INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL
*Int. J. Robust. Nonlinear Control* (2010)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/rnc.1577

# Controller coefficient truncation using Lyapunov performance certificate

Joëlle Skaf and Stephen P. Boyd*, †

*Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305, U.S.A.*

SUMMARY

We describe a method for truncating the coefficients of a linear controller while guaranteeing that a given set of relaxed performance constraints is met. Our method sequentially and greedily truncates individual coefficients, using a Lyapunov certificate, typically in linear matrix inequality (LMI) form, to guarantee the performance. Numerical examples show that the method is surprisingly effective at finding controllers with aggressively truncated coefficients, which meet typical performance constraints. We give an example showing that how the basic method can be extended to handle nonlinear plants and controllers. Copyright © 2010 John Wiley & Sons, Ltd.

Received 5 March 2009; Revised 29 October 2009; Accepted 18 January 2010

KEY WORDS: coefficient truncation; controller design; LMI methods

## 1. INTRODUCTION

### *1.1. The controller coefficient truncation problem*

We consider a discrete-time linear time-invariant control system, with plant

$$x_p(t+1) = A_p x_p(t) + B_1 w(t) + B_2 u(t),$$
$$z(t) = C_1 x_p(t) + D_{11} w(t) + D_{12} u(t),$$
$$y(t) = C_2 x_p(t) + D_{21} w(t),$$

and controller

$$x_c(t+1) = A_c x_c(t) + B_c y(t),$$
$$u(t) = C_c x_c(t) + D_c y(t).$$

*Correspondence to: Stephen P. Boyd, Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305, U.S.A.
†E-mail: boyd@stanford.edu

Contract/grant sponsor: Focus Center Research Program Center for Circuit and System Solutions (www.c2s2.org); contract/grant number: 2003-CT-888
Contract/grant sponsor: AFOSR; contract/grant number: AF F49620-01-1-0365
Contract/grant sponsor: NSF; contract/grant number: ECS-0423905
Contract/grant sponsor: NSF; contract/grant number: 0529426
Contract/grant sponsor: DARPA/MIT; contract/grant number: 5710001848
Contract/grant sponsor: AFOSR; contract/grant number: FA9550-06-1-0514
Contract/grant sponsor: DARPA/Lockheed; contract/grant number: N66001-06-C-2021
Contract/grant sponsor: AFOSR/Vanderbilt; contract/grant number: FA9550-06-1-0312

Here $x_p(t)$ is the plant state, $u(t)$ is the control input, $y(t)$ is the sensor output, $w(t)$ and $z(t)$ are the exogenous input and output, respectively, and $x_c(t)$ is the controller state.

The vector $\theta \in \mathbf{R}^N$ will represent the design parameters or coefficients in the controller. Typically these are (some of) the entries in the matrices $A_c$, $B_c$, $C_c$, and $D_c$. We are given a *nominal controller design*, described by the coefficient vector $\theta^{\mathrm{nom}}$, and a set of acceptable controller designs $\mathscr{C} \subseteq \mathbf{R}^N$. The set $\mathscr{C}$ gives the (coefficients of the) controllers that achieve acceptable closed-loop performance. We assume that $\theta^{\mathrm{nom}} \in \mathscr{C}$, i.e. the nominal controller meets the performance specifications. For example, we can give $\mathscr{C}$ in terms of a single scalar performance measure $J : \mathbf{R}^N \to \mathbf{R}$, as

$$\mathscr{C} = \{\theta \mid J(\theta) \leqslant (1+\varepsilon) J(\theta^{\mathrm{nom}})\},$$

which are the designs no more than $\varepsilon$ worse than the nominal. If the nominal design is the controller that minimizes $J$, then $\mathscr{C}$ is the set of $\varepsilon$-suboptimal designs. Our goal is to find $\theta \in \mathscr{C}$ that achieves closed-loop performance close to the nominal, and at the same time low complexity.

The *complexity* of a vector of controller coefficients $\theta$ is measured by the function $\Phi : \mathbf{R}^N \to \mathbf{R}$,

$$\Phi(\theta) = \sum_{i=1}^{N} \phi_i(\theta_i),$$

where $\phi_i(\theta_i)$ gives the complexity of the $i$th coefficient of $\theta$. We can take, for example, $\phi_i(a)$ to be the number of bits needed to express $a$, or the total number of 1s in the binary expansion of $a$, in which case $\Phi(\theta)$ gives the total number of bits (or 1s) in the controller coefficients. Of course the functions $\phi_i$, and therefore also $\Phi$, can be discontinuous.

Our goal is to find the lowest complexity controller among the acceptable designs. We can express this as the optimization problem

$$\begin{array}{ll} \text{minimize} & \Phi(\theta) \\ \text{subject to} & \theta \in \mathscr{C}, \end{array} \tag{1}$$

with variable $\theta \in \mathbf{R}^N$. We call this as the *controller coefficient truncation problem* (CCTP), since we can think of the controller coefficient $\theta_i$ as a truncated version of the nominal controller coefficient $\theta_i^{\mathrm{nom}}$.

The CCTP (1) is in general very difficult to solve. For example, when $\Phi$ measures bit complexity, the CCTP can be cast as a combinatorial optimization problem, with the binary expansions of the coefficients as Boolean (i.e. $\{0, 1\}$) variables. Branch-and-bound, or other global optimization techniques, can be used to solve small CCTPs, with perhaps 10 coefficients. But we are interested in methods that can handle much larger problems, with perhaps hundreds (or more) of controller coefficients. In addition, it is not crucial to find the global solution of the CCTP (1); it is enough to find a controller with low (if not lowest) complexity.

In this paper we describe a heuristic algorithm for the CCTP (1) that runs quickly and scales to large problems. While the designs produced are very likely not globally optimal, they appear to be quite good. The method typically produces aggressively truncated controller designs, even when the allowed performance degradation over the nominal design is just a few per cent.

In our method, we greedily truncate individual coefficients sequentially, in random order, using a Lyapunov certificate (which is updated at each step) to guarantee the performance, i.e. $\theta \in \mathscr{C}$. When the algorithm is run multiple times, the randomness in the truncation order produces designs that are different, but have very similar total complexity. Running the algorithm a few times, and taking the best controller found, can give a modest improvement over running it just once.

Before proceeding we mention a related issue that we do *not* consider, at least until Section 6: the effects of truncation or saturation of the control signals $u(t)$, $y(t)$, and $x_c(t)$. This makes the entire control system nonlinear, and can lead to instability, large and small limit cycles, and other behavior. However, the Lyapunov-based methods described in the paper can be extended to handle nonlinearities; we briefly describe one such extension in Section 6.



DOI: 10.1002/rnc

### 1.2. Previous and related work

The subject of coefficient truncation is relatively old. It was initially discussed in the context of filter design: there was an understandable interest in designing finite word-length filters that would be easily implemented in hardware with a small degradation in performance (see [1, 2]). The idea of coefficient truncation subsequently appeared in other fields, such as speech processing [3] and control [4].

Several methods have been proposed for coefficient truncation: exhaustive search over possible truncated coefficients [1], successive truncation of coefficients and re-optimization over remaining ones [2, 5], local bivariate search around the scaled and truncated coefficients [6], tree-traversal techniques for truncated coefficients organized in a tree according to their complexity [7, 8], coefficient quantization using information-theoretic bounds [9], weighted least squares [10], simulated annealing [11, 12], genetic algorithms [13, 14], Tabu search [15], and design of optimal filter realizations that minimize coefficient complexity [11, 16]. Other approaches have formulated the problem as a nonlinear discrete optimization problem [17], or have used integer programming techniques over the space of powers-of-two coefficients [18, 19]. Barua *et al.* [20] surveys different methods for quantizing lifting coefficients for wavelet filters: mostly uniform bit allocation, exhaustively searched allocation, and simulated annealing with lumped scaling and/or gain compensation. Liu *et al.* [21] presents how to choose the optimal realization for an Linear Quadratic Gaussian (LQG) controller to be robust to finite word-length effects. The effects of quantization and finite word length on robust stability of digital controllers and performance bounds derived using Lyapunov theory are presented in [22].

### 1.3. Outline

In Section 2 we describe the general algorithm. In the next three sections we present examples, in each case working out the details for the general case, and illustrating the algorithm with a numerical instance of the problem. In Section 3 the controller has constant state feedback form, the nominal controller is linear quadratic regular (LQR) optimal, and the set of acceptable controllers is determined by the LQR cost. In Section 4 the controller is dynamic, and the objective is the decay rate of the closed-loop system. In Section 5 the controller is dynamic, with order equal to the plant; the nominal controller is a central $H_\infty$ optimal controller, and the set of acceptable controllers is determined by an $H_\infty$ criterion. In Section 6 we consider a simple example of a *nonlinear* feedback system, consisting of a linear plant and a controller that is nominally linear, but includes saturation in the state update and output equations. The objective is the decay rate of the *nonlinear* system.

## 2. THE ALGORITHM

Our algorithm uses two subroutines or methods: **interv**, which finds an interval of acceptable values of a coefficient and **trunc**, which truncates a coefficient, given an interval of acceptable choices. We first describe these methods more precisely, but still abstractly; more concrete descriptions will be given later in Sections 2.1 and 2.2.

The method **interv**$(\theta, i)$ takes as input the coefficient vector $\theta \in \mathscr{C}$ and a coefficient index $i$. It returns an interval $[l, u]$ of allowed values for $\theta_i$, with the other parameters held fixed, i.e. numbers $l$ and $u$, with $\theta_i \in [l, u]$, with

$$(\theta_1, \ldots, \theta_{i-1}, z, \theta_{i+1}, \ldots, \theta_N) \in \mathscr{C} \quad \text{for } z \in [l, u].$$

Of course the simple choice $l = u = \theta_i$ is always valid. At the other extreme, the largest valid interval that can be returned by **interv** is given by

$$l^\star = \inf\{l \mid (\theta_1, \ldots, \theta_{i-1}, z, \theta_{i+1}, \ldots, \theta_N) \in \mathscr{C} \quad \text{for } z \in [l, \theta_i]\},$$

$$u^\star = \sup\{u \mid (\theta_1, \ldots, \theta_{i-1}, z, \theta_{i+1}, \ldots, \theta_N) \in \mathscr{C} \quad \text{for } z \in [\theta_i, u]\}.$$



DOI: 10.1002/rnc

A typical implementation of **interv** falls between these two extremes, returning a reasonably large interval guaranteed to lie in $\mathscr{C}$, with reasonable computational effort. In the examples we will consider, this can be done using linear matrix inequalities (LMIs).

The $\mathbf{trunc}_i(x, l, u)$ is a truncation method which, given a number $x$ to be truncated and an interval $[l, u]$ of acceptable choices (containing $x$), returns a number $z$ in the interval $[l, u]$ with $\Phi_i(z) \leqslant \Phi_i(x)$. One valid choice is $z = x$; at the other extreme, the algorithm can return the point with smallest complexity in the interval, i.e. the minimizer of $\Phi_i(z)$ over $[l, u]$. For the complexity measures that we use in the examples shown later, we can easily compute the latter.

The algorithm is initialized with the nominal design, which we assume has finite complexity. At each step an index $i$ is chosen, and all parameters except $\theta_i$ are fixed. We use **interv** to find an interval of acceptable values for $\theta_i$, and then **trunc** to find a value of $\theta_i$ with (possibly) lower complexity. We have experimented with various methods for choosing the index $i$ in each step, and found the best results by organizing the algorithm into passes, each of which involves updating each parameter once; in each pass, the ordering of the indices is random. The algorithm stops when the parameter does not change over one pass. A high-level description of the algorithm is presented next:

$\theta := \theta^{\text{nom}}$
**repeat**
  $\theta^{\text{prev}} := \theta$
  **choose a permutation** $\pi$ of $(1, \ldots, N)$
  **for** $i = 1$ **to** $N$
    $j := \pi(i)$
    $[l, u] := \mathbf{interv}(\theta, j)$
    $\theta_j := \mathbf{trunc}_i(\theta_j, l, u)$
**until** $\theta = \theta^{\text{prev}}$

As the algorithm is random, it can and does converge to different points in different runs. It can be run several times, taking the best controller coefficient vector found as our final choice.

### *2.1. Complexity measures and truncation methods*

In this section we describe various possible complexity measures and the associated truncation methods. Any $z \in \mathbf{R}$ can be written as

$$z = s \sum_{i=-\infty}^{\infty} b_i 2^{-i},$$

where $s \in \{-1, 1\}$ is the sign and $b_i \in \{0, 1\}$ are the bits of $z$ in a binary expansion. (This representation can be made unique by ruling out any sequence that ends with all ones, i.e. $b_i = 1$ for $i \geqslant k$, for some $k$).

One possible complexity measure is the number of ones in the binary expansion of $z$,

$$\phi_{\text{ones}}(z) = \sum_{i=-\infty}^{\infty} b_i,$$

which gives the number of adders needed to implement multiplication by $z$ using a shift and sum method.

Another complexity measure is the width of the range of the non-zero bits, more commonly referred to as the number of bits in the expansion of $z$,

$$\phi_{\text{bits}}(z) = \max\{i \,|\, b_i \neq 0\} - \min\{i \,|\, b_i \neq 0\} + 1.$$

This measure is useful if multiplication by $z$ will be carried out in fixed-point arithmetic.

Yet another complexity measure is the number of bits needed in the fractional part of the binary expansion of $z$,

$$\phi_{\text{frac-bits}}(z) = \max\left\{0, \max_i\{i \,|\, b_i \neq 0\}\right\}.$$



DOI: 10.1002/rnc

For these complexity measures, it is straightforward to find the number $z$ that minimizes the measure in a given interval, i.e. to implement (the most powerful) **trunc** method. We assume that the binary expansions of $l$ and $u$ are finite (though possibly long),

$$s_l l_{-L} \ldots l_0.l_1 \ldots l_R, \quad s_u u_{-L} \ldots u_0.u_1 \ldots u_R,$$

respectively. The number $z$ will have at most $L$ bits in its integer part and $R$ bits in its fractional part, and we denote its bits as

$$s_z z_{-L} \ldots z_0.z_1 \ldots z_R.$$

With complexity measure $\phi_{\text{ones}}$ or $\phi_{\text{bits}}$, $z$ can be found as follows:

$z_i := 0$ for all $i$
**for** $i = -L$ **to** $R$
  **if** $l_i = u_i$, $z_i := l_i$
  **else**
  **if all bits after** $l_i$ **are** 0**, break**
  **else** $z_i := 1$

When the complexity measure is $\phi_{\text{frac-bits}}$, the same algorithm can be used, with $z_i$ initially set to zero for $i > 0$ and the for loop index modified to run from 1 to $R$, instead of from $-L$ to $R$.

### 2.2. *Interval computation via Lyapunov performance certificate*

Our approach to determining an interval $[l, u]$ for which

$$(\theta_1, \ldots, \theta_{i-1}, z, \theta_{i+1}, \ldots, \theta_N) \in \mathscr{C} \quad \text{for } z \in [l, u]$$

will be based on a conservative approximation of $\mathscr{C}$. Given $\theta \in \mathscr{C}$ we first find a convex set $\hat{\mathscr{C}}$ that satisfies $\theta \in \hat{\mathscr{C}}$ and $\hat{\mathscr{C}} \subseteq \mathscr{C}$. We then take

$$\begin{aligned} l &= \inf\{z | (\theta_1, \ldots, \theta_{i+1}, z, \theta_{i+1}, \ldots, \theta_N) \in \hat{\mathscr{C}}\}, \\ u &= \sup\{z | (\theta_1, \ldots, \theta_{i+1}, z, \theta_{i+1}, \ldots, \theta_N) \in \hat{\mathscr{C}}\}. \end{aligned} \tag{2}$$

Since $\hat{\mathscr{C}}$ is convex, it follows that

$$(\theta_1, \ldots, \theta_{i+1}, z, \theta_{i+1}, \ldots, \theta_N) \in \hat{\mathscr{C}} \subseteq \mathscr{C} \quad \text{for } z \in [l, u].$$

This is illustrated in Figure 1. For more on convex sets, see [23].

To find the set $\hat{\mathscr{C}}$, we use a *Lyapunov performance certificate*. The details depend on the particular performance measure or measures, but the common form is as follows. We express the set of acceptable controllers using LMIs:

$$\theta \in \hat{\mathscr{C}} \Longleftrightarrow \exists v L(\theta, v) \succcurlyeq 0,$$

where $L$ is a function that is bi-affine, i.e. affine in $\theta$ for fixed $v$ and affine in $v$ for fixed $\theta$. The symbol $\succcurlyeq$ refers to matrix inequality, between symmetric matrices, so the condition above is that

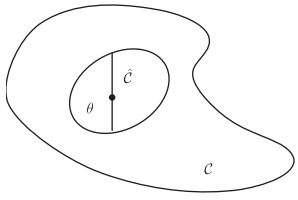


Figure 1. The set $\mathscr{C}$ of acceptable design parameters need not be convex, as shown in this example. The set $\hat{\mathscr{C}}$ is a convex subset of $\mathscr{C}$ that contains $\theta$. The interval of values of $\theta$, shown as the vertical line segment, gives an interval of values in $\mathscr{C}$.



DOI: 10.1002/rnc

$L(\theta, v)$ is positive semidefinite. The variable $v$ represents the coefficients in the Lyapunov function used to certify the performance. For more on representing control system specifications via LMIs, see, e.g. [24–26].

For a given $\theta \in \mathscr{C}$, we compute a value of $v$ such that $L(\theta, v) \succcurlyeq 0$. We then fix $v$, and take

$$\hat{\mathscr{C}} = \{\theta | L(\theta, v) \succcurlyeq 0\}. \tag{3}$$

This set depends on the particular choice of $v$; but in all cases, it is convex, indeed, it is described by an LMI in $\theta$. For a given $\theta \in \mathscr{C}$, $v$ can be typically chosen to maximize the minimum eigenvalue of $L(\theta, v)$ or to maximize the (log of the) determinant of $L(\theta, v)$. Both of these problems are convex: maximizing the minimum eigenvalue can be reduced to solve a semidefinite program (SDP) [27] and maximizing the determinant can be reduced to solving a MAXDET problem [28].

To find $l$ or $u$ in (2), we need to minimize or maximize a scalar variable over an LMI. This can be reduced to an eigenvalue computation [23, Exer. 4.38], and can be carried out efficiently. Since $L(\theta, v)$ is bi-affine in $\theta$ and $v$, it can be expressed as

$$L(\theta, v) = L_0 + \sum_{i=1}^{N} \theta_i L_i,$$

where we have obscured the fact that the matrices $L_0$ and $L_i$ depend on $v$. When $\tilde{\theta} = (\theta_1, \ldots, \theta_{i-1}, z, \theta_{i+1}, \ldots, \theta_N)$, we have

$$L(\tilde{\theta}, v) = L(\theta, v) + (z - \theta_i) L_i.$$

Assuming that $L(\theta, v) \succ 0$, the range $[l, u]$ of $\theta_i$ consists of the values of $z$ for which $L(\tilde{\theta}, v) \succcurlyeq 0$. It can be shown that

$$l = \theta_i - \min\{1/\lambda_i | \lambda_i > 0\}, \quad u = \theta_i - \max\{1/\lambda_i | \lambda_i < 0\}, \tag{4}$$

where $\lambda_i$ are the eigenvalues of $L(\theta, v)^{-1/2} L_i L(\theta, v)^{-1/2}$.

In the examples we will consider, the LMIs that arise have an even more specific form,

$$L(\theta, v) = \begin{bmatrix} I & Z^T \\ Z & I \end{bmatrix} \succcurlyeq 0,$$

where

$$Z = Z_0 + \sum_{i=1}^{N} \theta_i v_i w_i^T.$$

Here $Z_0$ is a matrix, and $v_i$ and $w_i$ are vectors, with dimensions and data that depend on the particular problem. In the general notation used above, this corresponds to

$$L_0 = \begin{bmatrix} I & Z_0^T \\ Z_0 & I \end{bmatrix}, \quad L_i = \begin{bmatrix} 0 & w_i v_i^T \\ v_i w_i^T & 0 \end{bmatrix}, \quad i = 1, \ldots, N.$$

We can then express $\hat{\mathscr{C}}$ as

$$\hat{\mathscr{C}} = \{\theta | \|Z\| \leqslant 1\},$$

where $\|\cdot\|$ denotes the spectral norm (maximum singular value).

We now give the details of how to find the range of the coefficient $\theta_i$ in the convex set $\hat{\mathscr{C}}$, i.e. how to compute $l$ and $u$ in (2).

Note that the rank of $L_i$ is exactly 2. Assuming that $L(\theta, v) \succ 0$ and $v_i$ and $w_i$ are both nonzero, the matrix $L(\theta, v)^{-1/2} L_i L(\theta, v)^{-1/2}$ has one positive eigenvalue $\lambda_{\max}$, $(2n + m - 2)$ zero eigenvalues, and one negative eigenvalue $\lambda_{\min}$. We now proceed to find more explicit expressions for $\lambda_{\min}$ and



DOI: 10.1002/rnc

$\lambda_{\max}$. Let $Z=U\Sigma V^T$ be the full singular value decomposition of $Z$ where $U$ and $V$ are orthogonal matrices and $\Sigma$ has the same dimensions as $Z$. If $m=n$, $\Sigma$ is diagonal. If $m\geqslant n$, we have

$$\Sigma=\begin{bmatrix}\mathbf{diag}(\sigma_1,\ldots,\sigma_n)\\0\end{bmatrix}.$$

Otherwise, we have

$$\Sigma=[\mathbf{diag}(\sigma_1,\ldots,\sigma_m)\ \ 0].$$

Let

$$x=U^T v_i,\quad y=V^T w_i,\quad E=\begin{bmatrix}I & \Sigma^T\\ \Sigma & I\end{bmatrix},\quad F=\begin{bmatrix}0 & yx^T\\ xy^T & 0\end{bmatrix}.$$

Using a block Cholesky factorization, we can write $E=CC^T$, where

$$C=\begin{bmatrix}I & 0\\ \Sigma & (I-\Sigma\Sigma^T)^{1/2}\end{bmatrix}.$$

Note that

$$C^{-1}=\begin{bmatrix}I & 0\\ -A\Sigma & A\end{bmatrix},$$

where $A=(I-\Sigma\Sigma^T)^{-1/2}$.

It is easy to show that $\lambda_{\min}$ and $\lambda_{\max}$ are, respectively, the minimum and maximum eigenvalues of $C^{-1}FC^{-\mathrm{T}}$. Since

$$F=\begin{bmatrix}0 & y\\ x & 0\end{bmatrix}\begin{bmatrix}y^T & 0\\ 0 & x^T\end{bmatrix},$$

and since non-zero eigenvalues of $MN$ and $NM$ are identical for any two matrices $M\in\mathbf{R}^{n\times m}$ and $N\in\mathbf{R}^{m\times n}$, $\lambda_{\min}$ and $\lambda_{\max}$ are the eigenvalues of

$$\begin{bmatrix}y^T & 0\\ 0 & x^T\end{bmatrix}C^{-\mathrm{T}}C^{-1}\begin{bmatrix}0 & y\\ x & 0\end{bmatrix}.$$

These can be found analytically as

$$\lambda_{\min}=-\alpha-\sqrt{3\alpha^2+\beta y^T y+\beta\gamma}, \tag{5}$$

$$\lambda_{\max}=-\alpha+\sqrt{3\alpha^2+\beta y^T y+\beta\gamma}. \tag{6}$$

The terms $\alpha$, $\beta$, and $\gamma$ can be computed more easily as

$$\alpha=x^T A^2\Sigma y=\sum_{i=1}^{\min\{m,n\}}\frac{x_i y_i\sigma_i}{1-\sigma_i^2}, \tag{7}$$

$$\beta=x^T A^2 x=\sum_{i=1}^{m}\frac{x_i^2}{1-\sigma_i^2}, \tag{8}$$

$$\gamma=y^T\Sigma^T A^2\Sigma y=\sum_{j=1}^{n}\frac{y_j^2\sigma_j^2}{1-\sigma_j^2}. \tag{9}$$



DOI: 10.1002/rnc

In summary, to find $l$ and $u$, we start by computing the SVD of the $Z$ and setting $x=U^T v_i$, $y=V^T w_i$. We then proceed to compute the three terms in (7)–(9) and compute $\lambda_{\min}$ and $\lambda_{\max}$ from (5) and (6). Finally, $l$ and $u$ are found from (4):

$$l=\theta_i-1/\lambda_{\max}, \quad u=\theta_i-1/\lambda_{\min}. \tag{10}$$

## 3. STATE FEEDBACK CONTROLLER WITH LQR COST

We will demonstrate how to apply the algorithm to a specific problem class where the plant is given by

$$x(t+1)=Ax(t)+Bu(t), \quad x(0)=x_0, \tag{11}$$

and is controlled by a state feedback gain controller given by

$$u(t)=Kx(t), \tag{12}$$

where $A\in\mathbf{R}^{n\times n}$, $B\in\mathbf{R}^{n\times m}$, $C\in\mathbf{R}^{k\times n}$, $K\in\mathbf{R}^{m\times n}$ is the feedback gain matrix, $x(t)\in\mathbf{R}^n$ is the state of the system, and $u(t)\in\mathbf{R}^m$ is the input to the system. The design variables are the entries of the matrix $K$.

### *3.1. Admissible controllers*

Given $Q\in\mathbf{R}^{n\times n}$ as positive semidefinite and $R\in\mathbf{R}^{m\times m}$ as positive definite, the performance measure is given by the LQR cost

$$J(K)=\mathbf{E}\left[\sum_{t=0}^{\infty} x(t)^T Qx(t)+u(t)^T Ru(t)\right]=\mathbf{E}\left[\sum_{t=0}^{\infty} x(t)^T (Q+K^T RK)x(t)\right], \tag{13}$$

where the expectation is taken over $x_0\sim\mathcal{N}(0,\Sigma)$. If $A+BK$ is unstable $J(K)$ is infinite. Otherwise, let $P$ be the (unique) solution to the Lyapunov equation

$$(A+BK)^T P(A+BK)-P+Q+K^T RK=0. \tag{14}$$

The cost in (13) can be expressed as $J(K)=\mathbf{Tr}(\Sigma P)$. This holds because

$$\begin{aligned}
J(K) &= \mathbf{E}\left[\sum_{t=0}^{\infty} x(t)^T Px(t)-x(t)^T (A+BK)^T P(A+BK)x(t)\right] \\
&= \mathbf{E}\left[\sum_{t=0}^{\infty} x(t)^T Px(t)-x(t+1)^T Px(t+1)\right] \\
&= \mathbf{E}[x_0^T Px_0] \\
&= \mathbf{Tr}(\mathbf{E}[x_0x_0^T]P) \\
&= \mathbf{Tr}(\Sigma P).
\end{aligned}$$

The nominal design $K^{\text{nom}}$ is chosen to be the optimal state feedback controller, i.e. the one that minimizes the LQR cost $J$. It can be found as follows:

$$K^{\text{nom}}=-(R+B^T P^{\text{nom}}B)^{-1}B^T P^{\text{nom}}A,$$

where $P^{\text{nom}}$ is the solution of the discrete-time algebraic Riccati equation

$$P^{\text{nom}}=Q+A^T P^{\text{nom}}A-A^T P^{\text{nom}}A(R+B^T P^{\text{nom}}B)^{-1}B^T P^{\text{nom}}A.$$



DOI: 10.1002/rnc

When $K = K^{\text{nom}}$, $P^{\text{nom}}$ is also the solution of the Lyapunov equation (14). The LQR cost associated with the optimal controller is $J^{\text{nom}} = \mathbf{Tr}(\Sigma P^{\text{nom}})$.

We define the set of admissible controller design as

$$\mathscr{C} = \{K \,|\, J(K) \leqslant (1+\varepsilon) J^{\text{nom}}\},$$

where $\varepsilon$ is a given positive number. This means that a controller design is admissible if and only if it is $\varepsilon$-suboptimal.

We choose the Lyapunov performance certificate $L$ to be

$$L(K, P) = \begin{bmatrix} P - (A+BK)^T P(A+BK) - Q - K^T RK & 0 & 0 \\ 0 & (1+\varepsilon)J^{\text{nom}} - \mathbf{Tr}(\Sigma P) & 0 \\ 0 & 0 & P \end{bmatrix}.$$

Here $K$ and $P$ correspond, respectively, to $\theta$ and $v$ introduced in Section 2.2. The condition that $L(K, P) \succcurlyeq 0$ is equivalent to

$$(A+BK)^T P(A+BK) - P + Q + K^T RK \preccurlyeq 0,$$

$$\mathbf{Tr}(\Sigma P) \leqslant (1+\varepsilon) J^{\text{nom}}, \tag{15}$$

$$P \succcurlyeq 0. \tag{16}$$

As (15) and (16) do not depend on $K$, and for a particular choice $P$, (3) becomes

$$\hat{\mathscr{C}} = \{K \,|\, (A+BK)^T P(A+BK) - P + Q + K^T RK \preccurlyeq 0\}. \tag{17}$$

Given $K \in \mathscr{C}$, any matrix $P$ that satisfies $L(K, P) \succcurlyeq 0$ is a valid choice. We take $P$ to be the solution of the following optimization problem

$$\begin{aligned} &\text{maximize} && \lambda_{\min}(L(K, P)) \\ &\text{subject to} && L(K, P) \succcurlyeq 0. \end{aligned}$$

Here $\lambda_{\min}(L(K, P))$ is the minimum eigenvalue of $L(K, P)$ and $P$ is the variable we are optimizing over. Recall that $K$ is fixed.

We will now show that $\hat{\mathscr{C}} \subseteq \mathscr{C}$. Let $K \in \hat{\mathscr{C}}$. Consider the Lyapunov function $V : \mathbf{R}^n \to \mathbf{R}$ defined as $V(z) = z^T Pz$. For any $T > 0$,

$$\begin{aligned} V(x(T)) - V(x(0)) &= \sum_{t=0}^{T} V(x(t+1)) - V(x(t)) \\ &= \sum_{t=0}^{T} x(t+1)^T Px(t+1) - x(t)^T Px(t) \\ &= \sum_{t=0}^{T} x(t)^T ((A+BK)^T P(A+BK) - P)x(t) \\ &\leqslant -\sum_{t=0}^{T} x(t)(Q + K^T RK)x(t). \end{aligned}$$

Therefore,

$$\sum_{t=0}^{T} x(t)(Q + K^T RK)x(t) \leqslant V(x(0)) - V(x(T)) \leqslant V(x(0)),$$

where the last inequality follows because $V(x(T)) \geqslant 0$ from (16). Letting $T$ tend to infinity and taking expectation over $x_0$, we obtain $J(K) \leqslant \mathbf{Tr}(\Sigma P)$. It follows from (15) that

$$J(K) \leqslant (1+\varepsilon) J^{\text{nom}}.$$



DOI: 10.1002/rnc

### 3.2. Coefficient range calculation

Let $(l, u)$ be the range of coefficient $K_{ij}$. Given (17), problem (2) becomes

$$l = \min\{K_{ij} | (A+BK)^T P(A+BK) - P + Q + K^T RK \preccurlyeq 0\},$$

$$u = \max\{K_{ij} | (A+BK)^T P(A+BK) - P + Q + K^T RK \preccurlyeq 0\}.$$

The inequality in (17) is equivalent to

$$\left\| \begin{bmatrix} P^{1/2}(A+BK) \\ R^{1/2}K \end{bmatrix} (P-Q)^{-1/2} \right\| \leqslant 1. \tag{18}$$

The method outlined in Section 2.2 can be used to compute $l$ and $u$ by taking

$$Z = \begin{bmatrix} P^{1/2}(A+BK) \\ R^{1/2}K \end{bmatrix} (P-Q)^{-1/2},$$

$$v = \begin{bmatrix} P^{1/2}B \\ R^{1/2} \end{bmatrix} e_i, \quad w = (P-Q)^{-1/2} e_j,$$

where $e_i$ and $e_j$ are, respectively, the $i$th unit vector in $\mathbf{R}^m$ and $j$th unit vector in $\mathbf{R}^n$ and $K \in \hat{\mathscr{C}}$ is the current admissible controller design.

### 3.3. Numerical instance

Our example has dimensions $n = 10$ and $m = 5$. We generated the plant randomly, as $A = I + 0.1X/\sqrt{n}$, where $X_{ij}$ are independent and identically distributed (IID) $\mathcal{N}(0, 1)$. We generated the matrix $B \in \mathbf{R}^{10\times 5}$ with $B_{ij}$ IID $\mathcal{N}(0, 1)$. We take $\Sigma = I$, $Q = I$, and $R = I$.

The complexity measures $\phi_i(z)$ are chosen to be $\phi_{\text{frac-bits}}$. The fractional part of each entry of $K^{\text{nom}}$ is expressed with 40 bits, requiring a total of 2000 bits to express $K^{\text{nom}}$, i.e. $\Phi(\theta^{\text{nom}}) = 2000$ bits. We take $\varepsilon = 15\%$, i.e. admissible feedback controllers are those that are up to 15% suboptimal.

The progress of the complexity $\Phi(\theta)$ during a sample run of the algorithm is shown in Figure 2. In this sample run the algorithm converges to a complexity of 85 bits in one pass over the variables. During the run of the algorithm the cost $J$ is approximately constant and equal to its maximum allowed value of $1.15J^{\text{nom}}$.

The best design after 10 random runs of the algorithm achieves a complexity of $\Phi(\theta) = 81$ bits, with a cost of $J(\theta) = 1.1494J(\theta^{\text{nom}})$. The best design found after 100 random runs of the algorithm achieves a complexity of $\Phi(\theta) = 75$ bits and $J(\theta) = 1.1495J(\theta^{\text{nom}})$.

This best design gives very aggressive coefficient truncation with only 1.5 bits per coefficient. This is illustrated in Figure 3, which shows the distribution of the (50) coefficients of the nominal design and the coefficients of the best design. We observe that most of the coefficients in the best design are 0 while the remaining ones have a complexity of at most 3 bits (for example for $\theta_i = 0.125$).

*3.3.1. Multiple random instances.* We report above the results for one random instance of the problem. We now generate 100 random instances of $A$ and $B$ with other problem data the same (i.e. $n = 10, m = 5, \Sigma = I, Q = I, R = I, \varepsilon = 15\%$). For each instance, we compute $K^{\text{nom}}$ and express the fractional part of each of its entries with 40 bits (i.e. $\phi(\theta^{\text{nom}}) = 2000$ bits).

For each instance, we record the best design after 10 random runs of the algorithm. The complexities of the best designs range between 74 and 127 bits, with a mean of 98.29 bits and a standard deviation of 11.2 bits. The performance degradations of these designs ranged between 14.88 and 14.99%. For these designs, the algorithm converged in an average of 1.1250 passes over the variables. Thus the results reported in single instance case above are quite typical.



DOI: 10.1002/rnc

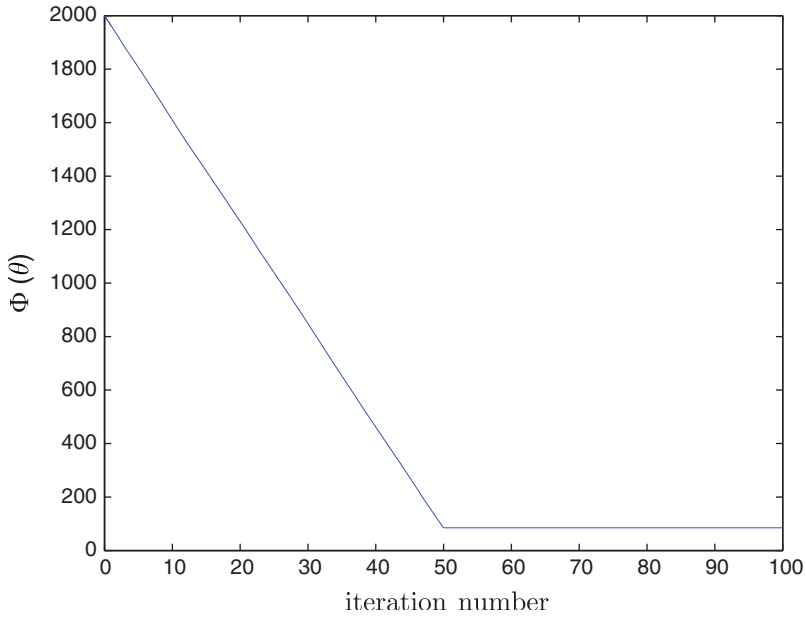


Figure 2. Total number of bits required to express $\theta$ versus iteration number.

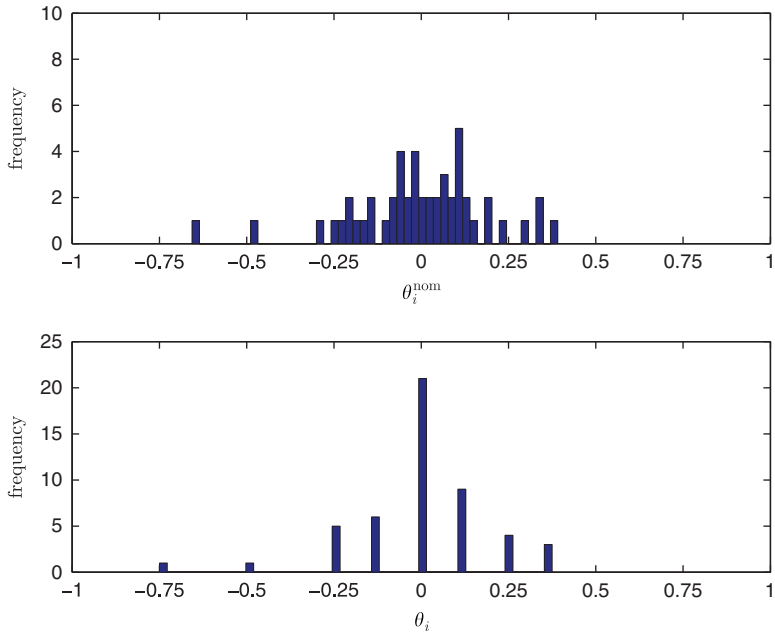


Figure 3. Top: Histogram of coefficients in the nominal design. Bottom: Histogram of coefficients in the best design out of 10 random runs.

We can compare the results obtained with a simpler approach, in which we truncate the binary expansions of the coefficients to precision $2^{-q}$ (i.e. with $q$ bits in their fractional parts), choosing $q$ as small as possible while still maintaining $\theta \in \mathscr{C}$. For the 100 random instances that were generated, $q$ varied between 3 and 5 bits per coefficient with a mean of 4.57 bits. In contrast, our algorithm gave aggressively truncated controllers, with coefficient complexity between 1.48 and 2.54 bits per coefficient, with a mean of 1.96 bits per coefficient.



DOI: 10.1002/rnc

## 4. DYNAMIC CONTROLLER WITH DECAY RATE SPECIFICATION

We demonstrate how to apply the algorithm to the problem class where the plant is given by

$$x_p(t+1)=A_p x_p(t)+B_p u(t), \quad y(t)=C_p x_p(t), \tag{19}$$

and is controlled by a dynamic controller given by

$$x_c(t+1)=A_c x_c(t)+B_c y(t), \quad u(t)=C_c x_c(t), \tag{20}$$

where $x_p(t)\in\mathbf{R}^{n_p}$, $u(t)\in\mathbf{R}^{m_c}$, $y(t)\in\mathbf{R}^{m_p}$, $A_p\in\mathbf{R}^{n_p\times n_p}$, $B_p\in\mathbf{R}^{n_p\times m_c}$, $C_p\in\mathbf{R}^{m_p\times n_p}$, $x_c(t)\in\mathbf{R}^{n_c}$, $A_c\in\mathbf{R}^{n_c\times n_c}$, $B_c\in\mathbf{R}^{n_c\times m_p}$, and $C_c\in\mathbf{R}^{m_c\times n_c}$.

The closed-loop system is given by $x(t+1)=Ax(t)$ where

$$x(t)=\begin{bmatrix} x_p(t) \\ x_c(t) \end{bmatrix}, \quad A=\begin{bmatrix} A_p & B_pC_c \\ B_cC_p & A_c \end{bmatrix}. \tag{21}$$

The design variables are the entries of the controller matrices $A_c$, $B_c$, and $C_c$.

### 4.1. Admissible controllers

A controller $(A_c, B_c, C_c)$ is admissible if the decay rate of the closed-loop system is less than a given rate $\alpha$, where $0\leqslant\alpha\leqslant 1$. The decay rate is given by $\rho(A)$, where $A$ is the matrix specified in (21).

The performance measure is chosen to be the decay rate of the closed-loop system, i.e. $J(A_c, B_c, C_c)=\rho(A)$.

We are given a nominal controller design $(A_c^{\mathrm{nom}}, B_c^{\mathrm{nom}}, C_c^{\mathrm{nom}})$ such that

$$J(A_c^{\mathrm{nom}}, B_c^{\mathrm{nom}}, C_c^{\mathrm{nom}})=\rho.$$

We define the set of admissible controller designs as

$$\mathscr{C}=\{(A_c, B_c, C_c)\,|\,J(A_c, B_c, C_c)\leqslant\alpha\},$$

where $\alpha=(1+\varepsilon)\rho$, and $\varepsilon$ is a given positive number.

We choose the Lyapunov performance certificate $L$ to be

$$L(A_c, B_c, C_c, P)=\begin{bmatrix} \alpha^2 P-A^T PA & 0 \\ 0 & P \end{bmatrix},$$

where $A$ is the matrix defined in (21). Here $(A_c, B_c, C_c)$ and $P$ correspond, respectively, to $\theta$ and $v$ introduced in Section 2.2. The condition that $L(A_c, B_c, C_c, P)\succcurlyeq 0$ is equivalent to

$$\begin{aligned} A^T PA &\preccurlyeq \alpha^2 P, \\ P &\succcurlyeq 0. \end{aligned} \tag{22}$$

As (22) does not depend on $(A_c, B_c, C_c)$, for a fixed choice of $P$, (3) becomes

$$\hat{\mathscr{C}}=\{(A_c, B_c, C_c)\,|\,A^T PA\leqslant\alpha^2 P\}. \tag{23}$$

Any matrix $P$ that satisfies $L(A_c, B_c, C_c, P)\succcurlyeq 0$ for $(A_c, B_c, C_c)\in\mathscr{C}$ is a valid choice. We take $P$ to be the solution of the following optimization problem

$$\begin{aligned} \text{maximize} \quad & \lambda_{\min}(L(A_c, B_c, C_c, P)) \\ \text{subject to} \quad & L(A_c, B_c, C_c, P)\succcurlyeq 0 \\ & \mathbf{Tr}(P)=1. \end{aligned}$$



DOI: 10.1002/rnc

Here $\lambda_{\min}(L(A_c, B_c, C_c, P))$ is the minimum eigenvalue of $L(A_c, B_c, C_c, P)$, and $P$ is the variable we are maximizing over. Recall that $A_c$, $B_c$, and $C_c$ are fixed. The constraint $\mathbf{Tr}(P)=1$ is added because $L(A_c, B_c, C_c, P)$ is homogeneous in $P$.

We will now show that $\hat{\mathscr{C}} \subseteq \mathscr{C}$. Let $(A_c, B_c, C_c) \in \hat{\mathscr{C}}$. Consider the Lyapunov function $V: \mathbf{R}^{n_p+n_c} \to \mathbf{R}$ defined as $V(z)=z^T Pz$. Since $A^T PA \leqslant \alpha^2 P$ then for all $t \geqslant 0$

$$x(t)^T A^T PAx(t) \leqslant \alpha^2 x(t)^T Px(t)$$

$$x(t+1)^T Px(t+1) \leqslant \alpha^2 x(t)^T Px(t)$$

$$V(x(t+1)) \leqslant \alpha^2 V(x(t)).$$

This means that for all $t \geqslant 0$, $V(x(t)) \leqslant \alpha^{2t} V(x(0))$ and

$$\lambda_{\min}(P)\|x(t)\|^2 \leqslant x(t)^T Px(t) \leqslant \alpha^{2t} x(0)^T Px(0) \leqslant \alpha^{2t} \lambda_{\max}(P)\|x(0)\|^2,$$

then $\|x(t)\| \leqslant \sqrt{\kappa(P)}\alpha^t \|x(0)\|$, where $\kappa(P)$ is the condition number of $P$. The decay rate of the system is then less than $\alpha$, as required.

### 4.2. Coefficient range calculation

Let $(l, u)$ be the range of coefficient $(A_c)_{ij}$. Given (23), problem (2) becomes

$$l = \min\{(A_c)_{ij} | A^T PA \leqslant \alpha^2 P\},$$

$$u = \max\{(A_c)_{ij} | A^T PA \leqslant \alpha^2 P\}.$$

The inequality in (23) is equivalent to

$$\| P^{1/2} AP^{-1/2} \| \leqslant \alpha. \tag{24}$$

The method outlined in Section 2.2 can be used to compute $l$ and $u$ by taking

$$Z = (1/\alpha) P^{1/2} AP^{-1/2}, \quad v = (1/\alpha) P^{1/2} \begin{bmatrix} 0 \\ e_i \end{bmatrix}, \quad w = P^{-1/2} \begin{bmatrix} 0 \\ e_j \end{bmatrix},$$

where $e_i$ and $e_j$ are, respectively, the $i$th and $j$th unit vectors in $\mathbf{R}^{n_c}$ and $A$ is the closed-loop matrix associated with $(A_c, B_c, C_c) \in \hat{\mathscr{C}}$.

The same method can be used to find the ranges of coefficients in $B_c$ and $C_c$ and the same formulas can be used but with slightly modified definitions for $v$ and $w$. To find the range of coefficient $(B_c)_{ij}$, use the same definitions for $Z$ and $v$ but let

$$w = P^{-1/2} \begin{bmatrix} C_p^T e^j \\ 0 \end{bmatrix},$$

where $e_j$ is the $j$th unit vector in $\mathbf{R}^{m_c}$. To find the range of coefficient $(C_c)_{ij}$, use the same definitions for $Z$ and $w$ but let

$$v = (1/\alpha) P^{1/2} \begin{bmatrix} B_p e^i \\ 0 \end{bmatrix},$$

where $e_i$ is the $i$th unit vector in $\mathbf{R}^{m_c}$.

### 4.3. Numerical instance

We test the proposed method in the case where the plant is given by

$$x_p(t+1) = A_p x_p(t) + B_p u(t) + w(t), \quad y(t) = C_p x_p(t) + v(t), \tag{25}$$



DOI: 10.1002/rnc

where $w(t) \sim \mathcal{N}(0, I)$ is the input noise and $v(t) \sim \mathcal{N}(0, I)$ is the measurement noise. The plant is controlled by an LQG controller with $Q = I$, $R = I$. The matrices describing the controller are

$$A_c = A_p + B_p K - L C_p, \quad B_c = L, \quad C_c = K, \tag{26}$$

where

$$K = -(B_p^T P_1 B_p + R)^{-1} B_p^T P_1 A_p, \quad L = A_p P_2 C_p^T (C_p P_2 C_p^T + V)^{-1}. \tag{27}$$

$P_1$ and $P_2$ are the unique positive semidefinite solutions to the discrete-time algebraic Riccati equations

$$P_1 = A_p^T P_1 A_p + Q - A_p^T P_1 B_p (R + B_p^T P_1 B_p)^{-1} B_p^T P_1 A_p,$$

$$P_2 = A_p P_2 A_p^T + W - A_p P_2 C_p^T (C_p P_2 C_p^T + V)^{-1} C_p P_2 A_p^T.$$

Our example has dimensions $n_p = 5$, $m_c = 2$, and $m_p = 2$. The plant matrix $A_p$ is randomly generated using the same method used to generate $A$ in Section 3.3. The entries of $B_p$ and $C_p$ are IID $\mathcal{N}(0, 1)$. The matrices $A_c^{\text{nom}}$, $B_c^{\text{nom}}$, and $C_c^{\text{nom}}$ are then computed using the formulas presented above.

The complexity measures $\phi_i(z)$ are chosen to be $\phi_{\text{frac-bits}}$. The fractional part of each entry of $A_c^{\text{nom}}$, $B_c^{\text{nom}}$, and $C_c^{\text{nom}}$ is expressed with 40 bits, requiring a total of 1800 bits, i.e. $\Phi(\theta^{\text{nom}}) = 1800$ bits. We run the algorithm with $\varepsilon = 5\%$.

The progress of the complexity $\Phi(\theta)$ and percentage deterioration in performance $100(J - J^{\text{nom}})/J^{\text{nom}}$ during the three sample runs of the algorithm are shown in Figure 4.

It is interesting to note that the three sample runs achieve designs with similar complexities but with a varying range of performance degradation (although all within the allowable limit). This suggests the following methodology: run the algorithm many times, choose the designs achieving complexity below a specific value, and pick among those the design with the lowest degradation in performance. Applying this idea to the 100 sample runs we obtained, we found that among

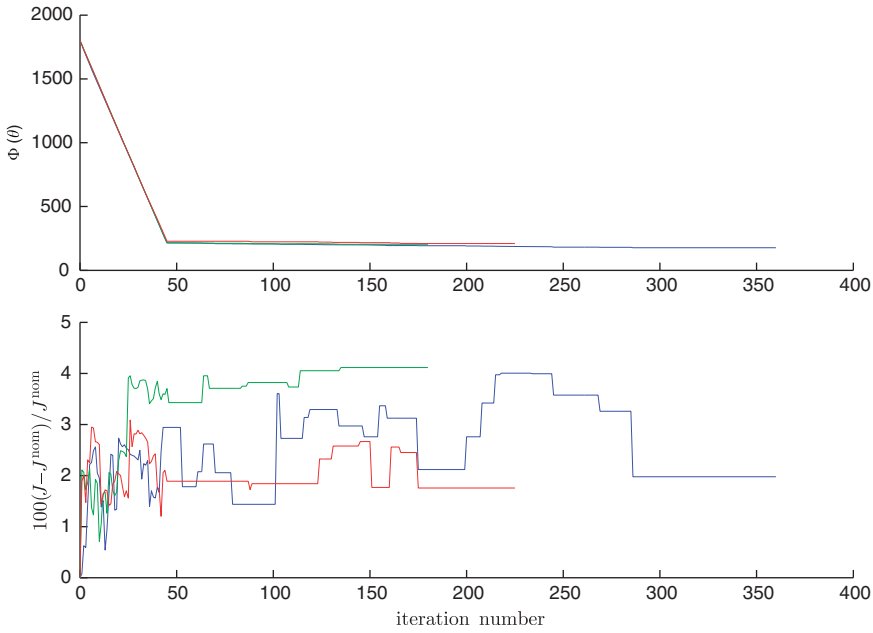


Figure 4. Top: Total number of bits required to express $\theta$, versus iteration number. Bottom: Percentage deterioration in performance versus iteration number.



DOI: 10.1002/rnc

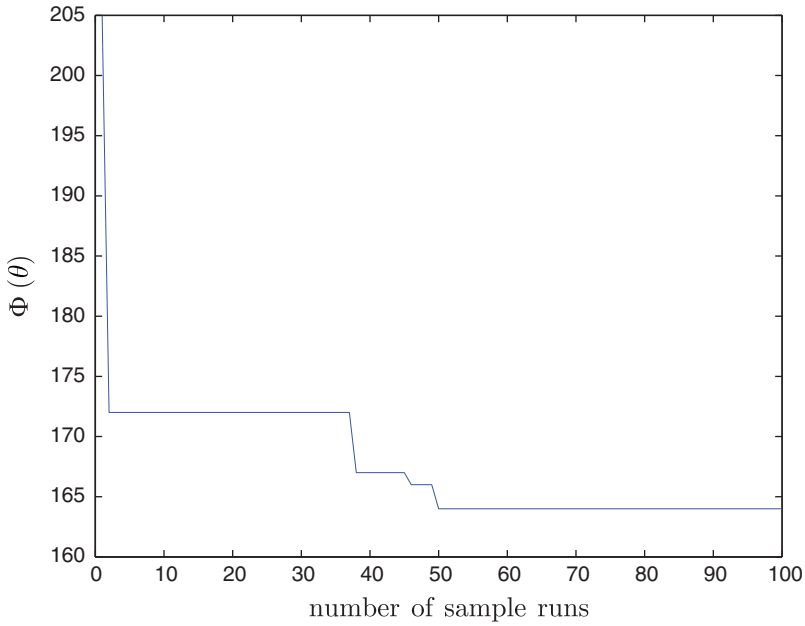


Figure 5. Best design complexity versus number of sample runs of the algorithm.

the designs achieving a complexity less than 170 bits, there exists a design whose performance is degraded by only 2.15%.

The best design after 10 random runs of the algorithm achieves a complexity of $\Phi(\theta)=171$ bits with a cost of $J(\theta)=1.0246J(\theta^{\text{nom}})$. The best design after 100 random runs of the algorithm achieves a complexity of $\Phi(\theta)=164$ bits and $J(\theta)=1.0362J(\theta^{\text{nom}})$. Figure 5 shows the best available design complexity versus the number of sample runs of the algorithm.

*4.3.1. Multiple random instances.* We now generate 100 random instances of the matrices $A_p$, $B_p$, $C_p$, with the same values for other data: $n_p=5$, $m_c=2$, $m_p=2$, $Q=I$, $R=I$, and $\varepsilon=5\%$. For each instance, we compute $A^{\text{nom}}$, $B^{\text{nom}}$, $C^{\text{nom}}$ and express the fractional part of each of their entries with 40 bits (i.e. $\phi(\theta^{\text{nom}})=1800$ bits).

For each instance, we record the best design after 10 random runs of the algorithm. The complexities of the best designs range between 17 and 472 bits, with a mean of 228.68 bits and a standard deviation of 107.72 bits. The performance degradations of the best designs ranged between 0.32 and 4.80%. The algorithm takes an average of 3.31 passes over the variables to converge.

We compare these results with the simple approach of truncating the binary expansions of the coefficients to precision $2^{-q}$ (i.e. with $q$ bits in their fractional parts), choosing $q$ as small as possible while still maintaining $\theta\in\mathscr{C}$. For the 100 random instances that were generated, $q$ varied between 5 and 14 bits per coefficient, with a mean of 8.39 bits. In contrast, our algorithm gave controllers whose coefficient complexity was between 0.38 and 10.48 bits per coefficient, with a mean of 5.08 bits per coefficient. In particular, for the instance in which our algorithm returned a controller with complexity 17 (i.e. 0.38 bits per coefficients), $q$ was 5 bits per coefficients.

## 5. DYNAMIC CONTROLLER WITH $H_\infty$-NORM SPECIFICATION

We demonstrate how to apply the algorithm to the problem class where the plant is given by

$$x_p(t+1)=A_1x_p(t)+B_1w(t)+B_2u(t),\quad x(0)=0,$$



DOI: 10.1002/rnc

$$z(t) = C_1 x_p(t) + D_{11} w(t) + D_{12} u(t),$$
$$y(t) = C_2 x_p(t) + D_{21} w(t),$$

and is controlled by a dynamic controller given by

$$x_c(t+1) = A_c x_c(t) + B_c y(t), \quad u(t) = C_c x_c(t) + D_c y(t), \quad x_c(0) = 0.$$

where $x_p(t) \in \mathbf{R}^{n_p}$, $u(t) \in \mathbf{R}^{m_c}$, $w(t) \in \mathbf{R}^k$, $y(t) \in \mathbf{R}^{m_p}$, and $x_c(t) \in \mathbf{R}^{n_c}$.

The closed-loop system is given by

$$x(t+1) = Ax(t) + Bw(t), \quad z(t) = Cx(t) + Dw(t),$$

where

$$x(t) = \begin{bmatrix} x_p(t) \\ x_c(t) \end{bmatrix}, \quad A = \begin{bmatrix} A_1 + B_2 D_c C_2 & B_2 C_c \\ B_c C_2 & A_c \end{bmatrix}, \quad B = \begin{bmatrix} B_1 + B_2 D_c D_{21} \\ B_c D_{21} \end{bmatrix}, \tag{28}$$

$$C = [C_1 + D_{12} D_c C_2 \;\; D_{12} C_c], \quad D = D_{11} + D_{12} D_c D_{21}. \tag{29}$$

The design variables are the entries of the controller matrices $A_c$, $B_c$, $C_c$, and $D_c$.

### 5.1. Admissible controllers

Let $G(s)$ be the transfer function of the closed-loop system. The $H_\infty$ norm of $G(s)$ is its largest input/output RMS gain, i.e.

$$\|G\|_\infty = \sup_{u \neq 0} \frac{\|z\|_2}{\|w\|_2},$$

where $z(t)$ is the output of the closed-loop system for a given input $w(t)$, and where

$$\|x\|_2 = \left( \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} \|x(t)\|^2 \right)^{1/2},$$

for any signal $x(t)$, $t \geqslant 0$.

The performance measure $J(A_c, B_c, C_c, D_c)$ is chosen to be the $H_\infty$ norm of the transfer function of the closed-loop system.

We are given a nominal controller design $(A_c^{\text{nom}}, B_c^{\text{nom}}, C_c^{\text{nom}}, D_c^{\text{nom}})$ such that

$$J(A_c^{\text{nom}}, B_c^{\text{nom}}, C_c^{\text{nom}}, D_c^{\text{nom}}) = \gamma^{\text{nom}}.$$

A controller $(A_c, B_c, C_c, D_c)$ is admissible if the $H_\infty$ norm of the transfer function of the closed-loop transfer function is less than a given value $\gamma > 0$, where $\gamma = (1+\varepsilon)\gamma^{\text{nom}}$ for given a positive number $\varepsilon$. In other words, the set of admissible controller designs is

$$\mathscr{C} = \{(A_c, B_c, C_c, D_c) \mid J(A_c, B_c, C_c, D_c) \leqslant \gamma\}.$$

Note that $(A_c^{\text{nom}}, B_c^{\text{nom}}, C_c^{\text{nom}}, D_c^{\text{nom}}) \in \hat{\mathscr{C}}$.

We take the Lyapunov performance certificate to be

$$L(A_c, B_c, C_c, D_c, P) = \begin{bmatrix} P - A^T P A - C^T C & -A^T P B - C^T D & 0 \\ -B^T P A - D^T C & \gamma^2 I - B^T P B - D^T D & 0 \\ 0 & 0 & P \end{bmatrix},$$



DOI: 10.1002/rnc

where $A$, $B$, $C$, $D$ are the matrices introduced in (28) and (29). Here $(A_c, B_c, C_c, D_c)$ and $P$ correspond, respectively, to $\theta$ and $v$ introduced in Section 2.2. The condition that $L(A_c, B_c, C_c, P) \succcurlyeq 0$ is equivalent to

$$\begin{bmatrix} A^T PA - P + C^T C & A^T PB + C^T D \\ B^T PA + D^T C & B^T PB + D^T D - \gamma^2 I \end{bmatrix} \preccurlyeq 0,$$
$$P \succcurlyeq 0. \tag{30}$$

As (30) does not depend on $(A_c, B_c, C_c, D_c)$, for a fixed choice of $P$, (3) becomes

$$\hat{\mathscr{C}} = \left\{ (A_c, B_c, C_c, D_c) \left| \begin{bmatrix} A^T PA - P + C^T C & A^T PB + C^T D \\ B^T PA + D^T C & B^T PB + D^T D - \gamma^2 I \end{bmatrix} \preccurlyeq 0 \right. \right\}. \tag{31}$$

Any matrix $P$ that satisfies $L(A_c, B_c, C_c, D_c, P) \succcurlyeq 0$ for $(A_c, B_c, C_c, D_c) \in \mathscr{C}$ is a valid choice. We take $P$ to be the solution of the following optimization problem

$$\begin{aligned} &\text{maximize} \quad \lambda_{\min}(L(A_c, B_c, C_c, D_c, P)) \\ &\text{subject to} \quad L(A_c, B_c, C_c, D_c, P) \succcurlyeq 0. \end{aligned}$$

Here $\lambda_{\min}(L(A_c, B_c, C_c, D_c, P))$ is the minimum eigenvalue of $L(A_c, B_c, C_c, P)$, and $P$ is the variable we are optimizing over. Recall that $A_c$, $B_c$, $C_c$, and $D_c$ are fixed.

We will now show that $\hat{\mathscr{C}} \subseteq \mathscr{C}$. Let $(A_c, B_c, C_c, D_c) \in \hat{\mathscr{C}}$. Consider the Lyapunov function $V: \mathbf{R}^{n_p + n_c} \to \mathbf{R}$ defined as $V(z) = z^T Pz$. For all $x$, $w$,

$$\begin{bmatrix} x \\ w \end{bmatrix}^T \begin{bmatrix} A^T PA - P + C^T C & A^T PB + C^T D \\ B^T PA + D^T C & B^T PB + D^T D - \gamma^2 I \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} \leqslant 0,$$

or equivalently

$$(Ax + Bw)^T P(Ax + Bw) - x^T Px \leqslant \gamma^2 w^T w - (Cx + Dw)^T (Cx + Dw).$$

For all $t$,

$$V(x(t+1)) - V(x(t)) \leqslant \gamma^2 w(t)^T w(t) - z(t)^T z(t).$$

We know that $V(x(0)) = 0$. Since (30) implies that $V(x(T)) \geqslant 0$ and since

$$V(x(T)) - V(x(0) = \sum_{t=0}^{T} V(x(t+1)) - V(x(t)),$$

we deduce that

$$0 \leqslant \sum_{t=0}^{T} V(x(t+1)) - V(x(t)) \leqslant \sum_{t=0}^{T} \gamma^2 w(t)^T w(t) - z(t)^T z(t).$$

Therefore

$$\sum_{t=0}^{T} z(t)^T z(t) \leqslant \gamma^2 \sum_{t=0}^{T} w(t)^T w(t) \tag{32}$$

holds for all $T > 0$. Dividing by $T$ and letting $T$ tend to infinity on both sides of (32), we obtain $\|z\|^2 \leqslant \gamma^2 \|w\|^2$ which implies that the $H_\infty$ norm of the transfer function of the closed-loop system is less than or equal to $\gamma$ as desired.



DOI: 10.1002/rnc

### 5.2. Coefficient range calculation

Let $(l, u)$ be the range of coefficient $(A_c)_{ij}$. Given the equation in (31), problem (2) becomes

$$l = \min\{(A_c)_{ij} | A^T PA - P + C^T C + (A^T PB + C^T D)Q(B^T PA + D^T C) \preccurlyeq 0\},$$

$$u = \max\{(A_c)_{ij} | A^T PA - P + C^T C + (A^T PB + C^T D)Q(B^T PA + D^T C) \preccurlyeq 0\},$$

where $Q = \gamma^2 I - B^T PB - D^T D$. The inequality in (31) is equivalent to

$$\left\| \begin{bmatrix} P^{1/2} A P^{-1/2} & \gamma^{-1} P^{1/2} B \\ C P^{-1/2} & \gamma^{-1} D \end{bmatrix} \right\| \leqslant 1. \tag{33}$$

The method outlined in Section 2.2 can be used to compute $l$ and $u$ by taking

$$Z = \begin{bmatrix} P^{1/2} A P^{-1/2} & \gamma^{-1} P^{1/2} B \\ C P^{-1/2} & \gamma^{-1} D \end{bmatrix}, \quad v = \begin{bmatrix} P^{1/2} \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ e_i \end{bmatrix}, \quad w = \begin{bmatrix} P^{-1/2} \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ e_j \end{bmatrix}, \tag{34}$$

where $e_i$ and $e_j$ are the $i$th and $j$th unit vectors in $\mathbf{R}^{n_c}$ and $(A, B, C, D)$ is the closed-loop matrix associated with $(A_c, B_c, C_c, D_c) \in \hat{\mathscr{C}}$.

The same method can be used to find the ranges of coefficients in $B_c$, $C_c$, and $D_c$. The same formulas can be used but with slightly modified definitions for $v$ and $w$. To find the range of coefficient $(B_c)_{ij}$, use the same definitions for $Z$ and $v$ in (34) but let

$$w = \begin{bmatrix} P^{-1/2} \begin{bmatrix} C_2^T \\ 0 \end{bmatrix} \\ \gamma^{-1} D_{21}^T \end{bmatrix} e_j, \tag{35}$$

where $e_j$ is the $j$th unit vector in $\mathbf{R}^{m_p}$. To find the range of coefficient $(C_c)_{ij}$, use the same definitions for $Z$ and $w$ in (34) but let

$$v = \begin{bmatrix} P^{1/2} \begin{bmatrix} B_2 \\ 0 \end{bmatrix} \\ D_{12} \end{bmatrix} e_i, \tag{36}$$

where $e_i$ is the $i$th unit vector in $\mathbf{R}^{m_c}$. To find the range of coefficient $(D_c)_{ij}$, use the definition for $Z$ given in (34), the definition of $v$ given in (36) and the definition of $w$ given in (34).

### 5.3. Numerical instance

Our example has dimensions $n_p = 5$, $n_c = 4$, $m_p = 2$, $m_c = 2$, and $k = 3$. We generate matrix $A_p$ using the method used to generate $A$ in Section 3.3. The entries of the matrices $B_1$, $B_2$, $C_1$, $C_2$, $D_{11}$, $D_{12}$, $D_{21}$ are IID $\mathscr{N}(0, 1)$.

The nominal controller $(A_c^{\text{nom}}, B_c^{\text{nom}}, C_c^{\text{nom}}, D_c^{\text{nom}})$ is chosen to be the central $H_\infty$ controller, i.e. the one that minimizes the $H_\infty$ norm of the transfer function of the closed-loop system (see, e.g. [29, 30]).

The complexity measures $\phi_i(z)$ are chosen to be $\phi_{\text{frac-bits}}$. The fractional part of each entry of $A_c^{\text{nom}}$, $B_c^{\text{nom}}$, $C_c^{\text{nom}}$, $D_c^{\text{nom}}$ is expressed with 40 bits, requiring a total of 1440 bits, i.e. $\Phi(\theta^{\text{norm}}) =$ 1440 bits. We run the basic algorithm with $\varepsilon = 15\%$, which means that we allow the $H_\infty$ norm of the controller to be up to 15% suboptimal.

The progress of the complexity $\Phi(\theta)$ and percentage deterioration in performance $100(J - J^{\text{nom}})/J^{\text{nom}}$ during the three sample runs of the algorithm are shown in Figure 6.

It is interesting to note that the three sample runs achieve designs with similar complexities but with a varying range of performance degradation (although all within the allowable limit). This



DOI: 10.1002/rnc

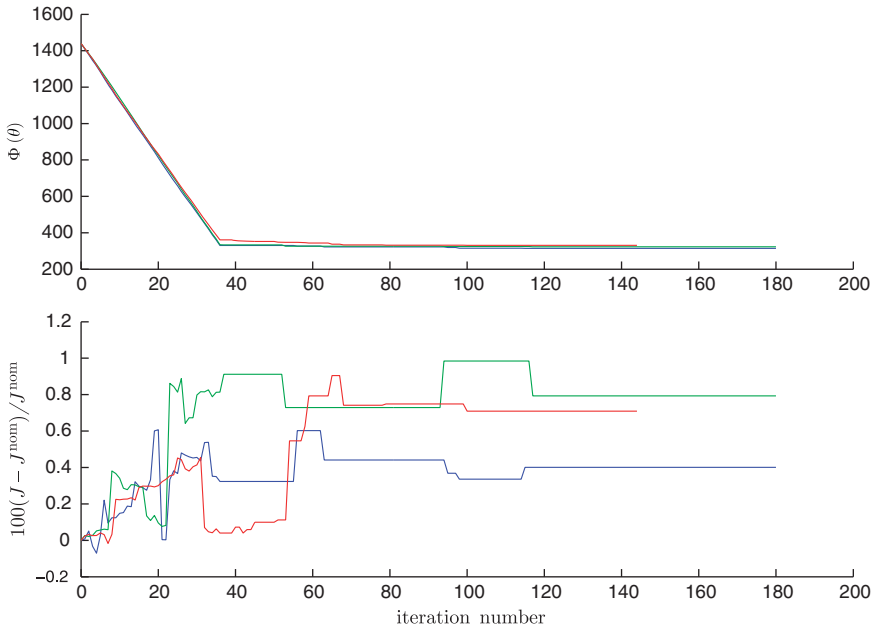


Figure 6. Top: Total number of bits required to express $\theta$ versus iteration number. Bottom: Percentage deterioration in performance versus iteration number.

suggests the following methodology: run the algorithm many times, choose the designs achieving complexity below a specific value, and pick among those the design with the lowest degradation in performance. Applying this idea to the 100 sample runs we obtained, we found that among the designs achieving a complexity less than 260 bits, there exists a design whose performance is degraded by only 1.15%.

The best design after 10 random runs of the algorithm achieves a complexity of $\Phi(\theta)=270$ bits with a cost of $J(\theta)=1.002J(\theta^{\text{nom}})$. The best design after 100 random runs of the algorithm achieves a complexity of $\Phi(\theta)=247$ bits and $J(\theta)=1.030J(\theta^{\text{nom}})$.

*5.3.1. Multiple random instances.* We now generate 100 random instances of matrices $A_p$, $B_1$, $B_2$, $C_1$, $C_2$, $D_{11}$, $D_{12}$, $D_{21}$, with other data remaining the same: $n_p=5$, $n_c=4$,$m_p=2$, $m_c=2$, $k=3$, and $\varepsilon=15\%$. For each instance, we compute $A^{\text{nom}}$, $B^{\text{nom}}$, $C^{\text{nom}}$, $D^{\text{nom}}$ and express the fractional part of each of their entries with 40 bits.

For each instance, we record the best design after 10 random runs of the algorithm. The complexities of the best designs range between 170 and 403 bits, with a mean of 274.21 bits and a standard deviation of 59.54 bits. The performance degradations of the best designs ranged between 0.04 and 14.52%. The algorithm takes an average of 1.5 passes over the variables to converge.

We compare these results with the simple approach of truncating the binary expansions of the coefficients to precision $2^{-q}$ (i.e. with $q$ bits in their fractional parts), choosing $q$ as small as possible while still maintaining $\theta\in\mathscr{C}$. For the 100 random instances that were generated, $q$ varied between 8 and 40 bits per coefficient with a mean of 18.72 bits. In contrast, our algorithm gave controllers whose coefficient complexity was between 4.77 and 11.19 bits per coefficient, with a mean of 7.63 bits per coefficient.

## 6. DYNAMIC CONTROLLER WITH SATURATION

We have been dealing with specifications on linear closed-loop systems until now. The methodology that we covered can be easily extended to the case where we are given specifications on closed-loop



DOI: 10.1002/rnc

systems with nonlinear dynamics. We illustrate this idea through the following example where the plant is given by

$$x_p(t+1)=A_p x_p(t)+B_p u(t), \quad y(t)=C_p x_p(t),$$

and is controlled by a dynamic controller given by the nonlinear dynamical system

$$x_c(t+1)=\mathbf{sat}(A_c x_c(t)+B_c y(t)),$$

$$u(t)=\mathbf{sat}(C_c x_c(t)),$$

where $x_p(t)\in\mathbf{R}^{n_p}$, $u(t)\in\mathbf{R}^{m_c}$, $y(t)\in\mathbf{R}^{m_p}$, $A_p\in\mathbf{R}^{n_p\times n_p}$, $B_p\in\mathbf{R}^{n_p\times m_c}$, $C_p\in\mathbf{R}^{m_p\times n_p}$, $x_c(t)\in\mathbf{R}^{n_c}$, $A_c\in\mathbf{R}^{n_c\times n_c}$, $B_c\in\mathbf{R}^{n_c\times m_p}$, and $C_c\in\mathbf{R}^{m_c\times n_c}$, and $\mathbf{sat}:\mathbf{R}\to\mathbf{R}$ is defined as

$$\mathbf{sat}(z)=\begin{cases} z & \text{if } |z|\leqslant 1 \\ -1 & \text{if } z<-1 \\ 1 & \text{if } z>1. \end{cases}$$

The closed-loop system is a nonlinear dynamical system of the form

$$\begin{aligned} x(t+1) &= Ax(t)+Bp(t), \\ q(t) &= Cx(t), \\ p_i &= \mathbf{sat}(q_i), \quad i=1,\ldots,m_c+n_c, \end{aligned} \tag{37}$$

where

$$\begin{aligned} x(t)&=\begin{bmatrix} x_p(t) \\ x_c(t) \end{bmatrix}, \quad A=\begin{bmatrix} A_p & 0 \\ 0 & 0 \end{bmatrix}, \\ B&=\begin{bmatrix} B_p & 0 \\ 0 & I \end{bmatrix}, \quad C=\begin{bmatrix} 0 & C_c \\ B_c C_p & A_c \end{bmatrix}. \end{aligned} \tag{38}$$

### *6.1. Admissible controllers*

We define the decay rate of the closed-loop system to be the infinimum of $\alpha$ for which every trajectory of the closed-loop system satisfies

$$\|x(t)\|\leqslant M\alpha^t\|x(0)\|,$$

for all $t$ and for some $M>0$. Here $M$ is a constant that depends on the trajectory.

The performance measure $J$ is chosen to be the decay rate of the closed-loop system. Unlike the other cases we presented, it is very difficult to compute the decay rate $J$ of the closed-loop system described in (37). However, as we shall subsequently see, an upper bound on the decay rate can be found using a Lyapunov method.

We are given nominal design $(A_c^{\mathrm{nom}}, B_c^{\mathrm{nom}}, C_c^{\mathrm{nom}})$ with a decay rate less than $\rho$.

A controller $(A_c, B_c, C_c)$ is admissible if the decay rate of the closed-loop system is less than a given rate $\alpha$, where $\alpha=(1+\varepsilon)\rho$ and $\varepsilon$ is a given positive number. The set of admissible controller designs is then

$$\mathscr{C}=\{(A_c, B_c, C_c)\,|\,J(A_c, B_c, C_c)\leqslant\alpha\}.$$

We choose the Lyapunov performance certificate to be

$$L(A_c, B_c, C_c, P, \tau_1, \ldots, \tau_{m_c+n_c})=\begin{bmatrix} \alpha^2 P-A^T PA & -A^T PB-(1/2)C^T D & 0 & 0 \\ -B^T PA-(1/2)DC & D-B^T PB & 0 & 0 \\ 0 & 0 & P-I & 0 \\ 0 & 0 & 0 & D \end{bmatrix}$$



DOI: 10.1002/rnc

where $D=\mathbf{diag}(\tau_1,\ldots,\tau_{m_c+n_c})$ and $A$, $B$, and $C$ are the matrices introduced in (38). Here $(A_c, B_c, C_c)$ correspond to $\theta$ and $(P, \tau_1, \ldots, \tau_{m_c+n_c})$ correspond to $v$ introduced in Section 2.2. The condition that $L(A_c, B_c, C_c, P, \tau_1, \ldots, \tau_{m_c+n_c})\succcurlyeq 0$ is equivalent to

$$\begin{bmatrix} A^T PA-\alpha^2 P & A^T PB+(1/2)C^T D \\ B^T PA+(1/2)DC & B^T PB-D \end{bmatrix} \preccurlyeq 0,$$

$$P \succcurlyeq I, \tag{39}$$

$$D \succcurlyeq 0. \tag{40}$$

Since (39) and (40) do not depend on $(A_c, B_c, C_c)$ for a fixed choice of $P$ and $D$, (3) becomes

$$\hat{\mathscr{C}}=\left\{(A_c, B_c, C_c)\,|\begin{bmatrix} A^T PA-\alpha^2 P & A^T PB+(1/2)C^T D \\ B^T PA+(1/2)DC & B^T PB-D \end{bmatrix}\preccurlyeq 0\right\}. \tag{41}$$

Any matrix $P$ and scalars $\tau_1, \ldots, \tau_{m_c+n_c}$ that satisfy $L(A_c, B_c, C_c, P, D)\succcurlyeq 0$ for given $(A_c, B_c, C_c)\in\mathscr{C}$ are a valid choice. We take $(P, D)$ to be the solution of the following optimization problem

$$\begin{aligned} \text{maximize} \quad & \lambda_{\min}(L(A_c, B_c, C_c, P, D)) \\ \text{subject to} \quad & L(A_c, B_c, C_c, P, D)\succcurlyeq 0, \\ & D=\mathbf{diag}(\tau_1, \ldots, \tau_{m_c+n_c}). \end{aligned}$$

Here $\lambda_{\min}(L(A_c, B_c, C_c, P))$ is the minimum eigenvalue of $L(A_c, B_c, C_c, P)$, and $P, \tau_1, \ldots, \tau_{m_c+n_c}$ are the variables that we are optimizing over. Recall that $A_c$, $B_c$, and $C_c$ are fixed here.

We will now show that $\hat{\mathscr{C}}\subseteq\mathscr{C}$. Let $(A_c, B_c, C_c)\in\hat{\mathscr{C}}$. Consider the Lyapunov function $V:\mathbf{R}^{n_p+n_c}\to\mathbf{R}$ defined as $V(z)=z^T Pz$. Note that $p_i$ and $q_i$ must satisfy $(p_i-q_i)p_i\leqslant 0$ for $i=1,\ldots,m_c+n_c$. Recalling that $q=Cx$, this means that, for all $x$ and $p$,

$$\begin{bmatrix} x \\ p \end{bmatrix}^T \begin{bmatrix} 0 & -(1/2)C^T D \\ -(1/2)DC & D \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix} \leqslant 0. \tag{42}$$

Since $(A_c, B_c, C_c)\in\hat{\mathscr{C}}$, the following holds for all $x$, $p$,

$$\begin{bmatrix} x \\ p \end{bmatrix}^T \begin{bmatrix} A^T PA-\alpha^2 P & A^T PB+(1/2)C^T D \\ B^T PA+(1/2)D^T C & B^T PB-D \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix} \leqslant 0. \tag{43}$$

It follows from (42) and (43) that

$$\begin{bmatrix} x \\ p \end{bmatrix}^T \begin{bmatrix} A^T PA-\alpha^2 P & A^T PB \\ B^T PA & B^T PB \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix} \leqslant \begin{bmatrix} x \\ p \end{bmatrix}^T \begin{bmatrix} 0 & -(1/2)C^T D \\ -(1/2)DC & D \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix} \leqslant 0.$$

This inequality implies that

$$(Az+Bp)^T P(Az+Bp)\leqslant\alpha^2 z^T Pz.$$

Therefore we have $V(x(t+1))\leqslant\alpha^2 V(x(t))$ for all $t\geqslant 0$. This implies that the decay rate of the closed-loop system is less than $\alpha$, as demonstrated in Section 3.1.



DOI: 10.1002/rnc

### 6.2. Coefficient range calculation

The inequality in (41) is an LMI in $P$ and $D$, but in the given form it is a convex matrix quadratic inequality in $(A, B, C)$. It can easily be shown that it is actually equivalent to the LMI in $(A, B, C)$:

$$\begin{bmatrix} \alpha^2 P & -(1/2)C^T D & A^T \\ -(1/2)DC & D & B^T \\ A & B & P^{-1} \end{bmatrix} \succcurlyeq 0. \tag{44}$$

Since $(A, B, C)$ are linear in $(A_c, B_c, C_c)$, (44) is also an LMI in $(A_c, B_c, C_c)$. The range of coefficient of $(A_c)_{ij}$ can, therefore, be found from (4) by the eigenvalue computation method outlined in Section 2.2 with $L(\theta, v$ given by (44) and

$$L_i = \begin{bmatrix} 0 & -(1/2)vw^T D & 0 \\ -(1/2)Dwv^T & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where

$$w = \begin{bmatrix} 0 \\ e_i \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ e_j \end{bmatrix},$$

and $e_i$ and $e_j$ are, respectively, the $i$th and $j$th unit vectors in $\mathbf{R}^{n_c}$.

The same method can be used to find the ranges of coefficients in $B_c$ and $C_c$ but with slightly modified definitions for $v$ and $w$. To find the range of coefficient $(B_c)_{ij}$, take

$$w = \begin{bmatrix} 0 \\ e_i \end{bmatrix}, \quad v = \begin{bmatrix} C_p^T e_j \\ 0 \end{bmatrix},$$

where $e_i$ is the $i$th unit vector in $\mathbf{R}^{n_c}$ and $e_j$ is the $j$th unit vector in $\mathbf{R}^{m_p}$. To find the range of coefficient $(C_c)_{ij}$, take

$$w = \begin{bmatrix} e_i \\ 0 \end{bmatrix}, \quad v = \begin{bmatrix} 0 \\ e_j \end{bmatrix},$$

where $e_i$ is the $i$th unit vector in $\mathbf{R}^{m_p}$ and $e_j$ is the $j$th unit vector in $\mathbf{R}^{n_c}$.

### 6.3. Numerical instance

We test the proposed method on the problem described in Section 4.3 where the plant is described by (25) and where the nominal controller is described by (26) and (27). The decay rate $\rho$ of the closed-loop system is the minimum value of $\alpha$ for which there exists matrices $P$ and $D$ that satisfy

$$L(A_c^{\text{nom}}, B_c^{\text{nom}}, C_c^{\text{nom}}, P, \tau, \ldots, \tau_{m_c+n_c}) \succcurlyeq 0.$$

Our example has dimensions $n_p = 5$, $m_c = 2$, and $m_p = 2$. The plant matrix $A_p$ is randomly generated using the same method used to generate $A$ in Section 3.3. The entries of $B_p$ and $C_p$ are IID $\mathcal{N}(0, 1)$. The matrices $A_c^{\text{nom}}$, $B_c^{\text{nom}}$, and $C_c^{\text{nom}}$ are then computed using the formulas presented in Section 4.3.

The complexity measures $\phi_i(z)$ are chosen to be $\phi_{\text{frac-bits}}$. The fractional part of each entry of $A_c^{\text{nom}}$, $B_c^{\text{nom}}$, and $C_c^{\text{nom}}$ is expressed with 40 bits, requiring a total of 1800 bits, i.e. $\Phi(\theta^{\text{nom}}) = 1800$ bits. The decay rate of the nominal system is found to be 0.9326. We run the algorithm with $\varepsilon = 5\%$.

The best design after 10 random runs of the algorithm achieves a complexity of $\Phi(\theta) = 35$ bits, with a cost of $J(\theta) = 0.9770 J(\theta^{\text{nom}})$. The best design found after 100 random runs of the algorithm



DOI: 10.1002/rnc

achieves a complexity of $\Phi(\theta)=18$ bits and $J(\theta)=0.9288J(\theta^{\text{nom}})$ (which is *smaller* than the value for our nominal controller).

*6.3.1. Multiple random instances.* We generate 100 random instances of matrices $A_p$, $B_p$, $C_p$, with other data remaining the same: $n_p=5$, $m_c=2$, $m_p=2$, $Q=I$, $R=I$, and $\varepsilon=5\%$. For each instance, we compute $A^{\text{nom}}$, $B^{\text{nom}}$, $C^{\text{nom}}$ and express the fractional part of each of their entries with 40 bits (i.e. $\phi(\theta^{\text{nom}})=1800$ bits).

For each instance, we record the best design after 10 random runs of the algorithm. The complexities of the best designs range between 10 and 52 bits, with a mean of 22.85 bits and a standard deviation of 9.6 bits. Moreover, the performance degradations of the best designs ranged between $-9.21$ and 0.61% and the algorithm takes an average of six passes over the variables to converge. (Here negative values mean that the objective value of our truncated controller is smaller than that of the nominal controller).

We compare these results with the simple approach of truncating the binary expansions of the coefficients to precision $2^{-q}$ (i.e. with $q$ bits in their fractional parts), choosing $q$ as small as possible while still maintaining $\theta\in\mathscr{C}$. For the 100 random instances that were generated, $q$ varied between 3 and 8 bits per coefficient with a mean of 5.55 bits. In contrast, our algorithm gave aggressively truncated controllers whose complexity was between 0.22 and 1.16 bits per coefficient, with a mean of 0.51 bits per coefficient.

## ACKNOWLEDGEMENTS

This work was funded in part by Focus Center Research Program Center for Circuit and System Solutions (www.c2s2.org), under contract 2003-CT-888, by AFOSR grant AF F49620-01-1-0365, by NSF grant ECS-0423905, by NSF grant 0529426, by DARPA/MIT grant 5710001848, by AFOSR grant FA9550-06-1-0514, DARPA/Lockheed contract N66001-06-C-2021, and by AFOSR/Vanderbilt grant FA9550-06-1-0312.

## REFERENCES

1. Avenhaus E. On the design of digital filters with coefficients of limited word length. *IEEE Transactions on Audio and Electroacoustics* 1972; **20**(3):206–212.
2. Brglez F. Digital filter design with short word-length coefficients. *IEEE Transactions on Circuits and Systems* 1978; **25**(12):1044–1050.
3. Gray A, Markel J. Quantization and bit allocation in speech processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 1976; **24**(6):459–473.
4. Rink R, Chong H. Performance of state regulator systems with floating point computation. *IEEE Transactions on Automatic Control* 1979; **14**:411–412.
5. Lim Y, Yu Y. A successive reoptimization approach for the design of discrete coefficient perfect reconstruction lattice filter bank. *IEEE International Symposium on Circuits and Systems*, vol. 2, 2000; 69–72.
6. Samueli H. An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients. *IEEE Transactions on Circuits and Systems* 1989; **36**(7):1044–1047.
7. Lee J, Chen C, Lim Y. Design of discrete coefficient FIR digital filters with arbitrary amplitude and phase response. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing* 1993; **40**(7):444–448.
8. Lim Y, Yu Y. A width-recursive depth-first tree search approach for the design of discrete coefficient perfect reconstruction lattice filter bank. *IEEE Transactions on Circuits and Systems II* 2003; **50**:257–266.
9. Frossard P, Vandergheynst P, Figueras R, Kunt M. A posteriori quantization of progressive matching pursuit streams. *IEEE Transactions on Signal Processing* 2004; **52**(2):525–535.
10. Lim Y, Parker S. Discrete coefficient FIR digital filter design based upon an LMS criteria. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 1983; **30**(10):723–739.
11. Chen S, Wu J. The determination of optimal finite precision controller realizations using a global optimization strategy: a pole-sensitivity approach. In *Digital Controller Implementation and Fragility: A Modern Perspective*, Istepanian R, Whidborne J (eds), Chapter 6. Springer: New York, 2001; 87–104.
12. Benvenuto N, Marchesi M, Orlandi G, Piazza F, Uncini A. Finite wordlength digital filter design using an annealing algorithm. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1989; 861–864.
13. Istepanian R, Whidborne J. Multi-objective design of finite word-length controller structures. *Proceedings of the Congress on Evolutionary Computation*, vol. 1, 1999; 61–68.



DOI: 10.1002/rnc

14. Gentili P, Piazza F, Uncini A. Efficient genetic algorithm design for power-of-two FIR filters. *International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*), 1995; 1268–1271.
15. Traferro S, Capparelli F, Piazza F, Uncini A. Efficient allocation of power-of-two terms in FIR digital filter design using tabu search. *International Symposium on Circuits and Systems* (*ISCAS*), 1999; III-411–414.
16. Li G, Covers M. Optimal finite precision implementation of a state-estimate feedback controller. *IEEE Transactions on Circuits and Systems* 1990; **37**(12):1487–1498.
17. Wah B, Shang Y, Wu Z. Discrete Lagrangian methods for optimizing the design of multiplierless QMF banks. *IEEE Transactions on Circuits and Systems II* 1999; **46**(9):1179–1191.
18. Kodek D. Design of optimal finite wordlength FIR digital filters using integer programming techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 1980; **28**(3):304–308.
19. Lim Y, Parker S. FIR filter design over a discrete powers-of-two coefficient space. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 1983; **31**(3):583–591.
20. Barua S, Kotteri K, Bell A, Carletta J. Optimal quantized lifting coefficients for the 9/7 wavelet. *IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), vol. 5, 2004; V-193–196.
21. Liu K, Skelton R, Grigoriadis K. Optimal controllers for finite wordlength implementation. *IEEE Transactions on Automatic Control* 1992; **37**(9):1294–1304.
22. Molchanov A, Bauer P. Robust stability of digital feedback control systems with floating point arithmetic. *Proceedings of the 34th Conference on Decision and Control* 1995; 4251–4258.
23. Boyd S, Vandenberghe L. *Convex Optimization*. Cambridge University Press: Cambridge, 2004.
24. Boyd S, El Ghaoui L, Feron E, Balakrishnan V. *Linear Matrix Inequalities in Systems and Control Theory*. SIAM Books: Philadelphia, 1994.
25. Boyd S, Barratt C. *Linear Controller Design*: *Limits of Performance*. Prentice-Hall: Englewood Cliffs, NJ, 1991.
26. Dullerud G, Paganini F. *A Course in Robust Control Theory*: *A Convex Approach*. Springer: New York, 2000.
27. Vandenberghe L, Boyd S. Semidefinite programming. *SIAM Review* 1996; **38**(1):45–95.
28. Vandenberghe L, Boyd S, Wu S. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications* 1998; **19**(2):499–533.
29. Gahinet P. Explicit controller formulas for LMI-based $H_\infty$ synthesis. *American Control Conference*, vol. 3, 1994; 2396–2400.
30. Gahinet P, Apkarian P. A linear matrix inequality approach to $H_\infty$ control. *International Journal on Robust and Nonlinear Control* 1994; **4**:421–448.



DOI: 10.1002/rnc