JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 23, No. 5, September–October 2000

State-Space Approach to l_1-Optimal Robust Tracking

Craig N. Scott* and Lincoln A. Wood†
Royal Melbourne Institute of Technology, Melbourne, Victoria 3001, Australia

In most l_1 optimization work, the plant model is typically expressed as a transfer function matrix, often in the more specialized coprime factorization form. The derivation of the optimization problem can be complicated, at times requiring an understanding of advanced mathematical concepts. State-space approaches to control problems, however, frequently result in intuitively simpler methods. A state-space alternative to the transfer function methods of our previous work is presented. The controllers generated are optimized for a specific tracking maneuver and are also designed to minimize the sensitivity to disturbances, sensor noise, and modeling errors. A block diagram similar to the model reference approach is used, which has the potential to give better controllers than the conventional block diagram.

Nomenclature

D_C = denominator feedback part of controller
d = denominator of plant right coprime factorization
I_j = $(j \times j)$ identity matrix
K = $-D_C^{-1} N_{C_2}$
N_{C_1} = feedforward part of controller
N_{C_2} = numerator feedback part of controller
n = numerator of plant right coprime factorization
P_0 = linear plant model that can be described by the discrete state space matrices A_P, B_P, C_P, and D_P; nd^{-1}
p = number of plant inputs
q = number of plant outputs
u = plant input
V = $(W_3^{-1} P_0 W_2)^T$
W = $(W_2^{-1} W_4)^T$
W_{2-4} = weighting filters on the respective inputs; for multivariable plants, these are diagonal matrices of weighting filters
w_1 = command reference (sometimes abbreviated to w)
w_2 = disturbance input
w_3 = sensor noise input
w_4 = modeling error input
y = plant output
ϕ = tracking error, $w_1 - y$

I. Introduction

THE literature focusing on l_1-optimal control has been growing at a healthy rate in recent times. There is now available to the designer a choice of methods for designing controllers using a variety of mathematical tools. Recently, the authors showed another formulation[1,2] that gave the designer a complete method for generating optimally robust controllers without sacrificing optimal performance. Dahleh and Pearson,[3,4] Diaz-Bobillo and Dahleh,[5] and Dahleh and Diaz-Bobillo[6] show the most common methods of setting out an l_1-optimization problem. Other methods have appeared[7–9] using a different, simpler form for the constraint set by making use of closed-loop relationships. All of these methods use a transfer function matrix to represent the plant model.

This paper provides a simpler approach by using the state-space matrices of the plant and the weighting filters of Ref. 2. The formulation makes use of less demanding mathematical tools than most other l_1 approaches, with coprime factorizations only being used in the derivation and not in the final optimization. The designer is mostly required to use only basic matrix algebra in forming the final optimization problem, which is carried out in two distinct and independent parts: tracking and regulation. The first determines the optimal behavior of the system in an ideal environment where there are no disturbances, sensor noise, or modeling errors. The second minimizes the sensitivity of the system to these environmental effects.

It is also shown that the conventional two-degree-of-freedom (DOF) controller format used in Ref. 2, and analyzed in detail in Ref. 10, is equivalent to a model reference block diagram. This is exploited here to eliminate a term from the feedforward path of the controller, thereby removing the need for numerical deconvolution when forming the controller from the optimization results. It is also shown that this relaxes the constraints associated with any nonminimum phase zeros of the command reference and has the potential to permit better controllers in special circumstances.

The reason for seeking other ways to express the optimization constraints, aside from simplifying the formulations, is to find more numerically stable alternatives. It has been observed that transfer function methods can generate problems that are numerically difficult to solve, and so more reliable methods are needed. The formulations presented in this paper show that there are significantly different ways of enforcing the necessary constraint sets. The state-space format of the tracking optimization shows excellent numerical reliability. The regulation optimization can be expressed in a number of different ways, one of which has been chosen to show how the constraints can be manipulated. The state-space formulation chosen to illustrate the concepts is not particularly numerically well-behaved, but does present techniques that can be applied to form other constraint sets that may well perform better.

II. Block Diagram

Figure 1 shows the conventional layout of a system with a two-DOF controller (see Ref. 2 for details). Because all terms are linear, the effects of w_1 can be separated from the effects of w_{2-4}, as shown in Fig. 2. All terms contained in the bottom part of Fig. 2 (labeled the optimal part) are either known or determined by an optimization prior to implementing the controller. This is because they have no dependence on the unknown environmental effects w_{2-4}. Therefore, it is possible to omit most of the optimal part and just inject u^* and y^* directly into the top section (the regulator). This also reduces any numerical roundoff that may occur with high-order N_{C_1}, N_{C_2}, and D_C terms. Tidying up the result leads to the simpler block diagram of Fig. 3. By eliminating N_{C_1} and injecting u^* and y^* in this way, unstable pole-zero cancellation between N_{C_1} and w_1 is no longer prohibited because it cannot lead to internal instability of the new system.

The regulator (feedback) parts of the controller N_{C_2} and D_C have been lumped together into the K term, whereas the optimal tracking (feedforward) term N_{C_1} has been replaced by the u^* and y^* terms. Note that y^* is added to the plant output after the sensor noise because the true plant output y is not available to the controller. Placing

Received 12 July 1999; accepted for publication 9 February 2000.

*Doctoral Candidate, Department of Aerospace Engineering, Wackett Aerospace Centre, GPO Box 2476V.

†Adjunct Professor of Aerospace Engineering, Department of Aerospace Engineering, Wackett Aerospace Centre, GPO Box 2476V.

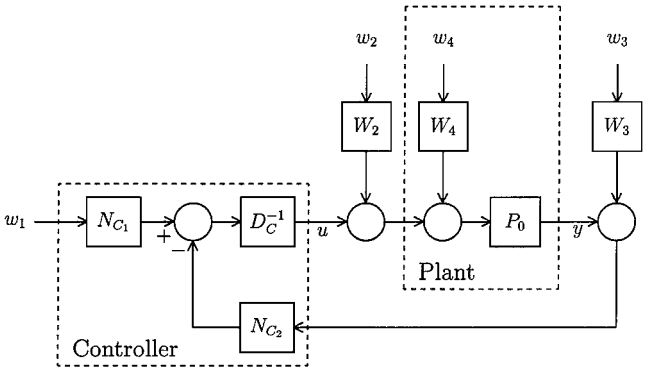


Fig. 1 Conventional two-DOF controller.

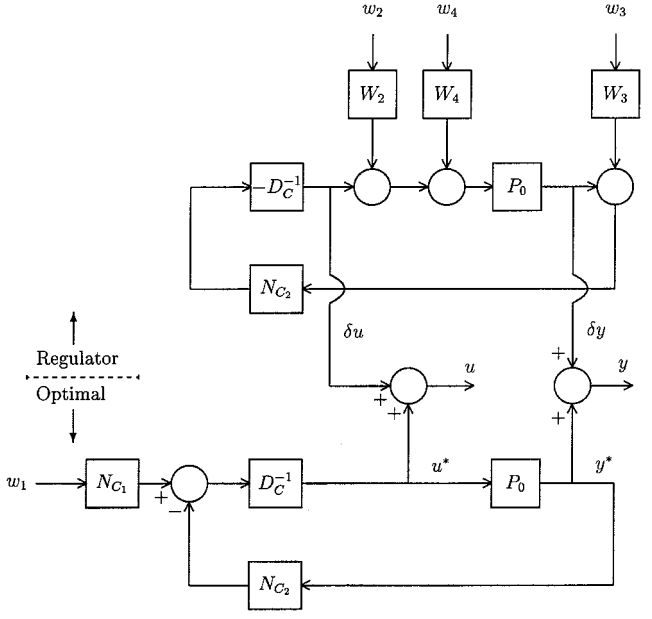


Fig. 2 Separated form of block diagram.

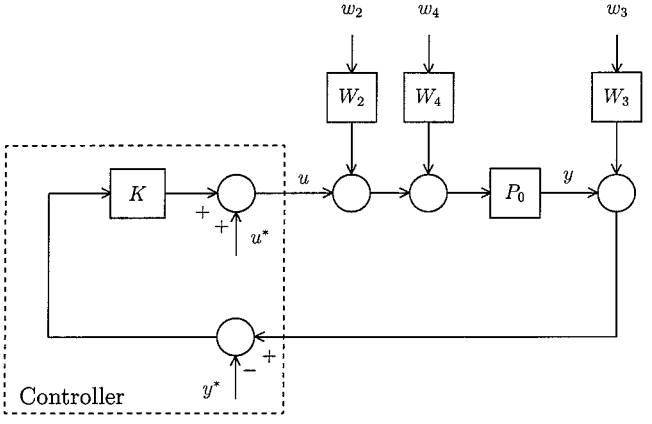


Fig. 3 Alternate two-DOF controller.

y^* after rather than before the sensor noise makes no difference because the noise is added as a linear operation.

Figure 3 is effectively the same as the model reference approach. The only difference is that y^* is known in advance instead of being calculated from putting u^* through a plant model online. This opens up the possibility of using mixed controller types. For example, u^* and y^* may be formed using an l_1 technique, but the feedback part of the controller K may instead be formed using some other method, such as H_∞, fuzzy logic, etc.

III. Tracking Optimization

The tracking problem amounts to finding a valid (u^*, y^*) pair [or, equivalently, a (u^*, ϕ^*) pair]. In Ref. 2, this was achieved by solving a linear program where the constraints were derived using discrete transfer function matrices. The constraints ensured that, for the specified reference (w_1, hereafter simply referred to as w to avoid later confusion), the optimal u generated a compatible tracking error ϕ. Instead of using transfer function matrices and coprime factorizations, this can easily be enforced using only the state-space matrices of the plant. Consider the ith time step of the usual discrete state-space equations

$$y_i = C_P x_i + D_P u_i \tag{1}$$

$$x_{i+1} = A_P x_i + B_P u_i \tag{2}$$

For the optimal solution, this can be written as

$$\phi_i^* = w_i - C_P x_i - D_P u_i^* \tag{3}$$

$$x_{i+1} = A_P x_i + B_P u_i^* \tag{4}$$

When we expand these equations for the first f time steps, we get the following:

$$\begin{Bmatrix} \phi_0^* \\ \phi_1^* \\ \vdots \\ \phi_{f-1}^* \end{Bmatrix} = \begin{Bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{f-1} \end{Bmatrix} - \begin{bmatrix} C_P & & & \\ & C_P & & \\ & & \ddots & \\ & & & C_P \end{bmatrix} \begin{Bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{f-1} \end{Bmatrix}$$

$$- \begin{bmatrix} D_P & & & \\ & D_P & & \\ & & \ddots & \\ & & & D_P \end{bmatrix} \begin{Bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{f-1}^* \end{Bmatrix} \tag{5}$$

$$\begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ x_f \end{Bmatrix} = \begin{bmatrix} A_P & -I & & & \\ & A_P & -I & & \\ & & \ddots & \ddots & \\ & & & A_P & -I \\ & & & & A_P \end{bmatrix} \begin{Bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{f-1} \end{Bmatrix}$$

$$+ \begin{bmatrix} B_P & & & \\ & B_P & & \\ & & \ddots & \\ & & & B_P \end{bmatrix} \begin{Bmatrix} u_0^* \\ u_1^* \\ \vdots \\ u_{f-1}^* \end{Bmatrix} \tag{6}$$

Note that if A_P is larger than a (1×1) matrix, each $\boldsymbol{x}_i$ is a vector. For multi-input/multi-output (MIMO) systems, $\boldsymbol{\phi}_i^*$, $\boldsymbol{w}_i$, and $\boldsymbol{u}_i^*$ will also be vectors. For instance, for a plant with p inputs and q outputs, they will have the form

$$\boldsymbol{\phi}_i^* = \begin{Bmatrix} \phi_{1_i}^* \\ \phi_{2_i}^* \\ \vdots \\ \phi_{q_i}^* \end{Bmatrix}, \qquad \boldsymbol{w}_i = \begin{Bmatrix} w_{1_i} \\ w_{2_i} \\ \vdots \\ w_{q_i} \end{Bmatrix}, \qquad \boldsymbol{u}_i^* = \begin{Bmatrix} u_{1_i}^* \\ u_{2_i}^* \\ \vdots \\ u_{p_i}^* \end{Bmatrix} \tag{7}$$

The subscript refers to the input/output channel, whereas the subscript refers to the time step of interest. It will be convenient to write Eqs. (5) and (6) in the more condensed form of Eqs. (8) and (9) where each term is the appropriate vector (denoted by boldface with tilde) or matrix (boldface font) of the expanded state-space equations:

$$\tilde{\boldsymbol{\phi}} = \tilde{\boldsymbol{w}} - \boldsymbol{C}\tilde{\boldsymbol{x}} - \boldsymbol{D}\tilde{\boldsymbol{u}} \tag{8}$$

$$\tilde{\boldsymbol{v}} = \boldsymbol{A}\tilde{\boldsymbol{x}} + \boldsymbol{B}\tilde{\boldsymbol{u}} \tag{9}$$

There may be occasions where the actuator has a steady-state component. This can be included in a similar way to the transfer function formulation by setting $\boldsymbol{u}^* = \bar{\boldsymbol{u}} + \boldsymbol{u}_{SS}$. It is not necessary to

explicitly include a step function on $\boldsymbol{u}_{SS}$ because it is assumed that $\boldsymbol{u}^*$ is zero before the initial time step anyway. The constraints may, therefore, be written as

$$\begin{bmatrix} I & \boldsymbol{D} & \boldsymbol{C} \\ & \boldsymbol{B} & \boldsymbol{A} \end{bmatrix} \begin{Bmatrix} \tilde{\phi} \\ \tilde{u} \\ \tilde{x} \end{Bmatrix} = \begin{Bmatrix} \tilde{w} - \tilde{u}_{SS} \\ \tilde{v} \end{Bmatrix} \quad \text{where} \quad \tilde{u}_{SS} = \begin{Bmatrix} D_P u_{SS} \\ D_P u_{SS} \\ \vdots \end{Bmatrix} \tag{10}$$

where I is of the appropriate dimension. Equation (9) requires a little more attention from the designer, particularly the initial and final states, $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$. Generally, the initial state will be zero, but there is no reason why it cannot be something else if the designer so wishes. The final state depends on the maneuver specified by $\boldsymbol{w}$. It is essential that $\boldsymbol{x}_f$ is consistent with $\boldsymbol{w}$, because it is usually assumed that $\phi^* = 0$ and $\boldsymbol{u}^* = \boldsymbol{u}_{SS}$ after time f. If $\boldsymbol{w}$ is a finite length sequence (or matrix of sequences in the multi-output case), $\boldsymbol{x}_f$ should be a zero vector. This will enforce zero steady-state actuator and tracking error. If $\boldsymbol{w}$ has a nonzero steady-state component, then $\boldsymbol{x}_f$ needs to be chosen to match it, taking into consideration any steady-state actuator, $\boldsymbol{u}_{SS}$, which may also be needed. The example in Ref. 2 shows how both $\boldsymbol{x}_f$ and $\boldsymbol{u}_{SS}$ can be obtained from $\boldsymbol{w}$ and the state-space matrices of the plant.

The linear program is completed by defining an objective that is essentially the same as the transfer function version. For a plant with p inputs and q outputs, the objective can be expressed as

$$\min \left\{ \kappa_1 \sum_{j=1}^{q} \|\tilde{\phi}_j\|_1 + \kappa_2 \sum_{j=1}^{p} \|\tilde{\bar{u}}_j\|_1 \right\} \tag{11}$$

Converting the norms into linear operations on bounded variables is handled in exactly the same way as in Ref. 2.

IV. Regulation Optimization

In Ref. 2, the regulation optimization was formulated using a coprime factorization of the plant. This allowed the problem to be expressed in terms of transfer function matrices relating the environmental inputs w_{2-4} and the optimal u^* and ϕ^* to the actuator demands and tracking error. This was expressed in a form equivalent to the following:

$$u = u^* + u_2 w_2 + u_3 w_3 + u_4 w_4 \tag{12}$$

$$\phi = \phi^* + \phi_2 w_2 + \phi_3 w_3 + \phi_4 w_4 \tag{13}$$

Each u_i or ϕ_i term is a matrix for MIMO plants. Extracting the necessary information from Ref. 2, the constraints can be summarized as follows:

$$u_2 = (dD_C - I)W_2 \tag{14}$$

$$u_3 = -dN_{C_2} W_3 \tag{15}$$

$$u_4 = (dD_C - I)W_4 \tag{16}$$

$$\phi_2 = -nD_C W_2 \tag{17}$$

$$\phi_3 = nN_{C_2} W_3 \tag{18}$$

$$\phi_4 = -nD_C W_4 \tag{19}$$

$$N_{C_2} n + D_C d = I \tag{20}$$

$$\|u_4\|_1 \leq 1 \tag{21}$$

$$N_{C_2}\big|_{z^0} = 0 \tag{22}$$

When we note that $P_0 = nd^{-1}$, and that W_{2-4} are diagonal matrices, performing simple algebraic manipulation allows N_{C_2} and D_C to be eliminated and Eq. (20) to be removed. The constraints then reduce to

$$\phi_2 + P_0 u_2 = -P_0 W_2 \tag{23}$$

$$\phi_3 + P_0 u_3 = 0 \tag{24}$$

$$\phi_4 + P_0 u_4 = -P_0 W_4 \tag{25}$$

$$u_4 - u_2 W_2^{-1} W_4 = 0 \tag{26}$$

$$u_2 - u_3 W_3^{-1} P_0 W_2 = 0 \tag{27}$$

$$\|u_4\|_1 \leq 1 \tag{28}$$

$$u_3|_{z^0} = 0 \tag{29}$$

To see how Eq. (29) enforces Eq. (22), assume that Eq. (29) holds and that the plant P_0 has N delays. By Eq. (24), ϕ_3 must have $(N+1)$ delays. d is derived from a coprime factorization and is not permitted to possess a delay, whereas n has N delays. Inspection of Eq. (18) reveals that if W_3 is assumed to have no delay (easily enforcable by the designer), then N_{C_2} must have a single delay. Therefore, Eq. (29) is both necessary and sufficient to enforce Eq. (22).

These constraints could be put into linear program form using transfer function matrices, but they can also be expressed in terms of state-space matrices. Consider Eq. (23), where ϕ_2, u_2, and W_2 are all matrices (which would imply that the plant is MIMO). When we use some of the operators defined in the Appendix, some reordering is possible (this transformation is most easily visualized by thinking of each term in transfer function matrix form)

$$\mathbf{col}(\phi_2) + \mathcal{D}(I_p, P_0)\mathbf{col}(u_2) = \mathcal{D}(I_p, P_0)\mathbf{col}(-W_2) \tag{30}$$

The term $\mathcal{D}(I_p, P_0)\mathbf{col}(u_2)$ can actually be interpreted as a plant defined by $\mathcal{D}(I_p, P_0)$ and receiving the input $\mathbf{col}(u_2)$. Furthermore, because $\mathcal{D}(I_p, P_0)$ is just p sets of uncoupled P_0 plants, its state-space matrices are easily found

$$A = \mathcal{D}(I_p, A_P), \qquad B = \mathcal{D}(I_p, B_P)$$

$$C = \mathcal{D}(I_p, C_P), \qquad D = \mathcal{D}(I_p, D_P) \tag{31}$$

A similar thing can be done with the right-hand side of Eq. (30), allowing Eq. (30) to be rewritten in the following state-space form:

$$[\mathbf{col}(\phi_2)]_i + C\boldsymbol{x}_i + D[\mathbf{col}(u_2)]_i = C\boldsymbol{x}_i^* + D[\mathbf{col}(-W_2)]_i \tag{32}$$

$$\boldsymbol{x}_{i+1} = A\boldsymbol{x}_i + B[\mathbf{col}(u_2)]_i \tag{33}$$

$$\boldsymbol{x}_{i+1}^* = A\boldsymbol{x}_i^* + B[\mathbf{col}(-W_2)]_i \tag{34}$$

where $\boldsymbol{x}$ and $\boldsymbol{x}^*$ are state variables and the i subscript refers to the coefficient of z^{-i} (or, equivalently, to the ith time step in the state-space context).

Equations (32–34) enforce the same conditions on u_2 and ϕ_2 as Eq. (30), but further simplification is possible. In Ref. 1, the single-input/single-output equivalents of u_2 and ϕ_2 were represented as impulse responses restricted to a designated number of terms by the designer, thereby ensuring that they were stable. The same can be done here with the matrix versions of u_2 and ϕ_2. If their elements are restricted to being polynomial sequences of no more than k terms per entry (where k is no less than the length of the W_2 impulse responses), then, for $i > k$, Eqs. (32–34) reduce to

$$C\boldsymbol{x}_i = C\boldsymbol{x}_i^* \tag{35}$$

$$\boldsymbol{x}_{i+1} = A\boldsymbol{x}_i \tag{36}$$

$$\boldsymbol{x}_{i+1}^* = A\boldsymbol{x}_i^* \tag{37}$$

Obviously, the only way Eq. (35) can be satisfied is if $\boldsymbol{x}_i = \boldsymbol{x}_i^*$ for $i > k$. Equations (36) and (37) mean that ensuring $\boldsymbol{x}_i = \boldsymbol{x}_i^*$ for any $i > k$ also ensures that it holds for all $i > k$. Because $\boldsymbol{x}_i^*$ has to be determined for all $i \leq k$ to generate the right-hand side of Eq. (32), $\boldsymbol{x}_{k+1}^*$ is readily obtainable and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k+1}^*$ is the easiest constraint to enforce. It is useful at this point to introduce some shorthand:

$$\tilde{\boldsymbol{u}}_2 = \begin{Bmatrix} [\mathbf{col}(u_2)]_{z^0} \\ [\mathbf{col}(u_2)]_{z^{-1}} \\ \vdots \end{Bmatrix}, \qquad \tilde{\phi}_2 = \begin{Bmatrix} [\mathbf{col}(\phi_2)]_{z^0} \\ [\mathbf{col}(\phi_2)]_{z^{-1}} \\ \vdots \end{Bmatrix} \tag{38}$$

$$\tilde{\boldsymbol{x}}_2 = \begin{Bmatrix} x_0 \\ x_1 \\ \vdots \end{Bmatrix}, \qquad \boldsymbol{F} = \begin{Bmatrix} F_0 \\ F_1 \\ \vdots \end{Bmatrix}, \qquad \boldsymbol{f} = \begin{Bmatrix} 0 \\ \vdots \\ 0 \\ x_{k+1}^* \end{Bmatrix} \tag{39}$$

$$\boldsymbol{F}_i = C\boldsymbol{x}_i^* + D[\mathbf{col}(-W_2)]_i \tag{40}$$

$$\bar{A}_p = \mathcal{Q}[\mathcal{D}(I_k, A), p] \tag{41}$$

$$\bar{B}_p = \mathcal{D}(I_k, B) \tag{42}$$

$$\bar{C}_p = \mathcal{D}(I_k, C) \tag{43}$$

$$\bar{D}_p = \mathcal{D}(I_k, D) \tag{44}$$

The constraint of Eq. (23) can then be expressed in terms of the state-space form,

$$\begin{bmatrix} I & \bar{D}_p & \bar{C}_p \\ & \bar{B}_p & \bar{A}_p \end{bmatrix} \begin{Bmatrix} \tilde{\phi}_2 \\ \tilde{u}_2 \\ \tilde{x}_2 \end{Bmatrix} = \begin{Bmatrix} F \\ f \end{Bmatrix} \tag{45}$$

with the added constraint $(\tilde{\boldsymbol{x}}_2)_{z^0} = 0$. Equations (24) and (25) can be expressed in an entirely analagous fashion. This format is able to be used directly in a set of linear program constraints.

Equations (26) and (27) are more difficult because the unknowns u_2 and u_3 are postmultiplied by terms, which makes them difficult to convert directly into linear program constraints. Consider the transpose of Eq. (26), expanded out for a plant with two inputs

$$u_4^T - Wu_2^T = 0 \tag{46}$$

$$\begin{Bmatrix} \mathrm{col}_1\left(u_4^T\right) \\ \mathrm{col}_2\left(u_4^T\right) \end{Bmatrix} - \begin{bmatrix} W & \\ & W \end{bmatrix} \begin{Bmatrix} \mathrm{col}_1\left(u_2^T\right) \\ \mathrm{col}_2\left(u_2^T\right) \end{Bmatrix} = 0 \tag{47}$$

When we treat W as a plant with state-space matrices A_w to D_w, this can be expressed in an alternative form

$$\begin{Bmatrix} \mathrm{col}_1\left(u_4^T\right) \\ \mathrm{col}_2\left(u_4^T\right) \end{Bmatrix}_i - \begin{bmatrix} C_w & \\ & C_w \end{bmatrix} \begin{Bmatrix} x_a \\ x_b \end{Bmatrix}_i - \begin{bmatrix} D_w & \\ & D_w \end{bmatrix} \begin{Bmatrix} \mathrm{col}_1\left(u_2^T\right) \\ \mathrm{col}_2\left(u_2^T\right) \end{Bmatrix}_i = 0 \tag{48}$$

$$\begin{bmatrix} A_w & \\ & A_w \end{bmatrix} \begin{Bmatrix} x_a \\ x_b \end{Bmatrix}_i + \begin{bmatrix} B_w & \\ & B_w \end{bmatrix} \begin{Bmatrix} \mathrm{col}_1\left(u_2^T\right) \\ \mathrm{col}_2\left(u_2^T\right) \end{Bmatrix}_i = \begin{Bmatrix} x_a \\ x_b \end{Bmatrix}_{i+1} \tag{49}$$

It would be more useful if u_2 and u_4 were expressed in terms of untransposed quantities, which can be achieved with a little reordering,

$$\begin{Bmatrix} u_{4_{11}} \\ u_{4_{21}} \\ u_{4_{12}} \\ u_{4_{22}} \end{Bmatrix}_i - \begin{bmatrix} \mathrm{row}_1(C_w) & \\ & \mathrm{row}_1(C_w) \\ \mathrm{row}_2(C_w) & \\ & \mathrm{row}_2(C_w) \end{bmatrix} \begin{Bmatrix} x_a \\ x_b \end{Bmatrix}_i - \begin{bmatrix} D_{w_{11}} & & D_{w_{12}} & \\ & D_{w_{11}} & & D_{w_{12}} \\ D_{w_{21}} & & D_{w_{22}} & \\ & D_{w_{21}} & & D_{w_{22}} \end{bmatrix} \begin{Bmatrix} u_{2_{11}} \\ u_{2_{21}} \\ u_{2_{12}} \\ u_{2_{22}} \end{Bmatrix}_i = 0 \tag{50}$$

$$\begin{bmatrix} A_w & \\ & A_w \end{bmatrix} \begin{Bmatrix} x_a \\ x_b \end{Bmatrix}_i + \begin{bmatrix} \mathrm{col}_1(B_w) & & \mathrm{col}_2(B_w) & \\ & \mathrm{col}_1(B_w) & & \mathrm{col}_2(B_w) \end{bmatrix} \begin{Bmatrix} u_{2_{11}} \\ u_{2_{21}} \\ u_{2_{12}} \\ u_{2_{22}} \end{Bmatrix}_i = \begin{Bmatrix} x_a \\ x_b \end{Bmatrix}_{i+1} \tag{51}$$

This can be extended to a plant with p inputs and expressed in operator notation as

$$\mathbf{col}(u_4)_i - \mathcal{E}_{\mathrm{row}}(C_w, p)(x)_i - \mathcal{E}(D_w, p)\mathbf{col}(u_2)_i = 0 \tag{52}$$

$$\mathcal{D}(I_p, A_w)(x)_i - \mathcal{E}_{\mathrm{col}}(B_w, p)\mathbf{col}(u_2)_i = \mathbf{col}(u_2)_{i+1} \tag{53}$$

The working from here is similar to the earlier derivations, with only some small differences in the shorthand notation,

$$\tilde{\boldsymbol{x}}_w = \begin{Bmatrix} x_a \\ x_b \\ \vdots \end{Bmatrix} \tag{54}$$

$$\bar{A}_p^* = \mathcal{D}(I_p, A_w) \tag{55}$$

$$\bar{A}_p = \mathcal{Q}\left[\mathcal{D}\left(I_k, A_w^*\right), p\right] \tag{56}$$

$$\bar{B}_p = \mathcal{D}[I_k, \mathcal{E}_{\mathrm{col}}(B_w, p)] \tag{57}$$

$$\bar{C}_p = \mathcal{D}[I_k, \mathcal{E}_{\mathrm{row}}(C_w, p)] \tag{58}$$

$$\bar{D}_p = \mathcal{D}[I_k, \mathcal{E}(D_w, p)] \tag{59}$$

The constraint of Eq. (26) can then be expressed in terms of the state-space form

$$\begin{bmatrix} -I & \bar{D}_w & \bar{C}_w \\ & \bar{B}_w & \bar{A}_w \end{bmatrix} \begin{Bmatrix} \tilde{u}_4 \\ \tilde{u}_2 \\ \tilde{x}_w \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{60}$$

with the added constraint $(\tilde{\boldsymbol{x}}_w)_{z^0} = 0$. Equation (27) is similar, except that the transpose of the plant is needed to find the A_v–D_v matrices (the equivalent of the A_w–D_w state-space matrices in the earlier working). This is a straightforward task using the well-known equation for finding the transfer function description of a plant from the state-space matrices

$$P = C_P(sI - A_P)^{-1}B_P + D_P \tag{61}$$

$$P^T = B_P^T\left[(sI - A_P)^{-1}\right]^T C_P^T + D_P^T \tag{62}$$

$$= B_P^T\left(sI - A_P^T\right)^{-1}C_P^T + D_P^T \tag{63}$$

By direct analogy, the transposed plant's state-space matrices (denoted by a t subscript) can be written

$$A_t = A_P^T, \qquad B_t = C_P^T$$

$$C_t = B_P^T, \qquad D_t = D_P^T \tag{64}$$

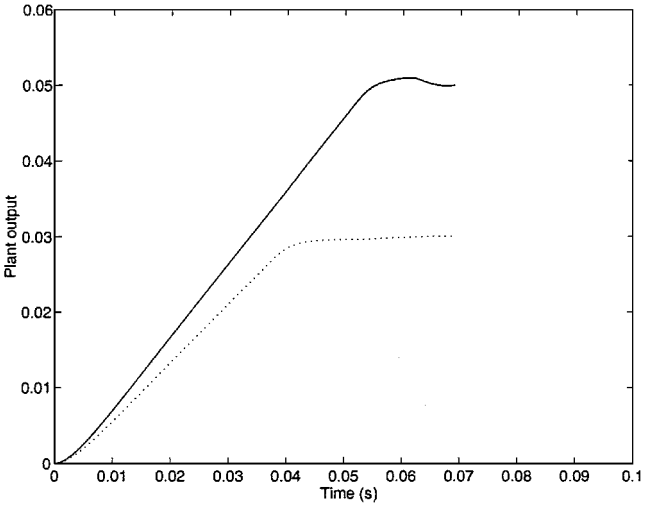
V. Comparison with Transfer Function Approach

The preceding state-space formulations were applied to the same plant as the example in Ref. 2. For the tracking optimization, the same u and ϕ sequence lengths were specified ($k = 100$) with the same set of performance constraints. The state-space results showed little difference to those for the transfer function formulation. The two linear programs required the same number of variables, but the state-space approach required about 40% more constraint equations. Despite this, the number of nonzeros for the transfer function approach was almost double that needed for the state-space version. This highlights the ability of the state-space tracking optimization to generate very sparse problems, making them well suited to sparse linear program solvers.

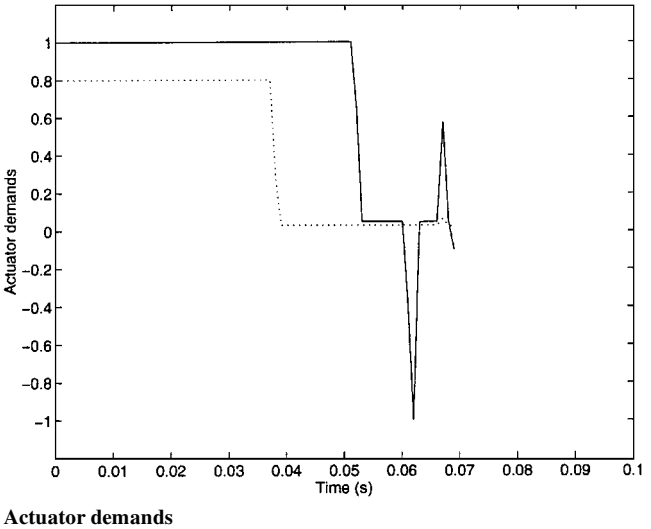
A second tracking optimization was performed for shorter u and ϕ sequence lengths. With $k = 70$ the state-space solution changed very little, but the transfer function formulation showed significant change at the end of the maneuver (see Figs. 4 and 5). This was a direct result of the state-space formulation excluding N_{C_1} from the optimization, whereas it appeared explicitly in the transfer function linear program.

The regulation optimization was also formed with an equivalent size ($k = 156$) to the transfer function formulation. It was observed that the state-space linear program required about 55% more variables and just over double the number of constraints. As was observed for the tracking optimization, however, the number of nonzeros was significantly less (a 40% reduction).

Unfortunately, just as the transfer function linear program could not be solved in its original form, the state-space version was also unable to be solved. This was handled in the transfer function case by converting the linear program to its dual. Doing this with the state-space formulation offers little advantage over the transfer function approach due to the much larger problem dimensions, even with fewer nonzeros. Another reason for seeking an alternative formulation is to find a simpler approach, but the need to convert the linear program to the dual does not really satisfy this goal.

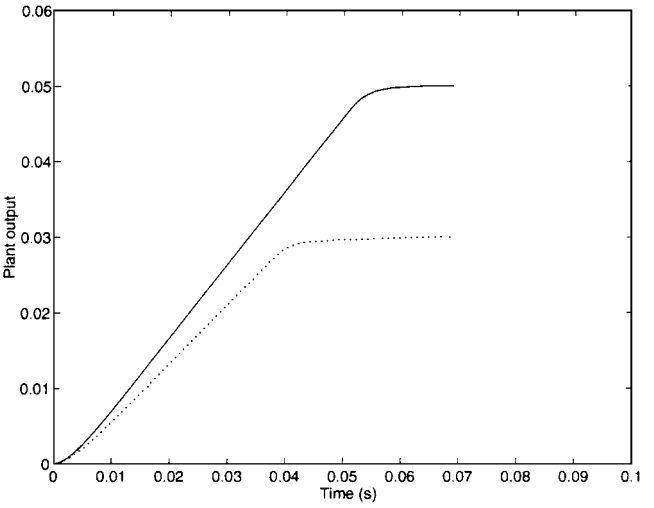


Plant outputs

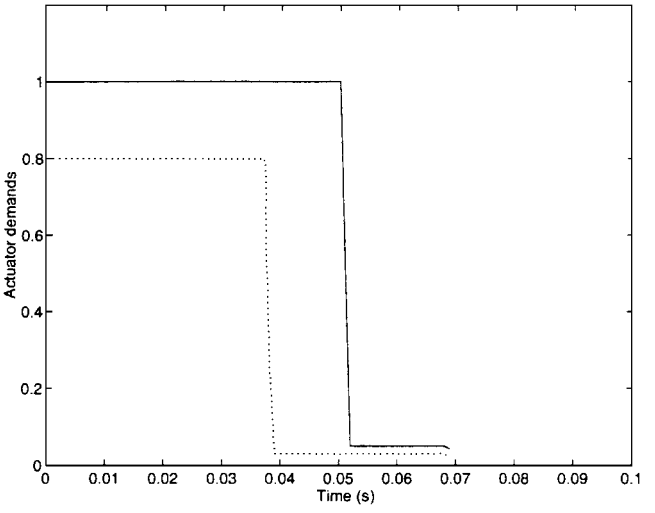


Actuator demands

Fig. 4 Transfer function tracking results with $k = 70$.



Plant outputs



Actuator demands

Fig. 5 State-space tracking results with $k = 70$.

VI. Conclusions

State-space constraint sets have been derived for both the tracking and regulation optimizations of Ref. 2. The tracking optimization was shown to be very successful and much simpler to form than the transfer function formulation. It was also shown for the example plant that the state-space approach returned a better solution when the maneuver was restricted to be completed in a shorter period of time (corresponding to a lower-order controller in the transfer function case).

The regulation optimization used simpler mathematical concepts, but ultimately offered few advantages over the transfer function formulation. No improvement was observed in numerical reliability and the problem dimensions increased, but far fewer nonzeros were required in the constraint set. The formulation did, however, show how the constraints can be manipulated and recast in a different form, highlighting some of the points that must be considered when doing so.

Recasting both optimizations relied on the alternate block diagram of Fig. 3. This format has great potential for other formulations as well, removing limitations on unstable pole-zero cancellation

between N_{C_1} and w_1. It was also shown in the example that the elimination of N_{C_1} permitted better results for lower-order tracking optimizations.

Appendix: Notation

For $\mathrm{col}_i(a)$, extract the ith column of a

$$\mathbf{col}(a) = \begin{bmatrix} \mathrm{col}_1(a) \\ \mathrm{col}_2(a) \\ \vdots \end{bmatrix}$$

For $\mathcal{D}(a, b)$, a is a diagonal matrix and b can be any matrix. The operator multiplies the entire b matrix by each diagonal element of a in turn, storing the results diagonally. The definition is best shown by example:

$$\begin{bmatrix} a_{11}b & & \\ & a_{22}b & \\ & & \ddots \end{bmatrix}$$

For $\mathcal{E}(a, i)$, expand each element of the matrix a into an $(i \times i)$ diagonal submatrix. The result will have i times as many rows and columns as the original a matrix. For example,

$$\mathcal{E}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, 2\right) = \begin{bmatrix} 1 & & 2 & \\ & 1 & & 2 \\ 3 & & 4 & \\ & 3 & & 4 \end{bmatrix}$$

For $\mathcal{Q}(a, i)$, place blocks of $-I$ above the main diagonal of $\mathcal{D}(I_i, a)$. For example,

$$\mathcal{Q}(a, 4) = \begin{bmatrix} a & -I & & \\ & a & -I & \\ & & a & -I \\ & & & a \end{bmatrix}$$

For $\xi(p, k)$, assemble p blocks (side-by-side) of p rows consisting of k 1's. For example,

$$\xi(2, 3) = \begin{bmatrix} 1 & 1 & 1 & & & & 1 & 1 & 1 & & & \\ & & & 1 & 1 & 1 & & & & 1 & 1 & 1 \end{bmatrix}$$

Acknowledgment

This work has been associated with a research grant from the Australian Research Council.

References

[1]Scott, C. N., and Wood, L. A., "Optimal Robust Tracking Subject to Disturbances, Noise, Plant Uncertainty, and Performance Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 5, 1998, pp. 774–779.

[2]Scott, C. N., and Wood, L. A., "Extension of l_1-Optimal Robust Tracking to the Multivariable Case," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, 2000, pp. 850–856.

[3]Dahleh, M. A., and Pearson, J. B., Jr., "l^1-Optimal Feedback Controllers for MIMO Discrete-Time Systems," *IEEE Transactions on Automatic Control*, Vol. AC-32, No. 4, 1987, pp. 314–322.

[4]Dahleh, M. A., and Pearson, J. B., Jr., "Optimal Rejection of Persistent Disturbances, Robust Stability, and Mixed Sensitivity Minimization," *IEEE Transactions on Automatic Control*, Vol. 33, No. 8, 1988, pp. 722–731.

[5]Diaz-Bobillo, I. J., and Dahleh, M. A., "Minimization of the Maximum Peak-to-Peak Gain: The General Multiblock Problem," *IEEE Transactions on Automatic Control*, Vol. 38, No. 10, 1993, pp. 1459–1482.

[6]Dahleh, M. A., and Diaz-Bobillo, I. J., *Control of Uncertain Systems, A Linear Programming Approach*, Prentice–Hall, Englewood Cliffs, NJ, 1995, Chaps. 10–12.

[7]Vethecan, J., "Design of Compensators for LTI Feedback Systems by Convex Optimization," Ph.D. Thesis, Dept. of Aerospace Engineering, Royal Melbourne Inst. of Technology, Melbourne, VIC, Australia, March 1996, Chaps. 3–6.

[8]Deodhare, G., Halpern, M., Hill, R., and Vethecan, J., "Simultaneous Minimization of Tracking Error and Actuator Activity in an l_1 Framework," Research Rept. 15, Royal Melbourne Inst. of Technology, Dept. of Mathematics, Melbourne, VIC, Australia, Oct. 1994.

[9]Riseborough, P., Hill, R., Vethecan, J., and Wood, L., "l_1 Norm Based Optimal Tracking and Vibration Control," *First Australasian Congress on Applied Mechanics*, Vol. 2, Inst. of Engineers, Melbourne, VIC, Australia, 1993, pp. 819–824.

[10]Vidyasagar, M., "Control System Synthesis: A Factorization Approach," MIT Press, Cambridge, MA, 1985, Chaps. 4 and 5.