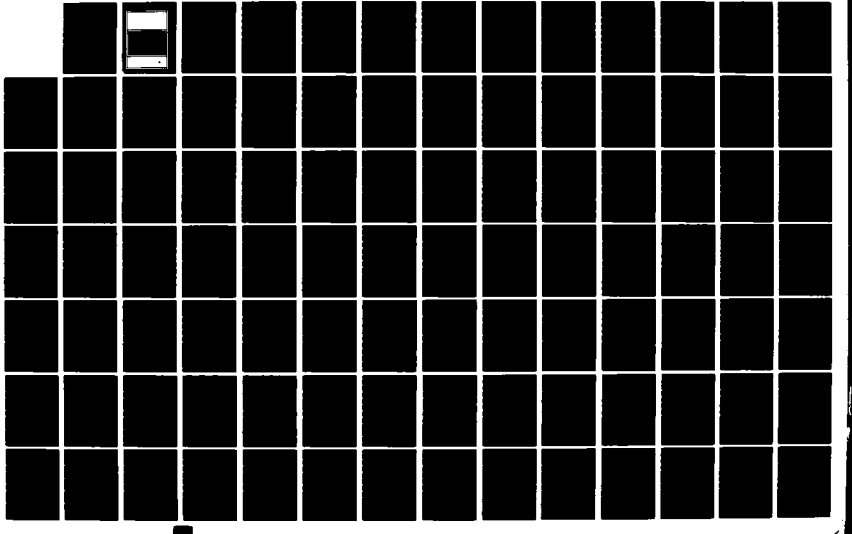
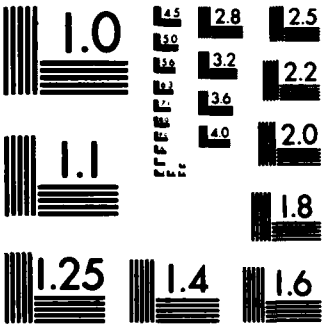
AD-A133 798 COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS(U) ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE) 1/2

UNCLASSIFIED JUL 83 AGARD-LS-128 F/G 17/7 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AGARD-LS-128

AGARD

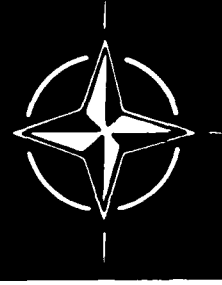
ADVISORY GROUP FOR AEROSPACE RESEARCH & DEVELOPMENT

7 RUE ANCELLE 92200 NEUILLY SUR SEINE FRANCE

AGARD LECTURE SERIES No. 128

Computer-Aided Design and Analysis of Digital Guidance and Control Systems

NORTH ATLANTIC TREATY ORGANIZATION



DISTRIBUTION AND AVAILABILITY
ON BACK COVER

AGARD-LS-128

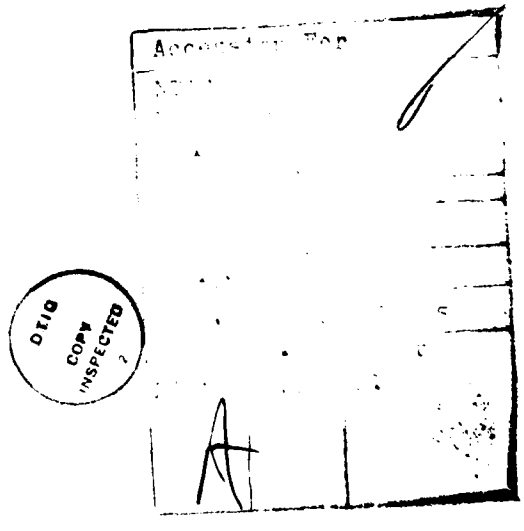
NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARD Lecture Series No.128

COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS



The material in this publication was assembled to support a Lecture Series under the sponsorship of the Guidance and Control Panel and the Consultant and Exchange Programme of AGARD on 5–6 September 1983 in Stuttgart, Germany; 8–9 September 1983 in Athens, Greece; and 12–13 September 1983 in Paris, France.

THE MISSION OF AGARD

The mission of AGARD is to bring together the leading personalities of the NATO nations in the fields of science and technology relating to aerospace for the following purposes:

- Exchanging of scientific and technical information;
- Continuously stimulating advances in the aerospace sciences relevant to strengthening the common defence posture;
- Improving the co-operation among member nations in aerospace research and development;
- Providing scientific and technical advice and assistance to the North Atlantic Military Committee in the field of aerospace research and development;
- Rendering scientific and technical assistance, as requested, to other NATO bodies and to member nations in connection with research and development problems in the aerospace field;
- Providing assistance to member nations for the purpose of increasing their scientific and technical potential;
- Recommending effective ways for the member nations to use their research and development capabilities for the common benefit of the NATO community.

The highest authority within AGARD is the National Delegates Board consisting of officially appointed senior representatives from each member nation. The mission of AGARD is carried out through the Panels which are composed of experts appointed by the National Delegates, the Consultant and Exchange Programme and the Aerospace Applications Studies Programme. The results of AGARD work are reported to the member nations and the NATO Authorities through the AGARD series of publications of which this is one.

Participation in AGARD activities is by invitation only and is normally limited to citizens of the NATO nations.

Published July 1983

ISBN 92-835-1455-6

Printed by Specialised Printing Services Limited
40 Chigwell Lane, Loughton, Essex IG10 3TZ

PREFACE

This Lecture Series is intended to provide the basic concepts, theories and computer methods involved in the design of advanced guidance and control systems and is sponsored by the Guidance and Control Panel.

The degree of advantages in the application of modern microprocessor technologies is already largely affected by the way corresponding systems are designed in the very early stage of a development programme.

Is intended to perform a comprehensive review of direct digital analysis and synthesis procedures and to include in this Lecture Series computer-aided and graphical techniques that can be employed in preliminary design, synthesis and real-time simulation.

This Lecture Series has been implemented by the Consultant and Exchange Programme.

LIST OF SPEAKERS

Lecture Series Director: Dr J.E.Wall
Systems and Research Center
Honeywell Inc.
2600 Ridgeway Parkway
Minneapolis, Minnesota 55413
USA

SPEAKERS

Professor K.J.Aström
Department of Automatic Control
Lund Institute of Technology
Box 725
S-220 07 Lund 7
Sweden

Professor G.F.Franklin
Department of Electrical Engineering
Stanford University
131 Durand Building
Stanford, California 94305
USA

Dr M.J.Denham
Kingston Polytechnic
School of Electronic Engineering
and Computer Science
Penrhyn Road
Kingston upon Thames KT1 2EE
Surrey
UK

Dr G.Grübel
D.F.V.L.R.
Institute for Flight System Dynamics
8031 Oberpfaffenhofen
Germany

Professor A.J.Laub
Department of Electrical Engineering
Systems
University of Southern California
Los Angeles, California 90089
USA

CONTENTS

*Paper not available at time of printing

FUNDAMENTALS OF ANALYSIS FOR DIGITAL CONTROL SYSTEMS

G. F. Franklin
Department of Electrical Engineering
Stanford University
Stanford, California 94305 USA

2.1 SUMMARY

The intent of this presentation is to provide the theoretical background and practical tools for the design of a control system which is to be implemented using a computer or microprocessor. The methods to be studied are primarily for closed-loop (feedback) systems in which the dynamic response of the process being controlled is a major consideration in the design. The design methods are applicable to any type of computer (from microprocessors to large scale computers); however, the effects of small word size and slow sample rates take on a more important role when using microprocessors.

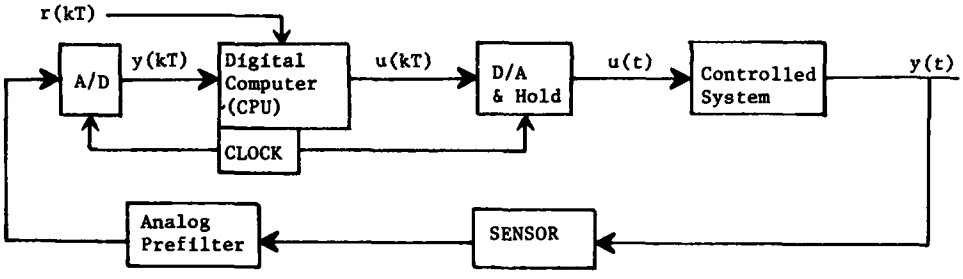
It will be assumed in the presentation that the reader has some knowledge of control system design methods for continuous (or analog) systems such as those covered in the textbooks by Dorf [1] or Ogata [2]. Furthermore, a more complete reference for the subject material can be found in a digital control textbook by Franklin and Powell [3].

A typical topology of the type of system to be considered is shown in Fig. 2.1. There are two fundamentally different methods for the design of digital algorithms:

(1) Continuous Design and digitization: perform a continuous design, then digitize the resulting compensation,

(2) Direct Digital Design: digitize the plant model, then perform a design using discrete analysis methods.

Both methods will be covered and their advantages and disadvantages discussed.



r = reference or command input
y = output quantities
u = actuator input signals
A/D = analog-to-digital converter
D/A = digital-to-analog converter

Figure 2.1: Basic Control System Block Diagram

2.2 THEORETICAL BACKGROUND

(a) z-Transform

In the analysis of continuous systems, we use the Laplace Transform which is defined by:

$$L\{f(t)\} = F(s) = \int_0^\infty f(t)e^{-st}\,dt \tag{2.1}$$

which leads directly to the important property that

$$L\{\dot{f}(t)\} = sF(s) \quad . \tag{2.2}$$

This relation enables us to easily find the transfer function of a linear continuous system given the differential equations of that system.

For discrete systems, a very similar procedure is available. The "z-transform" is defined by

$$Z\{f(n)\} = F(z) = \sum_{n=-\infty}^{\infty} f(n)z^{-n} \tag{2.3}$$

which also leads directly to a property analogous to Eq. (2.2), specifically that

This relation allows us to easily find the transfer function of a discrete system given the difference equations of that system. For example, the general 2nd order difference equation,

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_0 u(n) + b_1 u(n-1) + b_2 u(n-2) \tag{2.5}$$

can be converted from y(n), u(n), etc., to the z-transform of those variables by invoking Eq. (2.4) once or twice to arrive at,

$$Y(z) = (-a_1 z^{-1} - a_2 z^{-2}) Y(z) + (b_0 + b_1 z^{-1} + b_2 z^{-2}) U(z) \tag{2.6}$$

which results in the transfer function

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.7}$$

(b) z-Transform Inversion

A table relating a few simple discrete time functions to their z-transform is contained in Table 2.1 along with the Laplace transform for the same time functions.

Number	$\mathcal{F}(s)$	f(nT)	F(z)
1	-	1, n=k; 0, n≠k	z^{-k}
2	$\frac{1}{s}$	1(nT)	$\frac{z}{z-1}$
3	$\frac{1}{s^2}$	nT	$\frac{Tz}{(z-1)^2}$
4	$\frac{1}{s+a}$	e^{-anT}	$\frac{z}{z-e^{-aT}}$
5	$\frac{a}{s(s+a)}$	$1-e^{-anT}$	$\frac{z(1-e^{-aT})}{(z-1))z-e^{-aT})}$
6	$\frac{s+a}{(s+a)^2+b^2}$	$e^{-anT}\cos bnT$	$\frac{z(z-e^{-aT}\cos bT)}{z^2-2e^{-aT}(\cos bT)z+e^{-2aT}}$

$\mathcal{F}(s)$ is the Laplace transform of f(t) and F(z) is the z-transform of f(nT). Unless otherwise noted, f(t) = 0, t < 0.

Table 2.1: z-Transforms

Given a general z-transform, one can expand it into a sum of elementary terms using partial fraction expansion and find the resulting time series from the table. Again, these procedures are exactly the same as those used for continuous systems.

A z-transform inversion technique which has no continuous counterpart is long division. Given a z-transform,

$$Y(z) = \frac{N(z)}{D(z)} \tag{2.8}$$

one simply divides the denominator into the numerator using long division. The result is a series (perhaps infinite) in powers of z^{-1}, from which the time series can be found by using Eq. (2.3).

For example, a first order system described by the difference equations,

$$y(n) = ky(n-1) + u(n) \tag{2.9}$$

yields

$$\frac{Y(z)}{U(z)} = \frac{1}{1 - kz^{-1}} \tag{2.10}$$

for an impulsive input,

$$u(0) = 1$$

$$u(n) = 0 \qquad n \neq 0$$

$$\Rightarrow \quad U(z) = 1$$

and

$$Y(z) = \frac{1}{1 - kz^{-1}} \tag{2.11}$$

Therefore, to find the time series, use long division methods as follows:

$$\begin{array}{r} 1+kz^{-1}+k^2z^{-2}+k^3z^{-3}+\ \ldots \\ 1-kz^{-1}\ \overline{)\,1} \\ \underline{1-kz^{-1}} \\ kz^{-1} \\ \underline{kz^{-1}-k^2z^{-2}} \\ k^2z^{-2}+0 \\ \underline{k^2z^{-2}-k^3z^{-3}} \\ k^3z^{-3} \\ \vdots \end{array}$$

to yield the infinite series:

$$1 + kz^{-1} + k^2z^{-2} + k^{-3}z^{-3} + \ldots$$

The quotient, $1 + kz^{-1} + k^2z^{-2} + k^3z^{-3} + \ldots$ is $Y(z)$ which means that,

$$\begin{aligned} y(0) &= 1 \\ y(1) &= k \\ y(2) &= k^2 \\ &\vdots \\ y(n) &= k^n \end{aligned}$$

(c) Relationship Between s and z

For continuous systems, one often associates certain behavior for different pole locations in the s-plane: oscillatory behavior for poles near the imaginary axis, exponential decay for poles on the negative real axis, and unstable behavior for poles with a positive real part. The same kind of association is also useful to designers of discrete systems. The equivalent characteristics in the z-plane are related to those in the s-plane by the expression

$$z = e^{sT} \tag{2.12}$$

where T = sample period. This is obtained by comparing the z-transform of the sampled version of a signal with the Laplace transform of the signal itself. The z-transform Table 2.1 also includes the Laplace transforms, which demonstrates the $z = e^{sT}$ relationship in the denominators of all the table entries.

Figure 2.2 shows the mapping of lines of constant damping, ζ, and natural frequency, ω_n, from the s-plane to the upper half of the z-plane using Eq. (2.12). The mapping has several important features:

(1) The stability boundary is the unit circle, $|z| = 1$.

(2) The small vicinity around $z = +1$ is essentially identical to the vicinity around $s = 0$.

(3) z-plane locations give response information normalized to the sample rate, rather than with respect to time as in the s-plane.

(4) The negative real z axis always represents a frequency of $\omega_s/2$, where ω_s = sample rate.

(5) Vertical lines in the left-hand s-plane (constant real part or time constant) map into circles within the unit circle.

(6) Horizontal lines in the s-plane (constant imaginary part or frequency) map into radial lines in the z-plane.

(7) Every location in the z-plane represents many frequencies separated by ω_s. Physically, this is because many different signals can have the same samples and mathematically because of the nature of the exponential function in Eq. (2.12).

(d) Final Value Theorem

The final value theorem for continuous systems

$$\lim_{t\to\infty} x(t) = \lim_{s\to 0} sX(s) \tag{2.13}$$

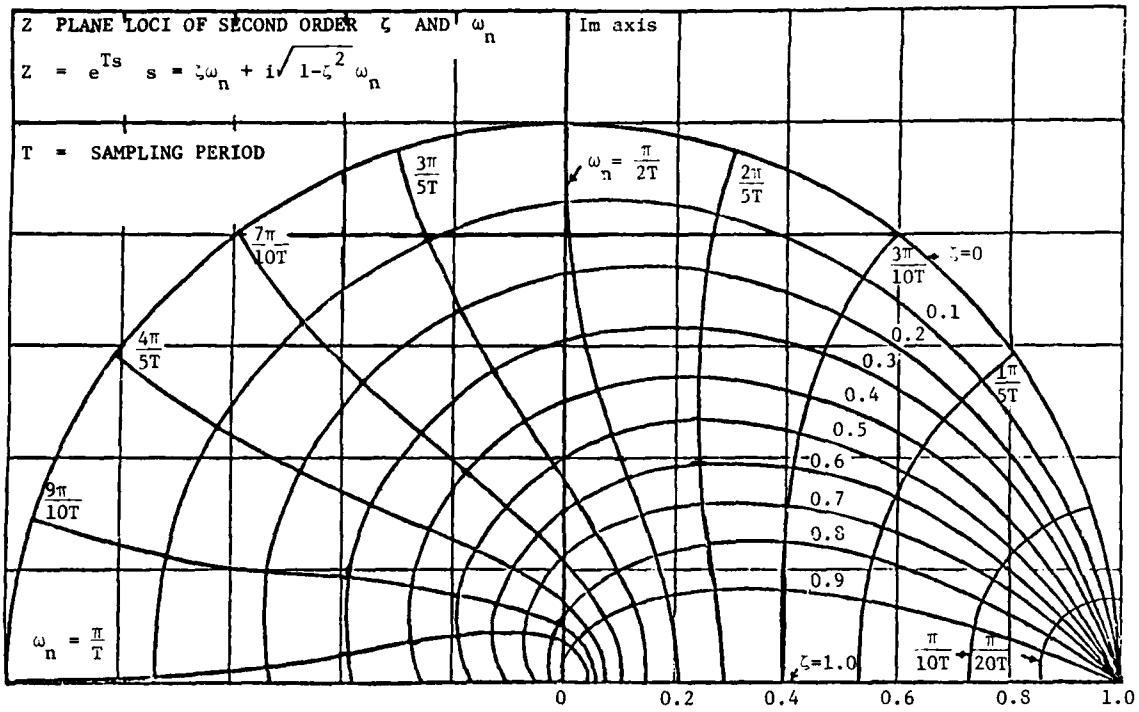


Figure 2.2: Natural Frequency and Damping Loci in z-Plane

is often used to find steady state system errors and/or steady state gains of portions of a control system. The analog for discrete systems is obtained by noting that a continuous constant steady response has the transform $X(s) = A/s$ and leads to the multiplication by s in Eq. (2.13). Therefore, since a constant steady response for discrete systems is $X(z) = A/1-z^{-1}$, the discrete final value theorem is:

$$\lim_{n\to\infty} x(n) = \lim_{z\to 1} (1 - z^{-1})X(z) \qquad (2.14)$$

For example, to find the DC gain of the transfer function, $G(z) = \frac{X(z)}{U(z)} = \frac{.58(1+z)}{z + .16}$, let $u(n) = 1$ for $n \geq 0$, so that

$$U(z) = \frac{1}{1 - z^{-1}}$$

and

$$X(z) = \frac{.58(1+z)}{(1-z^{-1})(z+.16)}$$

Applying the final value theorem yields

$$x(\infty) = \lim_{z\to 1} \left[\frac{.58(1 + z)}{z + .16}\right] = 1$$

and therefore, the DC gain of $G(z)$ is unity. In general, we see that to find the DC gain of any transfer function, simply substitute $z = 1$ and compute the resulting gain.

Since the gain of a system does not change whether represented continuously or discretely, this calculation is an excellent check on the calculations associated with determining the discrete model of a system.

2.3 CONTINUOUS DESIGN

The first part of this design procedure should be already familiar to the reader; that is, the design of feedback control compensation for a continuous system. This design is carried out as if the system were continuous and no changes are required to represent the fact that the control will eventually be implemented digitally.

(a) Digitization Procedures

The second part of the procedure is to digitize the resulting compensation. Therefore, the problem to be addressed is: given a $D(s)$, find the best equivalent $D(z)$. Or more exactly, given a $D(s)$ from the control system shown in Fig. 2.3, find the best digital implementation of that compensation. A digital implementation requires that y is sampled at some sample rate and that the computer output samples are smoothed in some manner so as to provide a continuous u. For ease of hardware design, the smoothing operation is almost always a simple hold (or zero order hold, "ZOH") which is shown in Fig. 2.4

Therefore, we can restate the problem as: Find the best $D(z)$ in the digital implementation shown in Fig. 2.5 to match a desired $D(s)$. It is important to note at the outset that there is no exact solution

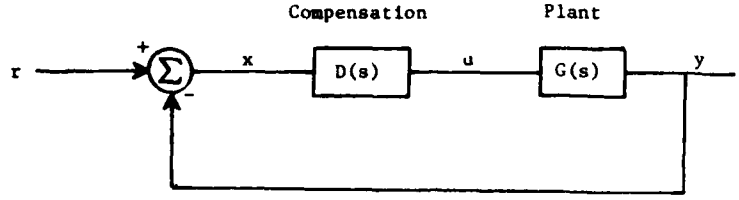


Figure 2.3: Continuous Control System

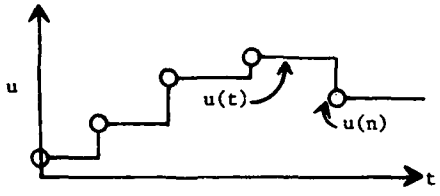


Figure 2.4: Zero Order Hold

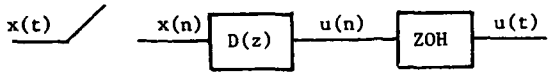


Figure 2.5: Digital Compensation Implementation

to this problem because D(s) responds to the complete time history of x(t) whereas D(z) only has access to the samples, x(n). In a sense, the various digitization approximations (and approximations they are) simply make different assumptions about what happens to x(t) between the sample points.

Tustin's Method: One digitization method is to approach the problem as one of numerical integration. Suppose

$$\frac{U(s)}{X(s)} = D(s) = \frac{1}{s} \quad , \text{ i.e., pure integration}$$

Therefore,

$$u(nT) = \int_0^{nT-T} x(t)\,dt + \int_{nT-T}^{T} x(t)\,dt \qquad (2.15)$$

$$= u(nT-T) + (\text{area under } x(t) \text{ over last } T)$$

where T = sample period, u(nT) is usually written u(n) for short and the task at each step is to use trapezoidal integration, i.e., approximate x(t) by a straight line between the two samples (Fig. 2.6). Therefore Eq. (2.15) becomes:

$$u(nT) = u(nT-T) + \frac{T}{2}\left[x(nT-T) + x(nT)\right] \qquad (2.16)$$

or taking the z-transform,

$$\frac{U(z)}{X(z)} = \frac{T}{2}\,\frac{1+z^{-1}}{1-z^{-1}} = \frac{1}{\frac{2}{T}\,\frac{1-z^{-1}}{1+z^{-1}}} \qquad (2.17)$$

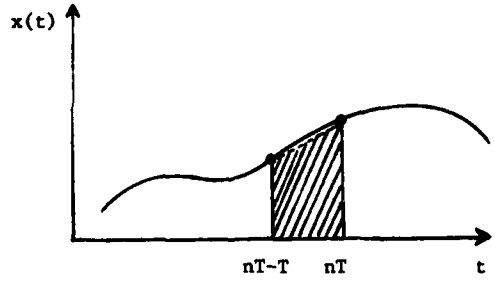


Figure 2.6: Trapezoidal Integration

For

$$D(s) = \frac{a}{s+a}$$

Application of the same integration approximation yields

$$D(z) = \frac{a}{\frac{2}{T}\,\frac{1-z^{-1}}{1+z^{-1}} + a}$$

and, in fact, the substitution

$$s := \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{2.18}$$

In any $D(s)$ yields a $D(z)$ based on the trapezoidal integration formula. This is called Tustin's or the Bilinear approximation.

Matched Pole Zero Method (MPZ): Another digitization method, called the "matched pole-zero" is found by extrapolation of the relation between the s and z planes stated in Eq. (2.12). If we take the z-transform of a sampled $x(t)$, then the poles of $X(z)$ are related to the poles of $X(s)$ according to $z = e^{sT}$. However, we must go through the z-transform process to locate the zeros of $X(z)$. The idea of the matched pole-zero technique is to apply $z = e^{sT}$ to the poles and zeros of a transfer function. Since physical systems often have more poles than zeros, it is also useful to arbitrarily add zeros of $D(z)$ at $z = -1$ (i.e., a $(1 + z^{-1})$ term) which causes an averaging of the current and past input values as in the trapezoidal integration (Tustin's) method. The gain is selected so that the low frequency gain of $D(s)$ and $D(z)$ match one another. The method summarized is:

(1) Map poles and zeros according to $z = e^{sT}$.

(2) Add $(1 + z^{-1})$ or $(1 + z^{-1})^2$, etc., if numerator is lower order than the denominator.

(3) Match DC or low frequency gain.

For example, the matched pole-zero approximation of

$$D(s) = \frac{s + a}{s + b} \quad \text{is} \quad D(z) = k\frac{z - e^{-aT}}{z - e^{bT}} \tag{2.19}$$

where $k = \frac{a}{b}\frac{1-e^{-bT}}{1-e^{-aT}}$ and for

$$D(s) = \frac{s+a}{s(s+b)} = D(z) = k\frac{(z+1)(z-e^{-aT})}{(z-1)(z-e^{-bT})} \tag{2.20}$$

where $k = \frac{a}{2b}\frac{1-e^{-bT}}{1-e^{-aT}}$.

In both digitization methods, the fact that an equal power of z appears in numerator and denominator of $D(z)$ implies that the difference equation at time n will require a sample of the input at time n. For example, the $D(z)$ in Eq. (2.19) can be written

$$\frac{U(z)}{X(z)} = D(z) = k\frac{1-\alpha z^{-1}}{1-\beta z^{-1}}$$

which results in the difference equation,

$$u(n) = \beta u(n-1) + k[x(n) - \alpha x(n-L)] . \tag{2.21}$$

Modified Matched Pole Zero Method (MMPZ): The $D(z)$ in Eq. (2.20) would also result in $u(n)$ being dependent on $x(n)$, the input at the same time point. If the structure of the computer hardware prohibits this relation, or if the computations are particularly lengthy thus rendering Eq. (2.21) impossible to implement, it may be desirable to arrive at a $D(z)$ which has one less power of z in the numerator than denominator and hence the computer output, $u(n)$, only requires input from the previous time, i.e. $x(n-1)$. To do this, we modify step (2) in the "matched pole-zero" procedure to add one less zero at $z = -1$, leaving a single zero at $z = \infty$. The second example

$$D(s) = \frac{s + a}{s(s+b)}$$

and then become

$$D(z) = k\frac{z - e^{-aT}}{(z-1)(z-e^{-bT})} \quad \text{where} \quad k = \frac{a}{b}\frac{1-e^{-bT}}{1-e^{-aT}}$$

and which results in

$$u(n) = (1+e^{-bT})u(n-1) - e^{-bT}u(n-2) + k[x(n-1) - e^{-aT}x(n-2)] .$$

Method Comparison: A numerical comparison of a magnitude frequency response is made in Fig. 2.7 for the three approximation techniques at two sample rates. The results of the $D(z)$ computations used in arriving at Fig. 2.7 are shown in Table 2.2.

The figure shows that all the approximations are quite good at frequencies below about 1/4 the sample rate, $\omega_s/4$. If $\omega_s/4$ is sufficiently larger than the filter break frequency, i.e., if the sampling is fast enough, the break characteristics are accurately reproduced. Tustin's and the MPZ show a notch at

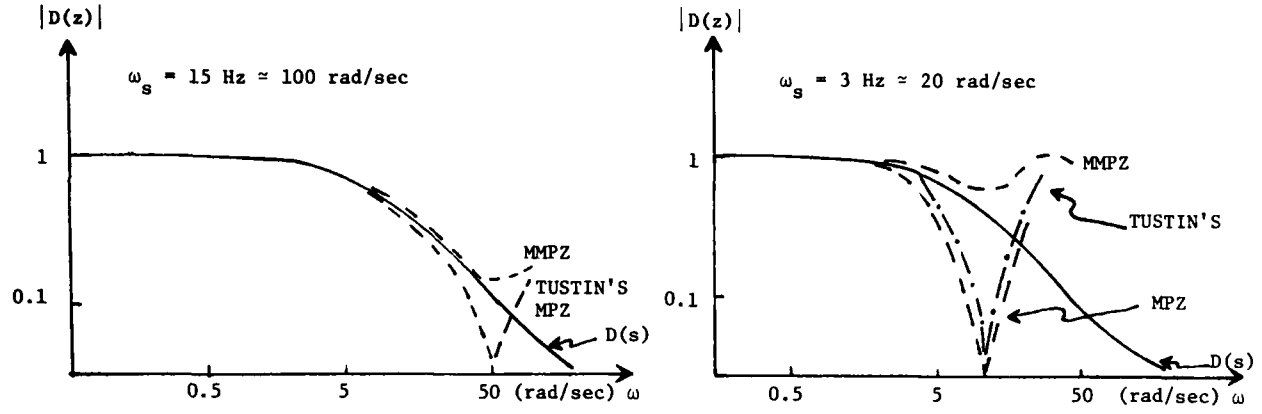


Figure 2.7: Comparison of Discrete Approximations

	D(z) for $D(s) = \frac{5}{s+5}$	
	ω_s = 15 Hz	ω_s = 3 Hz
Matched Pole-Zero (MPZ)	$.143 \frac{z+1}{z-.715}$	$.405 \frac{z+1}{z-.189}$
Modified MPZ (MMPZ)	$.285 \frac{1}{z-.715}$	$.811 \frac{1}{z-.189}$
Tustin's	$.143 \frac{z+1}{z-.713}$	$.454 \frac{z+1}{z-.0914}$

Table 2.2: Digital Approximations

$\omega_s/2$ due to their zero term, (z+1). Other than the large difference at $\omega_s/2$ which is typically outside the range of interest, the three methods have similar accuracies. Since the MPZ techniques require much simpler algebra than Tustin's, they are typically preferred, although it should be pointed out that while the MPZ method requires knowledge of the pole and zero locations, Tustin's method does not require that the polynomials be factored.

(b) Design Example

For a $1/s^2$ plant, we wish to design a digital controller to have a closed-loop natural frequency, ω_n, of ~0.3 rad/sec and $\zeta = 0.7$. The first step is to find the proper D(s) defined in Fig. 2.8. The specifications can be met with

$$D(s) = k \frac{s+a}{s+b} \qquad (2.22)$$

where

$$a = 0.2$$
$$b = 2.0$$
$$k = 0.81$$

as can be verified by the root locus in Fig. 2.9. To digitize this D(s), we first need to select a sample rate. For a system with $\omega_n = 0.3$ rad/sec, a very "safe" sample rate would be a factor of 20 faster than ω_n, yielding

$$\omega_s = 0.3 \times 20 = 6 \text{ rad/sec} .$$

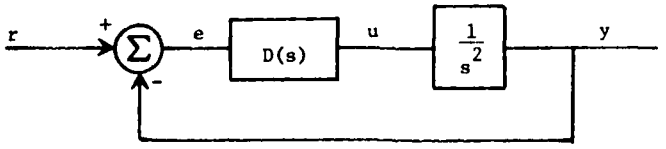


Figure 2.8: Continuous Design Statement

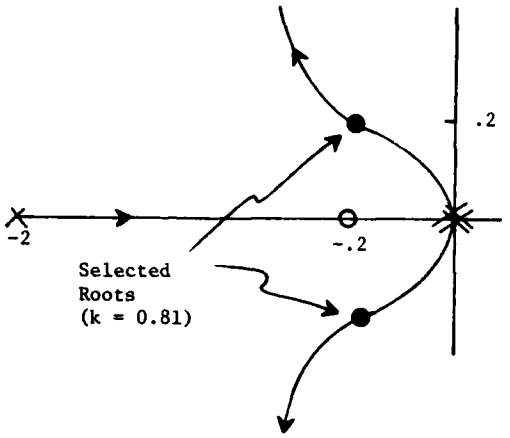


Figure 2.9: s-Plane Locus vs. k

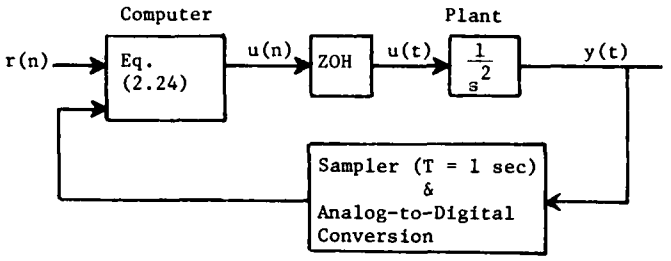


Figure 2.10: Digital Control System

Thus, let us pick $T = 1$ sec. The matched Pole-Zero digitization of Eq. (2.22) is given by Eq. (2.19) and yields

$$D(z) = (0.389)\frac{z - 0.82}{z - 0.135} \tag{2.23}$$

or

$$D(z) = \frac{0.389 - 0.319\, z^{-1}}{1 - 0.135\, z^{-1}}$$

which leads to

$$u(n) = 0.135u(n-1) + 0.389e(n) - 0.319e(n-1) \tag{2.24}$$

where

$$e(n) = r(n) - y(n)$$

and completes the digital algorithm design. The complete digital system is shown in Fig. 2.10.

(c) Applicability Limits of Method

If an exact discrete analysis or a simulation of the system was performed and the digitization was determined for a wide range of rates, the system would be unstable for sample rates slower than approximately $5 \times \omega_n$ and the damping would be substantially degraded for sample rates slower than $10 \times \omega_n$. At sample rates on the order of $20 \times \omega_n$ (or $20 \times$ bandwidth for more complex systems), this design method can be used with confidence.

Basically, the errors come about because the technique ignores the lagging effect of the ZOH. An approximate method to account for this is to assume that the transfer function of the ZOH is

$$G_{ZOH}(s) = \frac{2/T}{s + 2/T} \tag{2.25}$$

This is based on the idea that, on the average, the hold delays by $T/2$ and the above is a first order lag with a time constant of $T/2$, DC gain = 1. We could therefore patch-up the original $D(s)$ design by inserting this $G_{ZOH}(s)$ in the original plant model and finding the $D(s)$ that yields satisfactory response.

One of the advantages of using this design method, however, is that the sample rate need not be selected until after the basic feedback design is completed. Therefore the patch-up eliminates this advantage, although it does partially alleviate the approximate nature of the method, which is the primary disadvantage.

2.4 DISCRETE DESIGN

(a) Analysis Tools

The first step in performing a control design or analysis of a system with some discrete elements in it is to find the discrete transfer function of the continuous portion. For a system similar to that shown in Fig. 2.1, we wish to find the transfer function between $u(n)$ and $y(n)$. Unlike the previous section, there is an exact discrete equivalent for this system because the ZOH precisely describes what happens between samples and the output, $y(n)$, is only dependent on the input at the sample times, $u(n)$.

For a plant described by a $G(s)$ and preceded by a ZOH, the discrete transfer function is:

$$G(z) = (1 - z^{-1})Z\left\{\frac{G(s)}{s}\right\} \tag{2.26}$$

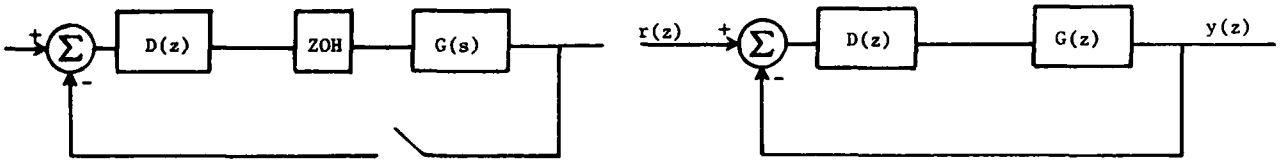


Figure 2.11:

(a) Mixed Control System (b) Pure Discrete Equivalent

where $Z\,F(s)$ means the z-transform of the time series whose Laplace transform is $F(s)$, i.e., the same line in Table 2.1. The formula has the term $G(s)/s$ because the control comes in as a step input during each sample period. The term $(1-z^{-1})$ is there because a one-sample duration step can be thought of as an infinite duration step followed by a negative step one cycle delayed. This formula (Eq. (2.26)) allows us to replace the mixed (continuous and discrete) system shown in Fig. 2.11a with the pure discrete equivalent system shown in Fig. 2.11b.

The analysis and design of discrete systems is very similar to continuous ones; in fact, all the same rules apply. The closed-loop transfer function of Fig. 2.11b is obtained using the same rules of block diagram reduction, i.e.,

$$\frac{Y(z)}{R(z)} = \frac{DG}{1 + DG} \tag{2.27}$$

Since we would like to find the characteristic behavior of the closed-loop system, we wish to find the factors of the denominator of Eq. (2.27), i.e., find the roots of the characteristic equation:

$$1 + D(z)G(z) = 0 \tag{2.28}$$

The root locus techniques used in continuous systems to find roots of a polynomial in s apply equally well here for the polynomial in z. The rules apply directly without modification; however, the interpretation of the results is quite different as we saw in Fig. 2.2. A major difference is that the stability boundary is now the unit circle instea of the imaginary axis.

A simple example of the discrete design tools discussed so far should help fix ideas. Suppose $G(s)$ in Fig. 2.11a is:

$$G(s) = \frac{a}{s + a}$$

It follows from Eq. (2.26) that

$$\begin{aligned} G(z) &= (1 - z^{-1})Z\left[\frac{a}{s(s+a)}\right] \\ &= (1 - z^{-1})\left[\frac{(1-e^{-aT})z^{-1}}{(1-z^{-1})(1-e^{-aT}z^{-1})}\right] \\ G(z) &= \frac{(1 - \alpha)}{z - \alpha} \end{aligned} \tag{2.29}$$

where

$$\alpha = e^{-aT} \quad .$$

To analyze the performance of a closed-loop proportional control law, i.e., $D(z) = k$, we use standard root locus rules. The result is shown in Fig. 2.12a and for comparison, the root locus for a continuous controller is shown in Fig. 2.12b. In contrast to the continuous case which remains stable for all values of k, the discrete case becomes oscillatory with a decreasing damping ratio as z goes from 0 to -1 and eventually becomes unstable. This instability is due to the lagging effect of the ZOH which is properly accounted for in the discrete analysis.

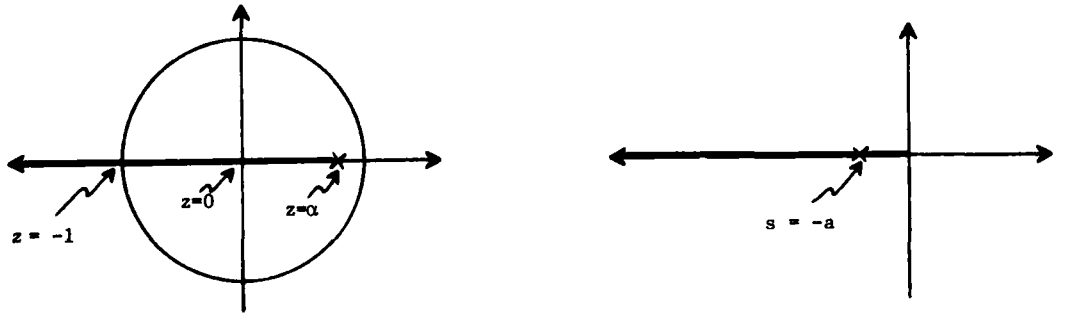


Figure 2:12

(a) z-Plane Root Locus (b) s-Plane Root Locus

(b) Feedback Properties

In continuous systems, we typically start the design process by using proportional, derivative, or integral control laws or combinations of these, sometimes with a lag included. The same ideas are used in discrete designs directly, or perhaps the $D(z)$ that results from the digitization of a continuously designed $D(s)$ is used as a starting point.

The discrete control laws are:

Proportional:

$$\begin{aligned} u(n) &= k_p e(n) \\ \Rightarrow \quad D(z) &= k_p \end{aligned} \tag{2.30}$$

Derivative:

$$\begin{aligned} u(n) &= k_D[e(n) - e(n-1)] \\ \Rightarrow \quad D(n) &= k_D(1-z^{-1}) = k_D \frac{z-1}{z} \end{aligned} \tag{2.31}$$

Integral:

$$\begin{aligned} u(n) &= u(n-1) + k_I e(n) \\ \Rightarrow \quad D(z) &= \frac{k_I}{1-z^{-1}} = \frac{k_I z}{z-1} \end{aligned} \tag{2.32}$$

(c) Design Example

For an example, let us use the same problem as we used for the continuous design; the $1/s^2$ plant. Using Eq. (2.26), we have

$$G(z) = \frac{T^2}{2} \frac{z+1}{(z-1)^2} \tag{2.33}$$

which becomes with $T = 1$ sec,

$$G(Z) = \frac{1}{2} \frac{z+1}{(z-1)^2} \tag{2.34}$$

Proportional feedback in the continuous case yields pure oscillatory motion and in the discrete case, we should expect even worse results. The root locus in Fig. 2.13 verifies this. For very low values of k (very low frequencies compared to the sample rate) the locus is tangent to the unit circle ($\zeta \simeq 0$ and pure oscillatory motion) thus matching the proportional continuous design. For higher values of k, the locus diverges into the unstable region due to the effect of the ZOH and sampling.

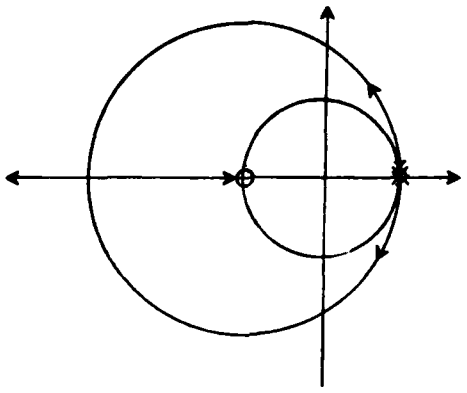


Figure 2.13: z-Plane for $1/s^2$ Plant

To compensate for this, let us add a velocity term to the control law, or

$$u(z) = k\left[1 + \gamma(1-z^{-1})\right]e(z) \tag{2.35}$$

which yields

$$D(z) = k(1+\gamma)\frac{z - \frac{\gamma}{1+\gamma}}{z} \tag{2.36}$$

Now the task is to find values of γ and k that yield good performance. When we did this design previously, we wanted $\omega_n = 0.3$ rad/sec and $\zeta = 0.7$. Figure 2.2 indicates that this s-plane root location maps into a z-plane location of:

$$z = 0.8 \pm 0.17\, j$$

Figure 2.14 shows that for $\gamma = 4$ and $k = 0.08$ or

$$D(z) = 0.4\,\frac{z - 0.8}{z} \tag{2.37}$$

the roots are at the desired location. Normally, it is not particularly advantageous to match specific z-plane root locations, rather it is only necessary to pick k and γ to obtain acceptable z-plane roots, a much easier task. In this example, we wanted to match a specific location only so we could compare the result with the previous design.

The control law that results is

$$u(z) = 0.08\,1 + 4(1 - z^{-1})\,e(z)$$

or

$$u(n) = 0.4e(n) - 0.32e(n-1) \tag{2.38}$$

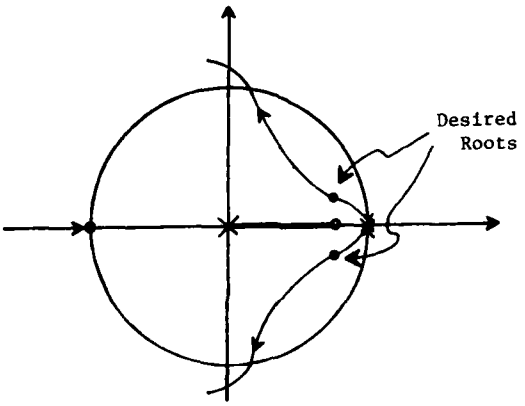


Figure 2.14: Compensated z-Plane Locus for $1/s^2$ Example

(d) Design Comparison

The controller designed using pure discrete methods, Eq. (2.38), basically only differs from the continuously designed controller, Eq. (2.24), by the absence of the $u(n-1)$ term. The $u(n-1)$ term in Eq. (11.24) resulted from the lag term, $(s+b)$, in the compensation, Eq. (2.22), which is typically included in analog controllers because of the difficulty in building pure analog differentiators and for noise attenuation. Some equivalent lag in discrete design naturally appears as a pole at $z = 0$ (see Fig. 2.14) and represents the one sample delay in computing the derivative by a first difference. For more noise attenuation, the pole could be moved to the right of $z = 0$, thus resulting in less derivative action and more smoothing, the same tradeoff that exists in continuous control design.

Other than the $u(n-1)$ term, the two controllers are very similar (Eqs. (2.24) and (2.38)). This similarity resulted because the sample rate is fairly fast compared to ω_n, i.e., $\omega_s \simeq 20 \times \omega_n$. For designs at slower sample rates, the numerical values in the compensations would become increasingly different as the sample rate decreased. For the discrete design, the actual system response would follow that indicated by the z-plane root locations, while the continuously designed system response would diverge from that indicated by the s-plane root locations.

As a general rule, discrete design should be used if sampling slower than $10 \times \omega_n$. At the very least, a continuous design with slow sampling ($\omega_s < 10\omega_n$) should be verified by a discrete analysis or simulation and the compensation adjusted if needed. A simulation of a digital control system is a good idea in any case. If it properly accounts for all delays and possibly asynchronous behavior of different digital modules, it may expose instabilities that are impossible to detect using continuous or discrete linear analysis.

2.5 STATE VARIABLE REPRESENTATIONS

An alternative to models based on the Laplace and z-transform is given by the differential equations of motion in normal form, known as state space models. A continuous model in state form is written

$$\begin{aligned} \dot{x} &= Fx + Gu \\ y &= Hx + Ju \end{aligned} \tag{2.39}$$

The n components of the column matrix x comprise the states, the m components of the matrix u are the controls and y is the $p \times 1$ output matrix. The elements of the matrices (F,G,H,J) are the parameters of the system. It can be shown by variation of parameters that the solution to (2.39) may be written in the form

$$x(t) = e^{F(t-t_0)} x(t_0) + \int_{t_0}^{t} e^{F(t-\tau)} Gu(\tau)\, d\tau \tag{2.40}$$

where the matrix exponential is defined by the series

$$e^{FT} = I + FT + F^2 \frac{T^2}{2!} + \ldots + \frac{F^k T^k}{k!} \tag{2.41}$$

To obtain a discrete model for the system described by (2.39) with the controls given by the piecewise constant output of a zero order hold, we set $t_0 = kT$, $t = kT + T$ and find

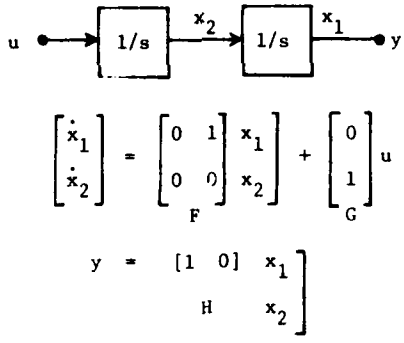
$$\begin{aligned} y(k) &= Hx(k) + Ju(k) \\ x(k+1) &= \Phi x(k) + \Gamma u(k) \end{aligned} \tag{2.42}$$

where

$$\begin{aligned} \Phi &= e^{FT} \\ \Gamma &= \int_0^T e^{F\tau}\, d\tau\, G \end{aligned} \tag{2.43}$$

The reliable and accurate computation of the exponential in (2.43) is a basic requirement of any computer design package for digital control systems.

To illustrate the state space method, we consider the double integrator example shown in Fig. 2.15.



$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{F} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{G} u$$

$$y = \underbrace{[1 \quad 0]}_{H} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Figure 2.15: An Example showing the relation between the Block Diagram and the State Variable Equations

In this case

$$\Phi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} T + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}^2 \frac{T^2}{2} + \ldots$$

$$= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$$

and

$$\Gamma = \int_0^T e^{F(\tau)}\, d\tau\, G = \int_0^T \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} d\tau \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} T & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} \tag{2.45}$$

One of the advantages of the state variable representation is that many of the features of interest in analysis and design can be computed by well known algorithms. In particular, we can find the poles and zeros of a discrete system by eigenvalue analysis. In order to see the relations between the state equations (2.42) and the transfer function given by (2.26), we need to take the z-transform of (2.42). The result is

$$zX(z) = \Phi X(z) + \Gamma U(z)$$

solving,

$$\begin{aligned} (zI-\Phi)X(z) &= \Gamma U(z) \\ X(z) &= (zI-\Phi)^{-1}\Gamma U(z) \end{aligned} \tag{2.46}$$

The z-transform of the output is thus

$$Y(z) = \left(H(zI-\Phi)^{-1}\Gamma + J\right)U(z) \tag{2.47}$$

If we use the $1/s^2$ example, we compute

$$\begin{aligned} (zI-\Phi)^{-1} &= \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}^2 \\ &= \begin{bmatrix} z-1 & -T \\ 0 & z-1 \end{bmatrix}^{-1} \\ &= \frac{1}{(z-1)^2}\begin{bmatrix} z-1 & T \\ 0 & z-1 \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} H(zI-\Phi)^{-1} &= [1 \quad 0]\begin{bmatrix} z-1 & T \\ 0 & z-1 \end{bmatrix}\frac{1}{(z-1)^2} \\ &= \frac{[z-1 \quad T]}{(z-1)^2} \end{aligned}$$

and, finally, $H(zI-\Phi)^{-1}\Gamma$ is

$$\begin{aligned} G(z) &= [z-1 \quad T]\begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix}\frac{1}{(z-1)^2} \\ &= \frac{T^2}{2}\frac{(z+1)}{(z-1)^2} \\ &= b(z)/a(z) \end{aligned} \tag{2.48}$$

Of course (2.48) is the same as (2.33).

The denominator $a(z)$ of (2.48), which defines the poles of the transfer function, arises from computing the inverse of $zI - \Phi$ and is given by

$$a(z) = \det(zI - \Phi) \tag{2.49}$$

While the poles of the transfer function are given by the roots of $a(z) = 0$ from (2.49), we can also compute the poles as "natural frequencies" of the state equations. Consider the equation with no input

$$x_{k+1} = \Phi x_k$$

and consider the states to be all moving together like z^k for some (perhaps complex) z. We have

$$x(k) = X_0 z_0^k$$

If this form is substituted into the equation, we find

$$X_0 z_0^{k+1} = \Phi X_0 z_0^k$$

or

$$X_0 z_0 = \Phi X_0 \tag{2.50}$$

Equation (2.50) is an <u>eigenvalue</u> equation. The scalar z_0 is the eigenvalue and the vector X_0 is an eigenvector. If we write (2.50) as

$$(z_0 I - \Phi)X_0 = 0$$

we can see from elementary matrix algebra that these equations have a solution only if

$$\det(z_0 I - \Phi) = 0$$

so z_0, the eigenvalue of Φ, is a pole of the transfer function.

The zeros of the system can be computed in a similar way. Suppose the states and the input move as z_1^k but the output remains identically zero. Such a z_1 is a zero of transmission of the system. The equations are

$$x = X_1 z_1^k, \qquad u = U_1 z_1^k$$

and

$$\begin{bmatrix} \Phi - z_1 I & \Gamma \\ H & J \end{bmatrix} \begin{matrix} X_1 \\ U_1 \end{matrix} = \begin{bmatrix} 0 \end{bmatrix} \tag{2.51}$$

The solution for z_1 from (2.51) is not an ordinary eigenvalue problem, but a generalized eigenvalue problem since we can write

$$\begin{bmatrix} \Phi & \Gamma \\ H & J \end{bmatrix} \begin{matrix} X_1 \\ U_1 \end{matrix} = z_1 \begin{bmatrix} -I & 0 \\ 0 & 0 \end{bmatrix} \begin{matrix} X_1 \\ U_1 \end{matrix} \tag{2.52}$$

As before, standard numerical methods can be used to solve for the vectors X_1, U_1 and the eigenvalues, z_1, which are the zeros.

The analysis of random inputs is also very direct when equations are in state variable form. One source of signals which may be realistically analysed this way is the quantization which takes place with analog to digital conversion and round-off in fixed point arithmetic. A sketch of the quantization characteristic is shown in Fig. 2.16. In this case, Widrow has shown that the quantization error resulting

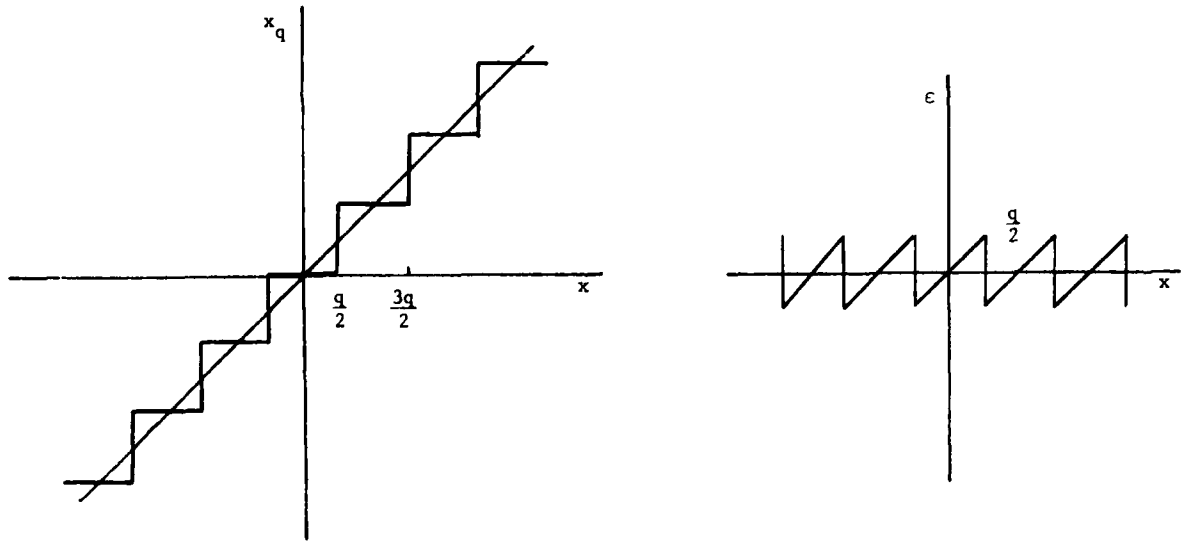


Figure 2.16: The input-output characteristic of quantization or round-off

from these operations can be reasonably modeled as random white noise with a uniform distribution between $-q/2$ and $+q/2$. This distribution results in a "noise" which has mean square value $q^2/12$.

To compute the standard deviation (root mean square error) of a system output having such a noise source, we proceed as follows. Call the noise $\varepsilon(k)$ and write the state equation for the system shown in Fig. 2.17 as

$$x(k+1) = \Phi\, x(k) + \Gamma_1\, \varepsilon(k)$$

We assume the system feedback, if any, is included in Φ and ignore any other external inputs except $\varepsilon(k)$. Next, we define the covariance matrix of $x(k)$ as

$$P(k) = E\, x(k) x^T(k) \tag{2.53}$$

and substitute the state equation to find

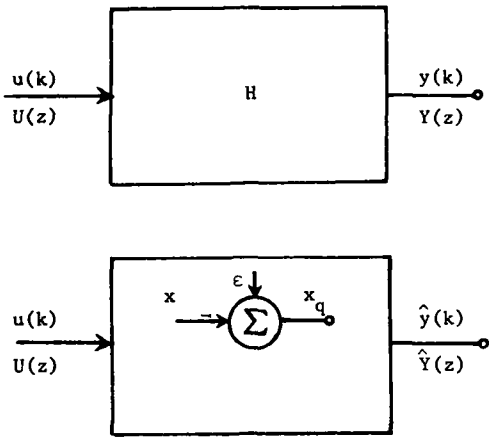


Figure 2.17: Diagram of an ideal system and its practical realization with a source of internal noise

$$P(k+1) = E\, x(k+1)x^T(k+1)$$

$$= E\left[\Phi\, x(k) + \Gamma_1 w(k)\right]\left[x^T(k)\Phi^T(k) + w(k)\Gamma_1^T\right]$$

$$P(k+1) = \Phi\, P(k)\Phi^T + \frac{q^2}{12}\,\Gamma_1\Gamma_1^T \tag{2.54}$$

Assuming that we want the steady state solution, Eq. (2.54) can either be solved sequentially until P no longer varies, or else we can set $P(k+1) = P(k) = P$ and solve the resulting algebraic Lyapunov Equation. Once P is obtained, the mean square value of y is given by

$$E\, y^2 = E\, Hx(Hx)^T$$

$$= H\, P\, H^T \tag{2.55}$$

2.6 CONCLUSIONS

We have thus seen that digital control systems can be analysed by z-transform and state variable methods. With the z-transform representation, we can perform compensation designs using the familiar techniques of root locus and frequency response. With state variable representations we can also analyse the behavior of digital controls with computer aids such as eigenvalue analysis and matrix exponential computations. The design of digital systems using state variable methods will be described in Presentation 7 of the lecture series.

2.7 REFERENCES

[1] R. C. Dorf, Modern Control Systems, Reading, MA.:Addison-Wesley, 1980.

[2] K. Ogata, Modern Control Engineering, Englewood Cliffs, NJ.: Prentice-Hall, 1970.

[3] G. F. Franklin and J. D. Powell, Digital Control of Dynamic Systems, Reading, MA.: Addison-Wesley, 1980.

DESIGN ENVIRONMENTS AND THE USER INTERFACE FOR CAD OF CONTROL SYSTEMS

by

M.J.Denham

Reader in Industrial Computing and Control
Kingston Polytechnic
Penryhn Road
Kingston upon Thames KT1 2EE, *Surrey*
England

Summary

This paper deals with the design of a total CAD environment in which control systems design software can be embedded. This environment includes a rich set of software tools for the creation, modification, simulation and analysis of dynamic system models and a powerful user interface to these tools which incorporates an interactive algorithmic design language. An analogy can be made between such a CAD environment and those which exist currently for software development, e.g. Interlisp. The features which such an environment should possess and how these might be developed are described in the paper. Particular attention is devoted to the user interface since it is now generally recognised that the quality of the man-computer interface is crucial to the successful use of a CAD system. The human factors aspects of the interface are reviewed and examples given of how a CAD system can be designed to take account of these aspects. Reference will be made to the DELIGHT system from the University of California, Berkeley as an example of a user-orientated design environment.

Introduction

In recent years there has been a proliferation of CAD software developed for use in control system design. Several major packages have been developed, starting in the late sixties in the U.K., and many of these are in general use today, having been developed over the years to a high level of sophistication in respect of the design methods which they embody. Most of these packages however have developed less coherently in terms of such aspects as efficiency, ease of use, flexibility, reliability, extendability, etc. This situation is perpetuated by a large proportion of the more recently developed CAD software, of which there is a growing amount as a result of the now easier access to interactive computing facilities. In many other fields of application of computing, in particular in the design of those information systems which are to be accessed by both computer professionals and non-professionals, i.e. casual users whose role is not to develop software but to use it in their particular application, there has been a recent upsurge of interest in the need to provide both a total integrated computing environment in which the system can be used efficiently and effectively and the proper user interface to reduce the occurrence of user error, fatigue, frustration, etc. There are two major aspects to this in respect of what constitutes a good system:

i. the range, effectiveness and integration of the set of software based facilities, or "tools", which are available to the user

ii. the degree to which "human factors" have been taken into account in the design of the man-computer or user interface to the set of software tools

It would appear that we are now at a turning point in the period of development of CAD systems for control systems design. Although there are still serious deficiences in some areas of application, a large number of the methods for design now exist in a sufficiently mature and tested form to be adopted by industry. However industry is faced with a wide range of software, developed by many academic or industrial groups with varying attitudes to those human factors aspects mentioned above. This has resulted in many cases in a reluctance by industry to adopt existing CAD systems or to involve themselves in the costly software redevelopment necessary to turn the CAD system into a more user orientated, integrated form.

It is therefore clear that the time has come to review the situation and consider what properties or features should be essential components of the new generation of control systems CAD software which would overcome these deficiences. It is fortunate that this review can be carried out in the light of recent developments in integrated programming environments and of recent research into the human factors aspects of software based systems in general. In the remainder of this paper we consider the features required for a user orientated, integrated CAD system environment and the means by which they can be achieved. In particular we discuss those aspects of human factors which relate to the design of a CAD system user interface and provide examples of how these can produce a system which is both more efficient and less prone to user error. Prior to that however we consider the major components of an integrated control system design environment and how they can be integrated via a common user interface.

The design environment

Recent developments in the design and production of the integrated programming support environment (IPSE) have provided a clear direction which the design and development of an integrated design support environment (IDSE) might take. Amongst the best examples of such IPSE's are the Interlisp environment (1), (2) and the proposed Ada environment or APSE (3). Let us consider firstly what are the major components of an IPSE such as Interlisp.

The first consideration is that for a truely interactive programming environment, the software tools available to the user should in the main themselves interactive. In particular this implies the need for an interactive programming language, BASIC being a widely known example of such a language. The important distinction here is between languages which require a lengthy, tedious and error-prone code creation, editing, compilation, and linking cycle in order to reach a situation where useful results are being produced and languages where more immediate results can be obtained, for example subsequent to entering a single language statement. This consideration implies that the programming language for such an environment must possess a number of features, including :

i. syntax checking of language statements on entry - eliminates the frustration of syntax errors being detected late in the software development cycle, i.e. at the compilation or linking stage

ii. ability to incrementally compile and, if required, execute sections of the program as they are created. It is obvious that whilst not all semantic or syntactic errors can be detected in this way, since many of these will not be apparent until all the program is present, a large proportion of errors can be thus detected, resulting in less time being spent in debugging the software at later stages of the production cycle

iii. support for a modular program structure to permit the development of large pieces of software in a systematic manner, using a methodology such as "top-down" programming. This is related to the previous feature and implies the need for program modules to be compiled and, if possible, tested in an independent manner. Moreover, the system should provide consistency checking between modules at the time of their creation, together with the necessary tools for administration and documentation of the modules, including different generations fo the same module, e.g. after editing

iv. accurate and informative error reporting at every stage of the program production process.

This leads us to consider the first component of the design support environment for control systems CAD.

The design language

The primary purpose of the design language, analagous to that of the programming language in an IPSE, is to enable the creation of an executable description of the dynamic system under consideration. The process of design, as of programming, is to systematically correct, enhance, extend and optimize this system description until it performs, under execution, in accordance with the required performance specification. (This implies the need to be able to formally or informally specify the required performance of the system - see later in the paper the discussion on formal specification tools).

There are many examples of design languages which meet this primary requirement, since in this respect the design language is equivalent to a system modelling language. Such examples include the SIMNON language (4) and DYMOLA (5), the former having interactive features such as syntax checking on entry, and the latter illustrating the use of a modular structure for modelling large system by decomposing them into a set of smaller subsystems.

Most examples of design languages are entirely textually based, i.e. the descriptions are created by means of text statements, entered via a terminal keyboard. SIMNON and DYMOLA are typical of this mode of use. One could argue however that this is not the most natural method for entry of the structural or topological aspects of the system description and is prone to error, especially in defining the detailed interfaces between the various segments of the description. A more natural method for the system designer to enter the structural or topological part of the system description is a graphical one and this method has been well utilised in other areas of CAD such as electronic circuit design, mechanical component design etc. Techniques for graphical system description have recently been incorporated into control system CAD packages, providing either a block diagram or signal flow graph form of description. However, few, if any, of these support a specific modelling methodology, such as top-down decomposition, nor do they provide a full range of interactive features such as "syntax" checking on entry, incremental compilation and execution, and automatic consistency checking between segments of the system description.

It is clear however that such a graphical design language takes the form of a very high-level langua[illegible] for structural description and is in fact only a pre-processor into the main design language in which non-structural aspects of the system are described and which is designed for direct execution by an interpreter or incremental compilation. The design language must therefore encompass this graphical pre-processor, allowing the user a choice of either using the graphical or the textual mode for entry of the system description. Clearly this demands a close integration between the design language and its graphical pre-processor. The major components of the design environment with respect to the design language requirement are therefore

i. a pre-processor which translates the high-level graphical system description into design language statements

ii. a "syntax" checker which provides error reporting at both the graphical pre-processor and the design language levels of entry of system descriptions

iii. a consistency checker which provides error reporting on inconsistencies in the interfaces between different segments of the system description (graphical pre-processor and design language levels of entry)

iv. an interpreter or incremental compiler which will execute (compiled) segments of the system description in the design language,

v. a compiler from the design language into the "machine language" of the system for fast execution of the system description.

A useful but not essential addition to this list is a "reverse processor" which will translate system descriptions in the design language into the equivalent graphical high level description.

The editors

The second major set of components of the design environment are the system description editors. These components allow the user to create and modify system descriptions in the design language and in its high level graphical equivalent and are necessarily interactive tools. In describing above the need for syntax checking on entry, we have already defined one of the most important functions of the editors, which therefore must be integrated components of the design environment. It is not sufficient to rely on use of the general purpose editor available on the local computer system for entry of system descriptions, since this is, by definition, a non-integrated component of the design environment and hence since it has no knowledge of the design language cannot provide, for example, the syntax checking requirement described above.

The form that the graphical high level language editor takes will depend on the sophistication of the available terminal equipment. Fast, high resolution interactive graphics workstations, such as incorporated in the Three Rivers/ICL Perq and the Apollo machines allow full exploitation of interactive techniques using devices such as tablets and "mouses" and graphical enhancement methods using colour, variable intensity, animation, etc. Current examples mostly incorporate the use of a screen-based menu for selecting a graphical symbol by positioning of a cursor over the appropriate symbol. Cursor positioning can be either by light pen or "mouse"/tablet combination. The more sophisticated systems make use of multiple windows on the graphical display for menus, text from keyboard interaction and display of various segments of the graphical system description. A "window manager" allows the user to create and manipulate the windows on the screen, introducing priority levels for overlapping windows, higher priority windows obscuring those with lower priority, just as a stack of overlapping drawing sheets. This permits, for example, the user to enter a description of a segment of the system in the design language in one window whilst viewing the structure of the overall system description in another, retaining further windows for menu, error messages, system prompts, etc.

For the design language itself, a screen based editor is preferred, ideally using the same techniques and tools as the graphical language editor, e.g. tablet for positioning the cursor, windowing for menus, simultaneous display of multiple segments of the system description to aid editing, visual enhancement techniques for reporting the position of syntax errors, etc.

We will consider further the interactive features of the editors when we discuss the human factors aspects of the design environment later in this paper.

The interactive support tools

The third major set of components of the design environment comprises a range of tools with the sole function of supporting the interactive use of the design environment. The principal requirements of a user of any interactive computer system are

i. to obtain knowledge of the state of the system at any moment in its use, e.g. what data entities are available, what software tools are available, what software tool is in current use, what mode of the editor is in current use, etc.,

ii. to obtain a "history" of the interactive session so far, in order to determine what sequence of actions has led to the current state,

iii. to obtain hard-copy documentation of the current state of various data entities in the system, e.g. the current system description,

iv. to administer the various data entities in the system, e.g. delete unwanted descriptions, provide back-up copies after modification of current descriptions,

v. to "undo" or reverse the effects of a previous sequence of actions, used in conjunction with the "history" facility.

All the above tools are necessarily integrated components of the design environment, some more obviously, e.g. the "history" and "undo" facilities, some less so, e.g. the documentation tools. However, in the latter case, whilst general purpose tools, e.g. for producing listings of text files or graphical hard-copies can be used, user support can be considerably enhanced by the use of special purpose integrated tools, which, for example, automatically incorporate standard system notation or specific project identification on the documentation output, updated according to the current state of the design procedure.

The debugger

The fourth major component is the interactive system description debugger. Since it is highly probable that the system description will contain errors, it is necessary to provide a means of tracing the errors. These are essentially "run-time" errors, i.e. occuring at the time of system execution, and will become apparent to the user either as a result of the run-time system or "executor" flagging an error, e.g. a numerical underflow or overflow, or, more likely, by the user detecting an unusual occurrence in the output data. In the case of a complex, large system description (as in the case of a large program) it is not an easy task to detect such errors and depends upon the user both being able to access and monitor the execution at various points in the system description and having at least an informal criterion against which to judge the performance of the system at the point or points monitored. In order to access and monitor specific points in the system description, e.g. the output of various transducers, control inputs to subsystems, etc. the graphical level system description provides a very useful tool. This allows the user to identify points by means of the light pen or cursor and then further refine his selection by referencing specific variables via the keyboard or by menu selection.

The technique used here bears a strong resemblance to the methods which are in current use for accessing and monitoring signals in an industrial computer-based process control system. In this situation the process operator can select subsystems of his overall process by name or graphically from the display of the overall process flow chart. Further refinements can be made in this way to lower level subsystems until the one in which the error is occuring is detected. A graphical display can then be obtained of suspect signal vs. time plots, the operator then employing his knowledge of plant operation to detect errors.

To assist in the detection of errors, formal verification criteria can be used. These are defined, in the case of the process operator, at commissioning of the plant, and represent bounds on signal levels, rates of change of signals, etc. Use of colour graphics allows the operator to instantly detect violation of bounds from the signal vs. time plot, on which the bounds are displayed. Indeed, the violation of such bounds is often detected automatically and the situation brought to the attention of the process operator by means of alarms.

Whilst the situation in the CAD environment is not so critical from a safety or security viewpoint as it is in the process control case, many of the same techniques can be employed. The use of formal specifications for the system description will permit the automatic verification of "correct" operation of certain components

of the system description at execution time, where a formal verification criterion has been attached to that components. We write "correct" in quotes, since it is clear that the description cannot necessarily be verified entirely in this manner, but only the fact that signals, or their rates of change, etc., lie within certain limits.

Much can be learnt for CAD from the process control industry concerning the use of colour graphics and modes of interaction. A considerable study has been made in that industry into human factors aspects of the techniques of interaction between the process and its operator, since in this field errors due to fatigue, unclear displays, etc. are critical if not disasterous. Many of these techniques, though not of such a critical nature, carry over into the CAD environment. In particular the use of colour to flag errors on the graphical display of the system description, the use of colour on signal vs. time plots to indicate signals going out of bounds, the ability to expand selected subsystems on the graphical display to examine in more detail specific signal monitoring points, are examples of such techniques.

An obvious requirement is for the system description debugger to be interactive during execution of the description. This implies a need for the user to be able to interrupt the system execution either

i. at any point of time defined by the user, e.g. at n seconds after the start of the execution, or

ii. on any signal or set of signals attaining a specified condition, e.g. violating an upper or lower bound, or

iii. instantaneously, from the keyboard, e.g. on visual detection of a specific system condition.

On interruption, the user should be capable of a wide range of actions, including :

i. examination of all signal levels, parameter values, etc. in the system description and their time histories,

ii. the resetting of signal levels, parameter levels, error bounds, etc.,

iii. restarting the system execution, either from the point of interruption or another suitable start point,

iv. storing intermediate results,

v. aborting the execution.

The need for an interactive debugger again stresses the importance of an integrated set of tools, since it is clear that the debugger must be closely integrated with the system execution component, the system description editors and the full range of interactive support tools.

The optimisation language

Closely related to the design language, on which we will expand later, is the optimisation language. Unlike the program design and development process, where new code is added or existing code modified in a manual way in order to get the program to behave in a required manner, the system design process, whilst also possible in the above manner, can also be performed, entirely or in part, by an automatic process using an algorithmic approach. This algorithmic approach is generally based on some optimisation procedure, whereby "free" parameters in the system description, i.e. variables whose values can be adjusted externally to the description, are given a sequence of values according to a pre-defined algorithm until performance of the system description on execution satisfies some criteria or falls within specified bounds. Such algorithms can be single step, as in pole allocation algorithm for state feedback, or multiple step, as in hill-climbing parameter optimisation algorithms.

This major component of the design environment requires close integration with the system design language, since it must access the system description during execution, i.e. be able to interrupt execution, monitor values and reset parameters. Close integration is also necessary at the user interface level. Many of the same language contructions are required by both the design language for defining system functions and by the optimisation language for defining functions of the optimisation algorithm. In practice the design language and the optimisation language become indistinguishable and henceforth we will refer to the design/optimisation language as a single entity, called, for notational simplicity, just the design language (by analogy, an integrated programming support environment = a programming language + tools, an integrated design support environment = a design language + tools).

The shell

This is the final major component of the design environment and is the means by which each of the tools in the environment are accessed or invoked by the user. From the user's viewpoint it is basically a command language processor, where each command invokes a specific tool, e.g. the editors, executor, debugger, etc. From the viewpoint of an integrated environment however, it is unnecessary to distinguish between the shell or command language and the design language. Since the latter is capable of incremental compilation and execution, a command can take the form of a procedure in this language, perhaps with a reserved name. Simply stating the name of the procedure, together with any arguements, will invoke its execution and hence the execution of the appropriate design tool.

This latter aspect of the design environment has a very important implication. This is that since the design tools are simply procedures in the design language, the design tools must themselves be written in the design language itself or some "lower level" language into which the design language is translated or in which the design language can call procedures. This implication for the internal form of the design environment will become clearer when we consider the example of the DELIGHT environment.

Since, from the user's viewpoint, the shell is the major means by which interaction is carried out with the environment, it is important to make this as adaptable as possible, both in terms of replacing existing command procedures by new user-defined names and of introducing new command procedures. This can be done in a number of ways. For example, the user can own a personal "profile" file (as in the Three Rivers/ICL Perq PQCS system) which contains a list of personalised command names and a corresponding list of standard or

user-defined procedure names. In this way, one user command name can be used to invoke one or a sequence of standard procedure names. To add to or otherwise modify this profile file the user can employ the system editor, or, more conveniently can use a command procedure, e.g. "define" or "macro" in DELIGHT, to specify a unique correspondence between invoking a personalised user procedure call and invoking a sequence of one or more standard (built in) procedure calls. Once the "define" or "macro" equivalence has been specified, it is added to that user's personal profile file and maintained either for all subsequent use or just during the remainder of the current user session.

The DELIGHT environment

It would be useful at this point, in order to illustrate those aspects of the design environment described above, to consider the DELIGHT (Design Laboratory with Interaction and Graphics for a Happier Tommorrow) (6), as an example of a currently operational design environment which incorporates many of these aspects. DELIGHT was developed (and is still under development) by Optimization Based Computer Aided Design Group of Professor E. Polak, at the University of California, Berkeley.

The DELIGHT system is modelled to a large extent on the concept of software tools as described in (7). The system has been programmed in Ratfor (Rational Fortran) (7), a Fortran pre-processor which employs structured programming concepts and other features to improve software readability, ease of programming, etc.

The rationale behind the design of the DELIGHT system was the need for a flexible system which would accomodate a variety of design situations, in particular, different system modelling and simulation environments, which would be easy to port to any Fortran environment, which would accomodate both experienced and inexperienced users and which would be efficient in terms of machine use yet allow the user the maximum possible flexibility in expanding and modifying the available set of software tools in the system and its command language structure. Finally, since the system was primarily intended to support an optimisation-based design methodology, the design language is formulated in such a way that optimisation algorithms can be directly implemented in the language.

The design language in DELIGHT is called RATTLE and is an interactive form of the language Ratfor. As such it is sufficiently similar to a range of popular structured programming languages to make it easy to learn, as well as ensuring good programming practice. The interactive feature is obtained by a rapid compilation in an intermediate form, using only a single pass over the source code. This permits incremental compilation, i.e. the direct compilation and execution of the intermediate code at any point in the code creation process, without the need for a lengthy linking and loading phase. Thus the language satisfies our requirement for interaction via incremental compilation.

Since the DELIGHT system is not primarily intended for CAD of control systems, the design language has no specific constructs, data types, etc. which relate specifically to control systems (these are currently in the process of being introduced into DELIGHT in a joing project between Berkeley, Lawrence Livermore Laboratory, Imperial College, London and Kingston Polytechnic). Nevertheless, system descriptions can be created directly in the RATTLE language, in the same way as one would write a FORTRAN or Pascal based system simulation program. A high level graphical design language also does not currently exist (under development at the Lawrence Livermore Labs). However many of the other important features discussed earlier in the paper as of prime importance in a design language exist, mainly in terms of its closely integrated relationship with the other components of the DELIGHT environment.

In particular, the environment provides automatic syntax checking of the language statements on entry; for example, if the user enters :

```
if size > 0
```

the system responds immediately with :

```
if size > 0

ERROR (1) assignment syntax error (size > 0)
```

indicating that the "if" statement is in error - "if" must be followed by a logical expression, which implies, since spaces are used in RATTLE to indicate the end of an expression, that the correct statement is :

```
if size> 0
```

or

```
if (size > 0)
```

In the second case, parentheses override the end of expression indicated by the blank after "size".

The editor in DELIGHT can also be used to enter RATTLE programs in wh ich case entry is related to a file on the system which is created by the editor and retains the code after the end of the terminal session. In this case, however, the editor provides no syntax checking. This is a current deficiency of DELIGHT since, although the editor may be used to enter other text files than RATTLE programs, the editor should give the option to the user to include syntax checking on entry. Currently this can be got around in DELIGHT by the use of the "include" command. If a file "fred" is created by the editor, the command :

```
include fred
```

causes DELIGHT to read lines from the file just as if they had been typed at the terminal, in which case syntax checking is carried out at this time. In addition, use of the "echo" command causes DELIGHT to type each line on the terminal as it is being entered, so that the user is able to see precisely where the errors have occured.

The DELIGHT environment also cc tains a number of interactive support tools. These include the commands :
"display" - allows the user to view all the DELIGHT symbol table entries which are of a specified class and which match an optional specified patte n

e.g. display arrays a*
would list all the user's arrays beginning with the letter "a".

"display-time" displays a list of RATTLE procedure names, the associated cpu times for execution and number of times called so far.

"time" - displays total cpu time since starting the DELIGHT user session.

"date" - displays the current date and time

"history" - displays the last 22 lines of input entered from the terminal

"whatis" - together with an item name as an argument, displays the nature of the item, e.g. variable, procedure, etc.

"list" - displays a listing of a file created in DELIGHT

"output-to" - copies all output to a file, but not to the display

"store" - writes the binary values of all DELIGHT and user variables and arrays into a optionally specified file (default file is "memfile") so that everything may be "restored" at a later date

"restore" - see above

"whoami" - displays the current user name

The real use of the "history" command is not just to view the last 22 lines of input, but to permit the user to re-execute a previous command or other input line. This can be done by typing "!" followed by the initial characters of the line to be reissued or its line number.

The DELIGHT environment also includes powerful debugging facilities. The major commands available for debugging are :

"trace" - used for debugging DELIGHT run-time errors after the program has been suspended for some reason. It prints a list of called Rattle procedures prior to the program suspension.

"enter" - used in conjunction with DELIGHT commands such as "display" to modify, list, etc. local variables in a RATTLE procedure which has been "entered".

The DELIGHT environment allows these debugging tools to be used interactively by providing the user with the ability to interrupt the current program execution. Two kinds of interrupts are available : "hard and "soft" interrupts. A "hard" interrupt is generated when the interrupt key on the terminal is pressed twice in succession. This suspends program execution at the current state, but allows the user to enter new input line at a "second level" of execution, including the debug commands "trace" and "enter" described above. A further interrupt can be issued at this level if required in which case, a "third level" of execution is entered, and so on. A "soft" interrupt is generated when the interrupt key on the terminal is pressed once. An executing program can detect such an interrupt if the word "interrupt" has been included in an "if" statement within the program. This statement can then cause either the execution flow in the program to alter in the normal way, e.g.
if interrupt print x else print y
or the execution to the suspended by means of the "suspend" statement in RATTLE, e.g.
if interrupt suspend

The "suspend" statement is a powerful feature of RATTLE since it allows many of the facilities described above, under the debugger heading, as desirable in an interactive design environment, e.g. suspension of execution on a variable or set of variables attaining a specified condition. On suspension, the user can examine variables values, etc. in a lower execution level in DELIGHT, e.g. using the "enter" command, and then resume program execution from the point at which it was suspended, using the "resume" command.

For optimisation, the DELIGHT environment provides a particularly powerful set of procedures built into a library on the system. These procedures are however simply written in the RATTLE language or in RATFOR or FORTRAN, procedures in both of the latter being callable from RATTLE. In fact, since DELIGHT was primarily intended to be an optimisation tool which could be interfaced to any application dependent modelling and simulation environment, the design language of DELIGHT, namely RATTLE, is as we have already noted, not biased towards a particular application environment such as control system design, but is if anything biased towards the optimisation environment. Hence, in the case of DELIGHT, the design language and the optimisation language, as defined earlier in this paper, are one and the same.

Two possible courses of action therefore exist for extending DELIGHT into an application specific design environment. Firstly the design language RATTLE could be extended to include, for example, application specific data types, e.g. systems and subsystems for control system design, and operations directly on these data types. This could be carried out, to a limited extent and not very efficiently, with the DELIGHT environment by using the language extension facilities such as "defines" and "macros", or major internal extensions could be made to the language parser and compiler to include acceptance and translation of the extended features.

An alternative, which is the current course adopted by DELIGHT, is to provide internal mechanisms within the environment which allow it to be interfaced to another, application specific, modelling and simulation environment. The problem with this course is that the simulation environment cannot be totally integrated with the DELIGHT environment in that it may not use the same techniques of user interaction and would not allow the user to mix RATTLE language statements into the system description which is built in the design language of the simulation environment.

At present, a project is in operation (jointly between Berkeley, Lawrence Livermore Labs, Imperial College, London and Kingston Polytechnic) to extend the DELIGHT environment into a control system design environment using the first course of action, i.e. using the existing language enhancement features of DELIGHT to permit the user to create, modify and execute system descriptions in the RATTLE language in as flexible and easy-to-use manner possible, within the constraints this course of action imposes. This will require the inclusion of RATTLE commands which allow the user to define system descriptions as interconnections of subsystems, execute these descriptions to produce time and frequency responses and incorporate optimisation procedures and system analysis procedures into the design process.

It is a powerful feature in DELIGHT which allows this course of action to be taken, that of the ability for the user to extend and modify the existing set of user commands by means of the "define" and "macro" commands. Firstly we should note that the "shell" in DELIGHT is actually the RATTLE language itself, since commands are implemented in DELIGHT simply as procedure calls in RATTLE, or in the case of more complex commands as a combination of other RATTLE statements and procedure calls. As examples, consider first the DELIGHT command "include". This is implemented as a single call to the procedure "include" and the correspondence between the command and the call is made using the "define" statement in RATTLE:

```
define (include, include_ )
```

The issue of this statement (carried out automatically at the beginning of a DELIGHT session as part of a default user "profile" for the session), effectively replaces all references to the command "include" by a call to the procedure "include_".

Consider also the command "whoami" This is translated by the DELIGHT system into a set of RATTLE statements by means of the "macro" statement in the RATTLE language :

```
macro whoami (
import user_name_
printf 'You are %p.  You are running DELIGHT./n' user_name_
)
```

The "macro" statement simply replaces the reference to "whoami" by the block of RATTLE statements within the pair of brackets. Note that the length of the macro is unlimited, so complex operations can be carried out. Also, both macros and defines can accept arguments to the command as well as simple commands, as in the command to list a file, for example :

```
list fred
```

In summary, therefore, DELIGHT offers a good example of an integrated design environment, which whilst not being application specific to the area of control systems CAD, offers many of the major interactive features and components to be expected in such an environment and the expandability to make it application specific, albeit in a limited and not highly efficient way. A control system CAD environment could well employ many of the features of DELIGHT but would have in addition a design language which might take the form of an extended version of RATTLE, with implicit data types and language constructs which more closely relate to the application and hence make an easier-to-use user interface for the creation and modification of system descriptions.

In addition, the next generation of CAD environments must take into full consideration those human factors aspects which are essential to the design of a good user interface. As the final section of this paper, we will now briefly review some of these aspects.

Human factors considerations in user interface design

No matter how powerful the set of design tools are in a control system CAD environment, the successful use of these tools will depend to a large extent on the quality of the user interface. In order to ensure that the user interface is designed in such a way that human errors are minimized, it is necessary for the designer of the interface to be aware of and sensitive to the abilities and needs of users, from the novice user to the expert. It is also important when potential users of CAD systems come to judge competetive systems for selection for use in their organisation, that they too are aware of the factors which will determine to a great extent the degree of acceptance and benefit which will be achieved in respect of each system due to easy of use, freedom from human error, etc.

A recent paper by Norman (8) classifies the types of human error which can occur in the use of any computer system, including a CAD system and analyses the relationship of these errors to a set of derived design guidelines for the user interface. In the remainder of this paper, we will review those various classes of human error and relate them to the characteristics and components of the proposed integrated design support environment discussed earlier in the paper using the DELIGHT environment to demonstrate various aspects where appropriate.

Based on Norman's analysis, we can classify the classes of human error, in this case errors in carrying out a given intention rather than error in formulating the intention, as follows.

Mode errors - these are errors which result from the user misunderstanding the current state of the system and as a result carrying out an action appropriate to a mode of operation other than the one currently applying. This implies the need either to eliminate different modes entirely or to make the modes highly distinctive. The first alternative is difficult to accomplish fully in a CAD environment which has to accomplish a wide range of complex tasks. Therefore it would appear that the preferred solution is to abolish different modes where they are unnecessary and to make the modes highly distinctive when they have to be incorporated. The implication for the CAD environment, clearly exemplified by DELIGHT, is that as far as possible the user should communicate his design procedures to the CAD system using a single language, for issuing commands to create and modify the system description, for analysing and debugging the results, etc. Where it is unavoidable, other modes can be introduced of a highly distinctive nature, for example, a graphical high level design language for describing structural or topological relationships in the system, since interaction via a light pen or mouse/tablet combination is sufficiently distinctive to avoid mode

errors. Where keyboard interaction is envisaged in combination with and as part of the graphical interaction, this must be made either sufficiently distinctive to avoid error, or must be identical to the normal form of interaction via the textually based design language, i.e. the latter language should be the one used.

A second class of errors is generated if the CAD system offers a range of very similar commands which can be easily confused and yet lead to vastly different results. A good example, quoted by Norman, of this situation is provided by the Berkeley/UNIX "vi" screen editor where each of the letters d, f, g and u has a different meaning if typed in lower case, upper case or in combination with the "control" key. Another example which is very common is related to the confusion which can occur if closely related arguments in a command can be wrongly ordered with possible disasterous consequences, as in a file "copy" command common in many systems, especially if the source file overwrites the destination file without adequate checks, e.g. that the source file is not empty. The implications for the design of CAD environments is broadly to make commands distinctive and where arguments occur in commands to separate similar arguments and provide consistency checking. This is especially difficult to ensure however in an environment such as DELIGHT which allows the user to create new commands, with arguments if required. The rules here then apply to the user as well as the initial designer of the CAD environment.

A further class of errors results from lack of consistency when a user attempts to use experience to derive a new command sequence and the correct form for the new sequence is inconsistent with that prior experience. The requirement here is for the language structure used in the CAD environment to be consistent within its own context, but also with languages which the user may already be familiar with. This is achieved to great effect in DELIGHT where the design language RATTLE is highly consistent with RATFOR and other block structured languages. Since RATFOR is the language in which procedures can be written which are callable from RATTLE, it is additionally likely that the user will not suffer from lack of consistency in moving from one to the other.

If a command in the environment is formed by the simple modification of another, this can also lead to a further class of errors especially if the unmodified command is used much more frequently than the other. In this case, the error which occurs is that in the infrequent case in which the user wishes to issue the modified command, he nevertheless issues the unmodified one due to an unconscious reflex born from familiarity. The lesson to be learned here is not to make commands "overlap" in this way, but as in the earlier case, make all commands distinct. A possible source of error of this class exists in DELIGHT in the commands "echo-oi-to" and "echo-to", although not with disasterous results.

Finally, a major class of errors are so-called "activation" errors, i.e. errors resulting from either the activation of an inappropriate action or failure to activate an appropriate action. The first type of activation errors occur if the CAD system presents misleading information, e.g. inaccurate error reports or ambiguous status reports, causing the user to take a path of action inappropriate to the real condition. This clearly implies the need for accurate, unambiguous display of information by the system. However, since errors of this type are unavoidable to some extent, especially for novice users, it is important to allow the user to "undo" the sequence of inappropriate actions once this has become apparent, or at least be able to view a record of the sequence and take independent action to return to the initial condition. Hence the requirement for the "history" facility in the CAD environment.

This facility is also desirable to cope with the second type of activation error, failure of the user to activate an appropriate action. This generally results from memory failure occurring between the time when the intention to carry out an operation is formed and its final execution. The use of a "history" list permits the user to refresh his memory, especially when the occurence of intermediate actions, e.g. graphical output, listing of a file, has obliterated the normal terminal record of previous actions. Alternatively, or in addition, the use of multiple windows on the display screen of the terminal can be used to great effect to overcome this type of error, e.g. by maintaining separate windows for graphics, listing files, etc. and for command sequences, allowing the windows to overlap, reappear, etc. as required by the user.

In this section of the paper we have given a review of some of the causes of human error in using CAD systems. It is clear that the design of the environment proposed in this paper must incorporate a proper consideration of these factors and incorporate features in the user interface design which will avoid or minimise the occurence of human error as far as possible. These features can be broadly classified as (8):

feedback - an unambiguous, accurate and readily available display of information on the current state of the system

distinctive modes and actions - a provision for the user to clearly distinguish between commands or other language structures or modes of use of the system.

consistency - an adherence of all parts of the system to a common command or language structure and the close relationship to this to other language structures likely to be familiar to the user

reversibility - an ability for all actions, where ever possible, to be reversed, and failing this a provision of obstacles to the initiation of irreversible actions.

Conclusions

In this paper we have attempted to portray something of a vision of the future. The present state of development of control system CAD tools is such that, after a long period of refinement of the methods of design and the more recent attention to the efficiency and reliability of the implementation of these tools, we must now pay careful attention to the total environment in which the design tools are to be used by the design engineer. This requires consideration of the set of software components that this environment must incorporate over and above the basic design tools, and of the user interface which the environment presents, from the viewpoint of increasing the ease of use of the design tools, in particular providing freedom from unnecessary human error.

We expect that integrated design support environments of the type envisaged here will appear over the

next few years in greater number and with a greater level of sophistication than hitherto and lead to a greater willingness on the part of the industrial control system designer to make full use of the powerful design methods which are now available.

References

(1) Teitelman, W. and Masinter, L. The Interlisp programming environment. Computer 14, 4(April 1981), 25 - 33

(2) Sandewall, E. Programming in an interactive environment : the LISP experience. Computing Surveys 10, 1(March 1978), 35 - 71

(3) Wegner, P. The Ada language and environment, ACM SIGSOFT, Software Engineering Notes 5, 2(April 1980)

(4) Elmquist, H. SIMNON - an interactive simulation language for non-linear systems, Proceedings of Simulation '77, Montreux, Switzerland (June 1977).

(5) Elmquist, H. DYMOLA - a structured model language for large continous systems. Proceedings of the Summer Computer Simulation Conference, Toronto, Canada (July 1979).

(6) Nye, W.T. et al. DELIGHT : an optimisation-based computer-aided design system. Proceedings of IEEE ISCAS, Chicago, U.S.A. (April 1981).

(7) Kernighan, B.W. and Plauger, P.J. Software Tools. Addison-Wesley, 1976.

(8) Norman, D.A. Design rules based on analyses of human error, Communications of the ACM 26, 4(April 1983), 254 - 258.

MODELING AND SIMULATION TECHNIQUES

K.J. Åström

Department of Automatic Control
Lund Institute of Technology
Lund, Sweden

Abstract

Systematic methods for design of control systems require mathematical models of the dynamics of processes and disturbances. This lecture is an overview of techniques for obtaining such models. Modeling from first principles and modeling from data are discussed. Particular emphasis is given to computer aided tools for obtaining and verifying the models. Two interactive software packages Simnon for nonlinear simulation and Idpac for data analysis and identification are described. The paper ends by some speculation on the tuture trends.

1. INTRODUCTION

The design of a control system is frequently divided into two steps: determination of a mathematical model and design of a control strategy. In fact most of the control theory that has been developed postulates that a model of the system and its environment is available. To use much of the existing control theory it is therefore necessary to have techniques to determine suitable models for the processes to be controlled.

Before the advent of modern control theory most results were restricted to linear systems, assuming a model specified by a transfer function. The modeling then reduces to the determination of transfer functions. This is conveniently done experimentally by introducing a sinusoidal variation in the input and measuring amplitude and phase between the input and the output.

A characteristic feature of many significant results in control theory that have been developed over the past 30 years is that they require other models than transfer functions. Typically, *many of the results of modern theory assume* that the system is described by time domain models like

$$\frac{dx}{dt} = f(x, u, v)$$

$$y = g(x, u, v) \qquad (1)$$

where u is the input, y the output, x the state variable and v a disturbance.

To apply the results of modern control theory it is therefore necessary to have techniques available to determine models like Eq. (1). In this paper we will outline some progress towards the solution of this problem.

2. MODELS AND MODELING

The notion of a mathematical model is fundamental to science and engineering. A model is a very useful and compact way to summarize our knowledge about a process. A model is also a very effective tool for education and communication. For control engineering, models are significant because virtually all existing control theory is based on the assumption that mathematical models of the process, its environment and the criterion are available. The models are also used to select the structure of the control system, appropriate sensors and actuators. They are also useful for process design.

It is important to emphasize the danger of believing that a process can be characterized by *one mathematical model*. It is much more fruitful to represent a process with a hierarchy of models, ranging from very detailed and complex simulation models of whole processes to the 'back of an envelope model' which is easily manipulated analytically. The simple models are used for exploratory purpose and to obtain the gross features of the system behaviour. The very complicated simulation models, which also may contain pieces of the real process, are used for a detailed check of the control system to make sure that nothing has been neglected. The complicated models take a long time to develop and they are costly to maintain. They do, however, reproduce the properties of the real system with high fidelity and they are a necessity for design of critical processes. Between the two extremes there may be many different types of models which are

suited for design of control systems. The crucial problem is to steer between oversimplification with the danger of disaster and overcomplication which is too expensive. The trademark of good engineering is to choose the right model for each specific purpose.

There are in principle two different sources from which models can be obtained, from prior experiences in terms of physical laws, (modeling from physics) or by experimentation on a process (identification). When it is attempted to obtain a specific model it is of course beneficial to combine both approaches.

Classical control theory was based on the idea to model a dynamical system by a transfer function or an impulse response. Such a model is referred to as an external_description or a black-box_model because it gives only a relation between the system input and output i.e. the external variables. The success of the classical control theory can partly be attributed to the fact that there were powerful experimental techniques, frequency and transient response analysis, which made it possible to obtain the appropriate models. These classical methods for system identification are still very useful for process modeling and they should always be kept in mind even if they have largely vanished from most current papers on automatic control.

The so-called modern control theory is largely based on a process model in terms of a state-equation. This is called an internal_model or a white-box_model because the state model describes explicitly all the internal couplings between the inputs, outputs and the state-variables. The problem of obtaining suitable internal descriptions for different process is one of the major problems in applications of modern control theory. In aerospace applications the desired models can sometimes be derived from basic physical laws. When this is not possible process it is necessary to use experiments. Much of the current research in system identification has been inspired by the desire to obtain process models from process experiments.

In control system design it is also important to have models of disturbances. The external models are often given in terms of spectral densities and covariance functions. When using internal representations the disturbances are instead represented as outputs of dynamical systems driven by white noise. Models for disturbances can only, rarely, be determined from first principles. Process experiments combined with system identification is thus often the only possibility to model disturbances.

3. MODELING FROM PHYSICS

The required models can in principle be derived from basic physical laws expressing conservation of mass, momentum and energy, combined with material equations like Boyle's law or Hooke's law. The models obtained in this way have the advantage of a wide range of validity. Usually, they also provide a good insight into the behaiviour of the system. The drawbacks of modeling from physical laws are that; the required knowledge is not always available; the modeling is frequently time-consuming (consider the required to develop Newtonian mechanics); and it is often difficult to make sensible approximations. A typical difficulty is to find good approximations of distributed parameter systems. Experience has shown that the models developed from basic physical principle tend to be complex. A complex model indirectly implies a complex control strategy and vice verse. If a system can be successfully controlled by a simple strategy it can probably also be modeled satisfacorily by a simple model. In the area of flight control systems rigid body dynamics is well understood in the sense that models can conveniently be derived from physical principles. The models are also supported by experimental data from wind tunnel experiments. On the other hand phenomena like flutter, aeroelasticity and large angle of attach behavior are not sufficiently well known for models suitable for control to be derived from physical principles alone.

The problem of obtaining a model like Eq. (1) for a dynamical system is sometimes called the inverse problem because a solution is given and the problem is to find the equation which has the given solution. Problems of this type do of course arise in many fields: biolgy, medicine, economy, physics and chemistry. There are certain advantages in the modeling and identification problems originating from the field of automatic control. There is a specific purpose in doing the modeling (design of control strategies). It is often fairly easy to do experiments. (Control systems are designed in such a way that control variables can be manipulated and outputs measured.)

Strictly speaking, the problem may not be so well defined. Even if the design of a control system is the final goal, it is of course valuable to have insight and understanding of system properties which do not enter the control design directly. The possibility of making experiments may be limited because it may be necessary to experiment under normal operating conditions, and large changes in inputs may be prohibitive for safety and economic reasons.

Process models can be obtained from basic physical laws, from pure input-output experiments or from a combination of these approaches. Determination of a model from input-output measurements only has the advantage of being done quickly. Experience has also shown that it usually leads to fairly simple models. One serious disadvantage is that in most methods it is possible to determine linear models only. This means that the

validity of the model is limited. A change in operating conditions, input signals, etc., may thus lead to a different model. Another disadvantage is that available a priori knowledge is not used. For example, it is almost impossible to exploit a priori knowledge when a transfer function is determined using frequency response methods. However, in som cases the input-output approach may be the only possibility. This may be the case in the characterization of disturbances such as load variations. Recognizing the advantages and disadvantages of modeling from physical equations and from input-output experiments alone it seems highly desirable to try to exploit both methods in order to solve the modeling problem.

4. MODELING FROM DATA

Techniques for determining dynamic models from experimental data is called identification in the control engineering literature. A brief review is given in this section. For more details we refer to the survey papers [1] and [2], the books [3], [4], [5], [6] and the proceedings from the IFAC Symposia on system identification in Prague 1967, 1970, the Hague 1973, Tbilisi 1975, Darmstadt 1979 and Washington 1982.

It was mentioned in the introduction that identification was the experimental aspect of process modeling. In particular system identification includes

1. Experimental planning
2. Selection of model structure
3. Parameter estimation
4. Validation

Experimental planning includes the decision to make open- or closed-loop experiments, selection of input signals and sampling rates. It also includes considerations of many of the practical problems that are associated with experiments in an industrial environment. The experiment will result in data D in the form of records of inputs and outputs from the process. The selection of model structure is frequently based on physical principles or on a priori knowledge of the process dynamics. The purpose of the parameter estimation is to determine the parameters of the model based on the experimental data. Model validation is the procedures used to ensure that the model obtained is reasonable. This frequently requires more experiments.

In practice the procedure is iterative. When investigating a process where the a priori knowledge is poor it is reasonable to start with transient and frequency response analysis to get crude estimates of the dynamics, the region of linearity, and the disturbances. Based on these results it can then be attempted to derive physical models where the results of the frequency response analysis are used to guide various approximations. The results of the preliminary investigation can then be used to plan suitable experiments where the plant is perturbed and the output observed. The data obtained are then used to estimate the unknown parameters. New experiments are done for the validation. Based on the results and the experience obtained, the model may be improved and new experiments can be planned etc.

The different phases of system identification will now be discussed in more detail.

Experimental planning

It is often difficult and costly to perform experiments on real processes. It has therefore been a desire to develop methods that will relax the constraints on the experiments at the expense of increased computations. While many classical methods depended strongly on the input to have a precise form the newer techniques can handle virtually any type of input signal. The only requirements on the input signal is that it should excite all the modes of the process sufficiently (persistent excitation).

There is a substantial literature on the planning of statistical experiments [7], [8]. The purpose is to find optimal designs of experiments. In process modeling this corresponds to finding optimal input signals. Considerable research has been devoted to this problem [9], [10]. All results on optimal input design are, however, based on the assumption that a model of the process is known. This means that the results can only be used when a reasonably good a priori knowledge of the dynamics of the process and its environment is available. Good applications to determination of aircraft flight dynamics and ship steering dynamics are known. The results may, however, also be strongly misleading if the process dynamics differ from the a priori assumptions. The results on design of optimal inputs are also restricted because it is frequently assumed that the process is open loop during the experiment.

The possibility to base system identification on data obtained under closed loop control of processes has been explored recently [11]. The results obtained are very useful from the point of view of applications. The main difficulty with data obtained from a process under feedback is that it may be impossible to determine the desired models i.e. lack of identifiability. It has, however, been demonstrated that identifiability can be recovered if the feedback is sufficiently complex. It helps to

make the feedback nonlinear, time-varying and to change the set points. A practical way to make the feedback time-variable is to switch between different linear feedbacks. There are cases where data from closed loop experiments will give better results than open loop experiments [11]. A practical way to arrange the experiments is to provide the process with a self-tuning regulator to minimize the fluctuations in process variables and to change the set-point of the regulator with as large signal as possible.

Model structures

The model structures used are derived from prior knowledge of the dynamics of the process and its environment.

In some cases the only a priori knowledge available is that the process can be described as a linear system in a parcticular operating range. It is then natural to use general representations of linear systems i.eg. black box model. Typical examples of black box models are the transfer function model

$$U(s) = G(s)U(s) + H(s)E(s) \tag{2}$$

and the difference equation model

$$\begin{aligned} &Y(t) + A_1 y(t-1) + \ldots + A_n y(t-n) = \\ &= B_1 u(t-1) + \ldots + B_u u(t-n) + e(t) + C_1 e(t-1) + \ldots + C_u e(t-n) \end{aligned} \tag{3}$$

where u is the input, y the output and e is a white noise disturbance. The parameters as well as the order n in the vector difference Eq. (3) are considered as unknown parameters.

Sometimes it is possible to apply known physical laws to derive models of the process which anly contain a few parameters. For lumped parameter processes such white box models may be of the form

$$\begin{aligned} \frac{dx(t)}{dt} &= f\left[x(t), u(t), v(t), \theta\right] \\ y(t) &= g\left[x(t), u(t), e(t), \theta\right] \end{aligned} \tag{4}$$

where u is the input, y the output, x the state, e and v disturbances and θ a vector of unknown parameters. Linear models with

$$\begin{aligned} f(x, u, v, \theta) &= A(\theta)x + B(\theta)u + v \\ g(x, u, e, \theta) &= C(\theta)x + D(\theta)u + e \end{aligned} \tag{5}$$

are particularly common.

For distributed parameter processes the model given by Eq. (4) is replaced by a partial differential equation.

In many practical cases the models may be composed of parts which are black box models and parts which are white box models. Such models are called grey box models. Notice that a significant trend in the recent development is to attempt to model both the process dynamics and the disturbances. This is of course in close agreement with the needs of the control engineer because without disturbances there is no control problem.

Criteria

When formulating an identification problem a criterion is introduced to give a quantity expressing how well a model M fits the experimental data D. The criteria can be postulated. By making statistical assumptions it is also possible to derive criteria from probalistic arguments. Criteria can therefore be viewed from two points of view. They are often expressed as

$$V(e) = \int_0^T h\left[e(t)\right]dt \tag{6}$$

or for discrete time systems

$$V(e) = \sum_{t=0}^{N} h\left[e(t)\right] \tag{7}$$

where e is the input error, the output error or the generalized error. See [1]. The prediction error is a typical example of a generalized error. The function h is frequently chosen as a quadratic but it is also possible to have many other forms. Particularly it may be useful to have functions which do not grow as rapidly as e^2 for large e. See [12].

The first formulation, solution and application of an identification problem was given by Gauss in his famous determination of the orbit of the planet Ceres [13]. Gauss formulated the identification problem as an optimization problem and introduced the principle of least squares. Ever since, the least squares criterion has been used extensively. Nowadays the least squares method (LS) commonly refers to a method where not only the criterion is quadratic but also the model is such that the errors (i.e. the differences between the observed and computed values) are linear in the parameters. The solution of the problem can then be given in closed form. It should, however, always be remembered that least squares is often chosen for mathematical convenience.

Because of the simplicity of the least squares problem it is always the tempting to use this formulation. It is, however, useful to remember that if the identification problem is solved using a digital computer there is no particular reason to choose a quadratic criterion. When the disturbances of a process are described as stochastic processes, the identification problem can be formulated as a statistical parameter estimation problem and the whole artillery of statistical estimation methods become available. The maximum likelihood method is a popular technique which has many attractive statistical properties. See e.g. [14], [15] and [16]. This method can also be interpreted as a least squares criterion if the quantity to be minimized is taken as the sum of squares of the prediction errors or more precisely in the case of discrete time observations at times $t_0, t_1, \ldots, t_N$ the criterion is given by

$$V(\theta) = N/2 \log \det R + \frac{1}{2}\sum_{1}^{N} \varepsilon^T(t_i) R^{-1} \varepsilon(t_i) + \frac{Np}{2} \log 2\Pi \tag{8}$$

where $\varepsilon(t_i)$ are the prediction errors

$$\varepsilon(t_i) = y(t_i) - \hat{y}(t_i|t_{i-1}) \tag{9}$$

The maximum likelihood criterion Eq. (8) is based on the assumption that the prediction errors ε are normally distributed. Notice, however, that the criterion Eq. (9) can still be postulated even if the prediction errors are not normal. The corresponding identification method then becomes a prediction error method.

Parameter estimation methods

The parameter estimation paroblem can be formulated as follows. Given data D, in the form of input-output records from a process, class of models M and a criterion C. Find a model in the class M which fits the data as well as possible according to the criterion C. There are many possibilities to combine experimental conditions, model classes and criteria. There are also many different ways to organize the calculations of the estimate. Consequently there is a large number of different identification methods available. It is useful to remember, however, that they are all based on the same principle and that they only differ in the choice of model structures, criteria and organization of the calculations.

One broad distinction is between on-line methods and off-line methods. The on-line methods give estimates in real time as the measurements are obtained. The on-line methods are the only alternative if parameters are timevarying and when the estimates must be produced in real time. The on-line algorithms available are also frequently simpler to program than the off-line methods. The draw-back with the on-line methods is that they are less reliable. They may not necessarily converge. Even if they converge they may converge to the wrong solution. In many cases the off-line methods will also give estimates with higher precision. Off-line techniques are therefore preferrable unless the processes are timevarying or it is necessary to obtain estimates in real time.

Choice_of_methods

The large number of identification methods available are of course very confusing for an industrial engineer who is primarily interested in having a tool to obtain a model. Several attempts to compare different identification methods have also been made. See e.g. [17] and [19]. The comparisons are largely inconclusive in the sense that there is no method that is universally best. Fortunately it appears, however, that the choice of techniques is not crucial. Personally I would recommend a prospective user to learn the classical methods (frequency and transient response analysis, correlation and spectral analysis), least squares with extensions and maximum likelihood. The least squares method is very simple and easy to understand. Under some circumstances it will give estimates with the wrong mean values (bias). This can, however, be overcome by using various extended least squares and generalized least squares. The major drawback by least squares is that it requires a model structure which is linear in the parameters. The maximum likelihood method is a very general technique which can be applied to a wide variety of model structures.

Model_validation

When a model has been obtained from experimental data it is necessary to check the model in order to reveal its inadequacies. Black box models should be given particular attention in this respect. For model validation it is useful to determine step responses, impulse responses, poles and zeros, model- and prediction errors etc. Calculation of statistical quantities like correlation of prediction errors and cross correlations between inputs and prediction errors can also be revealing. Since the purpose of the model validation is to scrutinize the model with respect to inadequacies it is useful to look for quantities that are sensitive to model changes.

Provided that assumptions on the data generation can be made, many useful results can be obtained. For example it is sometimes possible to determine the statistical properties of the estimates for large data sets. Assumpting that the mechanism which generated the data is known it is also possible to analyse if the estimates converge with increasing data sets. In particular if the model structure is flexible enough to include the data generation mechanism it is then also possible to obtain conditions such that the estimates will converge to their "true values". Statistical methods can also be used to decide between models having different structures. For example, the choice between the models having a different number of parameters can be formulated as a hypothesis test using the test quantity

$$t = \frac{V_1 - V_2}{V_2} \cdot \frac{N - p_2}{p_2 - p_1}, \quad p_2 > p_1 \qquad (10)$$

where V_i is the loss function (e.g. the negative logarithm of the likelihood function) of the model having p_i parameters and N the number of sampling points. The model with more (p_2) parameters is preferred if the value t is sufficiently large.

An interesting approach to this problem has recently been given by Akaike [19] who suggests using the criterion

$$AIC = -2 \log (ML) + 2p \qquad (11)$$

where ML is the maximum likelihood and p is the number of parameters. Akaike's criterion, which is based on information theoretic considerations, is equivalent to Eq. (10) if V_1 is close to V_2. Other tests are given in [8].

When the identification problem is formulated as a statistical parameter estimation problem, there are many ideas and results from statistics that can be exploited. For example it is possible to assign accuracies to the parameter estimates by using the second derivative of the likelihood function. The statistical approach requires, however, that certain assumptions are made on the mechanism which generated the data i.e. the real process. This is most unpleasant because the real process is often nonlinear, timevarying, and infinite dimensional and little is known about it.

Great care should therefore be used when the results of statistical analyses are interpreted. It has been found empirically that many methods work very well on simulated data but very poorly on real data. This reflects the fact that certain results are sensitive to variations in the data generation and it indicates the needs for research into the problem of mismatch between the model structure and the data generation. A particular problem of overfitting clearly illustrates what can happen. If a model which has too many parameters is fitted to a given data set an extremely good fit can of course be obtained for a particular data set. The high order model may, however, be very poor when applied to another data set. It is therefore a good practical rule to work with at

least two data sets. One set is used for the identification and the other for the validation.

5. COMPUTER AIDS

It is a substantial effort to solve a system identification problem for an industrial process if no prior experience and no software is available. The effort can be reduced substantially if good computer software is available. In particular it has been our experience that the time and effort can be reduced substantially if suitable software for interactive computing is available.

Interactive computing requires an efficient man-machine interface. A graphical display which can be used to show curves is a necessity. Interactive software allows the problem solver to combine his insight and intuition with extensive calculation. It also gives a direct link between the problem solver and the computer without needing programmers as intermediaries.

Interaction principles

When designing a system for man-machine interaction it is important to realize that there is a wide range of users, from novices to experts, with different abilities and demands. For a novice who needs a lot of guidance it is natural to have a system where the computer has the initiative and the user is gently led towards a solution of his problem. For an expert user it is much better to have a system where the user keeps the initiative and where he gets advice and and help on request only. Attempts of guidance and control by the computer can lead to frustration and inefficiency. It is highly desirable to design a system so that it will accomodate a wide range of users. This makes it more universal. It also makes it possible to gradually shift the initiative from the computer to the user as he becomes more proficient.

To obtain an efficient man-machine interface it is desirable to have hardware with a high communication rate and a communication language with a good expression power. When our projects were started we were limited to a teletype and a storage oscilloscope. There were also limited experiences of design of man-machine interfaces. The predominant approach was a question-and-answer dialog. See [20].

In our projects it was discovered at an early stage that the simple question-and-answer dialog was too rigid and very frustrating for an experienced user. The main disadvantage is that the computer is in command of the work rather than the user. This was even more pronounced because of the slow input-output device (teletype) which was used initially.

Our primary design goal was to develop tools for the expert. A secondary goal was to make the tools useful also for a novice. To make sure that the initiative would remain with the user it was decided to make the interaction command oriented. This was also inspired by experiences from programming in APL. Use of a command dialog also had the unexpected effect that it was possible to create new user defined commands easily. It was thus possible to use the packages in ways which were not anticipated when they were designed. The decision to use commands instead of a question and answer dialog thus had far reaching consequences. A more detailed discussion of the different types of dialogs and of our experiences of them is given in [21]. Today there is a wide range of experiences of designing man-machine interfaces in many different fields. Our own conclusions agree well with those found in [22] and [23]. Conclusions are based on different hardware.

Examples of commands

The structure of the commands we introduced will now be described. The general form of a command is

```
NAME    LARG1    LARG2...    ←    RARG1    RARG2...
```

A command has a name. It may also have left arguments and right arguments. The arguments may be numbers or names of objects in a data base. In our packages the objects are implemented as files because this is a simple way to deal with objects having different types. A few examples of commands are given to further illustrate the notion of a command. The command

```
MATOP    S  ←  A * B + C
```

simply performs the matrix operation expressed to the right of the arrow.

The command

```
POLOP    S  ←  A * B + C
```

performs the same operation on polynomials.

The command

```
INSI U 100
   )PRBS 4 7
   )EXIT
```

generates an input signal of length 100 called U. The command has options to generate several input signals. The options are selected by additional subcommands. PRBS is a subcommand which selects a PRBS signal. The optional arguments 4 and 7 indicate that the PRBS signal should change at most every fourth sampling period and that its period should be 2^7-1. The subcommand EXIT denotes the end of the subcommands.

The command

```
DETER  Y  ←  SYST  U
```

generates the response of the linear system called SYST to the input signal U.

The command

```
ML   PAR  ←  DAT   N
```

fits an ARMAX model of order N to the data in the file called DAT and stores the parameters in a file called PAR.

The command

```
OPTFB   L   CLSYS  ←  LOSS   SYS
```

computes the optimal feedback gain L and the corresponding closed loop system CLSYS for the system SYS and the loss function LOSS.

Short form commands and default values

In a command dialog it is highly desirable to have simple commands. This is in conflict with the requirement that commands should be explicit and that it may sometimes be desirable to have variants of the commands. These opposite requirements may be resolved by allowing short forms of the commands. The standard form for the simulation command is SIMU. If no other command starts with the letter S it is, however, sufficient to type S alone. It may also be useful to have a simple way of renaming the commands. We have experimented with short form commands and renaming mechanisms. These functions are, however, not implemented in our standard packages.

A similar mechanism may be used for commands which use arguments by introducing a default mechanism so that previous values of the arguments are used unless new values are specified explicitly. The concept is illustrated by an example.

The syntax diagram for the command SIMU is shown in Fig. 1. The diagram implies that any form of the command which is obtained by traversing the graph in the directions of the arrows is allowed. For example the command

```
SIMU   0   100
```

simulates a system from time 0 to time 100. If we want to repeat the simulation a second time with different parameters it suffices to write

```
SIMU
```

The arguments 0 and 100 are then taken as the previously used values.

It follows from Fig. 1 that start and stop times and the initial time increment may be specified. It is also possible to mark curves by the argument MARK. A simulation may also be continued by using the end conditions of a previous simulation as initial values. This is done by the command extension CONT. The results of a simulation may also be stored in a file.

Macros

The commands are normally read from a terminal in a command driven system. It is, however, useful to have the option of reading a sequence of commands from a file in storage instead. Since this is analogous to a macro facility in an ordinary programming language the same nomenclature is adopted. See [24]. The construction

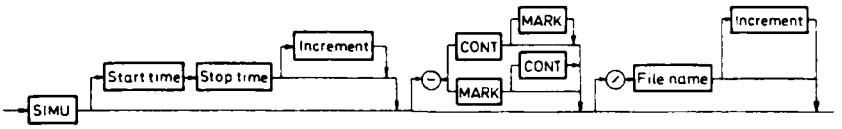


Fig. 1 Syntax diagram for the command SIMU.

```
MACRO NAME
   Command 1
   Command 2
   Command 3
END
```

thus indicates that the commands 1, 2 and 3 are not executed but stored in memory. The command sequence is then activated simply by typing NAME.

Macros are convenient for simplification of a dialog. Command sequences that are commonly used may be defined as macros. A simple macro call will then activate a whole sequence of commands. The macro facility is also useful in order to generate new commands. Macros may also be used to rename commands. This is useful in order to tailor a system to the needs of a particular user.

The usefulness of macros may be extended considerably by introducing commands to control the program flow in a macro, facilities for handling local and global variables and by allowing macros to have arguments. By having commands for reading the keyboard and for writing on the terminal it is also possible to implement menu driven dialogs using macros.

An interactive CAD program based on a command dialog with a macro-facility may be viewed as an extendable high level problem solving language.

Error checking

It is important in interactive systems to have test for avoiding errors. It is thus useful to check data types and to test problems for consistency whenever possible.

Implementation

It is straightforward to implement a command driven interactive program. The structure used in all packages is shown in Fig. 2. The main loop reads a command, decodes it and performs the required actions. All parts of Fig. 2 except the action routines are implemented as a package of subroutines called Intrac. These subroutines perform command decoding, file handling and plotting. Intrac also contains the macro facility. Macros may have formal arguments, local and global variables. They permit conditional and repeated execution of commands as well as nested use of macros. There are read and write commands, which can be used to implement menu dialogs. It is possible to mix command mode and

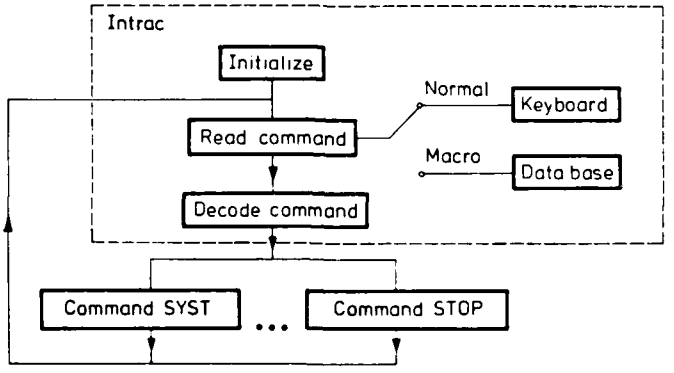


Fig. 2 Skeleton flow chart for a command driven program with a macro facility.

question mode, since the execution of a macro may be suspended and resumed later. A description of Intrac is given in [25] and [26]. The commands available in Intrac are listed in Appendix A.

To build a package using Intrac it is necessary to write the action routines i.e. the subroutines that performs the desired tasks. The commands are then entered in the command table of the command decoder. It is also easy to add a command to a package, to move commands between packages and to create special purpose packages. Intrac may thus be viewed as a tool for converting a collection of Fortran subroutines into an interactive package. Intrac has also been used to implement other packages by other groups.

The structure with a common user interface for all packages is advantageous for the user because the interaction and the macro commands are the same in all packages. This simplifies learning and use of the packages.

How to choose commands?

The selection of commands is one of the major issues when designing a CAD package. The commands determine how useful a package is and how easy it is to learn. It is important that commands are complete in the sense that they allow use of a wide range of techniques in an area. Otherwise the designer will only try those approaches for which commands are available. Commands should also have a considerable expression power so that a control system designer can do what he wants with a few commands. The commands should also reflect the natural concepts from a theoretical point of view. This would make it easy for a user well versed in control theory to use a package. The commands should also be few and simple so that they are easy to learn and remember. This is of course in conflict with requirements on completeness and expression power. Selection of commands is thus a good exercise in engineering design.

Based on experiences from our projects we have arrived at some design principles. A set of basic commands which correspond to the elements of the theory and which allow coverage of a certain problem area are first determined. Simplifications and extensions are then generated using the macro facility.

6. SIMNON

Simnon is a package for interactive simulation of nonlinear continuous time systems with discrete time regulators. See [27] and [28]. The package also includes noise generators, time-delays, a facility for using data files from Idpac as inputs to the system and an optimizer.

Simnon allows a system to be described as an interconnection of subsystems. There are two types of subsystems, continuous time systems and discrete time systems. This makes Simnon well suited for simulation of digital control systems. The characteristics of Simnon are illustrated by an example.

Listing 1 gives a description of a feedback loop consisting of a continuous time process called PROC and a digital PI regulator called REG. The process is an integrator with input saturation. The interconnections are described by the connecting system CON.

The following annotated dialog illustrates how Simnon is used.

```
CONTINUOUS SYSTEM PROC
"Integrator with input saturation
Input u
Output y
State x
Der dx
upr=if u<-0.1 then -0.1 else if u<0.1 then u else 0.1
dx=upr
END
```

```
DISCRETE SYSTEM REG
"PI regulator with anti-windup
Input yr y
Output u
State i
New ni
Time t
Tsamp ts
e=yr-y
v=k*e+i
u=if v<ulow then ulow else if v<uhigh then v else uhigh
ni=i+k*h*e/ti+u-v
ts=t+h
k:1
ti:1
h:0.5
ulow:-1
uhigh:1
END


CONNECTING SYSTEM CON
"Connecting system for simulation of process PROC
"with PI regulation by system REG
yr[REG]=1
y[REG]=y[PROC]
u[PROC]=u[REG]
END
```

Listing 1 - Simnon description of a simple control loop consisting of a continuous time process and a discrete PI regulator.

Command	Action
SYST PROC REG CON	Activate the systems.
AXES H 0 100 V -1 1	Draw axes.
PLOT yr y[proc] u[reg]	Determine variables to be plotted.
STORE yr y[proc] u[reg]	Select variable to be stored.
SIMU 0 100	Simulate.
SPLIT 2 1	Form two screen windows.
ASHOW y	⎫Draw y with automatic
SHOW yr	⎬scaling and yr with the
	⎭same scales in first window.
ASHOW u	Draw u with automatic scaling in second window.

The result is shown by the curves in thin lines shown in Fig. 3. These curves show that there is a considerable overshoot due to integral windup. The regulator REG has anti-windup. The state of the regulator is reset when its output is equal to ulow or uhigh. The limits were set to ulow=-1 and uhigh=1 in the simulations shown with thin lines in Fig. 3. These values are so large that the integral is never reset. The simulations shows in thin lines in Fig. 3, thus correspond to a regulator without wind-up. The actual actuator limitations correspond to ulow=-0.1 and uhigh=0.1. The commands

```
PAR  ulow:-0.1
PAR  uhigh:0.1
```

change the parameters and the command SIMU now generates the curves shown in thick lines in Fig. 3. Notice the drastic improvements due to the nonlinearity in the regulator.

The first version of Simnon was implemented in an MS project. Simnon has gone through several stages of development. A list of the commands in Simnon is given in *Appendix B.*

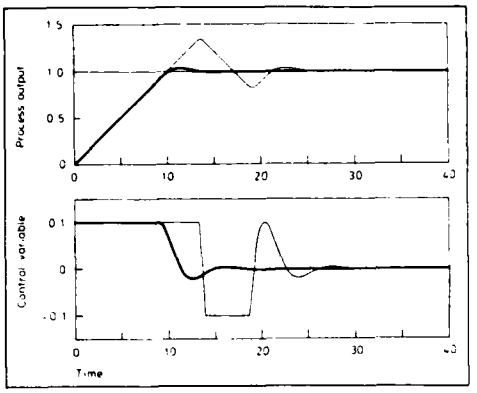


Fig. 3 Results of simulation of process with PI regulator. Thin lines show results with ordinary regulator and thick lines show results for regulator with anti-windup.

7. IDPAC

Idpac is a package for data analysis and identification of linear systems having one output and many inputs, [29] and [30]. Time series analysis of ARMA and ARIMA models is a special case. The package has commands for manipulation and plotting of data, correlation analysis, spectral analysis and parametric system identification. There are also commands for model validation and simulation. The basic techniques used for parameter estimation are the least squares method and the maximum likelihood method. By using the macro facility it is, however, possible to generate commands for most of the parameter estimation methods which are proposed in literature. It was actually in the development of Idpac that the power of the macro concept became apparent. In the early Idpac versions there were many commands necessary to cover the available identifications methods. It was, however, discovered that almost all methods could be obtained by combinations of correlation analysis, spectral analysis, least squares and maximum likelihood estimation. Commands were thus constructed to give primitives for these operations and the spectral methods were then implemented as macros which used the primitive commands. This approach is also a pedagogical way to structure the problem area.

Idpac can be viewed as a convenient way of packaging the research in systems identification that has been done at our department for a period of 20 years. Idpac has gone through several steps of development. The relevant theoretical basis is given in [30] which also contains a comprehensive examples of using Idpac. A summary of the commands are given in Appendix C. A short description of some features of IDPAC is given below.

The program has facilities for input-output, editing and display of data. It includes several estimation procedures like correlation and spectral analysis, least squares and maximum likelihood estimation. It has facilities for simulation and model analysis. The program is command driven, which means that the user initiates the different operations by typing commands on a terminal. The program also has a MACRO facility, which means that a user can combine several commands. In this way it is possible both to have a large flexibility for the experienced user and to allow for a simple use of standardized procedures for an inexperienced user. An example of the use of the program is given below.

```
1. MOVE DK WORK ← WORK (2) 1
2. PLOT WORK
3. TREND ← WORK 1
4. ML PAR1 ← WORK 1
5. ML PAR2 ← WORK 2
6. ML PAR3 ← WORK 3
```

The first command simply moves the columns 1 and 3 on the data file DATA from magnetic tape to a work area on the disc. The second command plots the data on the graphical display. The third command removes a first order trend from the second column in the file WORK. The commands 4, 5 and 6 perform Maximum Likelihood estimation of the parameters in the model (2.2) for orders 1, 2 and 3 using the data in the file WORK. The estimated parameters are stored in the files PAR1, PAR2 and PAR3.

The analysis of the models can proceed as follows.

```
7. RESID RES ← PAR2 WRK 20
```

This means that the residuals of the model with parameters PAR2 are computed and stored in the file RES. In this computation the covariance function of the residuals and the cross covariance function between the input and the residuals are also computed and automatically displayed. The commands

```
8. DETER DET ← PAR2 WORK (1)
```

computes the deterministic output of the model with parameters PAR2 when the input is the process input WORK (1) and the disturbances neglected. The command

```
9. PLOT NL WORK (2) DET
```

finally plots the process output work (2) as separate points and the output of the simulated model.

Command-driven programs like IDPAC have several advantages. The commands can be read from a file on disc instead from the input terminal. By combining this with the macro facility it is easy to obtain new commands simply by combining already existing commands. In this way it is easy to generate commands for multistage least squares, extended least squares, by basic least squares command. IDPAC may be viewed as a special high level language for system identification.

The use of a macro will be demonstrated using an example. Assume that a transfer function model given by Eq. (3) has been estimated using a parameter estimation scheme which also estimates the parameter uncertainties. Since the transfer function parameters and their uncertainties do not give much physical insight it is useful to make a Monte Carlo simulation of the responses of a system whose parameters have a distribution with the estimated means and covariances. This is simple in principle byt tedious to program. Using the MACRO facility the problem is solved as follows.

```
1. MACRO MCSIM Y ← MOD U NL
2. FOR I = 1 TO NL
3. RANPA P ← MOD
4. DETER Y(I) ← P U
5. NEXT I
6. PLOT Y
7. END
```

This macro generates the new command

```
MCSIM Y ← MOD U NL
```

which performs NL number of Monte Carlo simulations of a system MOD having uncertain parameters. The input signal is *U* and the output signals are stored as columns in the file Y. The first line is simply the macro definition. Lines 2 and 5 control the iteration. The third line generates a parameter vector P by sampling a gaussian distribution whose mean value is the estimated parameters θ and whose covariance is the estimated covariances R. These are stored in the file MOD. The fourth line is a simple simulation command. It generates the output Y from a model with parameters P having the input U.

Having defined the Macro it can now be used as follows:

```
1. ML MODEL ← DATA 2 SA
2. SAVE COMAT
3. LET NPLX. = 100
4. INSI U NPLX
5. )PULSE
6. MCSIM Y ← MODEL U 6
```

The first command is an ML-command to generate a second order ML model from the measured data stored in the file DAT. The argument SA in the command 1 means that a special command is required. The second line specifies that the covariance matrix should be saved and stored in the file called MODEL. The third command defines that the variable NPLX should be given the value 100. A signal called U of length NPLX is defined in statement 4 and statement 5 specifies this signal to be a unit pulse. Command number 6 finally calls the macro command that was just generated. The curves shown in Fig. 4 are then displayed on the graphic screen.

The experiences with the interactive package IDPAC have been very good. The program has made it possible to analyse results from industrial experiments quickly and at a reasonable cost. The program has also been a very useful teaching aid. It has made it possible to teach system identification efficiently in a short time (about a week) both to students in the university and to engineers in industry. The program package is now being used by a number of industries.

8. CONCLUSIONS

Computer aided design of control systems is still in its early stages. There are a number of packages like those described in this paper. An overview of some packages are round in [31], [32], [33], [34], [35], [36], [37], [38] and [39]. More references are also found in these papers. Special workshops and symposia devoted to CAD for control systems have been organized by IFAC, GE-RPI, and IEEE CSS. See [40], [41], [42] and [43]. Computer aided tools are also popular in many other fields e.g. mechanical design and VLSI design. The seminal work on computer graphics [22] contain much material and many rererences.

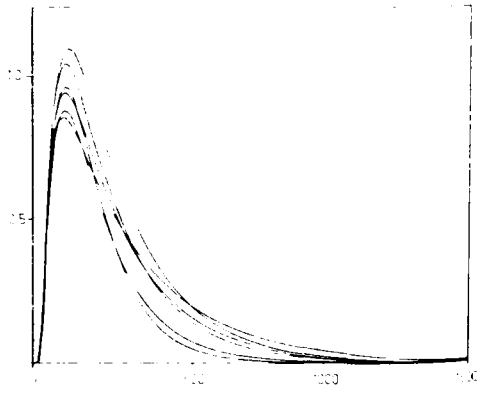


Fig. 4 Monte Carlo simulation of a model of ship dynamics. The curves show how the uncertainties in the parameters of a transfer function model are reflected in uncertainties in the impulse response.

The field is in a state of rapid development due to an increased understanding of the technology and the drastic development of computer and graphics hardware. It is safe to predict that future computer aided design tools will be much more powerful than the packages described in this paper. Some speculations on future development are given in [44].

Interactive computing is a powerful tool for problem solving. An engineer can come to the work station with a problem and he can leave with a complete solution after a few hours. The results are well documented in terms of listings, text and graphs. The problem solver can obtain the solution by himself without relying on programmers as intermediaries. Our projects have shown that the productivity in analysing and designing control systems can be increased substantially by using these tools. We believe that interactive computer aided design tools is one possibility to make modern control theory cost effective.

Computer aided design of control systems is still in its infancy. A small number of systems have been implemented in a few places. There are many possible future developments which are mainly driven by the computer development. Packages of the type we have been experimenting with can easily be fitted into the personal computers or work stations that will be available in a few years time. The bit mapped high resolution color displays that will be available on these computers offer new possibilities for an efficient man-machine dialog. With the drastic increase in computer capacity, that is forth coming, it is also possible to make much more ambitious projects. Applications of computer aided design also appear in many other branches of engineering. Cross fertilization between the fields will most likely lead to a rapid development.

The procedure for obtaining control law consists of the steps: experimental design, experiments, parameter estimation, control design and implementation. It may be a considerable effort to go through these steps. As an alternative we may consider adaptive control which may be viewed as an automation of the procedure.

9. REFERENCES

[1] K.J. Astrom and P. Eykhoff: System identification - a survey. 2nd IFAC Symposium on Identification and Process Parameter Estimation, Prague. Also in Automatica 7 (1971) 123-162.

[2] R.E. Nieman, D.G. Fisher and D.E. Seborg: A review of process identification and parameter estimation techniques. Int J Control 13 (1971) 209-264.

[3] P. Eykhoff: System identification - parameter and state estimation. Wiley, New York (1974).

[4] R. Isermann: Prozessidentifikation. Springer-Verlag, Berlin (1974).

[5] H. Unbehauen, B. Göhring and B. Bauer: Parameterschätzverfahren zur systemidentifikation. Oldenbourg, Munchen (1974).

[6] D. Graupe: Identification of systems. Krieger, New York (1976).

[7] D.R. Cox: Planning of experiments. Wiley, New York (1958).

[8] V.V. Fedorov: Theory of optimal experiments. Academic Press, New York (1972).

[9] R.K. Mehra: Optimal input signals for parameter estimation in dynamic systems - survey and new results. IEEE Trans AC-19 (1974) 753-768.

[10] C.G. Goodwin, M.B. Zarrop and R.L. Payne: Coupled design of test signals sampling interval, and filters for system identification. IEEE Trans AC-19 (1974) 748-752.

[11] I. Gustavsson, L. Ljung and T. Söderström: Identification of processes in closed loop - Identifiability and accuracy aspects. Preprints IV IFAC Symposium on Identification and System Parameter Estimation, Tiblisi (1976).

[12] B.T. Polyak and Ya.Z. Tsypkin: Robust (noiseproof) identification. Preprints of IV IFAC Symposium on Identification and System Parameter Estimation, Tiblisi (1976).

[13] K.F. Gauss: Theoria motus corporum coelestium. Göttingen 1809. English translation published by Dover.

[14] K.J. Aström and T. Bohlin: Numerical identification of linear dynamic systems from normal operating records. Second IFAC Symposium on the Theory of Self-Adaptive Control Systems (1965). Also in Hammond, P.H. (editor): Theory of self-adaptive control systems. Plenum Press (1966).

[15] A.V. Balakrishnan: Technique of system identification. Lecture notes, Institute Electrotecnio, University of Rome (1969).

[16] R.K. Mehra: Identification of stochastic linear systems. Proc IEEE Symp on Adaptive Processes, Pennsylvania State University (1969).

[17] R.L. Kashyap: A Bayesian comparison of different classes of dynamic models using empirical data. Report TR-EE 76-40 School of Electrical Engineering, Pardue University, West Lafayette, Indiana (1976).

[18] R. Isermann: Comparison of different identification methods. Fourth IFAC Symposium on Identification and System Parameter Estimation, Tiblisi (1976).

[19] H. Akaike: Use of an information theoretic quantity for stastical moden identification. Proc 5th Hawaii Intl Conf on System Science (1972) 249-250.

[20] H.H. Rosenbrock: Computer-aided control system design. Academic Press, New York (1974).

[21] J. Wieslander: Interaction in computer aided analysis and design of control systems. PhD thesis, Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN:LUTFD2/(TFRT-1019)/1-222/(1979).

[22] W.M. Newman and R.F. Sproull: Principles of interactive computer graphics. McGraw-Hill, New York (1979).

[23] J.D. Foley and A. van Dam: Fundamentals of interactive computer graphics. Addison Wesley, Reading, Mass. (1982).

[24] P. Wegner: Programming languages. Information structures and machine organization. McGraw-Hill, New York (1968).

[25] J. Wieslander and H. Elmqvist: INTRAC, a communication module for interactive programs. Language manual. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3149)/1-60/(1978).

[26] J. Wieslander: Interactive programs - General guide. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3156)/1-30/(1980).

[27] H. Elmqvist: SIMNON - An interactive simulation program for nonlinear systems. Simulation 77, Montreux, Switzerland (1977).

[28] K.J. Aström: A Simnon tutorial. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3168)/1-052/(1982).

[29] J. Wieslander: IDPAC commands - User's guide. Dept of Automatic Control, Lund Institute of Technology, Lund, Sweden, Report CODEN: LUTFD2/(TFRT-3157)/1-108/(1980).

[30] K.J. Åström: Maximum likelihood and prediction error methods. Automatica 16 551-574 (1980).

[31] D.P. Atherton: The role of CAD in education and research. IFAC Congress VIII, Kyoto, Japan (1981).

[32] T.F. Edgar: New results and the status of computer-aided process control system design in North America. Engineering Foundation Conference on Chemical Process Control-II, Sea Island, Georgia (1981).

[33] J.M. Edmunds: Cambridge linear analysis and design programs. IFAC Symposium on Computer Aides Design of Control Systems, Zurich, (1979) 253-258.

[34] D.K. Frederick: Computer packages for the simulation and design of control systems. Lecture notes. Arab school on science and technology (1982).

[35] I. Hashimoto and Y. Takamatsu: New results and the status of computer aided process control systems design in Japan. Engineering Foundation Conference on Chemical Process Control-II, Sea Island, Georgia (1981).

[36] W.J.M. Lemmens and A.J.W. Van den Boom: Interactive computer programs for education and research: a survey. Automatica 15 (1979) 113-121.

[37] N. Munro: The UMIST control system design and synthesis suites. IFAC Symposium on Computer Aided design of Control Systems, Zurich, (1979) 343-348.

[38] A. Tyssö: CYPROS: Cybernetic program packages. IFAC Symposium on Computer Aided Design of Control Systems, Zurich (1979) 383-389.

[39] J. Wieslander: Design principles for computer aided design software. Preprints, IFAC Symposium on CAD of Control Systems, Zurich (1979) 493.

[40] M. Mansour (ed): Preprints first IFAC Symposium on CAD of Control systems. Zurich. Pergamon (1979).

[41] G. Leininger (ed): Computer aided design of multivariable technological systems. Preprints second IFAC Symposium on Computer Aided Design of Multivariable Technological Systems. West Lafayette, Indiana, USA (1982).

[42] H.A. Spang III and L. Gerhart (eds): Preprints GE-RPI, Workshop on control design. Schenectady, N.Y. (1981).

[43] C.J. Herget and A.J. Laub (ed): Proc IEEE CSS Workshop on Computer Aided Control System Design. Berkeley, Calif. IEEE CSM 2-4. Special issue on Computer-Aided Design of Control Systems (1982).

APPENDIX A - Intrac commands

1. Input and output

READ	- Read string or variable from keyboard
SWITCH	- Utility command
WRITE	- Write string or variable on terminal

2. Assignment

DEFAULT	- Assign default values
FREE	- Release assigned global variables
LET	- Assignment of variables and global parameters
STOP	- Stop execution and return to OS

3. Control of program flow

FOR..TO	- Loop
NEXT V	
LABEL L	- Declaration of label
GOTO L	- Transfer control
IF..GOTO	- Transfer control

4. Macro
END - End of macro definition
FORMAL - Declaration of formal arguments
MACRO - Macro definition
RESUME - Resume execution of macro
SUSPEND - Suspend execution of macro

APPENDIX B - Simnon commands

1. Utilities
EDIT - Edit system description
GET - Get parameters and initial values
LIST - List files
PRINT - Print files
SAVE - Save parameter values and initial values in a file
STOP - Stop

2. Graphic output
AREA - Select window on screen
ASHOW - Plot stored variables with automatic scaling
AXES - Draw axes
HCOPY - Make hard copy
SHOW - Plot stored variables
SPLIT - Split screen into windows
TEXT - Transfer text string to graph

3. Simulation Commands
ALGOR - Select integration algorithm
DISP - Display parameters
ERROR - Choose error bound for integration routine
INIT - Change initial values of state variables
PAR - Change parameters
PLOT - Choose variables to be plotted
SIMU - Simulate a system
STORE - Choose variables to be stored
SYST - Activate systems

APPENDIX C - Idpac Commands

1. Utilities
CONV - Conversion of data to internal standard format
DELET - Delete a file
EDIT - Edit system description
FHEAD - Inspect and change file parameters
FORMAT - Conversion of data to symbolic external form
FTEST - Check existence of a file
LIST - List files
MOVE - Move data in database
TURN - Change program switches

2. Graphic output
BODE - Plot Bode diagrams
HCOPY - Make hard copy
PLMAG - Magnify plot and allow changes of data
PLOT - Plot curves with linear scales

3. Time series operations
ACOF - Compute autocorrelation function
CCOF - Compute cross-correlation function
CONC - Concatenate time series
CUT - Extract a part of a time series
INSI - Generate time series
PICK - Pick equidistant time points
SCLOP - Do scalar operations on a time series
SLIDE - Introduce relative delays between time series
STAT - Compute
TREND - Remove a trend
VECOP - Do vector operations on a time series

4. Frequency response operations
ASPEC - Compute an auto spectrum
CSPEC - Compute a cross spectrum
DFT - Discrete Fourier Transform
FROP - Operate on frequency responses
IDFT - Inverse Discrete Fourier Transform

5. Simulation and model analysis
DETER - Deterministic Simulation
DSIM - Simulation with noise
FILT - Compute a filter system
RANPA - Pick parameters from a random distribution
RESID - Compute residuals with statistical tests
SPTRF - Compute the frequency response of a transfer function

6. Identification
LS - Least Squares identification
ML - Maximum Likelihood identification
SQR - Least Squares data reduction
STRUC - Least Squares structure definition

NUMERICAL ASPECTS OF CONTROL DESIGN COMPUTATIONS

Professor Alan J. Laub
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089

SUMMARY

The interplay between recent results and methodologies in numerical linear algebra and mathematical software and their application to problems arising in systems, control, and estimation theory is discussed. The impact of finite precision, finite range arithmetic (including the implications of the proposed IEEE Floating Point Standard(s)) on control design computations is illustrated with numerous examples as are pertinent remarks concerning numerical stability and conditioning. Basic tools from numerical linear algebra such as linear equations, linear least squares, eigenproblems, generalized eigenproblems, and singular value decomposition are then outlined. A selected list of applications of the basic tools then follows including algorithms for solution of problems such as matrix exponentials, frequency response, system balancing, and matrix Riccati equations. The implementation of such algorithms as robust mathematical software is then discussed. A number of issues are addressed including characteristics of reliable mathematical software, availability and evaluation, language implications (Fortran, Ada, etc.), and the overall role of mathematical software as a component of computer-aided control system design.

1. INTRODUCTION

This paper provides an introduction to various aspects of the numerical solution of selected problems of interest in systems, control, and estimation theory. Space limitations preclude an exhaustive survey; rather, a compact "introduction to the literature" will lead the interested reader to sources of additional, detailed information.

Many of the problems considered here arise in the study of the "standard" linear model

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$

$$y(t) = Cx(t) + Du(t). \tag{2}$$

Here, $x(t)$ is an n-vector of states, $u(t)$ is an m-vector of controls or inputs, and $y(t)$ is an r-vector of outputs. The standard discrete-time analogue of (1), (2) takes the form

$$x_{k+1} = Ax_k + Bu_k \tag{3}$$

$$y_k = Cx_k + Du_k. \tag{4}$$

Of course, considerably more elaborate models are also studied, including time-varying, stochastic, and nonlinear versions of the above. In fact, the above linear models are usually derived as linearizations of nonlinear models about selected nominal points. The interested reader is referred to standard textbooks such as [1]-[4] for further details.

The matrices considered here will, for the most part, be assumed to have real coefficients and be small (of order a few hundred or less) and dense with no particular exploitable structure. Calculations for most problems in classical single-input, single-output control fall into this category. It must be emphasized that consideration of large, sparse matrices or matrices with special, exploitable structure may involve significantly different concerns and methodologies than those to be discussed here.

The systems, control, and estimation literature is replete with ad hoc algorithms to solve the computational problems which arise in the various methodologies. Many of these algorithms work quite well on some problems (e.g., "small order" matrices) but encounter numerical difficulties, often severe, when "pushed" (e.g., on larger order matrices). The reason for this is that little or no attention has been paid to how the algorithms will perform in "finite arithmetic", i.e., on a finite-word-length digital computer.

A simple example due to Moler and Van Loan [5] will illustrate a typical pitfall. Suppose it is desired to compute the matrix e^A in single precision arithmetic on an IBM 370 computer. In this particular computing environment we have, roughly speaking, about 6 decimal places of precision in the fraction part of floating point numbers. Consider the case $A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix}$ and suppose the computation is attempted using the formula

$$e^A = \sum_{k=0}^{+\infty} \frac{1}{k!} A^k. \tag{5}$$

This is easily coded and it is determined that the first 60 terms in the series suffice for the computation, in the sense that terms for $k \geq 60$ are $O(10^{-7})$ and no longer add anything significant to the sum. The resulting answer is

$$\begin{pmatrix} -22.2588 & -1.43277 \\ -61.4993 & -3.47428 \end{pmatrix}.$$

Unfortunately, the true answer is (correctly rounded)

$$\begin{pmatrix} -0.735759 & 0.551819 \\ -1.47152 & 1.10364 \end{pmatrix}$$

and one sees a rather alarming disparity. What happened here was that the intermediate terms in the series got very large before the factorial began to dominate. In fact, the 17th and 18th terms, for example, are of the order of 10^7 but of opposite signs so that the less significant parts of these numbers – while significant for the final answer – are "lost" because of the finiteness of the arithmetic.

Now for this particular example various fixes and remedies are available. But in more realistic examples one seldom has the luxury of having the "true answer" available so that is is not always easy to simply inspect or test an answer such as the one obtained above and determine it to be in error. Mathematical analysis (truncation of the series, in the example above) alone is simply not sufficient when a problem is analyzed or solved in finite arithmetic (truncation of the arithmetic). Clearly, a great deal of care must be taken.

The finiteness inherent in representing real or complex numbers as floating-point numbers on a digital computer manifests itself in two important ways: floating point numbers have only finite precision and finite range. In fact, it is the degree of attention paid to these two considerations that distinguishes many reliable algorithms from more unreliable counterparts. Reference [6] still provides the definitive introduction to the vagaries of floating-point computation while [7] and the references therein may be consulted to bring the interested reader up to date on roundoff analysis.

An even more recent development in floating-point arithmetic has been the work of the Floating-Point Working Group of the Microprocessor Standards Subcommittee of the IEEE Computer Society Standards Committee. An early draft (Draft 8.0) of the IEEE Standard 754 for Binary Floating-Point Arithmetic appears in [8] along with several related articles. Draft 10.0 of P754 was sent in December 1982 to the appropriate sponsoring committee for balloting at the next level (a successful vote would send the draft to the IEEE standards board). A parallel effort (Task 854) is also under way to draft a Radix-Free Standard. These standards define families of commercially feasible ways for new systems (originally microprocessor but also now minicomputers and large mainframes) to perform binary floating-point arithmetic in a numerically "sensible" way. The adoption of these Standards will have a major impact on algorithmic development and software. In fact, early versions of the Binary Standard are already available for certain microprocessor systems (e.g., the Intel 8087) which will come into ever-increasing use in control and estimation in the 1980's and 1990's.

The development, in systems, control, and estimation theory, of stable, efficient, and reliable algorithms which respect the constraints of finite arithmetic is only now in its infancy. Much of current research in numerical analysis is directly applicable but there are many computational issues in control (e.g., the presence of hard or structural zeros) where numerical analysis does not yet provide a ready answer or guide. A symbiotic relationship has already developed which is sure to provide a continuing source of challenging research areas.

The abundance of numerically fragile algorithms is partly explained by the following observation which will be emphasized by calling it a *"folk theorem"*:

> If an algorithm is amenable to "easy" hand calculation, it's probably a poor method if implemented in the finite floating-point arithmetic of a digital computer.

For example, when confronted with finding the eigenvalues of a 2 x 2 matrix most people would find the characteristic polynomial and solve the resulting quadratic equation. But when extrapolated as a general method and implemented on a digital computer this turns out to be a very poor procedure indeed for a variety of reasons (such as roundoff and overflow/underflow). Of course the preferred method now would generally be the double Francis QR algorithm (see [9], [10] for the messy details) but few of us would attempt that by hand -- even for very small order problems.

In fact, it turns out that many algorithms which are now considered fairly reliable in the context of finite arithmetic are not amenable to hand calculation (e.g., various classes of orthogonal similarities). This is sort of a converse to the folk theorem. Particularly in linear control and systems theories, we have been too easily seduced by the ready availability of closed-form solutions and numerically naive methods to implement those solutions. For example, in solving the initial value problem

$$\dot{x}(t) = Ax(t) \; ; \; x(0) = x_0 \tag{6}$$

it is not at all clear that one should want to compute the intermediate quantity e^{tA}. Rather, it is the vector $e^{tA}x_0$ that is desired, a quantity that may be computed more reasonably by treating (6) as a system of (stiff) differential equations and using, say, an implicit method for numerical integration of the differential equation. But such techniques are definitely not attractive for hand computation.

Remedying the present situation is largely a matter of awareness and education. While it is a slow process, we are now just beginning to see some of the background material

(well-known to numerical analysts) mentioned in this paper filter down to the undergraduate and graduate curriculum in mathematics and engineering. Introductory textbooks such as [11]-[13] are now also reflecting a strong software component. This process is certain to have a significant impact on the future directions and development of control and systems theory and applications as witness the growth of CACSD as an intrinsic tool. Algorithms implemented as mathematical software are a critical "inner" component of a CACSD system and the remainder of this paper will address some of the issues involved.

Before proceeding further we shall list here some notation to be used in the sequel.

$\mathbb{F}^{nxm}$ the set of all nxm matrices with coefficients in the field $\mathbb{F}$ ($\mathbb{F}$ will generally be $\mathbb{R}$ or $\mathbb{C}$)

$\mathbb{F}_r^{nxm}$ the set of all nxm matrices of rank r with coefficients in the field $\mathbb{F}$

A^T the transpose of $A \in \mathbb{R}^{nxm}$

A^H the complex-conjugate transpose of $A \in \mathbb{C}^{nxm}$

A^+ the Moore-Penrose pseudoinverse of A

$\|A\|$ the spectral norm of A (i.e., the matrix norm subordinate to the Euclidean vector norm: $\|A\| = \max_{\|x\|_2=1} \|Ax\|_2$)

$\mathrm{diag}(a_1,\ldots,a_n)$ the diagonal matrix $\begin{pmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_n \end{pmatrix}$

$\Lambda(A)$ the set of eigenvalues $\lambda_1,\ldots,\lambda_n$ (not necessarily distinct) of $A \in \mathbb{F}^{nxn}$

$\lambda_i(A)$ the i^{th} eigenvalue of A

$\Sigma(A)$ the set of singular values $\sigma_1,\ldots,\sigma_m$ (not necessarily distinct) of $A \in \mathbb{F}^{nxm}$

$\sigma_i(A)$ the i^{th} singular value of A.

Finally, let us define a particular number to which we shall make frequent reference in the sequel. The *machine epsilon* or *machine precision* can be defined, roughly speaking, as the smallest positive number ϵ which, when added to 1 on our computing machine, gives a number greater than 1. In other words, any machine representable number δ less than ϵ gets "rounded off" when (floating-point) added to 1 to give exactly 1 again as the rounded sum. The number ϵ varies, of course, depending on the kind of computer being used and the precision with which the computations are being done (single precision, double precision, etc.). But the fact that there exists such a positive number ϵ is entirely a consequence of finite word length.

2. NUMERICAL STABILITY AND CONDITIONING

In this section we give a very brief discussion of two concepts of fundamental importance in numerical analysis: numerical stability and conditioning. While this material is standard in introductory textbooks such as [14]-[17] it is presented here both for completeness and because the two concepts are frequently confused in the systems/control/estimation literature.

Suppose we have some mathematically defined problem represented by f which acts on data $d \in \mathcal{D}$ = some set of data, to produce a solution $f(d) \in \mathcal{S}$ = some set of solutions. These notions are kept deliberately vague for expository purposes. Given $d \in \mathcal{D}$ we desire to compute f(d). Suppose d* is some approximation to d. If f(d*) is "near" f(d) the problem is said to be *well-conditioned*. If f(d*) may potentially differ greatly from f(d) even when d* is near d, the problem is said to be *ill-conditioned*. Again, the concept "near" cannot be made precise without further information about a particular problem.

A simple example of an ill-conditioned problem is the following. Consider the n x n matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1 \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

with n eigenvalues at 0. Now consider a small perturbation of the data (the n^2 elements of A) consisting of adding the number 2^{-n} to the first element in the last (n^{th}) row of A. This perturbed matrix then has n distinct eigenvalues $\lambda_1,\ldots,\lambda_n$ with $\lambda_k = \frac{1}{2}\exp(\frac{2k\pi i}{n})$. Thus we see that this small perturbation in the data has been magnified by a factor on the order of 2^n to result in a rather large perturbation in the solution (the eigenvalues of A). Further details and related examples are to be found in [9].

Note that we have so far made no mention of how the problem f above (computing $\Lambda(A)$ in the example) was to be solved. Conditioning was a function solely of the problem itself. To solve a problem numerically we typically must implement some numerical pro-

cedure or algorithm which we shall denote by f*. Thus, given d, f*(d) represents the result of applying the algorithm to d (for simplicity, we assume d is "representable"; a more general definition can be given when some approximation d** to d is used). The algorithm f* is said to be *numerically stable* if, for all $d \in \mathcal{D}$, there exists $d^* \in \mathcal{D}$ near d such that f*(d) is near f(d*) (= the exact solution of a nearby problem). If the problem is well-conditioned then f(d*) will be near f(d) so that f*(d) will be near f(d). In other words, f* does not introduce any more sensitivity to perturbation than is inherent in the problem. Example 1. below will further illuminate this definition of stability which, on a first reading, can seem somewhat confusing.

Of course, one can't expect a stable algorithm to solve an ill-conditioned problem any more accurately than the data warrant but an unstable algorithm can produce poor solutions even to well-conditioned problems. Example 2. below will illustrate this phenomenon. There are thus two separate factors to consider in determining the accuracy of a computed solution f*(d). First, if the algorithm is stable, f*(d) is near f(d*) and second, if the problem is well-conditioned then, as above, f(d*) is near f(d). Thus f*(d) is near f(d) and we have an "accurate" solution.

Roundoff errors can cause unstable algorithms to give disastrous results. However, it would be virtually impossible to account for every roundoff error made at every arithmetic operation in a complex series of calculations such as those involved in most linear algebra calculations. This would constitute a forward error analysis. As a more practical alternative, J.H. Wilkinson and others have advanced the notion of *backward error analysis* to account for roundoff error. Specifically, for many problems (particularly in numerical linear algebra), it is possible to show that what is actually computed is near the exact solution of a nearby problem. One then attempts to show that the nearby problem is near enough which, if the problem is well-conditioned, can be translated into a quantitative statement regarding the accuracy of the solution. Examples of this will be quoted in later sections.

We close this section with two simple examples to illustrate some of the concepts introduced above.

Example 1: Let x and y be two floating-point numbers and let $fl(x*y)$ denote the result of multiplying them in floating-point arithmetic. In general, the product x*y will require more precision to be represented exactly then was necessary to represent x or y. But what can be shown for most realistic models of floating-point computation is that

$$fl(x*y) = x*y\ (1+\delta) \tag{7}$$

where $|\delta| < \epsilon$. In other words, $fl(x*y)$ is x*y correct to within a unit in the last place. Now, another way to write (7) is as

$$fl(x*y) = x(1+\delta)^{\frac{1}{2}} * y(1+\delta)^{\frac{1}{2}} \tag{8}$$

where $|\delta| < \epsilon$. This can be interpreted as follows: the computed result $fl(x*y)$ is the *exact* product of the two slightly perturbed numbers $x(1+\delta)^{\frac{1}{2}}$ and $y(1+\delta)^{\frac{1}{2}}$. Note that the slightly perturbed data (not unique) may not even be representable floating-point numbers. The representation (8) is simply a way of accounting for the roundoff incurred in the algorithm by an initial (small) perturbation in the data.

Example 2: Gaussian elimination with no pivoting for solving the linear system

$$Ax = b \tag{9}$$

is known to be numerically unstable. The following data will illustrate this phenomenon. Let $A = \begin{pmatrix} 0.0001 & 0.2345 \\ 0.6789 & 0.1000 \end{pmatrix}$, $b = \begin{pmatrix} 0.2346 \\ 0.7789 \end{pmatrix}$. All computations will be carried out in 4-decimal place arithmetic. The "true answer" $x = A^{-1}b$ is easily seen to be $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Using row 1 as the "pivot row" (i.e., subtracting 6789 x row 1 from row 2) we arrive at the equivalent triangular system

$$\begin{pmatrix} 0.0001 & 0.2345 \\ 0 & -1592. \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.2346 \\ -1591. \end{pmatrix}$$

From the equation

$$-1592x_2 = -1591$$

we find $x_2 = 0.9994$ and from the equation

$$0.0001x_1 = 0.2346 - fl(0.2345 * 0.9994)$$

$$= 0.0002$$

we find $x_1 = 2$. This extremely bad approximation to x_1 is the result of numerical instability. The problem itself can be shown to be reasonably well-conditioned.

3. FUNDAMENTAL PROBLEMS IN NUMERICAL LINEAR ALGEBRA

In this section we give a brief overview of some of the fundamental problems in numerical linear algebra which serve as building blocks or "tools" for the solution of pro-

blems in systems, control, and estimation.

3.1 Linear Algebraic Equations and Linear Least Squares Problems

Probably the most fundamental problem in numerical computing is the calculation of a vector x which satisfies the linear system

$$Ax = b \tag{10}$$

where $A \in \mathbb{R}_n^{nxn}$ (or $\mathbb{C}_n^{nxn}$). A great deal is now known about solving (10) in finite arithmetic both for the general case and for a large number of special situations. Some of the standard references include [14], [18]-[20].

The most commonly used algorithm for solving (10) with general A and small n (say $n \leq 200$) is Gaussian elimination with some sort of pivoting strategy, usually "partial pivoting." This essentially amounts to factoring some permutation of the rows of A into the product of a unit lower triangular matrix L and an upper triangular matrix U. The algorithm is effectively stable, i.e., it can be proved that the computed solution is near the exact solution of the system

$$(A + E)x = b \tag{11}$$

with $|e_{ij}| \leq \phi(n)\cdot\gamma\cdot\beta\cdot\varepsilon$ where $\phi(n)$ is a modest function of n depending on details of the arithmetic used, γ is a "growth factor" (which is a function of the pivoting strategy and is usually - but not always - small), β behaves essentially like $\|A\|$, and ε is the machine precision. See [14] for further details. In other words, except for moderately pathological situations, E is "small" - on the order of the machine precision.

The following question then arises. If, because of roundoff errors, we are effectively solving (11) rather than (10), what is the relationship between $(A+E)^{-1}b$ and $A^{-1}b$? To answer this question we need some elementary perturbation theory and this is where the notion of condition number arises. A condition number for the problem (10) is given by

$$\kappa(A) := \|A\| \, \|A^{-1}\|. \tag{12}$$

Simple perturbation results can be used to show that perturbations in A and/or b can be magnified by as much as $\kappa(A)$ in the computed solution. Estimation of $\kappa(A)$ (since, of course, A^{-1} is unknown) is thus a crucial aspect of assessing solutions of (10) and the particular estimation procedure used is usually the principal difference between competing linear equation software packages. One of the more sophisticated and reliable condition estimators presently available is based on [21] and is implemented in LINPACK [19]. In addition to the ℓ_1 condition estimator of [21], LINPACK features many codes for solving (10) in case A has certain special structures.

Another important class of linear algebra problems and one for which codes are available in LINPACK is the linear least squares problem:

$$\min\|Ax-b\|_2 \tag{13}$$

where $A \in \mathbb{R}_k^{mxn}$ with (in the simplest case) $k = n \leq m$. The solution of (13) can be written formally as $x = A^{+}b$. Here, standard references include [14], [19], [22]. The method of choice is generally based upon the QR factorization of A: (for simplicity, $A \in \mathbb{R}_n^{mxn}$)

$$A = QR \tag{14}$$

where $R \in \mathbb{R}_n^{nxn}$ is upper triangular and $Q \in \mathbb{R}^{mxn}$ has orthonormal columns, i.e., $Q^TQ = I$. With special care and analysis the case $k < n$ can also be handled similarly. The factorization is effected through a sequence of Householder transformations H_i applied to A. Each H_i is symmetric and orthogonal and of the form $I - \frac{2uu^T}{u^Tu}$ where $u \in \mathbb{R}^m$ is specially chosen to introduce zeros at appropriate places when H_i premultiplies A. After n such transformations we have

$$H_nH_{n-1}\cdots H_1A = \begin{pmatrix} R \\ 0 \end{pmatrix}$$

from which the factorization (14) follows. Defining c and d by

$$H_nH_{n-1}\cdots H_1b =: \begin{pmatrix} c \\ d \end{pmatrix}$$

where $c \in \mathbb{R}^n$, it is easily shown that the least squares solution x of (13) is given by the solution of the linear system

$$Rx = c. \tag{15}$$

The above algorithm can be shown to be numerically stable and, again, a well-developed perturbation theory exists from which condition numbers can be obtained, this time in terms of

$$\kappa(A) := \|A\| \, \|A^{+}\|.$$

Least squares perturbation theory is fairly straightforward in case $A \in \mathbb{R}_n^{mxn}$ but is

considerably more complicated when A is rank-deficient. The reason for this is that while the inverse is a continuous function of the data, the pseudoinverse is discontinuous. For example, consider $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = A^+$ and perturbations $E_1 = \begin{pmatrix} 0 & 0 \\ \delta & 0 \end{pmatrix}$, $E_2 = \begin{pmatrix} 0 & 0 \\ 0 & \delta \end{pmatrix}$ with δ small. Then $(A + E_1)^+ = \begin{pmatrix} \frac{1}{1+\delta^2} & \frac{\delta}{1+\delta^2} \\ 0 & 0 \end{pmatrix}$ which is close to A^+ but $(A + E_2)^+ = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\delta} \end{pmatrix}$ which gets arbitrarily far from A^+ as δ is decreased towards 0. For a complete survey of perturbation theory for the least squares problem and related questions see [23].

In lieu of Householder transformations, Givens transformations may also be used to solve the linear least squares problem. Details can be found in [9], [19], [22], [24], [25]. Recently, Givens transformations have received considerable attention for the solution of both linear least squares problems as well as systems of linear equations in a parallel computing environment. The capability of introducing zero elements selectively and the need for only local interprocessor communication make the technique ideal for "parallelization." Indeed, there have been literally dozens of "parallel Givens" algorithms proposed and we include [26]-[30] as representative references.

3.2 Eigenvalue and Generalized Eigenvalue Problems

In the algebraic eigenvalue/eigenvector problem for $A \in \mathbb{R}^{n \times n}$ one seeks nonzero solutions $x \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$ which satisfy

$$Ax = \lambda x. \tag{16}$$

The classic reference on the numerical aspects of this problem is Wilkinson [9] with Parlett [25] providing an equally thorough and up-to-date treatment of the case of symmetric A (in which $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$). A more brief textbook introduction is given in [14].

It is really only rather recently that some of the computational issues associated with solving (16) - in the presence of rounding error - have been resolved or even understood. Even now some problems such as the invariant subspace problem continue to be active research areas. For an introduction to some of the difficulties which may be encountered in trying to make numerical sense out of mathematical constructions such as the Jordan canonical form the reader is urged to consult [31].

The most common algorithm now used to solve (16) for general A is the QR algorithm of Francis [32]. A shifting procedure is used to enhance convergence and the usual implementation is called the double-Francis-QR algorithm. Before the QR process is applied, A is initially reduced to upper Hessenberg form A_H ($a_{ij} = 0$ if $i-j \geq 2$) [33]. This is accomplished by a finite sequence of similarities which can be chosen to be of the Householder form discussed above. The QR process then yields a sequence of matrices which are orthogonally (again, of Householder type) similar to A and which converge (in some sense) to a so-called quasi-upper triangular matrix S. The matrix S is block upper triangular with 1x1 blocks corresponding to real eigenvalues of A and 2x2 blocks corresponding to complex-conjugate pairs of eigenvalues. The orthogonal transformations from both the Hessenberg reduction and the QR process may be accumulated into a single orthogonal transformation U so that

$$U^T A U = S \tag{17}$$

compactly represents the entire algorithm.

An analogous process can be applied in the case of symmetric A and considerable simplifications and specializations result. Moreover, references [9] and [25] may be consulted regarding an immense literature concerning stability of the QR and related algorithms and conditioning of eigenvalues and eigenvectors. Both subjects are vastly more complex for the eigenvalue/eigenvector problem than for the linear equation problem.

Quality mathematical software for eigenvalues and eigenvectors has been only recently available - in the 1970's - and the EISPACK [10], [34] collection of subroutines represents a pivotal point in the history of mathematical software. This collection is primarily based on the algorithms collected in [35].

Closely related to the QR algorithm is the QZ algorithm [36] for the generalized eigenvalue problem

$$Ax = \lambda Mx \tag{18}$$

where $A, M \in \mathbb{R}^{n \times n}$. Again, a Hessenberg-like reduction, then an iterative process are implemented with orthogonal (Householder) transformations to reduce (18) to the form

$$QAZy = \lambda QMZy \tag{19}$$

where QAZ is quasi-upper-triangular and QMZ is upper triangular. For a review and references to results on stability, conditioning, and software related to (18) and the QZ algorithm see [37]. The generalized eigenvalue problem is both theoretically and numerically more difficult to handle than the ordinary eigenvalue problem but it finds numerous appli-

cations in control and systems theory.

3.3 The Singular Value Decomposition and Some Applications

One of the basic and important tools of modern numerical analysis, particularly numerical linear algebra, is the singular value decompositon. We shall define it here and make a few comments about its properties and computation as well as its significance in various numerical problems.

Singular values and the singular value decomposition have a long history particularly in statistics and more recently in numerical linear algebra. Even more recently the ideas are finding applications in the control and signal processing literature, although their use there has been overstated somewhat in certain applications. For a survey of the singular value decomposition, its history, numerical details, and some applications in control and systems theory, see [38].

The fundamental result can be stated as follows for the real case. For complex matrices the result is virtually identical with complex-conjugate transposes replacing transposes and unitary matrices replacing orthogonal matrices.

Theorem 1: Let $A \in \mathbb{R}^{m\times n}$. Then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that

$$A = U\Sigma V^T \tag{20}$$

where $\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}$ and $S = \mathrm{diag}(\sigma_1,\ldots,\sigma_r)$ with $\sigma_1 \geq \ldots \geq \sigma_r > 0$.

The proof of Theorem 1 is straightforward and can be found in, for example, [14] and [39]. Geometrically, the theorem says that bases can be found (separately) in the domain and co-domain spaces of a linear map with respect to which the matrix representation of the linear map is diagonal.

The numbers $\sigma_1,\ldots,\sigma_r$ together with $\sigma_{r+1} = 0,\ldots, \sigma_n = 0$ are called the *singular values* of A and they are the positive square roots of the eigenvalues of A^TA. The columns $\{u_k, k = 1,\ldots,n\}$ of U are called the *left singular vectors* of A (the orthonormal eigenvectors of AA^T) while the columns $\{v_k, k = 1,\ldots,n\}$ of V are called the *right singular vectors* of A (the orthonormal eigenvectors of A^TA). The matrix A can then also be written (as a dyadic expansion) in terms of the singular vectors as follows:

$$A = \sum_{k=1}^{r} \sigma_k u_k v_k^T.$$

The matrix A^T has m singular values, the positive square roots of the eigenvalues of AA^T. The r (=rank (A)) nonzero singular values of A and A^T are, of course, the same. The choice of A^TA rather than AA^T in the definition of singular values is arbitrary. Only the nonzero singular values are usually of any real interest and their number, given the SVD, is the rank of the matrix. Naturally, the question of how to distinguish nonzero from zero singular values in the presence of rounding error is a nontrivial task.

It is not generally advisable to compute the singular values of A by first finding the eigenvalues of A^TA (remember the folk theorem!), tempting as that is. Consider the following example with μ a real number with $|\mu| < \sqrt{\epsilon}$ (so that $fl(1+\mu^2) = 1$ where $fl(\cdot)$ denotes floating point computation). Let $A = \begin{pmatrix} 1 & 1 \\ \mu & 0 \\ 0 & \mu \end{pmatrix}$. Then $fl(A^TA) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ so we compute $\hat{\sigma}_1 = \sqrt{2}$, $\hat{\sigma}_2 = 0$ leading to the (erroneous) conclusion that the rank of A is 1. Of course, if we could compute in infinite precision we would find $A^TA = \begin{pmatrix} 1+\mu^2 & 1 \\ 1 & 1+\mu^2 \end{pmatrix}$ with $\sigma_1 = \sqrt{2+\mu^2}$, $\sigma_2 = |\mu|$ and thus rank (A) = 2. The point is that by working with A^TA we have unnecessarily introduced μ^2 into the computations. The above example illustrates a potential pitfall in attempting to form and solve the normal equations in a linear least squares problem and is at the heart of what makes square root filtering so attractive numerically. See [40] for further details and references.

Square root filtering is usually implemented using QR factorization (or some closely related algorithm) as described previously rather than SVD. The key thing to remember is that in most current computing environments the condition of the least squares problem is squared, unnecessarily, in solving the normal equations and, moreover, critical information may be lost, irrecoverably, by simply forming A^TA. These caveats may not be of such great concern, however, if one has available certain computing environments which implement, for example, IEEE arithmetic with extended length registers (e.g., the Intel 8087 floating-point processor chip).

Returning now to the SVD there are two features of this matrix factorization that make it so attractive in finite arithmetic: it can be computed stably and singular values are

well-conditioned. Specifically, there is an efficient and numerically stable algorithm due to Golub and Reinsch [41] (based on [39]) which works directly on A to give the SVD. The computed U and V are orthogonal to approximately the working precision and the computed singular values can be shown to be the exact σ_i's for A+E where $\frac{\|E\|}{\|A\|}$ is a modest multiple of ϵ. Fairly sophisticated implementations of this algorithm can be found in [19] and [34]. The well-conditioned nature of the singular values follows from the fact that if A is perturbed to A + E then

$$|\sigma_i(A + E) - \sigma_i(A)| \leq \|E\|.$$

Thus the singular values are computed with small error although the relative errors of sufficiently small singular values is not guaranteed small.

It is now acknowledged that the singular value decomposition is the most generally reliable method of determining rank numerically (see [31] for a more elaborate discussion). However, it is considerably more expensive to compute than, for example, the QR factorization which, with column pivoting [19], can usually give equivalent information with less computation. Thus, while the SVD is a useful theoretical tool its use for actual computations should be weighed carefully against other approaches.

Only rather recently has the problem of numerical determination of rank been well-understood. One of the best treatments of the subject, including a careful definition of numerical rank, is a paper by Golub, Klema, and Stewart [42]. The essential idea is to try to determine a "gap" between "zero" and the "smallest nonzero singular value" of a matrix A. Since the computed values are exact for a matrix near A it makes sense to consider the rank of all matrices in some δ-ball (w.r.t. the spectral norm $\|\cdot\|$, say) around A. The choice of δ may also be based on measurement errors incurred in estimating the coefficients of A or the coefficients may be uncertain because of roundoff errors incurred in a previous computation to get them. We refer to [42] for further details. We must emphasize, however, that even with SVD, numerical determination of rank in finite arithmetic is a highly nontrivial problem.

That other methods of rank determination are potentially unreliable is demonstrated by the following example which is a special case of a general class of matrices studied by Ostrowski [43]. Consider the matrix $A \in \mathbb{R}^{n\times n}$ whose diagonal elements are all -1, whose upper triangle elements are all +1, and whose lower triangle elements are all 0. This matrix is clearly of rank n, i.e., is invertible. It has a good "solid" upper triangular shape. All of its eigenvalues (all = -1) are well away from zero. Its determinant is $(-1)^n$ -- definitely not close to zero. But this matrix is, in fact, very near singular and gets more nearly so as n increases. Notice, for example, that

$$\begin{pmatrix} -1 & +1 & \cdots & +1 \\ & \ddots & \ddots & \vdots \\ & & \ddots & +1 \\ 0 & & & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 2^{-1} \\ \vdots \\ 2^{-n+1} \end{pmatrix} = \begin{pmatrix} -2^{-n+1} \\ -2^{-n+1} \\ \vdots \\ -2^{-n+1} \end{pmatrix} \to \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad (n \to +\infty).$$

Moreover, adding 2^{-n+1} to every element in the first column of A gives an exactly singular matrix. Arriving at such a matrix by, say Gaussian elimination, would give no hint as to the near-singularity. However, it is easy to check that $\sigma_n(A)$ behaves as 2^{-n+1}. A corollary for control theory: eigenvalues don't necessarily give a reliable measure of "stability margin." As an aside it is useful to note here that in this example of an invertible matrix, the crucial quantity, $\sigma_n(A)$, which measures nearness to singularity, is simply $\frac{1}{\|A^{-1}\|}$ and the result is familiar from standard operator theory. There is nothing intrinsic about singular values in this example and, in fact, $\|A^{-1}\|$ might be more cheaply computed or estimated in another matrix norm. This is precisely what is done in estimating the condition of linear systems in LINPACK where $\|\cdot\|_1$ is used [21].

Since rank determination, in the presence of roundoff error, is a nontrivial problem, all the same difficulties will naturally arise in any problem equivalent to or involving rank determination such as determining the independence of vectors, finding subspace bases, etc. Such problems arise as basic calculations throughout systems, control, and estimation theory. Selected applications are discussed in more detail in [38].

Finally, let us close this section with a brief example illustrating a totally inappropriate use of SVD. The rank condition

$$\text{rank}[B, AB, \ldots, A^{n-1}B] = n \tag{21}$$

for the controllability of (1) is (too) well-known. Suppose $A = \begin{pmatrix} 1 & \mu \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 1 \\ \mu \end{pmatrix}$ with $|\mu| < \sqrt{\epsilon}$. Then

$$f^?[B, AB] = \begin{pmatrix} 1 & 1 \\ \mu & \mu \end{pmatrix}$$

and now even applying SVD the erroneous conclusion of uncontrollability is reached. Again

the problem is in just forming AB; not even SVD can come to the rescue after that numerical faux pas.

4. APPLICATIONS

In this section we shall present a representative selection of numerical problems which arise in linear systems, control, and estimation theory and which have been examined using some of the techniques described in Sections 2 and 3. Some of the topics are described in more detail in [38] and [44] while still other topics are surveyed in [45].

4.1 Numerical Solution of Linear Ordinary Differential Equations

The "simulation" or numerical solution of linear systems of ordinary differential equations (ODE's) of the form

$$\dot{x}(t) = Ax(t) + f(t);\ x(0) = x_0 \tag{22}$$

is a standard problem. However, there is still debate as to what is the most effective numerical algorithm, particularly when A is defective (a deficiency of eigenvectors) or near-defective. The most common approach involves computation of the matrix exponential, e^{tA}. A delightful survey of this topic is given in [5]. Nineteen "dubious" ways are explored (there exist many more ways which are not discussed) but no clearly superior algorithm is singled out. Methods based on Padé approximation or reduction of A to real Schur form are seen as generally attractive while methods based on Taylor series or the characteristic polynomial of A are generally found to be unattractive. An interesting open problem is the design of a special algorithm for the matrix exponential when the matrix is known a priori to be stable ($\Lambda(A)$ in LHP).

The reason for the adjective "dubious" in the title of [5] is that in many (maybe even most) circumstances, it is better to treat (22) as a system of differential equations, typically stiff, and to apply various ODE techniques, specially tailored to the linear case. This approach is discussed in [46]. ODE techniques are to be preferred when A is large and sparse for, in general, e^{tA} will be unmanageably large and dense. The relationship between ODE techniques and matrix exponential techniques when A has an ill-conditioned eigenstructure or when the "exponential problem" is ill-conditioned [47], [48] is not well-understood.

4.2 Controllability and Other "Abilities"

Basic to the study of linear control and systems theory are the various "abilities" such as controllability, observability, reachability, reconstructibility, stabilizability, and detectability [1]. Our remarks here will be confined, but are not limited, to the notion of controllability.

A large number of algebraic and dynamic characterizations of controllability have been given; see [44] for a sample. But each and every one of these has difficulties when implemented in finite arithmetic. For a survey of this topic and numerous examples see [49]. Part of the difficulty in dealing with controllability numerically lies in the intimate relationship with the invariant subspace problem [31]. The controllable subspace associated with (1) is the smallest A-invariant subspace containing the range of B. Since A-invariant subspaces are extremely sensitive to perturbation, it follows that so too is the controllable subspace. Similar remarks apply to the computation of the so-called controllability indices.

Recently, attempts have been made to provide numerically stable algorithms for the pole placement problem; we would cite [50]-[53] as examples (only one representative and recent reference is chosen for each author or group). The methods are based on reduction of A to a Hessenberg form rather than a controllable or Luenberger canonical form which is known to be numerically unstable [9]. For example, in the single-input, single-output case it can easily be shown, using the tools developed in Section 3.1, that there exists an orthogonal transformation U such that U^TAU is upper Hessenberg and U^TB = a multiple of $(1,0,\ldots,0)^T$. The pair (A,B) is then controllable if and only if all (n-1) subdiagonal elements of U^TAU are nonzero. If a subdiagonal element is 0, the system is uncontrollable and a basis for the uncontrollable subspace is easily constructed. The transfer function gain or first nonzero Markov parameter is also easily constructed from this "canonical form." In fact, the numerically more robust general Hessenberg form will probably play an ever-increasing role in systems theory in replacing the numerically more fragile special case of the companion or rational canonical or Luenberger canonical form.

A more important aspect of controllability is to better understand topological notions such as "near-uncontrollability." But there are numerical difficulties lurking here, also, and we refer to [49] for further details. Related to this is an interesting new system-theoretic concept called "balancing"; see [54]. The computation of "balancing transformations" is discussed in [55].

There are at least two distinct notions of near-uncontrollability: in the parametric sense and in the energy sense. In the parametric sense a controllable pair (A,B) is said to be near-uncontrollable if the parameters of (A,B) need be perturbed by only a relatively small amount to become uncontrollable. In the energy sense, a controllable pair is near-uncontrollable if large amounts of control energy ($\int u^Tu$) are required to effect a state transfer. The pair

$$A = \begin{pmatrix} 0 & 1 & & 0 \\ \vdots & \ddots & \ddots & \\ \vdots & & \ddots & 1 \\ 0 & \cdots & \cdots & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

is near-uncontrollable in the energy sense but apparently not in the parametric sense. Of course, both measures are co-ordinate dependent and "balancing" is one attempt to try to remove this co-ordinate bias.

4.3 Computation of Objects Arising in the Geometric Theory of Linear Multivariable Control

A great many numerical problems arise in the geometric approach [1] to control of systems modeled as (1), (2). Some of these are discussed in [38] and [1] remains a fertile source of numerical problems. In fact, the power of the geometric approach derives in large part from its divorce from matrices and specific co-ordinate systems. Numerical issues are a separate concern. Two of the more elaborate but still fundamental objects in that theory are supremal (A,B) - invariant and controllability subspaces contained in a given subspace. An initial attempt at characterizing these spaces in terms of eigenvectors and a generalized eigenvalue problem was given in [56].

However, a rather different and very thorough numerical treatment of the problem has been done by Van Dooren [57], [58]. He has done the most definitive numerical study to date of the matrix pencil $(L-\lambda M)$ problem. This work has implications for most calculations done with linear state-space models. For example, one by-product is an extremely reliable algorithm (which amounts to an orthogonal version of Silverman's structure algorithm) for the computation of multivariable system zeros [59]. Like [37] this method involves a generalized eigenvalue problem (the Rosenbrock pencil) but the "infinite zeros" are first deflated out.

4.4 Frequency Response Calculations

Many properties of a linear system (1), (2) are known in terms of its frequency response matrix

$$G(j\omega) := C(j\omega I - A)^{-1} B + D; \quad (\omega \geq 0) \tag{23}$$

(or $G(e^{j\theta})$; $\theta \in [0, 2\pi]$ for (3), (4)). In fact, various norms of the return difference matrix $I + G(j\omega)$ and related quantities have recently been investigated as providing measures of robustness of a linear system with respect to stability, noise response, disturbance attenuation, sensitivity, etc. See [60] for some of the numerical aspects and [61], [62] for surveys of some of the control aspects.

It is thus a problem of considerable computational interest to efficiently compute $G(j\omega)$, given A,B, and C, for a (possibly) large number of values of ω (without loss of generality, D can be taken to be 0). A generally applicable algorithm for this problem is presented in [63]. Rather than solve the linear equation (with dense, unstructured A) $(j\omega I - A)X = B$ which would require $O(mn^3)$ operations for each value of ω, the new method does an initial reduction of A to upper Hessenberg form, H. The orthogonal matrices used to effect the Hessenberg form of A are incorporated into B and C giving $\tilde{B}$ and $\tilde{C}$. Now as ω varies, the coefficient matrix in the linear equation $(j\omega I - H)\ X = \tilde{B}$ remains in upper Hessenberg form. The advantage is that X can now be found in $O(mn^2)$ operations rather than $O(mn^3)$ as before, a substantial savings. Moreover, the method is numerically very stable and has the advantage of being independent of the eigenstructure (possibly ill-conditioned) of A.

Portable mathematical software, in the sense to be discussed in Section 5, is also available for this problem [64].

We note here that the above method can also be extended to state-space models in implicit form, e.g., (1) is replaced by

$$E\dot{x} = Ax + Bu. \tag{24}$$

Then (23) is replaced with

$$G(j\omega) = C(j\omega E - A)^{-1} B + D \tag{25}$$

and the initial triangular/Hessenberg reduction employed in [36] can be employed to again reduce the problem to one of updating the diagonal of a Hessenberg matrix and consequently an $O(n^2)$ linear equation problem.

4.5 Lyapunov, Sylvester, and Riccati Equations

Certain matrix equations arise naturally in linear control and systems theory. Among those frequently encountered in the analysis of continuous-time systems are the Lyapunov equation

$$FX + XF^T + H = 0, \tag{26}$$

and the Sylvester equation

$$FX + XG + H = 0. \tag{27}$$

The appropriate discrete-time analogues are

$$FXF^T - X + H = 0 \tag{28}$$

$$FXG - X + H = 0. \tag{29}$$

Various hypotheses are made on the coefficient matrices F, G, H to ensure certain properties of the solution X.

Surprisingly little attention has been paid to solution of these equations in the numerical linear algebra literature. There is, however, a voluminous literature in control and systems theory but most of that is ad hoc, at best, from a numerical point of view, with little attention paid to questions of numerical stability, conditioning, machine implementation, and the like.

For the Lyapunov equation the overall best algorithm in terms of efficiency, accuracy, reliability, availability, and ease of use appears to be that of Bartels and Stewart [65]. The basic idea is to reduce F to quasi-upper-triangular form (or real Schur form (RSF)) and perform a back substitution for the elements of X.

For the Sylvester equation the Bartels-Stewart algorithm reduces both F and G to real Schur form (RSF) and then a back substitution is done. It has been demonstrated in [66] that some improvement in this procedure is possible by only reducing the larger of F and G to upper Hessenberg form.

A promising new algorithm for solving Lyapunov equations has recently been proposed by Hammarling [67]. This algorithm is a variant of the Bartels-Stewart algorithm which solves directly for the Cholesky factor Y of X : $Y^TY = X$ and Y is upper-triangular. Clearly, given Y, X is easily recovered if necessary. But in many applications, for example [55], only the Cholesky factor is required.

Open questions remain concerning estimating the condition of Lyapunov and Sylvester equations efficiently and reliably in terms of the coefficient matrices.

A deeper analysis of the Lyapunov and Sylvester problems is probably a prerequisite to at least a better understanding of conditioning of the Riccati equation for which again, there is a considerable theoretical literature but rather little known from a purely numerical point of view. The symmetric algebraic Riccati equation takes the form

$$XGX + FX + XF^T + H = 0 \tag{30}$$

for continuous-time systems and

$$FXF^T - X - FXG_1(G_2 + G_1^TXG_1)^{-1}G_1^TXF^T + H = 0 \tag{31}$$

for discrete-time situations. Again, appropriate assumptions are made on the coefficient matrices to guarantee the existence and/or uniqueness of certain kinds of solutions X. Nonsymmetric Riccati equations of the form

$$XGX + F_1X + XF_2 + H = 0 \tag{32}$$

for the continuous-time case (along with an analog for the discrete-time case) are also studied and can be solved numerically by the techniques discussed below.

One of the more reliable general-purpose methods for solving Riccati equations is the Schur method [68]. For the case of (30), for example, this method is based upon the reduction of the associated Hamiltonian matrix

$$\begin{pmatrix} F^T & G \\ -H & -F \end{pmatrix} \tag{33}$$

to RSF. If the RSF is ordered so that its stable eigenvalues (there will be exactly n of them under certain assumptions) are in the upper left triangle, the corresponding first n vectors of the orthogonal matrix which effects the reduction will form a basis for the stable eigenspace from which the Riccati solution is then easily found.

Extensions to the basic Schur method have been made [69], [70] which were motivated by the following situations:

(i) G in (30) is of the form $BR^{-1}B^T$ where R may be near-singular.
(ii) F in (31) is singular (F^{-1} is required in the classical approach involving a symplectic matrix which plays the role of (33)).

In fact, these extensions can be generalized even further and the following problem will illustrate. Consider the optimal control problem

$$\text{Min } \frac{1}{2}\int_0^{+\infty} [x^TQx + 2x^TSu + u^TRu]dt \tag{34}$$

$$\text{subject to} \quad E\dot{x} = Ax + Bu. \tag{35}$$

The Riccati equation associated with (34), (35) then takes the form

$$E^TXBR^{-1}B^TXE - (A - BR^{-1}S^T)^TXE - E^TX(A - BR^{-1}S^T) - Q + SR^{-1}S^T = 0 \qquad (36)$$

This so-called "generalized" Riccati equation can be solved by considering the associated matrix pencil

$$\begin{pmatrix} A & 0 & B \\ -Q & -A^T & -S \\ S^T & B^T & R \end{pmatrix} - \lambda \begin{pmatrix} E & 0 & 0 \\ 0 & E^T & 0 \\ 0 & 0 & 0 \end{pmatrix} . \qquad (37)$$

Notice that S in (34) and E in (35) are handled directly and no inverses appear. Numerical methods for handling (37) and a large variety of related problems are described in [70], [71] and a thorough survey of the Schur method, generalized eigenvalue/eigenvector extensions, and the underlying algebraic structure in terms of "Hamiltonian pencils" and "symplectic pencils" is described in [72].

Schur techniques can also be applied to Riccati differential and difference equations [73] and to nonsymmetric Riccati equations which arise in, for example, invariant imbedding methods for solving linear two-point boundary value problems [72].

As with the linear Lyapunov and Sylvester equations there are few satisfactory results concerning conditioning of Riccati equations, a topic of great interest independent of what solution method is used, be it a Schur-type method or one of numerous alternatives. One fairly reliable method is to estimate the condition of U_{11} with respect to inversion where $\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}$ is a basis for the stable eigenspace. This turns out to be essentially equivalent to the somewhat more elaborate procedure proposed in [74]. But it is easy to provide examples of ill-conditioned Riccati equations where U_{11} is well-conditioned and so a much more sophisticated analysis needs to be performed and efforts are under way by numerous groups towards this end.

A software package for Riccati equations called RICPACK has been partially completed (Jan. 1983) by this author and W.F. Arnold. Highlights of the capabilities of this general software (in Fortran) include:

(i) Ward's balancing [75] for the generalized eigenvalue problem
(ii) direct handling of singular control weighting or measurement noise covariance
(iii) direct handling of cross-weighting or noise correlation
(iv) direct handling of descriptor variable systems
(v) spectral factorization
(vi) iterative refinement by Newton's method and Sylvester equations
(vii) residual calculations and condition estimates

5. MATHEMATICAL SOFTWARE

5.1 General Remarks

The previous two sections have highlighted some topics from numerical linear algebra and their applications to numerical problems arising in systems, control, and estimation theories. Of course, these problems represent only a very small subset of numerical problems of interest but even for problems which are apparently "simple" from a mathematical point of view, the myriad of little details which constitute a sophisticated implementation become so overwhelming that the only effective means of communicating an algorithm is through its embodiment as mathematical software. Mathematical or numerical software simply means an implementation on a computing machine of an algorithm for solving a mathematical problem. Such software must be reliable, portable, and unaffected by the machine or system environment in which it is used.

The prototypical work on reliable, portable mathematical software for the standard eigenproblem was started in 1968. EISPACK [10], [34] Editions I and II were an outgrowth of that work. Subsequent efforts of interest to control engineers include LINPACK [19] for linear equations and linear least squares problems, FUNPACK (Argonne) for certain function evaluations, MINPACK (Argonne) for certain optimization problems, ROSEPACK for robust statistical estimation, and various ODE and PDE codes. High quality algorithms are published regularly in the ACM Transactions on Mathematical Software.

Moreover, many pre-processors that are, themselves, portable software have been designed and implemented to assist in instituting and verifying portability. Among such machine aids that are in use are the PFORT verifier [76] from Bell Labs. and the Fortran Converter from International Mathematical and Statistical Libraries, Inc. [77]. Also available are machine aids such as TAMPR and POLISH. Technology to aid in the development of mathematical software in Fortran is being assembled as a package called TOOLPACK (University of Colorado). Mechanized code development offers other advantages with respect to, for example, modifications, updates, versions, and maintenance. Excellent references on portability and other aspects of mathematical software include [78]-[80].

Inevitably numerical algorithms are strengthened when their mathematical software is made portable since their widespread use is greatly facilitated. Furthermore, such software has been shown to be markedly faster by factors ranging from 10 to 50 than earlier and less reliable code.

One can list many other features besides portability, reliability, and efficiency which are characteristic of "good" mathematical software. For example, one can include:

(i) high standards of documentation and style
(ii) ease of use; ability of the user to interact with the algorithm
(iii) consistency/compatibility/modularity in the context of a larger package or more complex problem
(iv) error control, exception handling
(v) robustness with respect to unusual situations
(vi) graceful performance degradation as problem domain boundaries are approached
(vii) program size (a function of intended use: e.g. low accuracy, real-time applications)
(viii) availability and maintenance
(ix) "tricks" such as underflow-/overflow- proofing if necessary and implementation of columnwise or rowwise linear algebra [81].

Clearly, the list can go on.

What becomes apparent from the above considerations is that the evaluation of mathematical software is a highly nontrivial task. Clearly the quality of software is largely a function of its operational specification. It must also reflect the numerical aspects of the algorithm being implemented. The language used and the compiler (e.g., optimizing or not) used for that language will both have an enormous impact on quality - both perceived and real as will the underlying hardware and arithmetic. Different implementations of an algorithm can have markedly different properties and behavior – even of the same good underlying algorithm. Further discussions on this subject can be found in [79], [80] and [82].

5.2 Mathematical Software in Control

Many aspects of systems, control, and estimation theory are ready for the research and design that is necessary to produce reliable, portable mathematical software that performs in finite arithmetic. Certainly many of the underlying linear algebra tools (for example, in EISPACK and LINPACK) are considered sufficiently reliable as to be used as black – or at least gray – boxes by control engineers. Much of that theory and methodology can and has been carried over to control problems. However, much of the work done in control, particularly the design and synthesis aspects, is not amenable to nice, "clean" algorithms and the ultimate software must have the capability to enable a dialogue between the computing machine and the control engineer but with the latter probably still making the final engineering decisions. We might never see a "control package" (CONPACK) that will look like EISPACK or LINPACK. To even attempt it would be a CONJOB. Instead, a better analogy would be made by trying to emulate a good ODE or PDE package.

What mathematical software can provide is a "toolbox" from which the control engineer can choose software tools and robustly coded algorithms to easily implement new or modified theories or designs. Mathematical software forms the foundation of a computer-aided control system design (CACSD) package but is only one of many interlocking parts.

Most CACSD packages – and there are now hundreds of them – divide fairly naturally into two fundamental levels, each with further subdivisions, of course. The lower level contains the numerical software and this can be written very portably. The upper level contains the basic design and analysis procedures which call the low level procedures or subroutines for their actual implementation. Also in the upper level is the key part, as far as the control engineer is concerned, and that is the user interface. This interface interacts with the upper level procedures as well as the *I/Ø*, graphics, and file and database management systems. Here the question of portability is considerably more complex and most packages aim for particular "target environments". For further comments and examples see [83], [84].

Finally, we mention a few recent developments which are relevant to future control and estimation software developments. With respect to languages, Fortran is likely to remain the most common (and very efficient) language for quite some time. Fortran coding and portability is certain to be aided by the adoption and use of the Fortran 77 standard (ANSI X3.9-1978) and work continues on Fortran 8X. We may also see some movement towards Ada™. The use of other languages such as Pascal, C, PL1, and APL seems generally limited despite their attractiveness in particular environments. In hardware we are seeing more and more microprocessor-based systems. Parallel architectures will both demand and suggest new algorithms and control strategies. In arithmetic the implementation of IEEE arithmetic in some computing environments could have a major beneficial impact on mathematical software. In graphics substantial progress is being made towards development of standards.

6. CONCLUDING REMARKS

Some numerical issues and techniques from numerical linear algebra together with some applications of these ideas have been outlined. A key question in these and other problems in systems, control, and estimation theory is what can be reliably computed and used in the presence of parameter uncertainty or structural constraints (e.g., certain "hard zeros") in the original model and roundoff errors in the calculations. However, the ultimate goal is to solve real problems and reliable tools (mathematical software) and experience must be available to effect real solutions or strategies. Only a serious interdisciplinary

Ada is a trademark of the U.S. Department of Defense

effort is capable of making substantial progress in improving the present state of affairs. As we move out of the "just programming" era we can expect to see soon some high quality control software. We have already witnessed a fruitful symbiosis between numerical analysis and numerical problems from control. We can expect a further symbiotic relationship as control engineering realizes the full potential of graphics, "cheap" memory, and substantial computing power. However, as in other applications areas, software will continue to dominate both as a constraint and as a vehicle for progress. Unfortunately, exceptionally high quality software is exceptionally expensive, in terms of both money and time.

REFERENCES

[1] Wonham, W.M., Linear Multivariable Control: A Geometric Approach, 2nd. Ed., Springer-Verlag, New. York, 1979.
[2] Brockett, R.W., Finite Dimensional Linear Systems, Wiley, New York, 1970.
[3] Kwakernaak, H., and Sivan, R., Linear Optimal Control Systems, Wiley, New York, 1972.
[4] Anderson, B.D.O., and Moore, J.B., Optimal Filtering, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.
[5] Moler, C.B., and VanLoan, C.F., Nineteen dubious ways to compute the exponential of a matrix, SIAM Review, 20, pp. 801-836, 1978.
[6] Wilkinson, J.H., Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, New Jersey, 1963.
[7] Miller, W., and Wrathall, C., Software for Roundoff Analysis of Matrix Algorithms, Academic Press, New York, 1980.
[8] Computer, Vol. 14, No. 3, March 1981.
[9] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford University Press, London, 1965.
[10] Smith, B.T., et al., Matrix Eigensystem Routines - EISPACK Guide, 2nd Ed., Lect. Notes in Comp. Sci., Vol. 6, Springer-Verlag, New York, 1976.
[11] Forsythe, G.E., Malcolm, M.A., and Moler, C.B., Computer Methods for Mathematical Computations, Prentice-Hall, Englewood Cliffs, New Jersey, 1977.
[12] Johnston, R.L., Numerical Methods: A Software Approach, Wiley, New York, 1982.
[13] Rice, J.R., Matrix Computations and Mathematical Software, McGraw-Hill, New York, 1981.
[14] Stewart,G.W., Introduction to Matrix Computations, Academic Press, New York, 1973.
[15] Dahlquist, G., and Björck, A., Numerical Methods, Prentice-Hall, Englewood Cliffs, New Jersey, 1974.
[16] Stoer, J., and Bulirsch, R., Introduction to Numerical Analysis, Springer-Verlag, New York, 1980.
[17] Henrici, P., Essentials of Numerical Analysis, Wiley, New York, 1982.
[18] Forsythe, G.E., and Moler, C.B., Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs, New Jersey, 1967.
[19] Dongarra, J., et al., LINPACK User's Guide, SIAM, Philadelphia, 1979.
[20] Householder, A.S., The Theory of Matrices in Numerical Analysis, Blaisdell, New York, 1964.
[21] Cline, A.K., et al., An estimate for the condition number of a matrix, SIAM JNA, 16, pp. 368-375, 1979.
[22] Lawson, C.L., and Hanson, R.J., Solving Least Squares Problems, Prentice-Hall, Englewood Cliffs, New Jersey, 1974.
[23] Stewart, G.W., On the perturbation of pseudo-inverses, projections, and linear least squares, SIAM Rev., 19, pp. 634-662, 1977.
[24] Stewart, G.W., The economic storage of plane rotations, Numer. Math., 25, pp. 137-138, 1976.
[25] Parlett, B.N., The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.
[26] Gannon, D., A note on pipelining a mesh connected microprocessor for finite element problems by nested dissection, Proc. 1980 Int'l. Conf. on Parallel Proc., pp. 197-204.
[27] Gentleman, M., and Kung, H.T., Matrix triangularization by systolic arrays, Proc. SPIE, Vol. 298, Real-Time Signal Processing IV, 1981.
[28] Heller, D.E., and Ipsen, I.C.F., Systolic networks for orthogonal equivalence transformations and their applications, Proc. Conf. on Advanced Research in VLSI, P. Penfield (Ed.), MIT, pp. 113-122, Jan. 1982.
[29] Lord, R.E., Kowalik, J.S., and Kumar, S.P., Solving linear algebraic equations on a MIMD computer, Proc. 1980 Int'l. Conf. on Parallel Proc., pp. 205-210.
[30] Sameh, A.H. and Kuck, D.J., On stable parallel linear system solvers, JACM, 25, pp. 81-91, 1978.
[31] Golub, G.H., and Wilkinson, J.H., Ill-conditioned eigensystems and the computation of the Jordan canonical form, SIAM Review, 18, pp. 578-619, 1976.
[32] Francis, J.G.F.- The QR transformation I, II., Computer J., 4, pp. 265-271, 4, pp. 332-345, 1961, 1962.
[33] Martin, R.S., and Wilkinson, J.H., Similarity reduction of a general matrix to Hessenberg form, Numer. Math., 12, pp. 349-368, 1968.
[34] Garbow, B.S., et al., Matrix Eigensystem Routines - EISPACK Guide Extension, Lect. Notes in Comp. Sci., Vol. 51, Springer-Verlag, New York, 1977.
[35] Wilkinson, J.H., and Reinsch, C., Handbook for Automatic Computation, Vol. II, Linear Algebra, Springer-Verlag, New York, 1971
[36] Moler, C.B., and Stewart, G.W., An algorithm for generalized matrix eigenvalue problems, SIAM JNA, 10, pp. 241-256, 1973.
[37] Laub, A.J., and Moore, B.C., Calculation of transmission zeros using QZ techniques, Automatica, 14, pp. 557-566, 1978.
[38] Klema, V.C., and Laub, A.J., The singular value decomposition: its computation and some applications, IEEE TAC, AC-25, pp. 164-176, 1980.
[39] Golub, G.H., and Kahan, W., Calculating the singular values and pseudo-inverse of a matrix, SIAM JNA, 2, pp. 205-224, 1965.

[40] Bierman, G.J., Factorization Methods for Discrete Sequential Estimation, Academic Press, New York, 1977.
[41] Golub, G.H., and Reinsch, C., Singular value decomposition and least squares solutions, Numer. Math, 14, pp. 403-420, 1970.
[42] Golub, G.H., Klema, V.C., and Stewart, G.W., Rank Degeneracy and Least Squares Problems, Tech. Rep't. No. STAN-CS-76-559, Comp. Sci. Dept., Stanford Univ., Aug. 1976, (also issued as NBER Rep't. and Univ. of Maryland, CS Dept., Rep't.)
[43] Ostrowski, A.M., On the spectrum of a one-parametric family of matrices, Journal für die reine und angewandte Mathematik, Band 193, Heft 3/4, pp. 143-160, 1954.
[44] Laub, A.J., Survey of computational methods in control theory, in Erisman, A.M., et al., (Eds.), Electric Power Problems: The Mathematical Challenge, SIAM, Philadelphia, pp. 231-260, 1980.
[45] Van Dooren, P., Numerical linear algebra: an increasing interest in linear system theory, Proc. Eur. Conf. Circ. Th. and Design 81, The Hague, The Netherlands, pp. 243-251, Aug. 1981.
[46] Enright, W., On the efficient and reliable numerical solution of large linear systems of ODE's, IEEE TAC, AC-24, pp. 905-908, 1979.
[47] Van Loan, C.F., The sensitivity of the matrix exponential, SIAM J. Num. Anal., 14, pp. 971-981, 1977.
[48] Kågström, B., Bounds and perturbation bounds for the matrix exponential, BIT, 17, pp. 39-57, 1977.
[49] Paige, C.C., Properties of numerical algorithms related to computing controllability, IEEE TAC, AC-26, pp. 130-138, 1981.
[50] Miminis, G.S., and Paige, C.C., An algorithm for pole assignment of time-invariant multi-input linear systems, Proc. 21st Conf. on Decis. and Control, Orlando, FL, pp. 62-67, Dec. 1982.
[51] Patel, R.V., Computational algorithms for pole assignment in linear multivariable systems, Proc. Second IFAC Symp. on Comp. Aided Design of Mltvbl. Tech. Sys., Purdue Univ., West Lafayette, IN, pp. 79-89, Sept. 1982.
[52] Petkov, P. Hr., Christov, N.D., and Konstantinov, M.M., A computational algorithm for pole assignment of linear single-input systems, Submitted to IEEE Trans. Aut. Contr., Dec. 1982.
[53] Varga, A., A schur method for pole assignment, IEEE Trans. Aut. Contr., AC-26, pp. 517-519, 1981.
[54] Moore, B.C., Principal component analysis in linear systems: controllability, observability, and model reduction, IEEE TAC, AC-26, pp. 17-32, 1981.
[55] Laub, A.J., On computing "balancing" transformations, Proc. 1980 JACC, San Francisco, CA, pp. FA8-E, 1980.
[56] Moore, B.C., and Laub, A.J., Computation of supremal (A,B)-invariant and controllability subspaces, IEEE TAC, AC-23, pp. 783-792, 1978.
[57] Van Dooren, P., The generalized eigenstructure problem. Applications in linear system theory, Ph.D. Thesis, Katholieke Universiteit Leuven, Belgium, 1979.
[58] Van Dooren, P., The generalized eigenstructure problem in linear system theory, IEEE TAC, AC-26, pp. 111-129, 1981.
[59] Emami-Naeini, A., and P. VanDooren, Computation of zeros of linear multivariable systems, Automatica, 18, pp. 415-430, 1982.
[60] Laub, A.J., Robust stability of linear systems - some computational considerations, in A.L. Schoenstadt, et al. (Eds.), Information Linkage Between Applied Mathematics and Industry II, Academic Press, New York, pp. 57-84, 1980.
[61] 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 TAC, AC-26, pp. 47-65, 1981.
[62] Doyle, J.C., and Stein, G., Multivariable feedback design: concepts for a classical/modern synthesis, IEEE Trans Aut. Contr., AC-26, pp. 4-16, 1981.
[63] Laub, A.J., Efficient multivariable frequency response calculations, IEEE TAC, AC-26, pp. 407-408, 1981.
[64] Laub, A.J., ALGORITHM: Efficient calculation of frequency response matrices from state space models, submitted to ACM Trans Math. Software, June 1982.
[65] Bartels, R.H., and Stewart, G.W., Solution of the matrix equation AX + XB = C., Comm. ACM, 15, pp. 820-826, 1972.
[66] Golub, G.H., Nash, S., and Van Loan, C.F., A Hessenberg-Schur method for the problem AX + XB = C., IEEE TAC, AC-24, pp. 909-913, 1979.
[67] Hammarling, S.J., Numerical solution of the stable, non-negative definite Lyapunov equation, IMA J. Numer. Anal., 2, pp. 303-323, 1982.
[68] Laub, A.J., A Schur method for solving algebraic Riccati equations, IEEE TAC, AC-24, pp. 913-921, 1979.
[69] Pappas, T., Laub, A.J., and Sandell, N.R., On the numerical solution of the discrete time algebraic Riccati equation, IEEE TAC, AC-25, pp. 631-641, 1980.
[70] Van Dooren, P., A generalized eigenvalue approach for solving Riccati equations, SIAM J. Sci. Stat. Comp., 2, pp. 121-135, 1981.
[71] Lee, K.H., Generalized Eigenproblem Structures and Solution Methods for Riccati Equations, Ph.D. Thesis, Dept. of Elec. Eng.-Systems, Univ. of Southern California, Jan. 1983.
[72] Laub, A.J., Schur techniques in invariant imbedding methods for solving two-point boundary value problems, Proc. 21st CDC, Orlando, FL, pp. 56-61, Dec. 1982.
[73] Laub, A.J., Schur techniques for Riccati differential equations, in Hinrichsen, D. and Isidori, A., (Eds.), Feedback Control of Linear and Nonlinear Systems, Springer-Verlag, New York, pp. 165-174, 1982.
[74] Paige, C.C., and Van Loan, C.F., A Schur decomposition for Hamiltonian matrices, to appear in Linear Alg. and Its Applics.
[75] Ward, R.C., Balancing the generalized eigenvalue problem, SIAM J. Sci. Stat. Comp., 2, pp. 141-152, 1981.

[76] Ryder, B.G., The PFORT Verifier: User's Guide, CS. Tech. Rept. 12, Bell Labs, 1975.
[77] Aird, T.J., The FORTRAN Converter User's Guide, IMSL, 1975.
[78] Cowell, W., Portability of Numerical Software, Oak Brook, 1976, Lect. Notes in Comp. Sci., Vol. 57, Springer-Verlag, New York, 1977.
[79] Fosdick, L.D. (Ed.), Performance Evaluation of Numerical Software, North-Holland, New York, 1979.
[80] Hennel, M.A., and Delves, L.M., (Eds.), Production and Assessment of Numerical Software, Academic Press, New York, 1980.
[81] Moler, C.B., Matrix computations with Fortran and paging, Comm. ACM, 15, pp. 268-270, 1972.
[82] Cowder, H., Dembo , R.S., and Mulvey, J.M., On reporting computational experiments with mathematical software, ACM Trans. Math. Software, 5, pp. 193-203, 1979.
[83] Control Systems Magazine, Special Issue on CACSD, Vol. 2, No. 4, Dec. 1982.
[84] Herget, C.J. and Laub, A.J., Editorial in [83].

This research was supported by the U.S. Army Research Office under Contract DAAG29-81-K-0131.

PERFORMANCE AND ROBUSTNESS ASPECTS OF DIGITAL CONTROL SYSTEMS

J. E. Wall, J. C. Doyle, G. L. Hartmann, N. A. Lehtomaki, G. Stein

Honeywell, Incorporated
Systems and Research Center
2600 Ridgway Parkway
Minneapolis, Minnesota 55413

SUMMARY

A recent formulation of the feedback control problem has been proposed which captures both performance and robustness aspects of feedback. The structured singular value provides the solution to this problem. This paper reviews the new problem formulation and solution and extends its applicability to digital feedback control systems. A digital compensator is treated as though it were an analog compensator through the use of sectors. An integrated flight-propulsion control system is used as an illustrative example.

1. INTRODUCTION

The basic requirement of feedback systems is to achieve certain desired levels of performance and also to be tolerant of uncertainties. Performance levels concern such things as command following, disturbance rejection, sensitivity, etc., while uncertainty tolerances deal with the inevitable differences which exist between a physical plant and its mathematical model. As discussed in various textbooks and references, these two aspects of the feedback problem lead to fundamental tradeoffs and compromises which motivate the entire body of feedback theory. Incorporation of a digital computer does not alter this basic requirement.

An essential difficulty in the theory has been to capture both the performance and uncertainty aspects of feedback in a single problem statement. Thus we have optimization theories which emphasize performance, robustness theories which emphasize uncertainties, and a host of ad hoc tools which attempt to compromise the two. A recent problem formulation has been proposed which captures both aspects of feedback under the umbrella of what is called the "block-diagonal bounded perturbation problem." The solution to this problem involves a generalization of the ordinary singular value decomposition. It provides a reliable, nonconservative measure to determine whether both the performance and robustness requirements of a feedback loop are satisfied. This measure is called the structured singular value and serves as the essential analysis tool. One of the major goals of this paper is to disseminate this problem formulate.

The second goal of this paper is to expound a method of analysis for digital control systems that uses the theory of sectors. When used in a feedback configuration, a digital computer is embedded in a compensator that consists of a prefilter, a sampler, the computer, and a hold device. Although such a compensator has a digital components, it transforms an analog input signal into an analog output signal. An accurate model of its input/output behavior requires an analog, time-varying operator. Such a model is too complicated, however, for easy use in control system design and analysis. The theory of sectors provides linear, time-invariant approximations for digital controllers. A fundamental theorem of Thompson[1] is used extensively in this development.

The basic idea of sectors is that a very complicated operator can be reliably approximated by a simple "center", provided that the approximation error is properly accounted. The "radius" of the sector bounds the errors of this approximation. Finite-dimensional, linear, time-invariant operators will be presented for the center and radius of a sector that describes a quite general digital compensator. The error introduced by this approximation is in addition to the usual modeling error associated with the controlled plant. The use of sectors together with the structured singular value successfully addresses simultaneous performance and robustness issues.

The paper is organized into seven major sections. Section 2 provides a review of some requisite background material. Section 3 shows how to use conic sectors in the analysis of hyorid compensators.

This work has been supported by Honeywell Internal Research and Development Funding, The Office of Naval Research under ONR Research Grant N00014-82-C-0157, and the U.S. Air Force Office of Scientific Research Grant F49620-82-C-0090.

In Section 4, the robustness and performance aspects of feedback are formulated as a stability problem involving block-diagonal bounded perturbations. This problem is solved in Section 5 using the new structured singular value concept. These ideas are illustrated through the example problem of an integrated control mode for an advanced fighter aircraft in Section 6. Section 7 contains concluding remarks.

2. MATHEMATICAL PRELIMINARIES

2.1 Topics from Functional Analysis

The foundation for the development in this paper is provided by a review of a number of basic concepts from functional analysis. The first of these is the normed linear space L_2^{rxm}. This function space is the collection of all rxm-dimensional functions which are square integrable on R. An inner product for any two functions x and y in L_2^{rxm} is defined as

$$\langle x,y\rangle = \int_{-\infty}^{+\infty} \mathrm{Tr}[y^H(t)x(t)]dt \tag{2.1}$$

The norm associated with this inner product is

$$\|x\|_2 = \langle x,x\rangle^{1/2} \tag{2.2}$$

Elements of L_2^{rxm} have finite norm.

The chief limitation of the space L_2^{rxm} for control system analysis is that it contains no unstable functions, i.e., functions with $\|x\|_2 = \infty$. This can be remedied by introducing the extended normed linear space L_{2e}^{rxm}. This space is the collection of all functions which are square integrable on all finite intervals of R. More precisely, we introduce the truncated norm as

$$\|x\|_{2,\tau} = \{\int_{-\tau}^{\tau} \mathrm{Tr}[x^H(t)x(t)]dt\}^{1/2} \tag{2.3}$$

Then L_{2e}^{rxm} contains all functions $x:R\to C^{rxm}$ which satisfy $\|x\|_{2,\tau} < \infty$ for all τ. Functions such as $x(t)=e^t$ are included in L_{2e}^1, for example, while functions such as $x(t) = \tan(t)$ are not. Note that all elements of L_2^{rxm} are included in the extension L_{2e}^{rxm} and have the property that $\|x\|_{2,\tau} \to \|x\|_2$ as $\tau \to \infty$.

The other function space we will need is L_∞^{rxm}. This space consists of all functions which are measurable and essentially bounded. It is a normed linear space with norm

$$\|x\|_\infty = \text{ess sup } \bar{\sigma}(x(t))^{\#} \tag{2.4}$$

No distinction will be made between functions differing over sets of measure zero and in the sequel we will use sup for the essential supremum.

The Fourier transform of a function x is

$$\hat{x}(j\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt \tag{2.5}$$

The operation of the Fourier transform is a linear isometry of L_2^{rxm} onto L_2^{rxm}. The Parseval formula relates inner products

$$\langle x,y\rangle = \frac{1}{2\pi}\langle \hat{x}, \hat{y}\rangle \tag{2.6}$$

and specializes to

$$\|x\|_2^2 = \frac{1}{2\pi}\|\hat{x}\|_2^2 \tag{2.7}$$

An operator G is a mapping which associates which each function in its domain exactly one function in its range. For our purposes, the domain of an operator will be L_{2e}^m and its range will be some

The symbol $\bar{\sigma}(\cdot)$ denotes the maximum singular value of a matrix. The singular values of a matrix A are the non-negative square roots of the eigenvalues of the Hermitian form A^HA.

subset of L_{2e}^r. The mapping of an operator is denoted

$$y = Gx \tag{2.8}$$

It is assumed that G is a causal operator. Causality means that the output of G at time t, does not depend on values of the input at future times, say $t_2 > t_1$.

An operator has a norm induced by the norms on L_{2e}^m and L_{2e}^r. The induced operator norm is

$$\|G\|_2 = \sup_{\tau} \sup_{\|x\|_{2,\tau}=0} \frac{\|Gx\|_{2,\tau}}{\|x\|_{2,\tau}} \tag{2.9}$$

It is a common abuse of notation to use $\|\cdot\|_2$ to denote both function norms and the induced operator norm. The distinction is made clear by the arguments used with the symbol.

The causal operator G is said to be L_{2e} stable if it has finite gain, i.e., $\|G\|_2 < \infty$. Thus L_{2e} stable operators map L_2^m into L_2^r.

We now restrict attention to linear operators G defined by the convolution integral

$$(Gx)(t) = \int_{-\infty}^{\infty} g(t-\tau)\, x(\tau) d\tau \tag{2.10}$$

where g is absolutely integrable. Fubini's theorem provides a relationship between the Fourier transform of y=Gx and the transforms of g and x,

$$\hat{y} = \hat{g}\hat{x} \tag{2.11}$$

We refer to $\hat{g}$ as the transfer matrix of the operator G. The Fourier transform converts the original convolution into multiplication in the transformed (frequency) domain. Moreover, the Fourier transform relates the norm of an L_{2e} stable operator G to the transform g. By the Parseval formula (2.7),

$$\|G\|_2 = \sup_{\|x\|_2 \neq 0} \frac{\|Gx\|_2}{\|x\|_2} = \sup_{\|\hat{x}\|_2 \neq 0} \frac{\|\hat{g}\hat{x}\|_2}{\|\hat{x}\|_2} \tag{2.12}$$

In fact, it can be shown [6] that

$$\|G\|_2 = \|\hat{g}\|_\infty = \sup_{\omega} \bar{\sigma}(\hat{g}(j\omega)) \tag{2.13}$$

Thus for stable convolutional operators, the norm is simply equal to the supremum over frequency of the largest singular value of the transfer matrix. This makes convolutional operators into a normed linear algebra.

2.2 Performance Measures for Linear Systems

Two alternate measures of performance for linear systems are related to the basic concepts of functional analysis in this section. Broadly speaking, "good" performance means that some error response is "small" in an appropriate sense. An example of such an error response is the classical output sensitivity function of a feedback loop which relates command following errors to output commands. Another example is the response at the input of a plant to sensor noise on the measured variable. We will characterize an error response as an operator and measure its size through norms on the operator. The operator is denoted G and maps inputs u in L_2^m into outputs y in L_2^r.

The first type of performance considered is stochastic performance, i.e. the statistical behavior of the error subject to random inputs. If the input u is zero mean white noise with unit intensity, then the covariance of the error is simply

$$E(|y(t)|^2) = \frac{1}{2\pi} \|\hat{g}\|_2^2 \tag{2.14}$$

This performance measure is the integral in the frequency domain of $\mathrm{Tr}(g^H g)$. More generally, the input can be colored noise formed by shaping white noise through the operator R^{-1}. Also, we have the freedom to examine the covariance of a filtered version of the error, say $z = Ly$. For example, such a filter operator could be used to emphasize a particularly crucial band in the frequency domain or to deemphasize low or high frequencies. Including the operators L and R^{-1} yields

$$E(|z(t)|^2) = \frac{1}{2\pi} \|\hat{\ell}\hat{g}r^{-1}\|_2^2 \tag{2.15}$$

The stable operators L and R^{-1} serve as weightings in the frequency domain in the integral (2.15). This measure of performance has received a great deal of attention in the literature on Wiener and Kalman filtering and the linear-quadratic-Gaussian control problem.

The second type of performance measure is "worst case" performance. The idea in this case is to let the input be any arbitrary function in L_2^m with unit norm. Then the size of the error in the worst case is

$$\sup_{\|u\|_2=1} \|y\|_2 = \|G\|_2 = \|\hat{g}\|_\infty \tag{2.16}$$

Good performance in this sense requires having a small value of $\bar{\sigma}(g(j\omega))$ at every frequency ω. The utility of this approach to measuring performance is increased by introducing weighting operators L and R much as was done for stochastic performance. With the weightings, we have

$$\sup_{\|Ru\|_2=1} \|Ly\|_2 = \|\hat{\ell}\hat{g}r^{-1}\|_\infty \tag{2.17}$$

This measure of performance can be written in a slightly different form to emphasis its use as a performance specification. Mathematically,

$$\begin{aligned} & \|Ly\|_2 < 1 \quad \text{for all} \quad \|Ru\|_2 < 1 \\ \Leftrightarrow \quad & \|LGR^{-1}\|_2 < 1 \\ \Leftrightarrow \quad & \|\hat{\ell}\hat{g}r^{-1}\|_\infty < 1 \end{aligned} \tag{2.18}$$

This says that having the maximum singular value of the (weighted) transfer matrix less than one at every frequency is necessary and sufficient for the (weighted) error to have norm less than one for any (weighted) input with norm no larger than one. This measure has been the subject of recent theoretical interest within the control community [7], [8]. Also, when viewed as condition on the norm of LGR^{-1}, this measure of performance is applicable to nonlinear systems.

Both of these performance measures are expressed in terms of norms on weighted transfer matrices. Both are useful in the analysis of linear systems. The design problem in the first case is to minimize "average" error in the integral square sense. In the second case, the design problem is a mini-max problem: minimize the worst case error in the integral square sense. The remainder of this paper will use the second measure of performance as expressed in (2.18). This will allow us to obtain conditions on simultaneous performance and robustness. As an aside, we note that the performance measure in (2.15) can also be interpreted as a mini-max problem with the error signal measured by the L_∞ norm.

2.3 Sectors

In the last several years, the sector concept has been recognized as an important tool in feedback design and analysis [.9], [10], [11]. The basic idea is that very complicated plant operators (perhaps nonlinear, infinite dimensional, time-varying) can be reliably approximated by simple sector centers (usually finite dimensional, linear, time-invariant systems), provided that the approximation error is properly accounted for in the design process. This "proper accounting" usually means that a design based on the sector center must be restricted to maintain stability. Such restrictions generally increase as the magnitudes of the approximation errors grow. This section provides an introduction to sectors and gives a stability test in terms of a sector condition. It concludes with an interpretation of the stability test as imposing restrictions on the nominal design.

In abstract terms, a sector condition is a functional inequality describing the set of operators in a specified neighborhood of some nominal operator. Formally, the sector (C,L,R) is the set of all operators G mapping L_{2e}^m into L_{2e}^r which satisfy

$$\|L(G-C)R^{-1}x\|_{2,\tau} < \|x\|_{2,\tau} \quad \forall\, \tau \in R_+,\ x \in L_{2e}^m \tag{2.19}$$

The nominal operator C is referred to as the center of the sector. The operators L and R specify the size of the neighborhood about C. We will require L and R to be L_{2e}-stable operators and to have L_{2e}-stable inverses. When L and R are linear, time-invariant operators, this means they have no poles or zeros in the right-half of the complex plane and no excess of poles or zeros. In the case of an L_{2e}-stable center, condition (2.19) can be rewritten as

$$\| L(G-C)R^{-1} \|_2 < 1 \tag{2.20}$$

It is noteworthy to compare this condition with the performance specification (2.18).

It is helpful to introduce two alternate representations of a sector. From the defining relationship (2.19), it is clear that the sector (C,L,R) is equivalent to the parallel combination of the center C and the sector (0,L,R) having the null operator as its center. This equivalence is illustrated in Figure 2-1. There is a "multiplicative" representation of a sector as well as this "additive" one. The sector (C,L,R) is equivalent to the cascade of the center C with sector (I,L,RC^{-1}) having the identity operator as its center. This decomposition is also shown in Figure 2-1. Both of these representations will be useful in the sequel.

Conic sectors are those sectors for which the operator L is simply the identity. If G is a member of the conic sector or cone (C,R) with L_{2e}-stable center, then

$$\| G-C \|_2 < \| R \|_2 \tag{2.21}$$

and so the output of C approximates the output of G to within approximation error bounded by the norm of R. Conic sectors have simple graphical interpretations in two important special cases. One case is when G is a scalar nondynamical (i.e. memoryless) nonlinearity and C and R are scalar linear gains. The conic sector condition (2.21) is equivalent to

$$|G(x)-Cx| < |Rx| \tag{2.22}$$

This inequality has the simple graphical interpretation that the nonlinearity G(x) has its graph in the conic region between the line of slope (C-R) and the line of slope (C+R). This condition is illustrated in Figure 2-2. The second case is when G, C, and R are all scalar stable linear time-invariant operators. For this case, the frequency domain can be used to express the conic sector condition (2.21) as

$$|\hat{g}(j\omega)-\hat{c}(j\omega)| < |\hat{r}(j\omega)| \qquad \forall\, \omega \tag{2.23}$$

This condition describes the conic sector as the set of all frequency responses $\hat{g}(j\omega)$ which are within a distance $|\hat{r}(j\omega)|$ of the nominal transfer function $\hat{c}(j\omega)$. Figure 2-3 provides a graphical interpretation of this condition in terms of the Nyquist locus of $\hat{g}(j\omega)$ and $\hat{c}(j\omega)$. A circular "template" centered at $\hat{c}(j\omega)$ and of radius $\hat{r}(j\omega)$ in the Nyquist plane is defined at each frequency. The value of $\hat{g}(j\omega)$ must lie within each of these circular templates. These two examples show how conic sectors enable us to work with sets of models in the design process. This is an important function since no single model can represent a physical system with perfect fidelity.

2.4 Stability Conditions

Having introduced sectors, we now turn to characterizing the stability of a feedback system when an operator in the system is described by a sector. Consider the feedback configuration shown in Figure 2-4. The two closed-loop operators of this system are E_1 and E_2, mapping both inputs u_1 and u_2 into the outputs e_1 and e_2, respecively. It is assumed that the system is causal and well-posed, [6] i.e., the operators M,D,E_1, and E_2 are all causal. The system is stable if botn E_1 and E_2 are.

Theorem 2-1. (Small Gain Teorem) Under these conditions, the closed-loop system is L_{2e}-stable if

$$\| M \|_2 \| \Delta \|_2 < 1$$

A brief discussion of the proof of this theorem follows. The error e_1 is given by

$$\begin{aligned} e_1 &= u_1 - \Delta e_2 \\ &= u_1 - \Delta u_2 - \Delta M e_1 \end{aligned} \tag{2.24}$$

and so

$$(I+\Delta M)\, e_1 = u_1 - \Delta\, u_2 \tag{2.25}$$

The norm of the right hand side of (2.25) is bounded from above by

$$\| u_1 \|_2 + \| \Delta \|_2 \| u_2 \|_2$$

and the left hand side is bounded from below by

$$(1 - \| \Delta \|_2 \| M \|_2) \| e_1 \|_2$$

Combining these bounds and using the inequality of the theorem yields

$$\| e_1 \| < \frac{1}{1-\|M\|_2\|\Delta\|_2} (\|u_1\|_2+\|\Delta\|_2\|u_2\|_2) \tag{2.26}$$

Thus the operator E_1 has finite gain and is stable. Stability of E_2 is shown similarly.

The Small Gain theorem will now be used to obtain stability robustness conditions for a feedback system containing a sector. Stability robustness conditions will be obtained which guarantee the stability of the closed loop system for any operator that is an element of the sector. The feedback system under consideration is shown in Figure 2-5. Also shown in the figure are two alternative representations of the feedback loop. The first alternate uses the earlier observation that a sector may be expressed or the parallel combination of its center and a sector with the same radius centered about the null operator. The second alternative representation employs the closed loop operator for the feedback combination of G and C. It is assumed that the closed loop operator M is L_{2e}-stable.

The Small Gain Theorem will be applied to the feedback system in Figure 2-5 involving operator M and sector (0,L,R). Let the operator Δ be any element (0,L,R). By (2.20), this means that

$$\| L\Delta R^{-1} \|_2 < 1 \tag{2.27}$$

Rather than applying the Small Gain Theorem directly to M and Δ, we apply it to the operators RML^{-1} and $L\Delta R^{-1}$, shown in Figure 2-6. These operators are both L_{2e}-stable, and closed loop stability of the system in Figure 2-6 is equivalent to stability of the representations in Figure 2-5. By the Small Gain Theorem, closed loop stability is guaranteed if

$$\|RML^{-1}\|_2 \|L\Delta R^{-1}\|_2 < 1 \tag{2.28}$$

When combined with (2.27),

$$\|RML^{-1}\|_2 < 1 \tag{2.29}$$

suffices to ensure closed loop stability. This is summarized in the following theorem:

Theorem 2-2. Consider the feedback system of Figure 2-5 and assume $(C+G^{-1})^{-1}$ is L_{2e}-stable. If

$$\|R(C+G^{-1})^{-1}L^{-1}\|_2 < 1$$

then the closed loop system is stable for any operator in the sector (C,L,R).

Theorem 2-2 can be interpretted in classical frequency domain terms for the special case of a conic sector (C,R). Consider the closed loop operator

$$CG(I+CG)^{-1} = C(C+G^{-1})^{-1} = (I+(CG)^{-1})^{-1} \tag{2.30}$$

Theorem 2.2 shows that stability can be achieved for all operators in the conic sector (C,R) if the nominal closed loop response (2.30) of the feedback system is restricted to be small for all inputs

which have large normalized conic sector approximation errors, RC^{-1}. For linear, time-invariant operators, condition (2.13) can be used to express this in the frequency domain. The inequality of Theorem 2-2 is satisfied if

$$\|RC^{-1}\|_2 < \frac{1}{\|C(C+G^{-1})^{-1}\|_2} \tag{2.31}$$

or, in the frequency domain,

$$\bar{\sigma}[\hat{r}(j\omega)\hat{c}^{-1}(j\omega)] < \underline{\sigma}\,[I+(\hat{c}(j\omega)\hat{g}(j\omega))^{-1}] \qquad \forall\,\omega \tag{2.32}$$

Note that (2.32) imposes explicit magnitude constraints on the nominal closed loop frequency response. At all frequencies where the normalized error $r(j\omega)c^{-1}(j\omega)$ is large compared with unity, the inverse loop transfer matrix $(c(j\omega)g(j\omega))^{-1}$ must also be large. Hence, the loop transfer matrix itself must be small. Since $r(j\omega)c^{-1}(j\omega)$ typically grows large at higher frequencies, this constraint imposes explicit limitations on achievable crossover frequencies and closed loop bandwidths.

It will be helpful to compare the stability robustness condition of Theorem 2.2 with the performance specification (2.18). Stability robustness is ensured if a closed loop operator when "weighted" by R and L^{-1} has norm less than one. Performance is achieved if some closed loop error operator has "weighted" norm less than one. The same type of condition applies in both cases. Because the same condition is used for both a performance specification and stability robustness, we will be able to obtain a combined condition which can guarantee simultaneous robust performance and stability.

3.0 CONIC SECTORS FOR HYBRID COMPENSATORS

3.1 The Hybrid Compensator

A hybrid compensator consisting of both analog and digital elements is shown in Figure 3-1. Although it contains a digital computer, the hybrid compensator transforms an analog input signal to an analog output signal. In Section 3, we will show how to construct useful conic sectors which contain this operator. The hybrid compensator is introduced in 3.1. A fundamental result of Thompson [1] is given in 3.2 which enables us to place a stable compensator inside a cone. In 3.3, we show how to use this result for unstable compensators and how to reduce the radius of the cone. The approach is extended to multi-rate hybrid compensators in 3.4.

As shown in Figure 3-1, the hybrid compensator consists of a prefilter, a sampler, a digital computer, and a hold device. The prefilter F is a linear, time-invariant operator mapping the input e into a filtered signal e_f. It serves to reduce the high-frequency content of the signal to the sampler and thereby reduces the problems associated with aliasing. The sampler converts the analog signal e_f into the discrete time sequence e_d. It operates with a sample time T or sampling frequency $\omega_s = 2\pi/T$ and is considered ideal, i.e.

$$e_d(n) = e_f(nT) \tag{3.1}$$

for any integer n. The operator D associated with the digital computer is a linear, shift-invariant mapping of e_d into u_d. The hold device transforms the discrete-time sequence u_d back into an analog output u.

An equivalent analog representation of the hybrid compensator is shown in Figure 3-2. Here the ideal sampling operation is represented as modulation of the signal $e_f(t)$ by an infinite train of impulses. The *-notation is used to deonte the analog representation of the corresponding discrete-time signal. It is well-known that modulation by an impulse train is equivalent to convolution by an impulse train in the frequency-domain, i.e.

$$\hat{e}_d^*(j\omega) = \frac{1}{T}\sum_n \hat{e}_f(j\omega - jn\omega_s) \tag{3.2}$$

The analog signals $e_d^*(t)$ and $u_d^*(t)$ are related to their discrete-time counterparts by

$$e_d^*(t) = \sum_n e_d(n)\,\delta(t-nT) \tag{3.3}$$

and

$$u_d^*(t) = \sum_n u_d(n)\,\delta(t-nT) \tag{3.4}$$

The Fourier Transform of e_d^* is related to the Z transform of e_d by

$$\hat{e}_d^*(j\omega) = \hat{e}_d(z) \mid_{z=e^{j\omega T}} \tag{3.5}$$

The digital computer is now represented as the linear, time-invariant operator D*. It is easily verified that (3.4) requires D* to be defined as

$$\hat{d}^*(j\omega) = \hat{d}(z) \mid_{z=e^{j\omega T}} \tag{3.6}$$

The hold operator is simply a linear, time-invariant mapping of analog signals. By abuse of notation, we use the same symbol H to represent the hold device in both figures. The representation of the hybrid compensator in Figure 3.2 will be adopted in the sequel.

The hybrid compensator will be denoted by the linear operator K. It maps e into u according to the time varying convolution

$$u(t) = \int_{-\infty}^{+\infty} k(t,\tau)e(\tau)d\tau \tag{3.7}$$

The operator K is time-varying because of the sampling operation. Thus it cannot be represented in the frequency-domain as a transfer function. It is possible, however, to relate the Fourier Transforms of e and u, viz.

$$\hat{u}(j\omega) = \frac{1}{T}\hat{h}(j\omega)\hat{d}^*(j\omega)\sum_n \hat{f}(j\omega - jn\omega_s)\hat{e}(j\omega - jn\omega_s) \tag{3.8}$$

This equation is discussed at length by Franklin and Powell [12]. It comes from the concatenation of the operations in Figure 3-2. It is used extensively in 3.2 to construct a conic sector containing K.

3.2 Construction of a Conic Sector that Contains a Hybrid Compensator

In his recent PhD dissertation, Thompson [1] shows that a stable hybrid operator K can be placed inside a computable conic sector (C,R). This allows the sector concepts discussed 2.3 to be applied to such compensators. We now review and discuss Thompson's result.

Theorem 3.1 [1] Consider an L_{2e}-stable hybrid operator K, and let C, R, R^{-1} be linear, time-invariant, L_{2e}-stable operators. Under these conditions, K is in the cone (C,R) if

$$\underline{\sigma}\,[\hat{r}(j\omega)] > [r_1(j\omega)+r_2(j\omega)]^{1/2} \qquad \forall\omega \tag{3.9}$$

where

$$r_1(j\omega) = \sum_m \sum_{n=m} \left\| \frac{1}{T}\hat{h}(j\omega+jm\omega_s)\hat{d}^*(j\omega)\hat{f}(j\omega+jn\omega_s) \right\|_2^2$$

$$r_2(j\omega) = \sum_m \left\| \frac{1}{T}\hat{h}(j\omega+jm\omega_s)\hat{d}^*(j\omega)\hat{f}(j\omega+jm\omega_s) - \hat{c}(j\omega+jm\omega_s) \right\|_2^2$$

This significant theorem is proved in Appendix A. It says that any hybrid compensator can be placed inside some appropriate conic sector. The utility of this fact depends on the size of the radius, cf. Theorem 2-2. Note that r_1 and r_2 are periodic with period ω_s. Given a center, condition (3.9) imposes a magnitude constraint on the radius R, and it is always possible to choose R so that (3.9) is satisfied with equality.

Altnough any stable, linear, time-invariant operator can be used as tne center in the theorem, only tnose centers that result in a "small" radius are useful. From the discussion in 2.3 on conic sectors, the center represents an approximation to the hybrid operator. One very useful choice for the center is

$$\hat{c}(j\omega) = \frac{1}{T}\hat{h}(j\omega)\,\hat{d}^*(j\omega)\hat{f}(j\omega) \tag{3.10}$$

This center makes $r_2(j\omega) = 0$ and so is referred to as the optimal center. Since r_1 is independent of the center, the choice (3.10) results in the cone with the smallest radius. An alternate choice for the center is

$$\hat{c}(j\omega) = \frac{1}{T}\hat{h}(j\omega)d_a(j\omega)\hat{f}(j\omega) \tag{3.11}$$

where $d_a(j\omega)$ represents some desired analog compensation.

It is noteworthy that given a center, the optimal radius is computable. Thompson [1] discusses the convergence properties associated with the defining sums for r_1 and r_2. He also provides analytic solutions for r_1 in important special cases.

The conic sector representation of a hybrid operator allows the application of the stability tests of Section 2.3. Consider a plant G in a negative feedback configuration with hybrid compensator K. Theorem 2-2 can be applied to guarantee stability of the feedback system. The theorem assumes that the nominal closed loop operator $(C+G^{-1})^{-1}$ is stable. Stability is maintained provided

$$\bar{\sigma}\{\hat{r}(j\omega)\hat{c}^{-1}(j\omega)[1+(\hat{c}(j\omega)\hat{g}(j\omega))^{-1}]^{-1}\} < 1 \quad \forall\omega \tag{3.12}$$

From a design viewpoint, the analog center C is constructed to achieve closed-loop stability. The digital implementation maintains stability if the normalized error RC^{-1} is sufficiently small.

It is important to recognize that stability of the nominal analog closed-loop operator is not equivalent to stability of the nominal discrete time closed-loop operator. This is illustrated by the example systems shown in Figure 3-3. The Nyquist plots for these two systems are shown in Figure 3-4.

The discrete time system is stable, while the continuous-time system is unstable. The stabilizing effect of the sampling operation can be attributed to the fact that aliasing of the plant contributes phase lead near crossover, see Figure 3-4. Tou [13] discusses how a pure time delay can have a stabilizing impact on a discrete time system because of favorable phase interactions under aliasing. This phenomenon depends critically on the phase characteristics of the plant at frequencies above the half sample frequency. For usual aerospace applications, the phase of the plant is quite uncertain at such frequencies. Thus we feel that the requirement of Theorem 3-1 for stability of the nominal analog closed-loop operator is not overly restrictive for aerospace applications.

The simple example of a lead compensator is used to illustrate the theorem. Consider a desired analog compensator

$$d_a(j\omega) = \frac{10(j\omega+0.1)}{j\omega+1} \tag{3.13}$$

We first define a hybrid compensator which implements this lead and then compute the center and radius of a cone which contains it. The hybrid compensator is (somewhat arbitarily) defined as follows:

$T = 0.6283$ sec.

$\omega_s = 10$ rad/sec.

$\hat{f}(j\omega) = \frac{5}{j\omega+5}$

$\hat{d}(z) = 7.66\,\frac{z-0.9391}{z-0.5335}$

$\hat{h}(j\omega) = \frac{1-e^{-j\omega T}}{j\omega T}$

The prefilter is a single pole at $0.5\omega_s$. The hold device is just a zero order hold. The digital compensation was computed from the desired lead (3.13) by the pole-zero mapping technique [12].

The optimal center C_0 for this example is

$$\hat{c}_0(j\omega) = \frac{1}{T}\,\hat{h}(j\omega)\,\hat{d}(e^{j\omega})\,\hat{f}(j\omega) \tag{3.14}$$

For comparison purposes, we introduce an alternate center C_a as

$$\hat{c}_a = \frac{1}{T}\,\hat{h}_a(j\omega)\,\hat{d}_a(j\omega)\,\hat{f}(j\omega) \tag{3.15}$$

where $h_a(j\omega)$ is the first order approximation to the zero order hold,

$$\hat{h}_a(j\omega) = \frac{3.183}{s+3.183} \tag{3.16}$$

The center C_a is only third order, yet fairly close to the optimal center. Figure 3-5 shows the Bode plots of the desired analog compensation D_a and the two centers C_0 and C_a. Both centers show appreciably more phase lag than D_a above (say) 0.3 rad/sec. The radius R and the quantities r_1 and r_2 were computed according to Theorem 3-1 using center C_a. These quantities are plotted in Figure 3-6. Recall that r_1 is the radius associated with the optimal center C_0. Notice that there is little difference between r_1 and R above 1 rad/sec.

The normalized error RC_a^{-1} is plotted in Figure 3-7. Also shown in the figure are the normalized errors when D is computed from the forward rectangular rule, the backward rectangular rule, and Tustin's rule prewarped about 0.5 rad/sec. The differences do not appear significant. Notice that in each case, the normalized error is greater than one at every frequency. From stability robustness condition (2.31), it is not possible to achieve large loop transfer gains using this hybrid compensator. Increasing the sample rate, as shown in Figure 3-8, does lower the normalized error below one over a significant frequency range.

3.3 Techniques to Increase the Applicability of the Conic Sector

Theorem 3.1 determines a conic sector which contains a stable hybrid operator. Two shortcomings of the theorem as it stands are

(i) the hybrid operator must be stable, e.g. hybrid operators with integral action cannot be handled,

(ii) directionality information associated with multiloop compensators is lost in constructing the radius, i.e. stability tests using the conic sector will tend to be conservative in cases where the digital operator D has a large spread in its singular values.

This section shows how to exploit the inherent flexibility of Theorem 3.1 to overcome these two shortcomings.

The first topic to be treated is obtaining a conic sector representation that is useful with unstable hybrid compensators and has a "small" radius. Recall the analog representation of the hybrid compensator shown in Figure 3-2. F, D*, and H are all linear, time-invariant operators -- only the sampling operation is time-varying. Our approach is to obtain a conic sector that includes the sampling but excludes the other components of the hybrid compensator.

Consider the equivalent analog representation of the hybrid compensator shown in Figure 3-9. The operator A introduced in the figure is linear, time-invariant with no poles or zeros in the right-half-plane. Let the operator S denote the operator inside the dashed box in Figure 3-9, i.e. the concatenation of A, modulation by the impulse train, followed by A. We place the sampling operator S inside a conic sector according to Theorem 3-1 and use the freedom offered by A to reduce the normalized error of the cone.

Corollary 3.1 Let R and R^{-1} be linear, time-invariant, L_{2e}-stable operators.

Then S is in the cone $(\frac{1}{T} A^2, R)$ if

$$\underline{\sigma}\,[\hat{r}(j\omega)] > \left[\sum_{m}\sum_{n=m} \left\| \frac{1}{T}\, \hat{a}(j\omega+j\omega_s m)\hat{a}(j\omega+j\omega_s n)\right\|_2^2\right]^{1/2} \forall\, \omega \tag{3.17}$$

This corollary is a straightforward application of Theorem 3.1. It handles the situation where D* is unstable by excluding the digital compensation from the cone. For analysis purposes, the prefilter, digital compensator, and hold (as well as the two A^{-1} operators) are included with the plant. Let G denote the plant. Theorem 2-2 can be applied if the nominal closed-loop operator is stable. This operator is simply

$$(\tfrac{1}{T} A^2 + [A^{-1}FGHD^*A^{-1}]^{-1})^{-1} = TA^{-2}(I + [TFGHD^*]^{-1})^{-1} \tag{3.18}$$

Stability is maintained provided

$$\| R\,(\tfrac{1}{T}A^2 + [A^{-1}FGHD^*A^{-1}]^{-1})^{-1} \|_2 < 1 \tag{3.19}$$

or

$$\| RTA^{-2}\,(I + [TFGHD^*]^{-1})^{-1} \|_2 < 1 \tag{3.20}$$

Notice that the conic sector being used has center and radius that are essentially independent of the compensator and the plant being controlled. Only the sample time T and sampling frequency ω_s come into play in the definition of the cone. The operator A can be chosen to try to minimize the normalized error RTA^{-2}. More precisely, we are free to choose A to achieve (3.20) as long as stability of (3.18)

is not compromised. Since A has no right-half-plane zeros, the only way stability of could be compromised is if A rolls off too fast as a function of frequency and so A^{-2} would "roll up" too fast. This is not a major concern, just something that must be checked. If A has first order roll off characteristics, then A^{-2} is guaranteed to be acceptable.

Some reasonable choices for A are now illustrated by example. From (3.20), the objective is to make RTA^{-2} small. The examples are normalized by taking ω_s = 1/sec and so T = 2π sec. The first candidate operator is

$$A = \frac{1}{s+\epsilon} \tag{3.21}$$

As ϵ approaches zero, it can be shown analytically that the normalized error becomes

$$\hat{r}(j\omega)\hat{T}a^{-2}(j\omega) = \pi\sqrt{2/3}\,\omega \tag{3.22}$$

Notice that in this case the normalized error reaches 1 at about 0.39 rad/sec - not far from the half sample frequency. This behavior would be expected to be quite adequate in cases where the closed loop system rolls off significantly before the half sample frequency. More general candidate operators are

$$A = \frac{1}{(s+\epsilon)(\tau s+1)} \tag{3.23}$$

and again we examine the behavior as ϵ approaches zero. No general analytic results are available in this case, but Figure 3-10 shows the normalized error for τ = 0,1,10. The τ = 0 case is just the first candidate (3.21). These three curves show a tradeoff between high and low frequency behavior. As τ increases, the normalized error decreases for low-frequencies at the expense of increases at high frequencies. These choices for A appear to be useful, but their relative utility would depend on the particular application, cf. (3.20).

An example shows that the choice reflected in (3.21) cannot be uniformly improved. Let FGHD* be defined as

$$\frac{100\pi s+100}{100\pi\alpha s^2+(\pi+100)\alpha s+\alpha-100}$$

The closed loop system is stable for α=2.4 but unstable for α=2.3. Figure 3-11 is a graphical version of the stability test (3.20). The plot shows $|(1 + (TFGHD^*)^{-1})^{-1}|$ and $\pi\sqrt{2}/3\omega$ for α=2.3. This shows that condition (3.20) is violated, and thus stability cannot be guaranteed. Since the closed loop system is in fact unstable, there is little conservatism in the test. Hence, the suggested choice of A results in a normalized error that cannot be uniformly improved.

The above Corollary 3-1 applies to single or multiple loop hybrid compensators. The approach of placing just the sampling operation inside a conic sector does preserve the directionality information of the digital compensation. There is additional freedom in the multi-loop case, however, to enhance the stability test (3.20).

Figure 3-12 is an equivalent representation of the dashed box in Figure 3-9. The operators Q and Q^{-1} are linear, time-invariant and L_{2e}-stable. Also, the transfer matrices of Q and Q^{-1} are periodic with period ω_s. We intend to place the operator shown in the dashed box of Figure 3-12 inside a conic sector. Suppose A is restricted to be a scalar times the identity. Then Q commutes with A and the modulation by the impulse train. This means that Q and Q^{-1} inside the dashed box cancel each other, and so the conic sector of Corollary 3.1 still applies.

The net result of introducing Q and Q^{-1} is that stability condition (3.20) becomes

$$\| Q\,\{RTA^{-2}(I+[TFGHD^*]^{-1})^{-1}\}\,Q^{-1} \|_2 < 1 \tag{3.24}$$

Thus the operator in (3.20) is preceded by Q^{-1} and followed by Q. One interpretation is that Q introduces a particular weighting on the original cone. The additional freedom offered by Q expands the class of plants that can be guaranteed stable using conic sectors. The question of how to choose the right Q will be discussed in Section 5. For now, we simply note that the existence of a single Q so that (3.24) is satisfied is enough to guarantee stability.

3.4 Multi-Rate Hybrid Compensators

It is straightforward to extend the conic sector approach for hybrid compensators to the general case of multiple independent hybrid compensators having arbitrary sample rates. Each individual compensator can be placed in a cone, and then a composite cone can be obtained for the overall multi-rate compensator. This composite cone is suitable to use with Theorem 2-2 to assure stability.

A general multi-rate hybrid compensator employing k independent sample rates is shown in Figure 3-13. Following the development of 3.3, we place each sampler inside a conic sector. Thus the operators A_1, A_2 ..., A_k are introduced and the radii R_1, R_2 ..., R_k are given according to Corollary 3.1. Also, the weighting operators Q_1, Q_2, ..., Q_k having the appropriate periodicity are introduced. For notational simplicity, we define the following block diagonal operators:

$$F_M = \text{diag}\ \{F_1, F_2, \ldots, F_k\}$$

$$D_M^* = \text{diag}\ \{D_1^*, D_2^*, \ldots, D_k^*\}$$

$$H_M = \text{diag}\ \{H_1, H_2, \ldots, H_k\}$$

$$T_M = \text{diag}\ \{T_1 I_1, T_2 I_2, \ldots, T_k I_k\}$$

$$A_M = \text{diag}\ \{A_1, A_2, \ldots, A_k\}$$

$$Q_M = \text{diag}\ \{Q_1, Q_2, \ldots, Q_k\}$$

$$R_M = \text{diag}\ \{R_1, R_2, \ldots, R_k\}$$

where I_1, I_2, ..., I_k are identity operators of appropriate dimension.

Given these preliminaries, Theorem 2-2 can be used to guarantee stability of the closed-loop system in Figure 3-13. The nominal closed-loop operator is

$$T_M A_M^{-2} (I + [T_M F_M G H_M D_M^*]^{-1})^{-1} \tag{3.25}$$

If this operator is stable, then stability will be maintained with the multi-rate hybrid compensator provided

$$\| Q_M \ R_M T_M A_M^{-2} (I + [T_M F_M G H_M D_M^*]^{-1})^{-1} \ Q_M^{-1} \|_2 < 1 \tag{3.26}$$

Equations (3.25) and (3.26) are direct analogs of (3.18) and (3.24) in the case of a single hybrid compensator. The block elements of Q_M are periodic, but they are otherwise free and can be chosen to minimize the operator norm in (3.26). We postpone the discussion of the appropriate choice of these elements of Q_M until Section 5.

Summarizing this section, conic sectors have been found for single and multiple sample rate hybrid operators. This permits a hybrid compensator to be analyzed as an analog compensator with the resulting, quantified approximation error. Provided the analog compensator gives closed-loop stability, conditions (3.24) and (3.26) can be used to guarantee stability of the closed-loop system with the hybrid compensator. The real significance of using conic sectors for hybrid compensators will be shown in Section 5. There we will obtain conditions which guarantee stability and performance for a uncertain plant controlled by a hybrid compensator.

4.0 FEEDBACK ANALYSIS AS A BLOCK-DIAGONAL BOUNDED PERTURBATION PROBLEM

4.1 Robustness Characterization

This section formulates the basic feedback problem of achieving performance in the face of uncertainties as a stability problem in the presence of block-diagonal bounded perturbations [4]. The formulation involves sector-bounded transfer functions as basic building blocks. The robustness and

performance properties of a feedback system will be expressed in terms of a collection of linear, time-invariant operators, Δ_i , i=1,2,...,m, each of which is an element of a sector with zero center, i.e. $\Delta_i \epsilon (0,L_i,R_i)$. These operators are the basic building blocks in a combined robustness/performance characterization of feedback systems.

The use of sector bounded, linear, time-invariant operators to characterize robustness has been a central theme in many recent references, including [14] where such operators were inserted at the inputs or outputs of a plant model in order to represent so called unstructured uncertainties (modeling errors with no assumed structure except for known magnitude bounds on their transfer functions). Necessary and sufficient conditions such as Theorem 2-2 were then derived for stability robustness in the face of such uncertainties. For example, a stable feedback loop with plant G and compensator K will remain stable in the face of all possible perturbed plants G' = [I + Δ]G, with $\Delta \epsilon (0,L,R)$, if and only if

$$\bar{\sigma}[RGK(I+GK)^{-1}L^{-1}] < 1 \quad \forall \omega \tag{4.1}$$

Note that with R and L specified, this inequality imposes conditions on the shape of the closed loop frequency response, $GK(I + GK)^{-1}$, which must be satisfied in order to assure robust stability. These conditions are unique to the assumed form of plant perturbations (e.g., G' = (I + Δ)G in the present case). Each such assumed form corresponds to a specific location where Δ is inserted in the nominal feedback loop. The location for our present case is shown in Row 1 of Table 4.1. Other locations correspond to other assumed forms for G' and produce different necessary and sufficient stability robustness conditions. A representative set of possibilities is summarized in the remaining rows of Table 4.1. (Most of these cases can be found in [15]).

Table 4.1 also indicates representative types of physical uncertainties which can be usefully represented by sector bounded perturbations inserted at the indicated locations. For example, the representation G' = (I + Δ)G in Row 1 is useful for output errors at high frequencies, covering such things as unmodelled high frequency dynamics of sensors or plant, including diffusion processes, transport lags, electro-mechanical resonances, etc. The representation G' = G(I + Δ) in Row 2 covers similar types of errors occurring at the inputs. Both cases should be contrasted with Rows 4 and 5 which treat $G' = (I + \Delta)^{-1}G$ and $G' = G(I + \Delta)^{-1}$. These representations are more useful for variations in modelled dynamics, such as low frequency errors produced by parameter variations with operating conditions, with aging, or across production copies of the same plant. Discussion of still other cases is left to the table. Note from the table that the stability requirements on Δ do not limit our ability to represent variations in either the number or location of rhp singularities.

The most significant thing to understand about Table 4.1 is that the stability robustness conditions shown are sufficient to assure stability only if all the uncertainties occur at the indicated locations and none occur elsewhere. In order to use the conditions directly, therefore, designers are obliged to reflect all known sources of uncertainty from their known point of occurrence to a single reference location in the loop. Such reflected uncertainties invariably have a great deal of structure which must then be "covered up" with a larger, arbitrarily more conservative perturbation in order to maintain a simple cone bounded representation at the reference location.*

Alternatively, designers could choose to treat uncertainties occurring at several different locations in the feedback loop as a single uncertainty occurring at one location in a larger feedback loop. To be specific about this alternative, let Δ_i, i=1, 2, ..., m, denote a collection of such uncertainties positioned at location ℓ_i, i=1, 2, ..., m. Note that at each ℓ_i, the feedback loop has an input, where it receives the signals from Δ_i, and also an output, where it supplies signals to Δ_i. Let M_{ii} be the operator between these two sets of signals. Further, let M_{ij} denote the operator between the inputs at location ℓ_j and the outputs at location ℓ_i. Then the block- structured operator

$$M \triangleq (M_{ij}) \tag{4.2}$$

* By "arbitrarily more conservative," we mean that examples can be constructed where the degree of conservatism is arbitrarily large. Of course, other examples exist where it is quite reasonable.

represents all interactions of the feedback loop with its uncertainties, and indeed, the block-diagonal bounded perturbation diagram in Figure 4.1 is an equivalent representation of the loop. Here we have $\Delta = \text{diag}(\Delta_1, \Delta_2, \ldots, \Delta_m)$.

Note that the feedback elements in this larger loop are zero in the absence of uncertainties. Hence, M will be a stable "plant" whenever the original nominal loop is stable. As an example of this representation, consider the system in Figure 4.2. This system, with two uncertainties present simultaneously, the first from Row 2 and the second from Row 4 of Table 4.1, is described by the following M operator:

$$M = \begin{bmatrix} (I + KG)^{-1}KG & (I + KG)^{-1}K \\ (I + GK)^{-1}G & (I + GK)^{-1} \end{bmatrix} \tag{4.3}$$

Given the equivalent system in Figure 4.1 with sector bounded Δ_i, it follows from the Small Gain Theorem that the loop remains stable in the presence of these uncertainties if

$$\|RML^{-1}\|_2 < 1$$

or, in the frequency domain,

$$\bar{\sigma}\,[R(j\omega)M(j\omega)L(j\omega)^{-1}] \; < 1 \;\forall\, \omega \geq 0 \tag{4.4}$$

where $R = \text{diag}(R_1\ R_2\ \ldots\ R_m)$ and $L = \text{diag}(L_1\ L_2\ \ldots\ L_m)$. This condition provides an alternate test for stability robustness. Like the procedure of reflecting all uncertainties to one reference location, however, the new test can be arbitrarily more conservative because it ignores the known block-diagonal structure of the uncertainties in Figure 4.1.

One of the objectives of the results in this paper is precisely to reduce the conservatism of robustness and performance tests for block diagonal structures such as Figure 4.1. We do this by introducing a generalized notion of the maximum singular value for block-diagonal structures. This generalization is developed in Section 5. It is called the structured singular value (SSV) and is denoted by the symbol μ. It yields the following necessary and sufficient conditions for robust stability of the BDBP problem:

$$\mu[R(j\omega)M(j\omega)L^{-1}(j\omega)] < 1 \qquad \forall\, \omega \tag{4.5}$$

This represents our extension of the Small Gain Theorem which we call the Small μ Theorem.

Since all simultaneous uncertainties can be put into block-diagonal form by merely constructing the associated operator M, the SSV allows us to nonconservatively analyze simultaneous occurences of uncertainties anywhere in a feedback system. The uncertainties may be sector bounded errors of individual components of the system (SISO or MIMO), they may be individual parameter variations in the model, or even polynomial approximations of parameters entering nonlinearly. In fact, the only restrictions which remain is that all variations must be allowed to be complex. Pure real variations or pure imaginary variations cannot be separated into individual blocks.

4.2 Performance Characterization

The ability to treat simultaneous, structured uncertainties also offers, almost as a free byproduct, the ability to deal simultaneously with the performance and robustness aspects of feedback. As discussed in 2.2 and 2.4, the test for satisfaction of a performance specification is identical to a test for robustness with respect to some uncertainty. That is, any performance specification has a corresponding robustness requirement such that one is satisfied if and only if the other is. Thus the SSV tests for robustness can be used directly to evaluate performance.

The equivalence between performance and robustness is elaborated in Column 4 of Table 4.1, where each of the conditions imposed on feedback loop shapes by perturbation Δ_i at location ℓ_i is given a performance interpretation. For example, the perturbations in Row 4 impose requirements (through L and R) on the operator $(I + KG)^{-1}$. This operator is, of course, the classical (output) sensitivity function of the feedback loop. Small values over some frequency range guarantee low closed

loop sensitivity to open loop variations and low command following errors to output commands over that range. A particular specification on these performance parameters can thus be thought of as imposed on a design by introducing a "fictitious uncertainty" at the location in Row 4 with sector bounds R and L selected to meet the performance requirement.

To illustrate how such fictitious uncertainties actually enforce performance specs, consider the simple case where a single true uncertainty, say Δ_r from Row 2, and a single fictitious (performance) uncertainty, say Δ_p from Row 4, are specified for our feedback system. Let the structured singular value condition (4.5) be satisfied for the corresponding M matrix (equation 4.3). Then the system remains stable in the face of Δ_r and Δ_p occurring simultaneously. Obviously, it will also remain stable for Δ_p with $\Delta_r = 0$. This means that the nominal system must satisfy the performance condition

$$\bar{\sigma}[R_p(I + KG)^{-1}L_p^{-1}] < 1 \qquad \forall\, \omega \tag{4.6}$$

because the latter is also a necessary and sufficient condition for robust stability with Δ_p only. This much is straightforward. What is not so evident but much more important is that Condition (4.6) is also satisfied for all perturbed feedback loops. That is, for all true plants $G' = G(I + \Delta_r)$ we have

$$\bar{\sigma}[R_p(I + KG')^{-1}L_p^{-1}] < 1 \qquad \forall\, \omega \tag{4.7}$$

Hence, the performance spec is satisfied in the face of all possible true uncertainties. A proof of this consequence of the structured singular value condition is left to Section 5.

5. STRUCTURED SINGULAR VALUE ANALYSIS OF FEEDBACK SYSTEMS

5.1 Introduction to the Structured Singular Value

We have discussed how the problem of analyzing performance in the face of structured uncertainty can be expressed as a BDBP problem. The standard singular value tests applied to the BDBP can be excessively conservative because they ignore the block diagonal structure. A more general non-conservative test (the Small μ Theorem) is developed in this section which removes this limitation. By non-conservative we mean providing a necessary and sufficient condition. The test is expressed in terms of a new measure, the structured singular value μ. This section begins with review of the results in [5] where μ was introduced.

To provide a more precise description of block diagonal perturbations, let $K = (m_1, m_2, \ldots, m_n, k_1, k_2, \ldots, k_n)$ be a 2n-tuple of positive integers. All the definitions that follow depend on K, but to simplify notation this dependency will not be explicitly represented. Let

$$k = \sum_{j=1}^{n} m_j k_j \quad \text{and} \quad m = \sum_{j=1}^{n} m_j.$$

Let $\mathbf{X}$ be a set of kxk, rational, block-diaginal matrices defined by

$$\mathbf{X} = \{\mathrm{diag}(\overbrace{\Delta_1, \Delta_1, \cdots, \Delta_1}^{m_1}, \overbrace{\Delta_2, \Delta_2, \cdots, \Delta_2}^{m_2}, \Delta_3, \cdots, \Delta_{n-1}, \overbrace{\Delta_n, \Delta_n, \cdots, \Delta_n}^{m_n}) \mid \text{for each } j=1, 2, \cdots, n,\ \Delta_j \text{ is a } k_j x k_j \text{ matrix}\}$$

Let $\mathcal{U}$ be the set of block diagonal unitary matrices, and $\mathcal{D}$ the set of real diagonal matrices such that

$$\mathcal{D} = \{\mathrm{diag}(d_1 I_k, d_2 I_k, \ldots d_{m_1} I_{k_1}, d_{m_1+1} I_{k_2}, \ldots, d_m I_{k_m}) \mid d_i \in R^+(0, \infty)\} \tag{5.2}$$

What is desired is a function (depending on K)

$$\mu : \mathcal{M}(k) \to [0, \infty) \tag{5.3}$$

with the property that $\forall$ M

$$\det(I+M\Delta) \neq 0 \ \forall\, \Delta \in \mathbf{X},\ \bar{\sigma}(\Delta) < \delta \quad \text{iff} \quad \delta\mu(M) < 1 \tag{5.4}$$

This could be taken as a definition of μ. Alternatively, μ could be defined as

$$\mu(M) = \begin{cases} 0 & \text{if no } \Delta\in X \text{ solves } \det(I+M\Delta) = 0 \\ \min\limits_{\Delta\in X} \{\bar\sigma(\Delta) \mid \det(I+M\Delta) = 0\}^{-1} & \text{otherwise} \end{cases} \tag{5.5}$$

This definition shows that a well-defined function satisfies (5.4). It probably has little additional value since the optimization problem involved does not appear to have useful properties.

Using these definitions, the following useful properties of μ are easily proven.

1) $\mu(\alpha M) = |\alpha|\mu(M)$ $\quad \forall\, M\in\mathcal{M}(k)$
2) $\mu(I) = 1$
3) $\mu(AB) \le \sigma(A)\mu(B)$ $\quad \forall\, A,B\in\mathcal{M}(k)$
4) $\mu(\Delta) = \sigma(\Delta)$ $\quad \forall\, \Delta\in X$
5) If $n=1$ and $m_1=1$ then $\mu(M) = \sigma(M)$ $\quad \forall\, M\in\mathcal{M}(k)$
6) If $n=1$, $k_1=1$, then $k=m_1$, $X = \{\lambda I \mid \lambda\in R\}$ and $\mu(M) = \rho(M)$ $\quad \forall\, M\in\mathcal{M}(k)$
7) If $\Delta\in X$, $U\in\mathcal{U}$ then $U\Delta\in X$ and $\Delta U\in X$ and $\bar\sigma(U\Delta)=\bar\sigma(\Delta U)=\bar\sigma(\Delta)$
8) $\forall\, \Delta\in X$ and $\forall\, D\in\mathcal{D}$ $\quad D\Delta D^{-1}=\Delta$
9) $\forall\, U\in U$ and M $\quad \mu(MU) = \mu(UM) = \mu(M)$
10) $\forall\, D\in$ and M $\quad \mu(DMD^{-1}) = \mu(M)$
11) $\max\limits_{U\in\mathcal{U}} \rho(UM) < \mu(M) \le \inf\limits_{D\in\mathcal{D}} \bar\sigma(DMD^{-1})$ $\quad \forall\, M\in\mathcal{M}(k)$

Properties 5) and 6) show that the structured singular value has as special cases both the spectral radius and the maximum singular value. Property 9) means that μ is $\mathcal{U}$-invariant.

The most important results from [5] are the following, which deal with the bounds in property 11):

a) The left-hand-side inequality in 11) is always an equality. This expresses μ in familiar linear algebraic terms, but the optimization problem involved may have multiple local maxima.

b) The right-hand-side inequality in 11) is an equality when there are three or fewer blocks, and the blocks are not repeated. The blocks themselves, and therefore M, may be of arbitrarily large dimension. A tedious but straightforward computation shows that the optimization problem involved is always convex [16]. Furthermore, the minimization is over only n-1 parameters for n blocks, independent of block size, making this an attractive alternative to a).

Note that the transformation DMD^{-1} is simply a rescaling of the inputs and outputs of M. The SSV is invariant with respect to such rescaling (property 10), while singular values do, of course, vary with rescaling. This implies, for example, that the ad hoc method of performing a change of units can reduce the conservatism associated with singular values. For some time we have been using Osborne's technique [17], which minimizes the Frobenius norm of DMD^{-1} to compute frequency-dependent D matrices. We now have new algorithms which compute D to directly minimize $\sigma(DMD^{-1})$.

While the strongest results on the use of the DMD^{-1} scaling are for nonrepeated blocks, the use of scaling before computing singular values can be quite effective in treating repeated block problems. These typically arise when analyzing robustness with respect to variations in scalar parameters occurring in several places within a system. This leads naturally to problems where the block-diagonal perturbations are made of either nonrepeated matrix blocks or repeated scalars. In this case the set x takes the special form:

$$X = \{\text{diag}\,(\lambda_1 I, \lambda_2 I, \cdots \lambda_{h_1} I, \Delta_1, \Delta_2, \cdots \Delta_{n_2})$$

Suppose now we let

$$D = \{\text{diag}\,(D_1, D_2, \cdots D_{n_1}, d_1 I, d_2 I, \cdots d_{n_2} I)\} \tag{5.6}$$

where the structure of D matches X and the D_i are full block matrices and the d_i are real scalars. Then an upper bound for μ with this structure is again

$$\mu(M) \leq \inf_{D \in \mathcal{D}} \bar{\sigma}(DMD^{-1})$$

The use of full block D_i's improves the bound and a simple argument shows that the problem is still convex. Of course, the scaling could be extended to handle repeated nonscalar blocks, but these have not yet been bound to be applicable to system problems.

Numerical software for computing μ has been developed using algorithms based on these results. In addition to using this software to analyze some simple feedback designs, test runs have been made on a large number of psuedo-random matrices. It appears that the global maximum in a) is often easily found, although a simple gradient search is inadequate. Also, the bound obtained in b) appears to be quite good (to within 15%) for cases of more than 3 blocks. These observations are most encouraging, especially considering the experimental and preliminary nature of the software.

There are essentially two direct applications of singular values to the BDBP problem, which provide bounds for μ:

1) Ignore the block diagonal structure and compute $\bar{\sigma}(M)$. This gives an upper bound for μ.

2) Treat each perturbation one at a time. Compute the largest maximum singular value for each of the corresponding diagonal blocks. This gives a lower bound for μ.

The gap between these two bounds may be arbitrarily large.

An extension to 1) was proposed by Lehtomaki ([15],[18]), who uses the singular vectors for $\sigma(M)$ to sharpen the bound. Lehtomaki's method checks for structure but not in the BDBP form. The optimism of 2) can be reduced by using a method suggested by Freudenberg, et al [19], who evaluate the differential sensitivity of the singular values at one point with respect to perturbations at another. Although this method does not apply to simultaneous, large perturbations, it can be quite useful in indicating when the lower bound for μ obtained by method 2) is optimistic. It should be mentioned that Lehtomaki and Freudenberg did not present their techniques in the context of the BDBP problem.

5.2 Robustness Analysis: The Small μ Theorem

The preceding discussion of μ and the BDBP problem has dealt with determining the size of the minimum structured perturbation Δ that causes $I + M\Delta$ to be nonsingular. We are interested in using the structured singular value to answer robustness, sensitivity, and performance questions for multivariable feedback systems. The connection between μ and these essential feedback properties is provided by the Small μ Theorem, which characterizes the stability robustness properties of a feedback system with respect to block diagonal perturbations. In order to state the Small μ Theorem we need the following additional definitions depending on K:

Let $L, R \in X$ be such that L and R have no poles or zeros in the open right-half-plane. Then let

$$\mathcal{X} = \{L^{-1}\theta R \mid \theta \in X \text{ and } \bar{\sigma}(\theta(s)) \leq 1 \quad \forall \operatorname{Re}(s) > 0\} \tag{5.7}$$

For the BDBP problem in Figure 4.1, $\mathcal{X}$ is the set of allowable block diagonal perturbations, and L and R are the weightings for the Δ such that $\bar{\sigma}(L\Delta R^{-1}) \leq 1$. We will say the canonical system in Figure 4.1 is stable iff $I+M\Delta$ is nonsingular in the closed right-half-plane. Although this definition does not distinguish between ill-posedness and instability, it is adequate for our purposes. We can now state and prove the following:

Theorem 5.1 (Small μ): The canonical system is closed loop stable for all $\Delta \in \mathcal{X}$ iff

$$\mu_c = \sup_{\omega} \mu(RML^{-1}) < 1 \tag{5.8}$$

Proof: To prove the if part, suppose $\mu_c < 1$ and let $\Delta \in X$. Then using Properties 3) and 11) and the definition of X

$$\sup_{\mathrm{Re}s\geq 0} \rho(M\Delta) = \sup_{s=j\omega} \rho(M\Delta) \leq \sup_{s=j\omega} \mu(RML^{-1}) = \mu_c < 1.$$

Thus $I + M\Delta$ is nonsingular for all Res > 0. Since Δ was arbitrary, the canonical system is stable for all $\Delta \in \mathbf{X}$.

Conversely, suppose $\mu(RML^{-1})\big|_{\omega=\omega_0} \geq 1$ (ω_0 may be ∞). Then $\exists\ \theta \in \mathbf{X}$ such that $\bar{\sigma}(\theta) \leq 1$ and

$\det(I + RML^{-1}\theta)_{\omega=\omega_0} = 0$. Thus, $\exists\ \Delta \in \mathbf{X}, \ni \det(I + M\Delta)\big|_{\omega=\omega_0} = 0$ and the canonical system is not stable for all $\Delta \in \mathbf{X}$. □

This theorem guarantees that if $\mu(RML^{-1})$ is less than 1 at every frequency, then the closed-loop system is stable for all structured perturbations $\Delta \in \mathbf{X}$. Conversely, if $\mu(RML^{-1})$ is greater than or equal to 1 at some frequency, then there exists a structured perturbation $\Delta \in \mathbf{X}$ that results in closed-loop instability. Note that a destabilizing Δ can be expressed as $L\theta R^{-1}$ for some constant θ.

5.3 Performance Implications

As noted in Section 4, the Small μ Theorem can also guarantee a pre-specified performance level by including a performance block in the BDBP problem. Furthermore, this performance level is guaranteed for all structured perturbations $\Delta \in \mathbf{X}$. These claims are made precise by a corollary to the Small μ Theorem that treats performance. Suppose that the plant uncertainties are given by

$$\Delta_r = \mathrm{diag}(\Delta_1, \Delta_2, \ldots, \Delta_m) \in \mathbf{X}_r$$

with corresponding weighting matrices L_r, R_r, interconnection matrix M_r, and 2n-tuple $K_r = (m_1, \ldots, m_n, k_1, \ldots, k_n)$. Suppose that a performance specification is given as

$$\bar{\sigma}(R_p M_p'(\Delta_r) L_p^{-1}) < 1 \qquad \forall\ \omega\ \forall\ \Delta_r \in \mathbf{X}_r \tag{5.9}$$

Here M_p is a $k_p \times k_p$ performance matrix which we desire to be small (as weighted by R_p and L_p). Examples include $M_p = (I+G'K)^{-1}$, $M_p = G'K(I + G'K)^{-1}$, etc., as discussed in Section 4. Note that M_p depends on the perturbation Δ_r, indicating that this performance should be met for all uncertainties.

Let M_{pr} and M_{rp} denote the transfer function matrices between performance outputs and perturbation outputs and between perturbation inputs and performance inputs, respectively. In terms of these matrices, it can be shown that

$M_p(\Delta_r) = M_p + M_{pr}\Delta_r(I + M_r\Delta_r)^{-1} M_{rp}$ where M_p is the performance matrix in the absence of uncertainties. Define

$$M_T = \begin{bmatrix} M_p & M_{pr} \\ M_{rp} & M_r \end{bmatrix}$$

$$K_T = (1, m_1, \ldots, m_n, k_p, k_1, \ldots, k_n)$$

$$L_T = \mathrm{diag}(L_p, L_r)$$

$$R_T = \mathrm{diag}(R_p, R_r)$$

$$\mathbf{X}_T = \mathbf{X}(K_T)$$

We have noted the dependence on K here to avoid confusion. Of course, the nominal interconnection operator M is assumed to be stable.

For these definitions, the following relationship exists between performance, stability robustness, and the SSV.

Theorem 5-2: (Robust Performance)

$M_p'(\Delta_r)$ stable and $\bar{\sigma}(R_p M_p'(\Delta_r) L_p^{-1}) < 1 \ \forall \omega$ and $\forall \Delta_r \in X_r$ iff $\mu(R_T M_T L_T^{-1}) < 1 \quad \forall \omega$

Proof: It follows from the Small μ Theorem that $\mu(R_T M_T L_T^{-1}) < 1 \quad \forall \omega$

iff $I + R_T M_T L_T^{-1} \theta > 0, \ \forall \text{Res} \geq 0, \quad \forall \theta$ such that $L_T^{-1} \theta R_T \in \mathcal{X}$

iff $\forall \text{Res} \geq 0, \ |I + R_r M_r L_r^{-1} \theta_r| > 0, \quad \forall \theta$ such that $L_r^{-1} \theta_r R_r \in \mathcal{X}_r$

and $|I + R_p M_p'(L_r^{-1} \theta_r R_r) L_p^{-1} \theta_p| > 0, \quad \forall \theta_p$ such that $L_p^{-1} \theta_p R_p \in \mathcal{X}_p$

iff $\forall \text{Res} \geq 0, \ |I + M_r \Delta_r| > 0$ and $|I + R_p M_p'(\Delta_r) L_p^{-1} \theta_p| > 0 \quad \forall \Delta_r \in \mathcal{X}_r, \forall \theta_p$ such that $L_p^{-1} \theta_p R_p \in \mathcal{X}_p$

iff $M_p'(\Delta_r)$ stable and $\bar{\sigma}(R_p M_p'(\Delta_r) L_p^{-1}) < 1 \ \forall \omega$ and $\forall \Delta_r \in X_r$. □

We note that this theorem extends the Small μ Theorem's robust stability results to a composite, simultaneous result on robust stability and performance. Thus, given an uncertain plant model with structured perturbations and a performance specification, we have a necessary and sufficient condition in terms of μ for satisfaction of the performance spec in the face of the uncertainty. If the condition $\mu < 1$ is met, then the desired performance is achieved for all perturbed plants. If $\mu \geq 1$, then there exists a structured perturbation which causes the performance spec to be violated. The robust performance condition may be thought of as arising from an equivalent "fictitious uncertainty," although this interpretation is not necessary.

5.4 Hybrid Compensators

Consider a multi-rate hybrid compensator for a continuous plant with uncertainty. The compensator is represented via a sector as discussed in Section 3.4. The resulting system can be rearranged as usual to obtain the representation of Figure 4.1 where

$$\Delta = \text{diag}(\Delta_1, \Delta_2)$$

$$\Delta_1 = \text{diag}(\lambda_{11} I, \lambda_{12} I, \cdots, \lambda_{1n_1}, \Delta_{11}, \Delta_{12}, \Delta_{1n_2})$$

$$\Delta_2 = \text{diag}(\Delta_{21}, \Delta_{22}, \cdots, \Delta_{2n_3})$$

The Δ_1 contains all the perturbations to the continuous plant as discussed previously in this section. The Δ_2 comes from the sector model of the multi-rate hybrid compensator.

Consider the set

$$D = \{\text{diag}(D_1, D_2, \cdots D_{n1}, d_1 I, d_2 I, \cdots d_{n_2} I, Q_1, Q_2, \cdots, Q_{n_3})\}$$

where the D_i's and d_j's are as in (5.6) and the Q_i's are as in Section 3.4. A sufficient condition for stability is that

$$\sup_{\omega} \inf_{D \in \mathcal{D}} \bar{\sigma}(DMD^{-1}) < 1 \qquad (5.10)$$

This stability test combines the analysis methods of Sections 3 and 4 to provide a reliable stability test for uncertain plants with hybrid controllers. Note that the $\{Q_i\}$ are required to be periodic with the same period as the associated hybrid compensator. This constraint has not yet been exploited in a systematic way. A somewhat ad hoc approach would be to ignore the periodicity and compute D as in Section 4. The periodicity of the Q's could then be imposed separately.

Note that the condition in equation (5.10) is sufficient for stability but not necessary. The use of the D scaling will reduce the conservativeness of the stability test but not eliminate it entirely.

A performance result analogous that in Section 5.3 can be easily obtained for the use of discrete-time controllers. As in Section 5.3 as "fictitious uncertainty" block can be introduced to reflect the performance specification. This performance block must be treated as a time-varying

perturbation and a sufficient performance test such as (5.10) must use a periodic Q for the performance block. For single-rate problems, the controller's period can be used. It is not clear how to best extend the performance analysis to multi-rate problems.

6.0 INTEGRATED FLIGHT AND PROPULSION CONTROL EXAMPLE

6.1 Problem Formulation

The use of SSV analysis is illustrated by an application of integrated control to an advanced fighter aircraft powered by a mixed flow, low bypass ratio, turbo-fan engine. Increased thrust-minus-drag is possible in large portions of the flight envelope by varying the engine control schedules and limits. In the subsonic regime, increased thrust is available during flight operation with low inlet distortion. For example, in straight and level flight with low sideslip, increased thrust could be obtained by shifting the fan operating line upward by increasing the engine pressure ratio (EPR) (refer to Figure 6.1). Since thrust is proportional to EPR (at constant airflow), this "uptrim" of EPR accomplishes the desired increase in thrust.

Thrust increases are in the range of 3-4% at a 0.9 Mach/30,000 feet operating point. Unfortunately, moving the operating point decreases the fan stall margin. Hence, operation in an uptrimmed state requires better regulation about the operating point to prevent inadvertent compressor stalls. Hence the control objective is tighter regulation of certain engine variables. Conventionally controlled engines like the F-100 operate with 16-18% margins as shown in Table 6.1. Here "margin" is defined as

$$\frac{P_{stall} - P_{nominal}}{p_{nominal}}.$$

One of the large contributors to the margin requirements is maneuver transients. Existing engine controls treat maneuvers ($\Delta\alpha$, $\Delta\beta$) as unknown disturbances when, in fact, this information is available from the airframe/inlet for improved control. The allocation to maneuvers can be significantly reduced by integrated control of the flight and propulsion systems.

It is desired that margin regulation be accomplished without adversely affecting thrust, i.e., both margin and thrust are to be regulated. For this example, performance goals are specified in terms of the output sensitivity function. Loop properties for performance are specified at the outputs because the example is focusing on output regulation. The performance goals were defined as a low frequency gain of about 40 about 1 rad/sec bandwidth, and high frequency sensitization of no more than three. This means disturbances will be reduced by a factor of about 25 at low frequencies. The performance goals are described by a performance block of size

$$3\,\frac{1s+0.01}{1s+1.2}$$

The next section describes the airframe, inlet and engine model used to design the "uptrim regulator".

6.2 Design Models

An integrated model was developed by interconnecting linearized airframe, inlet, and engine models. The techniques used are similar to those used in the Flight and Propulsion Control Coupling studies [20].

Airframe - A linearized longitudinal and lateral-directional model was obtained for the augmented airframe at a 0.9 Mach 30,000 foot trim condition. An eight-state model describes linear perturbations:

State =
- Filter state
- Pitch state
- Pitch angle
- Angle of Attack
- Velocity
- Altitude
- Angle of side-slip
- Side-slip rate

The input is

$$u = \begin{matrix} \text{Pitch stick} \\ \text{Rudder pedal} \\ \text{Net thrust (from engine)} \end{matrix}$$

In addition to the usual airframe measurement of load factor and pitch rate, linearized expression were included to relate changes in total temperature and Mach number with velocity and altitude changes.

$$\Delta T_T = f_1(\Delta v, \Delta h)$$
$$\Delta M = f_2(\Delta v, \Delta h)$$

These quantities will be inputs to the inlet model below.

Inlet Model - The inputs to the inlet model are:

α - Angle-of-attack
β - Angle-of-Sideslip
M - Mach number
T_T - Total temperature
ω_E - Engine airflow

The outputs are:

P_{T2} - Total pressure at engine face
K_{a2} - Distortion index

The inlet used in this example has fixed geometry and was designed for subsonic operation. One actuator positions a ramp (ρ_A) based on an angle-of-attack schedule. The model has the following functional form which was linearized.

$$\rho_A = f(\alpha, V, h, \rho_A)$$
$$P_{T2} = f(M, T_T, \rho_A, \alpha, \beta, \omega_E)$$
$$K_{a2} = f(\rho_A, M, T_T, \alpha, \beta, \omega_E)$$

Engine Model - The engine model is a version of the F-100, a low bypass ratio, turbo-fan engine. The linear model was supplied by Pratt & Whitney and was developed for 0.9 mach, 30,000 feet, max power operating point. The engine model is in a state-space format

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

where the state, control and outputs are defined in Table 4.2.

Of particular interest for this design is the relation of fan stall margin to engine inputs. The fan stall margin SMAF is defined as

$$SMAF = 1 - P_{RFAN}/SPR_{FAN}$$

where SPR_{FAN} is the fan surge pressure ratio. The fan surge pressure ratio is determined from compressor maps as a function of corrected air flow and distortion indes. Thus,

$$SPR_{FAN} \triangleq f(W_c, K_{a2}).$$

Standard linearization techniques are used to derive the SMAF response.

Model Uncertainty - The integrated model is linear, time invariant, and finite dimensional, and thus can be represented by transfer function matrices with rational elements. Model uncertainty is modeled as sector-bounded, linear, time-invariant operators.

For this example, four uncertainties are added to the nominal loop as shown in Figure 6-2. Three are associated with the uncertain plant, and one is for the digital compensator. Δ_A represents actuator uncertainty, Δ_T is the uncertainty in deriving thrust and Δ_M the uncertainty in

deriving fan stall margin. Δ_D is the cone radius for the digital compensator. The Δ_i are specified by $L_i = I$ and

$$R_A(j\omega) = \frac{s+100}{1000}$$

$$R_M(j\omega) = \frac{s+10}{20}$$

$$R_T(j\omega) = \frac{s+1}{100}$$

$$R_D(j\omega) = 0.05\sqrt{\frac{2}{3}}\,s$$

and their frequency characteristics are plotted in Figure 6-3.

The performance goals are given in terms of the return difference matrix at the outputs satisfying the following inequality:

$$\underline{\sigma}\,[I+G(j\omega)K(j\omega)] \geq \frac{3|j\omega+0.01|}{|j\omega+1.2|} \qquad \forall\omega$$

Equivalently,

$$\bar{\sigma}\,[(I+G(j\omega)K(j\omega))] \leq \frac{|j\omega+1.2|}{3|j\omega+0.01|} \qquad \forall\omega$$

or

$$\bar{\sigma}\,[(I+G(j\omega)K(j\omega))^{-1}\,\frac{3|j\omega+0.01|}{|j\omega+1.2|} \leq 1 \quad \forall\omega$$

This constraint on output sensitivity is equivalent to stability robustness for an uncertainty Δ_p from row 4 of Table 4.1 where

$$\bar{\sigma}\,(W_p(j\omega)) \leq \frac{3|j\omega+0.02|}{|j\omega+1.2|}$$

The "fictitious uncertainty" Δ_p is modeled as an element of $(0,L_p,R_p)$ with

$$R_p(j\omega) = \frac{3|j\omega+0.01|}{|j\omega+1.2|}, \quad L_p(j\omega) = 1$$

The performance specification is included with the three component uncertainties and handled with the structured singular value analysis in the next section.

6.3 Design Application

The design for the uptrim regulator is intended to demonstrate the systematic design methodology summarized below. The design of this multivariable compensator uses advanced LQG techniques for shaping loop transfer matrices in the frequency-domain [14]. These designs are presented as illustrative examples and are not intended as final, flight-quality control laws.

The problem formulation emphasized regulation of fan margin during maneuvers. This may result in undesirable responses in other variables such as thrust. As noted in the modeling section, the F-100 engine has available multiple inputs for regulating engine variables. For the bare engine design two outputs, fan margin and thrust, were identified as two important variables to be controlled. Two appropriate engine inputs, nozzle area and augmentor fuel flow, were selected to accomplish the regulation. Other combinations of inputs and/or outputs could be selected. Preliminary designs using the six inputs to the engine model indicated that nozzle area and augmentor fuel flow were effective in regulating the selected variables. In general, one would use the inlet contols and engine controls to hold margins during maneuvers. For our design example, however, the inlet has minimal effect subsonically so our example will use just engine controls.

Variables not regulated would have to be checked to see they don't exceed normal operating limits, for example, temperature constraints. This evaluation was not part of our example design. Nevertheless, this two-input, two-output regulator design provides a realistic exercise of our LQG-based methodology and SSV analysis techniques.

The design methodology is summarized in the following steps:

Step 1 - Design a KBF such that the loop transfer function meets performance and stability robustness requirements.

Step 2 - Design a sequence of LQR's with the scalar parameter q allowed to take on consecutively larger values.

Step 3 - Select an element of the sequence of transfer functions which adequately approximates the desired transfer function over the frequencies of interest.

The system description is a state space model consisting of A, B, and C matrices as developed in the modeling section. For the KBF design we follow the advanced loop shaping procedure and augment the plant with the desired loop shapes. This yields

$$\tilde{A} = \begin{bmatrix} A & 0 & 0 \\ 0 & -.01 & 0 \\ 0 & 0 & -.01 \end{bmatrix} \qquad \tilde{B} = \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix}$$

$$\tilde{C} = \begin{bmatrix} & 1 & 0 \\ C & & \\ & 0 & 1 \end{bmatrix} \qquad \tilde{D} = \begin{bmatrix} 0 \\ - - - \\ 1 \end{bmatrix}$$

Letting N = I, and solving for the KBF gains results in the loop properties $C(sI-A)^{-1}KF$ plotted in Figure 6-4. This is nothing more than the desired response and completes Step 1.

To achieve loop recovery, a sequence of LQR's will now be designed. The state weighting matrix is

$$q^2 C^T C \qquad q^2 \to \infty$$

and the control weighting is fixed at R=I. Several steps of the sequence for q = 0.003, 0.01, 0.03, and 0.1 were used to achieve asymptotic recovery. The final iteration is judged to be satisfactory for this design (refer to Figure 6-5).

The corresponding LQR gains are combined with the previous KBF to form the overall compensator. Use of the asymptotic procedure results in fast modes in the filter which may be undesirable from an implementation viewpoint. They are easily eliminated by standard residualization techniques. In this case, there were two modes at 310 and 1190 rad/s which are clearly outside out frequencies of interest. Elimination of these two modes results in a reduced order compensator whose loop properties are indistinguishable from the full state design shown in Figure 6-5.

6.4 Analysis of the Design

The singular values of the return difference matrix and the closed-loop response matrix for the design with the reduced-order compensator are shown in Figure 6-6. These plots are indistinguishable from plots where the loop is broken at the input. At low frequencies (below the 1 rad/sec crossover), the return difference matrix is dominated by the loop transfer matrix (Figure 6-5). At high frequencies (above the 1 rad/sec. crossover), the closed-loop response matrix is dominated by the return difference matrix. Both of these plots demonstrate a well-behaved crossover. The example design provides disturbance rejection by a factor of at least 60 at frequencies less than 0.01 rad/sec.

A discrete-time version of this compensator was implemented with a sample time of 0.1 sec. A third order Butterworth prefilter cutting off at 10 rad/sec was included.

The structured singular value (SSV) discussed in Section 4 is plotted in Figure 6-8. Its value is less than unity indicating the design simultaneously satisfies the robustness and performance requirements previously presented.

A time-domain simulation complements the frequency domain syntnesis used to this point. Time histories were computed for a pitch stick command. Plots are made with and without the "uptrim regulator". Variables plotted are:

- Angle-of-attack
- Velocity
- Altitude
- Net thrust
- Fan stall margin
- Engine controls (nozzle area, augmentor fuel flow)
- Engine states (N1, N2, burner metal temperature)

In both cases the airframe's response is nearly identical. With the regulator, thrust variations are smaller and hence the velocity and altitude profiles vary slightly. The peak excursion in margin is reduced about a factor of 2 over the unregulated case. Figure 6-9 shows the modulation of AJ and WFAB used to accomplish regulation. Figure 6-10 shows the response of the three engine states. The simulation shown was for a 1/4g incremental "pull-up maneuver". If the results are scaled to a "5g" pull-up the perturbation of the variables are shown in Table 6.3. Note that less than a 10% modulation of nozzle area and augmentor fuel flow is used to regulate margin.

7. CONCLUSIONS

This paper has addressed two primary topics:

(i) the use of conic sectors to represent hybrid compensators and

(ii) the use of the structured singular value to analyze linear feedback systems.

Starting from a result of Thompson [1], we showed how general (even unstable) hybrid compensators with arbitary, multiple sample rates could be reliably approximated by linear, time-invariant operators. The approximation error is properly accounted for in a conic sector. The structured singular valve provides a nonconservative measure of performance in the face of structured uncertainty. The Small μ Theorem gives a necessary and sufficient condition in terms of μ for stability of a linear system with multiple, simultaneous, norm-bounded perturbations of arbitrary, fixed structure. The Robust Performance Theorem provides a similar condition for the satisfaction of performance specifications in the presence of structured perturbations.

The use of SSV analysis with hybrid compensators was illustrated through the design example of an integrated airframe/inlet/engine control mode. The control objective of regulating thrust and fan stall margin during maneuvers was expressed as magnitude constraints on the output sensitivity function in the frequency domain. Model uncertainty was assumed at the actuators, in the estimate of thrust and stall margin, and in the representation of the hybrid compensator. An LQG-based procedure was used to achieve loop properties which satisfied performance and robustness properties one-at-a-time. The SSV was used to assess simultaneously the effects of uncertainty on the performance and robustness of the system.

Structured singular value analysis is a powerful technique for quantifying the impact of uncertainty occurring throughout a feedback system on overall closed-loop performance. At the present time, synthesis techniques for designing directly in terms of the SSV are lacking. A number of researchers are addressing this topic and progress is expected.

References

[1] P.M. Thompson, "Conic Sector Analysis of Hybrid Control Systems," PhD Dissertation, MIT, August 1982.

[2] I.M. Horowitz, Syntnesis of Feedback Systems, New York: Academic, 1963.

[3] Special Issue on Linear Multi-Variable Control Systems. IEEE Trans. Automatic Control, February 1981.

[4] J.C. Doyle, J.E. Wall and G. Stein, "Performance and Robustness Analysis for Structured Uncertainty" IEEE Conference on Decision and Control, December, 1982.

[5] J.C. Doyle, "Analysis of Feedback Systems with Structured Uncertainties, "Proc. IEE, November 1982.

[6] C.A. Desoer and M. Vidyasagar, Feedback Systems: Input-Output Properties, Academic Press 1975.

[7] B.A. Francis and G. Zames, "Design of H∞ - Optimal Multivariable Feedback Systems," IEEE Conference on Decision and Control, December 1983.

[8] J.C. Doyle, "Synthesis of Robust Controllers and Filters with Structural Plan Uncertainty," IEEE Conference on Decision and Control, December 1983.

[9] G. Zames, "On the input-output stability of time-varying nonlinear feedback systems - Part I, "IEEE Trans. Automatic Control, Volume AC-11, no. 2. pp. 228-238, April 1966.

[10] G. Zames, "On the input-output stability of time-varying nonlinear feedback systems - Part II." IEEE Trans. Automatic control, Volume AC-11, no. 3, pp. 465-476, July 1966.

[11] M.G. Safonov, Stability and Robustness of Multivariable Systems, Cambridge, MA, MIT Press, 1980.

[12] Gene F. Franklin and J. David Powell, Digital Control of Dynamic Systems, Addison-Wesley Publishing Co., Reading, Massachusetts, 1980.

[13] J.T. Tou, Digital and Sampled-Data Control Systems, McGraw-Hill, New York, 1959.

[14] J.C. Doyle and G. Stein, "Multivariable Feedback Design: Concepts for a classical/modern Synthesis, IEEE Trans. on Automatic Control Vol. AC-26, No. 1, February, 1981.

[15] N.A. Lehtomaki, "Practical Robustness Measures in Multivariable Control System Analysis," Ph.D. Dissertation, MIT, May 1981.

[16] J.C. Doyle and MG. Safonov, "Convexity of the Block Diagonal Scaling Problem," Honeywell Internal Memo, August 1982.

[17] E.E. Osborne, "On pre-conditioning of matrices," I. Assoc. Comput., Mach. 7, 338-345, 1960.

[18] N.A. Lehtomaki, et. al., "Robustness Tests Utilizing the Structure of Modeling Error," Proc. 1981 CDC, San Diego, CA, December 1981.

[19] J.S. Freudenberg, D.P. Looze, and J.B. Cruz, "Robustness Analysis Using Singular Value Sensitivities," Int. J. Control, Vol 35, No. 1, pp. 95-116, 1982.

[20] R.L. Heimbold, C.J. Yi, and R.J. Miller, "Flight/Propulsion Control Coupling (FPCC) and Dynamic Interaction Investigation. Phases I, II, and III." U.S. Air Force, The Flight Dynamics Laboratory, AFFDL-TR-153.

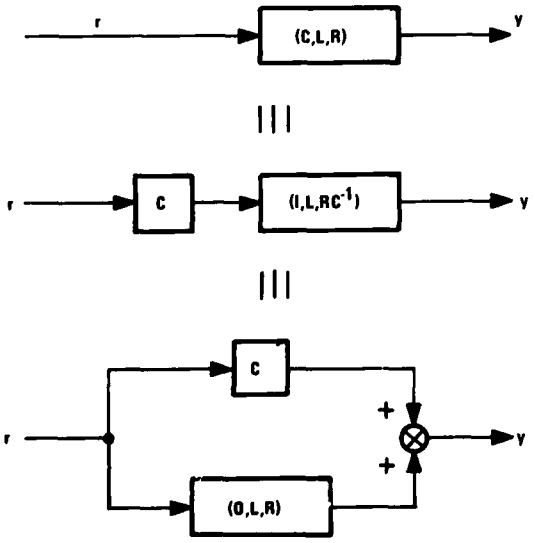


Figure 2-1. Three Equivalent Representations of a Sector

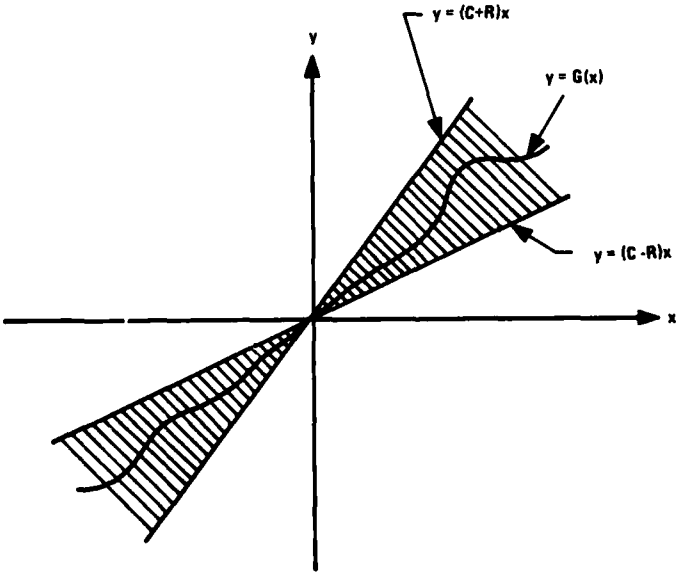


Figure 2-2. Illustration of a Nonlinearity Contained in a Conic Sector.

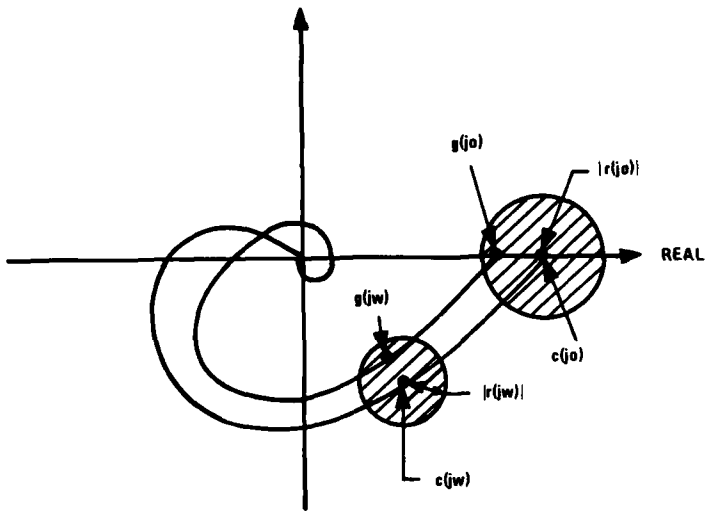


Figure 2-3. Nyquist Plane Interpretation of of a Conic Sector Condition

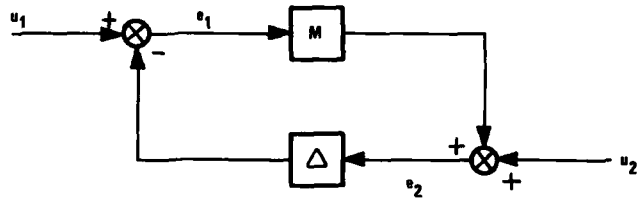


Figure 2-4. Feedback Configuration for the Small Gain Theorem.

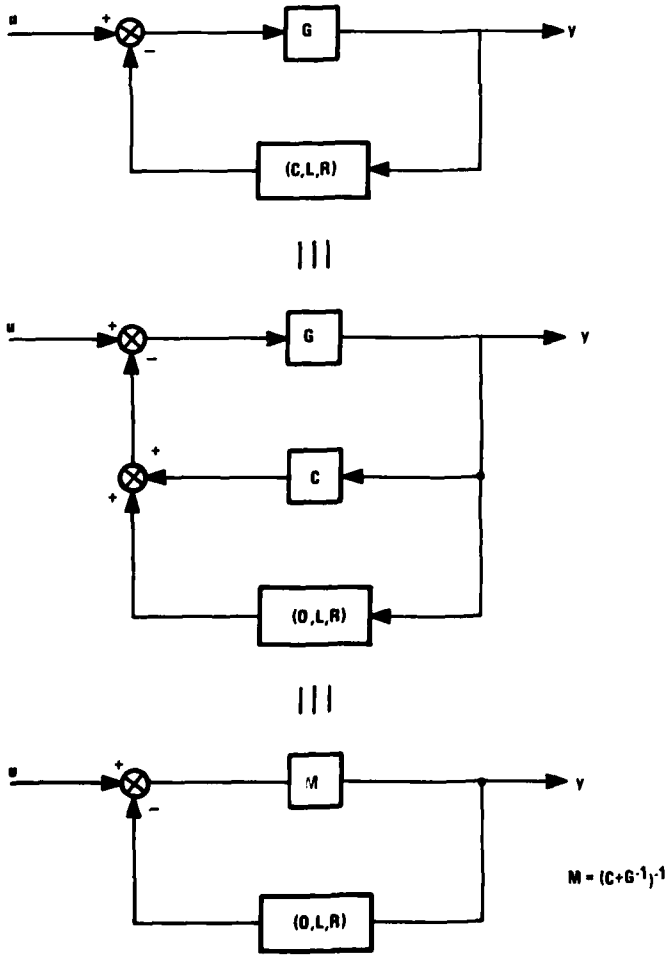


Figure 2-5. Three Equivalent Representations of the Feedback Configuration Involving Operator G and Sector (C, L, R).

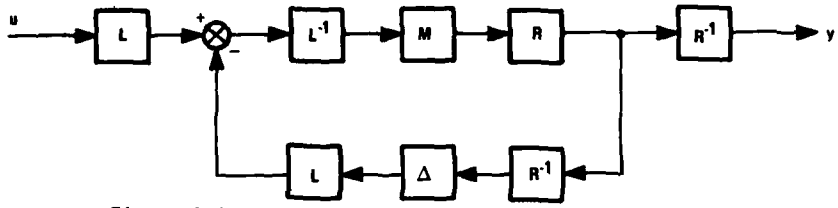


Figure 2-6. The Operators L and R With Their Inverses Do Not Alter the Stability of the Closed Loop System.

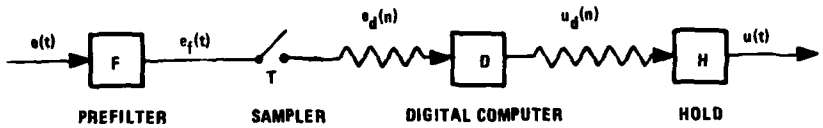


Figure 3-1. The Hybrid Compensator.

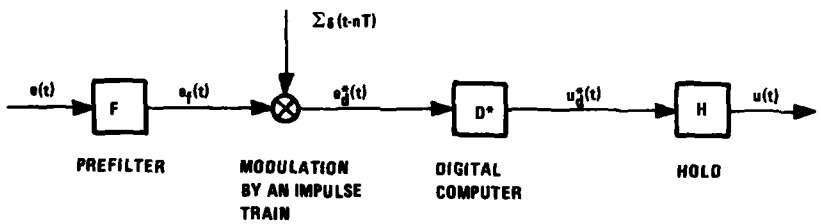


Figure 3-2. An Analog Representation of the Hybrid Compensator.

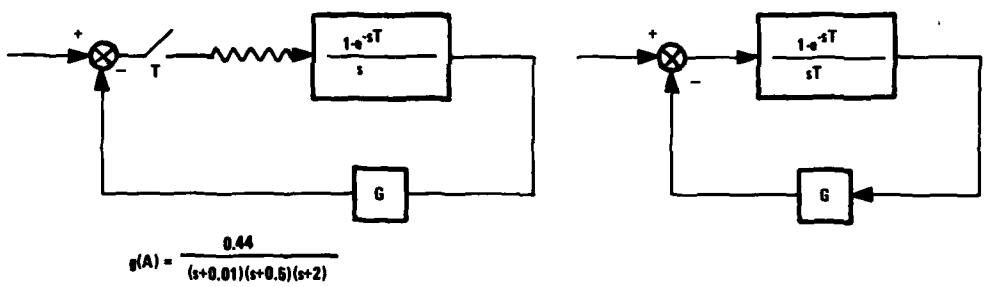


Figure 3-3. Example of a Stable Discrete Time System (a) for Which the Corresponding Continuous System is Unstable (b).

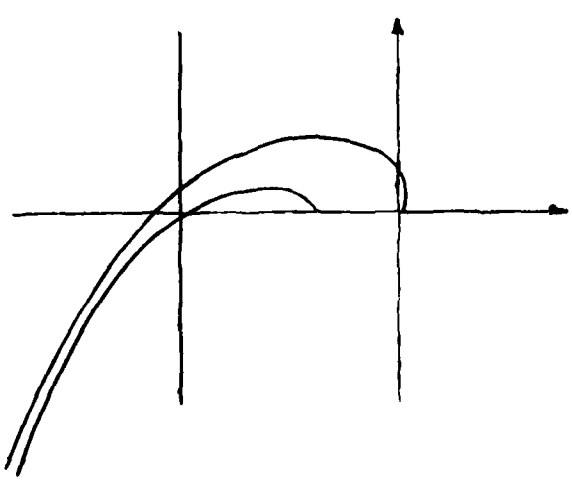


Figure 3-4. Nyquist Plots for the Systems of Figure 3-3.

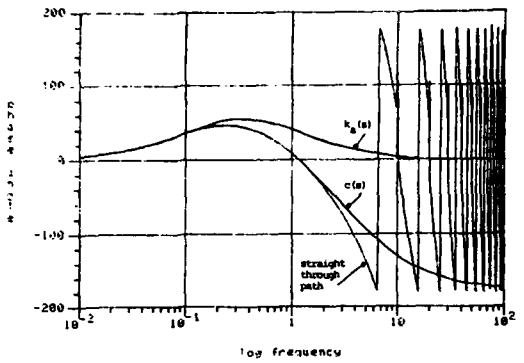


Figure 3-5a. Lead Compensator.

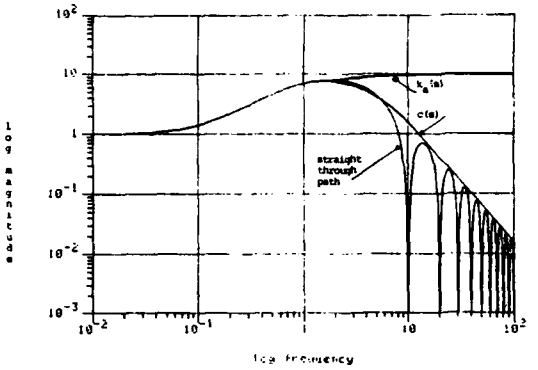


Figure 3-5b. Lead Compensator

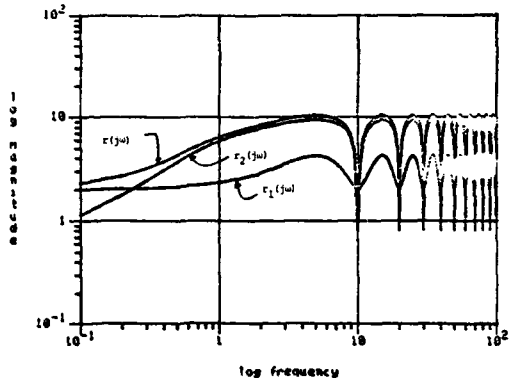


Figure 3-6. Radius

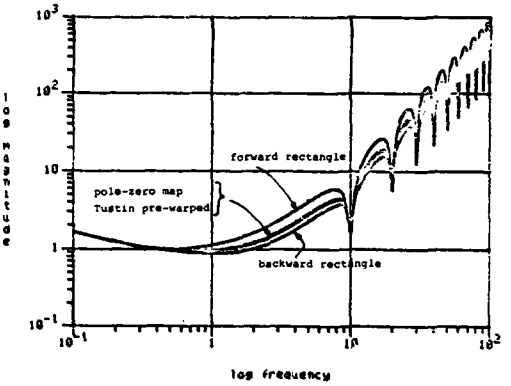


Figure 3-7. Multiplicative Radii, Comparison of Different Discretization Techniques.

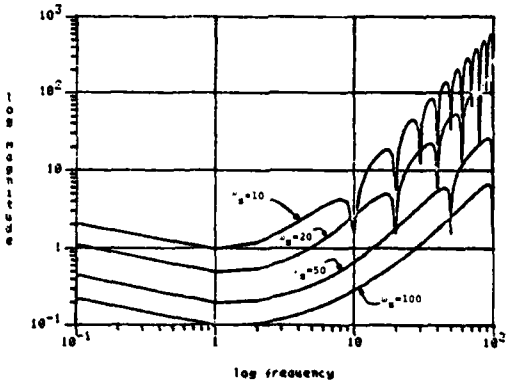


Figure 3-8. Multiplicative Radii, Comparison of Different Sample Intervals.

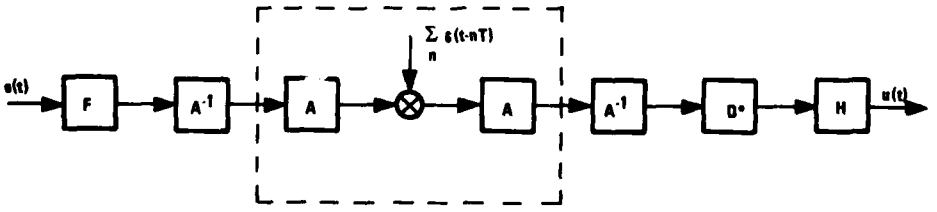


Figure 3-9. An Equivalent Analog Representation of the Hybrid Compensator.

Figure 3-10. Normalized Error for the Conic Sector Representation of a Sampler With Three Candidate Choices of the Operator A.

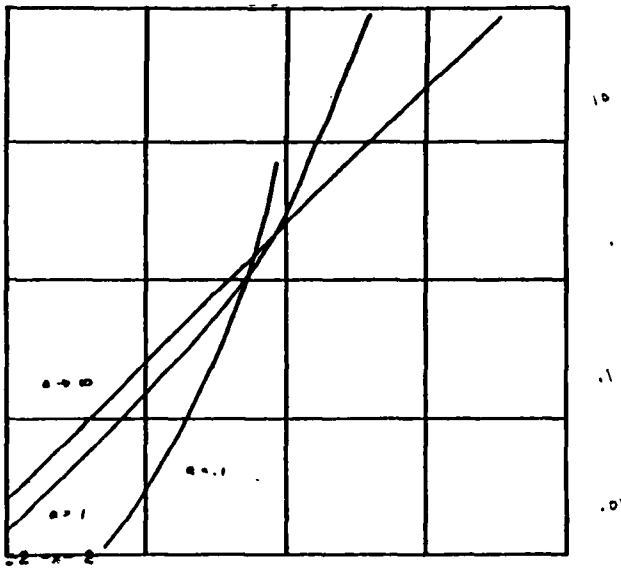
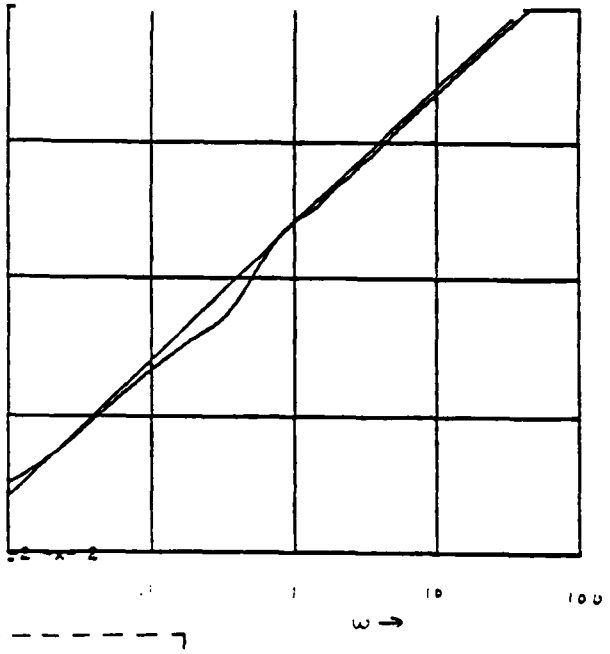


Figure 3-11. Graphical Version of (3.20).



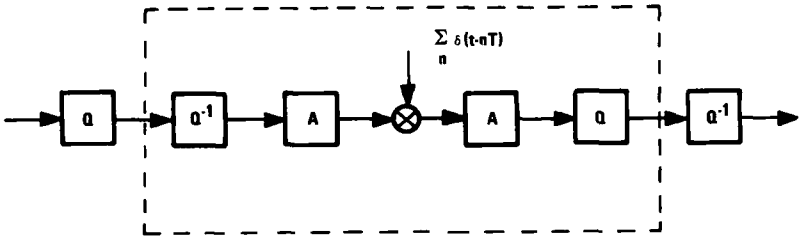


Figure 3-12. Equivalent Representation of the Sampling Operation.

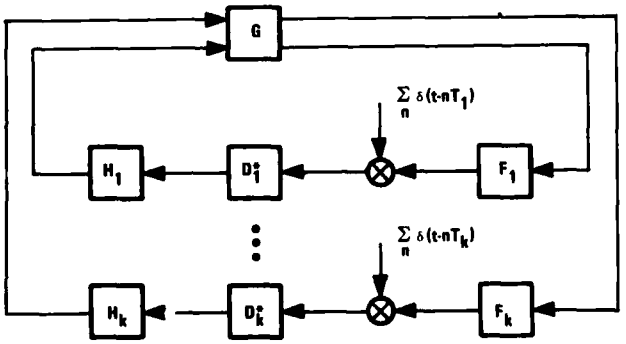


Figure 3-13. The Multi-Rate Hybrid Compensator.

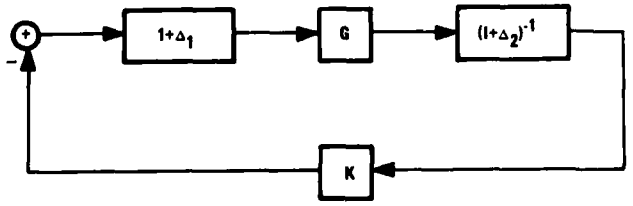
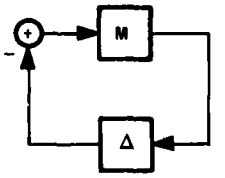


Figure 4-2. Feedback Loop With Two Uncertainties.



$\Delta = \text{diag}(\Delta_1, \Delta_2, \ldots \Delta_m)$

Figure 4-1. Feedback Loop as a BDBP Problem.

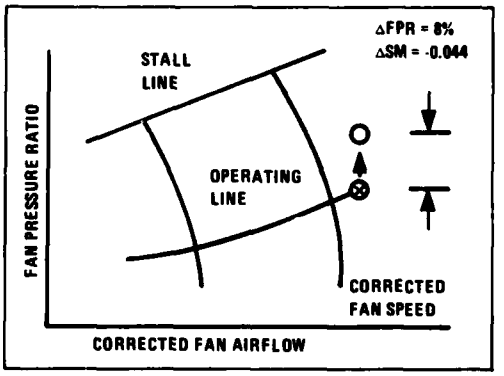


Figure 6-1. Subsonic Uptrim for Increased Thrust

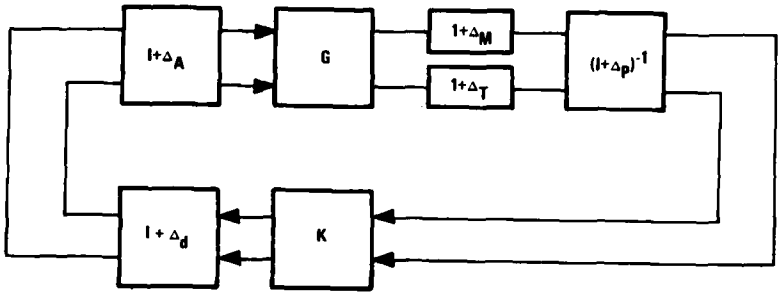


Figure 6-2. Structured Uncertainty.

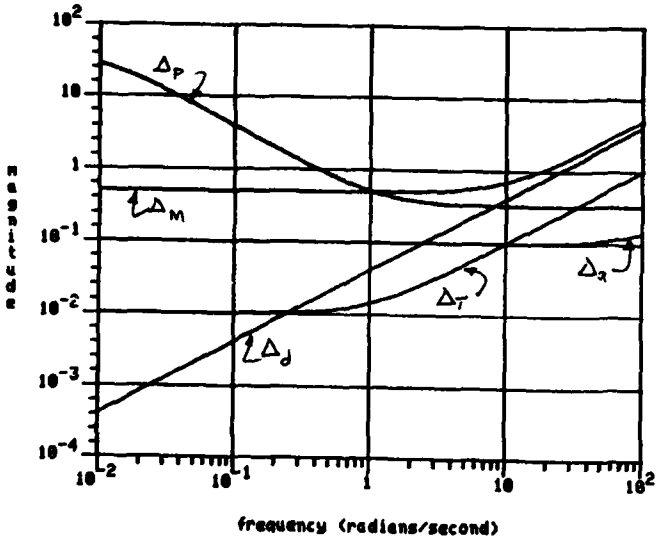


Figure 6-3. Frequency characteristics of the assumed uncertainty.

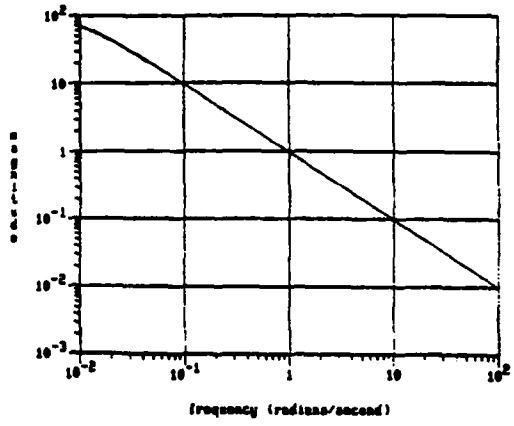


Figure # 6-4
Singular Values Of Loop Transfer $c(sI-A)^{-1}KF$

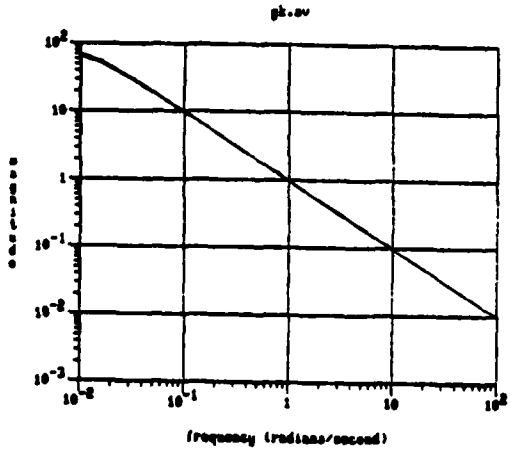


Figure # 6-5
Singular Values of Loop Transfer operator

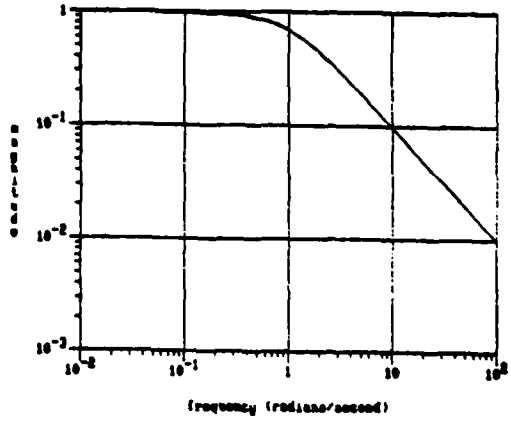


Figure # 6-6
Singular Values of Closed Loop Response

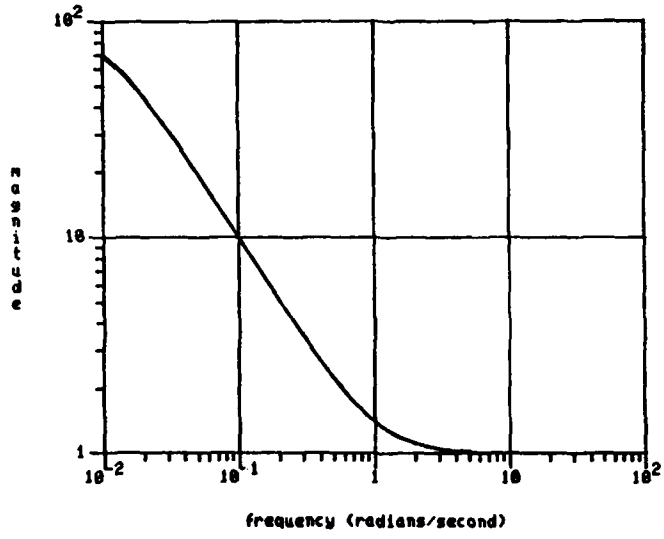


Figure 6.7 Singular values of the return difference operator

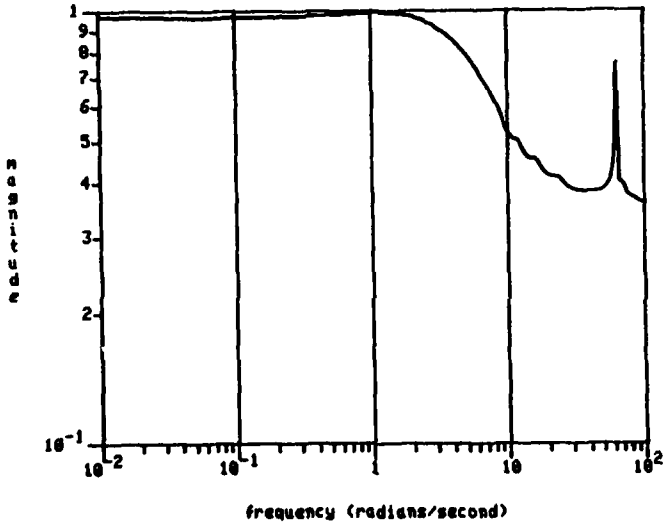


Figure 6.8 Structured Singular Value for the Design Example.

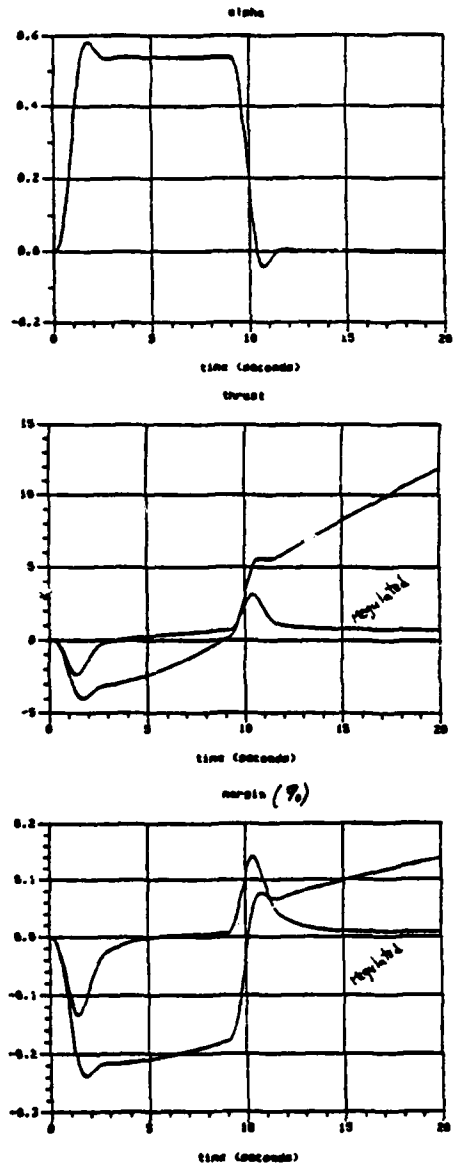


Figure 6-9. Response of Thrust and Margin With and Without Regulation

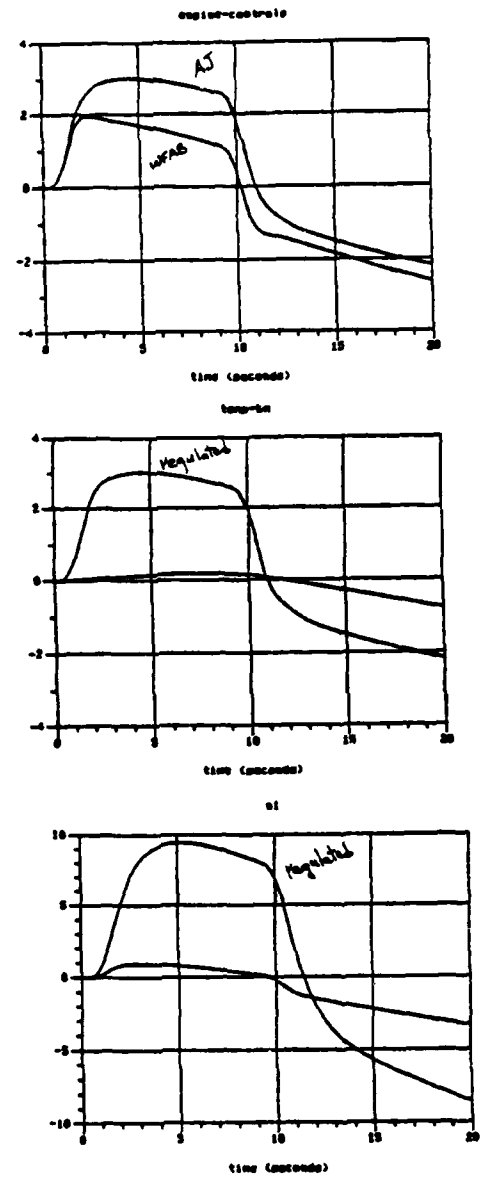
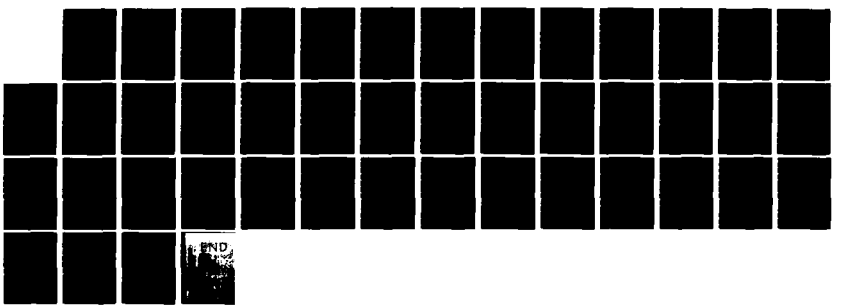
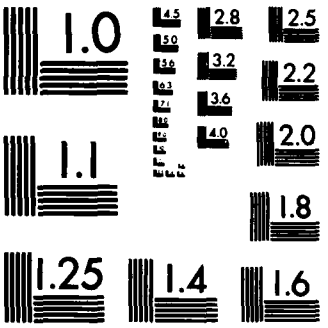


Figure 6-10. Response of Engine States With and Without Regulation.

AD-A133 798 COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS(U) ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE) 2/2

UNCLASSIFIED JUL 83 AGARD-LS-128 F/G 17/7 NL





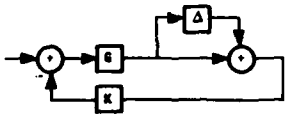
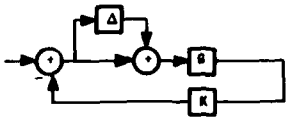
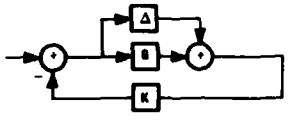
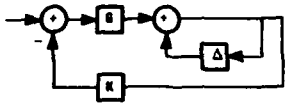
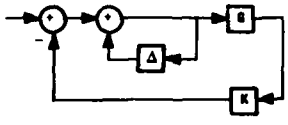
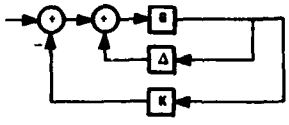
LOCATION OF Δ_I	CONDITIONS IMPOSED ON NOMINAL FEEDBACK LOOP SHAPES	REPRESENTATIVE TYPES OF UNCERTAINTY CHARACTERIZED	REPRESENTATIVE TYPES OF PERFORMANCE SPECS
	$\bar{\sigma}[R\ GK(I+GK)^{-1}L^{-1}] < 1$	– OUTPUT (SENSOR) ERROR – NEGLECTED HF DYNAMICS – CHANGING NUMBERS OF rhp ZERO $G' = (I+\Delta)G$	– SENSOR NOISE ATTENUATION – OUTPUT RESPONSE TO OUTPUT COMMANDS
	$\bar{\sigma}[R\ KG(I+KG)^{-1}L^{-1}] < 1$	– INPUT (ACTUATOR) ERRORS – NEGLECTED HF DYNAMICS – CHANGING NUMBERS OF rhp ZEROS $G' = G(I+\Delta)$	– INPUT RESPONSE TO INPUT COMMANDS
	$\bar{\sigma}[R\ K\ (I+GK)^{-1}L^{-1}] < 1$	– ADDITIVE PLANT ERRORS – UNCERTAIN rhp ZEROS $G' = G+\Delta$	– INPUT RESPONSE TO OUTPUT COMMANDS
	$\bar{\sigma}[R\ (I+GK)^{-1}L^{-1}] < 1$	– LF PLANT PARAMETER ERRORS – CHANGING NUMBERS OF rhp POLES $G' = (I+\Delta)^{-1}G$	– OUTPUT SENSITIVITY – OUTPUT ERRORS TO OUTPUT COMMANDS AND DISTURBANCES
	$\bar{\sigma}[R\ (I+KG)^{-1}L^{-1}] < 1$	– LF PLANT PARAMETER ERRORS – CHANGING NUMBERS OF rhp POLES $G' = G(I+\Delta)$	– INPUT SENSITIVITY – INPUT ERRORS TO INPUT COMMANDS AND DISTURBANCES
	$\bar{\sigma}[R\ (I+GK)^{-1}G\ L^{-1}] < 1$	– LF PLANT PARAMETER ERRORS – UNCERTAIN rhp POLES $G' = (G^{-1}+\Delta)^{-1}$	– OUTPUT ERRORS TO INPUT COMMANDS AND DISTURBANCES

Table 4-1. Representative Robustness/Performance Conditions.

DIRECT DIGITAL DESIGN VIA POLE PLACEMENT TECHNIQUES

G. F. Franklin
Department of Electrical Engineering
Stanford University
Stanford, California 94305 USA

7.1 SUMMARY

The design of the dynamics of a digital control for satisfactory transient response can be done in a number of ways. One of the more effective ways is to do the design so that the poles of the closed loop system are in desired or at least acceptable locations. Such design schemes are known as pole placement methods. In this section the method of pole placement will be described and formulas suitable for computer implementation will be given. Also, the method will be compared to both the transform methods described earlier and to the methods based on optimal control, including stochastic control and the Kalman filter. Several examples will be given to illustrate the methods.

7.2 SPECIFICATIONS IN TERMS OF POLE LOCATIONS

The first requirement of pole placement design is the formulation of the specification in terms of pole locations. Three approaches to this part of the problem will be described. These are the dominant second order, the higher order prototypes, and the symmetrical root locus.

(a) Dominant Second Order. The second order transient with complex roots at radius ω_n and damping ratio ζ is well documented and has a response sketched in Fig. 7.1. From this plot, characteristics such as overshoot, rise time, and settling time can be identified. For example, the overshoot is mainly dependent on the damping ratio and the relation is plotted in Fig. 7.2. Also, we can associate rise time with the

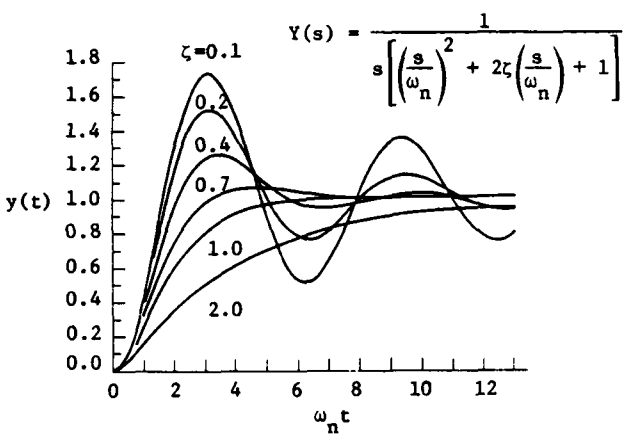


Figure 7.1: Response of second order system

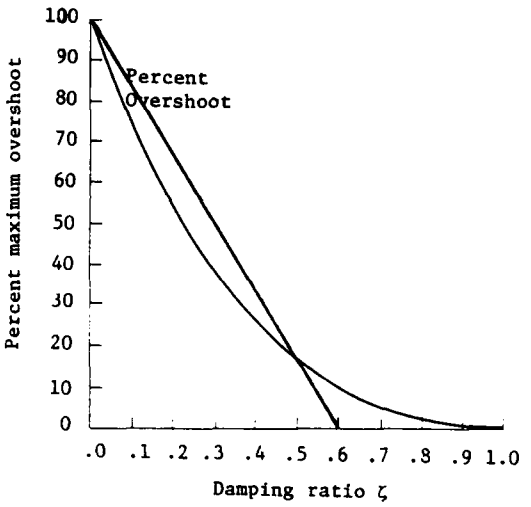


Figure 7.2: Relation between damping ratio and per cent overshoot

natural frequency n bandwidth and the settling time with $\zeta\omega_n$, the real part of the corresponding poles. Thus, for the second order system, we can express the relation between time domain specification and pole location as

$$M_p \cong 100\left(1 - \frac{\zeta}{.6}\right) \quad \text{overshoot}$$

$$t_r \cong 2.5/\omega_n \quad \text{rise time}$$

$$t_s \cong 4.6/\zeta\omega_n \quad \text{settling time}$$

Solving for the pole parameter, we find

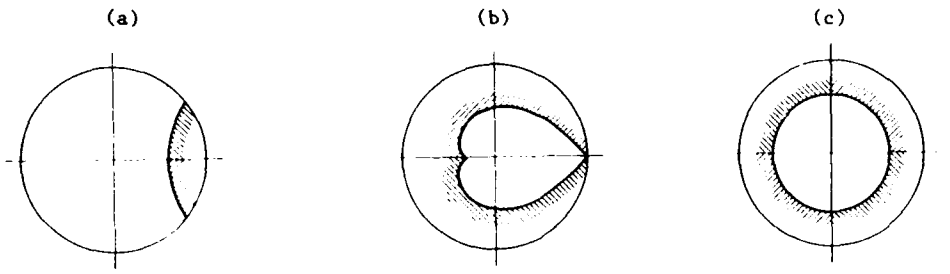
$$\zeta \geq 0.6\left(1 - \frac{M_p}{100}\right) \quad \text{damping ratio}$$

$$\omega_n \geq 2.5/t_r \quad \text{natural frequency (radius)} \qquad (7.1)$$

$$\zeta\omega_n \geq 4.6/t_s \quad \text{real part}$$

The relations 7.1 tell us how specifications on a second order response would be expressed in terms of s-plane pole locations. To relate these to digital design, we use the relation

$$z = e^{sT} \qquad (7.2)$$



k	r
5	0.4
10	0.63
15	0.75
20	0.80
50	0.91

Figure 3: Maps of pole locations to meet transient specifications
(a) Rise time; (b) Overshoot; (c) Settling time

With Eq. (7.2), the relations of Eq. (7.1) map into the z-plane as sketched in Fig. 7.3.

To apply the technique of dominate second order poles, it is necessary to be sure that the response is not significantly modified by other poles (or zeros) in the system. While bounds are hard to obtain, experiment reveals that if the dominate poles are in the right half of the z-plane, then an extra real pole or zero in the left half of the z-plane will have negligible effect and will have small effect so long as the added pole or zero has a smaller real part than the dominate poles do.

(b) Prototype Design. In some cases a higher order than second can be used effectively with all poles interacting in a specific way. Two interesting higher order systems are the Butterworth and the ITAE propotypes. Butterworth filters are designed to display a frequency response which is maximally flat. The ITAE prototypes were developed for servomechanisms so the step response would minimize the error integral

$$I = \int_0^\infty t|e|\,dt$$

which is the Integral of Time times Absolute Error. The transient response of these systems are shown in Fig. 7.4 and the normalized coefficients for the ITAE are given in Fig. 7.5 The poles of the Butterworth are equally spaced on the unit circle.

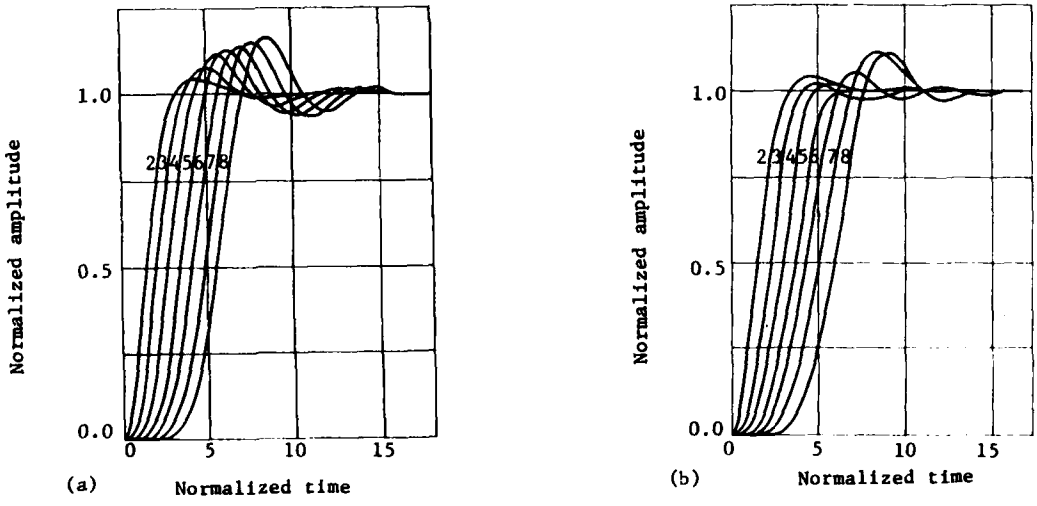


Figure 7.4: Transient responses of (a) Butterworth and (b) ITAE prototypes

k	
1	s + 1
2	s + 0.707 ± j0.707
3	(s + 0.7081)(s + 0.521 ± j1.068)
4	(s + 0.424 ± j1.263)(s + 0.626 ± j0.4141)
5	(s + 0.8955)(s + 0.376 ± j1.292)(s + 0.5758 ± j0.5139)
6	(s + 0.3099 ± j1.263)(s + 0.5805 ± j0.7828)(s + 0.7346 ± j0.2873)

Figure 7.5: Table of pole locations in the s-plane for the ITAE prototype

(c) The Symmetric Root Locus. One of the most effective and widely used methods of linear control system design is the linear equation, quadratic loss, Gaussian noise or LQG optimal control. The method separates into two problems: the linear quadratic regulator (no noise, complete state feedback) and the least squares estimator (Kalman filter) to produce estimates of the state. The LQR problem is to find the control so that the performance index

$$J = \sum_{0}^{\infty} \rho\, y_k^2 + u_k^2 \tag{7.3}$$

is minimized for the system

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_k \\ y_k &= H x_k \end{aligned} \tag{7.4}$$

We can show that the optimal control is given by the linear state feedback $u = -Kx$. If we define the open loop transfer function as

$$H_0(z) = H(zI-\Phi)^{-1}\Gamma \tag{7.5}$$

then we can show that (with some uninteresting exceptions) the poles of the optimal closed loop system are given by the stable roots of the symmetric root locus

$$\det\left(I + \rho\, H_0^T(z^{-1})\, H_0(z)\right) = 0 \tag{7.6}$$

In the case of single input systems, Eq. (7.6) is a scalar root locus problem with respect to the parameter ρ which weights the tracking error, y^2, versus the control effort, u^2 in the loss function of Eq. (7.3). The design may thus be accomplished by first selecting the matrix H, which defines the "tracking error," and then selecting the ρ which balances the importance of tracking error against control effort. As we will shortly see, once the poles have been chosen, in the single input case, the control law K can be computed by explicit formula. An example of the use of this method may be given by selecting the double integrator plant as follows. The discrete state equations for sampling period 1 second in terms of position x_1 and velocity x_2 are

$$\begin{aligned} x_1(k+1) &= x_1 + x_2(k) + \tfrac{1}{2} u(k) \\ x_2(k+1) &= x_2(k) + u(k) \end{aligned} \tag{7.7}$$

We can select the tracking error to be a linear combination of position and velocity error by choosing

$$y = x_1 + \alpha x_2 \tag{7.8}$$

From Eqs. (7.7) and (7.8) we compute the open loop transfer function to be

$$\begin{aligned} H_0(z) &= H(zI-\Phi)^{-1} \\ &= \left(\alpha + \frac{1}{2}\right) \frac{z - \dfrac{\alpha - 1/2}{\alpha + 1/2}}{(z-1)^2} \end{aligned} \tag{7.9}$$

From Eq. (7.9) we see that the open loop transfer function can be given a zero anywhere from $z = -1$ corresponding to $\alpha = 0$ to $z = +1$ corresponding to $\alpha = \infty$. The symmetric root locus of Eq. (7.6) for this case is

$$\begin{aligned} 1 + \rho\, H_0^T(z^{-1})\, H_0(z) &= 0 \\ 1 + \rho' \frac{z - \beta}{(z-1)^2} \frac{z(z-1/\beta)}{(z-1)^2} &= 0 \end{aligned} \tag{7.10}$$

where $\beta = (\alpha - 1/2)/(\alpha + 1/2)$ and $\rho' = \rho(\alpha + 1/2)^2$.

The root locus for $\beta = -1$ ($\alpha = 0$) is sketched in Fig. 7.6a, and for $\beta = 0.5$ ($\alpha = 1.5$) in Fig. 7.6(b).

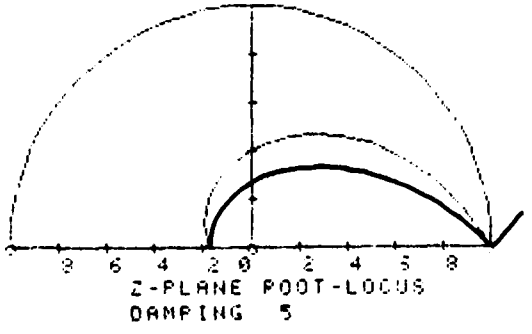


Figure 7.6a: Symmetric Root Locus for $1/s^2$ Plant and $\alpha = 0$

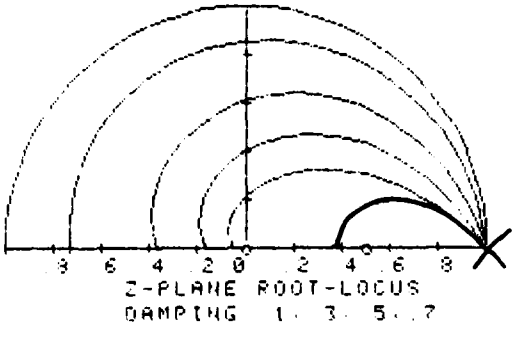


Figure 7.6b: Symmetric Root Locus for $1/s^2$ Plant and $\alpha = 1.5$

As can be seen from these sketches, the location of the optimal closed loop poles can be substantially varied by choice of α and ρ. However, in every case the resulting locations are optimal for some quadratic performance criterion of the form of Eq. (7.3).

7.3 POLE ASSIGNMENT BY STATE FEEDBACK

Once the dynamic performance has been expressed in terms of pole locations, it is necessary to consider the structure of the controller which will cause the closed loop system to have these poles. There are several techniques to solve this problem, among the most effective being the separation of the design into two parts: a state feedback control law and a state estimator to replace the unmeasured states with their estimates. We begin with the state feedback to assign closed loop poles to specified locations.

The algebra of the case is given by the state equations

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{7.11}$$

and the control law is given by

$$u = -Kx \tag{7.12}$$

where K is a row matrix.

The specifications are given by the fact that the closed loop poles should be at desired locations. One way of expressing this requirement is that the closed loop characteristic polynomial is specified as $\alpha_c(z)$. Combining these relations, we arrive at the requirement that

$$\det(zI - \Phi + \Gamma K) = \alpha_c(z) \tag{7.13}$$

by choice of K. Equation (7.13) is an algebraic equation in the components of the control law matrix K and one way to solve the problem is to work out the form of the determinant on the left hand side and equate the coefficients of the several powers of z term by term. One obtains n equations this way, and if u is a scalar, K has n components so we have n equations in n unknowns and the promise of a unique solution. A more interesting approach leads to an explicit formula easily implemented on a computer, first worked out by Dr. J. Ackermann (1972). The derivation given here was suggested by K.J. Åström in a private conversation with the author.

The derivation begins with the observation that if the closed loop system matrix is $\Phi_c = \Phi - \Gamma K$ and the closed loop characteristic polynomial is $\alpha_c(z)$ then a well known result of matrix algebra is the Cayley-Hamilton theorem which says that Φ_c satisfies $\alpha_c = 0$, namely

$$\Phi_c^n + \alpha_1\Phi_c^{n-1} + \ldots + \alpha_{n-1}\Phi_c + \alpha_n I = [0] \tag{7.14}$$

The closed loop system equations are

$$x(k+1) = \Phi_c x(k)$$

which has the easily obtained solution for any initial condition

$$x(k) = \Phi_c^k \underline{x}_0 \tag{7.15}$$

Now we can show that

$$x(k+n) + \alpha_1 x(k+n-1) + \ldots + \alpha_{n-1}x(k+1) + \alpha_2 x(k) = 0 \tag{7.16}$$

Because we obtain, with Eq. (7.15) substituted into Eq. (7.16),

$$\left(\Phi_c^n + \alpha_1\Phi_c^{n-1} + \ldots + \alpha_n I\right)\Phi_c^k \underline{x}_0 = 0 \tag{7.17}$$

Now we return to the question of the control and what control law K will cause Eq. (7.16) to be true for given Φ and Γ. Proceeding step by step with $x(k+1 = \Phi x(k) + \Gamma u(k)$ we find that Eq. (7.16) requires that

$$\left(\Phi^n + \alpha_1\Phi^{n-1} + \ldots + \alpha_n I\right)x(k) + \Gamma u(k+n-1) + (\Phi\Gamma + \alpha_1\Gamma)u(k+m-2) + \ldots$$
$$+ \left(\Phi^{n-1}\Gamma + \alpha_1\Phi^{n-1}\Gamma + \ldots + \alpha_{n-1}\Gamma\right)u(k) = 0 \tag{7.18}$$

Collecting terms, this expression can be written as

$$\alpha_c(\Phi)x(k) + \left[\Gamma \quad \Phi\Gamma \ldots \Phi^{n-1}\Gamma\right]\begin{bmatrix} 1 & \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} \\ 0 & 1 & \alpha_1 & \cdots & \cdot \\ & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \alpha_1 \\ & & & & 1 \end{bmatrix}\begin{matrix} u(k+n-1) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ u(k) \end{matrix} = 0 \tag{7.19}$$

We define the $n \times n$ matrix (u is a scalar here)

$$\mathcal{C} \triangleq \left[\Gamma \quad \Phi\Gamma \ \ldots \ \Phi^{n-1}\Gamma\right] \tag{7.20}$$

and

$$\mathcal{A} = \begin{bmatrix} 1 & \alpha_1 & \alpha_2 & \cdots & \\ 0 & 1 & \alpha_1 & & \\ \cdot & & & \cdot & \\ \cdot & & & \cdot & \alpha_1 \\ \cdot & & & \cdot & \\ \cdot & \cdots & \cdots & & 1 \end{bmatrix}$$

Then Eq. (7.19) can be written as

$$\mathcal{A}\begin{bmatrix} u(k+n-1) \\ \cdot \\ \cdot \\ \cdot \\ u(k) \end{bmatrix} = -\mathcal{C}^{-1}\,\alpha_c(\Phi)x(k)$$

The entry in the last row on the left is seen to be $u(k)$ alone so, finally, if $e_n^T \triangleq [0 \quad 0 \ \ldots \ 1]$, the $n^{\underline{th}}$ unit vector of order n,

$$u(k) = -e_n^T\,\mathcal{C}^{-1}\alpha_c(\Phi)x(k)$$

so that the control law is

$$K = e_n^T\,\mathcal{C}^{-1}\alpha_c(\Phi) \tag{7.21}$$

Equation (7.20) is Ackermann's formula.

The form of Eq. (7.21) immediately raises the question of the existence of a solution for K. Solution according to this formula requires that the matrix $\mathcal{C}$ given by Eq. (7.20) must be non-singular. This matrix is called the Controllability matrix and if it is singular then some dynamic mode of the system is not connected to the input and cannot be controlled. The structure of the system must be changed if this mode is unstable or is so slow that it can prevent the satisfactory control of noise and other disturbances. Computing the controllability of a dynamic model is an important requirement of computer aided control design.

To illustrate the design of state feedback two examples will be given. In the first instance, we consider the $1/s^2$ plant for which

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}; \qquad \Gamma = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}$$

and the closed loop characteristic equation is

$$\det(zI - \Phi + \Gamma K) = 0$$

$$\det\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix}\begin{bmatrix} K_1 & K_2 \end{bmatrix} = 0$$

$$z^2 + TK_2 + (T^2/2)K_1 - 2z + (T^2/2)K_1 - TK_2 + 1 = 0 \tag{7.22}$$

Suppose the sampling period is 0.1 sec and the dynamic response specifications can be expressed by the desired closed loop characteristic polynomial,

$$\alpha_c(z) = z^2 - 1.6z + 0.7 = 0 \tag{7.23}$$

which has poles at radius 0.836 and angle 17.0°. (In the s-plane, these correspond to $\omega_n = 3.6$, $\zeta = 0.5$).

Equating coefficients in Eq. (7.22) with Eq. (7.23), we can solve for the control law as

$$\begin{aligned} K_1 &\cong 10 \\ K_2 &\cong 3.5 \end{aligned} \tag{7.24}$$

These gains can also be computed by Ackermann's Formula, as shown by the computer printout in Fig. 7.7

```
PHI

1   .1
Ø   1

GAMMA

.ØØ5
.1

INPUT 2 POLES IN THE FORM
RADIUS,ANGLE(DEGREES)
POLE 1 =
?
.836,17

CONTROL GAIN
9.995444803
3.508122415
D C GAIN .1ØØØ45572729
```

Figure 7.7: Computer Print-out of Program CONLAW for $1/s^2$ Example

The response to an initial condition of $x_1(0) = 1.0$, also provided by the program CONLAW, is shown in Fig. 7.8. CONLAW is one of the programs in the set called DIGICON-85 for which a brief description is given in the appendix to this presentation.

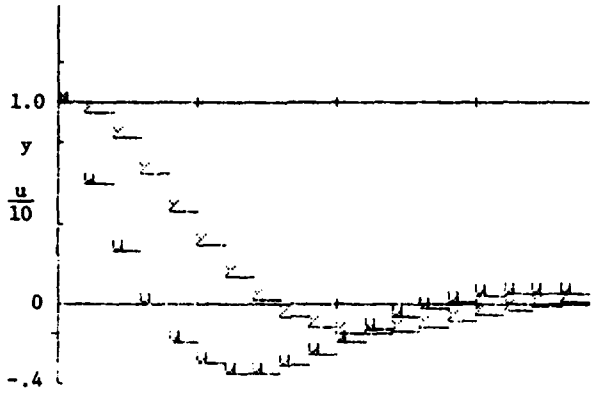


Figure 7.8: Response of $1/s^2$ Plant to Initial State as Produced by CONLAW

The second example illustrating the design by state feedback models a servomechanism with the sensor on a mass coupled to the motor by a flexible structure. The transfer function with time measured in units of 10 milliseconds is

$$\frac{\theta}{V} = \frac{K}{s(s + 0.513)(s + 0.24 \pm j3.12)} \tag{7.25}$$

With a sampling time of $T = 0.6$ which is 6 milliseconds in real time, the discrete system Φ and Γ are computed by the program SAMPLEØ which also calculates that the system has zeros at -6.99, -0.884 and -0.113 and has poles at 1.0, $-.257 \pm j.827$, and 0.735.

For this case, the specifications are satisfied by the fourth-order ITAE propotype with poles at radius 0.654, angle 72° and radius 0.535, angle 24°. The output of CONLAW is shown in Fig. 7.9 and the corresponding step response in Fig. 7.10.

```
INPUT 4 POLES IN THE FORM
RADIUS,ANGLE(DEGREES)
POLE 1 =
?
.654,72
POLE 3 =
.535,24
CONTROL GAIN
 4.41149595241
 5.48399355082
-3.36940938182
 .334075120487
TRY NEW POLE POSITIONS
?
N
```

Figure 7.9: Print-out of CONLAW for Fourth-order Servomechanism

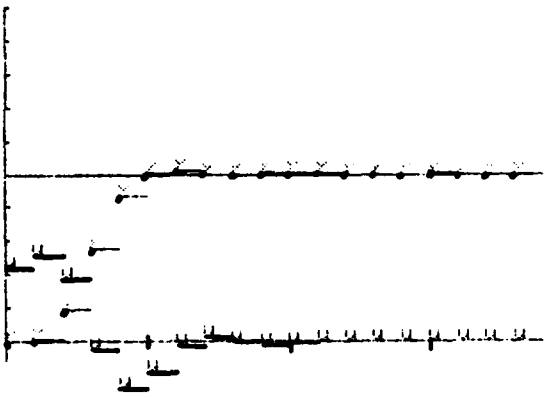


Figure 7.10: Step Response of Fourth-order Servomechanism showing ITAE Response. "y" Marks the Output; "u" Marks the Control

7.4 STATE ESTIMATION

The control given by state feedback assumes that all the states are available. In practice, of course, only a very few states are measured and a state feedback control law alone is almost never useable. However, we can replace the actual states by estimates and recover the response to command inputs as if true state feedback is used. The calculations go like this.

Given a system with equations

$$\begin{aligned} y(k) &= H\,x(k) \\ x(k+1) &= \Phi\,x(k) + \Gamma\,u(k) \end{aligned} \tag{7.26}$$

we want to obtain an estimate $\hat{x}(k)$ such that the error $\tilde{x} = x - \hat{x}$ is quickly small. One method is to use a model of the system, which would have output $\hat{y} = H\hat{x}$ and to use the error in the output to correct the model. The resulting equations are

$$\hat{x}(k+1) = \Phi\,\hat{x}(k) + \Gamma\,u(k) + L\big(y(k) - H\hat{x}(k)\big) \tag{7.27}$$

The error in this estimate is easily calculated to be

$$\tilde{x}(k+1) = (\Phi - LH)\tilde{x} \tag{7.28}$$

Two items are important to notice with respect to this error equation. In the first place, we notice that the control, which appears in Eq. (7.26) and (7.27) has cancelled out of the error equation, (7.28). Thus the estimation error is <u>independent</u> of the scheme used to compute the control. In the second place, the correction gain matrix, L, appears in the error equation in a manner that promises to give the designer some influence over the poles and hence the dynamic response of the estimator. In fact, since $\det A = \det A^T$, the characteristic equation of the error system (7.28) is

$$\begin{aligned} \det(zI - \Phi + LH) &= 0 \\ \det(zI - \Phi^T + H^T L^T) &= 0 \end{aligned} \tag{7.29}$$

and Eq. (7.29) is identical to the control equation (7.13) except in place of Φ we have Φ^T and for Γ we have H^T and for K we solve for L^T. Furthermore, because of this identity, we can be sure that a solution for L exists for an arbitrary Φ and H, provided Φ^T and H^T are "controllable." The property is actually called "observable" and, if it fails the cause is a dynamic mode which <u>cannot be seen</u> from the sensor output y. Also, because of this identity, we can solve for L by an estimator version of Ackermann's Formula.

The estimator described by Eq. (7.27) computes the estimate at time $k+1$ from the sensor output at time k and is called a prediction estimator. Two natural variations on the estimator equations give the current estimator and the reduced order estimator. The current estimator equations are

$$\begin{aligned} \hat{x}(k) &= \bar{x}(k) + L\big(y(k) - H\,\bar{x}(k)\big) && \text{Measurement up-date} \\ \bar{x}(k+1) &= \Phi\,\hat{x}(k) + \Gamma\,u(k) && \text{Time up-date} \end{aligned} \tag{7.30}$$

These equations are structurally the same as the Kalman filter. The Kalman filter involves an optimal choice of L for a specific set of assumptions. The reduced order estimator is a prediction estimator for the reduced state which remains when we use the output $y(k)$ as a known state which does not need to be estimated. For details, see Franklin and Powell (1980).

7.5 COMBINED CONTROL AND ESTIMATION

If we now use the states estimated by Eq. (7.27) in place of the actual states in the control law, we obtain the combined system

$$\begin{aligned} x(k+1) &= \Phi\, x(k) + \Gamma\, u(k) \\ \hat{x}(k+1) &= \Phi\, \hat{x}(k) + \Gamma\, u(k) + L(y - H\hat{x}) \\ u(k) &= -K\hat{x}(k) \\ y(k) &= Hx(k) \end{aligned} \tag{7.31}$$

If we replace $\hat{x}$ by $x - \tilde{x}$ and substitute for u and y in (7.31), we obtain the combined equations

$$\begin{aligned} x(k+1) &= (\Phi - K)x(k) - \Gamma K\tilde{x}(k) \\ \tilde{x}(k+1) &= (\Phi - LH)\tilde{x}(k) \end{aligned} \tag{7.32}$$

Since the second equation in (7.32) depends only on $\tilde{x}$, the dynamics of this equation are those of $\Phi - LH$. These are set by the designer by choice of L to correspond to an arbitrary estimator characteristic polynomial $\alpha_e(z)$. The first equation in (7.32) has $\tilde{x}$ as a forcing function, but the feedback is entirely in the matrix $\Phi - \Gamma K$ which imparts dynamics according to the control characteristic polynomial $\alpha_c(z)$, selected by the designer by choice of K. The overall characteristic polynomial is the product $\alpha_c(z)\alpha_e(z)$. Thus we see that the control and estimation separate.

If we have an external input as a reference command, this signal can be added to the system via the estimator in the form of input matrices M and N as follows:

$$\begin{aligned} \hat{x}(k+1) &= (\Phi - \Gamma K)\hat{x} + L(y - H\hat{x}) + Mr \\ u &= -K\hat{x} + Nr \end{aligned} \tag{7.33}$$

By choice of M and N the designer can select zeros so as to cause the overall transfer function to be

$$\frac{Y(z)}{U(z)} = \frac{T(z)b(z)}{\alpha_e(z)\alpha_c(z)} \tag{7.34}$$

where $b(z)$ are the zeros of the process being controlled, $\alpha_e(z)$ is selected arbitrarily by choice of L, $\alpha_c(z)$ is selected arbitrarily by choice of K, $T(z)$ is selected arbitrarily by choice of M, and the overall d.c. gain is selected by choice of N. It is most common to select $N=1$ and $M=\Gamma$ for which choice $T(z) = \alpha_e(z)$ and the estimator poles are cancelled from the command input-output transfer function.

As an example of this design method, we consider again the $1/s^2$ plant with sampling rate $T=0.1$ whose response with state feedback is shown in Fig. 7.8. The estimator used is a reduced order estimator with L chosen to make $\alpha_e(z) = z-0.5$. The input matrices M and N are selected to make $T(z) = \alpha_e(z)$ and gain such that $r(\infty) - y(\infty) = 0$ for constant reference input. Figure 7.11 shows the printout of the program REDEST, which computes L and the dynamics equations of the controller as given by equations 7.34.

The step response of the resulting controller is shown in Fig. 7.12, as given by the program RESP. The response to an initial state $x_1(0) = 1$ is shown in Fig. 7.13. Figure 7.13 should be compared to Fig. 7.8 where it will be seen that the response with estimator has more "undershoot" (-0.5 compared to -0.16) and a longer duration. This is the price paid for having to estimate the second state compared to measuring it directly.

An alternative to the state space methods given above is to postulate a dynamic controller with two inputs (y_r and y) and one output (u) and to solve for the transfer functions directly. We model the plant as a transfer function rather than by state equations

$$\frac{Y(z)}{U(z)} = \frac{b(z)}{a(z)} \tag{7.35}$$

and model the controller similarly as

$$R(z)U(z) = -S(z)Y(z) + T(z)Y_r \tag{7.36}$$

We use $Y_r(z)$ for the command signal to avoid confusion with the polynomial $R(z)$ in Eq. (7.36). To complete the design, we request that the closed loop transfer function by given by

$$\frac{Y}{Y_r} = \frac{T(z)b(z)}{\alpha_e(z)\alpha_c(z)} \tag{7.37}$$

From Eqs. (7.35) and (7.36) we have

$$\begin{aligned} a(z)Y(z) &= b(z)\left[\frac{-S}{R}\,Y(z) + \frac{T(z)}{R(z)}\,Y_r(z)\right] \\ [R(z)a(z) + b(z)S(z)]Y(z) &= b(z)T(z)Y_r(z) \end{aligned} \tag{7.38}$$

Comparing Eq. (7.38) with Eq. (7.37) we see immediately that the design can be accomplished if we can solve the Diophantine Equation

$$R(z)a(z) + b(z)S(z) = \alpha_e(z)\alpha_c(z) \tag{7.39}$$

for given arbitrary a, b, α_e, α_c. If $a(z)$ is of degree n and b is of degree n or less and $\alpha_e\alpha_c$

```
OLD FILE NAME
?
92.DE
PHI
   1.ØØØ    .1ØØ
   Ø.ØØØ   1.ØØØ
GAMMA
    .ØØ5
    .1ØØ
INPUT 1  POLES IN THE FORM
RADIUS,ANGLE(DEGREES)
POLE 1 =
?
.5,Ø
ESTIMATOR GAIN
-.5
AØ
    .237
BØ
 16.Ø311163168
CØ
  1.ØØØ
DØ
-27.5495Ø6ØØ37
CONTROLLER DENOM
 1
-.236689Ø81988
CONTROLLER NUMBER
-27.5495Ø6ØØ37
 22.551783Ø62
MØ
  -2.632
NØ
   9.995
CONTROLLER REF INPUT NUMERATOR
 9.99544480323
-4.99772240161
 NEW POLE LOCATIONS
?
```

Figure 7.11: Print-out for Program REDEST for $1/s^2$ Plant

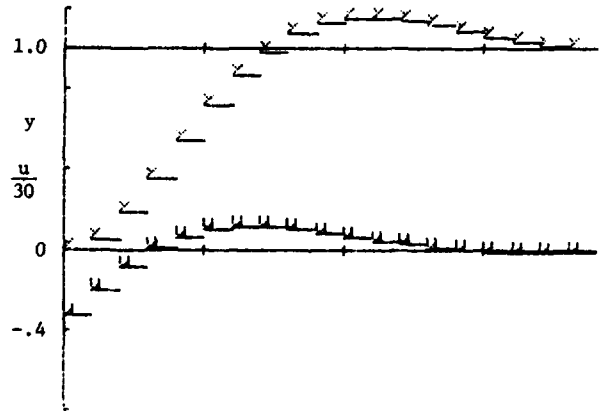


Figure 7.12: Step Response of $1/s^2$ Plant with Controller Including Estimator

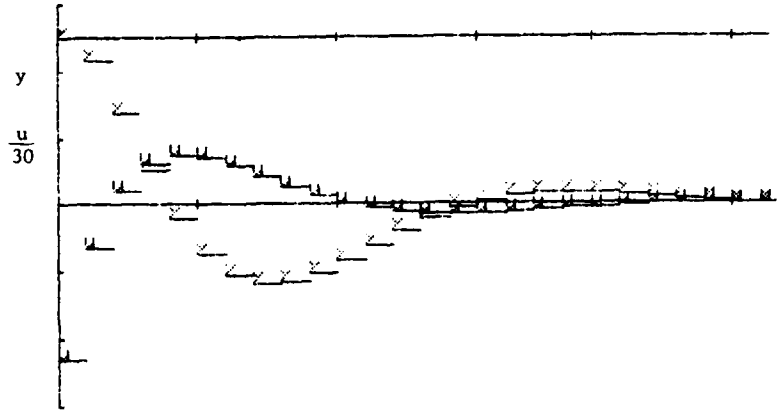


Figure 7.13: Response of $1/s^2$ Plant to Initial State $x_1(0) = 1$

is of degree $2n$, then a solution exists for R and S of degree $n-1$ each if and only if a and b have no common factors. Again using the $1/s^2$ plant for illustration, we have

$$G(z) = \frac{T^2}{2} \frac{z+1}{(z-1)^2}$$

from which

$$a(z) = z^2 - 2z + 1$$

$$b(z) = \frac{T^2}{2}(z + 1)$$

$$\alpha_c(z) = z^2 - 1.6z + .7$$

$$\alpha_e(z) = z - .5$$

Letting the constant $T^2/2$ be absorbed in $S(z)$ for simplicity, Eq. (7.39) appears as

$$(r_0z+r_1)(z^2-2z+1) + (s_0z+s_1)(z+1) = z^3 - 2.1z^2 + 1.5z - .35$$

Equating equal powers of z in this expression we see immediately that $r_0=1$ and for the other unknowns we write

$$\begin{bmatrix} 1 & 1 & 0 \\ -2 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{matrix} r_1 \\ s_0 \\ s_1 \end{matrix} = \begin{bmatrix} -0.1 \\ 0.5 \\ -0.35 \end{bmatrix}$$

From these equations we find

$$r_1 = -.2375$$

$$s_0 = .1375$$

$$s_1 = -.1125$$

Removing the normalizing factor of $T^2/2 = 0.005$ we get

$$s_0 = 27.5$$

$$s_1 = -22.5$$

These are the values given as the controller numerator in Fig. 7.11 computed by state variable methods.

7.6 CONCLUSIONS

The design of digital control systems for arbitrary pole placement can be (and has been) implemented in a very straightforward way on a computer. However, the simple algebraic formulas suitable for low order non-critical systems require considerable care when the intention is to implement a control design package which will provide the designer with trouble-free, reliable, inexpensive graphic responses to the requirements of computing controllers and finding discrete poles, zeros, controllability, observability, control laws, estimator laws, root loci, frequency responses and alternative parameters for realization. To meet these objectives a great deal of work needs to be done.

7.7 APPENDIX

DIGICON/85:
Computer Aid for the Design of Digital Controls Using the HP-85

I. Introduction

The set of programs which constitute DIGICON/85 are for the computation of digital controls by root locus and pole placement methods. The programs are written in BASIC for use on the HP-85 desk top computer with a MATRIX and memory ROM expansion and consist of a set of individual programs to be loaded from the tape cartridge.

The notation of the problems to be solved may be summarized as follows. The process (chemical plant, vehicle, servomechanism, or whatever) is described by equations in the output, y, the input u, and the state as follows:

$$y_k = Hx_k + Ju_k \qquad \text{Output equation} \tag{1a}$$

$$x_{k+1} = Fx_k + G(u_k + w_k) \qquad \text{State up-date equation} \tag{1b}$$

the transfer function of the process is

$$g(z) = H(zI - F)^{-1}G + J \tag{2}$$

$$= \frac{b(z)}{a(z)} \tag{3}$$

$$= \frac{b_o z^n + b_1 z^{n-1} + \ldots + b_n}{z^n + a_1 z^{n-1} + \ldots + a_n} \tag{4}$$

$$= K \frac{\Pi(z - z_1)}{\Pi(z - p_1)}$$

The process is to be controlled by a dynamic system which may or may not include integral action. With no integral action, the controller equations are in terms of the process output y, the process input u, the controller state $x1$, and the external command input as follows:

$$u_k = Cx1_k + Dy_k + Nr_k \tag{5a}$$

$$x1_{k+1} = Ax1_k + By_k + Mr_k \tag{5b}$$

If the controller is to include integral action, then the equations include the integrator state, $x2$, as follows:

$$u_k = x2_k + Cx1_k + Dy_k + Nx2_k \tag{6a}$$

$$x1_{k+1} = Ax1_k + By_k + Mx2_k \tag{6b}$$

$$x2_{k+1} = x2_k + K_o(r_k - y_k) \tag{6c}$$

The controller has two transfer functions since the output of the controller, u (which is the input to the process under control) depends on <u>two</u> inputs: a feedback input from y and a feedforward input from r with no integral action, and from $x2$ with integral action. For the case of Eq. (6) with no integral action, we write

$$U(z) = \frac{C_1(z)Y(z) + C_2(z)R(z)}{d(z)} \tag{7}$$

where the roots of $d(z) = 0$ are the controller poles, the roots of $C_1(z) = 0$ are the feedback zeros, and the roots of $C_2(z) = 0$ are the feedforward zeros. In a totally similar way we express the control with integral action by the two equations

$$U(z) = \frac{C_1(z)Y(z) + C_2(z)X2(z)}{d(z)} \tag{8}$$

$$X2(z) = \frac{K_0}{z - 1}(R(z) - Y(z)) \tag{9}$$

where as before d, C_1 and C_2 correspond to poles, feedback zeros, and feedforward zeros.[1] In Eq. (6c), the parameter K_o is part of the control law designed for a system with integral control. We can show that using the effective states $e = y-r$ and $z_k = x_{k+1}-x_k$ and controls $\nu_k = u_{k+1}-u_k$ then, if $r_{k+1}-r_k = 0$ [a constant or step input] the equation is

$$\begin{bmatrix} e \\ z \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & H \\ 0 & F \end{bmatrix} \begin{matrix} e_k \\ z_k \end{matrix} + \begin{bmatrix} 0 \\ G \end{bmatrix} \nu_k \tag{10}$$

From Eq. (10) a control law is generated as

$$\nu = -K_o e - \underline{K} z \tag{11}$$

and the K_o in (11) is the parameter that appears in the realization (6).

The organization of the computer programs of DIGICON/85 is as a set of separate programs to perform different parts of the task of the design process and in each case the data for the particular task is obtained either from the keyboard or from a data file on the machine tape cartridge. The data structure is a sequential file so we must define beforehand the exact sequence of data to be stored. for DIGICON we have two sequences, depending on whether the data represents a process (including perhaps integral control) or a controller (which will include feedforward zeros.) The process data the sequence is

N1,N2,N3,N4,N5,T,L1,Z1,F,G,H,J,A,R9,B,R8,K,L

and for controller data, the sequence is identical to R8 but is then different, as follows:

N1,N2,N3,N4,N4,T,L1,Z1,F,G,H,J,A,R9,B,R8,C,R7,M,N

The meaning of these data is as follows:

N1 the number of states

N2 the number of inputs

N3 the number of outputs

N4 the number of finite zeros

N5 the number of finite feedforward zeros in a controller (set to zero in a file of process data)

T sampling period (set to zero in a file of data for a continuous (non-discrete) model)

L1 pure time delay in the model

Z1 an indicator or switch variable to identify the data type

Z1 = 0 implies a file of process data with no extra integral action included

Z1 = 1 implies a file of process date with integral action

Z1 = 2 implies a file of controller data

F process or controller state feedback matrix N1 × N1

G process or controller input matrix N1 × N2

H process or controller output matrix N3 × N1

J process or controller direct transmission matrix N3 × N2

A matrix coefficients of system characteristic polynomial. If the polynomial is $\sum_{i=0}^{n} a_i z^{n-i}$ then $A(I) = a_{I+1}$. A is N1 + 1 × 1

R9 matrix of system poles, the roots to $a(z) = 0$. R9 is N1 × 2 and the first column holds the real parts and the second column holds the corresponding imaginary parts

B matrix of coefficients of system zero polynomial N1 + 1 × 1 stored as in A

R8 matrix like R9 but holding the system zeros. R8 is N4 × 2

K Control law

L Estimator law

C matrix of coefficients of the feedforward zeros polynomial of a controller. Elements stored as in A and B

[1] It is recognized that (6) does not represent the most general case since terms $N1_r$ could be added to (6a) and $M1_r$ could be added to (6b). These terms would change the location of the feedforward zeros.

R7 matrix of roots of feedforward zeros polynomial, stored as in R8 and R9. R7 is N5 × 2

M matrix of controller feedforward as in equations (5) and (6). N1 × 1

N matrix of controller direct transmission as in (5) and (6)

To establish, manipulate and produce a design from the data base there are in DIGICON eight programs. Brief descriptions of these programs follow.

II. Program Descriptions

ROOTLC: This is a program to plot the root locus of the equation $a(z) + Kb(z) = 0$ or $a(s) + Ke^{-\lambda s}b(s) = 0$ versus K. The data for a and b may be obtained either from the keyboard, a file, or a set of DATA statements contained in the program. The program computes points on the locus by searching for points when the phase of b/a is 180° and K is increasing. The user specifies a starting point (typically a root of $a = 0$, a pole) and also specifies a distance between points. If the distance is too large the program will not converge to a valid point and a new distance must be tried. No provision for marking points of a specific gain are made but the user is given a tabulation of root locations and gain values from which points of interest can be marked.

The program will plot the results in either the s-plane or the z-plane.

INPUT: This is a program to build a file of data describing either a process or a controller. The data is to be provided from the keyboard in state variable form. The program converts the state description to observer canonical form and thus obtains the characteristic polynomial and the zeros polynomial. These polynomials are solved and the roots obtained. Program INPUT initializes K and L for a process data file to zero and stores all this data in a file named by the user. A convenient mnemonic for files is to use a suffix to indicate file type. For example, the double integrator problem might have associated with it the three files S2.CS for the continuous process data (.CS for continuous); S2.DE for the discrete model data and S2.CO for the corresponding controller.

READ: This is a program to read a file, print a copy of its contents and make another copy on another tape, if desired.

SAMPLØ: This is a program to compute the discrete equation matrices ϕ, Γ, H1, J1 from a file of data for a continuous model. The program includes the capability to have a pure time delay in the model and to include integral control. In the latter case, the discrete equations are in the error space form of equation (10). The algorithm is given in Franklin and Powell, pages 171-177.

POLY: This is a program to solve for the roots of a polynomial. The program is an adaptation of the standard pac program included with the HP-85.

CONLAW: This is a program to compute K, the control law, by "pole placement" so that the closed loop system

$$x_{k+1} = (F - GK)x_k \tag{12}$$

will have a specified characteristic equation. The desired poles are specified by the user and the gains, K, are computed. A transient response of (12) is computed to aid in the evaluation of a selected set of poles.

REDEST: This program computes a Reduced order Estimator for a given process. The design is based on pole placement of the estimator poles and includes feedforward to guarantee that the reference input does not excite these. The program allows for integral control and produces matrices for either equation (5) or (6) as is appropriate.

RESP: This is a program to compute the RESPonse of a discrete system using either equation (5) or (6) as the controller, and (1) for the process. The disturbance w_k and reference r_k are constants as step functions.

7.8 REFERENCES

G. F. Franklin and J. D. Powell, Digital Control of Dynamic Systems, Re-ding, MA: Addison Wesley, 1980.

J. Ackermann, "Der Entwurf Linearer Regelungssysteme im Zustandsraum," Regelungstechnik und Prozessdatenverarbeitung, 7, 297-300, 1972.

SYSTEMATIC COMPUTER AIDED CONTROL DESIGN

Georg Grübel and Gerhard Kreisselmeier

DFVLR-Institut für Dynamik der Flugsysteme

Oberpfaffenhofen

D-8031 Wessling

F.R. Germany

Summary

Computerized synthesis techniques of modern control theory are in widespread use, but a number of fundamental design problems still remain. We call them: the design specifications problem, the free design parameter problem, the plant complexity versus controller simplicity problem and the dirty design environment problem. A design procedure which comes close to solving these design problems is recommended: It is an iterative design technique using a performance index vector which provides a systematic guidance for the designer to take care of multiple design objectives simultaneously and individually. As a design tool unconstrained parameter optimization is used. A practical application is briefly reported: The design of a robust control loop for a fighter aircraft where 42 performance criteria of 9 different sorts have been considered simultaneously.

1. Introduction

The ultimately achievable performance of a control system is limited by hardware constraints such as the dynamics of the plant, the actuators, and the sensors. The actually achieved performance is, in addition, a product of control design: The specification of a controller structure and the tuning of the controller parameters determine how well the design objectives are met within the given limitations. Hence, in order to improve control performance one should improve the control design as well as the underlying design procedures.

Design is a trade-off between various competing objectives. These objectives reflect the properties a good system should satisfy. They are rarely complete and quantitatively specified at the outset. Rather they are an open list and formulated in qualitative terms such as: The control system should be
- "fast and smooth" in response to reference inputs, there should be
- "small static and dynamic errors" due to disturbances. The system should have a
- "good stability margin" in order to
- "tolerate 'large' parameter variations and modelling inaccuracies." But the system should also have a
- "well limited bandwidth" and "low feedback gains" in order to be
- "insensitive with respect to measurement noise". Furthermore control action should be
- "well within actuator saturation limits" and so on ...

The designer has to satisfy all these objectives in the best possible way. How can this be achieved?

2. Problems when Using Synthesis Techniques of Modern Control Theory

Modern control theory using state feedback and observers e.g. for linear optimal control, pole placement, decoupling and disturbance accomodation, provides much insight into the analysis and synthesis of linear systems. Combining this conceptual insight with the numerical efficiency of a digital computer results in a set of useful tools to assist the design process. But although such computerized control synthesis techniques are in widespread use, a number of fundamental design problems still remain:

The Design Specifications Problem: A synthesis technique of modern control theory usually focuses on one type of design specification only (e.g. pole location; decoupling; disturbance accomodation) and is based on an exact numerical specification (e.g. all pole positions, exact decoupling, exact asymptotic disturbance accomodation for prespecified frequencies). This requests the designer to map all design objectives onto a single type of mathematical criterion (e.g. pole placement) which often narrows too much the designer's overall view of system performance. Furthermore an exact quantitative specification of design objectives is rarely possible in practice. What about a combination of different types of criteria to cover the overall system performance? What about specifying bounds rather than specifying exact numerical values for performance measures?

The Free Design Parameter Problem: The free design parameters in the Riccati formalism are the weighting coefficients in the cost functional. How to choose these weighting coefficients systematically? How to choose pole locations as the free design parameters in a pole placement approach? What about additional free parameters in, say, a full order state observer? To solve these decision problems, an iterative design loop is necessary based on an analysis of the overall system performance and notably the plant limitations. Is there a strategy available for systematically reaching an overall compromise?

The Plant Complexity vs. Controller Simplicity Problem: Modern control theory provides analytical relations between the structure of the plant and the structure of a suitable controller. This means: Simple plant models yield simple controllers, complex plant models yield complex controllers. What about simple controllers for complex plants which, as we know, often do very well in practice? We need an efficient design procedure for simple controllers taking into account realistic and thereby rather complex plant models. Why not use parameter optimization?

The Dirty Design Environment Problem: Modern control theory yields neat solutions for neat problems. Unfortunately, the design environment in practice is rather dirty. What about plant nonlinearities, parameter uncertainties, changing characteristics and control saturation? Can we handle such effects directly without the need to transcribe them into the linear framework of modern control synthesis?

These design problems when using synthesis techniques of modern control theory are conceptual ones: They are consequences of a search for analytical solutions, where the computer serves only the purpose of fast numerical evaluations of (complex) analytical relations. They cannot be removed by using faster computers or by improving the man-computer interface. What we need for control design is a multi-objective design procedure with a systematic decision strategy as a complement to control theory and the use of computers as fast searching devices for finding a "best" solution in a (nonlinearly defined) set of possible candidates.

In order to handle multiple design objectives in a systematic way, they have to be formulated by mathematical criteria. This is a question of control theory, where either the state space or the frequency domain may turn out to be more appropriate for quantifying a particular design objective. Control theory also helps to decide what kind of design criteria have to be included for a special design purpose and it yields some qualitative insight into conflicting demands. But control theory yields no strategy to handle design as a multiple-criteria decision problem as it naturally results out of the multitude of different design objectives which have to be considered. Hence we are looking for a systematic framework which guides the designer to cope with a multiple-criteria control design problem.

3. Systematic Design Via a Performance Index Vector and Parameter Optimization

A design procedure which comes close to solve most of the above mentioned design problems has been suggested in [1]. It has proven to be very useful in various practical designs and can be summarized as follows:

Given a plant, and given a suitably chosen controller structure with free parameters $\underline{k} = [k_1,\ldots,k_n]$ the design problem is to determine suitable values for $\underline{k}$.

In order that the design can proceed in a systematic fashion it is necessary that all design objectives are taken care of explicitly in the design. Therefore, every design objective shall be rated quantitatively by means of a suitable performance index $J_i(\underline{k})$.

A performance index $J_i(\underline{k})$ is called suitable if $J_i(\underline{k}') < J_i(\underline{k}'')$ means that $\underline{k}'$ satisfies the i-th control objective better than $\underline{k}''$ does, and if $J_i(\underline{k})$ is a sufficiently smooth function.

For ease of notation we define the performance vector $\underline{J}(\underline{k}) = [J_1(\underline{k}),\ldots,J_L(\underline{k})]$. Moreover, for any two real vectors $\underline{x}$, $\underline{y}$ we say that $\underline{x} < \underline{y}$ if for all components $x_i \leq y_i$ and $\underline{x} \neq \underline{y}$.

Iterative Technique for a Systematic Design

The design technique is iterative, where each design iteration (the ν-th, say) comprises two steps:

Step 1: Choose a vector of design parameters $\underline{c}^\nu$ ($c_i^\nu > 0$) such that

$$\underline{J}(\underline{k}^{\nu-1}) < \underline{c} < \underline{c}^{\nu-1}. \tag{1}$$

Step 2: Find $\underline{k}^\nu$ and $\alpha_o < 1$ such that

$$\underline{J}(\underline{k}^\nu) < \alpha_o \underline{c}^\nu. \tag{2}$$

The design iteration is initialized by using an initial guess $\underline{k}^o$ for the controller parameters and by taking $\underline{c}^o$ sufficiently large.

In step 1 of every design iteration, $\underline{c}^\nu$ has to be chosen where (1) provides a well defined margin. This choice determines the design direction. Step 2 then provides the margin for the next design iteration.

It is desirable that this margin be as large as possible, i.e. α_o should be as small as possible.

After the ν-th design iteration we have from (1), (2) that

$$\underline{J}(\underline{k}^{\nu}) < \underline{c}^{\nu} < \underline{c}^{\nu-1} < \ldots < \underline{c}^{o}, \tag{3}$$

i.e. we have a monotonically decreasing sequence of design vectors and the performance vector has become less than all of them. This establishes the systematic behaviour of the design process.

Design Tool

Basically, any method which is able to perform step 2 of the design iterations can serve as a design tool. Here we shall focus on a method which, in addition, tends to make α_o as small as possible.

The smallest value of α_o satisfying $\underline{J}(\underline{k}) < \alpha_o \underline{c}^{\nu}$ is given (as a function of $\underline{k}$) by

$$\alpha_o(\underline{k}) = \max_{1 \le i \le L} \{J_i(\underline{k})/c_i^{\nu}\}. \tag{4}$$

Since $\alpha_o(\underline{k})$ is not continuously differentiable everywhere, we consider a smooth, i.e. at least twice continuously differentiable approximation $\alpha(\underline{k})$ instead, where

$$\alpha(\underline{k}) = \frac{1}{\rho} \ln \sum_{i=1}^{L} \exp\{\rho J_i(\underline{k})/c_i^{\nu}\} \tag{5}$$

and $\rho > 0$ is arbitrary. It can be verified that

$$\alpha(\underline{k}) = \alpha_o(\underline{k}) + \frac{1}{\rho} \ln \sum_{i=1}^{L} \exp\{\rho[\frac{J_i(\underline{k})}{c_i^{\nu}} - \alpha_o(\underline{k})]\}. \tag{6}$$

Hence $0 \le \alpha - \alpha_o \le (\ln L)/\rho$, i.e. α can be made as close to α_o as desired by choice of ρ.

The minimization of $\alpha(\underline{k})$ (for example, by applying Powell's method for function minimization without calculating derivatives [2]) can now be used as a design tool for performing step 2 in each design iteration.

The sequence of design iterations finally terminates when the minimization of $\alpha(\underline{k})$ results in a $\underline{k}^{\nu}$ with $\alpha_o(\underline{k}^{\nu}) \ge 1$.

As in all parameter optimization problems there is a possibility of local minima. Therefore, instead of using efficient local optimization algorithms such as Powell's [2], one may as well use global optimization algorithms such as random search algorithms. However, the latter are known to be less efficient.

Note also that the use of optimization algorithms which assume that the function to be minimized is twice continuously differentiable (which most efficient local algorithms do), requires that the performance indices themselves must be twice continuously differentiable. This imposes a certain restriction on the mathematical formulation of performance indices.

4. Practical Application Considerations

The above design procedure using a performance index vector and parameter optimization solves most of the design problems stated in section 2:

The design procedure provides a systematic framework to take care of multiple design objectives simultaneously and individually. The design objectives can be formulated by criteria of different kind either in state space or frequency domain. Hence the design specifications problem can be tackled in the most direct way. However, the design procedure does not provide a priori guidelines what design objectives are appropriate for a particular design problem. It remains in the designer's responsibility to decide what design criteria he shall use. This decision must be based on his knowledge of control theory and the operational requirements he has to satisfy.

The design procedure provides a systematic guidance for the designer to cope with the free design parameter problem: For each individual perfomance index J_i there is associated a free design parameter c_i which can be chosen by the designer within a well-defined margin. Typically, one starts with sufficiently large values for the design parameters and then successively reduces them so as to improve certain criteria while keeping possible deteriorations of others in tolerable bounds. Whatever choice c_i^{ν} in the given margin is taken, it is always guaranteed that $J_i \le c_i^{\nu}$ and hence a possible degradation of the performance index J_i is well bounded. If $c_i^{\nu} = J_i(\underline{k}^{\nu-1})$ is chosen, then an improvement of J_i is guaranteed provided only that such an improvement is possible at all at the expense of degrading other performance indices. This allows to explore the design possibilities, and to reach a desirable trade-off between competing objectives, step by step.

In each step it is the responsibility of the designer to decide upon the design direction, i.e. which performance index shall be improved and what is a permissible expense in degrading other performance indices. The design procedure then guarantees a step by step monotonic improvement in the direction specified by the designer. The admissible margins of the design parameters which are updated in each design iteration, yield some information how easy or difficult it is to reach the individual design objectives. Hence this design procedure makes it possible to explore the system's limitations in control design.

The design approach is well suited to deal with practical controller realization constraints. This is a consequence of using parameter optimization as a design tool instead of using an analytical control synthesis technique. There is no "controller simplicity vs. plant complexity problem" as we called it earlier. The designer can specify the controller structure at will. He is free to combine control considerations with technical realizability constraints. This freedom, however, calls much at the designer's experience and physical insight into the type of system to be controlled. Furthermore, in view of the numerical convergence properties of parameter optimization, one also has to take some care in mathematically specifying the controller structure, i.e. in parameterizing the control law: There must not be redundant parameters. In addition, the free parameter to be optimized should have the same order of magnitude, i.e. proper scaling may be necessary. On the other hand, the freedom to specify a controller structure makes it possible to develop a proper structure in an iterative process: Start with a simple structure, say a P-I controller, and extend it successively by additional dynamic degrees of freedom, i.e. use a higher order controller, if the design iterations yield no further improvement with the formerly chosen structure. In this way, one can explore the trade-offs between controller simplicity and control system performance.

The design procedure is open to handle complex, nonlinear systems. There is no conceptual necessity for linearized or low order plant models. But there may be computer time limitations: In view of parameter optimization the performance indices have to be evaluated very often and hence sufficiently fast. For nonlinear systems there are usually no analytical relations for such evaluations, rather the performance indices have to be computed on the basis of system simulations. This limits the complexity of system models to be used by the speed of available computers and the efficiency of available simulation software.

For practical applications of the design procedure via performance index vector and parameter optimization, a user-oriented, modular design software package REMVG [3] is available. This design package has been successfully applied in solving various non-standard design problems. One such application is briefly described in the next section.

5. Application Example: Robust Control Loop Design for a Fighter Aircraft

This application is well documented in [4]. Here we shall give a brief overview of the problem to be handled, the sort and number of performance indices used and the final result.

Control problem: For a fighter aircraft (type F-4C) a stability augmentation system shall be designed to improve the longitudinal handling qualities. This shall be achieved with a fixed gain controller which covers the whole flight envelope without gain scheduling. Furthermore only pitch rate ($\dot{\theta}$), the variable to be controlled, shall be used for feedback. As structure of the controller, a third-order compensator has been specified, where ten constant controller parameters k_j are to be assigned (figure 1). The longitudinal motion of the aircraft is modeled by a linearized second order short period motion description of the aircraft plus a first order actuator system. In the flight evelope, five (extreme) flight conditions shall be considered (figure 2).

Performance Criteria: The following performance criteria are specified for each of the five selected flight conditions.

Trajectory criteria for step response:

$$1) \quad J_i(\underline{k}) = \int_0^{T_i} [\dot{\theta}_i(t)/\dot{\theta}_i(\infty) - \dot{\theta}_m(\alpha_i \cdot t)]^2 dt \qquad (i = 1,\ldots 5)$$

Here $\dot{\theta}_m$ is a desired, normalized step response modeled by a second-order system, α_i denote different time scales for each individual flight condition.

$$2) \quad J_{5+i}(\underline{k}) = \int_0^{T_i} [\dot{n}_i(t)/n_i(\infty)]^2 dt \qquad (i = 1,\ldots 5)$$

Here $\dot{n}_i$ is the associated rate of elevator motion.

Trajectory criteria for disturbance rejection:

$$3) \quad J_{10+i}(\underline{k}) = \int_{t_i}^{T_i} \dot{\theta}_i^2(t)\, dt \qquad (i = 1,\ldots 5)$$

Here t_i denote the desired settling times, after which the pitch rate regulation error is to be uniformly less in magnitude than 5 per cent of the initial perturbation.

$$4) \quad J_{15+i}(\underline{k}) = \int_{o}^{T_i'} [\dot{\eta}_i(t)]^2 dt \qquad (i = 1,\ldots 5)$$

Here $\dot{\eta}_i$ is the associated rate of elevator motion for disturbance rejection.

Eigenvalue criteria:

To guarantee a desired degree of damping:

$$5) \quad J_{20+i}(\underline{k}) = [\mathrm{Im}(\lambda)/\mathrm{Re}(\lambda)]_{max}^{(i)} \qquad (i = 1,\ldots 5)$$

To guarantee a desired degree of absolute stability:

$$6) \quad J_{25+i}(\underline{k}) = \exp\{[\mathrm{Re}(\lambda)]_{max}^{(i)}\} \qquad (i = 1,\ldots 5)$$

To limit the maximum eigenfrequency in order to avoid structural mode excitation:

$$7) \quad J_{30+i}(\underline{k}) = |\lambda|_{max}^{(i)}. \qquad (i = 1,\ldots 5)$$

Controller coefficients criteria:

To bound the controller eigenvalues:

$$8) \quad J_{35+j}(\underline{k}) = 1/|k_j| \qquad (j = 1,2,3)$$

To limit feedback gains:

$$9) \quad J_{35+j}(\underline{k}) = |k_j|. \qquad (j = 4,\ldots 7)$$

Hence in this control design, 9 different sorts of performance criteria are used. The total number of performance criteria is 42. The number of controller parameters to be designed is 10.

Result:

The final design result is shown in figures 3 and 4: It was possible to design a fixed gain controller which covers the entire flight envelope without gain scheduling and which uses pitch rate feedback only. This novel result has been achieved in about 20 design steps, where each design step required about 60 s computer time on a AMDAHL 470/V6 computer. More details and results can be found in [4].

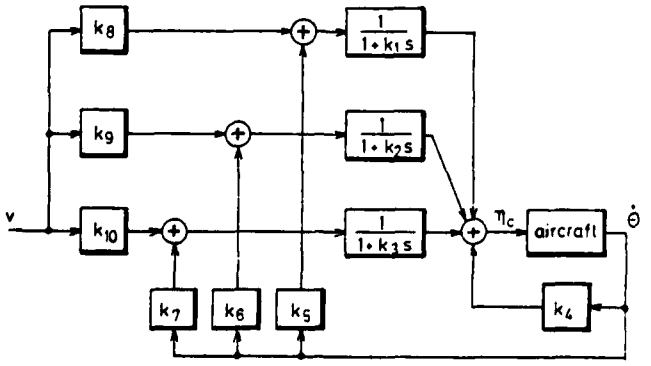


Figure 1:
Compensator structure (3^{rd} order)
Parameters to be designed $k_1 \ldots k_{10}$

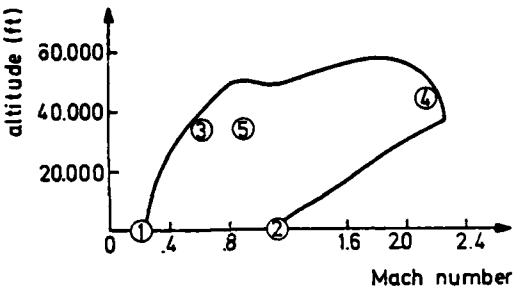
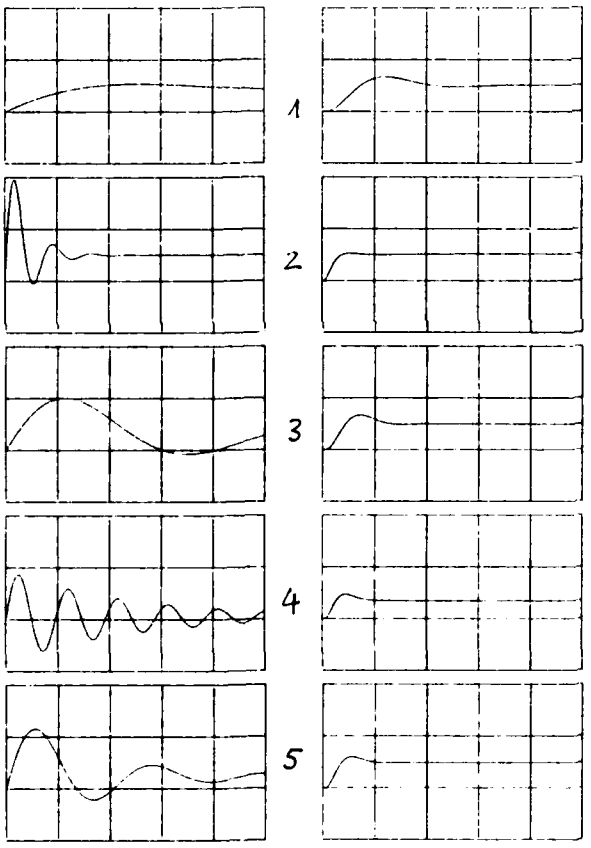
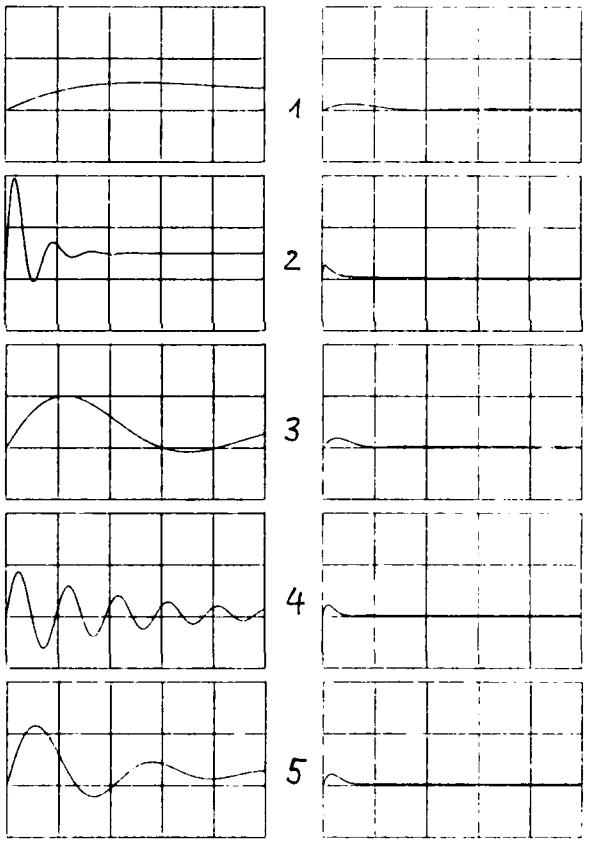


Figure 2: Flight envelope F - 4C
Flight conditions considered 1 ... 5



uncontrolled controlled

Figure 3: Command step response $\dot{\theta}$
Flight conditions 1 ... 5



uncontrolled controlled

Figure 4: Disturbance step response $\dot{\theta}$
Flight conditions 1 ... 5

Literature

[1] Kreisselmeier, G., Steinhauser, R. — Systematic Control Design by Optimizing a Vector Performance Index.
IFAC Symposium on Computer Aided Design of Control Systems, Zürich (Switzerland), Aug. 1979, pp. 113-117.

[2] Powell, M.J.D. — An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives.
Computer Journal 7 (1964), pp. 155-162.

[3a] Steinhauser, R. — Systematische Auslegung von Reglern durch Optimieren eines vektoriellen Gütekriteriums - Durchführung des Reglerentwurfes mit dem Programm REMVG.
DFVLR-Institut für Dynamik der Flugsysteme, Report No. DFVLR-Mitt. 80-18, 1980.

[3b] Henningsen, H.P. — User's Guide to REMVG.
DFVLR-Institute for Flight Systems Dynamics, Internal Report No. IB 515-83/7, 1983.

[4] Kreisselmeier, G., Steinhauser, R. — Application of Vector Performance Optimization to a Robust Control Loop Design for a Fighter Aircraft.
Int. J. Control, 1983, vol. 37, No. 2, 251-284.

PRACTICAL ASPECTS ON DIGITAL IMPLEMENTATION OF CONTROL LAWS

K.J. Aström

Department of Automatic Control
Lund Institute of Technology
Box 725, 220 07 Lund 7
Sweden

SUMMARY

Practical problems associated with digital computer implementation control laws are discussed. The key problem is to convert a digital control law in state space or polynomial form into a computer program which gives the desired results. The paper covers: sensor and actuator interfaces, analog prefiltering, actuator saturation, anti-windup, numerics and coding.

1. INTRODUCTION

This paper deals with practical aspects on implementation of digital control laws. The starting point is a description of a control algorithm in terms of a linear dynamical system either in state space form or in transfer function form. A summary of these form is given in Section 2. Analog prefiltering is a necessity when realising digital control laws. This is discussed in Section 3. The consequences of the dynamics of the prefilters and of the computational delay is also covered in this Section. Although many control laws can be designed using linear theory it is necessary to take nonlinearities into account in the implementation. The special case of actuator saturation which is very common is discussed in Section 4. Consequences of roundoff and finite word-length in the calculations are discussed in Section 6. A more detailed treatment of the topics of this paper is given in [1].

2. DIFFERENT REPRESENTATIONS OF THE REGULATOR

Linear design methods give control laws in the form of linear dynamical system. Such systems can be represented in many different ways.

State feedback with an explicit observer

Pole placement or LQG design result in a control law of the form

$$\begin{cases} \hat{x}(k|k) = \hat{x}(k|k-1) + K[y(k) - \hat{y}(k|k-1)] \\ u(k) = L[x_m(k) - \hat{x}(k|k)] + D_c u_c(k) \\ \hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \\ x_m(k+1) = f(x_m(k), u_c(k)). \\ \hat{y}(k+1|k) = C\hat{x}(k+1|k) \end{cases} \tag{1}$$

where $\hat{x}$ is an estimate of the process state and x_m is the state of the model which generates the desired response to command signals u_c. Notice that a nonlinear model for the desired state may be used in this representation.

General State Representation

If the function f in (1) is linear the regulator given by (1) is a linear system with the inputs y and u_c and the output u. Such a regulator may always be represented as

$$\begin{cases} u(k) = Cx(k) + Dy(k) + D_c u_c(k) \\ x(k+1) = Fx(k) + Gu(k) + G_c u_c(k). \end{cases} \tag{2}$$

This form is more compact than (1). The state may, however, not necessarily have a simple physical interpretation.

Transfer_Function_models

Several design methods result in a description of the regulator in terms of a linear transfer function model. The general form of such a model can be written as

$$R(q)\,u(k) = T(q)\,u_c(k) - S(q)\,y(k), \qquad (3)$$

where R(q), S(q) and T(q) are polynomials in the forward shift operator q.

There are simple transformations between the different representations of the regulators.

How_to_implement_a_discrete_time_system

The implementation of a discrete time system described by (2.1), (2.2) or (2.3) using a digital computer is straightforward. The computer code for implementation of the regulator given by equation (2.2) is:

```
     Procedure Regulate
     begin
1      Adin y  uc
2      u := C*x + D*y + Dc * uc
3      x := F*x + G*y + Gc * uc
4      Daout u
     end
```

Analog to digital conversion is commanded in the first line. The appropriate values are stored in the arrays y and uc. The control signal u is computed in the second line using matrix vector multiplication and vector addition. The state vector x is updated in the third line, and the digital to analog conversion is performed on line four. To obtain a complete code it is also necessary to have type declarations for the vectors u, uc, x and y and the matrices F, G, Gc, C, D and Dc. It is also necessary to assign values to the matrices and the initial value for the state x. When using computer languages which do not have matrix operations it is necessary to write appropriate procedures for generating matrix operations using operations on scalars.

The details depend on the hardware and software available. To show the principles it is assumed that the system described by (2) should be implemented using a digital computer with A-D and D-A converters and a real time clock. The execution of the program is controlled by the clock. which initiates the execution of the code at each clock interrupt. The sampling period is thus determined by the time between the clock pulses.

It is thus straight forward to implement a digital control law. To obtain a good control system it is however necessary to also consider: numerics, sensors, actuators, operational aspects and programming aspects. These will be discussed in the following sections.

3. PREFILTERING AND COMPUTATIONAL DELAY

To obtain a satisfactory digital system it is necessary to filter the analog signals before they are sampled. It is also necessary to consider the dynamics caused by the prefilter and the computational delay.

Analog_prefiltering

To avoid aliasing it is necessary to use an analog prefilter for elimination of disturbances with frequencies higher than the Nyquist frequency associated with the sampling rate. In signal processing applications the analog prefilter is determined frequency content of the signal, see [2] and [3]. In a control problem there is normally much more information available about the signals in terms of differential equations for the process models and possibly also for the disturbances. An analog Kalman filter would be a very good prefilter, because it can be based on a detailed description of the signal. There are several advantages in implementing the Kalman filter in a computer. In such a case it is useful to sample the analog signals at a comparatively high rate and to avoid aliasing by an ordinary analog prefilter designed from the signal processing point of view.

The bandwidth ω_B of the prefilter is inversely proportional to the sampling period h. A common rule of thumb is to choose the sampling period so that $\omega_B h \approx 0.5 - 1$.

The precise choice depends on the order of the filter and on the character of the measured signal. The dynamics of the prefilter should be taken into account when designing the system.

If the sampling rate is changed the prefilter must also be changed. With reasonable component values it is possible to construct analog prefilters for sampling periods shorter than a few seconds. For slower sampling rates it is often simpler to sample once per second or faster with an appropriate analog prefilter and apply digital filtering to the sampled signal. This approach also makes it possible to change the sampling period of the control calculations by software only.

Since the analog prefilter has dynamics it is necessary to include the filter dynamics in the process model. If the prefilter or the sampling rate is changed it is necessary to recompute the coefficients of the control law.

Crude estimates indicate that with normal sampling rates, like 10-20 times per period, it is indeed necessary to consider the prefilter dynamics.

Computational delay

Since A-D and D-A conversions and computations take time, there will always be a delay when a control law is implemented using a computer. The delay, which is called the computational delay, will depend on how the control algorithm is implemented. There are basically two different ways to do this. The measured variables read at time t_k may be used to compute the control signal to be applied at time t_{k+1}. This is called case A. Another possibility, case B, is to read the measured variables at time t_k and to make the D-A conversion as soon as possible.

The first scheme has the disadvantage that the control actions are delayed unnecessarily and second scheme has the disadvantage that the delay will be variable depending upon the programming. In both cases it is necessary to take the computational delay into account when computing the control law. This is easily done by including a time delay of h or τ respectively in the process model. Another practical detail is that there is a good rule to read the inputs before the outputs are set out. If this is not done there is always the risk of electrical cross coupling.

The computational delay can be made as small as possible by making as few operations as possible between the A-D and D-A conversions.

Consider the previously given program. Since the control signal u is available after executing the second line of code the D-A conversion can be done before the state is updated. The delay may be reduced further by also calculating the product C*x after the D-A conversion. The following algorithm is then obtained.

```
     Procedure Regulate
     begin
1      Adin  y  uc
2      u := u1 + D*y + Dc*uc
3      Daout u
4      x := F*x + G*y + Gc*uc
5      u1 := C*x
     end
```

It is useful to have good estimates of computing times for different control algorithms. A good way to obtain these is to run test programs. For linear control laws it is often possible to estimate times from results of a scalar product computation.

On simple microcomputers, which do not have floating point arithmetic in hardware, there will be a substantial difference in computing time between fixed point and floating point operations. The difference is much less if there is hardware for floating point operations.

To judge the consequences of computational delays it is also useful to know the sensitivity of the closed loop system with respect to a time delay. This may be evaluated from a root locus with respect to a time delay. A simpler way is to evaluate how much the closed loop poles change when a time delay of one sampling period is introduced.

Detection of outliers and measurement malfunctions

Linear filtering theory is very useful to reduce the influence of measurement noise. There may, however, also be other types of errors like instrument malfunctions and conversion errors. These are typically characterized by large deviations which occur with low probabilities. It is, of course, very important to try to eliminate such errors so that large errors do not enter into the control law calculations. There are many good ways to achieve this when using computer control.

The errors may be detected at the source. In systems with high reliability requirements this is done by duplication of the sensors. Two sensors are then combined with a simple logic, which gives an alarm if the difference between the sensor signals is larger than a threshold. A pair of redundant sensors may be regarded as one sensor, which either gives a reliable measurement or a signal that it does not work.

In more extreme cases three sensors may be used. A measurement is then accepted as long as two out of the three sensors agree (two-out-of-three logic). It is of course also possible to use even more elaborate combinations of sensors and filters.

It is also possible to use a Kalman filter for error detection. Consider for example the control algorithm (1) with an explicit observer. The one step prediction error

$$\varepsilon(k) = y(k) - \hat{y}(k|k-1) = y(k) - C\hat{x}(k|k-1)$$

appears explicitly in the algorithm. If estimates of the covariance matrix of the prediction error are available it is easy to test if a particular measurement is reasonable, see [4].

One possibility to obtain the error covariance is to update the covariance equation of the Kalman filter on line.

Kalman filters and redundant sensors pairs may also be combined. If measurement errors are checked in this way it is possible to obtain a very flexible system. The scheme should be augmented with tests to ensure observability. It is thus possible to obtain a system which can provide diagnosis of sensor errors.

Notice that the possibilities of making these types of test depend crucially on the fact that the representation of the control law (1) with an explicit observer is used.

In computer control there are also may other possibilities to detect different types of hardware and software errors. A few extra channels in the A-D converter, which are connected to fixed voltages, may be used for testing and calibration. By connecting a D-A channel to an A-D channel the D-A converter may also be tested and calibrated. The computer may be checked by performing calculations calculations whose results are known and compare the results with the known values.

4. NONLINEAR ACTUATORS

Although linear theory has a wide applicability there are often some nonlinearities which must be taken into account. Actuators often have a saturation characteristics. This nonlinearity may be important when large changes are made. There may be difficulties with the control system during start up and shut down as well as during large changes if the nonlinearities are not considered.

The rational way to deal with the saturation is to develop a design theory which takes the nonlinearity into account. This can be done using optimal control theory. Such a design method is, however, quite complicated. The corresponding control law is also complex. It is therefore practical to use simple heuristic methods.

The reason for the difficulties is that the regulator is a dynamical system. When the control variable saturates it is necessary to make sure that the state of the regulator behaves properly. Different ways of achieving this are discussed below.

State space regulators with an explicit observer

Consider first the case when the control law is described as an observer combined with a state feedback (1). The regulator is thus a dynamical system whose state is represented by the estimated state $\hat{x}$ in (1). In this case it is straightforward to see how the difficulties with the saturation may be avoided.

The estimator (1) will give the correct estimate if the variable u in (1) is chosen as the actual control variable u_p. If the variable u_p is measured the estimate given by (1) and the state of the regulator are thus correct even if the control variable saturates. If the actuator output is not measured it can be estimated provided that the nonlinear characteristics is known. For the case of a simple saturation the control law can thus be written as

$$\begin{cases} \hat{x}(k|k-1) = \hat{x}(k|k-1)+K[y(k)-C\hat{x}(k|k-1)] = [A - KC]\,\hat{x}(k-1|k-1) + B\hat{u}(k-1) \\ \hat{u}_p(k) = \text{sat}\{L[x_m(k) - \hat{x}(k|k)] + u_m\} \\ \hat{x}(k+1|k) = A\hat{x}(k|k) + B\hat{u}(k) \\ \hat{x}(k+1|k) = A\hat{x}(k|k) + B\hat{u}_p(k) \end{cases} \qquad (4)$$

$$\text{sat } u = \begin{cases} u_{low} & u \le u_{low} \\ u & u_{low} < u < u_{high} \\ u_{high} & u \ge u_{high} \end{cases}$$

for a scalar and

$$\text{sat } u = \begin{bmatrix} \text{sat } u_1 \\ \text{sat } u_2 \\ . \\ . \\ . \\ \text{sat } u_n \end{bmatrix}$$

for a vector. The values ulow and uhigh are chosen to correspond to the actuator limitations. Notice that even if the transfer function from y to u for (1) is unstable the state of the system (4) will always be bounded if the matrix A-KC is stable. It is also clear that $\hat{x}$ will be a good estimate of the process state even if the value saturates provided that ulow and uhigh are chosen properly.

The general state space model

The regulator may also be specified as a state space model of the form (2)

$$x(k+1) = F\,x(k) + G\,y(k) \tag{5}$$

$$u(k) = C\,x(k) + D\,y(k) \tag{6}$$

which does not include an explicit observer. The command signals have been neglected for simplicity. If the matrix F has eigenvalues outside the unit disc and the control variable saturates it is clear that windup may occur. Assume for example that the output is at its limit and there is a control error y. The state and the control signal will then continue to grow although the influence on the process is restricted because of the saturation.

To avoid the difficulty it is desirable to make sure that the state of (5) assumes the proper value when the control variable saturates. In conventional process controllers this is accomplished by introducing a special tracking mode which makes sure that the state of the system corresponds to the input output sequence $\{u_p(k), y(k)\}$. The design of a tracking mode may be formulated as an observer problem. In the case of state feedback with an explicit observer the tracking is done automatically by providing the observer with the actuator output u_p or its estimate $\hat{u}_p$. In the regulator given by (5) and (6) there is no explicit observer. To get a regulator which avoids the windup problem the solution for the regulator with an explicit observer will be imitated. The control law is first rewritten as indicated in Fig. 1. The systems in a) and b) have the same input-output relation. The system S_B is also stable. By introducing a saturation in the feedback loop in b) the state of the system S_B is always bounded if y and u are bounded.

This argument may formally be expressed as follows. Multiply (6) by K and add to (5). This gives

$$\begin{aligned} x(k+1) &= F\,x(k) + G\,y(k) + K[u(k) - C\,x(k) - D\,y(k)] \\ &= [F-KC]\,x(k) + [G-KD]y(k) + K\,u(k) \\ &= F_0 x(k) + G_0 y(k) + K\,u(k). \end{aligned}$$

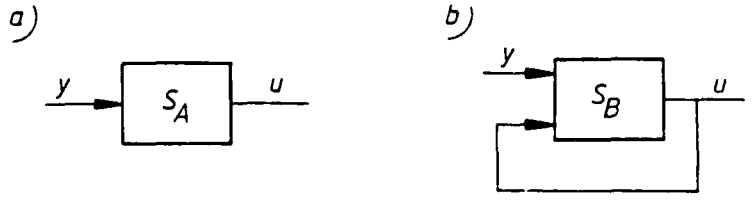


Fig. 1 Different representations of the control law.

If the system (5), (6) is observable the matrix K can always be chosen so that $F_0 = F - KC$ has prescribed eigenvalues inside the unit disc. Notice that this equation is analogous to (4). Applying the same arguments as for the regulator with an explicit observer the control law becomes

$$\begin{aligned} x(k+1) &= F_0 x(k) + G_0 y(k) + K u(k) \\ u(k) &= \text{sat } [Cx(k) + D y(k)]. \end{aligned} \tag{7}$$

The saturation function is chosen to correspond to the actual saturation in the actuator. A comparison with the case of an explicit observer shows that (7) corresponds to an observer with dynamics given by the matrix F_0. The system (7) is of course also equivalent to (2) for small signals.

Transfer Function Form

The corresponding constructions can also be carried out for regulators characterized by input-output models. Consider a regulator described by

$$R(q)\, u(k) = T(q)\, u_c(k) - S(q)\, y(k) \tag{8}$$

where R, S and T are polynomials in the shift operator. The problem is to rewrite the equation so that it looks like a dynamical system with the observer dynamics driven by three inputs, the command signal u_c, the process output y and the control signal u. This is accomplished as follows.

Let $A_o(q)$ be the desired characteristic polynomial of the observer. Adding $A_o(q)u(k)$ to both sides of (8) gives

$$A_o u = T u_c - S y + (A_o - R)\, u$$

A regulator with anti-windup compensation is then given by

$$\begin{cases} A_o v = T u_c - S y + (A_o - R)\, u \\ u = \text{sat } v. \end{cases} \tag{9}$$

This regulator is equivalent to (8) when it does not saturate. When the control variable saturates it can be interpreted as an observer with dynamics given by the polynomial A_o.

A particularly simple case is the case of a dead beat observer i.e. $A_0^* = 1$. The model can then be written as

$$u(k) = \text{sat } [T^*(q^{-1})\, u_c(k) - S^*(q^{-1})\, y(k) + (1 - R^*(q^{-1}))\, u(k)] \tag{10}$$

5. NUMERICS

When implementing a computer control system it is necessary to answer questions like: How accurate converters are needed? What precision is required in the computations? Should computations be made in fixed point or floating point arithmetic? To answer these questions it is necessary to understand the effects of the limitations and to estimate their consequences for the closed loop system. This is not a trivial question, because the result will depend on a complex interaction of the feedback, the algorithm and the sampling rate. The real issues fortunately involves crude questions like 10 or 12 bit resolution, 24 or 32 bit wordlength. Such questions may be answered using simplified analysis. A detailed treatment is given in [5].

Error sources

The major error sources are

- Quantization in A-D converters.
- Quantization of parameters.
- Round-off, overflow, and underflow in addition, subtraction, multiplication, division, function evaluation and other operations.
- Quantization in D-A converters.

Common types of A-D converters have accuracies of 8, 10, 12 and 14 bits which corresponds to a resolution of 0.4 %, 0.1 %, 0.025 % and 0.006 %. The percentages are in relation to full scale. The D-A converters have also a limited precision. An accuracy of 10 bits is typical. The error due to the quantization of the parameters will depend critically on the sampling period and on the chosen realization of the control law.

Word-length

Digital control algorithms are typically implemented on micro and minicomputers which have word-lengths of 8, 16 or 32 bits. Special purpose computers where the word-length may be chosen freely are used in applications like the space shuttle or in special products which are made in very large quantities.

There are many differences in number representations. The following representations are common.

- Fixed point single precision 16 bit
- Fixed point double precision 32 bit
- Floating point single precision 8 bit exponent 24 bit mantissa
- Floating point single precision 8 bit exponent 56 bit mantissa

A key problem is that floating point operations are neither associative nor distributive.

Overview of effects of round-off and quantization

An overview of the effects of round-off and quantization will now be given. Tools for analysing the effects will also be discussed.

The consequences of round-off and quantization depend on the feedback system and on the details of the algorithm. The properties may be influenced considerably by changing the representation of the control law or the details of the algorithm. It is thus important to understand the phenomena.

A detailed description of round-off and quantization leads to a complicated nonlinear model which is very difficult to analyse. Investigation of very simple cases shows, however, that quantization and round-off may lead to limit cycle oscillations, see [6] and [7].

Some properties of quantization and round-off in a feedback system may also be captured by linear analysis. Quantization and round-off are then modeled as ideal operations with additive or multiplicative disturbances. The disturbance may be either deterministic or stochastic. This type of analysis is particularly useful for order of magnitude estimation. It allows investigation of complex systems and it is useful when comparing different algorithms, see [8] and [9].

Techniques from sensitivity analysis and numerical analysis are also useful to find the sensitivity of algorithms to changes of parameters. Such methods may be used to compare and screen different algorithms. The methods are, however, limited to comparison of the open-loop performances of the algorithms. It is of course also necessary to compare the effects of quantization and round-off with the other disturbances in the system.

Different realizations

A control law is a dynamical system. Different realizations may be obtained by transforming the state space coordinates. The choice of a suitable realization is very important for the conditioning. In particular the companion forms are very bad from a numerical point of view, see [10]. It is much better to represent a system as a combination of first and second order systems.

If the dynamical system representing the regulator has nr distinct real poles and nc complex pole pairs the control algorithm may be transformed to the model form

$$
\begin{cases}
z_i(k+1) = \lambda_i z_i(k) + \beta_i y(k) \quad i = 1,\ldots,nr \\
v_i(k+1) = \begin{bmatrix} \sigma_i & \omega_i \\ -\omega_i & \sigma_i \end{bmatrix} v_i(k) + \begin{bmatrix} \gamma_{i1} \\ \gamma_{i2} \end{bmatrix} y(k) \quad i = 1,\ldots,nc \\
u(k) = Dy(k) + \sum_{i=1}^{nr} \gamma_i z_i(k) + \sum_{i=1}^{nc} \delta_i^T v_i(k)
\end{cases}
\tag{11}
$$

where the complex poles are represented using real variables. Notice that z_i are scalars and v_i are vectors with two elements.

To avoid numerical difficulties the control law should thus be transformed into the form (6.4) which is then implemented in the control computer. The transformation may easily be done in a package for computer aided design. Notice that it is easy to use fixed point calculations and scaling for equations in the form (6.4).

If the control law has multiple eigenvalues a Jordan canonical form replaces (6.4). An eigenvalue λ of multiplicity 3 thus corresponds to a block

$$z(k+1) = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix} z(k) + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} y(k).$$

Effects of the sampling period

The sampling period also has a considerable influence on the conditioning as is shown by the following examples.

EXAMPLE - Effect of sampling period on coefficient precision

Consider a first order system with time constant T. The discrete time equivalent of such a system is

$$x(t+h) = ax(t) + bu(t),$$

where

$$a = e^{-h/T}.$$

Simple calculations show that

$$\frac{dT}{T} = -\frac{T}{h}\frac{da}{a}.$$

For a given relative precision in the equivalent time constant is thus inverse proportional to the sampling period.

6. CONCLUSIONS

Implementation of control laws using a computer have been discussed in this paper. The key problem is to implement a discrete time system. The principles for doing this have been covered in detail. It was shown that it is straightforward to obtain the code from the control algorithm. The importance of prefiltering to avoid aliasing has been mentioned. Nonlinear digital filtering for removing outliers has also been discussed. It has been mentioned that the computational delay is influenced considerably by the organization of the computer code. Difficulties which arise from saturation in actuators and ways to avoid the difficulties have been discussed. This will also automatically give a solution to mode switching and initialization. Numerical problems and consequences of finite word-length have also been discussed. It was found to be very beneficial to transform the equations describing the control law to a form which is numerically well conditioned. Although the presentation is kept fairly brief the information given should be sufficient to implement control algorithms on mini and micro computers using high level languages.

7. REFERENCES

[1] K.J. Aström and B. Wittenmark: Computer-controlled Systems, Prentice Hall, Englewood Cliffs, N.J, 1984.

[2] F.F. Kuo: Network Analysis and Synthesis, Wiley, New York, 1962.

[3] A.B. Williams: Electronic Filter Design Handbook, McGraw Hill, 1981.

[4] A. Willsky: A survey of design methods for failure detection in dynamic systems. Automatica 12 (1976) 601-611.

[5] P. Moroney: Issues in the Implementation of Digital Feedback compensators. MIT Press, 1983.

[6] S.R. Parker and S.F. Hess: Limit cycle oscillations in digital filters. IEEE Trans. Circuit Theory CT-18 (1971) 687-697.

[7] L.B. Jackson: Limit cycles in state-space structures for digital filters. IEEE Trans. Circuits & Systems CAS-26 (1979) 67-68.

[8] A.V. Oppenheim and R.W. Schafer: Digital Signal Processing. Prentice Hall, New Jersey, 1975.

[9] L.R. Rabiner and B. Gold: Theory and Application of Digital Signal Processing. Prentice Hall, New Jersey, 1975.

[10] G. Björk, A. Dahlqvist and N. Andersson: Numerical Methods. Prentice Hall, Englewood Cliffs, 1974.

BIBLIOGRAPHY

This Bibliography with Abstracts has been prepared to support AGARD Lecture Series No.128 by the Scientific and Technical Information Branch of the US National Aeronautics and Space Administration Washington, D.C., in consultation with the Lecture Series director Dr J.E.Wall.

UTTL: Control science and technology for the progress of society; Proceedings of the Eighth Triennial World Congress, Kyoto, Japan, August 24-28, 1981. Volume 4. Part A - Mechanical systems and robots. Part B - Aerospace and transportation

AUTH: A/AKASHI, H. PAA: A/(Kyoto University, Kyoto, Japan) Congress sponsored by the International Federation of Automatic Control. Oxford, Pergamon Press, 1982. 685 p. (For individual items see A83-26578 to A83-26610)

ABS: Topics discussed include satellite attitude control using momentum wheels, a sensitivity analysis of a modal controller for flexible space structures, the design of an ion thruster system for satellite position control, the implementation of operational aspects in the European Spacelab System, software for the automatic control of spacecraft instruments, and the development of adaptation and identification algorithms in adaptive digital aircraft control systems. Also examined are altitude transitions in energy climbs, control and design aspects of magnetically suspended vehicles, the dynamics and control of maglev vehicles with parameter uncertainties, feedback laws for robotic systems, a CAD/CAM application for wheelchair analysis, a microprocessor-based power measurement device for control applications, and an electromagnetic damper for the vibration control of a transmission shaft. Several case studies are also presented including the development of an attitude stabilization and control system of a Japanese scientific satellite, the attitude and orbit control system of the MOS-1, the application of redundant processing to the Space Shuttle, and the results of a world survey of computer-aided manufacturing. 82/00/00 83A26577

UTTL: Illustration of the applicability of computer aided design packages

AUTH: A/MUNRO, N. PAA: A/(University of Manchester Institute of Science and Technology, Manchester, England) In: Conference on Decision and Control, 20th, and Symposium on Adaptive Processes, San Diego, CA, December 16-18, 1981. Proceedings. Volume 1. (A83-24701 09-63) New York, Institute of Electrical and Electronics Engineers, 1981, p. 413-419.

ABS: Interactive CAD facilities for the analysis, design, and simulation of control systems are introduced and described briefly. The use of these facilities is illustrated by their application to a selection of industrial control problems previously solved. Problem areas are highlighted and future developments are indicated. 81/00/00 83A24719

UTTL: Evaluation procedures to be used during the development of CAD systems

AUTH: A/CONSTANTINOU, S.; B/LEONARD, R.; C/RATHMILL, K. PAA: A/(Cassandra Trading, Ltd., Cyprus); B/(University of Manchester Institute of Science and Technology, Manchester, England); C/(Cranfield Institute of Technology, Cranfield, Beds., England) Computer-Aided Design, vol. 14, Nov. 1982, p. 321-328.

ABS: This paper describes the different types of Computer-Aided Design (CAD) systems that are available, and enters into a discussion on the choosing of systems. The paper then follows the path of modification, taking an existing system and changing it slightly to meet specific needs. One of the most important parts of this process is the methodology for evaluating existing CAD systems to determine if they could be modified to embrace the particular application. This procedure is described here. The paper concludes with an example in which an engineering company has attempted to speed design and drawing time by using a modified existing drawing package. 82/11/00 83A15146

UTTL: Some DDC system design procedures

AUTH: A/KNOWLES, J. B. In: Design of modern control systems. (A82-42558 21-63) Stevenage, Herts., England and New York, Peter Peregrinus, Ltd., 1982, p. 223-245.

ABS: A simply implemented comprehensive design technique is developed for single-input direct-digital-control (DDC) systems. The technique provides a simple calculation that ensures that the data sampling rate is consistent with the control system's accuracy specification or the fatigue life of its actuators. Pulsed transfer-function design for a plant controller is based on two simple rules and a few standard frequency-response curves, which are easily computed once and for all. Structural resonances are eliminated by digital notch filters, the pole-zero locations of which are directly related to the frequency and bandwidth of an oscillatory mode. In addition, a computationally simple formula gives an upper bound on the amplitude of the control error component due to multiplicative rounding effects in the digital computer. 82/00/00 82A42568

UTTL: Unified display hardware for computer-aided control systems for different purposes

AUTH: A/POMPUSHKIN, V. S. Elektronnoe Modelirovanie, vol. 4, May-June 1982, p. 52-56. In Russian.

ABS: The paper examines the development of unified display hardware for computer-aided control systems for

different purposes. This development is based on the implementation of the module-block principle, which provides for high reliability and simplifies the hardware. Particular attention is given to the use of color CRT displays with memory. 82/06/00 82A34455

UTTL: A frequency matching method for model reduction of digital control systems
AUTH: A/PUJARA, L. R.; B/RATTAN, K. S. PAA: A/(Wilberforce University, Wilberforce, OH); B/(Wright State University, Dayton, OH)
International Journal of Control, vol. 35, Jan. 1982, p. 139-148.
ABS: In this paper, a computer-aided method for reducing the order of a digital control system is developed. This is done by minimizing a weighted mean square error between the frequency responses of the reduced and the original systems. The method is demonstrated by a numerical example and the results are compared with those of the continued fractions method developed by Shih and Wu and the optimization method of model reduction developed by Luus. 82/01/00 82A34356

UTTL: Integrated structural analysis and design support for advanced launch vehicles
AUTH: A/WONG, D. G.; B/BOUSQUET, R. D.; C/FULLER, C. R. PAA: C/(Lockheed Missiles and Space Co., Inc., Sunnyvale, CA) In: Structures, Structural Dynamics and Materials Conference, 23rd, New Orleans, LA, May 10-12, 1982, Collection of Technical Papers. Part 2. (A82-30076 13-39) New York, American Institute of Aeronautics and Astronautics, 1982, p. 145-150.
ABS: The paper discusses a system incorporating the latest developments in computer hardware and software, conceived in accordance with a five year plan initiated to support future advanced launch vehicle programs at the Lockheed Missiles and Space Company. Engineering requirements for the integrated system forming the basis for the department upgrades are presented. Software attributes are discussed, including program modularity, data management, and programming standards. Hardware attributes are also presented, and include system definition and design, communication network links, and various graphical capabilities.
RPT#: AIAA 82-0675 82/00/00 82A30144

UTTL: Optimization-based computer-aided-design of control systems
AUTH: A/POLAK, E. PAA: A/(California, University, Berkeley, CA) In: Joint Automatic Control Conference, Charlottesville, VA, June 17-19, 1981, Proceedings. Volume 1. (A82-13076 03-63) New York, American Institute of Chemical Engineers, 1981, 4 p. (WP-4C).
ABS: The paper describes some of the progress that has been made in developing a new optimization-based computer-aided design methodology for multivariable control system design. The components of this methodology are: (1) techniques for expressing design requirements in the form of semiinfinite optimization problems; (2) new semiinfinite optimization algorithms developed specifically for engineering design; and (3) new ways for constructing interactive computer-aided design packages. 81/00/00 82A13103

UTTL: Digital redesign of existing multiloop continuous control systems
AUTH: A/RATTAN, K. S. PAA: A/(Wright State University, Dayton, OH) In: Joint Automatic Control Conference, Charlottesville, VA, June 17-19, 1981, Proceedings. Volume 1. (A82-13076 03-63) New York, American Institute of Chemical Engineers, 1981, 8 p. (WP-1D).
ABS: A computer-aided method for converting existing multiloop continuous-data control systems into digital control systems is presented. Digital controllers are synthesized by matching the frequency responses of the digital control system to that of the continuous control system with a minimum weighted mean square error. Formulas for computing the parameters of the digital controllers are obtained as a result. An example of digitalizing existing continuous flight controller for the longitudinal YF-16 aircraft is considered and the results obtained are compared with those obtained by the Tustin transform. 81/00/00 82A13093

UTTL: Pareto-optimal multi-objective design of airplane control systems
AUTH: A/SCHY, A. A.; B/JOHNSON, K. G.; C/GIESY, D. P PAA: B/(NASA, Langley Research Center, Hampton, VA); C/(Kentron International, Inc., Hampton, VA) CORP: National Aeronautics and Space Administration, Langley Research Center, Hampton, Va.; Kentron International, Inc., Hampton, Va. In: Joint Automatic Control Conference, San Francisco, CA, August 13-15, 1980, Proceedings. Volume 1. (A81-45502 21-63) New York, Institute of Electrical and Electronics Engineers, Inc., 1980, 8 p. (WP1-A).

ABS: A constrained minimization algorithm for the computer aided design of airplane control systems to meet many requirements over a set of flight conditions is generalized using the concept of Pareto-optimization. The new algorithm yields solutions on the boundary of the achievable domain in objective space in a single run, whereas the older method required a sequence of runs to approximate such a limiting solution. However, Pareto-optimality does not guarantee a satisfactory design, since such solutions may emphasize some objectives at the expense of others. The designer must still interact with the program to obtain a well-balanced set of objectives. Using the example of a fighter lateral stability augmentation system (SAS) design over five flight conditions, several effective techniques are developed for obtaining well-balanced Pareto-optimal solutions. For comparison, one of these techniques is also used in a recently developed algorithm of Kreisselmeier and Steinhauser, which replaces the hard constraints with soft constraints, using a special penalty function. It is shown that comparable results can be obtained. 80/00/00 81A45518

UTTL: Selecting and implementing a turnkey graphics system

AUTH: A/BLISS, F. W.; B/HYMAN, G. M. PAA: B/(Ford Motor Co., Computer Graphics Dept., Dearborn, MI) IEEE Computer Graphics and Applications, vol. 1, Apr. 1981, p. 55-62, 64, 70.

ABS: The selection and implementation process for a turnkey graphics system is shown to comprise: (1) initial preparation, involving the selection of an examination team and the development of a list of desired features; (2) the testing of the system's graphic capabilities and their evaluation and benchmarking; (3) the assessment of the computer system as a whole, including such factors as programming facilities, documentation and maintenance; (4) system acceptance through staff and site preparation; (5) the training of operators; and (6) system management through job tracking and the refinement of operating procedures. Overviews are provided of such proprietary turnkey CAD systems as the Intergraph IGDS, Applicon IMAGE, Redac PCB, Calma CADEC, Auto-Trol GS-1000, and Gerber IDS-80. 81/04/00 81A39279

UTTL: Robust decentralised control of a servomechanism problem for a class of nonlinear systems

AUTH: A/DORAISWAMI, R. PAA: A/(Santa Catarina, Universidade Federal, Florianopolis, Brazil) IEE Proceedings, Part D - Control Theory and Applications, vol. 128, pt. D, no. 2, Mar. 1981, p. 33-40. Research supported by the Universidade Federal de Santa Catarina, Financiadora de Estudos e Projetos, Conselho Nacional de Desenvolvimento Cientifico et Tecnologico, and Coordenacao do Aperfeicoamento do Pessoal de Ensino Superior.

ABS: Robust decentralised control of a class of nonlinear system so as to track a given reference input signal and reject a known class of disturbance signals is proposed. The controller consists of tracking-error-driven servocompensators and local stabilisers. The servocompensators contain the modes of not only the external signals but also the internal signals. The local stabiliser must be designed to yield large damping and low overshoot. A computer-aided-design procedure is given. 81/03/00 81A30214

UTTL: Some unifying concepts in multivariable feedback design

AUTH: A/OWENS, D. H. PAA: A/(Sheffield, University, Sheffield, England) International Journal of Control, vol. 33, Apr. 1981, p. 701-711.

ABS: Frequency domain techniques for computer-aided design of multivariable feedback systems are now well established in the form of several, apparently distinct, design techniques. It is shown that a unified design structure can be developed based on the theoretical concepts of precompensation, eigenvalue approximation and permissible, constant, input/output transformations. 81/04/00 81A28990

UTTL: Design of digital feedforward/preview controllers for processes with predetermined feedback controllers

AUTH: A/TOMIZUKA, M.; B/FUNG, D. H. PAA: A/(California, University, Berkeley, Calif.) (American Society of Mechanical Engineers, Winter Annual Meeting, Chicago, Ill., Nov. 16-21, 1980,) ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control, vol. 102, Dec. 1980, p. 218-225.

ABS: This paper deals with the design of digital feedforward/preview (F/P) controllers for multi-input, multi-output processes with predetermined feedback controllers. The feedback controller is assumed to include an integral action which operates upon the error between setpoint and process output. The

feedforward/preview controller is a weighted sum of previewed values of future disturbances. Formula for the F/P control gains are derived using a parameter optimization technique to minimize a quadratic cost criterion. Simulation examples are included to show some of the consequences of the developed design method

RPT#: ASME PAPER 80-WA/DSC-2 80/12/00 81A21165

UTTL: Parameter space design of robust control systems

AUTH: A/ACKERMANN, J. PAA: A/(Deutsche Forschungs- und Versuchsanstalt fuer Luft- und Raumfahrt, Institut fuer Dynamik der Flugsysteme, Oberpfaffenhofen, West Germany) IEEE Transactions on Automatic Control, vol. AC-25, Dec. 1980, p. 1058-1072. Research supported by the Deutsche Forschungs- und Versuchsanstalt fuer Luft- und Raumfahrt;

ABS: A state or output feedback with fixed gains was determined such that 'nice' stability defined by a region in the eigenvalue plane is robust with respect to large plant parameter variations, sensor failures, and quantization effects of the controller. Pole placement is formulated as an affine map from the parameter space of coefficients of the characteristic polynomial coefficients to the parameter space of state feedback gains. This method is illustrated by the design of a crane control system; the approach can be applied to graphical computer design of robust systems, algebraic robustness conditions, and algorithms for iterative design of robust control systems. 80/12/00 81A21027

UTTL: Local design of optimal control for digital systems

AUTH: A/DANILINA, E. V.; B/RASINA, I. V.; C/KHRUSTALEV, M. M. (Avtomatika i Telemekhanika, May 1980, p. 48-56.) Automation and Remote Control, vol. 41, no. 5, Oct. 10, 1980, pt. 1, p. 627-634. Translation.

ABS: A method of designing a feedback control is proposed that is near-optimal in the vicinity of a program which satisfies sufficient conditions for a relative minimum. A regular algorithm for computing the vicinity is described, and a technique for estimating the functional-wide derivation of the designed control from the strict optimum is presented. 80/10/10 81A15802

UTTL: Modern controls and the hybrid computer revisited

AUTH: A/TISDALE, H. PAA: A/(Electronic Associates, Inc., West Long Beach, N.J.) In: Summer Computer Simulation Conference, Newport Beach, Calif., July 24-26, 1978, Proceedings. (A80-49826 22-66) Montvale, N.J., AFIPS Press, 1978, p. 257-264.

ABS: The hybrid (digital + analog) computer concept is discussed, along with its application in the modern controls field. Modern optimal control theory is reviewed, and it is shown that the hybrid computer can be used in the fields of active controls, fault tolerant flight control systems, digital flight control systems, and all-weather operations. 78/00/00 80A49831

UTTL: CAD/CAM applications using the computervision design graphics system

AUTH: A/OMAN, B. H.; B/VELAZQUEZ, R. J. PAA: B/(General Dynamics Corp., Convair Div., San Diego, Calif.) Society of Allied Weight Engineers, Annual Conference, 38th, New York, N.Y., May 7-9, 1979, 29 p.

ABS: The paper presents an introduction to graphics systems relative to CAD/CAM applications. Attention is given to the role of the design graphics system within the overall design process with specific discussions on the Computervision Design Graphics System. Discussion covers the system's hardware and software capabilities in detail, as well as the current configuration and utilization. Results are surveyed which address design and drawing application for avionics printed circuit boards and electronic packaging and structural/mechanical applications for beams, frames, bulkheads, and mechanics. Finally, a typical manual versus computer aided cost savings is also included.

RPT#: SAWE PAPER 1287 79/05/00 80A20633

UTTL: Digital-computer design of control systems - Spectral and interpolation methods

AUTH: A/SOLODOVNIKOV, V. V.; B/SEMENOV, V. V.; C/PESCHEL, M.; D/NEDO, D. Moscow, Izdatel'stvo Mashinostroenie; Berlin, Verlag Technik, 1979, 664 p. In Russian.

ABS: Spectral and interpolation methods of dynamic analysis of automatic control systems using digital computers are presented. Theoretical principles of the spectral method are considered for the problems of linear analog, discrete, and analog-discrete systems of different classes: stationary, nonstationary, stochastic, and multidimensional with determinate and random actions. Theoretical principles of the interpolation method are given for both linear and

nonlinear analog systems of stationary and nonstationary classes. Software for both methods is described along with examples of several designs. 79/00/00 80A14025

UTTL: Computer-aided design of sampled-data control systems via complex-curve fitting
AUTH: A/RATTAN, K. S. PAA: A/(Wright State University, Dayton, Ohio) In: SOUTHEASTCON '79; Proceedings of the Region 3 Conference and Exhibit, Roanoke, Va., April 1-4, 1979. (A79-50026 22-31) New York, Institute of Electrical and Electronics Engineers, Inc., 1979, p. 63-66.
ABS: A method for the design of sampled-data control systems by means of a digital controller is presented. The synthesis of the digital controller is carried out by matching the frequency response of the sampled-data control system to that of the continuous model with a minimum weighted mean-square error. The transfer function of the continuous model is obtained from the given performance specifications. The design technique is illustrated with a numerical example. 79/00/00 79A50034

UTTL: Sequential design of linear multivariable systems
AUTH: A/MAYNE, D. Q. PAA: A/(Imperial College of Science and Technology, London, England) Institution of Electrical Engineers, Proceedings, vol. 126, June 1979, p. 568-572.
ABS: The paper describes and critically assesses, in the light of recent advances, the sequential return difference method for the computer-aided design of linear multivariable control systems. In this method, a sequence of single-loop designs, using classical procedures, yields a multivariable design satisfying various criteria such as stability, disturbance attenuation, low interaction and integrity. 79/06/00 79A41180

UTTL: Frequency response methods in the design of multivariable non linear feedback systems
AUTH: A/GRAY, J. O.; B/TAYLOR, P. M. PAA: A/(University of Manchester Institute of Science and Technology, Manchester, England); B/(Marconi, Ltd., Chelmsford, Essex, England) In: Multivariable technological systems: International Symposium, 4th, Fredericton, New Brunswick, Canada, July 4-8, 1977, Preprints. (A78-16301 04-63) Oxford and New York, Pergamon Press, 1977, p. 225-232.
ABS: The application of frequency domain methods to the study of a class of multivariable nonlinear feedback systems is considered and a computational procedure is devised for the determination of limit cycle operation which emphasizes the effects of off diagonal system elements. It is shown how compensation can be applied using classical frequency domain techniques. Several examples of use are given. 77/00/00 78A16315

UTTL: A computer aided design method of multivariable systems using output feedback
AUTH: A/BAYOUMI, M. M.; B/DUFFIELD, T. L. PAA: B/(Queen's University, Kingston, Ontario, Canada) In: Multivariable technological systems: International Symposium, 4th, Fredericton, New Brunswick, Canada, July 4-8, 1977, Preprints. (A78-16301 04-63) Oxford and New York, Pergamon Press, 1977, p. 119-125. National Research Council of Canada
ABS: A computer program for implementing the synthesis of output feedback decoupled multivariable systems is described. Attention is given to the problem of the joint stability of the closed loop decoupled system and the feedback subsystem. The calculations which must be carried out by the computer are described in a step-by-step manner, and specific details of the program are given. 77/00/00 78A16306

UTTL: Development of an interactive computer aided design program for digital and continuous control system analysis and synthesis
AUTH: A/LOGAN, G. T. CORP: Air Force Inst. of Tech., Wright-Patterson AFB, Ohio. CSS: (School of Engineering.)
ABS: This report details the development of an interactive computer aided design package for digital and continuous control system analysis and synthesis. The baseline from which the new package is developed is an existing computer aided design package originally developed and hosted on a Control Data Corporation CYBER computer A detailed analysis of the existing computer aided design package is provided. Structured analysis, structured design, and top-down design techniques are employed to design a modular program structure consisting of an executive, interactive user interface, processor (control system algorithms), and an output driver. An incremental top-down technique along with structured programming is used to implement the top-level structure of the new computer aided design package and a portion of the interactive user interface. The report also describes modifications to

the existing package required to rehost it on a Digital Equipment Corporation VAX-11/780 computer in a virtual memory environment.
RPT#: AD-A118042 AFIT/GE/EE/82M-5 82/03/00 83N11797

UTTL: Aspects of a computer-aided design and finite-element simulation by small computers
AUTH: A/DEWET, F. J. PAA: A/(Rekor, Edms, Bpk., Pretoria)
CORP: Council for Scientific and Industrial Research, Pretoria (South Africa). In its Proc. of the 2nd South African Computer Symp. on Res. in Theory, Software and Hardware 10 p (SEE N83-11748 02-59)
ABS: Computer aided design and finite element analysis developed to same extent along the same lines. The development of desktop computers and micros led to small systems with powerful capabilities and sophisticated graphics. Two important aspects in programming for computer aided design and finite element analysis on desktop computers are the data structure and the graphic manipulations. Most desktop have only BASIC as a high level language but some of the standard functions can be used quite effectively towards saving memory and simplifying graphic manipulations. 81/00/00 83N11764

UTTL: Design and Analysis of a multivariable, digital controller for the A-7D Digitac 2 aircraft and the development of an interactive computer design program
AUTH: A/PORTER, D. S. CORP: Air Force Inst. of Tech., Wright-Patterson AFB, Ohio. CSS: (School of Engineering.)
ABS: A multivariable error-actuated digital tracking controller is developed for the Mach 0.18 flight condition of the A-7D Digitac 2 aircraft using a singular perturbation method. The design is accomplished and simulated after the development of an interactive computer program named MULTI. The complete designed process is presented; a discussion of the robustness of the control law over a range of flight conditions and the effect of an aircraft flight control surface failure is also included. A more accurate aircraft model is required before further testing is accomplished to study the possibilities of other controller designs. The computer package is available to the engineering community.
RPT#: AD-A118134 AFIT/GE/EE/81D-48 81/12/00 83N11144

UTTL: Digital flight control system design using singular perturbation methods
AUTH: A/SMYTH, J. S. CORP: Air Force Inst. of Tech., Wright-Patterson AFB, Ohio. CSS: (School of Engineering.)
ABS: In this report a single longitudinal tracker is developed for the aircraft for three different flight conditions. The method used is the singular perturbation method applied to fast-sampling, output-feedback digital control. Each flight condition has three command modes: positive pitch pointing, vertical translation and straight climb. A sensitivity study is performed to validate the design and illustrate design parameter influences on system response. A computer-aided-design program, MULTI, is developed to assist in the iterative design process. The program is fully interactive, user-oriented, and provides error protection. The program allows complete design and simulation of three types of control law designs: known-regular plants, known irregular plants, and unknown plants. The report contains a brief but complete summary of each of these control law design methods. A user's manual and a programmer's manual are provided for further development of the program.
RPT#: AD-A118117 AFIT/EE/GE/81D-55 81/12/00 83N11143

UTTL: Design of advanced digital flight control systems via Command Generator Tracker (CGT) synthesis methods, volume 1
AUTH: A/FLOYD, R. M. CORP: Air Force Inst. of Tech., Wright-Patterson AFB, Ohio. CSS: (Dept. of Electrical Engineering.)
ABS: This study develops a computer program for interactive execution to aid in the design of Command Generator Tracker control systems employing Proportional-plus-Integral inner-loop controllers and Kalman Filters for state estimation (CGT/PI/KF controllers). Design parameters are specified in the continuous-time domain and the computer program obtains the corresponding discrete-time parameters and determines a direct digital design for sampled-data implementation. Designs are based upon the linear system mode, Quadratic cost, and Gaussian noise process (LOG) assumptions of optimal control theory. The report discusses the theoretical background and applications of optimal model-following designs which preceded the CGT theory. A development of the CGT/PI/KF controller theory is presented, and performance evaluation tools for the controller design are discussed. Following a brief description of the computer program developed, results of applying it to example aircraft-related controller design problems are presented and discussed.

RPT#: AD-A115510 AFIT/GE/EE/81-20-VOL-1 81/12/00
82N31331

UTTL: Computer aided design of digital control systems
AUTH: A/LIPP, H. M. CORP: Karlsruhe Univ. (West Germany). CSS: (Inst. fuer Nachrichtenverarbeitung.) In Kernforschungszentrum Karlsruhe G.m.b.H. INTERKAMA 1980 p 167-174 (SEE N82-20906 11-61)
ABS: Current trends in design techniques of digital control circuits are examined. The use of computer aided designs (CAD) are increasingly more important to satisfy the high demands for design quality and development time. Considerations in the design of digital controls are reviewed. The CAD system LOGE is a program for the synthesis of microprogrammed controllers. It aids the maxi and microcomputers and its function is to support microcomputations. It is found that the LOGE is an effective system for logic design automation. 80/07/25 82N20922

UTTL: PSICON: A simulation package for the design of digitally controlled systems
AUTH: A/LINKENS, D. A.; B/GRAY, L. S.; C/MENAD, M.; D/MORT, N. PAA: D/(Royal Naval Engineering College) CORP: Sheffield Univ. (England). CSS: (Dept. of Control Engineering.)
ABS: A simulation language based on a block oriented continuous systems simulator (PSI) is described. Using the message transfer capability of the multitasking operating system of a minicomputer, external digital control algorithms can be developed rapidly in FORTRAN, while retaining the advantages of a flow chart based and precompiled structure of the continuous simulation language. The use of the package is illustrated by (1) an anesthetics problem where time delay dynamics are controlled, using an external FORTRAN program comprising a Smith predictor plus a PID regulator; and (2) a self tuning adaptive autopilot for ship steering. The package gives fast simulation and good interactive properties, with the external controller segments giving the required flexibility for a wide range of uses.
RPT#: RR-162 81/10/00 82N19090

UTTL: LQG multivariable design tools
AUTH: A/STEIN, G.; B/PRATT, S CORP: Honeywell, Inc., Minneapolis, Minn. CSS: (Systems and Research Center.) In AGARD Multi-Variable Analysis and Design Tech. 16 p (SEE N82-10048 01-08)
ABS: The basic design algorithms needed for frequency domain oriented Linear-Quadratic-Gaussian feedback design are described and experimental interactive computer aided design package through which these algorithms can be effectively acessed is introduced. The algorithms and design package are illustrated with several flight control design examples for highly maneuverable aircraft. 81/09/00 82N10054

UTTL: Direct digital design method for reconfigurable multivariable control laws for the A-7D Digital 2 aircraft
AUTH: A/POTTS, D. W. CORP: Air Force Inst. of Tech., Wright-Patterson AFB, Ohio. CSS: (School of Engineering.)
ABS: This thesis investigates control of an aircraft when there is a primary control surface failure. The object of this study is to reconfigure the remaining control surfaces to compensate for the additional forces and moments generated by the inoperative control surface. To study this flight control problem, a comprehensive aircraft model is required which considers each control surface operating individually. A six degree-of-freedom aircraft model is developed, including all the individual control surfaces. A control surface input can produce both a lateral and/or a longitudinal response. Thus, the equations of motion cannot be decoupled for the design of the control laws. The coupling between the axes requires the derivation of several new non-dimensional control derivatives. Using the geometrical properties of the aircraft and the Digital Datcom computer program, the needed control derivatives are derived. The entire eigenstructure assignment method is used to assign both the eigenvalues and the eigenvectors to the closed-loop plant matrix. This method is used for the direct digital design of a multivariable discrete regulator and tracker control law. The effect of increasing the number of control inputs on the relative degree of controllability of the states was determined by singular value decomposition.
RPT#: AD-A100794 AFIT/GE/EE/80D-36 80/12/00 81N29136

UTTL: The Aerospace Vehicle Interactive Design system
AUTH: A/WILHITE, A. W. CORP: National Aeronautics and Space Administration, Langley Research Center, Hampton, Va.
ABS: The aerospace vehicle interactive design (AVID) is a computer aided design that was developed for the conceptual and preliminary design of aerospace vehicles. The AVID system evolved from the application of several design approaches in an advanced concepts environment in which both mission requirements and vehicle configurations are continually changing. The

basic AVID software facilitates the integration of independent analysis programs into a design system where the programs can be executed individually for analysis or executed in groups for design iterations and parametric studies. Programs integrated into an AVID system for launch vehicle design include geometry, aerodynamics, propulsion, flight performance, mass properties, and economics.
RPT#: NASA-TM-81957 81/02/00 81N20167

UTTL: Computer aided control system design. A conversion report
AUTH: A/BALLE, J. CORP: Christian Rovsing Ltd., Copenhagen (Denmark).
ABS: The conversion of a computer aided control system design package from running on PDP DEC-10 medium size computer to run on an HP 21 MX mini computer is described. In its original and its converted form the package contains about 30,000 FORTRAN statements and some assembler routines. Detailed documentation of how the conversion was performed is presented. Redesigned components such as the file handling and the graphic implementation are described as well as the conversion techniques used and the acceptance test performed.
RPT#: CRI-110 ESA-CR(P)-1351 79/07/00 81N12752

UTTL: Interactive design system for aircraft dynamic control problems
AUTH: A/KUBBAT, W. J.; B/OESTERHELT, G.; C/KORTE, U. CORP: Messerschmitt-Boelkow-Blohm G.m.b.H., Ottobrunn (West Germany). CSS: (Unternehmensbereich Flugzeuge.) Presented at SMP of AGARD Meeting, Neubiberg, West Germany, 3-6 Sep. 1979
ABS: An interactive system for control law design and synthesis is described. Available methods (continuous discrete, time domain-frequency domain) are reviewed and the system is illustrated. Selection of method (i.e., discrete vs. continuous complete vs. incomplete state feedback, optimal control vs. pole-placement etc.) is followed by a dialog designer-computer with immediate results presented in numerical and graphical form (plots, print-outs). Each result is stored and can be compared with any other via dual plots. The system also allows the input of disturbance like white or colored noise, ramps, steps, sine and cosine combinations. There is no practical restriction on the number of state variables. A helicopter control problem is used to demonstrate use of the system.
RPT#: MBB-FE-324/S/PUB/11 79/09/06 80N26329

UTTL: An interactive framework for design automation
AUTH: A/PEHRSON, B. CORP: Uppsala Univ. (Sweden). CSS: Inst. of Technology.)
ABS: The architecture of an integrated interactive system for use in computer aided design of control computer systems and other design engineering applications is presented. The design process is structured into the modeling synthesis, and the implementation phases. A human oriented dialogue system comprised of different design descriptions is emphasized as essential to computerized design. Suitable information structures and programming languages for the implementation of such a system are discussed.
RPT#: UPTEC-78-47-R 78/05/00 80N11821

UTTL: A method of computer-aided generation of software for digital system controllers using fixed-instruction machines
AUTH: A/MANWARING, M. L. CORP: Utah State Univ., Logan.
ABS: A method of applying a general purpose high-level language to the writing of software for digital system controllers which are implemented with fixed-instruction machines is presented. Several hardware configurations for synchronous digital system controllers are presented. Each of these fit the model of a Mealy or Moore sequential machine. An example of a programmable sequential machine applicable to the digital control process is presented. This microprogrammable machine may be converted to a fixed-instruction programmable machine. The concept of state of the machine is discussed. Software (program code) for a fixed-instruction machine is a sequence of states. Because of the limited set of instructions available on a machine, the state sequence is restricted. Designing software for a programmable machine requires a somewhat different philosophy than that of designing hardware-programmed machines. In the hardware programmed machine, the design is made to reflect the sequential nature of the control problem as the designer views it. In the software programmed machine, the sequential nature of the problem is forced to fit state sequence capabilities of the machine. 79/00/00 80N10812

UTTL: Multivariable digital control systems
AUTH: A/PORTER, B. CORP: Salford Univ. (England). CSS: Dept. of Aeronautical and Mechanical Engineering.)
ABS: The fundamental system-theoretic research and the parallel development of design techniques which have led to the production of the comprehensive software package EIGENFORTRAC are outlined. The capability of EIGENFORTRAC in relation to the computer-aided design

of high-performance digital control systems whose functions are simultaneously to reject the unmeasurable disturbances and to track multiple command inputs is described. Numerous references are provided to the system-theoretic research and to the computer algorithms embodied in EIGENFORTRAC.

RPT#: AD-A071662 USAME/DC/101/79 79/06/00 80N10226

REPORT DOCUMENTATION PAGE			
1. Recipient's Reference	2. Originator's Reference AGARD-LS-128	3. Further Reference ISBN 92-835-1455-6	4. Security Classification of Document UNCLASSIFIED
5. Originator	Advisory Group for Aerospace Research and Development North Atlantic Treaty Organisation 7 rue Ancelle, 92200 Neuilly sur Seine, France		
6. Title	COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS		
7. Presented at	on 5–6 September 1983 in Stuttgart, Germany; 8–9 September 1983 in Athens, Greece; and 12–13 September 1983 in Paris, France.		
8. Author(s)/Editor(s) Various			9. Date July 1983
10. Author's/Editor's Address Various			11. Pages 144
12. Distribution Statement	This document is distributed in accordance with AGARD policies and regulations, which are outlined on the Outside Back Covers of all AGARD publications.		

13. Keywords/Descriptors

Guidance	Design
Control equipment	Computer programs
Control theory	Computerized simulation
Digital systems	

14. Abstract

This Lecture Series is intended to provide the basic concepts, theories and computer methods involved in the design of advanced guidance and control systems.

The degree of advantages in the application of modern microprocessor technologies is altready largely affected by the way corresponding systems are designed in the very early stage of a development programme.

It is intended to perform a comprehensive review of direct digital analysis and synthesis procedures and to include in this Lecture Series computer-aided and graphical techniques that can be employed in preliminary design, synthesis and real-time simulation.

The material in this publication was assembled to support a Lecture Series under the sponsorship of the Guidance and Control Panel and the Consultant and Exchange Programme of AGARD.

AGARD Lecture Series No.128
Advisory Group for Aerospace Research and Development, NATO
COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS
Pubiished July 1983
144 pages

This Lecture Series is intended to provide the basic concepts, theories and computer methods involved in the design of advanced guidance and control systems.

The degree of advantages in the application of modern microprocessor technologies is already largely affected by the way corresponding systems are designed in the very early stage of a development programme.

P.T.O.

AGARD-LS-128

Guidance
Control equipment
Control theory
Digital systems
Design
Computer programs
Computerized simulation

AGARD Lecture Series No.128
Advisory Group for Aerospace Research and Development, NATO
COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS
Published July 1983
144 pages

This Lecture Series is intended to provide the basic concepts, theories and computer methods involved in the design of advanced guidance and control systems.

The degree of advantages in the application of modern microprocessor technologies is already largely affected by the way corresponding systems are designed in the very early stage of a development programme.

P.T.O.

AGARD-LS-128

Guidance
Control equipment
Control theory
Digital systems
Design
Computer programs
Computerized simulation

AGARD Lecture Series No.128
Advisory Group for Aerospace Research and Development, NATO
COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS
Published July 1983
144 pages

This Lecture Series is intended to provide the basic concepts, theories and computer methods involved in the design of advanced guidance and control systems.

The degree of advantages in the application of modern microprocessor technologies is already largely affected by the way corresponding systems are designed in the very early stage of a development programme.

P.T.O.

AGARD-LS-128

Guidance
Control equipment
Control theory
Digital systems
Design
Computer programs
Computerized simulation

AGARD Lecture Series No.128
Advisory Group for Aerospace Research and Development, NATO
COMPUTER-AIDED DESIGN AND ANALYSIS OF DIGITAL GUIDANCE AND CONTROL SYSTEMS
Published July 1983
144 pages

This Lecture Series is intended to provide the basic concepts, theories and computer methods involved in the design of advanced guidance and control systems.

The degree of advantages in the application of modern microprocessor technologies is already largely affected by the way corresponding systems are designed in the very early stage of a development programme.

P.T.O.

AGARD-LS-128

Guidance
Control equipment
Control theory
Digital systems
Design
Computer programs
Computerized simulation

It is intended to perform a comprehensive review of direct digital analysis and synthesis procedures and to include in this Lecture Series computer-aided and graphical techniques that can be employed in preliminary design, synthesis and real-time simulation.

The material in this publication was assembled to support a Lecture Series under the sponsorship of the Guidance and Control Panel and the Consultant and Exchange Programme of AGARD presented on 5–6 September 1983 in Stuttgart, Germany; 8–9 September 1983 in Athens, Greece; and 12–13 September 1983 in Paris, France.

ISBN 92-835-1455-6

It is intended to perform a comprehensive review of direct digital analysis and synthesis procedures and to include in this Lecture Series computer-aided and graphical techniques that can be employed in preliminary design, synthesis and real-time simulation.

The material in this publication was assembled to support a Lecture Series under the sponsorship of the Guidance and Control Panel and the Consultant and Exchange Programme of AGARD presented on 5–6 September 1983 in Stuttgart, Germany; 8–9 September 1983 in Athens, Greece; and 12–13 September 1983 in Paris, France.

ISBN 92-835-1455-6

It is intended to perform a comprehensive review of direct digital analysis and synthesis procedures and to include in this Lecture Series computer-aided and graphical techniques that can be employed in preliminary design, synthesis and real-time simulation.

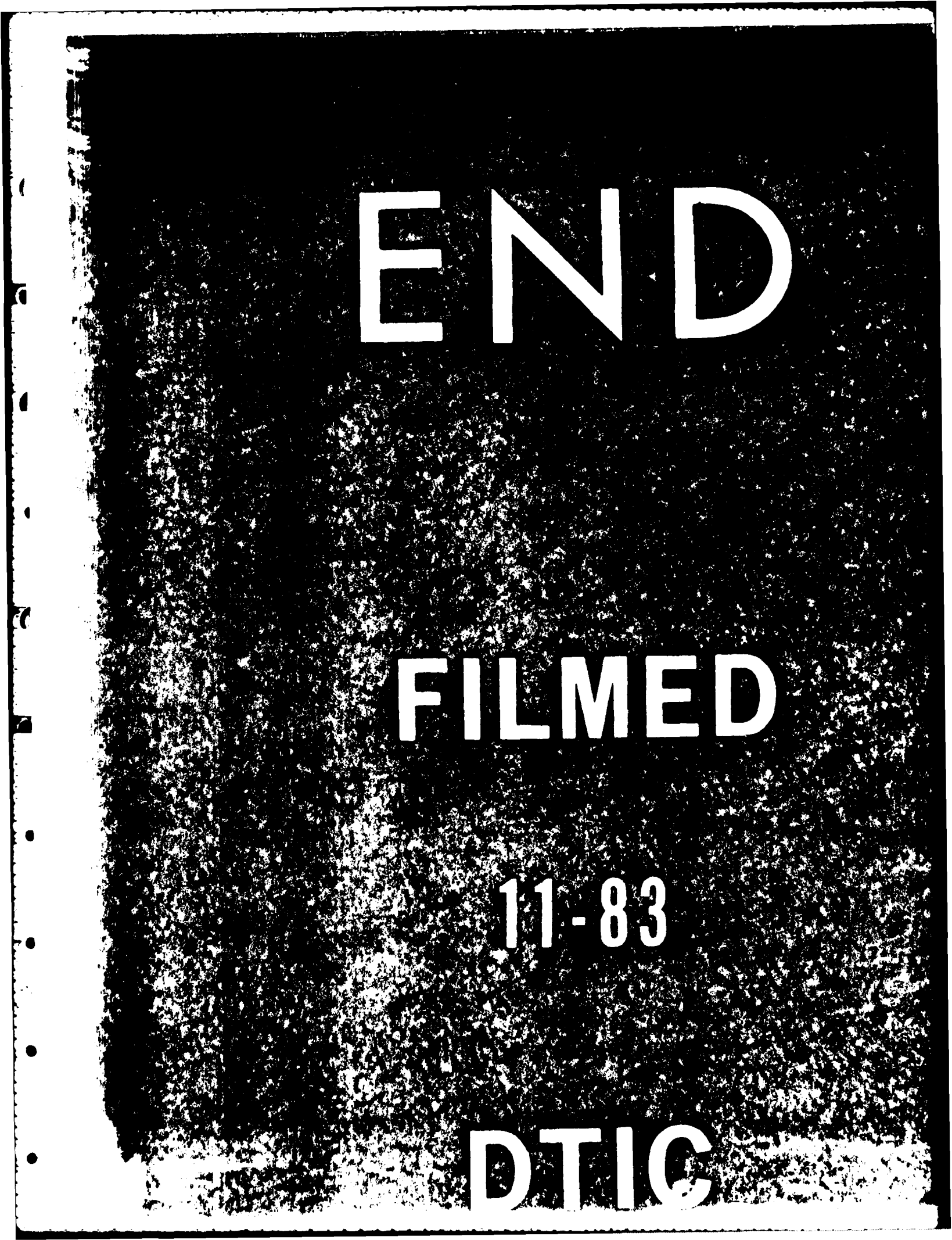
The material in this publication was assembled to support a Lecture Series under the sponsorship of the Guidance and Control Panel and the Consultant and Exchange Programme of AGARD presented on 5–6 September 1983 in Stuttgart, Germany; 8–9 September 1983 in Athens, Greece; and 12–13 September 1983 in Paris, France.

ISBN 92-835-1455-6

It is intended to perform a comprehensive review of direct digital analysis and synthesis procedures and to include in this Lecture Series computer-aided and graphical techniques that can be employed in preliminary design, synthesis and real-time simulation.

The material in this publication was assembled to support a Lecture Series under the sponsorship of the Guidance and Control Panel and the Consultant and Exchange Programme of AGARD presented on 5–6 September 1983 in Stuttgart, Germany; 8–9 September 1983 in Athens, Greece; and 12–13 September 1983 in Paris, France.

ISBN 92-835-1455-6


END
FILMED
11-83
DTIC