A Quasi-Newton Interior Point Method for Low Order H-Infinity Controller Synthesis

Daniel Ankelhed, Anders Helmersson and Anders Hansson

Daniel Ankelhed, Anders Helmersson and Anders Hansson, A Quasi-Newton Interior Point Method for Low Order H-Infinity Controller Synthesis, 2011, IEEE Transactions on Automatic Control, (56), 6, 1462-1467.
http://dx.doi.org/10.1109/TAC.2011.2118930
Postprint available at: Linköping University Electronic Press
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-69894

A quasi-Newton interior point method for low order H-infinity controller synthesis

Daniel Ankelhed, Anders Helmersson, *Member, IEEE* and Anders Hansson, *Senior Member, IEEE*

***Abstract*—This paper proposes a method for low order H-infinity synthesis where the constraint on the order of the controller is formulated as a rational equation. The resulting nonconvex optimization problem is then solved by applying a quasi-Newton primal-dual interior point method.**

The proposed method is evaluated together with a well-known method from the literature. The results indicate that the proposed method has comparable performance and speed.

***Index Terms*—H-infinity synthesis, Linear Matrix Inequalities, rank constraints, rational constraints, interior point methods, quasi-Newton methods.**

I. Introduction

THE development of robust control theory emerged during the 80s and a contributory factor certainly was the fact that the robustness of Linear Quadratic Gaussian (LQG) controllers can be arbitrarily bad as reported in [1]. A few years later, in [2], an important step in the development towards a robust control theory was taken, where the concept of H_∞ theory was introduced. The H_∞ synthesis, which is an important tool when solving robust control problems, was a cumbersome problem to solve until a technique was presented in [3], which is based on solving two Riccati equations. Using this method, the robust design tools became much easier to use and gained popularity. Quite soon thereafter, linear matrix inequalities (LMIs) were found to be a suitable tool for solving these kinds of problems by using reformulations of the Riccati equations. Also related problems, such as gain scheduling synthesis, fit into the LMI framework, see e.g. [4]. In parallel to the theory for solving problems using LMIs, numerical methods for this purpose were being developed and made available.

Typical applications for robust control include systems that have high requirements for robustness to parameter variations and for disturbance rejection. The controllers that result from these algorithms are typically of very high order, which complicates implementation. However, if a constraint on the maximum order of the controller is set, that is lower than the order of the augmented plant, the problem is no longer convex and it is then relatively hard to solve. These problems become very complex, even when the order of the system to be controlled is low. This motivates the development of efficient algorithms that can solve these kinds of problems.

D. Ankelhed, A. Helmersson and A. Hansson are with Automatic Control, Department of Electrical Engineering, Linköping University, Sweden. Email: {ankelhed, andersh, hansson}@isy.liu.se, phone: +46-1328{2804, 1622, 1681} or fax: +46-13282622. The first author is the corresponding author. The authors would like to thank The Swedish research council for financial support under contract no. 60519401.

When evaluating the proposed method, we iterate through orders and performance. Even if we can use the method to find a controller of a given order directly, evaluating the trade-off between controller order and performance is natural from an engineering perspective. In this way we can find a controller which gives a good balance between complexity and performance, even if the computational effort during the design becomes higher.

This article is based on the work in [5], but the difference is that the proposed algorithm in this work uses an approximate quasi-Newton update of the Hessian instead of a regularized, exact Hessian. New numerical results are also included to demonstrate the improved performance of the new algorithm.

Denote with $\mathbb{S}^n$ the set of symmetric $n \times n$ matrices and with $\mathbb{R}^{m\times n}$ the set of real $m \times n$ matrices. The notation $A \succ 0$ ($A \succeq 0$) and $A \prec 0$ ($A \preceq 0$) means that A is a positive (semi)definite matrix and negative (semi)definite matrix, respectively. Also, denote the symmetric vectorization operator by svec and the symmetric Kronecker product by $\otimes_S$, as defined in [6].

II. Preliminaries

We begin by describing a linear system, G, with state vector, $x \in \mathbb{R}^{n_x}$. The input vector contains the disturbance signal, $w \in \mathbb{R}^{n_w}$, and the control signal, $u \in \mathbb{R}^{n_u}$. The output vector contains the measurement, $y \in \mathbb{R}^{n_y}$, and the performance measure, $z \in \mathbb{R}^{n_z}$. In terms of its system matrices, we can represent the linear system as

$$G : \begin{pmatrix} \dot{x} \\ \hline z \\ y \end{pmatrix} = \left(\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right) \begin{pmatrix} x \\ \hline w \\ u \end{pmatrix}, \quad (1)$$

where D_{22} is assumed to be zero, i.e., the system is strictly proper from u to y. If this is not the case, we can find a controller $\tilde{K}$ for the system where D_{22} is set to zero, and then construct the controller as $K = \tilde{K}(I + D_{22}\tilde{K})^{-1}$. Hence, there is no loss of generality in making this assumption. For simplicity, it is also assumed that the whole system is on minimal form, i.e., it is both observable and controllable. However, in order to find a controller, it is enough to assume detectability and stabilizability (non observable and non controllable modes are stable), but the formulae will become more complex in this case.

The linear controller is denoted K. It takes the system measurement, y, as input and the output vector is the control signal, u. The system matrices for the controller are defined by the equation

$$K : \begin{pmatrix} \dot{x}_K \\ u \end{pmatrix} = \begin{pmatrix} K_A & K_B \\ K_C & K_D \end{pmatrix} \begin{pmatrix} x_K \\ y \end{pmatrix}, \quad (2)$$

where $x_K \in \mathbb{R}^{n_k}$ is the state vector of the controller.

Lemma 1 (H_∞ controllers for continuous plants): The problem of finding a linear controller such that the closed loop system G_c is stable and such that $\|G_c\|_\infty < \gamma$, is solvable if and only if there exist $X, Y \in \mathbb{S}^{n_x} \succ 0$, which satisfy

$$\begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} XA + A^T X & XB_1 & C_1^T \\ B_1^T X & -\gamma I & D_{11}^T \\ C_1 & D_{11} & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_X & 0 \\ 0 & I \end{pmatrix}^T \prec 0 \tag{3a}$$

$$\begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} AY + YA^T & YC_1^T & B_1 \\ YC_1 & -\gamma I & D_{11} \\ B_1^T & D_{11}^T & -\gamma I \end{pmatrix} \begin{pmatrix} \mathcal{N}_Y & 0 \\ 0 & I \end{pmatrix}^T \prec 0 \tag{3b}$$

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \succeq 0, \quad \operatorname{rank}(XY - I) \le n_k. \tag{3c}$$

where $\mathcal{N}_X$ and $\mathcal{N}_Y$ denote any base of the null-spaces of $\begin{pmatrix} C_2 & D_{21} \end{pmatrix}$ and $\begin{pmatrix} B_2^T & D_{12}^T \end{pmatrix}$ respectively.

Proof: See [7]. ∎

III. REFORMULATIONS

It is desirable to replace the rank constraint in (3c) with a smooth function in order to be able to apply gradient methods for optimization. To do this, the following lemma is used.

Lemma 2: Assume that the inequality

$$\begin{pmatrix} X & I \\ I & Y \end{pmatrix} \succeq 0 \tag{4}$$

holds. Let

$$\det(\lambda I - (I - XY)) = \sum_{i=0}^{n_x} c_i(X,Y)\lambda^i = \\ = \lambda^{n_x} + c_{n_x-1}(X,Y)\lambda^{n_x-1} + \ldots + c_1(X,Y)\lambda + c_0(X,Y) \tag{5}$$

be the characteristic polynomial of $(I - XY)$, where the functions $c_i(X,Y)$ are its coefficients. Then the following statements are equivalent if $n_k < n_x$:

1) $\operatorname{rank}(XY - I) \le n_k$
2) $c_{n_x-n_k-1}(X,Y) = 0$

Additionally, all coefficients are non-negative, i.e.

$$c_i(X,Y) \ge 0, \ \forall i. \tag{6}$$

How to compute $c_i(X,Y)$ and their derivatives is explained in [8] where also additional properties of the coefficients are shown.

Proof: See [8]. ∎

Definition 1 (Half-vectorization): Let

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & & \vdots \\ \vdots & & \ddots & \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix}.$$

Then

$$\operatorname{vech}(X) = (x_{11}\ x_{21}\ \ldots\ x_{n1}\ x_{22}\ \ldots\ x_{n2}\ x_{33}\ \ldots\ x_{nn})^T,$$

i.e., vech stacks the columns of X from the principal diagonal downwards in a column vector. See [9] for properties and details.

After merging the three LMIs in (3) into one by placing them in a block-diagonal matrix, we can choose appropriate symmetric matrices A_i and C and use vech to rewrite (3) equivalently as

$$\sum_{i=1}^{m} A_i x_i \prec C \\ \bar{c}_{n_x-n_k-1}(x) = 0. \tag{7}$$

where $x = \begin{pmatrix} \operatorname{vech}(X) \\ \operatorname{vech}(Y) \end{pmatrix} \in \mathbb{R}^m$ and $m = n_x(n_x+1)$. Since we know from Lemma 2 that $\bar{c}_{n_x-n_k-1}(x) \ge 0$ for all feasible x we can formulate a related problem to (7) as

$$\min_x \quad \bar{c}_{n_x-n_k-1}(x) \tag{8a}$$

$$\sum_{i=1}^{n_x(n_x-1)} A_i x_i \prec C, \tag{8b}$$

and if the solution x^* to (8) is such that $\bar{c}_{n_x-n_k-1}(x^*) = 0$, this means that x^* satisfies (3). However, we have noticed that scaling the objective function by the next coefficient in the characteristic polynomial makes it numerically sounder, i.e., we solve

$$\min_x \quad \frac{\bar{c}_{n_x-n_k-1}(x)}{\bar{c}_{n_x-n_k}(x)} \tag{9a}$$

$$\sum_{i=1}^{n_x(n_x+1)} A_i x_i \prec C, \tag{9b}$$

instead of (8). If the denominator $\bar{c}_{n_x-n_k}(x)$ is zero then $c_{n_x-n_k-1}(x)$ must also be zero and the quotient (9a) is defined as zero. For more properties of the function $\bar{c}_{n_x-n_k-1}(x)/\bar{c}_{n_x-n_k}(x)$, see [8].

IV. SOLVING THE OPTIMIZATION PROBLEM

A. Karush-Kuhn-Tucker conditions

The method we will use to solve (9) is based on the primal-dual framework described in [10] and [6]. The nonlinear part is from the former and the theory for SDPs is from the latter. The approach is based on solving the Karush-Kuhn-Tucker (KKT) conditions, which applied to (9) gives us the following:

$$\frac{df(x)}{dx_i} + \langle A_i, Z \rangle = 0, \quad i = 1, \ldots, m \tag{10a}$$

$$C - \sum_{i=1}^{m} A_i x_i - S = 0 \tag{10b}$$

$$ZS = \nu I \tag{10c}$$

$$Z \succeq 0, \quad S \succeq 0, \tag{10d}$$

where $\nu = 0$, $m = n_x(n_x+1)$, $Z \in \mathbb{S}$ is a dual variable, $S \in \mathbb{S}$ is a slack variable and $f(x) = \bar{c}_{n_x-n_k-1}(x)/\bar{c}_{n_x-n_k}(x)$ is a nonconvex function. Note that these are only necessary and not sufficient conditions for an optimal solution of (9).

B. Matrix-vector form

By slightly extending the framework in [6] by including gradient and Hessian in the equations in order to be able to handle a nonconvex objective function we can calculate the search direction $(\Delta x, \Delta Z, \Delta S)$ by solving the following matrix-vector equation

$$\begin{pmatrix} H & \mathcal{A}^T & 0 \\ \mathcal{A} & 0 & \mathcal{I} \\ 0 & E & F \end{pmatrix} \begin{pmatrix} \Delta x \\ \mathrm{svec}(\Delta Z) \\ \mathrm{svec}(\Delta S) \end{pmatrix} = \begin{pmatrix} r_p \\ \mathrm{svec}(R_d) \\ \mathrm{svec}(R_c) \end{pmatrix}, \tag{11}$$

where $\mathcal{I}$ is the identity matrix of appropriate size and

$$\begin{aligned} &E = P \otimes_s P^{-T}S, \quad F = PZ \otimes_s P^{-T}, \\ &r_p = \nabla_x f(x) + \mathcal{A}^T Z, \quad R_d = C - S - \sum_{i=1}^{m} A_i x_i, \qquad (12) \\ &R_c = \sigma\mu I - \mathcal{H}_P(ZS), \end{aligned}$$

and $\mathcal{A} = [\mathrm{svec}(A_1), \ldots, \mathrm{svec}(A_m)]$. The matrix H is a positive definite quasi-Newton approximation of $\nabla^2_{xx} f(x)$ obtained by using damped BFGS updating, see e.g. [10]. Additionally, the symmetry transform

$$\mathcal{H}_P(M) = \frac{1}{2}(PMP^{-1} + (PMP^{-1})^T) \tag{13}$$

as defined in [11], where P is chosen as the Nesterov-Todd (NT) scaling matrix, see [6], because it is known to perform well in general. For other choices of P, see e.g. [12], [13].

C. Computing step lengths

Once a search direction $(\Delta x, \Delta Z, \Delta S)$ is found, the next step is to determine how far to go in that direction. Step lengths $0 < \alpha, \beta < 1$ are chosen such that

$$Z^{(k+1)} = Z^{(k)} + \alpha \Delta Z \succeq 0, \quad S^{(k+1)} = S^{(k)} + \beta \Delta S \succeq 0, \tag{14}$$

i.e., such that the positive definiteness is maintained for the symmetric variables Z and S.

V. Initial point calculation

The initial point calculation used for the primal-dual method is based on what is suggested in [14] and will be described below.

Assume that the matrices A_i and C are block-diagonal of the same structure, each consisting of L blocks or square matrices of dimensions $n_1, n_2, \ldots, n_L$. Let $A_i^{(j)}$ and $C^{(j)}$ denote the jth block of A_i and C, respectively. Then the initial iterates can be chosen as

$$\begin{aligned} &x^{(0)} = \bar{\mathbf{1}}, \\ &Z^{(0)} = \mathrm{blkdiag}(\xi_1 I_{n_1},\ \xi_2 I_{n_2},\ \ldots,\ \xi_L I_{n_L}), \qquad (15) \\ &S^{(0)} = \mathrm{blkdiag}(\eta_1 I_{n_1},\ \eta_2 I_{n_2},\ \ldots,\ \eta_L I_{n_L}), \end{aligned}$$

where $\bar{\mathbf{1}}$ is a column vector containing ones, I_{n_j} is the identity matrix with dimension n_j and

$$\begin{aligned} \xi_j &= n_j \max_{1\le i\le m} \frac{1 + |b_i|}{1 + \|A_i^{(j)}\|_F}, \\ \eta_j &= \frac{1 + \max[\max_i\{\|A_i^{(j)}\|_F\}, \|C^{(j)}\|_F]}{\sqrt{n_j}}, \end{aligned}$$

where b is referring to the linear objective function which in our case is chosen as

$$b = \begin{pmatrix} \mathrm{vech}(I_{n_x}) \\ \mathrm{vech}(I_{n_x}) \end{pmatrix} \tag{16}$$

such that

$$b^T x = \mathrm{tr}(X + Y), \tag{17}$$

i.e., we follow the heuristics for minimizing rank as suggested in [15]. The only difference from [14] is that the authors suggest $x^{(0)} = 0$, but our choice seems to work better in our applications. By multiplying the identity matrix I_{n_j} by the factors ξ_j and η_j for each j, the initial point has a better chance of having the same order of magnitude as an optimal solution of the SDP, according to [14].

VI. A Mehrotra-like method

Now we will describe the steps of an algorithm based on Mehrotra's algorithm, [16], though reformulated several times by other authors, e.g. by [6], [14].

A. The predictor step

Set $\sigma = 0$ in (12) and solve (11) and denote the solution $(\Delta x^{\mathrm{aff}}, \Delta Z^{\mathrm{aff}}, \Delta S^{\mathrm{aff}})$. This step is sometimes called the *predictor step* or *affine scaling step*. Then calculate step lengths $\alpha^{\mathrm{aff}}, \beta^{\mathrm{aff}}$ as done in (14). Define μ^{aff} as the duality measure obtained using this step, i.e.,

$$\mu^{\mathrm{aff}} = \langle Z + \alpha^{\mathrm{aff}} \Delta Z^{\mathrm{aff}}, S + \beta^{\mathrm{aff}} \Delta S^{\mathrm{aff}} \rangle / n. \tag{18}$$

The centering parameter σ is chosen according to the following heuristic, [14], which does not have a solid analytical justification, but appears to work well in practice:

$$\sigma = \left(\frac{\mu^{\mathrm{aff}}}{\mu}\right)^e, \tag{19}$$

where the exponent e is chosen as follows

$$e = \begin{cases} \max\left(1, 3\ \min(\alpha^{\mathrm{aff}}, \beta^{\mathrm{aff}})^2\right) & \text{if } \mu > 10^{-6}, \\ 1 & \text{if } \mu \le 10^{-6}. \end{cases} \tag{20}$$

The algorithm does not update the iterate with the predictor step. It is only used to calculate μ^{aff}, which is needed for the computation of the corrector step, to be described next.

B. The corrector step

The search direction is now calculated by solving (11) again but replacing R_c with

$$R_c = \sigma\mu^{\mathrm{aff}} I - \mathcal{H}_P(ZS) - \mathcal{H}_P(\Delta Z^{\mathrm{aff}} \Delta S^{\mathrm{aff}}), \tag{21}$$

where σ and μ^{aff} are calculated as in (18) and (19) respectively, with the pre-calculated predictor step $(\Delta Z^{\mathrm{aff}}, \Delta S^{\mathrm{aff}})$ in a *second-order correction*, [6], assuming the predictor step is a decent approximation of the corrector step.

C. The quasi-Newton (QN) algorithm

By summarizing the last few sections we can now state an algorithm that we from now on denote QN. Define the residual

$$r(x,Z,S,\sigma,\mu) = \begin{pmatrix} r_p(x,Z) \\ \text{svec}\left(R_d(S,x)\right) \\ \text{svec}\left(R_c(Z,S,\sigma,\mu)\right) \end{pmatrix} \tag{22}$$

with $\mu = 0$, where r_p, R_d and R_c are calculated as in (12). This expression will be used in a stopping criterion in the algorithm. If the algorithm finishes with $f(x) = 0$, we can recover the controller parameters. This procedure is explained in e.g. [7].

A quasi-Newton Primal-Dual algorithm (QN)

```
Given $\gamma$ and $n_k$, calculate initial values $(x^{(0)}, Z^{(0)}, S^{(0)})$ as in (15). Set $k := 0$, $k_{\max} := 100$, $r_{\text{tol}} := 10^{-5}$.
while $\|r(x^{(k)}, Z^{(k)}, S^{(k)}, 0, 0)\|_2 < r_{\text{tol}}$ do
    $k := k + 1$
    if $k > k_{\max}$, abort, too many iterations.
    Calculate duality measure $\mu := \frac{\langle Z^{(k)}, S^{(k)} \rangle}{n}$
    Predictor step:
    Set $(x, Z, S) := (x^{(k)}, Z^{(k)}, S^{(k)})$ and solve (11) for $(\Delta x^{\text{aff}}, \Delta Z^{\text{aff}}, \Delta S^{\text{aff}})$ with $\sigma := 0$.
    Calculate step lengths $\alpha^{\text{aff}}, \beta^{\text{aff}}$ using (14).
    Calculate duality measure $\mu^{\text{aff}}$ using (18).
    Set centering parameter to $\sigma := (\mu^{\text{aff}}/\mu)^e$ where $e$ is calculated as in (20).
    Corrector step:
    Solve (11) for $(\Delta x, \Delta Z, \Delta S)$ with $R_c$ as in (21).
    Calculate step lengths $\alpha, \beta$ using (14)
    Set $(x^{(k+1)}, Z^{(k+1)}, S^{(k+1)}) := (x + \beta\Delta x, Z + \alpha\Delta Z,$
        $S + \beta\Delta S)$
end while
```

As a final step, recover the controller parameters (K_A, K_B, K_C, K_D) as described in [7] and verify that the closed loop system is stable and that $\|G_c\|_\infty < \gamma$ holds true. These requirements should normally be satisfied, but if there are numerical problems or if the algorithm stopped due to too many iterations this might not hold true.

VII. Numerical evaluation

All experiments were performed on a Dell Optiplex GX620 with 2GB RAM, Intel P4 640 (3.2 GHz) CPU running under Windows XP using Matlab, version 7.4 (R2007a).

Evaluation of the methods was done using the benchmark problem library COMPl$_e$ib, see [17] and [18]. We have chosen a mix of different systems from this library with the number of states ranging from 4 to 24.

A. Evaluated methods

We have evaluated the suggested *quasi-Newton primal-dual method* (QN) and compared it with Hifoo version 2.0 (using the default mode and the fast mode option, respectively), see [19]. A number of controllers have been synthesized with different number of states and the achieved norms for the closed loop systems have been compared. Hifoo was chosen as a reference because it performs well and is easily obtainable from a website and runs in Matlab, where also the suggested method has been implemented.

B. The tests

First, the full order controller (nominal controller) was computed. In these computations γ is minimized, which is a convex problem since there is no rank constraint. This controller was computed using the Control System Toolbox in Matlab, using the `hinfsyn` command, with the `'lmi'` option. The minimized upper bound on the performance measure obtained using the nominal controller is denoted γ^* and the achieved closed loop performance is denoted $\|G_c^*\|_\infty$. Let us then define a vector of multipliers

$$\bar{\gamma} = \begin{pmatrix} 1 & 1.05 & 1.1 & 1.2 & 1.35 & 1.5 & 2 & 3 & 5 & 10 \end{pmatrix} \tag{23}$$

where each element in the vector $\bar{\gamma}$ refers to different degrees of relaxations of the performance requirement. An increase of the upper bound of the H$_\infty$ norm, γ, by more than a factor 10 is considered not to be of any interest here because the performance we sacrifice then is too much.

The quasi-Newton primal-dual method was applied in order to find reduced order controllers with $n_c \leq n_k < n_{k,\max}$ states, where n_c is the lower bound on the number of states of the controller required to stabilize the system and $n_{k,\max} = \min(10, n_x{-}1)$. Let $\gamma = \gamma^*\bar{\gamma}_n$, where $1 \leq n \leq 10$ is the index of the vector $\bar{\gamma}$ and $n = 1$ at start. This procedure is described by the following algorithm.

Algorithm for iterating through orders and performance

```
Calculate nominal controller and calculate $\gamma^* = \|G_c^*\|_\infty$.
Set $n := 1$, $n_k := \min(10, n_x - 1)$.
while $n_k > n_c$ and $n < 10$ do
    Apply the QN algorithm using $\gamma = \gamma^*\bar{\gamma}_n$
    if success, save controller, set $n_k := n_k - 1$
    else, set $n := n + 1$
    end if
end while
```

If the QN algorithm is successful, a controller of a lower order is being searched for. If not, the performance requirement is relaxed (n is increased). Note however that the above algortithm of iterating through orders and performance is just one way of evaluating the QN algorithm, and that no initial controller is needed to run the algorithm, just a value of γ.

The evaluation of Hifoo was performed as follows. For each system, controller of orders from $n_{k,\max}$ down to n_c was searched for. For each order, Hifoo was applied ten times and the closed loop H$_\infty$ norm and the computation time was stored for each run. The reason for applying it ten times was to reduce the variance in the results, since the controllers depend on stochastic elements due to randomized starting points.

The minimum and median H$_\infty$ norm that was achieved by Hifoo for each system and controller order (n_k) order was calculated. These are denoted *min* and *med.* The associated required computation time is sum of the required time for all ten systems, while for the median H_∞ norm the mean time of these ten runs are calculated.

The required time for the algorithms to run was computed using the command `cputime`. The H_∞ norm was computed using the command `norm(Gc,inf,1e-6)`, where `Gc` is the closed loop system and the third argument is the tolerance used in the calculations. Note that we take all the time needed into account, even QN runs that fail to find a controller.

C. Extensive study

We applied the QN algorithm, HIFOO default and fast modes to a total of 44 systems. These systems are `AC1-9,11,12,15-18, ROC1-10, NN11, REA3, CM1, EB1-4, HE6, HE7, JE3, AGS, BDT1, IH, CSE1, TG1, WEC1-3 and DLR1`. Some systems resulted in too complex matrix computations for the QN method to handle and no controller was therefore calculated for these systems. These are `AC10, AC13-14, JE2`. Note that we used the same tuning parameters in the QN algorithm for all systems in the study, i.e., no individual tuning for different systems was done.

For each system in this study we have calculated the win percent values and averaged them over all systems. In cases where the QN algorithm cannot find a controller but HIFOO can, it was assumed a win for the latter. If the difference in norm obtained by the different methods was less than 1% it was declared a draw, so that minor numerical issues is not a big factor in the results. Also, the controllers calculated by any of the methods that result in an unstable closed loop system were removed before calculating statistics.

The results from applying the algorithms on all 44 systems used in the evaluation can be seen in [20], but are summarized in Table I and Table II. We can see that HIFOO default mode is better than the QN algorithm in 61 % of the cases when the minimum H_∞ norm of the ten runs are chosen, but the required time is more than a factor of 17 of what is required by the QN algorithm. However, if we compare the median H_∞ norm of the ten runs instead, the number is only 55 % of the cases but the required time is now approximately 1.7 times more compared to the QN algorithm.

As for the fast mode option of HIFOO, the required time is about a quarter needed compared to the default mode, but the difference in the results is quite small. The best achieved norm is approximately the same but the median values are better.

To summarize the evaluation, we have drawn the following conclusions. The QN algorithm is very fast for low order systems, as we will see in the example in Section VII-D. Though for higher order systems with more than 10 states, HIFOO is faster for most systems. The difference between the default mode and the fast mode options of HIFOO is not very significant, but the variance in the results is less when using the default mode.

D. A case study, AC6

For the system AC6 with seven states ($n_x = 7$), the closed loop H_∞ norms are listed in Table III. The time required to compute the results needed was 218 s using the QN algorithm

TABLE I
SUMMARY FOR ALL SYSTEMS WHEN COMPARING QN AND HIFOO, DEFAULT MODE. MORE DETAILS CAN BE FOUND IN SECTION VII-C.

	QN	HIFOO, min	HIFOO, med
Average win, min	39 %	**61 %**	-
Average win, med	45 %	-	**55 %**
Time required	$\mathbf{8.25 \cdot 10^4}$ **s**	$1.44 \cdot 10^6$ s	$1.44 \cdot 10^5$ s

TABLE II
SUMMARY FOR ALL SYSTEMS WHEN COMPARING QN AND HIFOO, FAST MODE. MORE DETAILS CAN BE FOUND IN SECTION VII-C.

	QN	HIFOO, min	HIFOO, med
Average win, min	38 %	**62 %**	-
Average win, med	**50 %**	-	**50 %**
Time required	$8.25 \cdot 10^4\ s$	$3.45 \cdot 10^5$ s	$\mathbf{3.45 \cdot 10^4}$ **s**

and 50033 s using HIFOO in default mode, i.e., almost a factor 230. When using HIFOO in fast mode, the required time is 8221 s, i.e. almost a factor 38 compared to QN. Figure 1 and Figure 2 illustrate the achieved norms for the system AC6 using HIFOO default mode and fast mode, respectively.

In Table III we can see that the minimum values from HIFOO are lower than those from QN in 3 out of 7 cases, while ending in a draw in 3 cases resulting in a win ratio of 64 % for HIFOO and 36 % for QN. However, if we use the median values, the win ratio is only 43 % for HIFOO, but then the time required should be divided by 10, resulting in a factor ≈ 4 in time compared to QN. For HIFOO fast mode the H_∞ norms are higher in general as expected, but for $n_k = 2$ the norm is actually lower than what is achieved when using default mode, which is the result of the nonconvexity of the problem combined with randomized starting points.

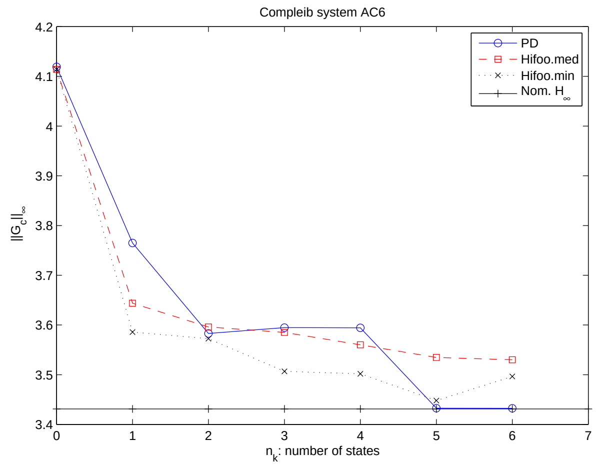

Fig. 1. The plot shows the closed loop H_∞ norm for the QN algorithm and HIFOO default mode, when applied to Compl$_e$ib system AC6.

VIII. CONCLUSIONS AND FUTURE WORK

A new algorithm for low order H_∞ controller synthesis has been proposed. The new algorithm has been evaluated and the results has been compared with HIFOO. The conclusion

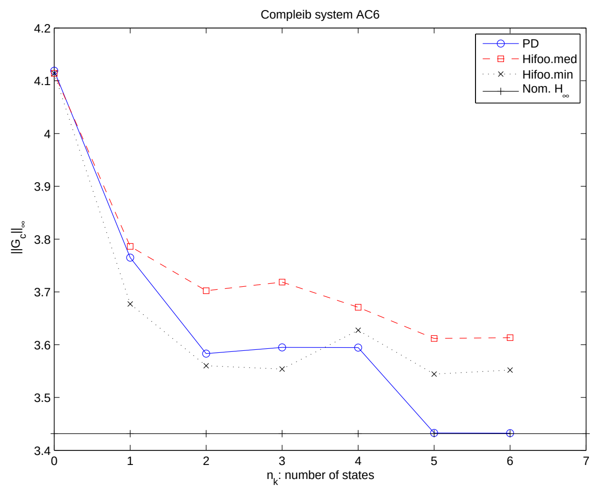

Fig. 2. The plot shows the closed loop H_∞ norm for the QN algorithm and HIFOO fast mode, when applied to Compl$_e$ib system AC6.

TABLE III
RESULTS FROM COMPUTING CONTROLLERS FOR THE COMPL$_e$IB SYSTEM AC6 WITH 7 STATES. THE BEST RESULTS (WITHIN AN INTERVAL OF 1 % OF THE BEST VALUE IF THERE ARE SEVERAL CLOSE TO EACH OTHER) FOR EACH CONTROLLER ORDER ARE MARKED WITH BOLD FONT. REQUIRED COMPUTATION TIME IS 218 S FOR QN, 50033 S FOR HIFOO DEFAULT MODE AND 8221 S FOR HIFOO FAST MODE.

		HIFOO, Default mode		HIFOO, Fast mode	
n_k	QN, $\|\|H\|\|_\infty$	$\|\|H\|\|_\infty^{med}$	$\|\|H\|\|_\infty^{min}$	$\|\|H\|\|_\infty^{med}$	$\|\|H\|\|_\infty^{min}$
6	**3.4325**	3.5301	3.4967	3.6132	3.5519
5	**3.4328**	3.5349	**3.4481**	3.6119	3.5444
4	3.5944	3.5602	**3.5019**	3.6709	3.6273
3	3.5948	3.5851	**3.5069**	3.7186	3.5539
2	**3.5831**	3.5960	**3.5725**	3.7023	**3.5601**
1	3.7649	3.6438	**3.5859**	3.7863	3.6772
0	**4.1189**	**4.1140**	**4.1140**	**4.1140**	**4.1140**

is that the proposed algorithm has comparable performance and speed, but HIFOO has an edge for higher order systems. For lower order systems ($\leq$ 10 states), e.g. AC6, the proposed algorithm is much faster than HIFOO. When trying to synthesize controllers for some high order systems ($>$ 20 states), it resulted in too big matrices for the proposed method to handle, while according to [21], HIFOO is able to find controllers for AC10 (55 states), which is one of these systems. How to handle systems with higher dimensions is something we are going to look into when developing the proposed method further. It would also be interesting to investigate if the calculation of the initial point can be done in a better way.

REFERENCES

[1] J. Doyle, "Guaranteed margins for LQG regulators," *IEEE Transactions on Automatic Control*, vol. 23, no. 4, pp. 756–757, 1978.

[2] G. Zames, "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses," *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 301–320, Apr 1981.

[3] J. Doyle, K. Glover, P. Khargonekar, and B. Francis, "State-space solutions to standard H_2 and H_∞ control problems," *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 831–47, 1989.

[4] A. Helmersson, "Methods for robust gain scheduling," Ph.D. dissertation, Linköping University, SE-581 83 Linköping, Sweden, Dec 1995.

[5] D. Ankelhed, A. Helmersson, and A. Hansson, "A primal-dual method for low order H_∞ controller synthesis," in *Proceedings of the 2009 IEEE Conference on Decision and Control*, Shanghai, China, Dec 2009.

[6] M. J. Todd, K. C. Toh, and R. H. Tütüncü, "On the Nesterov–Todd direction in semidefinite programming," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 769–796, 1998.

[7] P. Gahinet and P. Apkarian, "A Linear Matrix Inequality approach to H_∞ control," *International Journal of Robust and Nonlinear Control*, vol. 4, no. 4, pp. 421–48, 1994.

[8] A. Helmersson, "On polynomial coefficients and rank constraints," Department of Automatic Control, Linköping university, Sweden, Tech. Rep. LiTH-ISY-R-2878, 2009. [Online]. Available: http://www.control.isy.liu.se/publications/doc?id=2119

[9] H. Lütkepohl, *Handbook of Matrices*. John Wiley & Sons, Ltd, 1996.

[10] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[11] Y. Zhang, "On extending some primal–dual interior-point algorithms from linear programming to semidefinite programming," *SIAM Journal on Optimization*, vol. 8, no. 2, pp. 365–386, 1998.

[12] M. J. Todd, "A study of search directions in primal-dual interior-point methods for semidefinite programming," *Optimization methods and software*, vol. 11, no. 1-4, pp. 1–46, 1999.

[13] F. Alizadeh, J. Haeberly, and M. Overton, "Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 746–768, 1998.

[14] K. Toh, M. Todd, and R. Tütüncü, "SDPT3 a Matlab software package for semidefinite programming, version 1.3," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 545–581, 1999.

[15] M. Fazel, H. Hindi, and S. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *Proceedings of the American Control Conference*, Viginia, June 2001.

[16] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[17] F. Leibfritz, "COMPl$_e$ib: COnstrained Matrix optimization Problem library," 2006. [Online]. Available: http://www.complib.de

[18] ——, "COMPl$_e$ib: COnstraint Matrix optimization Problem library - a collection of test examples for nonlinear semidefinite programs, control system design and related problems." Department of Mathematics, Tech. Rep., 2004.

[19] S. Gumussoy, D. Henrion, M. Millstone, and M. Overton, "Multiobjective robust control with HIFOO 2.0," in *Proceedings of the IFAC Symposium on Robust Control Design*, Haifa, Israel, June 2009, pp. 144–149.

[20] D. Ankelhed, A. Helmersson, and A. Hansson, "Additional numerical results for the quasi-Newton interior point method for low order H-infinity controller synthesis," Department of Automatic Control, Linköping university, Sweden, Tech. Rep. LiTH-ISY-R-2964, 2010. [Online]. Available: http://www.control.isy.liu.se/publications/doc?id=2313

[21] S. Gumussoy and M. Overton, "Fixed-order H^∞ controller design via HIFOO, a specialized nonsmooth optimization package," in *Proceedings of the 2008 American Control Conference*, 2008, pp. 2750–2754.