i^*(X) = A_i X + X A_i^T$. Furthermore, the adjoint operators of $\mathcal{F}_i$ and $\mathcal{G}_i$ are

$$\mathcal{F}_i^*(Z_i) = \begin{bmatrix} A_i & B_i \end{bmatrix} Z_i \begin{bmatrix} I_n \\ 0 \end{bmatrix} + \begin{bmatrix} I_n & 0 \end{bmatrix} Z_i \begin{bmatrix} A_i^T \\ B_i^T \end{bmatrix} \tag{4}$$

and

$$\mathcal{G}_i^*(Z_i)_k = \langle M_{i,k}, Z_i\rangle,\ k = 1, \ldots, n_x \tag{5}$$

respectively, where $Z_i \in \mathbb{S}^{n+m}$.

When we study (1) on a higher level of abstraction the operator $\mathcal{A}(P,x) = \oplus_{i=1}^{n_i}(\mathcal{F}_i(P) + \mathcal{G}_i(x))$ is used. Its adjoint is $\mathcal{A}^*(Z) = \sum_{i=1}^{n_i}(\mathcal{F}_i^*(Z_i), \mathcal{G}_i^*(Z_i))$ where $Z = \oplus_{i=1}^{n_i} Z_i$. Also define $S = \oplus_{i=1}^{n_i} S_i$ and $M_0 = \oplus_{i=1}^{n_i} M_{i,0}$.

For later use we define $z = (x, P, S, Z)$ and the corresponding finite-dimensional vector space $\mathcal{Z} = \mathbb{R}^{n_x} \times \mathbb{S}^{n+m} \times \mathbb{S}^{n+m} \times \mathbb{S}^{n+m}$ with its inner product $\langle \cdot, \cdot \rangle_{\mathcal{Z}}$.

Throughout the paper it is assumed that the mapping $\mathcal{A}$ has full rank.

III. Inexact interior-point method

In this work a primal-dual interior-point method is implemented. For such algorithms the primal and dual problems are solved simultaneously. The primal and dual for (1) with the higher level of notation are

$$\begin{aligned} \min\ & c^T x + \langle C, P\rangle \\ \text{s.t}\ & \mathcal{A}(P,x) + M_0 = S \\ & S \succeq 0 \end{aligned} \tag{6}$$

and

$$\begin{aligned} \max\ & -\langle M_0, Z\rangle \\ \text{s.t}\ & \mathcal{A}^*(Z) = (C, c) \\ & Z \succeq 0 \end{aligned} \tag{7}$$

respectively. If strong duality holds, the Karush-Kuhn-Tucker conditions defines the solution to the primal and dual optimization problems, [10]. The Karush-Kuhn-Tucker conditions for the optimization problems in (6) and (7) are

$$\mathcal{A}(P,x) + M_0 = S \tag{8}$$

$$\mathcal{A}^*(Z) = (C, c) \tag{9}$$

$$ZS = 0 \tag{10}$$

$$S \succeq 0,\ Z \succeq 0 \tag{11}$$

For later use, define the complementary slackness ν as

$$\nu = \frac{\langle Z, S\rangle}{n} \tag{12}$$

To derive the equations for the search directions in the next iterate, $z^+ = z + \Delta z$ is defined and inserted into (8)–(11). Then a linearization of these equations is made. In order to obtain a symmetric update of the matrix variables we introduce the symmetrization operator $\mathcal{H} : \mathbb{R}^{n\times n} \to \mathbb{S}^n$ that is defined as

$$\mathcal{H}(X) = \frac{1}{2}\big(R^{-1} X R + (R^{-1} X R)^T\big) \tag{13}$$

where $R \in \mathbb{R}^{n\times n}$ is a so called scaling matrix. For a thorough description of scaling matrices, see [37] and [38]. The described procedure results in a linear system of equations for the search directions

$$\mathcal{A}(\Delta P, \Delta x) - \Delta S = -(\mathcal{A}(P,x) + M_0 - S) \tag{14}$$

$$\mathcal{A}^*(\Delta Z) = (C, c) - \mathcal{A}^*(Z) \tag{15}$$

$$\mathcal{H}(\Delta Z S + Z \Delta S) = \sigma\nu I - \mathcal{H}(ZS) \tag{16}$$

It is known that if the operator $\mathcal{A}$ has full rank, $Z \succ 0$ and $S \succ 0$, then the linear system of equations in (14)–(16) has a unique solution. See Theorem 10.2.2 in [37] for details.

Now we are ready to define the algorithm. The algorithm is based on a set Ω defined as

$$\begin{aligned} \Omega = \{ z = (x, P, S, Z) \mid\ & S \succeq 0, Z \succeq 0, \\ & \|\mathcal{A}(P,x) + M_0 - S\|_2 \le \beta\nu, \\ & \|\mathcal{A}^*(Z) - (C, c)\|_2 \le \beta\nu, \\ & \gamma\nu I \succeq \mathcal{H}(ZS) \succeq \eta\nu I \} \end{aligned} \tag{17}$$

where the scalars β, γ and η will be defined later on. Below the overall algorithm, which is taken from [30], is summarized, and adapted to semidefinite programming.

Algorithm: Interior-point method

0. Initialize the counter $j = 1$ and choose $0 < \eta < \eta_{max} < 1$, $\gamma \ge n$, $\beta > 0$, $\kappa \in (0,1)$, $0 < \sigma_{min} < \sigma_{max} < 1/2$, $\epsilon > 0$, $0 < \chi < 1$ and $z^0 \in \Omega$.
1. Evaluate stopping criteria. If fulfilled, terminate the algorithm.
2. Choose $\sigma \in (\sigma_{min}, \sigma_{max})$.
3. Compute the scaling matrix R.
4. Solve (14)–(16) for search direction Δz^j with a residual tolerance $\epsilon\sigma\beta\nu/2$.
5. Choose a step length α^j as the first element in the sequence $\{1, \chi, \chi^2, \ldots\}$ such that $z^{j+1} = z^j + \alpha^j \Delta z^j \in \Omega$ and such that $\nu^{j+1} \le \big(1 - \alpha\kappa(1-\sigma)\big)\nu^j$.
6. Update the variables, $z^{j+1} = z^j + \alpha^j \Delta z^j$ and the counter $j := j + 1$.
7. Return to step 1.

Note that any iterate generated by the algorithm is in Ω, which is a closed set, since it is defined as an intersection of closed sets, see [23].

Convergence

For a detailed description and a convergence proof, see [23].

IV. Search directions

It is the solution of (14)–(16), which is performed in step 4 of the algorithm, that requires the most effort in an interior-point method. In order to study (14)–(16) in more detail

rewrite them as

$$W_i \Delta Z_i W_i + \mathcal{F}_i(\Delta P) + \mathcal{G}_i(\Delta x) = D_{1,i}, \ \forall i \tag{18}$$

$$\sum_{i=1}^{n_i} \mathcal{F}_i^*(\Delta Z_i) = D_2 \tag{19}$$

$$\sum_{i=1}^{n_i} \mathcal{G}_i^*(\Delta Z_i) = D_3 \tag{20}$$

where $W_i = R_i R_i^T \in \mathbb{S}^n$. In this work W_i are the Nesterov-Todd (NT) directions. For details on the NT scaling matrix see [28]. Note that the linear system of equations (18)–(20) is indefinite.

V. Iterative solver

The number of available algorithms to solve a linear system of equations with an iterative solver is large. Choosing solver is highly problem dependent. Properties such as definite/indefinite coefficient matrix, Hermitian or non-Hermitian coefficient matrix determine which algorithm is applicable. Additionally the choice of preconditioner will affect what algorithm that is to be used. In [20] algorithms are explained and studied in detail and in [1] the implementational details are discussed.

In the described problem an indefinite system is to be solved, hence algorithms for indefinite systems will be the main focus. Additionally the preconditioner will be indefinite, which restricts the choice even further. Examples of iterative solvers that handle an indefinite coefficient matrix and an indefinite preconditioner are the bi-conjugate gradient method and its stabilized version (BiCG and BiCGstab), the quasi minimal residual (QMR) method, and various versions of the generalized minimal residual (GMRES) method.

Here the symmetric quasi-minimal residual method (SQMR) is chosen. SQMR is the only solver that utilizes that the coefficient matrix is symmetric. Another positive property is that SQMR does not require as much storage as the theoretically optimal GMRES solver. An undesired property is that the residual is not included in the algorithm. Hence, it must be calculated if a guaranteed residual is required from the iterative solver.

In [15] and [16] the original SQMR algorithm description is presented. To simplify the description, we rewrite (18)–(20) as $\mathcal{B}(\Delta z) = b$ and denote the invertible preconditioner $\mathcal{P}(\Delta z) = p$. The described algorithm is SQMR without look-ahead for the linear system of equations using operator formalism.

Algorithm: SQMR

0. Choose $\Delta z_0 \in \mathcal{Z}$ and preconditioner $\mathcal{P}(\cdot)$. Then set $r_0 = b - \mathcal{B}(z_0)$, $t = r_0$, $\tau_0 = \|t\|_2 = \sqrt{\langle r_0, r_0 \rangle}$, $q_0 = \mathcal{P}^{-1}(r_0)$, $\vartheta_0 = 0$, $\rho_0 = \langle r_0, q_0 \rangle$, and $d_0 = 0$.

For $j = 1, 2, \ldots$

1. Compute $t = \mathcal{B}(q_{j-1})$, $v_{j-1} = \langle q_{j-1}, t \rangle$.
   if $v_{j-1} = 0$, **then** Terminate
   else
   $\alpha_{j-1} = \frac{\rho_{j-1}}{v_{j-1}}$ and $r_j = r_{j-1} - \alpha_{j-1} t$
   end
2. Set $t = r_j$, $\vartheta_j = \|t\|_2 / \tau_{j-1}$, $c_j = 1/\sqrt{1 + \vartheta_j^2}$, $\tau_j = \tau_{j-1} \vartheta_j c_j$, $d_j = c_j^2 \vartheta_{j-1}^2 d_{j-1} + c_j^2 \alpha_{j-1} q_{j-1}$ and $\Delta z_j = \Delta z_{j-1} + d_j$.
   if Δz_j has converged, **then** Terminate
   end item[3.] **if** $\rho_{j-1} = 0$, **then** Terminate
   else
   $u_j = \mathcal{P}^{-1}(t)$, $\rho_j = \langle r_j, u_j \rangle$, $\beta_j = \frac{\rho_j}{\rho_{j-1}}$, and $q_j = u_j + \beta_j q_{j-1}$.

Here b, p, r, t, q, $d \in \mathcal{Z}$ and τ, ϑ, ρ, v, α, $c \in \mathbb{R}$.

VI. Preconditioning

The construction of a good preconditioner is highly problem dependent. A preconditioner should reflect the main properties of the original equation system and still be inexpensive to evaluate. There is a wide variety of preconditioners in the literature. In [4] the general class of saddle point problems are studied and some preconditioners are discussed. Here only the preconditioners applicable to an indefinite system of equations are discussed.

There are many strategies to approximate the linear system of equations to obtain a preconditioner. A popular choice is to approximate the symmetric and positive definite $(1,1)$-block of the coefficient matrix with some less complicated structure. Common approximations are to use a diagonal matrix or a block-diagonal matrix. A collection of such methods can be found in [7], [14], [13], [5] and [26]. The preconditioner used in the initial stage of the defined algorithm uses this approximation.

Another strategy of preconditioning is to replace the coefficient matrix with a non-symmetric approximation that is easier to solve, as described in [3] and [8].

Finally, incomplete factorizations can be used. This is recommendable especially for sparse matrices, see [31] for further details.

In this work a two phase algorithm is described. The two separate phases are due to the change of properties when the iterates tend toward the optimum. The use of two separate preconditioners have previously been applied to linear programming problems in [12] and [6].

Preconditioner I

This preconditioner is based on the assumption that the W_i matrices can be described by a scalar value, $W_i = w_i \cdot I_{n+m}$, $\forall i$ and that the constraints are closely related $\mathcal{F}_i \approx \bar{\mathcal{F}}$, $\forall i$ and $\mathcal{G}_i \approx \bar{\mathcal{G}}$, $\forall i$. This results in a preconditioner that can be condensed to solving a linear system of equations of the same size as if there were only one constraint in the optimization problem with a simple scaling matrix. For a thorough description and simulation results, see [25], where it was noted that the described assumption is only valid in the initial steps of the algorithm. An explanation is that when the iterates tend to the boundary of the feasible region the eigenvalues of W_i for the active constraint are not close to each other. A clustering of the eigenvalues into two clusters has been noted. Hence, the assumption that W_i can be described by a scalar value is not valid. Similar behaviour has been noted in [18]. However, when the assumption is valid the preconditioner is much faster than solving the original

system of equations. Thus it is used as a preconditioner for the initial phase of the algorithm.

Preconditioner II

The inspiration to Preconditioner II is found in [18]. In that work the analysis does not consider block structure in the coefficient matrix. Furthermore, the problem is reformulated to obtain a definite coefficient matrix since the chosen solver requires a definite preconditioner. Here we will construct an indefinite preconditioner by identifing the constraints that indicate large eigenvalues for the scaling matrices and look at the linear system of equations on a block structure introduced by the constraints to construct an indefinite preconditioner.

First define the symmetric vectorization operator $\mathbf{svec}(X) = (X_{11}, \sqrt{2}X_{12}, \ldots, X_{22}, \sqrt{2}X_{23}, \ldots)^T$. The **svec** operator yield a symmetric coefficient matrix when applied to (18)–(20). For notational convenience define

$$\mathbf{D}_{vec} = \begin{pmatrix} \mathbf{svec}(D_{1,1}) \\ \vdots \\ \mathbf{svec}(D_{1,n_i}) \\ \mathbf{svec}(D_2) \\ D_3 \end{pmatrix} \text{ and } \mathbf{\Delta} = \begin{pmatrix} \mathbf{svec}(\Delta Z_1) \\ \vdots \\ \mathbf{svec}(\Delta Z_{n_i}) \\ \mathbf{svec}(\Delta P) \\ \Delta x \end{pmatrix}.$$

To illustrate how Preconditioner II works, the vectorized version of (18)–(20) is studied. The linear system of equations for the search directions in a vectorized form can be written as

$$\begin{pmatrix} H_1 & & & F_1 & G_1 \\ & \ddots & & \vdots & \vdots \\ & & H_{n_i} & F_{n_i} & G_{n_i} \\ F_1^T & \ldots & F_{n_i}^T & & \\ G_1^T & \ldots & G_{n_i}^T & & \end{pmatrix} \mathbf{\Delta} = \mathbf{D}_{vec} \quad (21)$$

where H_i, F_i and G_i denote appropriate submatrices.

To simplify the expressions in this section, define

$$N_i = \begin{pmatrix} F_i & G_i \end{pmatrix} \quad (22)$$

Simple matrix manipulations give the solution of (21) as

$$\begin{pmatrix} \mathbf{svec}(\Delta P) \\ \Delta x \end{pmatrix} = \left(\sum_i N_i^T H_i^{-1} N_i \right)^{-1} \times \left(\sum_i N_i^T H_i^{-1} \mathbf{svec}(D_{1,i}) - \begin{pmatrix} \mathbf{svec}(D_2) \\ D_3 \end{pmatrix} \right) \quad (23)$$

and

$$\mathbf{svec}(Z_i) = H_i^{-1} \left(\mathbf{svec}(D_{1,i}) - N_i \begin{pmatrix} \mathbf{svec}(\Delta P) \\ \Delta x \end{pmatrix} \right) \quad (24)$$

It has been observed in simulations that the eigenvalues of W_i grow large when the iterates tend towards the optimum. This implies that the eigenvalues of H_i grow large and hence will $H_i^{-1} N_i \approx 0$. F_i and G_i do not change during the iterations and have elementvise moderate values which give the result.

To derive the preconditioner, assume that $H_i^{-1} N_i \approx 0$ is valid for all $i \neq s$. As an intermediate result, note that

$$\sum_i N_i^T H_i^{-1} N_i \approx N_s^T H_s^{-1} N_s \quad (25)$$

Then the approximate solution is

$$\begin{pmatrix} \mathbf{svec}(\Delta P) \\ \Delta x \end{pmatrix} = \left(N_s^T H_s^{-1} N_s \right)^{-1} \times \left(N_s^T H_s^{-1} \mathbf{svec}(D_{1,s}) - \begin{pmatrix} \mathbf{svec}(D_2) \\ D_3 \end{pmatrix} \right) \quad (26)$$

and

$$\mathbf{svec}(Z_i) = \begin{cases} H_i^{-1} \mathbf{svec}(D_{1,i}), & i \neq s \\ H_i^{-1} \left(\mathbf{svec}(D_{1,i}) - N_i \begin{pmatrix} \mathbf{svec}(\Delta P) \\ \Delta x \end{pmatrix} \right), & i = s \end{cases} \quad (27)$$

This can be interpreted as the solution to an approximation of (21). Written on vectorized form, the approximative solution (26)–(27) is the solution to

$$\begin{pmatrix} H_1 & & & & & \\ & \ddots & & & & \\ & & H_s & & F_s & G_s \\ & & & \ddots & & \\ & & & & H_{n_i} & \\ & & F_s^T & & & \\ & & G_s^T & & & \end{pmatrix} \mathbf{\Delta} = \mathbf{D}_{vec} \quad (28)$$

This linear system of equations have a nice structure. ΔP, Δx and ΔZ_s can be found by solving a system of equations, as if we had a single constraint. The remaining dual variables ΔZ_i, $i \neq s$ are easily found by matrix inversions.

The constraint s is found by studying the eigenvalues of the W_i matrices. A large condition number indicates that the assumption is valid. This results in that the preconditioner solves

$$W_i \Delta Z_i W_i = D_{1,i},\ i \neq s \quad (29)$$

$$W_s \Delta Z_s W_s + \mathcal{F}_s(\Delta P) + \mathcal{G}_s(\Delta x) = D_{1,s} \quad (30)$$

$$\mathcal{F}_s^*(\Delta Z_s) = D_2 \quad (31)$$

$$\mathcal{G}_s^*(\Delta Z_s) = D_3 \quad (32)$$

The solution of (30)–(32) is a well studied problem. By using the results in [24], (30)–(32) can be solved at a total cost of $\mathcal{O}(n^3)$. Finally, the dual variables ΔZ_i in (29) are easily obtained by matrix inversions.

Algorithm: Preconditioner II

1. Identify the constraint with the smallest condition number and denote it by s.
2. Solve (30)–(32) to obtain ΔP, Δx and ΔZ_s.
3. Compute $\Delta Z_i = W_i^{-1} D_{1,i} W_i^{-1}$, $i \neq s$

VII. Computational Evaluation

All experiments are performed on a Dell Optiplex GX620 with 2GB RAM, Intel P4 640 (3,2GHz) CPU running under CentOS 4.1. Matlab version 7.4 (R2007a) is used with YALMIP version 3 (R20070810), [27], as interface to the solver. As comparison SDPT3 Version 4.0 (beta), [33], is used as underlying solver. Since the intention is to solve large scale optimization problems, the tolerance for termination is set to 10^{-3} for the relative and absolute residual. It is noted that both SDPT3 and the written solver terminate due to the

relative residual beeing below the desired tolerance in all the problems in this simulation study.

A comparison of absolute solution times is not always fair since the choice of implementation language is crucial. In SDPT3 the expensive calculations are implemented in C while the overview algorithm is written in Matlab. For the algorithm described and evaluated in this work the main algorithm, the iterative solver and Preconditioner I are written in Matlab. However the construction of the H matrix from basis matrices of low rank is implemented in C and that is the operation that requires the most computational effort. Obviously an implementation in C of the block-diagonalization and the low rank decomposition would improve the described solver. The similarity in the level of implementation in C makes a comparison in absolute computational time applicable.

The parameters in the algorithm are set to $\kappa = 0.01$, $\sigma_{max} = 0.9$, $\sigma_{min} = 0.01$, $\eta = 10^{-6}$, $\chi = 0.9$, $\epsilon = 10^{-8}$ and $\beta = 10^7 \cdot \beta_{lim}$ where $\beta_{lim} = \max(\|\mathcal{A}(P,x) + M_0 - S\|_2, \|\mathcal{A}^*(Z) - (C,c)\|_2)$. The choice of parameter values are based on knowledge obtained during the development of the algorithm and through continuous evaluation. Note that the choice of σ_{max} is not within the convergence proof given in [23], however the choice is motivated by faster convergence in practice and as good convergence as with $\sigma_{max} \leq 0.5$. This can be motivated by the fact that values close to zero and one correspond to predictor and corrector steps respectively.

Switching between the preconditioners is made after ten iterations. This is equivalent to five predictor-corrector steps. A more elaborate switching technique could improve the convergence but for practical use the result is satisfactory.

The only information that Preconditioner II is given is if the active constraint has changed. This information is obtained by the main algorithm.

To obtain the solution times, the Matlab command `cputime` is used. Input to the solvers are the system matrices, so any existing preprocessing of the problem is included in the total solution time.

In order to monitor the progress of the algorithm the residual for the search directions is calculated in each iteration in the iterative solver. If further improvement is desired one could use the in SQMR for free available Biconjugate Gradient (BCG) residual. However this results in that the exact residual is not known and hence the convergence might be affected.

Initialization

For comparable results, the initialization scheme given in [33] is used for the dual variables,

$$Z_i = \max\left(10, \sqrt{n+m}, \max_{k=1,\ldots,n_x} \frac{(n+m)(1+|c_k|)}{1+\|M_{i,k}\|_F}\right) I_{n+m}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The slack variables are chosen as $S_i = Z_i$ while the primal variables are initialized as $P = I_n$ and $x = \bar{0}$.

Examples

To evaluate the suggested inexact algorithm with an iterative equations solver, randomly generated optimization problems are solved. The procedure to generate the examples is described below. For the examples in this section all randomly generated matrices have a condition number of 10.

Algorithm: Generate example

1. Define the scalar value δ.
2. Generate the mean system matrices $\bar{A}$,$\bar{B}$ and $\bar{M}_k$ using `gallery.m`.
3. Generate the system matrices A_i, B_i and $M_{i,k}$ as $A_i = \bar{A} \pm \delta \cdot \Delta_A$, $B_i = \bar{B} \pm \delta \cdot \Delta_B$ and $M_{i,k} = \bar{M}_{i,k} \pm \delta \cdot \Delta_{M_{i,k}}$. The matrix Δ_A is a diagonal matrix where the diagonal is generated by `rand.m` while Δ_B and $\Delta_{M_{i,k}}$ are generated by `gallery.m`.
4. Define c and C such that a feasible optimization problem is obtained.

Results

To make an exhaustive investigation, different problem parameters have been investigated:

$$\begin{aligned} n &\in \{10, 16, 25, 35, 50, 75, 100, 130, 165\} \\ m &\in \{1, 3, 7\} \\ n_i &\in \{2, 5\} \\ \delta &\in \{0.01, 0.02, 0.05\} \end{aligned}$$

For each case there are 15 generated examples in order to find the average solution time. Naturally all the simulations cannot be given in detail. As an example the case $\delta = 0.02$, $n_i = 5$ and $m = 3$ with varying n is shown in Figure 1.

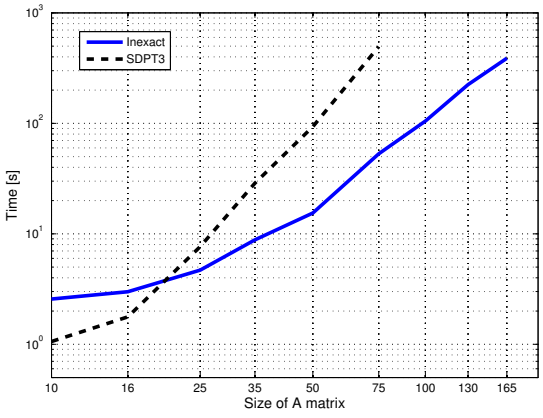

Fig. 1. Solution times for randomly generated problems. Here the problem parameters are set to $n_i = 5$ and $m = 3$. The solution times for SDPT3 and the inexact solver using two different preconditioners are plotted as a function of n.

First the properties of the SDPT3 solver is discussed. This solver is well tested and numerically stable. It solves all the problems generated up to $n = 75$. However, the solver does not solve the larger problems since the solution times tend to be unacceptable and for even larger problem the solver cannot proceed. When n and/or the number of constraints n_i is large, the solver will terminate due to memory restrictions. This motivate the use of inexact methods using an iterative solver since an iterative solver will require a substantial less amount of memory.

The suggested algorithm can solve large problems with much lower computational time required. A negative property that has been noted is that it will not converge on all the generated problems. Although, when convergence occurs the solver is always faster than SDPT3 for large problems, $n \geq 50$. For the 2420 generated problems 11% does not converge due to numerical problems. Naturally the inability to converge is not uniformly distributed. For $\delta = 0.01$ every problem is solved and the worst case is when $n = 165$, $n_i = 5$, $m = 7$ and $\delta = 0.05$ with a failure rate of 47%. This is natural since this example is the one where the assumptions made in Preconditioner II might not be valid. Best results are obtained for $n_i = 2$ with a failure rate of 8%.

VIII. Conclusions

A new preconditioner has been proposed to a primal-dual inexact interior-point method. The use of this preconditioner close to the optimum enables the solution of large scale problems. Although the algorithm does not converge due to numerical problems for some cases, several problems that are unsolvable with generic software are solved with the proposed algorithm. The results show that structure exploitation and the use of two separate preconditioners for the iterative solver gives an efficient algorithm.

References

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Philadalphia: Society for Industrial and Applied Mathematics., 1994.

[2] S. J. Benson and Y. Yinyu. DSDP5: Software for semidefinite programming. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, September 2005.

[3] M. Benzi and G. H. Golub. A preconditioner for generalized saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):20–41, 2004.

[4] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta numerica*, 14:1 – 137, 2005.

[5] L. Bergamaschi, J. Gondzio, and G. Zilli. Preconditioning indefinite systems in interior point methods for optimization. *Computational Optimization and Applications*, 28(2):149 – 171, 2004.

[6] S. Bocanegra, F. F. Campos, and A. R. L. Oliveira. Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods. *Computational Optimization and Applications*, 36(2-3):149–164, 2007.

[7] S. Bonettini and V. Ruggiero. Some iterative methods for the solution of a symmetric indefinite KKT system. *Computational Optimization and Applications*, 38(1):3 – 25, 2007.

[8] M. A. Botchev and G. H. Golub. A class of nonsymmetric preconditioners for saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1125–1149, 2006.

[9] S. Boyd, E. G. Laurent, E. Feron, and V. Balakrishnan. *Linear Matrix inequalities in System and Control Theory*. SIAM, 1994.

[10] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[11] S. Cafieri, M. D'Apuzzo, V. De Simone, and D. di Serafino. On the iterative solution of KKT systems in potential reduction software for large-scale quadratic problems. *Computational Optimization and Applications*, 38(1):27 – 45, 2007.

[12] J. S. Chai and K. C. Toh. Preconditioning and iterative solution of symmetric indefinite linear systems arising from interior point methods for linear programming. *Computational Optimization and Applications*, 36(2-3):221–247, 2007.

[13] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen. Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems. *SIAM Journal on Matrix Analysis and Applications*, 28(1):170–189, 2006.

[14] A. Forsgren, P. E. Gill, and J. D. Griffin. Iterative solution of augmented systems arising in interior methods. *SIAM Journal on Optimization*, 18(2):666–690, 2007.

[15] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-hermitian linear systems. *Numerische Mathematik*, 60(3):315 – 339, 1991.

[16] R. W. Freund and N. M. Nachtigal. A new Krylov-subspace method for symmetric indefinite linear systems. In *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, pages 1253–1256. IMACS, 1994.

[17] P. Gahinet, P. Apkarian, and M. Chilali. Parameter-dependent Lyapunov functions for real parametric uncertainty. *IEEE Transactions on Automatic Control*, 41(3):436 – 442, 1996.

[18] P. E. Gill, W. Murray, D. B. Ponceleón, and M. Saundeers. Preconditioners for indefinite systems arising in optimization. *SIAM journal on matrix analysis and applications*, 13(1):292 – 311, 1992.

[19] J. Gillberg and A. Hansson. Polynomial complexity for a Nesterov-Todd potential-reduction method with inexact search directions. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, page 6, Maui, Hawaii, USA, December 2003.

[20] A. Greenbaum. *Iterative methods for solving linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[21] A. Hansson. A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *Automatic Control, IEEE Transactions on*, 45(9):1639–1655, 2000.

[22] A. Hansson and L. Vandenberghe. A primal-dual potential reduction method for integral quadratic constraints. In *Proceedings of the American Control Conference*, pages 3013–3017, Arlington, Virginia, USA, 2001.

[23] J. Harju and A. Hansson. An inexact interior-point method, a description and convergence proof. Technical Report LiTH-ISY-R-2819, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, September 2007.

[24] J. Harju, R. Wallin, and A. Hansson. Utilizing low rank properties when solving KYP-SDPs. In *IEEE Conference on Decision and Control*, San Diego, USA, December 2006.

[25] J. Harju Johansson and A. Hansson. Structure exploitation in semidefinite programs for systems analysis. In *IFAC World Congress*, Seoul, Korea, July 2008. Accepted for publication.

[26] C. Keller, N. I. M. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300 – 1317, 2000.

[27] J. Löfberg. YALMIP : A toolbox for modeling and optimization in Matlab. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[28] Y. Nesterov and M. J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22(1):1 – 42, 1997.

[29] I. Pólik. Addendum to the SeDuMi user guide, version 1.1, 2005.

[30] D. Ralph and S. J. Wright. Superlinear convergence of an interior-point method for monotone variational inequalities. *Complementarity and Variational Problems: State of the Art*, 1997.

[31] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, USA, 1996.

[32] J. F. Sturm. Using SeDuMi 1.02, a matlab toolbox for optimization over symmetric cones, 2001.

[33] K. C. Toh, M. J. Todd, and R. H. Tütüncü. On the implementation and usage of SDPT3 — a Matlab software package for semidefinite-quadratic-linear programming, version 4.0. 2006.

[34] L. Vandenberghe, V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh. *Positive Polynomials in Control*, chapter Interior-point algorithms for semidefinite programming problems derived from the KYP lemma. Lecture Notes on Control and Information Sciences. Springer Verlag, 2005.

[35] L. Vandenberghe and S. Boyd. A primal-dual potential reduction method for problems involving matrix inequalities. *Mathematical Programming*, 69:205 – 236, 1995.

[36] R. Wallin and A. Hansson. KYPD: A solver for semidefinite programs derived from the Kalman-Yakubovich-Popov lemma. In *IEEE Conference on Computer Aided Control Systems Design*, Taipei, Taiwan, September 2004. IEEE.

[37] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*, volume 27 of *International series in operations research & management science*. KLUWER, 2000.

[38] Y. Zhang. On extending some primal–dual interior-point algorithms from linear programming to semidefinite programming. *SIAM Journal on Optimization*, 8(2):365–386, 1998.