A New CAD Method and Associated Architectures for Linear Controllers

STEPHEN P. BOYD, MEMBER, IEEE, VENKATARAMANAN BALAKRISHNAN, CRAIG H. BARRATT, NASSER M. KHRAISHI, STUDENT MEMBER, IEEE, XIAOMING LI, STUDENT MEMBER, IEEE, DAVID G. MEYER, AND STEPHEN A. NORMAN

***Abstract*—A new CAD method and associated architectures are proposed for linear controllers. The design method and architecture are based on recent results which parameterize all controllers which stabilize a given plant. With this architecture, the design of controllers is a *convex programming problem* which can be solved numerically.**

Constraints on the closed-loop system such as asymptotic tracking, decoupling, limits on peak excursions of variables, step response, settling time, and overshoot, as well as frequency domain inequalities are readily incorporated in the design. The minimization objective is quite general, with LQG, H_∞, and new l_1 types as special cases.

The constraints and objective are specified in a *control specification language* which is natural for the control engineer, referring directly to step responses, noise powers, transfer functions, and so on. This control specification language will be the input to a *compiler* which will translate the specifications into a standard convex program in R^L, which is then solved by some numerical convex program solver. A small but powerful subset of the language has been specified and its associated compiler implemented.

The architecture proposed simplifies not only *design* of the controller but also its *implementation*. These controllers can be built right now from off the shelf components or integrated using standard VLSI cells.

I. Introduction

EVEN with great advances in control theory and enormous advances in available computing power, the design of a linear time-invariant (LTI) controller which stabilizes a given LTI plant and meets some given engineering specifications can still be quite a challenge. There are two basic schemes for LTI controller design—one is based on parameter optimization, usually by numerical methods, and the other is based on analytical methods for determining a controller which is optimal in some well-defined sense, say in terms of a quadratic cost function.

With only a few parameters and not very demanding specifications, a controller can often be designed by hand, using techniques like root locus, loop shaping, and tuning (experienced guessing) based on tuning rules or reasoning from Bode or Nyquist plots. This is the case with many industrial single-loop PID controllers. Because of the simplicity of these design problems, it is possible to directly consider the specifications of interest, say, overshoot, settling time, and actuator authority.

For more complicated multiinput multioutput (MIMO) systems, tuning by hand is not feasible, and it is natural to use a computer to do the tuning or parameter optimization (see, e.g., [25], [31], [32], [27]). Despite the vast computing power now available, parameter optimization techniques of controller design have had certain basic problems. Given a plant and a controller which depends on a (vector) parameter, just finding a parameter value which results in a closed-loop stable system (or determining that none exists) is in fact a very difficult problem about which very little is known, and no practical algorithm exists which can do this. There is no practical algorithm which will determine whether *no* parameter value exists which results in a controller which meets the given specifications.

A simplified, informal explanation of this is as follows. Suppose the controller itself is affine in the parameters (this is often the case). Then any closed-loop quantity, say, a transfer function evaluated at a certain frequency, varies in a *linear fractional* way with the controller parameter. This linear fractional dependence, although simply described analytically, is quite complicated. Nice convex constraints on closed-loop quantities do not result in convex constraints in parameter space. Thus, the set of parameter values which satisfy the specifications is generally not convex, and in fact, it is often disconnected.

The second scheme is *optimal controller design*. The most obvious example is linear quadratic Gaussian (LQG) based controller design; more recently H_∞-optimal controller design has been developed. These schemes are based on an analytical method for finding a controller which minimizes a simple objective functional of the closed-loop system (we usually specify parameters, e.g., weight matrices, in the objective criterion). The advantage of such design schemes is that they always find stabilizing controllers.

A great disadvantage is that the actual engineering specifications must be translated into a choice of the weight matrices expected by the LQG design process. Various rules of thumb are available (e.g., Bryson's rule [5, p. 149]), and experienced LQG designers have a good feel for the relation of the weight matrices to control authority, peaking, and other properties of the closed-loop system, not unlike experienced designers of PID controllers.

This gap between the actual engineering specifications and the specification of the LQG weight matrices is large, and in practice growing. While the objective function for an LQG design can be given a physical interpretation in terms of rms response to disturbances having a specific spectral density, it is rarely used in this direct a way in practice. Instead, we find LQG used more as a *box-with-crank* method[1] for finding a stabilizing controller which sometimes has good properties: weight matrices are fed in, and a controller comes out. Thus, with recent techniques like loop transfer recovery (LTR), the actual physical interpretation of the objective function has all but disappeared. In a sense, a technique like LTR can be considered a sophisticated variant of a PID tuning rule. The result is that the task of translating lower level engineering specifications into LQG weight matrices is more an art than a science.

Similar comments could be made about other optimal controller

Manuscript received January 12, 1987; revised August 24, 1987. Paper recommended by Past Associate Editor, D. P. Looze. This work was supported by the Office of Naval Research under N00014-86-K-0112 and by the National Science Foundation under Grant ECS-85-52465.

S. P. Boyd, V. Balakrishnan, C. H. Barratt, N. M. Khraishi, X. Li, and S. A. Norman are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-4055.

D. G. Meyer is with the Robotics and Control Laboratory, University of Virginia, Charlottesville, VA 22903.

IEEE Log Number 8718716.

[1] We quote G. Stein.

design methods: these methods really only push the design problem back one stage, from direct design of a controller to design of weight matrices, or weighting transfer matrices in H_∞ controller design.

In this paper we describe a controller design method which has some of the good properties of each of these two basic design schemes. Like a very simple tuning method, our method will work directly with the given engineering specifications with no need to mold them into a single simple objective. Like the optimal controller design methods, our method is *effective,* that is, always finds a solution if one exists. Indeed, it answers the question of whether there is a stabilizing controller satisfying the given constraints.

The theoretical basis for the method we describe is the parameterization of stabilizing controllers, or more importantly the so-called Q-parameterization of the closed-loop input/output (I/O) maps achievable with controllers which stabilize the system. The usefulness of the Q-parameterization for controller design has been noted by several authors, e.g., Desoer and Gustafson [9], [10].

For single-input single-output (SISO) systems, a technique which is equivalent to the Q-parameterization has been known and used in controller design since 1958 [33, ch. 7]. The technique is very simple—the *closed-loop* transfer function $H = 1/(1 + PC)$ is designed, subject to the requirement that it must vanish at every unstable pole of the plant and equal one at every unstable zero of the plant. These conditions guarantee that the controller $C = (1 - H)/PH$ yields a closed-loop stable system.[2] This technique was used in a very interesting series of papers written by Fegley and his students starting in 1964 [16], [29], [17], [30], [8], and [4], inspired by a paper by Zadeh and Whalen [39]. Some of these papers, for example [29], propose a controller design method which is not far from the method we describe here.

One aspect of our proposed design method which is entirely new, as far as we know, is our proposal to use a *compiler* to translate the actual engineering constraints into constraints on Q. Of course, this entails the design of a *formal language* for specifying control system constraints and objectives.

Another new aspect is the observation that controller model reduction may not be necessary, if the controller is implemented with a special architecture—feedback around a MIMO finite impulse response (FIR) filter.

The structure of this paper is as follows: in Section II we describe the basic setup we consider; in Section III we discuss the Q-parameterization and its interpretation in terms of observer-based controllers. In Section IV we take a close look at many typical specifications on closed-loop quantities and point out that many result in convex constraints on the free (design) parameter Q. In Section V we consider the design of a control specification language to express the actual engineering constraints, and the design of a compiler to translate those constraints into constraints on the design parameters. In Section VI we briefly discuss special purpose architectures for directly implementing the controllers designed by our method. In Section VII we describe the design system we have already implemented, and briefly explain how we implemented it. A future paper will cover the implementation in detail.

A simple example, introduced in the next section, is used throughout the paper. In Section VIII we show how our method might be used to design a controller for this example.

II. Basic Setup

The basic plant we consider is shown in Fig. 1. We decompose its input into two signal vectors, w, the *exogenous inputs,* and u, the *control* or *actuator inputs.* Its outputs we decompose into y, the *measured* or *sensor outputs,* and z, the *regulated variables.* Our job is to design a controller K, with input y *only,* and output u: $u = -Ky$. (The negative sign reflects the tradition that feedback should be negative; more importantly, it is the standard use in the Q-parameterization formulas, e.g., in [36].) Similar decompositions of the inputs acting on a plant and the outputs coming from a plant can be found in the work of many authors, for example, Nett [28].

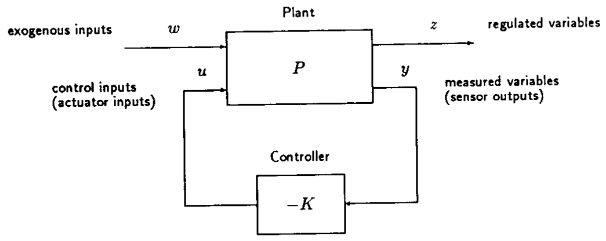


Fig. 1. Basic plant and controller.

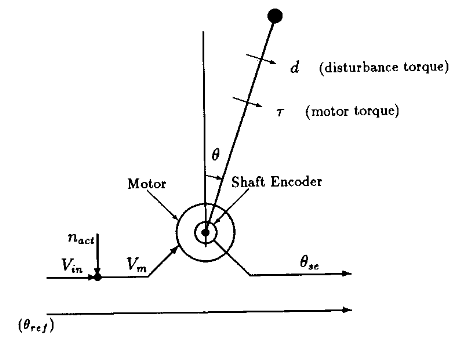


Fig. 2. Example system.

The signal y represents the signal actually accessible to the controller K, including any command inputs, which may be considered exogenous inputs (i.e., components of w) passed directly to some of components of y.[3] Similarly, the signal u represents those inputs to the plant which our controller may vary, that is, those inputs to the plant manipulable by the controller. Thus, it is by *definition* that the controller has input y only and output u only.

The exogenous input w will include real physical disturbances or noises (torques, forces) acting on the plant, any actuator or sensor noise, and, as mentioned above, any command inputs. w may also contain fictitious inputs injected anywhere in the plant.

The signal z represents any system signal about which we will express a specification, regardless of its accessibility to the controller. Obvious examples are the actual positions, forces, temperatures, etc., we wish to regulate or control. z may include internal states or variables we wish to limit. Some components can be fictitious, e.g., linear combinations of state variables or even filtered versions of signals. z may also include components of u or y.

Thus, H_{zw}, the multiinput multioutput closed-loop map from the exogenous inputs w to the regulated variables z, contains, by definition, every *closed-loop* map of interest.

To illustrate this decomposition of inputs and outputs, we present a simple example. A motor with shaft encoder is used to regulate the angle θ of a pointer to some (small) commanded angle, θ_{ref}, while a disturbance torque, d, acts on the pointer; see Figs. 2 and 3.

Let θ_{se} denote the shaft encoder output, and let $n_{\text{sens}} = \theta - \theta_{\text{se}}$,

[2] It is not clear whether the users of this method understood that these conditions are not only sufficient, but necessary as well. The method yields *all* controllers which stabilize P.

[3] We may think of the exogenous inputs which represent commands as the actual physical command inputs, e.g., angle of a potentiometer. The corresponding components of y are thought of as the sensed command, e.g., voltage output from a potentiometer.

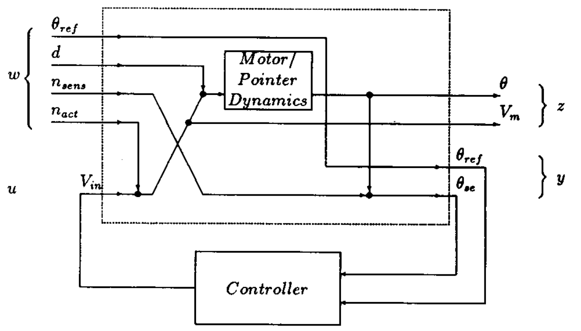


Fig. 3. Block diagram of example system.

the difference between the actual angle and the sensed angle, that is, sensor quantization noise. Let n_{act} denote a fictitious actuator noise signal; its use will become clear in the sequel. Exogenous inputs are

$$w=\begin{bmatrix}\theta_{\text{ref}}\\ d\\ n_{\text{sens}}\\ n_{\text{act}}\end{bmatrix}. \tag{1}$$

There are two sensed outputs

$$y=\begin{bmatrix}\theta_{\text{ref}}\\ \theta_{\text{se}}\end{bmatrix}. \tag{2}$$

Note that the command θ_{ref} is simply passed through P; thus the block diagram of Fig. 3 could be drawn in a more conventional way as a two degree of freedom SISO controller.

There is only one actuator input, the motor input voltage V_{in}; that is, $u = V_{\text{in}}$. A possible choice of regulated variables is

$$z=\begin{bmatrix}\theta\\ V_m\end{bmatrix}. \tag{3}$$

We will refer to this example throughout the sequel.

Returning to our general setup, we assume only that the I/O map of the plant P is linear and time-invariant. For the moment, we do not say whether the system is continuous or discrete time, since much of the ensuing discussion is independent of this. In any case, a real system is almost certainly hybrid with a continuous-time physical plant and a discrete, possible multirate, controller.

We partition the map P as

$$P=\begin{bmatrix}P_{zw} & P_{zu}\\ P_{yw} & P_{yu}\end{bmatrix}$$

so that

$$\begin{bmatrix}z\\ y\end{bmatrix}=P\begin{bmatrix}w\\ u\end{bmatrix}.$$

Of course, we have

$$H_{zw}=P_{zw}-P_{zu}K(I+P_{yu}K)^{-1}P_{yw} \tag{4}$$

where H_{zw} is the closed-loop map from the exogenous inputs w to the regulated variables z.

Note that H_{zw} depends on the controller K in a *linear fractional* fashion. A relatively simple constraint on H_{zw}, e.g., that a certain entry be zero, corresponds via (4) to a much more complicated constraint on the controller K. We remark here that if $P_{yu} = 0$, then (4) simplifies considerably to

$$H_{zw}=P_{zw}-P_{zu}KP_{yw}.$$

Here H_{zw} is *affine* in K. Note that this corresponds to the case where there is essentially *no feedback* (through K) in the system.

III. Parameterization of Stabilizing Controllers

In Section III-A we discuss the recent parameterization of all stabilizing controllers; in Section III-B we apply this to our pointer example system. Precise definitions and all details can be found in Vidyasagar's book [36].

A. Q-Parameterization: Theory

A basic requirement is that the controller stabilize the plant. A recent advance in control theory is a fairly explicit description of the set of H_{zw}'s achievable with controllers which stabilize the system [38], [11], [36].

This set is a *linear variety,* that is, a translation of a linear subspace. A linear variety $\mathcal{L}$ in a vector space $\mathcal{V}$ is often described as the nullspace or range of an affine map from or to $\mathcal{V}$. Thus, we might have

$$\mathcal{L}=\{v\in\mathcal{V}\mid Av=b\}$$

where A is a linear map from $\mathcal{V}$ to $\mathcal{W}$, $b\in\mathcal{W}$, and $\mathcal{W}$ is some vector space. This is a description of $\mathcal{L}$ in terms of a *linear equality constraint*; if $\mathcal{W}$ is $\boldsymbol{R}^k$, then we can interpret each component of $Av = b$ as a single linear functional equality constraint in $\mathcal{V}$. Let $\mathcal{H} = \{H_{zw}\mid$ system is closed-loop stable$\}$, the set of achievable H_{zw}'s. $\mathcal{H}$ may be described via linear equality constraints using the *interpolation conditions* (see, e.g., [18], [3]; early versions can be found in the references cited in Section I).

Alternatively, we may describe a linear variety, $\mathcal{L}$, as the range of an affine map

$$\mathcal{L}=\{Cu+d\mid u\in\mathcal{U}\}$$

where C is a linear map from some vector space $\mathcal{U}$ to $\mathcal{V}$ and $d\in\mathcal{V}$. Such a representation can be regarded as a *free parametric representation* of $\mathcal{L}$; u is a free parameter.

A free parametric representation of $\mathcal{H}$ can be derived using stable coprime fractional theory [36]. One standard form, used by Doyle and colleagues at Cal Tech and Honeywell SRC, is the Q-parameterization

$$\mathcal{H}=\{T_1+T_2QT_3\mid Q \text{ stable}\}$$

where T_1, T_2, and T_3 are some stable maps.

Note that the Q-parameterization

$$H_{zw}=T_1+T_2QT_3 \tag{5}$$

has exactly the same form as the no-feedback formula mentioned in Section II. We will soon see a block diagram interpretation of the Q-parameterization in which the parameter Q sees no feedback. In particular, H_{zw} is affine in Q. We remark that Q is called a *parameter* in the sense that (5) is a *parameterization* of all closed-loop maps, and not in the sense of being an unspecified real number such as the integrator gain in a PID controller.

The derivation of the Q-parameterization starts with any controller, K_{nom}, which stabilizes the plant and which we will call the *nominal controller.* Let $K_{\text{nom}} = Y^{-1}X$ be a left stable coprime factorization [36] of the nominal controller and $P_{yu} = \tilde{D}^{-1}\tilde{N}$ be a left stable coprime factorization of P_{yu}. Then every controller of the form

$$K=(Y-Q\tilde{N})^{-1}(X+Q\tilde{D}),\ Q \text{ stable}$$

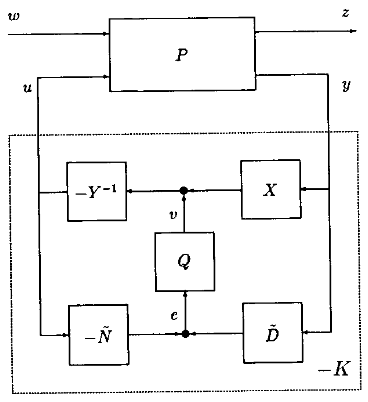


Fig. 4. Block diagram of Q-parameterization.

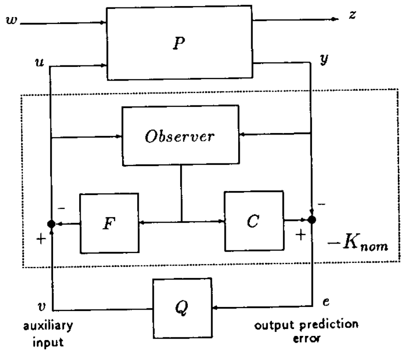


Fig. 6. Doyle's interpretation of Q-parameterization for estimated state feedback nominal controller. Here F is a stabilizing state feedback gain and $y = Cx$ where x is the plant state.

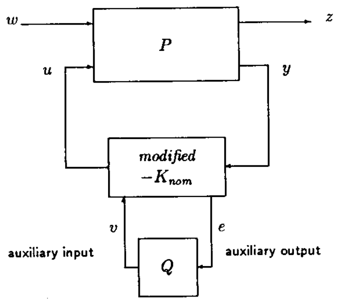


Fig. 5. Q-parameterization as modification to nominal controller.

stabilizes P and conversely every controller which stabilizes P has this form for some stable Q. There is a similar characterization of the stabilizing controllers in terms of right coprime factorizations.

Since K_{nom} stabilizes P_{yu}, there is a right coprime factorization $P_{yu} = ND^{-1}$ with $XN + YD = I$, the identity map. A little calculation yields the Q-parameterization formula, $H_{zw} = T_1 + T_2QT_3$, with

$$T_1 = P_{zw} - P_{zu}DXP_{yw},$$

$$T_2 = -P_{zu}D,$$

$$T_3 = \tilde{D}P_{yw}. \qquad (6)$$

A simple block diagram of the Q-parameterization is shown in Figs. 4 and 5.

T_1 is simply the H_{zw} achieved with the nominal controller K_{nom}; T_2 is the map from v to z, T_3 is the map from w to e (see Fig. 5). The key to the parameterization is that the closed-loop map from v to e is *zero*, so that Q sees no feedback.

Doyle [13] has given a very nice interpretation of the Q-parameterization when the nominal controller is an estimated state feedback. In this case e is simply the output prediction error $\hat{y} - y$ of the observer, and v is just added before the observer tap to the output of the nominal controller as shown in Fig. 6. The controller shown in Fig. 6 is sometimes called an *observer-based controller* (OBC); the point is that *every* controller which stabilizes P can be realized as an observer-based controller.

Doyle's interpretation can also be given to the general Q-parameterization. In this case the 'observer' is reconstructing a pseudostate associated with the plant factorization, and the auxiliary output e can be thought of as the pseudostate reconstruction error; similarly the auxiliary input v can be thought of as simply added to the plant input, but before the 'observer' tap. Thus, the Q-parameterization can be thought of as a *generalized OBC*.

B. Q-Parameterization: Example

To demonstrate the results of this section, we will now find a Q-parameterization for the pointer system of Section II. We first describe its dynamics.

We assume that there is a point mass m at the end of the pointer, connected to the motor shaft by a stiff, light rod of length l. The shaft angle θ is measured such that θ is zero when the pointer is standing straight up. The total torque on the shaft has three terms—the motor torque τ, the disturbance torque d, and the gravitational torque $mgl \sin \theta$. We neglect friction. The (nonlinear) equation of motion of the pointer is

$$ml^2\ddot{\theta} = \tau + d + mgl \sin \theta.$$

Since θ will be small, we will use instead the linearized dynamics near the equilibrium point $\theta = \dot{\theta} = 0$

$$ml^2\ddot{\theta} = \tau + d + mgl\theta$$

which gives a transfer function from τ to θ (or d to θ) of $(ml^2s^2 - mgl)^{-1}$. To be concrete we set $m = 2.67$ kg, $l = 0.613$ m, and $g = 9.8$ m/s^2, so the transfer function above is $(s^2 - 16)^{-1}$. For simplicity, we ignore motor dynamics and assume that $\tau = V_m$, that is, we model the motor as having a constant gain of 1 $N - m/V$.

We discretize the plant at 40 Hz, assuming that the motor torque τ is held constant between samples (zero order hold). The transfer function from the sequence of values of τ at sampling instants to the sequence of values of θ at sampling instants is

$$\frac{0.0003128(z+1)}{z^2 - 2.0100z + 1} = \frac{0.0003128(z+1)}{(z-0.9048)(z-1.1052)}. \qquad (7)$$

The effect of the disturbance torque on the pointer at sampling instants will be modeled by a piecewise constant (zero order holded) disturbance sequence. We will use the same symbol d to denote this disturbance torque sequence, since we will not mention the continuous-time disturbance torque in the sequel. Thus, the transfer function from d to θ is also given by (7).

We will take the definitions of u, y, w, and z given in Section I (1)–(3). The plant is then

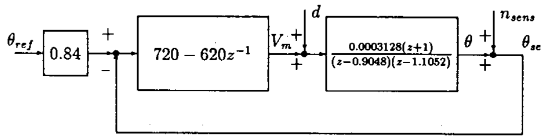


Fig. 7. Nominal controller for pointer.

$$P=\begin{array}{c}\left[\begin{array}{ccccc} 0 & \frac{0.0003128(z+1)}{z^2-2.0100z+1} & 0 & \frac{0.0003128(z+1)}{z^2-2.0100z+1} & \frac{0.0003128(z+1)}{z^2-2.0100z+1} \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{0.0003128(z+1)}{z^2-2.0100z+1} & 1 & \frac{0.0003128(z+1)}{z^2-2.0100z+1} & \frac{0.0003128(z+1)}{z^2-2.0100z+1} \end{array}\right]\begin{array}{c}(\theta)\\(V_m)\\(\theta_{\text{ref}})\\(\theta_{\text{se}})\end{array}. \\ \begin{array}{ccccc}(\theta_{\text{ref}}) & (d) & (n_{\text{sens}}) & (n_{\text{act}}) & (V_{\text{in}})\end{array}\end{array}$$

We have designed a very simple nominal controller, shown in Fig. 7, which is essentially a discretized PD controller. With this controller, the closed-loop transfer function from θ_{ref} to θ is

$$\frac{0.1892z^2+0.0263z-0.1629}{z^3-1.7848z^2+1.0313z-0.1939}$$
$$=\frac{0.1892z^2+0.0263z-0.1629}{(z-0.7908)(z-0.5394)(z-0.4545)}.$$

The 0.84 scale factor in Fig. 7 yields a D.C. gain from θ_{ref} to θ of 1, so this controller yields asymptotic tracking.

Using our notation we have nominal controller

$$K_{\text{nom}}=[-0.84(720-620z^{-1})\ \ 720-620z^{-1}].$$

Since this is stable, we use $Y=1$ and $X=K_{\text{nom}}$ as a left coprime factorization of K_{nom}.

Left and right coprime factorizations of P_{yu}

$$P_{yu}=\tilde{D}^{-1}\tilde{N}=ND^{-1}$$

with

$$XN+YD=I$$

are given by

$$P_{yu}=\begin{bmatrix}0\\ \frac{0.0003128(z+1)}{z^2-2.0100z+1}\end{bmatrix}$$
$$=\begin{bmatrix}1 & 0\\ 0 & \frac{z-1.1052}{z-0.9048}\end{bmatrix}^{-1}\begin{bmatrix}0\\ \frac{0.0003128(z+1)}{(z-0.9048)^2}\end{bmatrix}$$
$$=\begin{bmatrix}0\\ \frac{0.0003128z(z+1)}{z^3-1.7848z^2+1.0313z-0.1939}\end{bmatrix}$$
$$\cdot\left[\frac{z(z^2-2.0100z+1)}{z^3-1.7848z^2+1.0313z-0.1939}\right]^{-1}.$$

T_1, T_2, and T_3 can now be calculated from (6).

IV. Closed-Loop Constraints and Objectives

In this section and the next, we consider the problem of designing the parameter Q, given the nominal maps T_1, T_2, and T_3. We first observe that many typical requirements on the *closed-loop* performance of our system result in *convex* constraints on H_{zw}, and thus on the parameter Q. By this we mean precisely: if H_{zw} is a closed-loop response which meets our performance requirements, and $\tilde{H}_{zw}$ is any other closed-loop response which meets our requirements, then the closed-loop response $(H_{zw}+\tilde{H}_{zw})/2$ meets our requirements. Stated briefly, a specification on a control system is convex in H_{zw} if and only if *the average of two closed-loop responses meeting the specifications always meets the specifications.* Thus, one of the key results described in the previous section can be simply stated: *the requirement of closed-loop internal stability is a convex constraint on* H_{zw}.

Before proceeding, we note that requirements on the *open-loop* system generally do *not* result in convex constraints on H_{zw} of Q. One important example is the requirement that the controller K be diagonal, that is, that K be a decentralized controller. Another important example is the requirement that the controller be (open-loop) stable.

Suppose our design problem is specified as

$$\underset{H_{zw}\in\mathcal{K}\cap\mathcal{H}}{\text{minimize}}\ \Phi(H_{zw})$$

where Φ is a convex functional, $\mathcal{K}$ a convex (constraint) set, and $\mathcal{H}$ is the set of H_{zw}'s achievable with stabilizing controllers. This is equivalent to

$$\underset{Q\in\tilde{\mathcal{K}}}{\text{minimize}}\ \tilde{\Phi}(Q) \qquad (8)$$

where

$$\tilde{\mathcal{K}}=\{Q\,|\,T_1+T_2QT_3\in\mathcal{K},\ Q\text{ stable}\}$$

and

$$\tilde{\Phi}(Q)=\Phi(T_1+T_2QT_3).$$

Note that $\tilde{\Phi}$ and $\tilde{\mathcal{K}}$ are convex. Of course $\tilde{\mathcal{K}}$ may be empty; this simply means that *no stabilizing controller* can satisfy the constraint $H_{zw}\in\mathcal{K}$.

We now list some typical constraints (and objectives) on H_{zw}, some collection (sum) of which might describe the constraint set $\mathcal{K}$ (objective functional Φ). While reading through this list, the reader should keep in mind the criterion, given in the first paragraph of this section, for a constraint to be convex in H_{zw}. When an objective is described, the reader should check that the object value corresponding to the *average of two closed-loop responses* is less than or equal to the *average of the objective values* corresponding to the two closed-loop responses.

For purposes of discussion, we assume the system is discrete-time. H will denote the transfer matrix of H_{zw}, h its impulse response matrix, and $s(t)=\sum_{i=0}^{t}h(i)$ its step response matrix.

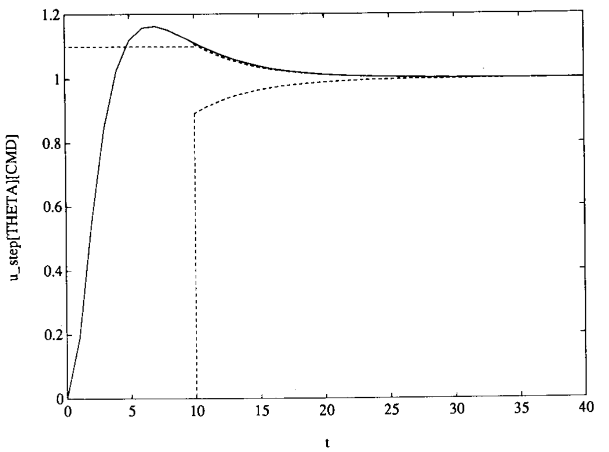


Fig. 8. Step response overshoot, undershoot, and settling time constraints. Step response with nominal controller is shown.

Of course, a convex constraint or functional of H, h, of s is a convex constraint or functional of H_{zw}.

When possible, we will refer to our pointer system of the previous sections. When referring to this system, we will use *symbolic* subscripts written in square brackets, for the reader's convenience. Thus, we will write $h[\theta][\theta_{\text{ref}}](t)$ instead of $h_{11}(t)$.

• *Asymptotic Tracking, Decoupling, and Regulation:* The step response from some command input to the regulated variable it is supposed to control must converge to one, e.g.,

$$\lim_{t\to\infty} s[\theta][\theta_{\text{ref}}](t)=1.$$

Equivalently,

$$H[\theta][\theta_{\text{ref}}](e^{j0})=1. \tag{9}$$

This constraint is a single linear functional equality constraint on H, hence, of course, a convex constraint. Alternatively, if s and $\tilde{s}$ are two step responses converging to one, then their average $(s+\tilde{s})/2$ also converges to one.

Asymptotic tracking of ramps or more complicated inputs can be handled as two or more linear functional equality constraints, e.g., $H[\theta][\theta_{\text{ref}}](e^{j0})=1$, $H'[\theta][\theta_{\text{ref}}](e^{j0})=0$.

Asymptotic regulation and *asymptotic decoupling* are similar constraints. We may require that a regulated variable asymptotically reject constant inputs appearing at certain exogenous inputs. When the exogenous input is another command input, this constraint is asymptotic decoupling; when the exogenous input is some disturbance, this constraint is asymptotic regulation. For example, to ensure that θ asymptotically rejects any constant disturbance torque, we might specify

$$H[\theta][d](e^{j0})=0. \tag{10}$$

• *Overshoot, Undershoot, and Settling-Time Limits:* We may require that some step response lie between the dashed limits shown in Fig. 8,

$$\begin{aligned} &0\le s[\theta][\theta_{\text{ref}}](t)\le 1.1 \qquad 0\le t\le 10,\\ &|s[\theta][\theta_{\text{ref}}](t)-1|\le 0.8^t \qquad t\ge 10. \end{aligned} \tag{11}$$

(The step response shown in Fig. 8 is that with the nominal controller.)

A constraint such as (11) can be expressed as a collection of linear functional inequalities on s (hence, H_{zw}): $L(t)\le s[\theta][\theta_{\text{ref}}](t)\le U(t)$ for $t=0,1,\cdots$. Thus, the set of H_{zw}'s satisfying this constraint is *convex* (it may of course be empty). This can also be seen by noting that if two step responses lie between given lower and upper limits, then so does their average.

• *Bounds on Closed-Loop Signal Peaks:* Given bounds W_j on each exogenous input, we may require that each regulated variable be bounded by some given maximum Z_i. This constraint could arise from the requirement not to saturate an actuator or sensor or exceed some internal variable force, torque, or current limit. This constraint is equivalent to

$$\sum_{j=1}^{N_{\text{exog}}} W_j \sum_{t=0}^{\infty} |h_{ij}(t)|\le Z_i \qquad \text{for } i=1,\cdots,N_{\text{reg}}$$

where N_{exog} is the number of exogenous inputs and N_{reg} the number of regulated variables.

Thus, the constraint

$$0.1\sum_{t=0}^{\infty}|h[V_m][\theta_{\text{ref}}](t)|+0.03\sum_{t=0}^{\infty}|h[V_m][d](t)| +0.001\sum_{t=0}^{\infty}|h[V_m][n_{\text{sens}}](t)|\le 50 \tag{12}$$

is necessary and sufficient to guarantee that whenever the command input θ_{ref} is bounded by 0.1, the disturbance d by 0.03, and the sensor noise n_{sens} by 0.001, the motor voltage input V_m will be bounded by 50.

Again, this is a convex constraint on h, and thus on H_{zw}.

• *Small rms Disturbance Response:* If the exogenous input w is driven by a wide-sense stationary stochastic process with some specified spectral distribution, then $Ez(t)^TG^TGz(t)$, where G is some weighting matrix, should be as small as possible.

We have

$$Ez(t)^TG^TGz(t) =\frac{1}{2\pi}\operatorname{Tr} G\left(\int_0^{2\pi} H(e^{j\Omega})S_w(e^{j\Omega})H(e^{-j\Omega})^T\,d\Omega\right)G^T$$

where S_w is the power spectral density of the stochastic process driving w. This objective is a *nonnegative quadratic functional* of H, and thus of H_{zw}, in particular it is convex; this is the form of the objective in LQG problems.[4] In words, the total noise power response of the average of two closed-loop responses is less than the average of their respective noise power responses.

For our pointer example, a possible physically motivated S_w would be

$$S_w(e^{j\Omega})=\begin{array}{c@{}c} \begin{bmatrix} 0 & 0 & 0 & 0\\ 0 & S_d(e^{j\Omega}) & 0 & 0\\ 0 & 0 & \Delta^2/12 & 0\\ 0 & 0 & 0 & 0 \end{bmatrix} & \begin{array}{l} (\theta_{\text{ref}})\\ (d)\\ (n_{\text{sens}})\\ (n_{\text{act}}) \end{array} \end{array}$$

where $S_d(e^{j\Omega})$ is the spectral density of the disturbance torque and Δ is the step size of the shaft encoder. If we are interested in the noise power in the pointer angle θ then we let $G=[1\ 0]$ and the functional above is simply

$$E\theta^2=\frac{1}{2\pi}\int_0^{2\pi}[|H[\theta][d](e^{j\Omega})|^2S_d(e^{j\Omega}) +|H[\theta][n_{\text{sens}}](e^{j\Omega})|^2\Delta^2/12]\,d\Omega \tag{13}$$

which can be interpreted as the total noise power in θ due to the disturbance and sensor noise. We remark that the second term

[4] Here *quadratic functional* includes a *linear functional* and a *constant (functional)* term.

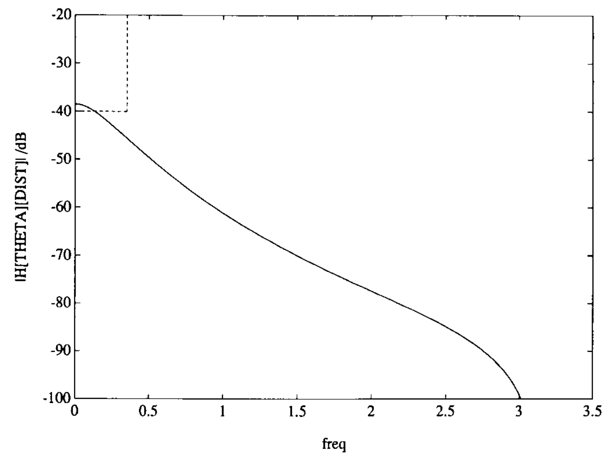


Fig. 9. Disturbance rejection bandwidth constraint (dashed line). Response shown is that of our pointer system with nominal controller.

above can be expressed

$$\frac{\Delta^2}{12}\sum_{t=0}^{\infty} h[\theta][n_{\text{sens}}](t)^2 = \frac{\Delta^2}{12}\, \|h[\theta][n_{\text{sens}}]\|_2^2.$$

It is rare for designers in practice to use an objective based solely on actual noises and disturbances present; usually fictitious noises are added to temper the design, improve robustness, and so on.

- *Bounds on Transfer Function Peak Magnitudes:* We may specify an upper bound for some entry (or block of entries) of H, e.g., $|H_{65}(e^{j\Omega})| \leq U(\Omega)$ for all Ω, where $U(\Omega)$ is some bound function. This important constraint may arise in several ways, which we discuss in the next few entries.

First, many classical specifications of bandwidth or peaking are expressed this way. Referring to our pointer system, we might specify

$$|H[\theta][d](e^{j\Omega})| \leq 0.01 \qquad \text{for } |\Omega| \leq \Omega_B \tag{14}$$

so that for frequencies below Ω_B, the disturbance torque to shaft angle transfer function is at most -40 dB. This constraint is shown in Fig. 9 with $\Omega_B = 0.35$, along with the $|H[\theta][d](e^{j\Omega})|$ achieved with the nominal controller. This is a convex constraint on H_{zw}, since if H and $\tilde{H}$ both satisfy (14), then so does $(H + \tilde{H})/2$.

- *H_∞ (Minimax Transfer Function) Objective:* The modern version of the previous classical constraint is the topic of H_∞ control theory. If we require a small rms response of some regulated variable, as above, but know only a bound on the total power in the process driving the exogenous input w, and not its specific spectral distribution, then we are led to a transfer function peak magnitude (singular value) constraint

$$\|H\|_\infty = \sup_\Omega \sigma_{\max}(H(e^{j\Omega})) \leq M \tag{15}$$

where M is the ratio of maximum allowable rms response (z) to maximum rms level of disturbance (w).

- *Classical Single-Loop Gain/Phase Margin (M-Circle) Constraints:* Suppose we break a single loop in our control system, as shown in Fig. 10(a) and (b), and require that the Nyquist plot of the transfer function from point A to point B maintain a distance at least M (the M-circle radius) from the point $+1$. Usually the loop is broken at a summing junction where a sign change occurs. This produces the usual requirement that the Nyquist plot maintain a distance at least M from the point -1, as shown in Fig. 11. This specification concerns *open-loop* maps,

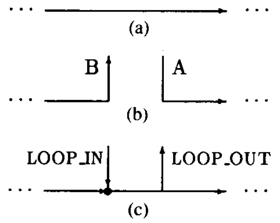


Fig. 10. (a) Some signal path in control system. (b) Signal path broken; margin of loop gain from A to B is to be constrained. (c) Fictitious input and output are added to original system to allow convex constraint on margin of loop gain from A to B.

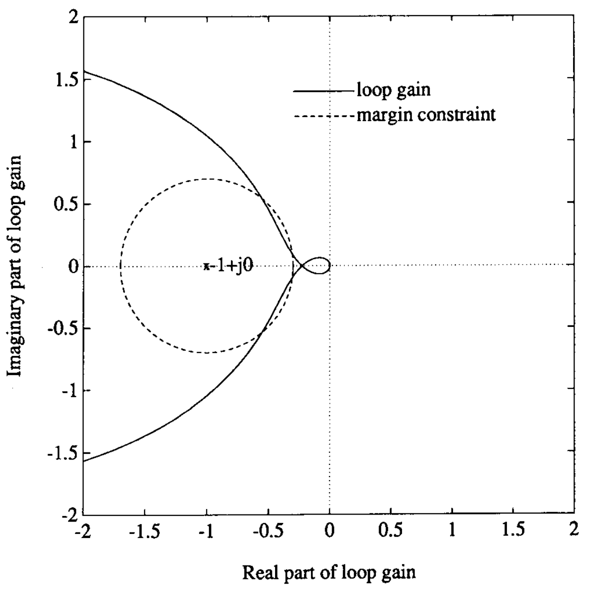


Fig. 11. Nyquist diagram is required to maintain a distance at least M for the point -1; shown is Nyquist diagram of loop gain for pointer system with nominal controller, with $M = 0.7$. Note that the nominal controller does not quite meet this constraint.

and hence, one might guess, would not result in convex constraints on H_{zw}. But in fact, this constraint can be recast as a maximum peaking constraint on a certain closed-loop map, as follows. A fictitious input LOOP_IN and a fictitious output LOOP_OUT are added to the original system, as shown in Fig. 10(c). Then the M-circle constraint described above is entirely equivalent to

$$\|H[\text{LOOP\_OUT}][\text{LOOP\_IN}]\|_\infty \leq M^{-1}$$

which is a constraint already discussed above. For our pointer example, to guarantee an M-circle radius of 0.7 we would use the constraint

$$\|H[V_m][n_{\text{act}}]\| \leq 0.7^{-1}. \tag{16}$$

This would guarantee a phase margin exceeding about 41 degrees and gain margins exceeding about $+10.4$, -4.6 dB (see Fig. 11).

- *Small Gain (Sufficient) Condition for Robustness:* The modern version of the previous constraint is considered in robustness theory. Briefly, a *sufficient* condition to stabilize not just the plant, but all plants near our given plant (in a peak frequency response deviation sense), may be formulated as a bound on certain closed-loop frequency response magnitudes (or singular values) [15], [6]; indeed in [7] a sufficient condition for robust disturbance rejection is given which is a bound on certain closed-loop frequency response magnitudes. These constraints have the form (15), and are convex constraints on H_{zw}.

Doyle [12] has demonstrated that these sufficient conditions for robustness of a control system can be quite conservative. His work has shown that a combination of good scaling along with the use of the constraints described above can greatly reduce their conservativeness. For a *fixed* choice of scaling $D(e^{j\Omega})$, we have the convex constraint

$$\sigma_{\max}(D(e^{j\Omega})H(e^{j\Omega})D(e^{j\Omega})^{-1}) \le M \qquad \text{for all } \Omega, \tag{17}$$

and the point is that, depending on D, (17) may be a far less conservative condition for robustness than (15). Unfortunately, if we seek the least conservative choice of D, say with the constraint

$$\min_{\text{appropriate } D} \sigma_{\max}(D(e^{j\Omega})H(e^{j\Omega})D(e^{j\Omega})^{-1}) \le M \qquad \text{for all } \Omega$$

we no longer have a convex constraint on H_{zw}. See, for example, Safonov [34].

• *Miscellaneous Bounds and Objectives:* We mention here some less common constraints. A *slew-rate* constraint on the step response, say s_{11}, may be enforced as

$$|h_{11}(t)| \le R \qquad \text{for all } t.$$

A *passivity* or *minimum dissipation* constraint, say on the 1, 1 entry of H_{zw}, can be formulated as

$$\Re H_{11}(e^{j\Omega}) \ge -D \qquad \text{for all } \Omega.$$

Such a constraint can be used, for example, to guarantee stability of the system with a saturating actuator. All constraints involving step responses can be generalized to other, arbitrary, input signals, or even sets of signals. Thus, we may specify a maximum peak tracking error for command inputs bounded by B_{cmd} and slew rate under R_{cmd}.

The boundary between constraint and objective is not sharp; we could, for example, try to minimize an overshoot in a certain step response, or put hard constraints on the rms response.

At this point it should be clear that a large number of practical constraints on, and objectives for, closed-loop system performance can be formulated as a standard convex program for the parameter Q. This program for Q is infinite dimensional and generally cannot be solved analytically except in special cases. The small rms disturbance response objective alone can be solved using Hiener–Hopf or LQG methods; certain forms of the small peak transfer function objective problem are solved by H_∞-optimal control theory; certain forms of the small peak disturbance objective problem are solved by the new l_1-optimal control theory introduced by Vidyasagar [37] and developed by Dahleh and Pearson [14]. None of these methods allows any inequality or time domain constraints.

We propose that this convex program be solved numerically. There has been some work on the numerical solution of infinite-dimensional convex programs [2]. Let us briefly describe a very naive method, in fact the one we have implemented. An *approximate* solution of the infinite-dimensional convex program for Q can be found numerically when the parameter Q is restricted to a large, but finite, dimensional space. Let

$$Q = \sum_{i=1}^{L} x_i Q_i$$

where $x_i \in \boldsymbol{R}$ and Q_i are fixed stable maps. For example, if Q is SISO, we could take $q_i(t) = \delta_{it}$, where q_i denotes the impulse response of Q_i, so that Q is a FIR filter with coefficients (tap weights) x_i. We will call $x = [x_1, \cdots, x_L]^T$ the *decision variables.* With this additional restriction, we have a standard finite-dimensional convex program in $\boldsymbol{R}^L$

$$\underset{x \in \tilde{\mathcal{K}}}{\text{minimize}}\, f(x) \tag{18}$$

where $f(x) = \tilde{\Phi}(\sum_{i=1}^{L} x_i Q_i)$ and $\tilde{\mathcal{K}} = \{x \in \boldsymbol{R}^L | \sum_{i=1}^{L} x_i Q_i \in \tilde{\mathcal{K}}\}$.

In our implementation we make use of one more naive approximation. Often the convex constraint set $\tilde{\mathcal{K}}$ involves *semi-infinite* constraints, for example, one frequency response constraint for each point on the unit circle in a peak transfer function constraint. We simply approximate these semi-infinite constraints by discretization, replacing a peak transfer function constraint by a very large number of single frequency constraints equally spaced on the unit circle. No doubt great improvements in performance would result from the use of sophisticated methods for semi-infinite constraints such as those described in Polak *et al.* [31].

We will demonstrate with a simple example how a control constraint is transformed into a convex constraint on the decision variables x_i in (18). For simplicity, we assume the system is discrete time and each of z, w, u, and y is scalar. Suppose the decision variables x_i are simply the impulse response of the Q filter, that is,

$$q(t) = \begin{cases} x_t, & t = 0, 1, \cdots, L-1, \\ 0, & t \ge L \end{cases}$$

where q is the impulse response of the Q filter. Let t_1, t_2, and t_3 denote the impulse responses of T_1, T_2, and T_3, and let h and s denote the impulse and step response, respectively, of H_{zw}. Of course, T_1, T_2, and T_3 are causal, so if k is negative, $t_i(k)$ is zero.

Suppose the specified constraint is on the step response:

$$\alpha \le s(i_0) \le \beta, \qquad i_0 \ge 0. \tag{19}$$

We will show that (19) is equivalent to a linear inequality constraint on x

$$\gamma \le c^T x \le \delta \tag{20}$$

for appropriate $\gamma, \delta \in \boldsymbol{R}$, $c \in \boldsymbol{R}^L$. Since $H_{zw} = T_1 + T_2 Q T_3$, we have

$$h(i) = t_1(i) + \sum_{k=0}^{i} \left(\sum_{j=0}^{L-1} x_j t_3(k-j) \right) t_2(i-k)$$

so that

$$s(i_0) = \sum_{i=0}^{i_0} \left(t_1(i) + \sum_{k=0}^{i} \sum_{j=0}^{L-1} x_j t_3(k-j) t_2(i-k) \right)$$

and (19) is equivalent to

$$\alpha - \sum_{i=0}^{i_0} t_1(i) \le \sum_{j=0}^{L-1} x_j \left(\sum_{i=0}^{i_0} \sum_{k=0}^{i} t_3(k-j) t_2(i-k) \right) \le \beta - \sum_{i=0}^{i_0} t_1(i)$$

which is (20) where

$$\gamma = \alpha - \sum_{i=0}^{i_0} t_1(i)$$

$$\delta = \beta - \sum_{i=0}^{i_0} t_1(i)$$

and

$$c_j = \sum_{i=0}^{i_0} \sum_{k=0}^{i} t_3(k-j) t_2(i-k), \qquad j = 0, 1, \cdots, L-1.$$

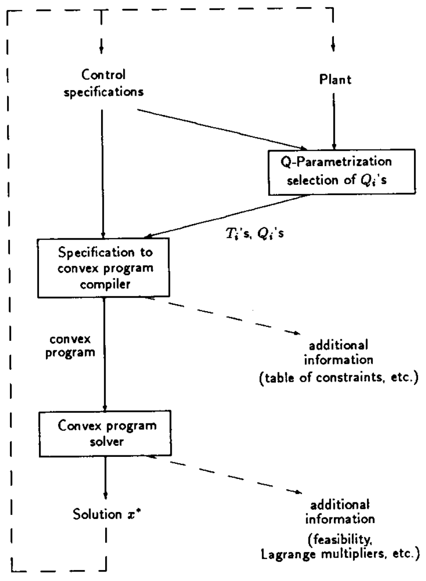

Control specifications
Plant
Q-Parametrization selection of Q_i's
T_i's, Q_i's
Specification to convex program compiler
convex program
additional information (table of constraints, etc.)
Convex program solver
additional information (feasibility, Lagrange multipliers, etc.)
Solution x^*

Fig. 12. Overall structure of design procedure.

Note that to evaluate γ, δ, and c, the impulse responses of T_i are required up to time i_0.

When there are several actuators and sensors, there are two additional summing loops involved in the translation, complicating the formulas above even more. Thus, while the translation from closed-loop constraints or objective on H_{zw} to convex program for the parameter Q is *mathematically* straightforward, the very simple example just given illustrates that the translation is quite cumbersome.

V. Specifications to Convex Program Compiler

We turn now to the *practical* problem of translating a large number of constraints on step responses, transfer functions, rms responses, and so on, along with a large number of objective terms, into the corresponding convex program for the decision variables.

We propose that this job be mechanized by the use of a *compiler* which accepts as input a list of closed-loop constraints and a description of the objective and as output produces the convex program for the decision variables. The constraints and objective would be specified in a *control specification language* (CSL), which would be natural for the control engineer, referring directly to step responses, noise powers, transfer functions, and so on. The output of the compiler would be a convex program which could be solved by a convex program solver such as NPSOL [20].

Fig. 12 is a flowchart of the whole design procedure. The engineer starts with a model of the plant and a set of control specifications given in the CSL. An estimate of the approximate bandwidth achievable with these constraints is formed, perhaps based on earlier design cycles (see the dashed arrow in Fig. 12). A nominal controller K_{nom} is designed which achieves this approximate bandwidth; this could be done using LQG or any other method. Coprime factorizations of K_{nom} and P_{yu} are formed; this step can also be done using LQG methods, that is, by solving a few Riccati equations [36]. T_1, T_2, and T_3 are determined from the coprime factorizations, and a set of Q_i's is selected.

The list of control specifications (in the CSL) along with T_1, T_2, T_3, Q_1, $\cdots$, Q_L form the input to the compiler. From these it produces a convex program in $\boldsymbol{R}^L$; it might also produce an auxiliary table relating the control specifications to the constraints in the convex program.

A convex program solver attempts to solve the convex program, perhaps using a previously designed Q as the starting condition. If successful, the solver finds a solution x^*; one useful additional output is a list of the Lagrange multipliers. Together with the auxiliary table generated by the compiler, we can find the sensitivities of the optimum objective with respect to each active constraint; based on this, the specifications might by tightened and some portion of the process repeated (see the dashed line).

If the solver finds the problem to be *infeasible*, it will produce x^* which minimizes some combination of the objective and a penalty function or some sum-of-infeasibilities if it is a Phase I/Phase II type algorithm. In this case the Lagrange multipliers are extremely useful since they tend to identify the offending constraints. After relaxing the specifications (dashed arrow terminating at *control specifications*) or changing the plant hardware (dashed arrow terminating at *plant*) the process is repeated.

We now discuss possible forms for a CSL. The discussion in Section IV suggests the overall structure

```
minimize     {
                 description of objective;
}
subject_to   {
                 first constraint;
                 ⋮
                 last constraint;
}
```

The constraints might have the forms encountered in Section IV such as given in the equations and inequalities (9)-(12), (14). To express, say, (10) and (12), we might use

```
H[THETA][DIST](1) = =0;
sum from t >= 0 (0.01*|h[V_MOTOR][CMD](t)|
        +0.03*|h[V_MOTOR][DIST](t)|
        +0.001*|h[V_MOTOR][SENS_NOISE](t)|) < = 50;
```

Similarly the objective (13) might be described as

```
1/(2*PI)* int from w = 0 to 2*PI
    (|H[THETA][DIST](w)|^2 *DIST_SPEC_DENS(w))
    +DELTA ^2 /12 * norm_h_sqr[THETA][SENS_NOISE];
```

Here we presume that the expressions DELTA, PI, and DIST_SPEC_DENS(w) have been previously defined.

The compiler we have implemented accepts a CSL of this basic form, although not of this generality; see Section VII. There are of course other forms a CSL might have. It is possible to have even lower level specifications than those mentioned above. To formulate specifications which ensure that

$$\text{whenever } \{\text{for all } t \ |d(t)| \le D_{\max}\}$$

$$\text{we have } \{\text{for all } t \ |\theta(t)| \le \theta_{\max}\};$$

we must specify the constraint

$$D_{\max} \sum_{t=0}^{\infty} |h[\theta][d](t)| \le \theta_{\max};$$

similarly to ensure that

whenever $\{n_{\text{sens}}$ is white with power $\Delta^2/12\}$

we have $\{E\theta^2 \le 0.03\}$;

we must specify the constraint

$$\frac{1}{2\pi}\,\|H[\theta][n_{\text{sens}}]\|_2^2\,\frac{\Delta^2}{12} \le 0.03.$$

If we refer to these syntactic constructions of the form

whenever {conditions on w}

we have {constraints on z};

as *behaviors,* then a control design problem could be specified as a *set of desired closed-loop behaviors.* Such a specification is even closer to the actual engineering problem than the language we discussed above; the disadvantage is that translating a set of desired behaviors into a convex program involves far more computation.

Generally, the use of a compiler to automate the controller design process allows the use of a specification of the design problem which is much closer to the actual engineering problem than is, say, a specification of a set of weight matrices for an LQG design. This may reduce the artistry required in controller design, or at the least, make more effective use of the artist's time.

VI. On Implementation of the Designed Controller

Let $Q^* = \sum_{i=1}^{L} x_i^* Q_i$ be a satisfactory Q for the control problem. The resulting controller

$$K^* = (Y - Q^*\tilde{N})^{-1}(X + Q^*\tilde{D}) \tag{21}$$

will generally be *full* (each input will affect every output) and have a large number of states. If the nominal controller is an estimated state feedback and the coprime factorization is that shown in Fig. 6, then the order of K^* will (generally) be the order of the plant plus the order of the entire Q-filter. At this point there are several alternatives.

1) Do not implement K^* at all; K^* is really only a *benchmark* against which to measure the performance of other (presumably much less complex) controllers.

2) Apply some sort of model reduction to arrive at a low-order controller, K_{red}, and then implement K_{red} in some standard way.

3) Implement the full K^* using a special purpose architecture.

We will discuss each of these alternatives in turn.

Alternative 1), while obvious, is a perfectly valid use of the design method proposed in this paper. The procedure finds the limit of achievable performance, defined in terms of the control specification, without regard for controller structure or complexity.

Alternative 2) is perhaps the most conventional route. Controller order reduction is by no means a straightforward task, indeed it is usually not even clear what criterion should be used to judge the *quality* or acceptability of a reduced-order controller [1]. In the context of a controller designed by the method proposed in this paper, however, there is a very natural criterion by which to judge the quality and acceptability of a proposed reduced-order controller: K_{red} is *acceptable* if and only if it stabilizes the plant and the resulting closed-loop system meets the constraints specified in the control specification; the *quality* of (an acceptable) K_{red} is then given by the relative increase in the objective specified in the CSL. Thus, the *performance of a reduced-order controller is judged precisely in terms of the control specifications already used to design the controller.*

We mention one ramification of the above principle. When designing a controller by the methods of this paper, it is tempting to crank down the constraints until they are very tight, just barely feasible. This makes the task of finding an acceptable reduced-order controller (in the sense defined above) extremely difficult, if not impossible. Thus, in the original design phase it is best to leave some room in the constraints, to be taken up in the model reduction phase.

We turn now to alternative 3), the least conventional alternative. We propose two methods for implementing the full K^*, both based on the Q-parameterization. Both methods rely on the availability and ease of VLSI implementation of many-tap FIR filters, used extensively in signal processing, for example, in equalizers for modems. Thus, controllers implemented in this way will tend to have far greater orders than is currently common.

A. Modified Nominal Controller Architecture

Consider the block diagram of the Q-parameterization given in Fig. 5. We propose that controllers *actually be built with the architecture shown in Fig. 5,* with Q realized (possibly) separately in hardware as an FIR filter. This architecture can be thought of as a simple modification of the nominal controller; the nominal controller must be modified to yield an auxiliary output signal e and accept an auxiliary input signal v. In some controllers, the signal e is easily available, even though Fig. 4 suggests that we need an additional $\tilde{N}$ and $\tilde{D}$. Similarly, injecting the signal v as shown in Fig. 4 is often straightforward.

If the nominal controller is an estimated state feedback, then as mentioned earlier, one choice for e is the output prediction error of the observer, and v can just be added to the output of the nominal controller, before the observer tap, as shown in Fig. 6. In this case the hardware and/or computational costs of modifying the nominal controller are slight, since the Q filter does all of the additional processing.

We note that this method can be used to improve *existing* controllers.

B. FIR Coprime Architecture

Generally, the nominal controllers which provide easy access to the auxiliary output e are those with complexity on the order of the plant, the estimated state feedback nominal controller a good example. The reason is, roughly speaking, that e is the pseudostate observer error, which means that the modified nominal controller really contains a pseudostate observer within it, which is likely to have the same complexity as the plant. This is admittedly vague, and is only offered as our interpretation.

In nominal controllers of much lower complexity than the plant, e.g., a diagonal PI controller, e is usually not easily accessible, and must be reconstructed. In this case, we propose that the entire controller be implemented using FIR filters.

Consider the coprime factorization of the controller K^* given in (21). Suppose Y, Q^*, $\tilde{N}$, X, and $\tilde{D}$ are all FIR, or very nearly approximated by FIR filters. Recall that unlike the plant or controller, these operators are *stable,* and hence their impulse matrices decay to zero. With proper choice of sample rate and parameterization, 50-tap FIR filters should be more than adequate. In this case, K^* can be realized as a feedback connection around one FIR filter, as we now show. K^* is governed by the equations

$$-Yu - Xy = v = -Q\tilde{N}u + Q\tilde{D}y$$

(see Fig. 4). We rewrite these equations as

$$u = (I - Y)u - Xy - v$$
$$v = -Q\tilde{N}u + Q\tilde{D}y$$

which we interpret as an FIR filter, F, with input u, y, and v, and output u and v, and

$$F = \begin{bmatrix} I - Y & -X & -I \\ -Q\tilde{N} & Q\tilde{D} & 0 \end{bmatrix}.$$

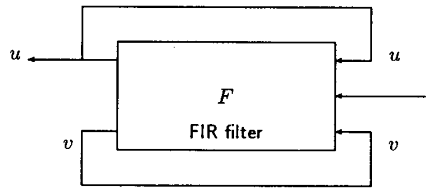


Fig. 13. FIR coprime architecture for controllers.

This is shown in Fig. 13. Of course there are many other F's which realize K^*, and we do not yet know a good procedure for picking one; this is an area for research.

A common criticism of the two architectures proposed above is that they will be too 'sensitive.' The argument is that an FIR filter is really a companion form (e.g., controllable canonical), and companion forms are notoriously 'numerically ill conditioned.' Let us respond to this criticism for the first architecture. What we will show is that the pole locations of the closed-loop system are indeed extremely sensitive, but nevertheless the closed-loop behavior of the system is *not* particularly sensitive.

With the architecture of Section VI-A, the Q filter will see no feedback. But with a practical observer, not quite matched to the actual plant, the closed-loop map from v to e will not be zero, but it will be small. By a small gain argument, suitably restricting the size of Q will guarantee stability of the closed-loop system with the mismatch between the plant and observer. Moreover, it is not hard to bound the variation in H_{zw} induced by variations in Q and small but nonzero closed-loop map from v to e induced by mismatch in the coprime factorizations. To be more explicit, let T_{mis} denote the (small) nonzero closed-loop transfer function from v to e in a practical implementation, and let δQ denote the perturbation in the Q filter, for example, that due to coefficient roundoff in a fixed point implementation. In this analysis we will ignore the perturbations in T_1, T_2, and T_3; it is an exercise to include these variations as well. In the following $\|\cdot\|$ will denote any operator norm. A small gain condition for stability is

$$\|T_{\text{mis}}\|(\|Q\|+\|\delta Q\|)<1;$$

assuming this to be satisfied, it can be shown that

$$\begin{aligned}&\|H_{zw,\text{desired}}-H_{zw,\text{actual}}\|\\ &\quad=\|T_2(Q+\delta Q)(I-T_{\text{mis}}(Q+\delta Q))^{-1}T_3-T_2QT_3\|\\ &\quad\le\frac{\|T_2\|\|T_3\|(\|\delta Q\|+\|Q\|\|T_{\text{mis}}\|(\|Q\|+\|\delta Q\|))}{1-\|T_{\text{mis}}\|(\|Q\|+\|\delta Q\|)}.\end{aligned}$$

Thus, small T_{mis} and small δQ guarantee not only closed-loop stability, but also *small variation in closed-loop response,* and that, almost by definition, is what we are interested in.

A very interesting observation is that while the closed-loop map varies only slightly with perturbations in Q and mismatch in the coprime factorizations, *the closed-loop poles can and do vary wildly.* A simplified example will demonstrate this. Let us consider the effect of wrapping a small constant feedback of 10^{-8} around a 100 tap SISO FIR filter whose coefficients are on the order of one. An argument as above shows that the perturbation in the I/O map of the filter is quite small, on the order of at most one part in 10^5. Thus, the I/O map hardly changes. On the other hand, let us now consider the effect of this small perturbation on the *poles.* With no perturbation they are all at zero; for an extremely small perturbation ϵ the poles move away from the origin on 100 3.6 degree spaced rays, at the incredible rate of $\epsilon^{1/100}$. In particular, the poles for our 10^{-8} perturbation could be anywhere *inside the unit disk.* We remind the reader that the I/O map of the filter has remained essentially unchanged.

The point is that for large-order systems, pole location may not be particularly meaningful, as in the example just presented. Even extreme pole sensitivity is possible in a functioning, well-designed robust engineering system. These points have broader implications, for example, for the eigenvalue assignment problem for large scale systems, and are not widely appreciated in the control community.

VII. Implementation of a Subset

We have implemented a compiler qdes which accepts a small but powerful subset of *control specification language.* The subset was carefully selected for ease of implementation and to allow interesting control design problems to be specified. In order to directly use the quadratic programming code LSSOL [19], we designed the subset so that our compiler output is a standard quadratic program which closely approximates the convex program for the decision variables. We are currently implementing a much more general compiler.

The system must be (single-rate) discrete-time. We restrict Q to be an n_sens input, n_act output FIR filter of length (in each channel) n_tap. Thus, the decision variables are

$$q[i][j](t),\ i=1,\cdots,\text{n\_act},\quad j=1,\cdots,\text{n\_sens},\ t=0,\cdots,\text{n\_tap}-1.$$

All calculations in qdes are based on the assumption that the T_i have impulse response matrices which are zero for $t\ \ge$ n_sample.

A control specification consists of up to three sections: declarations, constraints (or subject_to), and objective (or minimize). In the declarations section the parameters n_exog, n_reg, n_act, n_sens, n_tap, n_sample are specified, as well as the impulse matrices T_1, T_2, and T_3, and optionally an initial value for Q. More precisely, the first n_sample samples of these impulse matrices are specified. The (optional) constraints section consists of a list of constraints, and in the (optional) objective section the objective is specified as a sum of objective terms.

Let us first describe the constraints recognized in the constraints section. A constraint is simply a functional inequality, that is, an inequality involving some functional of the decision variables. Our compiler recognizes five basic affine functionals of the decision variables

$$\begin{array}{lll}\text{u\_step}[i][j](t), & i=1,\cdots,\text{n\_reg},\ j=1,\cdots,\text{n\_exog}, & t=0,\cdots,\text{n\_sample}-1,\\ \text{h}[i][j](t), & i=1,\cdots,\text{n\_reg},\ j=1,\cdots,\text{n\_exog}, & t=0,\cdots,\text{n\_sample}-1,\\ \text{q}[i][j](t), & i=1,\cdots,\text{n\_act},\ j=1,\cdots,\text{n\_sens}, & t=0,\cdots,\text{n\_tap}-1,\\ \text{Re\_H}[i][j](r,\theta), & i=1,\cdots,\text{n\_reg},\ j=1,\cdots,\text{n\_exog}, & r,\theta\in\mathbf{R}\\ \text{Im\_H}[i][j](r,\theta), & i=1,\cdots,\text{n\_reg},\ j=1,\cdots,\text{n\_exog}, & r,\theta\in\mathbf{R}.\end{array}$$

The expressions on the left are, respectively, the step and impulse responses from exogenous input j to regulated output i at time t [that is, $s_{ij}(t)$ and $h_{ij}(t)$], the impulse response of Q from its jth input to ith input at time t, and the real and imaginary part of the transfer function from exogenous input j to regulated output i at frequency $z = re^{j\theta}(\Re H_{ij}(re^{j\theta})$ and $\Im H_{ij}(re^{j\theta}))$.

Affine functional inequalities are allowed to take the forms

$$f\le e;$$
$$f\ge e;$$
$$f=e;$$
$$e\le f\le e;$$
$$|f|\le e$$

where f is one of the functionals above and e represents any *scalar expression.* Real scalar variables and scalar expressions involving standard mathematical functions are allowed.

Since we require our compiler output to be a standard quadratic program, which allows only linear (affine) constraints, we cannot directly handle inequality constraints involving the nonaffine functional $|H_{ij}(re^{j\theta})|$, such as $|H_{ij}(re^{j\theta})| \le e$, where e is some scalar expression. Instead, our compiler accepts inequalities of the form

$$\text{mag\_H}[i][j](r, \theta) \le e$$

but *approximates* them by

$$|\text{Re\_H}[i][j](r, \theta)| \le e\sqrt{\cos \pi/8};$$

$$|\text{Im\_H}[i][j](r, \theta)| \le e\sqrt{\cos \pi/8};$$

$$|\text{Re\_H}[i][j](r, \theta) + \text{Im\_H}[i][j](r, \theta)| \le e\sqrt{\frac{1}{2}\cos \pi/8};$$

$$|\text{Re\_H}[i][j](r, \theta) - \text{Im\_H}[i][j](r, \theta)| \le e\sqrt{\frac{1}{2}\cos \pi/8}$$

—at the expense of at most a ± 0.34 dB error. It could of course be modified to approximate the complex magnitude constraint as more (than four) affine constraints on the real and imaginary parts, yielding less error.

Several more complicated nonlinear convex functionals are recognized, among them

$$\begin{array}{ll} \text{max\_mag\_H}[i][j], & i=1, \cdots, \text{n\_reg}, \ j=1, \cdots, \text{n\_exog}, \\ \text{overshoot}[i][j], & i=1, \cdots, \text{n\_reg}, \ j=1, \cdots, \text{n\_exog}, \\ \text{undershoot}[i][j], & i=1, \cdots, \text{n\_reg}, \ j=1, \cdots, \text{n\_exog} \end{array}$$

which are, respectively, approximations to the maximum magnitude of the transfer function, maximum of the step response minus one, and minus the minimum of the step response, all from exogenous input j to regulated output i. The maximum magnitude is approximated by the maximum of the magnitude approximations described above at, typically, 2048 points on the unit disk. The overshoot and undershoot are approximated by substituting the maximum (minimum) of the step response up to time n_sample (as opposed to the maximum or minimum over all time).

We turn now to the objective section. The objective is a convex functional of the decision variables, which is described as the sum of any number of terms of the form *e* convex functional,* where the optional nonnegative scalar expression e allows weighting. *Convex functional* includes any of the functionals described above, such as max_mag_H, overshoot, or u_step, and in addition the following quadratic functionals:

$$\begin{array}{l} \text{h\_sqr}[i][j](t), \\ \text{mag\_H\_sqr}[i][j](r, \theta), \\ \text{q\_sqr}[i][j](t), \\ \text{norm\_h\_sqr}[i][j], \\ \text{norm\_q\_sqr}[i][j]. \end{array}$$

These are, respectively, $h_{ij}(t)^2$, $|H(re^{j\theta})|^2$, $q[i][j](t)^2$, $\Sigma_{t=0}^{\text{n\_sample}-1} h_{ij}(t)^2$, and $\Sigma_{t=0}^{\text{n\_tap}-1} q_{ij}(t)^2$, respectively.

Having described some of the functionals recognized, we turn now to some of the more general features. The standard C language preprocessor, cpp [24], [23] is used, which allows the use of macros and file include facilities, as well as C language style comments. The #define feature allows specification files to use the symbolic subscripts introduced in Section IV by placing

```
                        /* exogenous inputs */
#define CMD             1    /* command input */
#define DIST            2    /* disturbance torque */
#define SENS_NOISE      3    /* sensor noise */
#define LOOP_IN         4    /* for margin constraint */
                        /* regulated variables */
#define THETA           1    /* actual shaft angle */
#define MOTOR_V         2    /* motor voltage */
#define LOOP_OUT        2    /* for margin constraint */
```

near the top of the file. Thereafter, h[THETA][CMD](t) refers to $h_{11}(t)$, since cpp will convert h[THETA][CMD](t) to h[1][1](t).

Another more traditional use of the #define feature is to collect important constants or functions (which might change with new hardware or specification) in one place, e.g.,

```
#define DIST_REJ_BW 0.35
    /* dist. rejection bandwidth */
#define MARGIN    0.7  /* M-circle radius */.
```

To make repetitive constraints and objective terms easy to specify, (nested) looping is allowed, as in

```
for t=10 to n_sample-1
  1 - 0.80^t<=u_step[THETA] [CMD] (t)<=1 + 0.80^t;
```

A fairly complete specification file for our pointer system is given in the next section.

If qdes succeeds in its compilation, it invokes the quadratic programming code LSSOL; the final output starts with a listing of the form

functional value lower bd. upper bd. Lag. mult. status

where *status* is either blank or one of lb (lower bound), ub (upper bound), violates ub, or violates lb. The first functionals listed are simply the decision variables; then come all functionals cited in the constraints section; finally the objective value is listed. Thus, output has the form

```
q[1] [1] (0)                    0.12  -.90  0.90  0.00
                                        ⋮
q[n_act] [n_sens] (n_tap-1)     0.90  -.90  0.90  -.23  ub
u_step[THETA] [CMD] (0)         0.00  0.00  1.10  0.00  lb
                                        ⋮
final objective value = 0.0032.
```

qdes was implemented using the UNIX utilities lex and yacc [26], [22], [35], [23], a lexical analyzer generator and compiler compiler, respectively. yacc allows the user to describe a set of parsing rules along with code to execute (called *actions*) when an instance of a parsing rule is recognized. Our source code is in spirit quite like Section IV; yacc generated code recognizes the various constraints or objective terms, and then actions written in C perform the actual translation from control constraint to quadratic program constraint, as we did for the step response example at the end of Section IV.

VIII. Example

In this section we show how qdes might be used to design a controller for our pointer example. For purposes of presentation, we have limited the number of constraints and used a very simple objective. This example is presented only to illustrate how a program like qdes might be used, and is not presented as a particularly realistic or good controller design; indeed one of our points is that a real specification should be quite detailed.

In Section III we gave the various coprime factorizations required to compute T_1, T_2, and T_3 for our pointer with its nominal controller. These impulse matrices are specified in the declarations section. Recall that the nominal controller yielded asymptotic tracking, but not asymptotic disturbance rejection from the disturbance torque (see Figs. 8 and 9).

Our constraints are just equations (9)-(11), (14), (16) of Section IV. Thus, we will require asymptotic tracking from θ_{ref} (defined as CMD) to θ (THETA); the nominal controller meets this requirement. The step response from CMD to THETA will be constrained as shown in Fig. 8; note that the step response of the nominal design overshoots these constraints. We will require asymptotic rejection of constant disturbance torques in θ; our nominal design provides only 38 dB of asymptotic (DC) rejection (see Fig. 9). Of course, we could have designed a nominal controller which met this constraint, by augmenting our plant with an integrator, but we will see that qdes will automatically 'insert' an integrator, that is, a pole at 1 in the controller, to meet this constraint. To give an example of an H_∞ type constraint, we will insist on at least 40 dB of disturbance rejection from DIST to THETA for frequencies below 0.35 (14); this constraint too is violated by our nominal design (Fig. 9). Finally, we will require the controller to have M-circle margin at least 0.7 as shown in Fig. 11. The nominal controller violates this constraint as well.

We will take the objective to be the very simple (LQG) objective

$$\|H[\theta][n_{\text{sens}}]\|_2^2 + 100\|H[\theta][d]\|_2^2 + 0.0001\|H[V_m][\theta_{\text{ref}}]\|_2^2.$$

The two groups of #defines of Section VII appear at the top of our specifications file. The constraints and objective sections are:

```
minimize      {
    /* a simple LQG type objective */
    norm_h_sqr[THETA][SENS_NOISE];
    100*norm_h_sqr[THETA][DIST];
    .0001*norm_h_sqr[MOTOR_V][CMD];
}
subject_to  {
    /* bounds on step response from CMD to THETA */
    for t = 0 to 10
        0 <= u_step[THETA][CMD](t) <= 1.1;
    for t = 10 to n_sample-1
        1 - 0.80^t <= u_step[THETA][CMD](t)
    <= 1 + (0.80^t);

    /* asymptotic tracking */
    Re_H[THETA][CMD](1, 0) == 1.0;

    /* rejection of DIST over DIS_REJ_BW */
    max_mag_H[THETA][DIST](0,DIS_REJ_BW)
    <= 0.01;

    /* asymptotic rejection of DIST */
    Re_H[THETA][DIST](1, 0) == 0;

    /* margin constraint */
    max_mag_H[LOOP_OUT][LOOP_IN]
    <= 1/MARGIN;
}
```

The extra two arguments at the end of the maximum magnitude functional in the disturbance rejection line are lower and upper frequency limits. We also note that it is unnecessary to constrain the *imaginary* parts of transfer functions to be zero at DC; this is automatically satisfied.

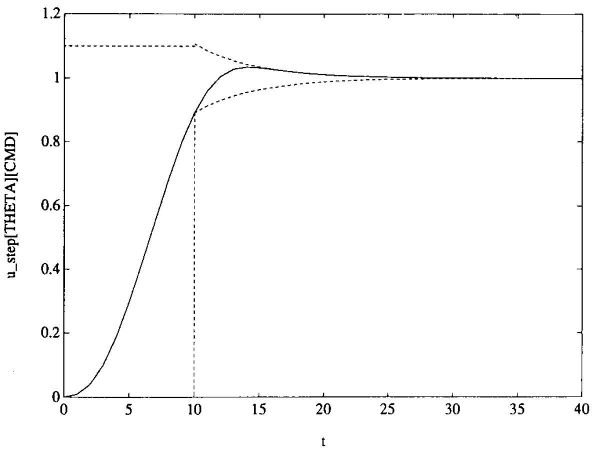


Fig. 14. Step response with designed controller. cf. Fig. 8.

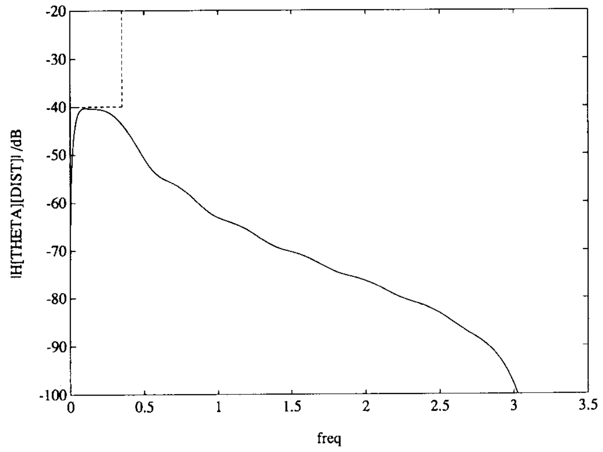


Fig. 15. Disturbance torque to shaft angle transfer function with designed controller. cf. Fig. 9.

These constraints are found to be feasible, and the resulting controller yields the command to angle step response shown in Fig. 14; this should be compared to the nominal response shown in Fig. 8. The disturbance torque to shaft angle transfer function with the designed controller is shown in Fig. 15 (cf. Fig. 9). One can see clearly that asymptotic rejection of constant torques has been achieved. The Nyquist plot of the loop gain with the designed controller is shown in Fig. 16 (cf. Fig. 11).

As mentioned above, the final controller has a pole at $z = 1$, a consequence of our requirement of asymptotic disturbance rejection. Aside from this pole at $z = 1$, however, the controller is stable. Of course, there is no guarantee in general that the designed controller will be (open loop) stable.

To give some idea of the computation involved, the quadratic program generated for this example had 2367 linear constraints. We first set n_tap = 5 (hence, 10 decision variables). Starting from the nominal controller ($Q = 0$) a feasible controller was found in 17 iterations, and the optimal controller in a total of 36 iterations. The 5-tap optimal controller yielded an objective value of 1.9. The total computation time for n_tap = 5 was 24 seconds CPU on a SUN 3/260 (68020 based) workstation with a floating point accelerator.

We then set n_tap = 15 (30 decision variables) and ran qdes using the optimal 5-tap Q coefficients, padded with 10 zeros in

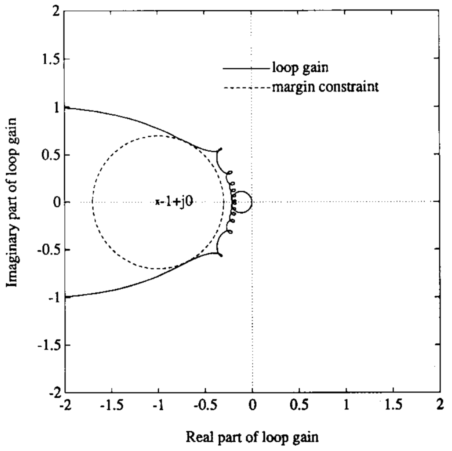


Fig. 16. Nyquist plot of loop gain with designed controller. cf. Fig. 11.

each channel, as the starting Q. The optimum 15-tap Q was found in 57 iterations, taking 40 seconds CPU time. The resulting objective was 0.72. Figs. 14–16 were constructed using the optimal 15-tap Q.

For n_tap > 15, the resulting optimal design hardly changed. The plots given in this paper were indistinguishable for n_tap = 15 and n_tap = 25. For n_tap = 25, the objective had dropped to 0.71. The initial 15 (pairs of) coefficients of the optimal 25-tap Q agreed very closely with the optimal coefficients of the 15-tap optimal Q, a good indication that 15 taps ($L = 30$) is yielding a good approximation to the infinite-dimensional problem.

Clearly the computation involved is orders of magnitude greater than, say, that involved in LQG or H_∞ controller design, but is nevertheless quite reasonable. Again, we emphasize that the method we used to approximate the infinite-dimensional convex program and the various semiinfinite constraints involved were naive, and not intended as the final word either in terms of accuracy or computation time.

IX. Conclusion

We close with some comments on the two basic schemes for controller design mentioned in Section I. The method we have described is in one sense in the second class—our algorithm finds a controller which is optimal in the sense of minimizing our objective while satisfying our constraints. (In another sense it is in the first class after we have chosen the Q_i, since then we have a (convex, however) parameter optimization problem). One big difference between the method described here and the other optimal controller design methods, say, LQG and H_∞, is that these methods produce a more or less explicit analytical solution; ours does not.[5]

It is very important to distinguish between algorithms which find a globally optimal controller and algorithms based on local parameter optimization techniques. Controller design methods based on local parameter optimization can have great advantages *when they succeed.* Controller structure and complexity can be constrained, and as a result the controllers produced are usually far simpler than those produced by other algorithms, especially the one we have described. For example, we may require a diagonal (decentralized) PI controller, which meets some set of specifications. Constraints such as stability of the controller or robustness to large plant deviations can be incorporated [32], [25].

The main disadvantage of such methods is that these algorithms are not *effective,* meaning that they are not guaranteed to find a solution if one exists, nor are they guaranteed to find the global minimum of the objective function. Thus, if a local method fails to find a diagonal PI controller which meets certain specifications, we cannot conclude that there is no diagonal PI controller which meets the specifications; the same algorithm with a different initial controller, or another algorithm could very well be successful. Similarly, we have no guarantee that the controller designed yields the smallest possible objective value. We hasten to add that in many applications, the requirement of low or fixed (e.g., decentralized) controller complexity will override the advantages of having an effective algorithm for controller design.

On the other hand, the method we describe *is* effective, since algorithms for convex programs are effective, meaning that the algorithms can determine if the program is feasible, and if so find the global minimum of the objective function. If we translate a certain set of specifications into a convex program which our convex program solver determines is not feasible, then we may conclude that no controller *of any form* will meet the specifications. Of course this must be qualified by reasonable choice of the T_i's and Q_i's, since in practice we do not consider the full infinite-dimensional convex program, but only a finite-dimensional approximation of it. Thus, if the design method we describe determines that some step response will always overshoot by at least 30 percent whenever a certain set of specifications is met, then 30 percent overshoot is close to the minimum achievable over all controllers meeting the specifications. A similar conclusion cannot be reached using any design algorithm based on local optimization.

Finally, we mention that our design method can be used to generate *upper bounds* on performance, which can be very useful information to a designer using a local parameter search method of control design. Here our method simply shows what performance is possible with no constraints on the controller structure. If a structured controller, say diagonal PI, yields a 40 percent overshoot, then the fact that any stabilizing controller which meets the specifications yields at least a 30 percent overshoot tells us our design cannot be improved much.

[5] The notion of explicit is quite vague, e.g., one could argue that an optimal controller described in terms of a quadratic program is just as explicit as an optimal controller described in terms of a stabilizing solution of a Riccati equation, since we have effective algorithms for solving both.

Acknowledgment

Various preliminary forms of this material were presented at seminars at Berkeley, USC, Cal Tech, and Honeywell SRC; several people made substantial suggestions (which the authors took): C. A. Desoer, J. Doyle, M. Safonov, P. Kokotovic, and R. L. Kosut. The authors thank the reviewers for careful, detailed, constructive reviews. They thank P. Gill, W. Murray, M. Saunders, and M. Wright of the Stanford Optimization Laboratory for providing the convex optimization code LSSOL [20], [21].

References

[1] B. D. O. Anderson, "Controller reduction: Concepts and approaches," in *Proc. Automat. Contr. Conf.,* 1987, pp. 1-9.

[2] E. J. Anderson and A. B. Philpott, Eds., *Infinite Programming* (Springer-Verlag Lecture Notes in Economics and Mathematical Systems). Sept. 1984.

[3] A. Bhaya and C. A. Desoer, "Necessary and sufficient conditions on $Q(= C(I + PC)^{-1})$ for stabilization of the linear feedback system $S(P, C)$," *Syst. Contr. Lett.,* vol. 7, no. 1, pp. 35-38, Feb. 1986.

[4] S. Blum and K. A. Fegley, "A quadratic programming solution of the minimum energy control problem," *IEEE Trans. Automat. Contr.,* vol. AC-13, pp. 206-207, Apr. 1968.

[5] A. E. Bryson and Y. C. Ho, *Applied Optimal Control.* New York: Hemisphere, 1975.

[6] M. J. Chen and C. A. Desoer, "Necessary and sufficient condition for robust stability of linear distributed feedback systems," *Int. J. Contr.,* vol. 35, no. 2, pp. 255-267, 1982.

[7] ——, "The problem of guaranteeing robust disturbance rejection in

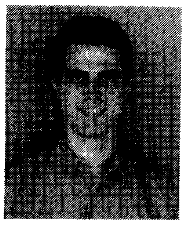
linear multivariable feedback systems," *Int. J. Contr.*, vol. 37, no. 2, pp. 305-313, 1983.
[8] A. J. Calise and K. A. Fegley, "Quadratic programming in the statistical design of sampled-data control systems," *IEEE Trans. Automat. Contr.*, pp. 77-80, Feb. 1968.
[9] C. A Desoer and C. L. Gustafson, "Algebraic theory of linear multivariable feedback systems," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 909-917, Oct. 1984.
[10] ——, "Design of multivariable feedback systems with simple unstable plant," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 901-908, Oct. 1984.
[11] C. A. Desoer, R.-W. Liu, J. Murray, and R. Saeks, "Feedback system design: The fractional representation approach to analysis and synthesis," *IEEE Trans. Automat. Contr.*, vol. AC-25, pp. 399-412, June 1980.
[12] J. C. Doyle, "Analysis of feedback systems with structured uncertainties," *IEE Proc.*, vol. 129, Nov. 1982.
[13] ——, "Matrix interpolation theory and optimal control," Ph.D. dissertation, Univ. Calif., Berkeley, CA, 1984.
[14] M. A. Dahleh and J. B. Pearson, "l^1-optimal feedback controllers for MIMO discrete time systems," Dept. Elec. Eng., Rice Univ., Houston, TX, Tech. Rep. 8502, Feb. 1986.
[15] J. C. Doyle and G. Stein, "Multivariable feedback design: Concepts for a classical/modern synthesis," *IEEE Trans. Automat. Contr.*, vol. AC-26, pp. 4-16, Feb. 1981.
[16] K. A. Fegley, "Designing sampled-data control systems by linear programming," *IEEE Trans. Appl. Ind.*, vol. AI-83, pp. 198-200, May 1964.
[17] K. A. Fegley and M. I. Hsu, "Optimum discrete control by linear programming," *IEEE Trans. Automat. Contr.*, vol. AC-9, pp. 114-115, Jan. 1964.
[18] B. A. Francis and G. Zames, "On h_∞-optimal sensitivity theory for SISO feedback systems," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 9-16, Jan. 1984.
[19] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright, "User's guide for LSSOL (Version 1.0): A Fortran package for constrained least-squares and convex quadratic programming," Dep. Operat. Res., Stanford Univ., Stanford, CA, Tech. Rep. SOL 86-1, Jan. 1986.
[20] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "User's guide for NPSOL (Version 4.0): A Fortran package for nonlinear programming," Dep. Operat. Res., Stanford Univ., Stanford, CA, Tech. Rep. SOL 89-2, Jan. 1986.
[21] P. E. Gill, W. Murray, and M. Wright, *Practical Optimization*. New York: Academic, 1981.
[22] S. C. Johnson, "Yacc: Yet another compiler compiler," in *UNIX Programmer's Manual, Supplementary Documents*, Univ. California, Berkeley, CA, 4.2 Berkeley Software Distribution, 1984.
[23] B. W. Kernighan and R. Pike, *The UNIX Programming Environment*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[24] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
[25] U.-L. Ly, A. E. Bryson, and R. H. Cannon, "Design of low-order compensators using parameter optimization," in *Applications of Nonlinear Programming to Optimization and Control*, IFAC, 1983.
[26] M. E. Lesk and E. Shmidt, "Lex—A lexical analyzer generator," in *UNIX Programmer's Manual, Supplementary Documents*, Univ. Calif., Berkeley, CA, 4.2 Berkeley Software Distribution, 1984.
[27] D. Q. Mayne, E. Polak, and A. Sangiovanni-Vincentelli, "Computer aided design via optimization," in *Control Applications of Nonlinear Programming*, IFAC, 1979.
[28] C. N. Nett, "Algebraic aspects of linear control system stability," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 941-949, Oct. 1986.
[29] G. Porcelli and K. A. Fegley, "Linear programming design of digitally compensated systems," in *Proc. Joint American Contr. Conf.*, 1964.
[30] G. Porcelli and K. A. Fegley, "Optimal design of digitally compensated systems by quadratic programming," *J. Franklin Inst.*, vol. 282, pp. 303-317, 1966.
[31] E. Polak, D. Q. Mayne, and D. M. Stimler, "Control system design via semi-infinite optimization: A review," *Proc. IEEE*, vol. 72, pp. 1777-1794, Dec. 1984.
[32] E. Polak, P Siegel, T. Wuu, W. T. Nye, and D. Q. Mayne, "DELIGHT.MIMO: An interactive, optimization-based multivariable control system design package," in *Computer-Aided Control Systems Engineering*, M. Jamshidi and C. J. Herget, Eds. Amsterdam, The Netherlands: North-Holland, 1985; reprinted from *IEEE Contr. Syst. Mag.*, no. 4, Dec. 1982.
[33] J. R. Raggazani and G. F. Franklin, *Sampled-Data Control Systems*. New York: McGraw-Hill, 1958.
[34] M. G. Safonov, "Optimal diagonal scaling for infinity-norm optimization," *Syst. Contr. Lett.*, vol. 7, pp. 257-260, 1986.
[35] A. T. Schreiner and H. G. Friedman, Jr., *Introduction to Compiler Construction with UNIX*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[36] M. Vidyasagar, *Control System Synthesis: A Factorization Approach*. Cambridge, MA: M.I.T. Press, 1985.
[37] ——, "Optimal rejection of persistent bounded disturbances, *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 527-535, June 1986.
[38] D. C. Youla, H. A. Jabr, and J. J. Bongiorno, Jr., "Modern Wiener-Hopf design of optimal controllers-Part II: The multivariable case," *IEEE Trans. Automat. Contr.*, vol. AC-21, pp. 319-338, June 1976.
[39] L. A. Zadeh and B. H. Whalen, "On optimal control and linear programming," *IRE Trans. Automat. Contr.*, pp. 45-46, July 1962.

Stephen P. Boyd (S'82-M'85) received the A.B. degree in mathematics, summa cum laude, from Harvard University, Cambridge, MA, in 1980, and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1985.

Since 1985 he has been an Assistant Professor in the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA.

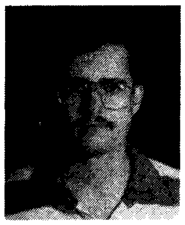
Venkataramanan Balakrishnan received the B. Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 1985.

He is currently pursuing the M.S./Ph.D. degree in electrical engineering at Stanford University, Stanford, CA.



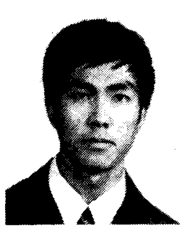
Craig H. Barratt received the B.Sc. degree in mathematics and physics in 1983 and the B.E. degree (Hons.) in electrical engineering in 1985 from the University of Sydney, Sydney, Australia, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1987.

He is currently pursuing the Ph.D. degree in electrical engineering at Stanford.



Nasser M. Khraishi (S'82) was born in Kuwait in 1962. He received the B.Sc. degree in electrical engineering from Kuwait University, Kuwait, in 1984, and the M.Sc. degree in engineering-economic systems from Stanford University in 1985.

He is currently pursuing the Ph.D. degree in electrical engineering at Stanford University, Stanford, CA. His current research interests include optimization, mathematical programming, and their links to system theory and controller design.



Xiaoming Li (S'86) was born in Henan, China, on July 4, 1963. He received the B.S. degree in automatic control from Tsinghua University, Beijing, China, in 1984 and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1986.

At present, he is a Ph.D. candidate in the Information Systems Laboratory at Stanford University. Since 1986 he has been a Research Assistant in the Department of Electrical Engineering, Stanford University, Stanford, CA. His research interests are in the areas of computer-aided-design of control systems, system implementation, and adaptive signal processing.

David G. Meyer received the B.S. degree in math and the B.S.E.E. degree from the University of Wyoming, Laramie, in 1982, the M.S.E.E. degree from Stanford University, Stanford, CA, in 1985, and the Ph.D. degree in electrical engineering from Stanford University in 1987.

Since 1987 he has been an Assistant Professor of Electrical Engineering in the Robotics and Control Laboratory, University of Virginia, Charlottesville, His current research interests include computer-aided design, multirate sampled-data systems, and the analysis of expert control systems.

Stephen A. Norman received the B.Sc. degree in mathematics and engineering from Queen's University, Kingston, Ont., Canada, in 1985, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1986.

He is currently pursuing the Ph.D. degree in electrical engineering at Stanford University.