THEORETICAL AND SOFTWARE ASPECTS OF OPTIMIZATION-BASED CONTROL SYSTEM DESIGN

by

E. Polak and D. Q. Mayne

Memorandum No. UCB/ERL M84/23

1 March 1984

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

THEORETICAL AND SOFTWARE ASPECTS OF OPTIMIZATION-BASED CONTROL SYSTEM DESIGN

by

E. POLAK
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

D. Q. MAYNE
Department of Electrical Engineering
Imperial College, London SW 2BT, England

1. INTRODUCTION.

Parametric optimization is a powerful tool for the selection of favorable values for design variables. In linear control system design, parametric optimization has been used for at least twenty years: since the introduction of the linear-quadratic regulator problem. Although linear quadratic regulator theory was derived in the context of optimal control theory rather than mathematical programming theory, it is nevertheless true that the optimal gain matrix for the linear-quadratic regulator problem is a solution to an unconstrained parametric optimization problem of the form

$$\min_{K \in R^{m \times n}} f(K) \tag{1.1}$$

where $f(K)$ is the largest eigenvalue of a symmetric matrix of the form

$$\int_0^\infty \exp[t(A+BK)]^T (Q+K^T R K) \exp[t(A+BK)]\,dt \tag{1.2}$$

with Q symmetric and positive semi-definite and R symmetric and positive definite.

In keeping with the state of the art in constrained optimization of the sixties (see e.g., [Ath.1]), the cost function $f(K)$, in the linear quadratic regulator problem, expresses a penalty function approach to the satisfaction of performance requirements. Over the last eight years a much more powerful approach has become possible with the development of semi-infinite optimization algorithms for engineering design (see e.g., [Gon.1, Pol.1, Pol.2, Pol.3]).

The major advantage of semi-infinite optimization over the penalty function approach implicit in LQR theory, is that it accepts as constraints exact bounds on time and frequency responses over time and frequency intervals. These bounds take on the form of infinite systems of inequalities which are parametrized by time or frequency, that the finite number of designable compensator parameters must satisfy (see e.g. [Pol.1]). Such inequalities are often refered to as *semi-infinite* inequalities. Clearly, semi-infinite optimization opens up totally new possibilities in control system design. This is particularly so since the introduction of nondifferentiable optimization theory has led to algorithms that can solve not only constraints on time and frequency responses which are (pointwise) differentiable, but also on eigenvalues of system matrices and singular values of transfer function matrices, which are not differentiable everywhere (see [Gon.1, Pol.2, Pol.3]). This paper summarizes our work on the use of semi-infinite optimization in control system design.

2. CONTROL SYSTEM DESIGN VIEWED AS AN OPTIMIZATION PROBLEM

Whenever design specifications include envelope constraints on time and frequency responses, control system design problems transcribe into semi-infinite optimization problems with, possibly, a dummy cost. These problems have the form

$$\min_{x \in \mathbb{R}^n} \{ f(x) \mid g^j(x) \leq 0,\ j = 1,2,\ldots,m;\ \varphi^k(x,\alpha_k) \leq 0,\ k = 1,2,\ldots,l,\ \forall\ \alpha_k \in A_k \} \quad (2.1)$$

where $x \in \mathbb{R}^n$ is the design vector (designable compensator parameters); $f: \mathbb{R}^n \to \mathbb{R}$, $g^j: \mathbb{R}^n \to \mathbb{R}$, $\varphi^k: \mathbb{R}^n \times \mathbb{R}^{p_k} \to \mathbb{R}$ are locally Lipschitz continuous and the sets $A_k \subset \mathbb{R}^{p_k}$ are compact. Most often, $p_k = 1$ so that the A_k are intervals (of time or frequency). Only in the case of parametric dynamic model uncertainty does one need to introduce sets A_k of higher dimension.

A simple example will illustrate the genesis of forms such as (2.1) in control system design. Consider the control system in Fig. 1, for which it is necessary to design a compensator C. The dynamics of this system are given by

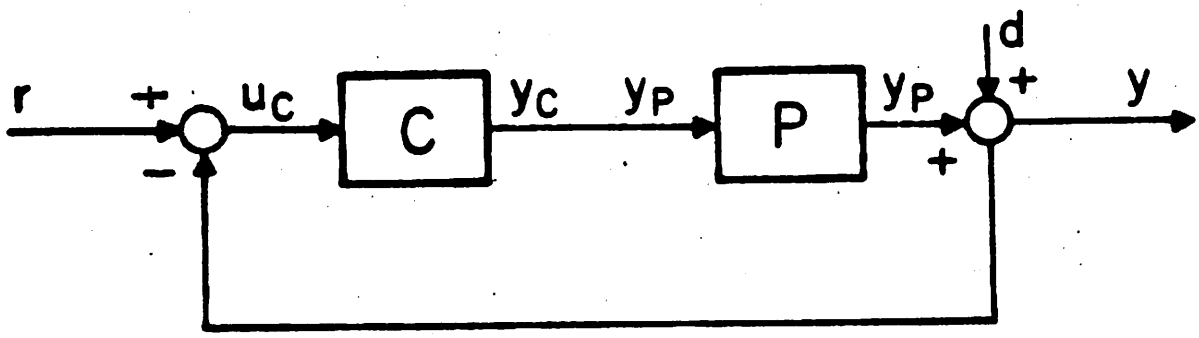

Fig. 1

$$\left.\begin{aligned} \dot{z}_P &= A_P z_P + B_P u_P \\ y_P &= C_P z_P + D_P u_P \end{aligned}\right\} \tag{2.2a}$$

$$\left.\begin{aligned} \dot{z}_C &= A_C z_C + B_C u_C \\ u_C &= C_C z_C + D_C u_C \end{aligned}\right\} \tag{2.2b}$$

$$\left.\begin{aligned} u_P &= y_C \\ u_C &= -y_P - d + r \end{aligned}\right\} \tag{2.2c}$$

where (2.2a) specifies the plant dynamics, (2.2b) specifies the compensator dynamics and (2.2c) specifies the system interconnection. The designer chooses the dimension of the compensator state vector z_C and has to compute the compensator matrices A_C, B_C, C_C, D_C, whose elements eventually form the components of the design vector x. In order to reduce the dimension of the design vector x, the designer may specify the system matrix A_C in block diagonal form

$$A_C = \text{diag}(A_{1C}, A_{2C}, \ldots, A_{kC}, \lambda_{2k+1}, \ldots, \lambda_N) \tag{2.3a}$$

where the λ_j are real (some may be frozen at zero for integral action), while

$$A_{jC} = \begin{bmatrix} 0 & 1 \\ a_{1j} & a_{0j} \end{bmatrix} \tag{2.3b}$$

Some structural simplification of the B matrix is also possible.

Now consider typical constraints.

(i) **Time Domain:** Given step inputs $r_i(t) = (0,0,\ldots,1,0\ldots0)$ (with the 1 in the i^{th} place), we require that the corresponding step responses $y^i(t,x,r_j)$ remain within the envelopes shown in Fig. 2. This leads to the two semi-infinite constraints

$$y^i(x,t,r_j) - b_i^{ij}(t) \le 0 \quad \forall\, t \in [0,T] \tag{2.4a}$$

$$-y^i(x,t,r_j) + b_i^{ij}(t) \le 0 \quad \forall\, t \in [0,T] \tag{2.4b}$$

We note that when $i \neq j$ (2.4a,b) express limitations on the permissible interaction.

(ii) **S-plane:** The system matrix of the closed loop system has the form

$$A(x) = \begin{bmatrix} A_P - B_P(I+D_C D_P)^{-1} D_C C_P & -B_P(I+D_C D_P)^{-1} C_C \\ B_C(I+D_P D_C)^{-1} C_P & A_C - B_C(I+D_P D_C)^{-1} D_P C_C \end{bmatrix} \tag{2.5}$$

We may require that all the eigenvalues of this matrix lie in a cone

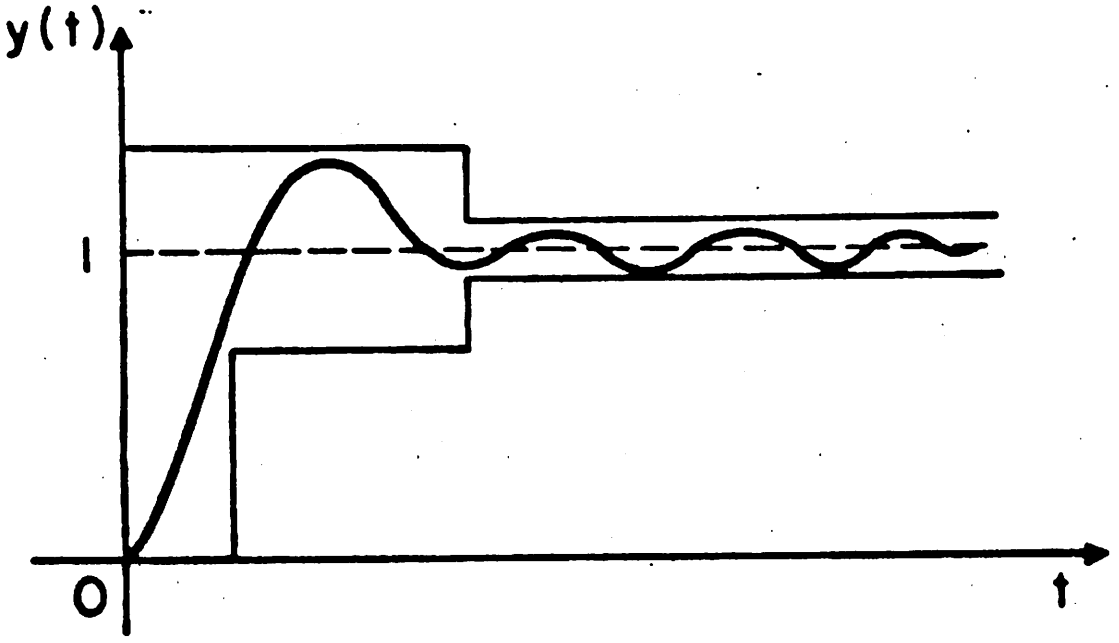

Fig. 2

in $\dot{C}_-$. Denoting the eigenvalues of $A(x)$ by $\lambda^j[A(x)]$, we are lead to the system of inequalities

$$\text{Im}[\lambda^j[A(x)]] + \xi[\text{Re}\lambda^j[A(x)]] + \zeta \leq 0 \quad \text{for } j = 1,2,\ldots,N \tag{2.6}$$

where $\xi, \zeta > 0$. Note that in (2.6) we are exploiting the fact that complex eigenvalues must come in conjugate pairs. Alternatively, the same result could have been specified in semi-infinite inequality form via the modified Nyquist criterion described in [Pol. 4].

(iii) **Frequency Domain:** Assuming that there is some unstructured uncertainty in the plant model, so that (2.2a) represents only the structured part P_0, while the actual plant has a transfer function matrix of the form $P(s) = P_0(s)(I + L(s))$, with $P_0(s)$ the transfer function matrix of (2.2a) and $L(s)$ a perturbation known only to the extent that

$$\bar{\sigma}_M[L(j\omega)] \leq b(\omega) \; \forall \; \omega \geq 0 \tag{2.7}$$

where $\bar{\sigma}[\cdot]$ denotes the largest singular value. In that case (see [Che. 1]), for the closed loop system to be stable, we require, in addition to (2.6), that

$$\bar{\sigma}[H^0_{yr}(x,j\omega)] - \frac{1}{b(\omega)} \leq 0 \; \forall \; \omega \geq 0 \tag{2.8}$$

where $H^0_{yr} = (I + P_0C)P_0C$. Clearly, (2.8) is a semi-infinite inequality.

Finally, we may elect to minimize the influence of the disturbance over a critical frequency range, which leads to the definition of the cost function (see [Doy. 1])

$$f(x) = \max_{\omega \in [\omega', \omega'']} \bar{\sigma}[H^0_{yd}(x,j\omega)] \tag{2.9}$$

where $H^0_{yd} = (I + P_0C)^{-1}$.

We see that this simple design example has the form of the problem (2.1). Note that in view of this example, one may not change the hypothesis that the functions in (2.1) are only locally Lipschitz continuous to the assumption that they are continuously differentiable. It should be pointed out that a number of the functions in the constraints are locally Lipschitz continuous only in the subset of the parameter space where the closed loop system is stable. Because of this, the algorithms discussed in the next section have to be slightly modified so as to preserve stability

throughout the entire optimization.

3. SEMI-INFINITE OPTIMIZATION ALGORITHMS FOR CONTROL SYSTEM DESIGN

In the preceeding section, we saw that control system design problems reduce to optimization problems with constraints of the form $\max_{y_j \in Y_j} \varphi^i(x,y_j) \leq 0$, where the y_j represent either time or frequency. To simplify notation, we concentrate on the simplest problem in this class:

$$\min\{f(x) \mid \varphi(x,y) \leq 0 \ \forall y \in Y\} \tag{3.1}$$

where $f:\mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and $\varphi:\mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is locally Lipschitz continuous and $Y \subset \mathbb{R}$ is a compact interval.

Our favorite algorithms for solving (3.1) are semi-infinite phase I-phase II methods [Pol.5, Gon.1, Pol.2, Pol.3] obtained by extension of simple methods of feasible directions (see [Pol.7]). These methods compute a feasible point (i.e., a design satisfying specifications) very rapidly. After that, they reduce the cost without violation of specifications. Since to a large measure, design consists of simply satisfying specifications, the advantage of the phase I-phase II methods is clear.

It is easiest to understand the principles governing these phase I-phase II algorithms by considering at first simpler problems such as

$$\min\{f(x) \mid g^j(x) \leq 0, j \in \underline{m}\} \tag{3.2}$$

where $f:\mathbb{R}^n \to \mathbb{R}$, $g^j:\mathbb{R}^n \to \mathbb{R}$, $j \in \underline{m}$ are continuously differentiable and $\underline{m} \triangleq \{1,2,\ldots,m\}$. For any $x \in \mathbb{R}^n$ let

$$\psi(x) \triangleq \max_{j \in \underline{m}} g^j(x) \tag{3.3a}$$

$$\psi_+(x) \triangleq \max\{0,\psi(x)\} \tag{3.3b}$$

and for any $\varepsilon \geq 0$, and $x \in \mathbb{R}^n$ let

$$I_\varepsilon(x) \triangleq \{j \in \underline{m} \mid \psi_+(x) - f^j(x) \geq \varepsilon\} \tag{3.3c}$$

Next we recall that the directional derivatives of $f(\cdot)$ and $\psi_+(\cdot)$ at x in the direction $h \neq 0$, are given by $df(x;h) = \langle f(x), h \rangle$ while

$d\psi_+(x;h) = \max_{j \in I_0(x)} \{<\nabla g^j(x), h>, 0\}$ when $\psi(x) = 0$, $d\psi_+(x;h) = \max_{j \in I_0(x)} <\nabla g^j(x), h>$ when $\psi(x) > 0$, and $d\psi_+(x;h) = 0$ when $\psi(x) < 0$. In order to avoid zig-zagging, it is common to introduce an ε-directional derivative of $\psi_+(x)$, defined, for $\varepsilon \geq 0$ by

$$d_\varepsilon\psi_+(x;h) \triangleq \begin{cases} \max_{j \in I_\varepsilon(x)} <\nabla g^j(x), h> & \text{if } \psi(x) \geq -\varepsilon \\ 0 \quad \text{if } \psi(x) < -\varepsilon \end{cases} \tag{3.4}$$

Note that for any x such that $\psi(x) \geq 0$, $d\psi(x;h) \leq d_\varepsilon\psi_+(x;h)$ for all $h \in \mathbb{R}^n$ and every $\varepsilon \geq 0$. A phase I-phase II *search direction* can now be defined by

$$h_\varepsilon(x) \triangleq \text{argmin } \{\tfrac{1}{2}\|h\|^2 + \max \{df(x;h) - \gamma\psi(x)_+, d_\varepsilon\psi(x;h)\}\} \tag{3.5}$$

where $\gamma > 0$. Let $\vartheta_\varepsilon(x)$ denote the *value* of the quadratic program (3.5). Clearly, if $\vartheta_\varepsilon(x) < 0$, then a) if $\psi_+(x) = 0$, then $h_\varepsilon(x)$ is a feasible usable direction (can decrease cost without constraint violation); b) if $\psi_+(x) > 0$, we get a direction of reduction of constraint violation, mitigated by the need to reduce the cost $f(x)$ as $\psi(x)$ approaches zero. Finally, it is necessary to reduce the anti-zigzagging precautions as a solution point is approached. This can be done by defining, with $\nu \in (0,1)$,

$$E \triangleq \{0, 1, \nu, \nu^2, \nu^3, \cdots\} \tag{3.6a}$$

and

$$\varepsilon(x) \triangleq \max\{\varepsilon \in E \mid \vartheta_\varepsilon(x) \leq -\varepsilon\} \tag{3.6b}$$

Putting together the elements we have, we obtain

Algorithm 3.1:

Parameters: $\alpha, \beta, \gamma \in (0,1)$, $\gamma > 0$.

Data: $x_0 \in \mathbb{R}^n$.

Step 0: Set $i = 0$.

Step 1: Compute $\varepsilon(x_i)$ and the search direction $h_i \triangleq h_{\varepsilon(x_i)}(x_i)$.

Step 2: Compute the step size λ_i,

$$\lambda_i = \arg\max_{k \in \mathbb{N}_+} \{\beta^k \mid \psi(x_i + \beta^k h_i) - \psi(x_i) \leq -\beta^k \alpha\varepsilon(x_i)\} \text{ if } \psi(x_i) > 0,$$

$$\lambda_i = \arg\max_{k \in \mathbb{N}_+} \{\beta^k \mid f(x_i + \beta^k h_i) - f(x_i) \leq -\beta^k \alpha\varepsilon(x_i);\ \psi(x_i + \beta^k h_i) \leq 0\} \tag{3.7}$$

if $\psi(x_i) \leq 0$

Step 3: Update: set $x_{i+1} = x_i + \lambda_i h_i$, replace i by $i + 1$ and go to step 1.

■

Referring to [Pol.5] we find the following result.

Theorem 3.1: Suppose that $0 \notin \text{co}\{\nabla g^j(x)\}_{j \in I_0(x)}$ for all $x \in \mathbb{R}^n$ such that $\psi(x) \geq 0$. If $\hat{x}$ is an accumulation point of $\{x_i\}_{i=0}^{\infty}$ constructed by Algorithm 3.1, then $\psi(\hat{x}) \leq 0$ and $\vartheta_0(\hat{x}) = 0$ (i.e., $\hat{x}$ satisfies the F. John condition of optimality (see [Clar. 1])). ■

In order to extend Algorithm 3.1 to nondifferentiable problems, we begin by refining the ε-search direction finding problem (3.5) by introducing an additional real variable h^0 so that we move to $\mathbb{R}^{n+1}$ and introduce the vector $\bar{h} = (h^0, h)$, with $h \in \mathbb{R}^n$, and we replace $\gamma\psi(x)_+$ by the computationally more desirable factor $\sqrt{\gamma\psi(x)_+}$. The new ε-search direction finding problem becomes

$$\bar{h}_\varepsilon(x) = (h_\varepsilon^0(x), h_\varepsilon(x)) \triangleq \text{argmin}\{\tfrac{1}{2}\|\bar{h}\|^2 \tag{3.8a}$$

$$+ \max\{df(x;h) - \sqrt{\gamma\psi_+(x)}h^0, d_\varepsilon\psi(x;h)\}$$

and by making use of the Von Neumann minmax theorem, we obtain from (3.8a) that

$$\bar{h}_\varepsilon(x) = \text{argmin}\{\tfrac{1}{2}\|\bar{\xi}\|^2 \mid \bar{\xi} \in G_\varepsilon^{f,\psi}(x)\} \tag{3.8b}$$

where

$$G_\varepsilon^{f,\psi}(x) \triangleq \text{co}\left\{\begin{bmatrix}\sqrt{\gamma\psi_+(x)} \\ \nabla f(x)\end{bmatrix}, \begin{bmatrix}0 \\ \nabla g^j(x)\end{bmatrix}\right\}_{j \in I_\varepsilon(x)} \tag{3.9}$$

It is easy to show that replacing $h_\varepsilon(x)$, as computed by (3.5) by $h_\varepsilon(x)$ as computed by (3.8b) and replacing $\vartheta_\varepsilon(x)$ computed as the value of (3.5) by $\vartheta_\varepsilon(x)$ computed as the value of (3.8b), in Algorithm 3.1, does not affect the truth of Theorem 3.1. We therefore assume from now on that $h_\varepsilon(x)$ and $\vartheta_\varepsilon(x)$ in Algorithm 3.1 are computed using (3.8b). First we extend Algorithm 3.1 to the case of problem (3.1) where $\nabla_x\varphi(x,y)$ exists and is continuous (cf. [Gon. 1]). For this case, for any $x \in \mathbb{R}^n$ and $\varepsilon \geq 0$, we define

$$\tilde{Y}_\varepsilon(x) \triangleq \{y \in Y \mid \psi_+(x) - \varphi(x,y) \leq \varepsilon\} \tag{3.10a}$$

where, now, $\psi(x) \triangleq \max_{y \in Y} \varphi(x,y)$ and $\psi_+(x) = \max\{0, \psi(x)\}$, as before, and

$$\hat{Y}_\varepsilon(x) \triangleq \{y \in Y_\varepsilon(x) \mid y \text{ is a local maximizer of } \varphi(x,\cdot) \text{ in } Y\} \quad (3.10b)$$

Engineering considerations allow us to assume that $Y_\varepsilon(x)$ is a finite set (see [Pol.1]). To extend Algorithm 3.1 to the case of Problem (3.1) in question, we define

$$G_\varepsilon^{f,\psi}(x) \triangleq \text{co}\left\{\begin{bmatrix}\sqrt{\gamma\psi_+(x)}\\ \nabla f(x)\end{bmatrix}, \begin{bmatrix}0\\ \nabla_x\varphi(x,y)\end{bmatrix}\right\}_{y \in Y_\varepsilon(x)} \quad (3.11)$$

Substituting from (3.11) into (3.8b), we get a finite quadratic, program in baricentric co-ordinates, to solve in computing $\bar{h}_\varepsilon(x)$. Referring to [Gon. 1], we see that the conclusions of Theorem 3.1 remain valid for this case, with the modification that the optimality condition now satisfied is a generalization of the F. John condition (see [Pol.7, Cla.1]).

Next, we turn to the special case where $\varphi(x,y) = (\bar{\sigma}[H(x,jy)])^2$ with $\bar{\sigma}$ the maximum singular value of an $m \times m$ transfer function matrix whose coefficients are differentiable in x (see [Pol.3]). We define the symmetric positive semi-definite matrix $Q(x,y) \triangleq H(x,jy)^* H(x,jy)$, with $*$ denoting the complex conjugate transpose. Noting that in (3.11) $\text{co}\{\nabla_x\varphi(x,y)\}_{y \in Y_\varepsilon(x)}$ is a superset of the Clarke generalized gradient, $\partial_x\varphi(x,y) = \text{co}\{\nabla_x\varphi(x,y)\}_{y \in Y_\varepsilon(x)}$, [Cla. 1], we proceed by analogy again.

Let $\varphi(x,y) \triangleq \lambda^1(x,y) \geq \lambda^2(x,y) \geq \cdots \geq \lambda^m(x,y)$ be the eigenvalues of $Q(x,y)$, let $\psi(x) \triangleq \max_{y \in Y} \varphi(x,y)$, as before, and let $Y_\varepsilon(x)$ be defined as in (3.10b). Next, we define

$$F_\varepsilon(x,y) \triangleq \begin{cases} \{\xi \in \mathbb{R}^n \mid \xi^i = \langle U_\varepsilon(x,y)z, \dfrac{\partial Q(x,y)}{\partial x^i} U_\varepsilon(x,y)z \rangle, \|z\| = 1\} & \text{if } \psi(x) \geq -\varepsilon \\ = \phi \text{ otherwise} \end{cases} \quad (3.12)$$

where $U_\varepsilon(x,y)$ is a unitary matrix whose columns span the eigenspace corresponding to the eigenvalues $\lambda^k(x,y)$, $k = 1,2,\ldots,k_\varepsilon(x,y)$ such that $\lambda^1(x,y) - \lambda^k(x,y) \leq \varepsilon$. Then we define

$$G_\varepsilon^{f,\psi}(x) \triangleq \text{co}\left\{\begin{bmatrix}\sqrt{\gamma\psi_+(x)}\\ \nabla f(x)\end{bmatrix}, \begin{bmatrix}0\\ \xi\end{bmatrix}\right\}_{\substack{\xi \in F_\varepsilon(x,y) \\ y \in Y_\varepsilon(x)}} \quad (3.13)$$

Substituting from (3.13) into (3.8b) we obtain a formula for

computing a search direction. This defines Algorithm 3.1 for constraints on $(\bar{\sigma}[H(x,jy)])^2$. The formulas for constraints on $\bar{\sigma}[H(x,jy)]$ are similar, but no longer symmetrical (i.e., one uses matrices of singular vectors $V_\epsilon(x,y)$ and $U_\epsilon(x,y)$ in (3.12)).

Again, referring to [Pol.3, Pol.8], we see that the conclusions of Theorem 3.1 remain valid with the modification that $\hat{x}$ satisfies an appropriate optimality condition in terms of generalized gradients (see [Cla. 1]). However, the computation of the search direction $\bar{h}_\epsilon(x)$ is no longer a finite quadratic program. Hence $\bar{h}_\epsilon(x)$ has to be computed by means of some nearest point algorithm such as the ones described in [Pol.3, Pol.7].

4. SOFTWARE FOR OPTIMIZATION-BASED CONTROL SYSTEM DESIGN

A software system, DELIGHT.MIMO (see [Pol.6]), implementing an optimization-based control system design methodology is currently being developed jointly by research teams at the University of California, Berkeley, and Imperial College, London. Some specific contributions to DELIGHT.MIMO are also being made at other institutions as well (see acknowledgement at the end of this paper).

DELIGHT.MIMO is a member of a family of optimization-based CAD packages currently being implemented in the DELIGHT system [Nye.1, Nye.2]. Hence a description of DELIGHT.MIMO must begin with a brief description of DELIGHT. DELIGHT can be thought of as a highly portable operating system for a FORTRAN or C machine. As can be expected from an operating system, it provides a certain number of commonly found features such as a text editor, a read and write files command, an ability to install and execute FORTRAN and C programs, a *help* command, a *history* command, a *repeat* command, hard interrupts, etc.

In addition, DELIGHT provides a number of rather special features. The most important of these are the following.

1) **Color graphics** for interaction with data and programs. The graphics provide a number of low level commands, such as

viewport, *window*, *vector*, *move*, *cursor* and *text*. These are used when display improvisation is necessary. In addition there are also a number of high level commands of the form *plot* data according to the options specified, which can be used to produce various orthodox type plots.

2) **High level language,** RATTLE, for the programming of optimization algorithms as well as information display options. RATTLE requires about 1/10 of the number of program lines compared to FORTRAN, but it executes considerably slower than FORTRAN. Because of this, in design packages a mixture of RATTLE and FORTRAN code is always used.

RATTLE compiles incrementally, it has binary matrix operation capability, it uses defines for command simplification and macros for producing simple RATTLE calls to complex FORTRAN programs (e.g. linprog $x = \mathit{argmin}\{<c,z> | Az = 0, Bz \geq 0, z \geq 0\}$. It is easy to use RATTLE to construct code for conversational data entry.

3) **Soft interrupts** for program debugging and temporary algorithm modification. Unlike hard interrupts which suspend a program the instant the break key is depressed, soft interrupts suspend a program only at designated break points in the program. When either a hard or a soft interrupt is executed, it is possible to *enter* suspended subprocedures and display and modify both local and global variables. After an interrupt the user may start up a totally unrelated computation or resume execution of the suspended program. To return to a suspended program after an unrelated side computation, the user executes the *reset* (a given number of interrupt levels) command.

4) **A modular, RATTLE code, optimization algorithm library** is being assembled. To use this library, the user assembles an algorithm from optional blocks, such as step size and direction finding procedures, via a menu. The problem to be solved must be described by means of several files containing either dimensional information or RATTLE code for: the cost function, ordinary

inequality constraints, functional inequality constraints, and gradients of the appropriate functions. The optimization problem and algorithm are linked by means of the *solve* command, e.g., *solve pid using polak_wardi*, when neither the problem pid nor algorithm polak_wardi has been compiled, or *solve pid* (or *solve using polak_wardi)* when the algorithm (problem) has been compiled earlier. Algorithms can be executed a desired number of iterations by means of the *run k* command, or they can be executed atomically, step by step, by means of the *step k* command. When execution of an optimization program is interrupted by means of a soft or hard interrupt, the user may adjust algorithm parameters, completely replace the algorithm, modify the problem description files, display variable values or plot response graphs.

DELIGHT.MIMO adds to the basic DELIGHT system a data base for control system interconnection description, programs for control system time and frequency response simulation, a symbolic differentiator for obtaining derivatives of these responses with respect to design parameters, interactive programs for initial design generation, an interactive program which assists the user in forming the RATTLE problem description files from design specifications, as required by the optimization algorithm library format, and both alpha-numeric and graphical means for entering the control system configuration. The optimization algorithm currently used for control system design is the Polak-Wardi method described in [Pol.3]; it has the form of the last algorithm described in Section 3.

5. THE DATA-BASE

The DELIGHT.MIMO data-base allows a system to be represented as an interconnection of subsystems. The subsystems may be either symbolic or state space representations. When the subsystems are represented symbolically, their names and interconnection data are stored in a *link table*. For the system in Fig. 3, where the block R generates the external system input, the link table consists of two blocks, as shown below:

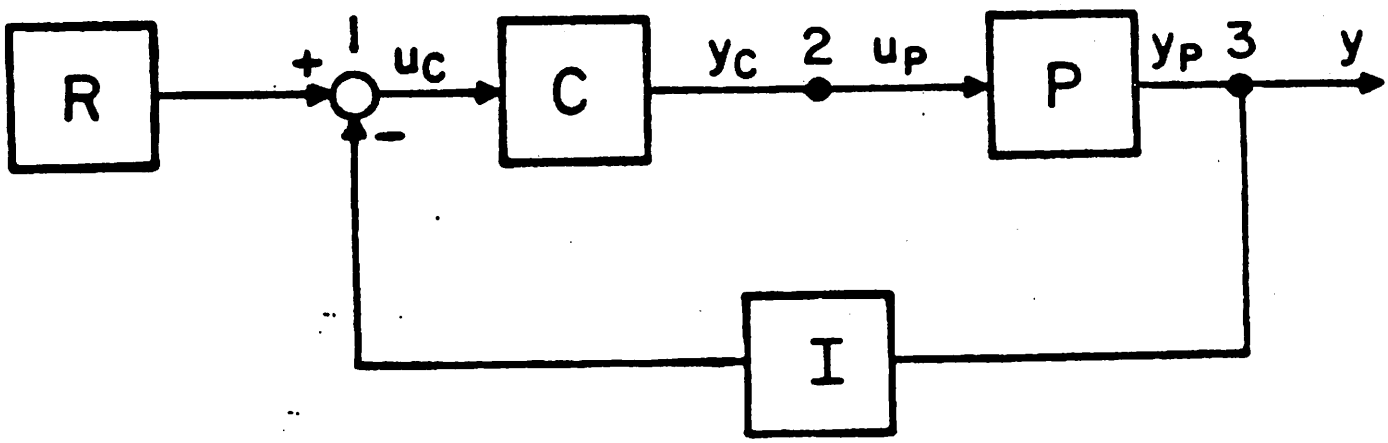

Fig. 3

Subsystem	"From" Node	"To" Node	Sign
P	2	3	+
C	1	2	+
I	3	1	-

Signal Generator	"To" Node
R	1

The link table can be constructed either alpha-numerically via the terminal keyboard or by means of the graphical block diagram editor.

When the subsystems are represented in state space form, as

$$S_i \begin{cases} \dot{z}_i = A_i z_i + B_i u_i \\ y_i = C_i z_i + D_i u_i \end{cases} \tag{5.1}$$

they define (assuming there are N subsystems) an *assembly of subsystems* S of the form

$$S \begin{cases} \dot{z} = Az + Bu \\ y = Cz + Du \end{cases} \tag{5.2}$$

where $A = diag(A_1, A_2, \ldots A_N)$, $B = diag(B_1, B_2, \ldots B_N)$, $C = diag(C_1, C_2, \ldots, C_N)$, $D = diag(D_1, D_2, \ldots, D_N)$. The interconnections between the subsystems are expressed algebraically:

$$u = Ey + Jr \tag{5.3}$$

where r is a vector of external inputs and E and J are matrices whose elements are zeros and ones. It should be clear that once the dimensions of the inputs and outputs are defined, the matrices E and J can be constructed from the data in the link table.

The matrices A_i, B_i, C_i, D_i, specifying the subsystems may be given either in numerical form or in parametric form. When given

in parametric form, their elements must be multinomials in the elements of the design parameter vector x. A symbolic differentiator is available for computing their derivatives with respect to the parameters.

The interconnection equation (5.3) can be eliminated by means of the *link* command which produces a reduced description of the form

$$S \begin{cases} \dot{z} = A_c z + B_c r \\ y = C_c z + D_c r \end{cases} \tag{5.4}$$

where $A_c = A+B[I-ED]^{-1}EC$, $B_c = B[I-ED]^{-1}J$, $C_c = C+D[I-ED]^{-1}EC$, and $D_c = D[I-ED]^{-1}J$, in terms of the matrices in (5.2). The link command can only be executed when specific values have been assigned to the design parameters.

In addition to the *link* command, three other commands are used in conjunction with the data-base. The first is the command which enables the user to *load* into the data-base numerical or parametrized descriptions of subsystems. The second is the *replace* command which associates subsystems in the data base for symbolically defined subsystems in the link table. The third is the *transfer* command, which can be used to transfer parametrized compensator descriptions and their initial values from a design initialization program.

6. COMPUTATION OF SYSTEM RESPONSES AND THEIR DERIVATIVES

Since the closed loop system (5.4) always has distinct eigenvalues (at least with probability 1), the computation of responses can be considerably simplified by diagonalization (more robust techniques, based on Schur decomposition, are also being contemplated). Thus, rewriting (5.2) with the design parameters made explicit, we get

$$\dot{z}(t,x) = A_c(x)z(t,x) + B_c(x)r(t) \tag{6.1a}$$

$$y(t,x) = C_c(x)z(t,x) + D_c(x)r(t) \tag{6.1b}$$

We begin with the time responses to inputs $r(t)$ which are

polynomials in t. With $W(x)$ a matrix of eigenvectors of $A(x)$, we obtain,

$$z(t,x) = W(x)e^{\Lambda(x)t}W(x)^{-1}z(0) + \int_0^t W(x)e^{\Lambda(x)(t-s)}W(x)^{-1}r(s)ds \qquad (6.2)$$

The output $y(t,x)$ is then computed according to (6.1b). Because the input $r(t)$ is a polynomial, the integral in (6.2) can be evaluated analytically (not numerically).

Next, the symbolic differentiator produces formulas for the derivatives of the components of the matrices A_c, B_c, C_c, D_c with respect to the components of the design parameter vector x. Numerical values for the derivatives are obtained by substituting current parameter values. We note that the derivatives with respect to x of $z(t,x)$ and $y(t,x)$ in (6.1) satisfy

$$(d/dt)(\partial z(t,x)/\partial x) = A_c(x)(\partial z(t,x)/\partial x) \qquad (6.3a)$$
$$+ (\partial A_c(x)/\partial x)z(t,x) + (\partial B_c(x)/\partial x)r(t)$$

$$\partial y(t,x)/\partial x = C_c(x)(\partial z(t,x)/\partial x) + (\partial C_c(x)/\partial x)z(t,x) + (\partial D_c(x)/\partial x)r(t) \qquad (6.3b)$$

The diagonalization matrix $W(x)$ can be used again to produce fairly simple formulas for the derivatives $(\partial z(t,x)/\partial x)$ and $(\partial y(t,x)/\partial x)$. Numerical substitution into these formulas yields efficient derivative evaluations.

Next we turn to the frequency response of the interconnected system. The input-output transfer function of the interconnected system is given by

$$G(jw,x) = C_c(x)[jwI - A_c(x)]^{-1}B_c(x) + D_c(x) \qquad (6.4a)$$

Since the derivative of G with respect to x is not a matrix, it is easiest to obtain componentwise expressions for it, viz.,

$$\partial G(jw,x)/\partial x = \partial C_c(x)/\partial x)[jwI - A_c(x)]^{-1}B_c(x) + D_c(x) \qquad (6.4b)$$
$$+ C_c(x)[jwI - A_c(x)]^{-1}(\partial A_c(x)/\partial x)[jwI - A_c(x)]^{-1}$$
$$+ C_c(x)[jwI - A_c(x)]^{-1}(\partial B_c(x)/\partial x) + (\partial D_c(x)/\partial x)$$

Assuming that the time response derivatives are computed first, the only major computation left in the evaluation of the frequency responses and their derivatives as specified by (6.4a), (6.4b) is the

evaluation of the matrix $[jwI-A_c(x)]^{-1}$. Since a diagonalization for $A_c(x)$ is already available, this computation can be considerably simplified by making use of the formula

$$[jwI - A_c(x)]^{-1} = W(x)[jwI - \Lambda(x)]^{-1}W(x) \qquad (6.5)$$

7. DESIGN INITIALIZATION TECHNIQUES

Design via optimization in not a totally automatic process. The designer is not only required to transcribe design specifications into semi-infinite inequalities, he or she is also required to decide on an initial compensator configuration as well as to produce a set of initial values for the compensator. This is a creative process which is very designer dependent. To facilitate the design initialization task, the DELIGHT.MIMO system will incorporate software implementing some of the more popular techniques, for example, such as those described in [Des.1, Doy.1, Mac.1, Moo.1, Ros.1, Saf.1, Saf.2, Ste. 1]. At the present time, there is software in DELIGHT.MIMO enabling design of compensators via LQG techniques as well as some model reduction algorithms. In the simplest case, these replace the observer dynamics with its DC gain matrix. It should be noted that in order to introduce into the design integrators for the elimination of steady state errors, a certain amount of ingenuity must be exercised in using LQG techniques. For example, consider the case in Fig.4. For the purpose of designing a state feedback matrix K, the external input r must be neglected, while the plant input is used as the feedback channel. Thus, suppose that the plant has dynamics given by

$$\dot{z}_P = A_P z_P + B_P u_P \qquad (7.1)$$

$$y_P = C_P z_P$$

Next, the integrator of the compensating block has dynamics

$$\dot{z}_C = u_C \qquad (7.2)$$

$$y_C = z_C$$

and the interconnection is specified by

$$u_C = -u_P \,. \qquad (7.3)$$

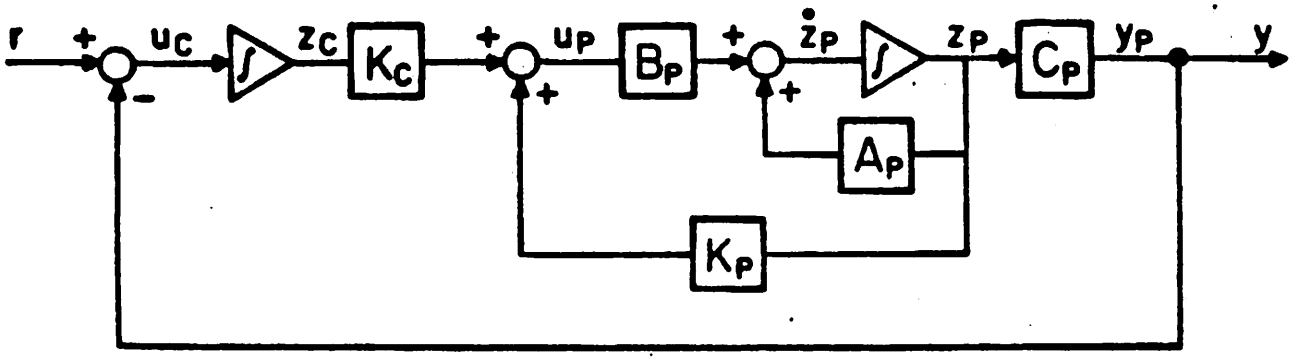

Fig. 4

Thus the assembly satisfies the state equation

$$\frac{d}{dt}\begin{bmatrix} z_P \\ z_C \end{bmatrix} = \begin{bmatrix} A_P & 0 \\ -C_P & 0 \end{bmatrix} + \begin{bmatrix} B_P \\ 0 \end{bmatrix} u_P \qquad (7.4)$$

LQR techniques can now be used to compute a state feedback matrix $K = [K_P \mid K_C]$ and the feedback law then becomes

$$u_P = K_P z_P + K_C z_C \qquad (7.5)$$

Since the state of the integrator block is available, an observer is needed only for estimating the plant state in this scheme.

8. CONCLUSION

We have seen that semi-infinite optimization is opening new possibilities in control system design. The numerical aspects of optimization-based control system design are still in an experimental stage, but with time, they should become part of standard design tools.

ACKNOWLEDGEMENT:

1. The DELIGHT.MIMO software was constructed by three teams:
At the University of California, Berkeley:
E. Polak (overall system specification and project coordination), **T.S. Wuu,** (optimization library, alphanumeric interaction tools), **P. Siegel** (Graphical block diagram editor specification), **T. Baker** (LQG design initialization tools), **W. T. Nye** (DELIGHT consultant).
At Imperial College, London:
D. Q. Mayne (overall system specification, initialization tools, symbolic differentiator and system response evaluator), **A. J. Heunis** (Symbolic differentiator and system response evaluator),
At the Lawrence Livermore National Laboratory:
C. J. Herget, D. Gavel, D. Tilly, S. Bly (graphical block diagram editor implementation).
Control system design subroutines were contributed by:
M. J. Denham, Kingston Polytechnic, and **A. J. Laub,** University of California, Santa Barbara.
Over the years, the DELIGHT.MIMO project was supported by The

National Science Foundation (ECS-7913148, ECS-8121149), The Joint Services Electronics Program (F49620-79-C-0178), The Office of Naval Research (N00014-83-K-0602), The Air Force Office of Scientific Research (AFOSR-83-0361), The Kirtland Air Force Weapons Laboratory, The Lawrence Livermore National Laboratory (I03403805, I03247705), The Semiconductor Research Consortium (SRC-82-11-008), and The Science and Engineering Research Council of Great Britain.

9. REFERENCES

[Ath.1] Athans, M., "The role and use of stochastic linear-quadratic-gaussian problem in control system design", *IEEE Trans.* vol. AC-16, no. 6, 1971.

[Bec.1] Becker, R. G., Heunis, A. J., and Mayne, D. Q., "Computer-Aided Design of Control Systems via Optimization," *Proc. IEE*, vol. 126, no. 6, 1979.

[Che.1] Chen, M. J. and Desoer, C. A., "Necessary and Sufficient Conditions for Robust Stability of Linear Distributed Feedback Systems", *Int. Journal on Control*, Vol. 35, No. 2, pp 255-267, 1982.

[Cla.1] Clarke, F. H., *Optimization and Nonsmooth Analysis*, Wiley-Interscience, New York, N.Y., 1983.

[Den.1] Denham, M. J., and Benson, C. J., "SLICE: a subroutine library for control system design," Internal Report 01/82, School of Electronic Engineering and Computer Science, Kingston Polytechnic, Kingston upon Thames KT1 2EE, 1982.

[Des.1] Desoer, C. A., and Gustafson C. L.,,"Algebraic Design of Two-Input Controllers for Linear Multivariable Feedback Systems", to appear *IEEE Transaction on Automatic Control.*

[Doy.1] [Doy.1] Doyle, J. C., and Stein, G. "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis", *IEEE Trans. on Control*, Vol. AC-26, No. 1, pp. 4-16, 1981.

[Gon.1] Gonzaga, C., Polak, E., and Trahan, R., "An Improved Algorithm for Optimization Problems with Functional Inequality Constraints", *IEEE Trans.*, Vol. AC-25, No. 1, 1980.

[Mac.1] MacFarlane, A. G. J., and Postlethwaite, I., "Generalizes Nyquist Stability Criterion and Multivariable Root Loci," *International Journal of Control*, vol. 25(1), 1977.

[Moo.1] Moore, B. C., "Principal component analysis in linear systems: controllability, observability and model reduction", *IEEE Trans., Vol. AC-26, No. 1 pp. 17-32, 1981.*

[Nye.1] Nye, W.T., Polak, E., Sangiovanni-Vincentelli, A., and Tits, A., "DELIGHT: an Optimization-Based Computer-Aided-Design System" Proc. IEEE Int. Symp. on Circuits and Systems, Chicago, Ill, April 24-27, 1981.

[Nye.2] Nye, W. T., "DELIGHT: An interactive system for optimization-based engineering design", Electronics Research Laboratory, University of California, Berkeley, Memo No. UCB/ERL M83/33, May 31, 1983.

[Pol.1] Polak, E., " Semi-infinite optimization in engineering design", in Lecture Notes in Economics and Mathematical Systems, Vol. 215: Semi-Infinite Programming and Applications, Edited by A. V. Fiacco and K. O. Kortanek, Springer-Verlag, Berlin, New York, Tokyo, 1983.

[Pol.2] Polak, E., and Mayne, D. Q., "An Algorithm for Optimization Problems with Functional Inequality Constraints," *IEEE Trans.*, vol. AC-21, no. 2, 1976.

[Pol.3] Polak, E., and Wardi, Y. Y., "A nondifferentiable optimization algorithm for the design of control systems subject to singular value inequalities over a frequency range", *Automatica*, Vol. 18, NO. 3, pp. 267-283, 1982.

[Pol.4] Polak, E., "A Modified Nyquist Stability Criterion for Use in Computer-Aided Design", ERL Memo No. M83/11, IEEE Trans. on Automatic Control, Vol. AC-28, No. 11, March 1984.

[Pol.5] Polak, E., Trahan, R., and Mayne, D. Q., "Combined phase I - phase II methods of feasible directions", *Math.*

Programming, Vol. 17, No. 1, 1979, pp 32-61.

[Pol.6] Polak, E., Siegel P., Wuu, T., Nye, W. T., and Mayne, D. Q., "DELIGHT-MIMO an interactive, optimization based multivariable control system design package", *IEEE Control Systems Magazine*, Vol.2, No.4, Dec. 1982, pp 9-14.

[Pol.7] Polak, E., *Computational Methods in optimization: A Unified Approach*, Academic Press, N.Y., 1971.

[Pol.8] Polak, E., and Mayne, D. Q., "Algorithm Models for Nondifferentiable Optimization," University of Calfornia, Electronics Research Laboratory Memo No. UCB/ERL M82/34, 10 May 1982.

[Ros.1] Rosenbrock, H. H., *Computer-Aided Control System Design*, Academic Press, London, 1974.

[Saf.1] Safonov, M. G., Laub, A. J., and Hartman, G. L., "Feedback Properties of Multivariable Systems: The Role and Use of the Return Difference Matrix," *IEEE Trans. on Control*, vol. AC-26, 1981.

[Saf.2] Safonov, M. G., "Choice of quadratic cost and noise matrices and the feedback properties of multiloop LQG regulators", *Proc. Asilomar Conf. on Circuits, Systems, and Computers*, Pacific Grove, California, 1979. Vol. 15, 1979.

[Ste.1] Stein, G., "Generalized quadratic weights for asymptotic regulator properties", *IEEE Trans.*, Vol. AC-24, 1979.